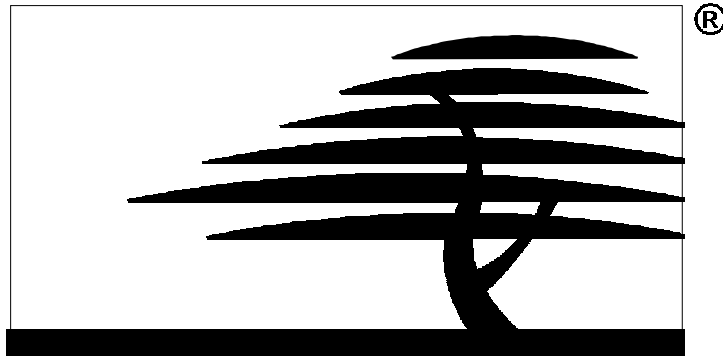




SCO

UnixWare[®] 7 NonStop[®] Clusters[®] Concepts

Student Guide

Course Release: 1.0

For Software: UnixWare[®] 7 NonStop[®] Clus-
ters Version 7.1.1

©2000 The Santa Cruz Operation, Inc. All rights reserved.

This publication is protected under copyright laws and international treaties.

Information in this document is subject to change without notice and does not represent a commitment on the part of The Santa Cruz Operation, Inc.

SCO, The Santa Cruz Operation, the SCO logos, VP/ix, SCO OpenServer, UnixWare, dbXtra, Panner, SCO Global Access, SCO MPX, SCO Doctor, SCO Doctor for Networks, SCO Doctor Lite, SCO Premier Motif, SCO Visual Tcl, SCO VisionFS, SCO ToolWare, SCO TermLite, SCO Vision 97, SCO CIFS Bridge, SCO WebTop, The Internet Way of Computing, and IWoC are trademarks or registered trademarks of The Santa Cruz Operation, Inc. in the USA and other countries. IXI, X.desktop, Deskworks, Wintif, IXI Panorama, Deskterm, and IXI Desktop are trademarks or registered trademarks of IXI Limited, a subsidiary of The Santa Cruz Operation, Inc. Visionware, Devkit.Vision, Esprit, Kodon, Super.Vision, Visionware Super.Vision, Term.Vision, Visionware Direction, the Visionware logo, Visionware SQL-Retriever, and XVision are registered trademarks of Visionware Limited, a subsidiary of The Santa Cruz Operation, Inc. SuperVision, TermVision, X-Visionware, SQL-Retriever, and Vision Builder are trademarks of Visionware Limited. UNIX is a registered trademark of The Open Group in the United States and other countries. Cheyenne and ARCserve are registered trademarks of Cheyenne Software, Inc. Netscape, Netscape Navigator, Netscape Communications Server, Netscape Commerce Server, Netscape Proxy Server, Netscape FastTrack Server, Netscape Enterprise Server, Netscape SuiteSpot, Netscape Catalog Server, Netscape News Server, Netscape Mail Server, and Netscape Navigator Gold are trademarks of Netscape Communications Corporation. NFS was developed by Legent Corporation (formerly Lachman Associates, Inc.) based on LACHMAN SYSTEM V NFS. LACHMAN is a trademark of Legent Corporation. NFS is a trademark of Sun Microsystems, Inc. TCP/IP was developed by Legent Corporation (formerly Lachman Associates, Inc.) based on LACHMAN SYSTEM V STREAMS TCP, a joint development of Lachman Associates and Convergent Technologies. MPX was developed by Corollary, Inc. VP/ix is a product developed and licensed by Phoenix Technologies, Ltd/INTERACTIVE Systems Corporation. XRemote is a registered trademark of Network Computing Devices, Inc. Oracle is a registered trademark of Oracle Corporation, Redwood City, California. Java is a trademark of Sun Microsystems, Inc. Compaq, Tandem, Integrity, NonStop, NetFlex, ProLiant, ROMPaq, ServerNet, SmartStart, registered in United States Patent and Trademark Office. Intel and Pentium are registered trademarks, and Pentium II Xeon and Pentium III Xeon are trademarks of Intel Corporation. SCO, Veritas, VxFs, and VxVA are registered trademarks of Veritas Software Corporation. Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies. All other brand and product names are or may be trademarks of, and are used to identify products or services of, their respective owners.

The SCO software that accompanies this publication is commercial computer software and, together with any related documentation, is subject to the restrictions on US Government use as set forth below. If this procurement is for a DOD agency, the following DFAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the US Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

If this procurement is for a civilian government agency, this FAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: This computer software is submitted with restricted rights under Government Contract No. _____ (and Subcontract No. _____, if appropriate). It may not be used, reproduced, or disclosed by the Government except as provided in paragraph (g)(3)(i) of FAR Clause 52.227-14 alt III or as otherwise expressly stated in the contract. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

Table of Contents

Lesson 1: Course Introduction	1-1
Introduction.....	1-3
Course goal and purpose.....	1-3
Course Lessons	1-5
Target audience	1-5
Course roadmap	1-5
<i>NSC Curriculum Hardware Supplement</i>	1-6
Student Materials	1-7
The <i>Student Guide</i>	1-7
Typographical Conventions	1-8
Lesson 2: Driving Forces and Clusters	2-1
The Demand for More Computing Resources.....	2-7
Faster and Less Expensive Hardware	2-9
More Reliable Computing Environments	2-11
The Monolithic Solutions	2-15
The NSC Solution	2-17
Lesson Summary.....	2-19
Answers to Exercise and Review Questions.....	2-21
Lesson 3: Key Cluster Concepts	3-1
Defining Clusters	3-7
Clustering definition and single system image (SSI).....	3-7
Features of SSI Clustering	3-9
Scalability	3-11
Manageability	3-13
<i>Nodes</i>	3-15
Server area network (SAN) and fabrics.....	3-17
Protected resources and components	3-18
Failover, fault zones, N+1, N+0.....	3-24
Load balancing.....	3-25

NSC Specific Terms and Components.....	3-27
SSI in NSC.....	3-27
Scalability and Availability in NSC.....	3-29
Manageability in NSC	3-31
Nodes in NSC	3-33
SAN options for NSC and the Virtual Interface Architecture (VIA).....	3-37
Protected storage methods in NSC	3-41
Load balancing in NSC.....	3-41
Physical and Virtual IP Network Addresses	3-43
Virtual UPS	3-45
Clusterizing.....	3-46
Lesson Summary.....	3-47
 Lesson 4: Cluster Comparisons	 4-1
Distributed Computing Solutions	4-6
IP-Spraying Solutions	4-7
High Availability “Clustering” Solutions	4-8
True Clustering Solutions	4-9
Comparison Summary	4-11
Lesson Summary.....	4-12
 Lesson 5: Designing an NSC Cluster	 5-1
Design Concepts	5-6
Vulnerability and single point of failure	5-6
Node design	5-8
Node format—tower or rack mount.....	5-12
Disk configuration	5-14
Electrical power and UPS	5-16
Case Studies.....	5-17
Small business or replicated site	5-17
Large storage, low processing.....	5-18
Low storage, high processing	5-19
Heavy processing, heavy I/O	5-20
Lesson Summary.....	5-21
Answers to Exercise and Review Questions.....	5-22

1

Course Introduction



Lesson 1

Course Introduction

1.1 Introduction

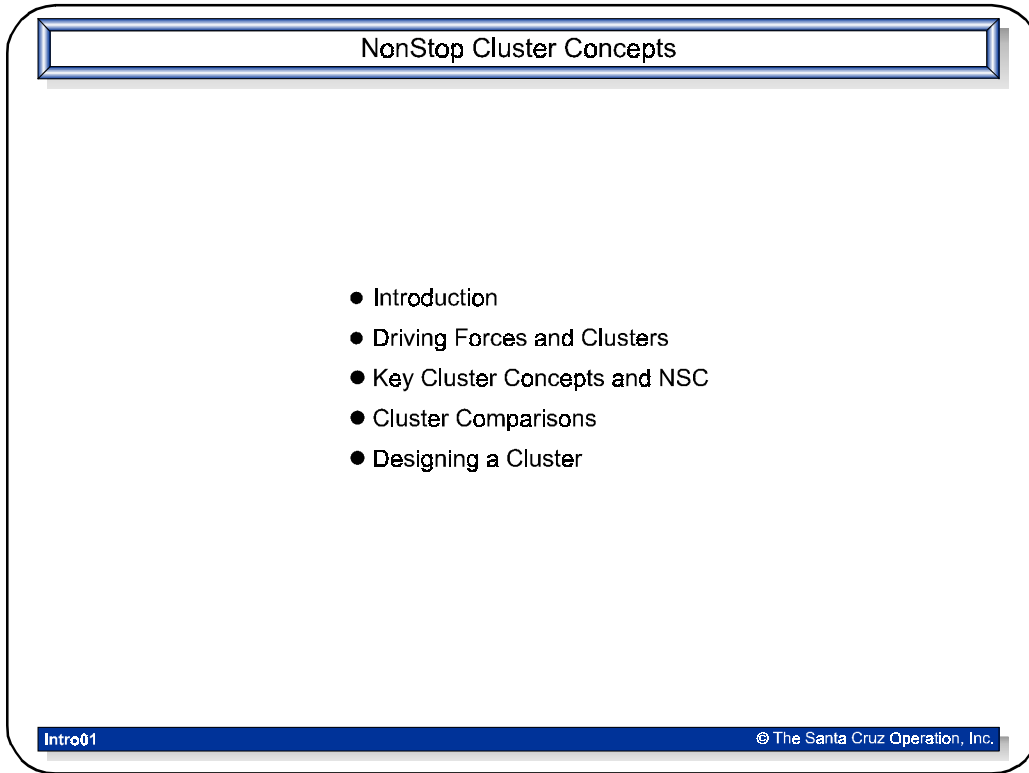
This **UnixWare 7 NonStop Clustering Concepts** course is designed for individuals who require a conceptual background on generic clustering and UnixWare 7 NonStop Clusters (NSC) features, components, and concepts. The target audience includes sales professionals who will sell NSC, technicians who will install and configure NSC, system administrators who will maintain NSC systems on a daily basis, and programmers who will develop cluster-aware applications.

1.1.1 Course goal and purpose

At the conclusion of this course, students should be able to:

- Explain the driving forces behind clusters
- Explain key clustering concepts and terms including *single point of failure*, *failover*, *nodes*, *node pairs*, *root nodes*, *dependent nodes*, *load balancing*, *protected storage*, *server area network*, and *single system image*
- Explain NSC features and capabilities
- List, compare, and contrast the various methods of protecting storage in NSC
- Differentiate NSC from other non-clustering solutions
- Differentiate NSC from other clustering solutions
- Design a cluster for small, medium, and large clients with various needs for failover, load balancing, and protected storage

AV 1-1

A presentation slide titled "NonStop Cluster Concepts" with a blue header bar. The slide contains a bulleted list of five topics: Introduction, Driving Forces and Clusters, Key Cluster Concepts and NSC, Cluster Comparisons, and Designing a Cluster. At the bottom, there is a blue footer bar with the text "Intro01" on the left and "© The Santa Cruz Operation, Inc." on the right.

NonStop Cluster Concepts

- Introduction
- Driving Forces and Clusters
- Key Cluster Concepts and NSC
- Cluster Comparisons
- Designing a Cluster

Intro01 © The Santa Cruz Operation, Inc.

1.1.2 Course Lessons

To achieve the course goals, this course contains the following lessons:

- Introduction (this lesson) — course goals, topics, lengths, and typographical conventions
- Driving Forces and Clustering — how market forces are driving demand for greater computing resources and how clustering answers those demands
- Key Cluster Concepts, NSC, and Cluster Components — cluster-specific terms, concepts, and components
- Cluster Comparisons — contrasting the SCO NonStop Cluster (NSC) solution with other “cluster” solutions
- Designing a Cluster — designing a cluster to meet specific criteria, including case studies

1.1.3 Target audience

This class is aimed at sales people, installation technicians, system administrators, support technicians, and developers who need to become familiar with clusters in general and specifically UnixWare 7 NonStop Clusters.

A general, high-level background in computers is a necessary prerequisite for this course. However, prior knowledge of a UNIX or any SCO operating system is not required.

1.1.4 Course roadmap

UnixWare 7 NonStop Clustering Concepts is the first course in a series on UnixWare 7 NonStop Clusters. The other courses are:

- ***UnixWare® 7 NonStop Clusters® Installation***
- ***UnixWare® 7 NonStop Clusters® Administration***

1.1.5 NSC Curriculum Hardware Supplement

The NSC product is constantly evolving. New hardware system vendors are announcing support for NSC, new server area network, and protected storage options are becoming available regularly.

The **NSC Curriculum Hardware Supplement** contains information. The supplement has one chapter per certified vendor. Each chapter contains the same relative information. For example, each chapter contains the current list of computers the vendor supports for use as nodes in a cluster.

1.2 Student Materials

This student guide is yours to keep and write notes in. Your instructor might have other reference materials in the classroom for your use while attending this class. Please do not take or write in these materials as they will be needed again for later classes.

1.2.1 The *Student Guide*

The *Student Guide* contains:

- Outline of presentation
- Note-taking space
- Copies of all transparencies

1.3 Typographical Conventions

This publication presents commands, filenames, keystrokes, and other special elements as shown here:

lp or lp(1)	Commands, device drivers, programs, and utilities (names, icons, or windows); the number in parentheses indicates the reference manual section in which the command, driver, program, or utility is documented
<i>/new/client.list</i>	Files, directories, desktops (names, icons, or windows), system, network, and printer names
root	User names
<i>arg</i> or <i>arg</i>	Placeholders (replace with appropriate name or value)
[<i>arg</i>]	Surrounds an <i>option</i> or <i>cmdarg</i> that is not mandatory
<i>arg ...</i>	Indicates multiple occurrences of the <i>option</i> or <i>cmdarg</i>
\$	Command-line prompt for commands executable from the Bourne shell by any user
#	Command-line prompt for commands only executable by the root user from the root account
F2	Keyboard keys
CTRL SPACE	Press CTRL and SPACE keys simultaneously
CTRL ALT DEL	Press all three keys, CTRL , ALT , and DEL , simultaneously
CTRL d	Press CTRL key. With CTRL still depressed, press D .
ALT 0240	Press ALT key. With ALT still depressed, press and release in sequence 0 , 2 , 4 , and 0 .
CTRL Q + i	Press CTRL key. With CTRL still depressed, press Q . Release both keys and press I .
CTRL Q + I	Press CTRL key. With CTRL still depressed, press Q . Release both keys and press SHIFT I .
Exit program?	System output (prompts, messages)

tar tv	User input, such as on a command line
“Description”	Field names or column headings (on screen or in database)
Cancel	Button names
Edit	Menu names
Copy	Menu items
File ⇨ Find ⇨ Text	Sequences of menus and menu items
open or open(2)	Library routines, system calls, kernel functions, C keywords; the number in parentheses indicates the reference manual section in which the file is documented
\$HOME	Environment or shell variables
SIGHUP	Named constants, signals, and kernel parameters
“adm3a”	Data values
<i>employees</i>	Database names and tables
buf	C program structures
b_b.errno	Structure members



2

Driving Forces and Clusters



Lesson 2

Driving Forces and Clusters

Introduction

Current market forces are calling for faster, less-expensive, and more reliable solutions to increasing computing needs. This is true in almost every market segment, from small to medium businesses (SMBs), to the Fortune 100 (enterprise) companies.

This lesson examines these market forces, the solutions being put forth to address them, and how NonStop Clusters fits as a solution.

Objectives

By the end of this lesson, you will be able to:

- Describe the market forces that lead to the development and adoption of clusters
- Define clustering terms, such as *cluster*, *monolithic solution*, *SMP*, *NUMA*, and *ccNUMA*

AV 2-1

The slide is titled "Driving Forces and Clusters" and contains a bulleted list of five points. The footer includes the text "Driving01" and "© The Santa Cruz Operation, Inc.".

Driving Forces and Clusters

- The Need for More Computing Resources
- Faster and Less Expensive Hardware
- More Reliable Computing Environments
- The Monolithic Solutions
- The Cluster Solution

Driving01 © The Santa Cruz Operation, Inc.

Topics

In order to meet these objectives, this lesson discusses:

- The Need for More Computing Resources — modern business dynamics that are creating the demand for greater resources, such as CPU, RAM, disk drive space, and disk drive access
- Faster and Less Expensive Hardware — how improvements in hardware technologies influence the market
- More Reliable Computing Environments — increasing demand for reliability in computing environments
- The Monolithic Solutions — extensions to the traditional computing solution
- The Cluster Solution — how clustering, and specifically NSC clustering, improves on monolithic solutions

AV 2-2

Demand for More Computing Resources

- Sophisticated user interfaces
- Improved business management features
- More Reliable Computing Environments
- Fast response times
- 24X7 Availability

Driving02 © The Santa Cruz Operation, Inc.

2.1 The Demand for More Computing Resources

One force driving the need for clusters is the increased demand for computing resources. This need comes from many places in many forms.

- Software requires more processing power to support sophisticated user-friendly interfaces. This is especially true of graphics software, which demands more resources than character applications. Such modern software requires more processor power, more RAM, and more disk space than its previous counterparts.
- Managers expect more business management features from their software. This also requires more resources. For example, managers expect applications to communicate with each other, reducing the requirement to re-enter the same data in multiple applications.
- People using computers as part of their every day job expect fast response from their computer hardware and software. Researchers have found a sharp decrease in the time that users expect to wait for a computer response.
 - Slow response times increase labor costs and inefficiency in all aspects of a business.
 - Fast responses require more resources.
- The explosion in popularity of the Internet means vastly increased demand on web servers. Simultaneously, users expectations are rising. If a response is not *fast enough*, they click to another web site. Therefore, web servers must have enough resources to handle increasing loads, for both browsing and e-business.

Further, web servers are becoming an increasingly critical component of many businesses. E-commerce demands service around the world, around the clock. Failure of a web server at any time is a critical problem that can cause major losses.

AV 2-3

Faster and Less Expensive Hardware

- Rapid Increases in CPU Speeds
- Lower Computer Equipment Costs

Driving03 © The Santa Cruz Operation, Inc.

2.2 Faster and Less Expensive Hardware

Due to many factors, computers become faster and less expensive every day. This phenomenon, almost unparalleled in the industrial revolution, provides inexpensive commodity computer hardware in large quantities. *Moore's Law* states that CPU speeds double every 18 months. This has many implications:

- Since some companies consider purchases under \$2000 to be supplies, most desktops, and even some servers are supplies. This means that computers are now considered disposable within many organizations.
- Organizations can purchase high-performance computers at little cost. This creates increased demand which feeds economies of scale. This further reduces the cost of computers. Alternatively, you may say that more performance is available for the same amount of money, especially when adjusting for inflation.
- In the first quarter of 2000, CPUs reached speeds of 1000 MHz, also known as one Giga Hertz (1GHz). Some time in 2001, processors will have speeds of 2GHz.
- Availability of relatively cheap and very fast computers provides a fertile development environment for groups, or *clusters*, of PCs to work together to provide improved computational ability and higher reliability than can be achieved by single computers.

NSC takes advantage of the wide availability of faster and cheaper hardware to build systems using *off-the-shelf* components that have scalability, availability and performance features formerly limited to high-end computing platforms.

AV 2-4

Downtime has a Price

Examples of Average Cost of 1 Hour Downtime

- Financial Market - Brokerage \$6.45M
- Credit Card Sales Authorization 2.60M
- Media Pay-Per-View Server 150K
- Retail Home Catalog Sales 90K

Internal SCO Research

DrivIng04 © The Santa Cruz Operation, Inc.

2.3 More Reliable Computing Environments

Many businesses are satisfied with the performance of their computer system. However, they would like one that is available non-stop—24 hours a day, 7 days a week. The need for this level of availability has several sources and implications:

- As business rely more heavily on data and voice communications, data mining, databases, and decision support systems, the computers that run these services must be up and running at all times.
- With society depending on computers for everything from traffic signals to military defense, computer down time becomes more than an inconvenience. Highly-available systems are required to turn computer failures from emergencies into maintenance problems.
- Businesses and individuals are increasingly reliant on their computer systems. As people become more dependent on computerized systems, they become mission critical to the organizations using them. Consider this example:

The failure of a computer can create havoc for a resort management company that has a weekly turnover of hundreds of families in hundreds of beach cottages. The critical time for these companies is the weekend when the check-outs and check-ins occur. Often, computer maintenance agreements are not 7x24x365, they are Monday through Friday from 8 to 5. Downtime on Saturday morning is a disaster for these types of businesses.

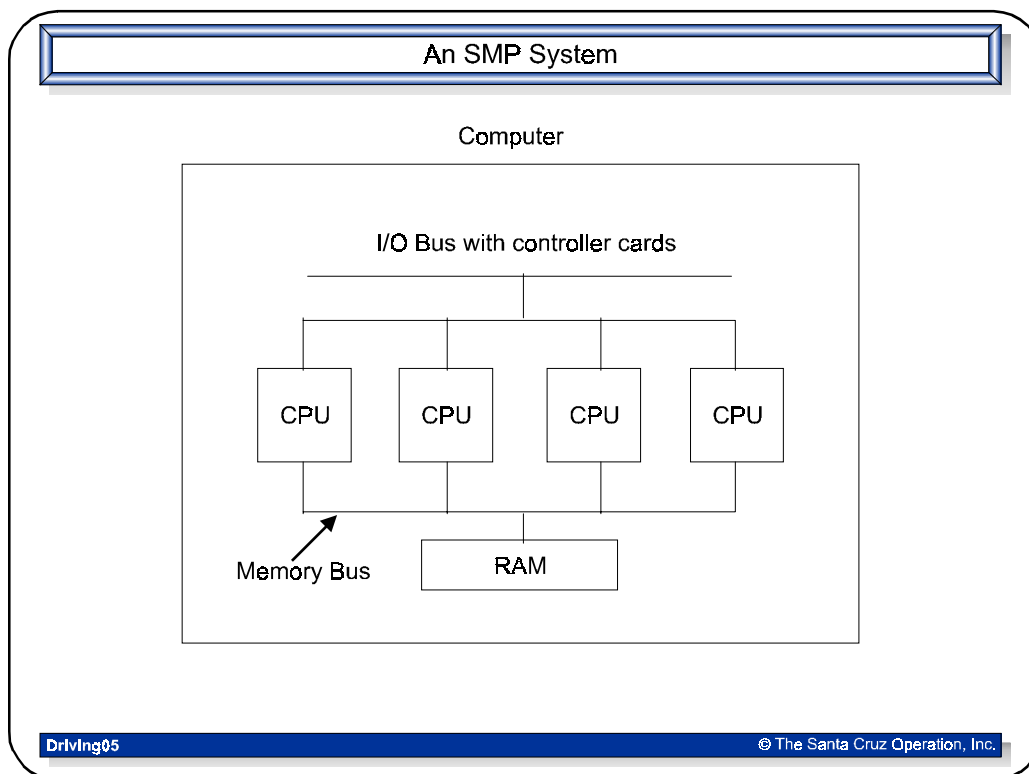
- The failure of a web server for an e-business means no business. For the customer seeking a web site, the failure of the web server leads to loss of service, dissatisfaction and possibly loss of business to a competitor who is only a click or two away.
- The failure of systems handling voice and data communications can cause great losses to the users of the systems as well as financial damage to the service provider



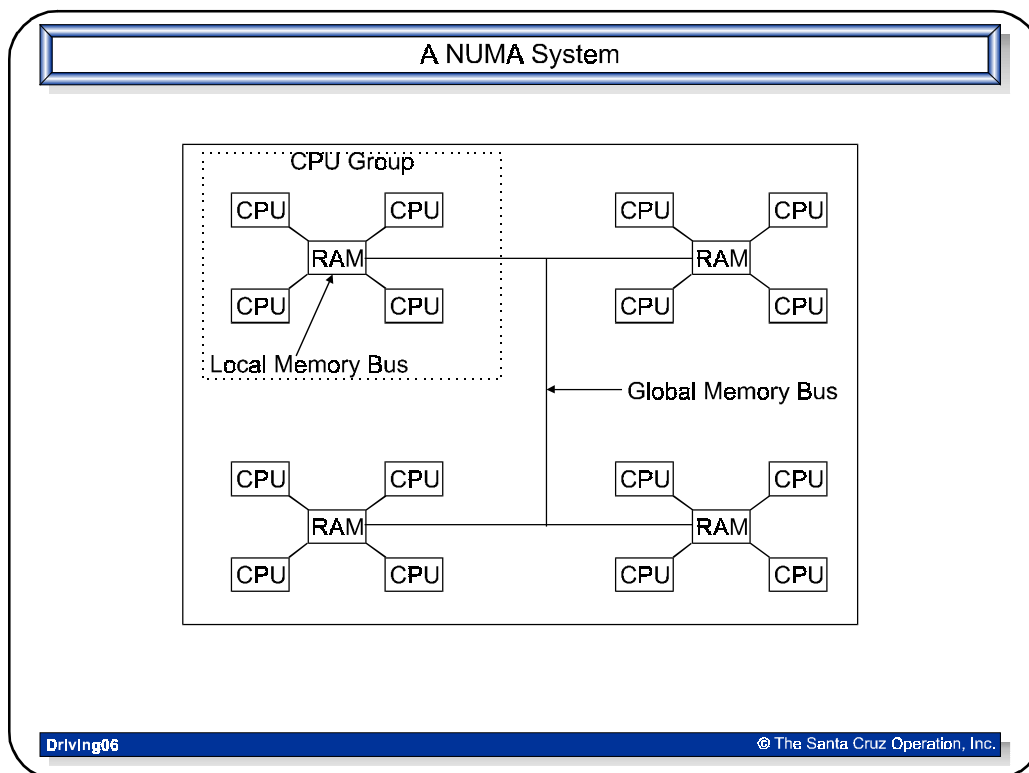
- In large businesses, some locations have no one with sufficient expertise to repair a inoperable server. For these locations, the reliability of the system is crucial. This is especially true where chain organizations, such as fast food stores or hotels, use computers in replicated sites. When was the last time you visited your favorite chain fast-food store and noticed a computer system administrator hanging around waiting for the system to crash?
- Downtime can have a substantial cost. AV 2-4 shows some sample loss figures for downtime.

NSC provides more reliable computing environments for all of these scenarios. It does so in ways that are discussed throughout the rest of this course.

AV 2-5



AV 2-6



2.4 The Monolithic Solutions

A *monolithic* solution for computing is one that employs a single computer which is typically large, complex, and proprietary. Such solutions:

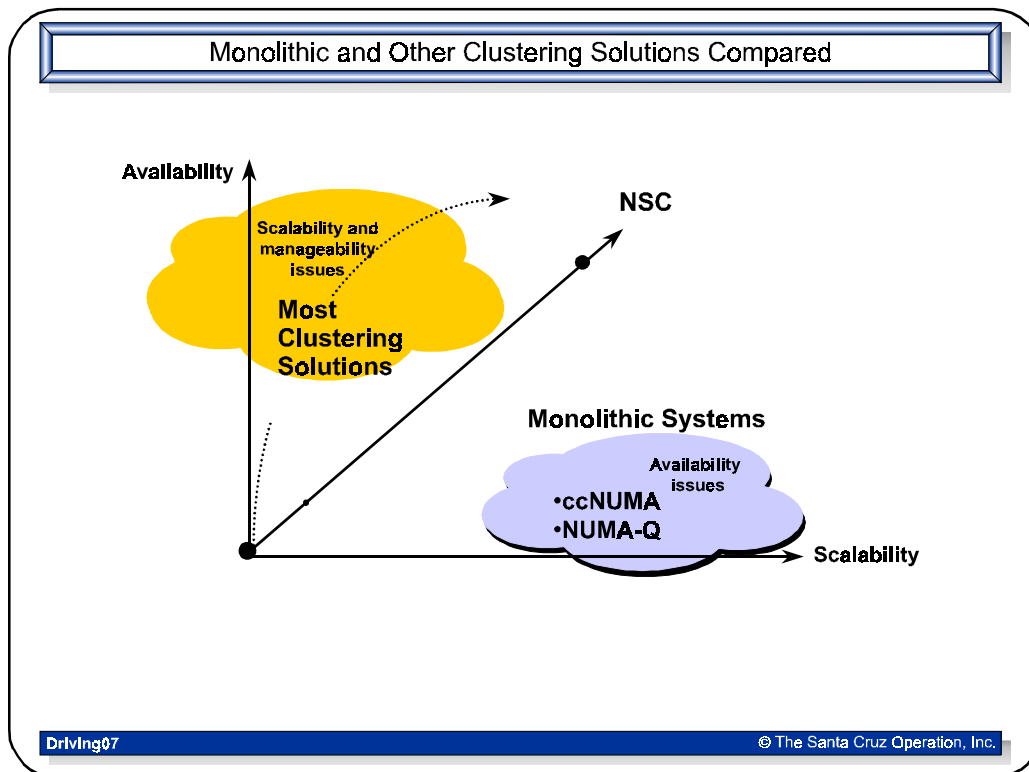
- Have been the traditional method of providing increased performance
- Always include adding some number of CPUs to the computer. The number of CPUs added can range from one on the low end, to thousands, in a *massively-parallel multiprocessor* solution.

While these solutions provide increased performance, the cost of the specialized hardware associated with this solution is high.

Some of the methods for building monolithic systems include SMP, NUMA, and ccNUMA:

- *Symmetric Multi Processing (SMP)* systems consist of multiple CPUs within a single computer. Multithreaded operating systems and applications spread the workload somewhat evenly over the CPUs to achieve higher throughput.
 - As the number of CPUs increases, the costs of computers that can accommodate the increase in CPUs grows at a non-linear rate. This is largely due to the expensive, high-speed interconnects needed to insure cache coherency between CPUs.
 - Due to bus bandwidth limitations and other hardware factors, there is a point at which scalability drops off as the number of CPUs increases.
 - Applications that are not multithreaded achieve no improvement in throughput when run on an SMP system.
- *Non Uniform Memory Access (NUMA)* systems use a number of small SMP modules, each of which has a few CPUs and some local RAM. The cost for any CPU within an SMP module to access their local RAM is the same. However, the cost for a CPU from one SMP module to access RAM in a different SMP module is greater than the local access. These costs are non-uniform, giving this solution its name.
- *ccNUMA* is a variation of NUMA. With ccNUMA, the CPU and other caches maintain coherency at all times.

AV 2-7



2.5 The NSC Solution

Clusters provide every aspect of high-performance, highly-available, reliable computing currently demanded by the market. With respect to:

Price	NSC runs on fast, low-cost, readily-available hardware. Incremental upgrades to an NSC cluster are possible with relatively low cost. NSC hardware components can be easily obtained and added to address computer emergencies quickly. The proprietary monolithic solutions are based on hardware that is low volume and therefore, not low cost or readily available.
Availability	Clusters use multiple computers, called <i>nodes</i>, to provide redundancy. If one node goes down, the cluster is still available to be used. Even between nodes, many resources are also provided in a redundant fashion, such as filesystem storage. Further, NSC provides the ability to quickly upgrade and expand a cluster. As computing resources run low, additional processors, additional nodes, and additional disk space can be added to a node <i>without shutting down the cluster</i>. This allows organizations the ability to easily expand the cluster without down time. Monolithic solutions may not offer all the online upgrade options of NSC.
Reliability	NSC uses the highly-tested and proven UnixWare 7 operating system as its foundation. This, combined with other well-tested clustering components, creates a system with a very degree of reliability. An estimated 99.99995% uptime rate can be achieved, translating to only 26 minutes of downtime per year, typically for upgrades and reboots. NSC provides a method to upgrade the operating system while the cluster is online. This means operating system upgrades become a matter of rebooting individual nodes while the cluster continues to function. Monolithic systems, even with redundant components, still have single points of failure, and lack a failover solution.

Scalability

Scalability refers to the ability of a system to expand and maintain a level of performance as it grows. Scalability is measured as the loss of potential performance as the system grows. Currently, NSC can scale to over 30 nodes with over 300 processors.

Future enhancements will exceed these limits. For extreme scalability, clusters are a better solution than monolithic systems.

Manageability

Manageability refers to the ease of administration offered by a system. The NSC design allows an administrator to manage the cluster as a single system and use familiar UNIX commands to manage the system. The skills needed to manage a single UNIX system are directly applicable to managing an NSC cluster.

Demand for NSC clusters is currently coming in two forms:

- Businesses with small systems that want greater reliability and availability. This includes many small business and large replicated sites.
- Those who need the increased scalability but are price sensitive.

2.6 Lesson Summary

This lesson described the forces driving cluster, specifically UnixWare 7 NonStop Clusters, to the forefront in computing environments. The points covered include:

- Faster and less expensive hardware
- The need for more computing resources
- More reliable computing environments
- Monolithic solutions
- The NSC solution

Review questions

1. List three market forces that are leading to the adoption of clustering.
2. What solution is available for high performance that does not provide clustering?
3. What is the difference between a NUMA and a SMP system?



2.7 Answers to Exercise and Review Questions

This section contains the answers to the review questions on page 2-19.

Review Question Answers

This section contains the answers to the review questions on page 2-19.

1. The market forces leading to the adoption of clustering were discussed throughout the course and include:
 - The need for higher availability
 - The need for more reliability
 - The need for more computing power including scalability beyond SMP systems.
2. Per Section 2.4, “The Monolithic Solutions,” on page 2-15, a solution that offers higher availability but does not provide the other features of clustering is SMP and NUMA systems.
3. The difference between a NUMA and an SMP system is that SMP systems use CPU-groups, which have faster access to local RAM and slower access to global RAM, whereas NUMA systems have all CPUs accessing the same RAM.



3

Key Cluster Concepts



Lesson 3

Key Cluster Concepts

Introduction

This lesson introduces you to the key concepts of both generic and UnixWare[®] 7 NonStop[®] Clusters, and the terms used to describe these concepts.

Objectives

By the end of this lesson, you will be able to:

- Explain key concepts of clusters, including *single point of failure*, *1+1* and *N+1* failover configurations, *N+0* and *N+1* load balancing configuration, *protected storage*, and *server area network (SAN)*
- Explain key concepts of NSC clusters including *single system image (SSI)*, *virtual UPSes*, *node pairs*, *active* and *potential root nodes*
- Define the components used in an NSC cluster

AV 3-1

The slide is titled "Key Cluster Concepts and NSC" in a blue header bar. Below the title, there is a bulleted list of two items: "Defining Clusters" and "NSC Specific Terms and Components". At the bottom of the slide, there is a blue footer bar containing the text "Key01" on the left and "© The Santa Cruz Operation, Inc." on the right.

Key Cluster Concepts and NSC

- Defining Clusters
- NSC Specific Terms and Components

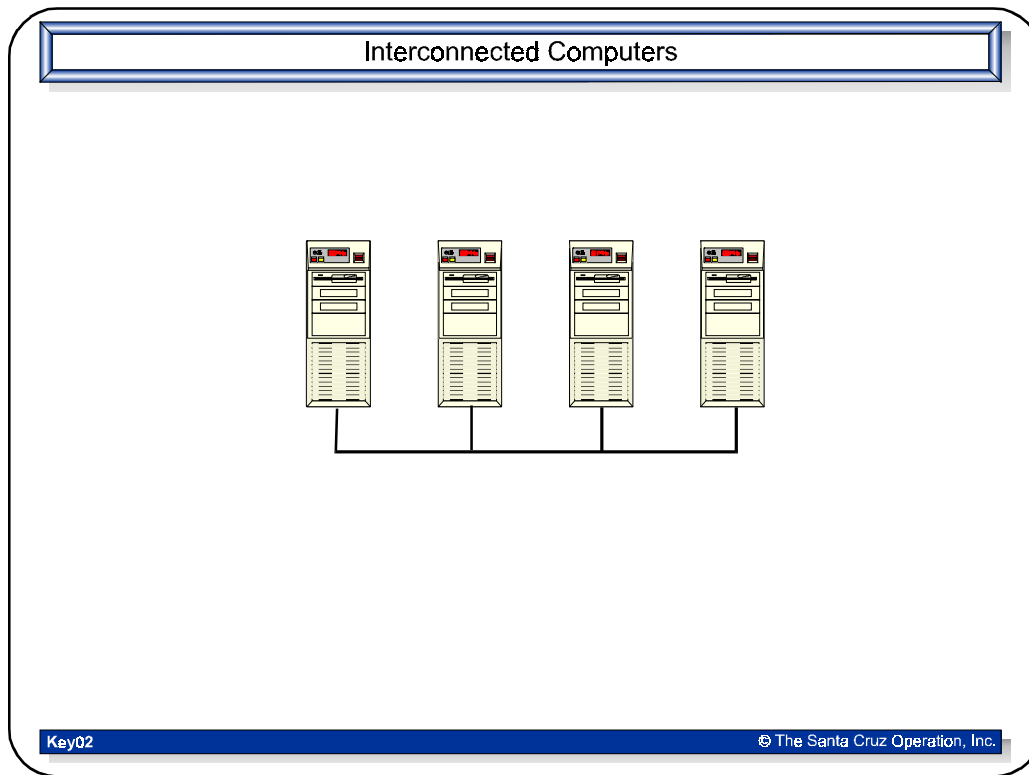
Key01 © The Santa Cruz Operation, Inc.

Topics

In order to achieve these objectives, this lesson discusses:

- Defining Clusters — defining terms and concepts used with clusters of all types
- NSC-specific Concepts and Components — defining terms and concepts specific to NSC. It also lists and describes the various components used in an NSC cluster.

AV 3-2



3.1 Defining Clusters

The term cluster, when associated with computers, has been used by various people to mean different things. However, all of the definitions have many things in common. This section describes the terms commonly used to describe clusters.

3.1.1 Clustering definition and single system image (SSI)

One of the best definitions of a cluster is found in the book *In Search of Clusters* by Gregory Pfister:

“A cluster is a type of parallel or distributed system that consists of a collection of two or more interconnected whole computers and is used as a single, unified computing resource.”

The key difference between a cluster and a collection of independent computers on a local area network (LAN), is that a cluster:

...is used as a single, unified computing resource.

AV 3-2 on page 3-6 illustrates a number of distinct, interconnected computers. They are either:

- A number of independent computers on a LAN
- A cluster—if they appear to users and administrators as a single computing resource.

In order for two or more computers to appear as a single computing resource, they must exhibit a property or behavior known as *single system image (SSI)*. For example, a UNIX cluster system that exhibits SSI:

- Shows a single process listing for all processes running on all computers, regardless of the computer on which the process listing command runs
- Shows a single set of devices, such as, printers, tapes, and disks, available cluster-wide
- Uses a single file tree, containing all filesystems on all disks attached to any computer in the cluster

A major advantage of SSI is that administrative effort of a cluster does not increase substantially as nodes are added. Management overhead for new nodes only adds incrementally to the administrative load.

Many clustering products offer some components of SSI. Others do not support SSI at all. NSC offers a full set of SSI capabilities. It is this full SSI capability that sets NSC apart from other clustering solutions.

3.1.2 Features of SSI Clustering

Clustering was conceived to address these issues:

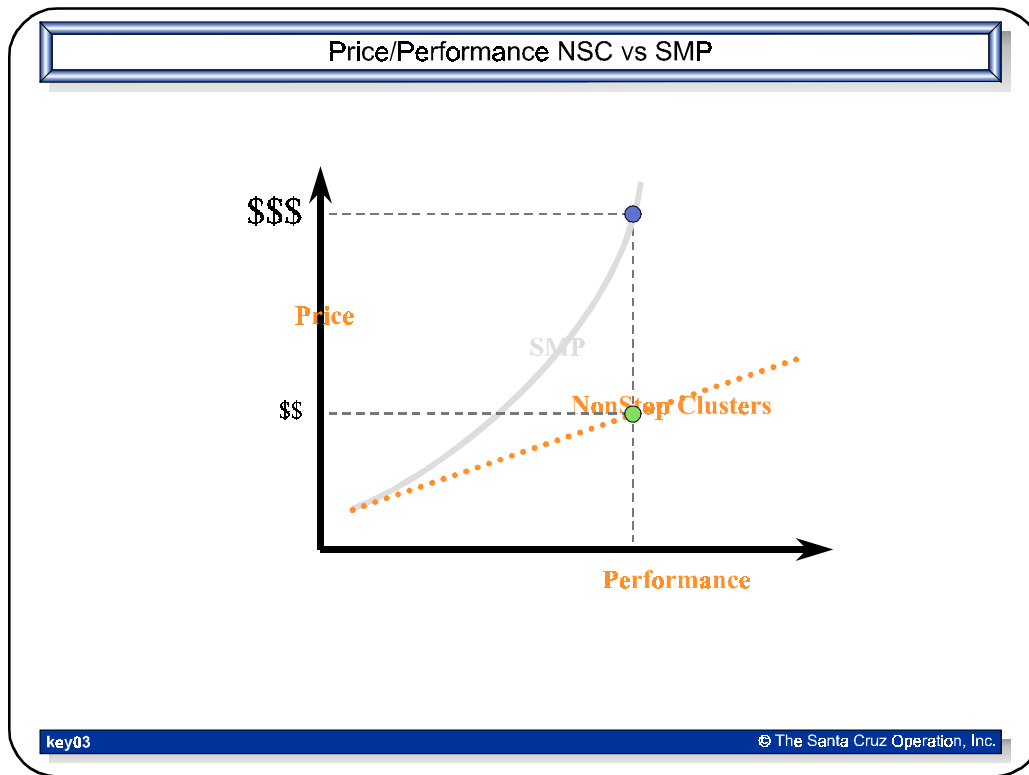
Reliability and Availability If one component, or even an entire node, fails, the failure does not effect the rest of the nodes in the cluster. Rather, they continue to operate and remain available to the users of the cluster. Specific information about how availability is increased in a cluster are presented throughout this lesson.

The key to ensuring availability is redundancy. If every component has a redundant counterpart that can be used in the event of failure, then there is no *single point of failure* within the cluster.

Scalability and Performance

By adding computers to the cluster, the total processing power of the cluster increases.

AV 3-3



3.1.3 Scalability

Scalability is a factor in many areas of a cluster, including:

- Number of nodes — How does the aggregate performance of all nodes in a cluster increase as nodes are added to a cluster?
- Number of CPUs in a node — How does performance increase in a node or throughout the cluster as CPUs are added to a node?
- Number of controllers — How does performance increase in a node or throughout a cluster as more devices (for example multiport serial controllers or SCSI HBAs) are added to a node?
- Number of devices attached to a controller — How does performance increase or decrease on a controller, node, or throughout a cluster, as more devices are attached to a controller? For example, how is performance affected as more disks are attached to a SCSI HBA?

Scalability is reported as a percentage of performance maintained at a certain system size compared to the performance yield at a minimal size.

The most common reason for performance degradation as systems grow is the increased overhead in keeping track of the additional items.

Even if a cluster is highly-scalable, it must provide services to applications and administrators that can take advantage of scalability in a manageable way. For example:

- It becomes difficult to manage multiple nodes in a cluster if each node has its own user database that must be synchronized with every other node. This concept is discussed further in Section 3.1.4, “Manageability,” on page 3-13.
- If applications must be re-written to use special APIs, then *off the shelf* applications will not scale. In NSC, the entire UNIX paradigm is preserved. NSC offers, but does not require, the use of any special APIs to enable applications to benefit from increased scaling performance from the cluster.

AV 3-4

Features of Manageability

- Cluster-wide System Administration
- Transparent Cluster-wide Processes List, Migration (For Load Balancing)
- Applications, utilities, programmers, users and administrators need not be cluster aware.
- Use of standard single node tools

key04 © The Santa Cruz Operation, Inc.

3.1.4 Manageability

Manageability of a cluster refers to the ease of administering the system. Without *clusterized* management tools, the cluster is just a collection of computers that must be separately managed.

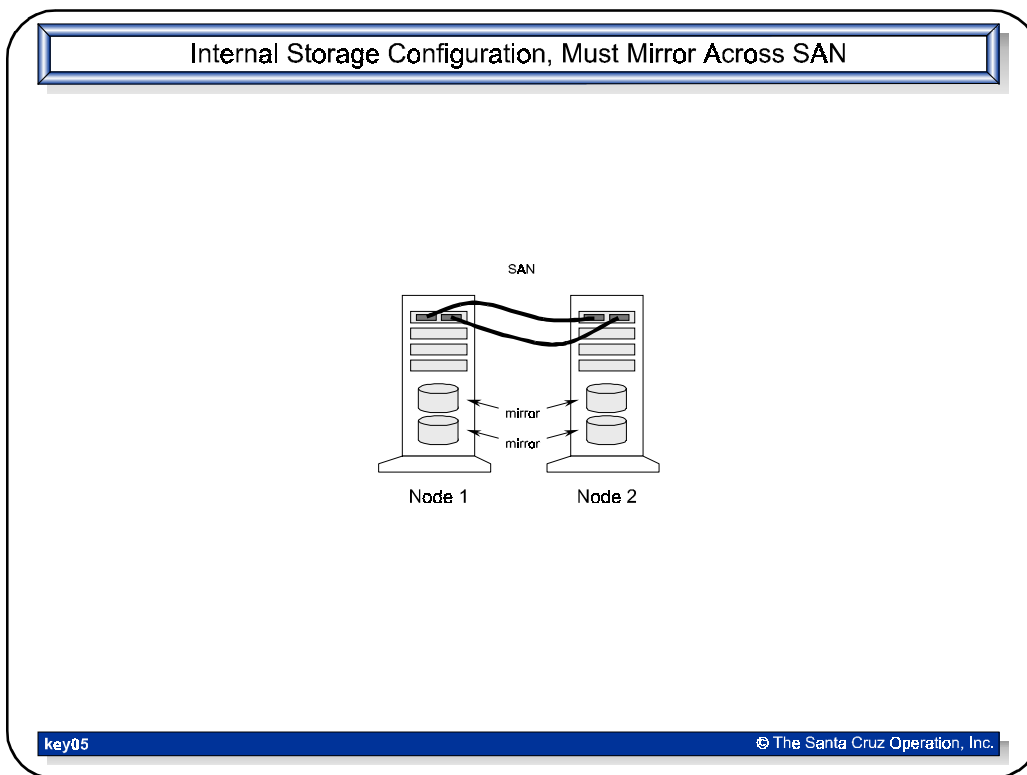
In a true cluster, you use the same tools to perform an operation on the cluster as you would on a single host-based system. For example:

- If you use **useradd**(1M) to add users to a single host, in a true cluster you should be able to use **useradd** to add users to the cluster.
- If you use **ifconfig**(1Mtcp) to configure network interfaces in a single host, in a true cluster you should be able to use **ifconfig** to configure an interface in the cluster.
- If you use **ps**(1) to look at the list of processes running on a single host, in a true cluster you should be able to use **ps** to look at a list of processes running on the entire cluster.
- If processes can migrate between processors within a single host, in a true cluster the processes should be able to migrate to any processor within the cluster.

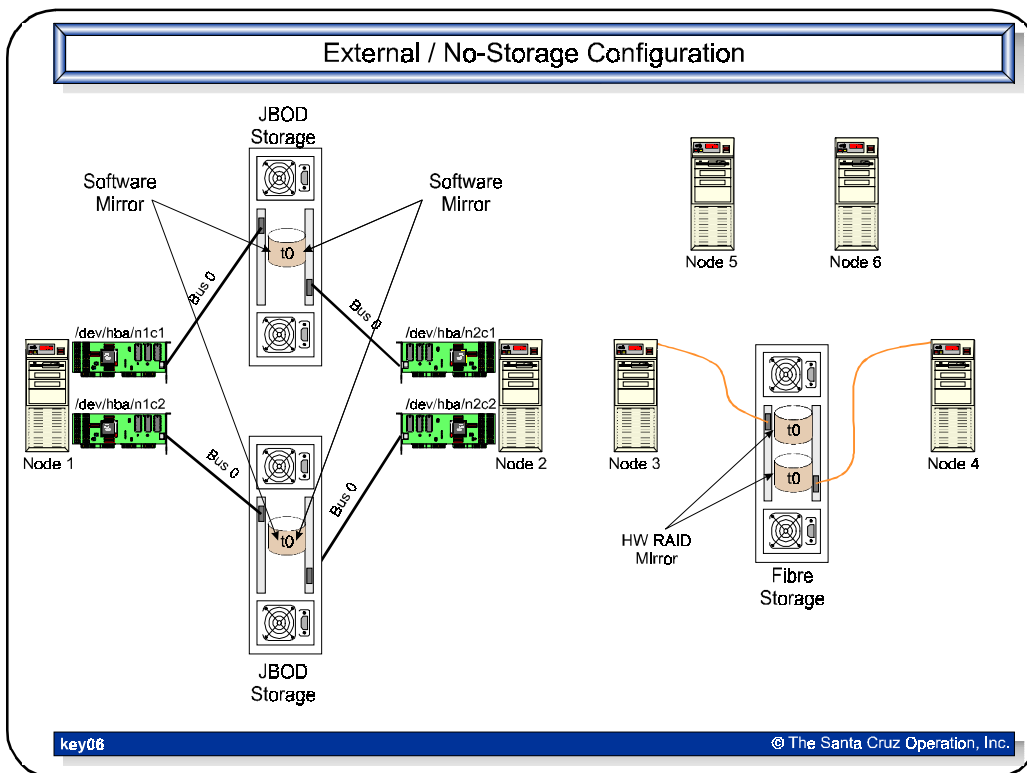
Further, any of the tasks listed above should be able to be performed from any node within the cluster. For example, you should not have to be on node 1 to configure a network interface on node 1.

Section 3.2.3, “Manageability in NSC,” on page 3-31 provides additional information on how NSC improves manageability of a cluster.

AV 3-5



AV 3-6



3.1.5 Nodes

The definition of a cluster said that the cluster contains:

“... two or more interconnected whole computers ...”

In clusters, each whole computer is called a *node*. As a whole computer, each node contains:

- One or more CPUs

Some clustering implementations allow nodes to have multiple CPUs running a highly-scalable operating system. Nodes with multiple CPUs running such operating systems are called *symmetric multi-processor (SMP)* nodes.

- RAM to hold executing programs and data
- At least one video display, keyboard, and mouse interface

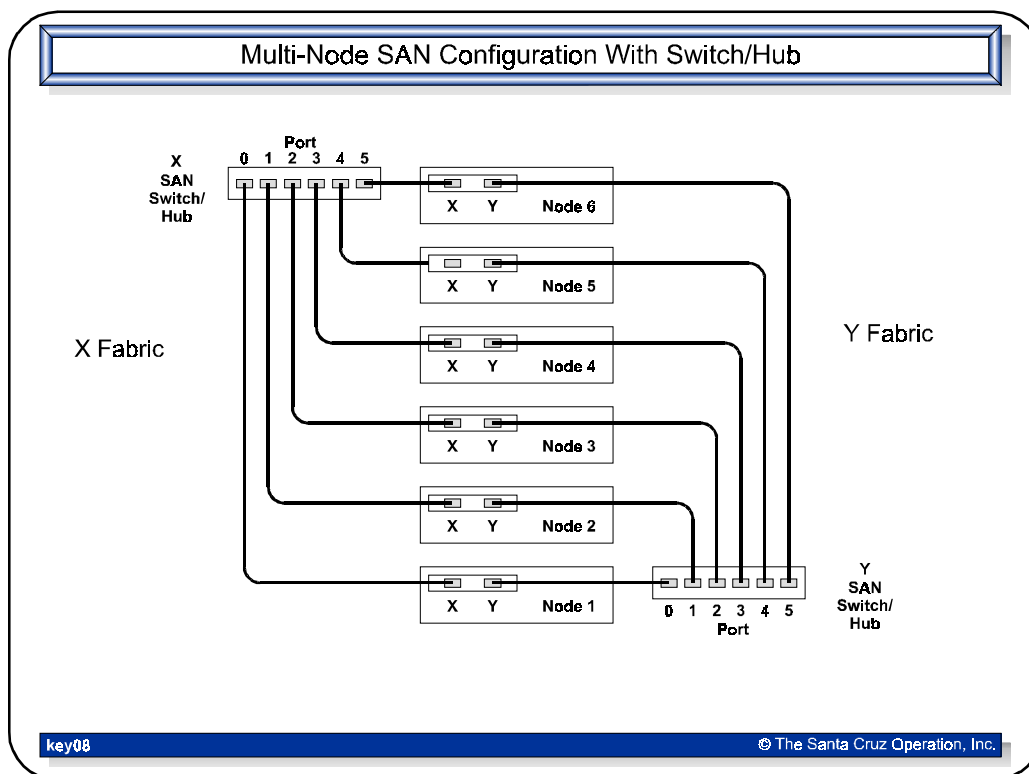
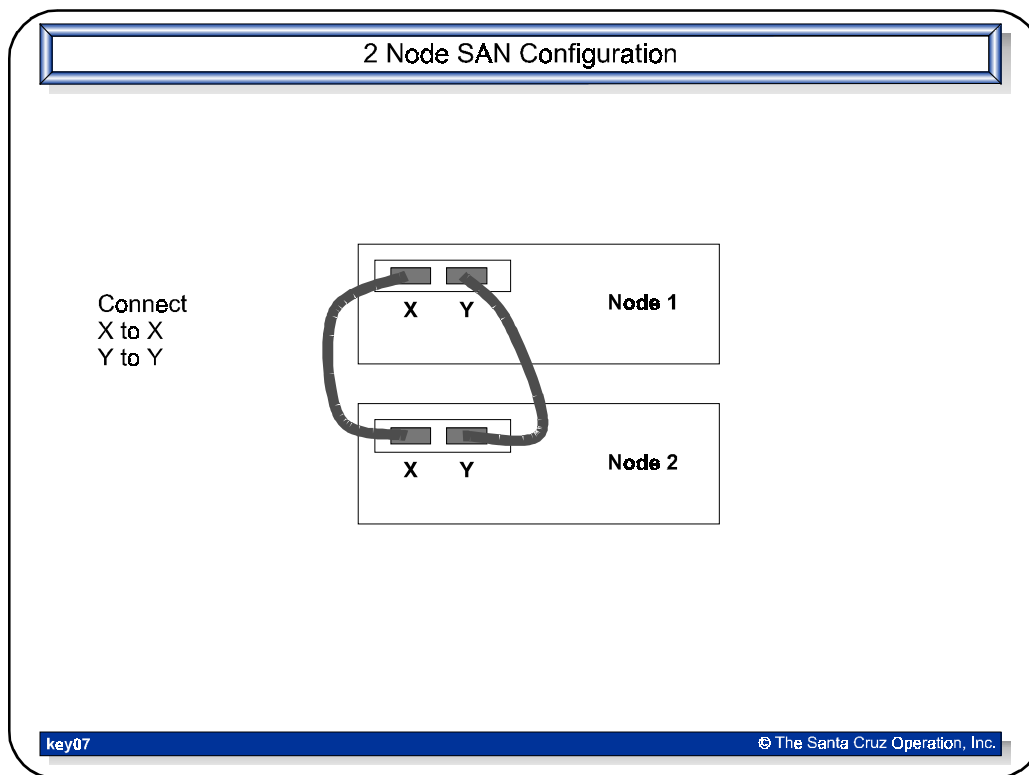
Nodes may also contain other components, such as:

- Networking Interface Cards (NICs), to communicate with the network
- Storage controllers and related devices, such as disks, tapes, CD-ROMs, and floppy drives. Some storage devices are:
 - Internal to a node, such as shown in AV 3-5 on page 3-14. For reasons discussed later, internal disks must be mirrored across the Server Area Network (SAN).
 - External to a node and may be attached to several nodes. An example of storage attached to nodes 1 and 2 is shown in AV 3-6 on page 3-14.
 - Non-existent on certain nodes, such as node three shown in AV 3-6 on page 3-14. However, if a cluster provides SSI, then such nodes can still access the storage devices attached to the other nodes in the cluster.

At least two nodes are required to insure no single point of failure. Hardware supported by NSC is provided on the SCO web site at URL:

<http://www.sco.com/products/clustering/uw7nschw.html>

AV 3-7



3.1.6 Server area network (SAN) and fabrics

The definition of a cluster stated that the nodes must be *interconnected*. In other words, the nodes of a cluster are interconnected through their own network. This network:

- Is a high-speed network
- Is called a *server area network (SAN)*
- Is often different from the networks which carry TCP/IP, SMB, or IPX/SPX traffic on an organization's LAN or WAN. This isolates node-to-node traffic from other client/server traffic.
- May employ proprietary hardware and protocols, which are optimized for node-to-node communications. The faster and more efficient the hardware and protocols, the more efficient the node-to-node communications.
- Can use various architectures, each of which must insure that no single point of failure causes the SAN to become unavailable. At a minimum, this means that the SAN must have:
 - Two sets of wires connecting each node. These wires may plug into the same or separate SAN cards within a node. AV 3-7 on page 3-16 illustrates a simple two-node cluster using a SAN topology that has only two cables.
 - Two switches or hubs, if such technology is required within a SAN. AV 3-7 on page 3-16 illustrates a SAN topology that uses switches or hubs.

Each separate set of SAN wires and hubs, if present, is called a *fabric*. Each of the slides have two labeled fabrics. For consistency with the NSC-specific discussion of fabrics later in this course, the fabrics are labeled X and Y.

The specific SAN topologies and hardware used in NSC are discussed in Section 3.2.5, "SAN options for NSC and the Virtual Interface Architecture (VIA)," on page 3-37.

3.1.7 Protected resources and components

All resources and components must be redundant in order to avoid a single point of failure. So far, redundancy has been discussed for nodes and SAN components. However, redundancy must be extended to:

Shared memory	If shared memory is not redundant, then failure of RAM that is shared will cause processes on other nodes to fail
Semaphores	If semaphores are not redundant, then failure of a node can result in processes losing contact with their source of information.
Filesystem connectivity	If filesystems connections are not redundant, then failure of a node results in the loss of the filesystem. Filesystems that are controlled and accessed by multiple nodes are considered <i>protected storage</i>. This is discussed in greater detail in Section 3.1.7.1, “Protected storage,” on page 3-18.

3.1.7.1 Protected storage

Whenever information stored on a disk is accessible from multiple nodes it is considered *protected*. Storage must be protected to avoid single point of failure. For example, when the node with control of the *root* filesystem fails, a *potential root* node will assume control of the *root* filesystem, thereby protecting the filesystem from loss of contact with the cluster.

Almost every method used to protect storage involves *redundant arrays of inexpensive disks (RAID)*.

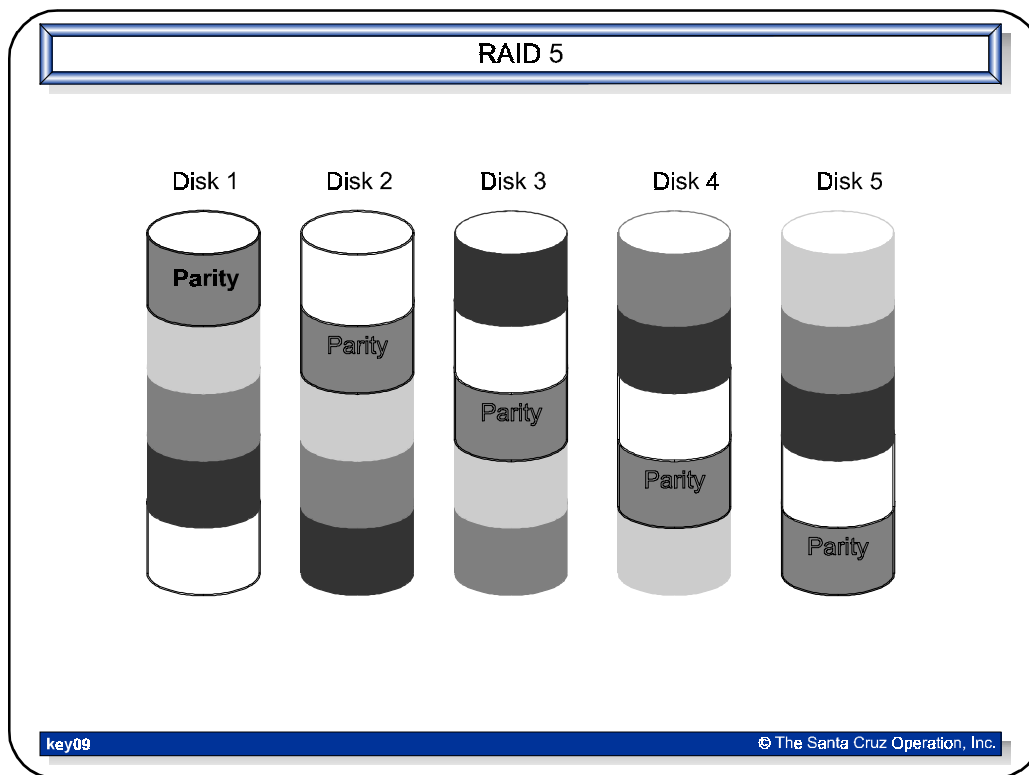
The methods used to protect storage in NSC are discussed in Section 3.2.6, “Protected storage methods in NSC,” on page 3-41.

3.1.7.2 Redundant arrays of inexpensive disks (RAID)

RAID (Redundant Arrays of Inexpensive Disks) provide redundancy of data for recovery in the event one disk in the array fails. Originally five RAID levels were defined. RAID levels 1 and 5 have become standards. Other variations have been developed.

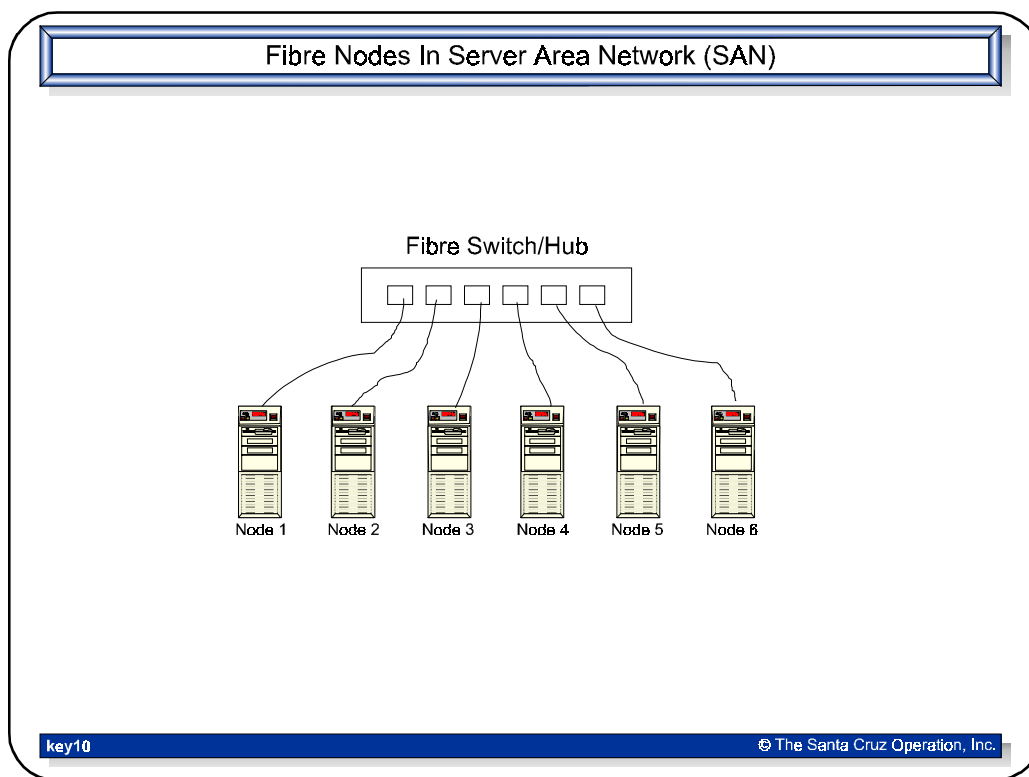
- RAID 0 stripes data across the disks in the array. It
 - Provides no data redundancy in the event of disk failure. If one disk fails, the entire array fails, and is therefore a single point of failure.
 - Provides improved read and write response times because the work is spread across several disks
 - Provides maximum use of available disk space since there is no redundancy as in RAID 1 and RAID 5 discussed below
- RAID 1 is also known as *mirroring*. It:
 - Uses disks in pairs
 - Writes all data to the primary disk and duplicates or mirrors it to the secondary disk
 - Allows the second drive of the mirrored pair to serve the data to ensure there is no single point of failure
 - Is the most expensive form of RAID because 50 percent of the disk space is consumed by mirroring

AV 3-8

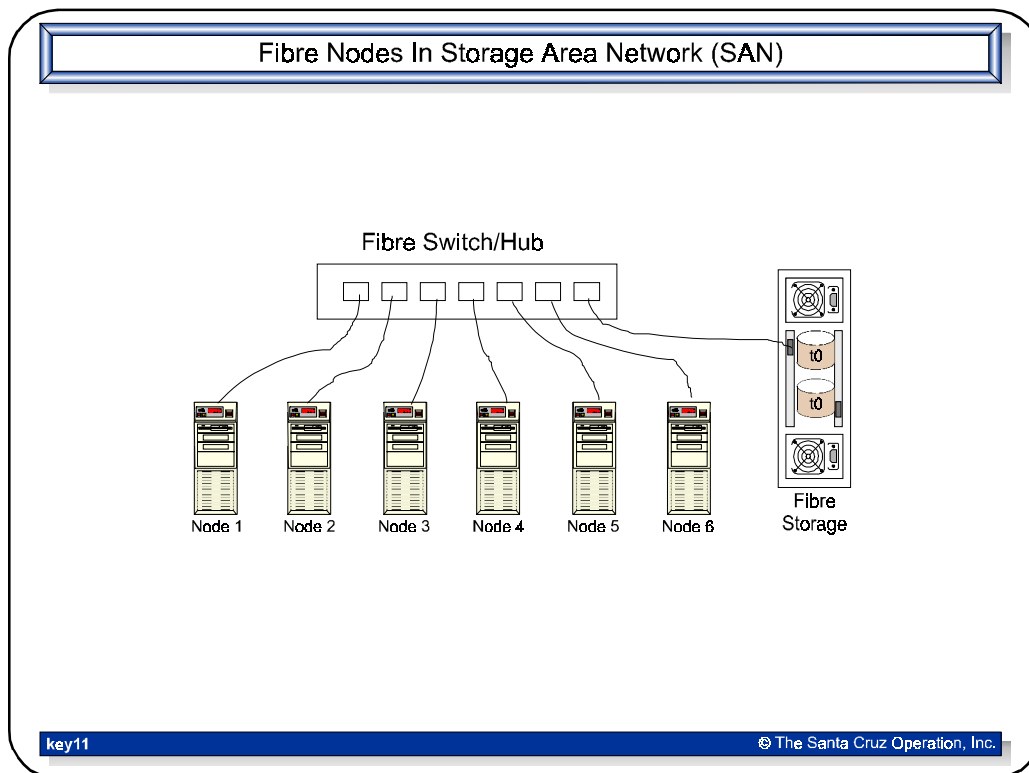


- RAID 5 *stripes* data across the disks with one segment of the stripe reserved for parity information. It:
 - Uses a minimum of 3 disks
 - Rotates parity segments across disks to provide even access to all disks. Thus, each parity stripe must be updated with each data write.
 - Does not have a single point of failure because loss of a single disk does not cause loss of data
 - Reconstructs lost data using parity information in conjunction with the remaining data
 - Provides good read performance because data can be read simultaneously from many disks simultaneously.
 - Provides only fair write performance because both the data segment and the parity segment must be updated on each data update. RAID 5 write performance is between 1/3 and 3/5 of the RAID 1 write performance according to DPT.
- Combinations of RAID
 - RAID 0+1
 - Multiple pairs can be striped together to appear as a single larger array
 - This configuration is sometimes referred to RAID 0+1 or RAID-10
 - RAID 0+5
 - Multiple RAID-5 groups can be striped together to appear as a single larger array
 - This configuration is sometimes referred to RAID 0+5 or RAID-50

AV 3-9



AV 3-10



3.1.7.3 Fibre Channel Arbitrated Loop (FCAL) and GBICS

Fibre Channel Arbitrated Loop (FCAL) is the term for fiber-optic based communications. In clusters, it can be used to connect:

- Nodes on a Server Area Network (SAN), illustrated in AV 3-9 on page 3-22. Note that some vendors use the acronym SAN to also refer to a storage area network.
- Nodes and storage devices on a storage array, illustrated in AV 3-10 on page 3-22.
- The devices that convert metal signals on controller cards inside the computer to fibre optic signals are called GigaBit Interface Converters (GBIC)s.

FCAL solutions support speeds of 100Mbps to 1Gbps. Currently there are two lengths supported:

Short wave	Supports lengths up to 500 meters. By fibre-optic industry standards, GBICs for short wave communications are colored black.
Long wave	Supports lengths up to 2 kilometers. By fibre-optic industry standards, GBICs for long wave communications are colored blue.

FCAL lengths are important for planning the location of nodes and storage. For example, with long wave FCAL, two nodes in a cluster could be 2 kilometers apart, allowing for catastrophe recovery.

3.1.8 Failover, fault zones, N+1, N+0

Whenever a protected resource fails, whether it is a node, SAN fabric, storage, or some other resource, the redundant component is pressed into service. This is called *failover*.

Failover has these important considerations:

- If the redundant component also fails while the original component is not operational, a catastrophic failure can occur, causing the entire cluster to fail and become unavailable.
- Each independent area where a single failure can occur without bringing down the entire cluster is called a *fault zone*.

For example, the X fabric SAN cable on node 1 could fail at the same time as a hard disk in the protected storage. As long as node 1's Y fabric SAN cable remains functional, and the mirror of the bad hard disk remains functional, the cluster remains available.

In this example, the SAN fabrics, and the disks serving as the protected storage are separate fault zones.

Again, to prevent a single point of failure, each fault zone must contain redundant components. Other areas of redundancy include the nodes, disk controllers for the external disk storage, UPSes, and network cards.

- Failover can be configured for software as well as hardware. If a node running a database fails, the software can be restarted on another node.
- Failover can be set up in two primary configurations, where N refers to the number of nodes:

N+0	All components are in use all the time. Loss of a single component will reduce overall capability, but not cause the cluster to become unavailable. An example of N+0 is using 2 nodes (2+0) to run data entry operations. If a single node fails, the other node will carry all of the work load.
N+1	A standby component, not in service, (the component represented by the 1) is available to take over the role of any one or more of the same components (the components represented by the N). An example is using 2 nodes (1+1) to run a database. If one node fails, the other node will start up the database applications.

3.1.9 Load balancing

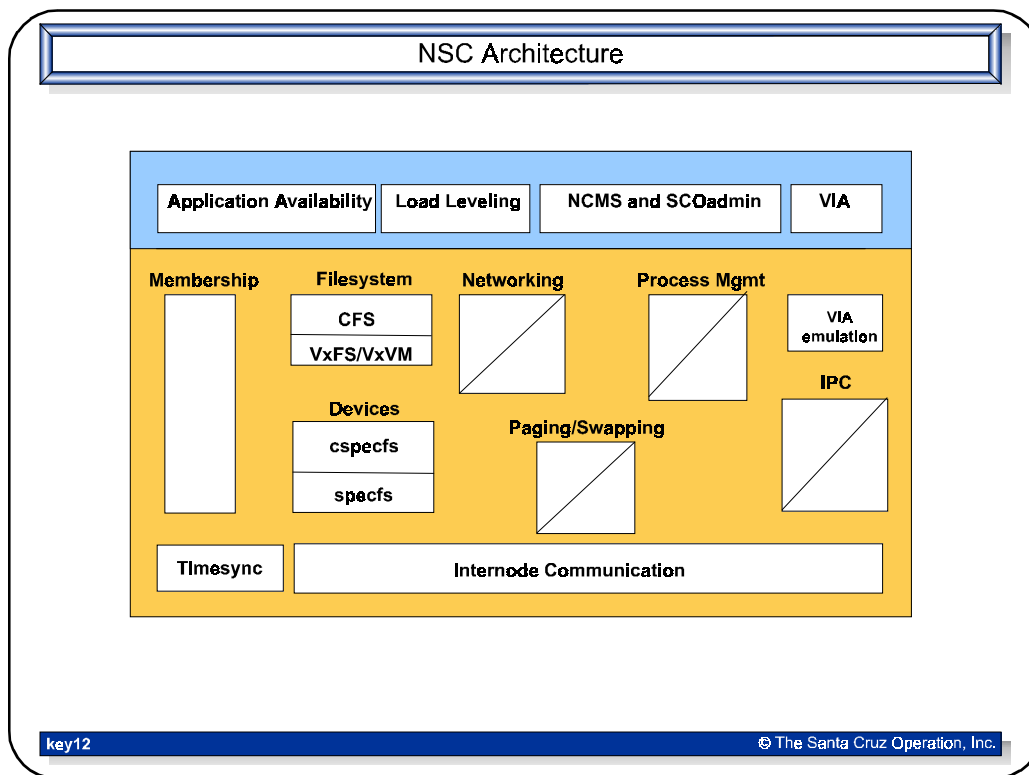
Load balancing is the movement of running processes between nodes so that the work load on each node is approximately equal. It is useful:

- When multiple nodes are running unrelated processes. This includes any heterogeneous mixture of jobs such as accounting, order entry, page layout, or file and print services.
- When the copies of the primary process can run on any one of several nodes. A good example is running the Apache web server on multiple nodes.
- To get the best performance out of the cluster as a whole

Many clustering solutions support only failover and do not provide load balancing. Another way to say this is that many clustering solutions provide only 1+1 failover. Examples of such solutions include Microsoft Wolfpack and Reliant HA.

The NSC load balancing is further described in Section 3.2.7, “Load balancing in NSC,” on page 3-41.

AV 3-11



3.2 NSC Specific Terms and Components

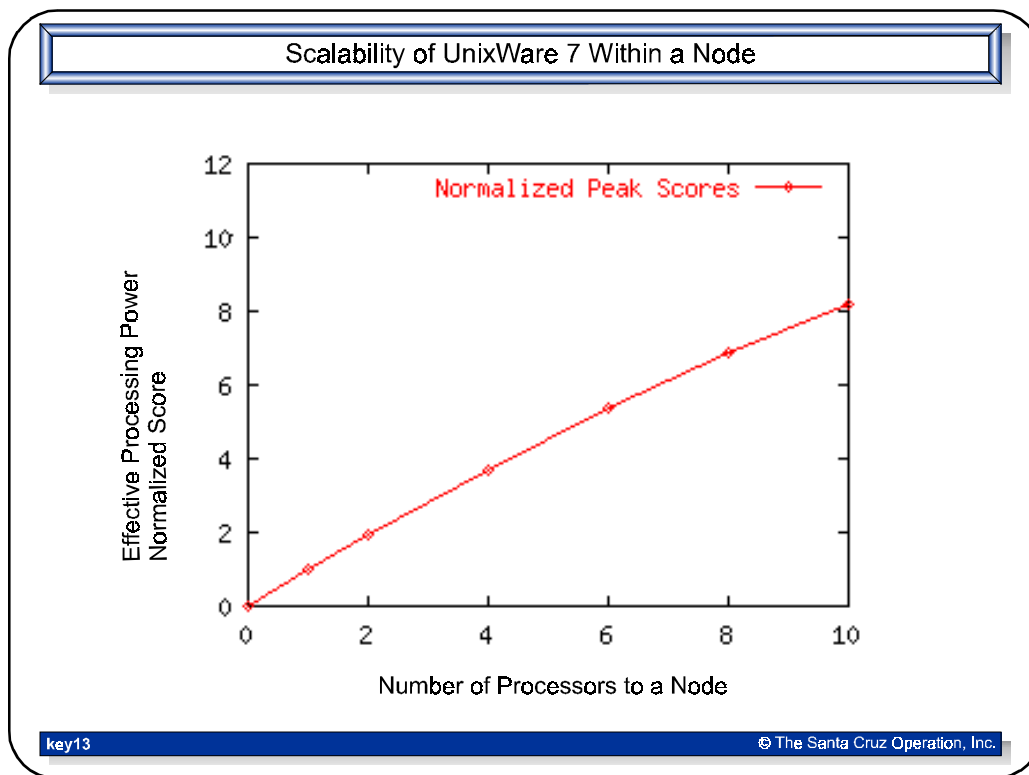
So far this course has discussed terms that apply to any clustering solution and has avoided mentioning any specific products that can be used to implement a cluster. This section introduces terms and components that are specific to NSC and discusses how these terms are used in the NSC environment.

3.2.1 SSI in NSC

NSC provides a full SSI implementation so that the entire cluster appears to the applications, users, and administrators as a single computing resource. At the present time, no other cluster implementation provides the level of SSI provided by NSC. NSC implements SSI in several ways:

- A clusterized kernel. As illustrated in AV 3-11 on page 3-26, the kernel has been modified from the standard UnixWare 7 kernel. This slide shows three types of modifications made when NSC is installed. Boxes with no lines are new capabilities, such as time-sync. Boxes with horizontal lines show added capability, such as *cspecfs* over *specfs*. Boxes with diagonal lines indicate merging of new capabilities, such as networking.
- A cluster-wide file tree and device nodes. The cluster filesystem (CFS) in the kernel implements a single file tree for the entire cluster. CFS intercepts filesystem calls and determines the actual location of the filesystem, routing the request to the node that controls the filesystem. All device nodes are visible to all processes without regard to location of device or process.
- A single process list. Only one process table exists cluster-wide. All process table entries are in the */proc* filesystem mounted at root. Administrators can manage processes without regard to the actual node on which the process is running.
- A single TCP space. Each node has a complete protocol stack and can use any NIC on the cluster.

AV 3-12



3.2.2 Scalability and Availability in NSC

There are two aspects to scalability within NSC:

- The scalability of adding another processor to a node. Since NSC uses the UnixWare 7 operating system, it has essentially the same scalability as UnixWare 7 for additional processors. This scalability is illustrated in AV 3-12 on page 3-28.

This slide shows the *normalized peak scores*, or the amount of work the node can do, with various numbers of processors present. At eight CPUs, the score is approximately 6.88. This indicates a scalability factor greater than 85 percent with eight CPUs.

- The scalability of adding another node to the cluster. NSC scales at 95% at 30 nodes. This means that if the performance of a single node yields throughput of 1 unit than the performance of 30 nodes yields throughput of $30 * .95 = 28.5$ units.

Several factors will increase the size and performance of NSC disk I/O in the future:

- Larger storage devices being certified for use with NSC
- Support for more than one HBA attached to protected storage per node. This is planned for a future release of NSC.
- Allowing more than two nodes to protect the same storage
- Allowing multiple nodes to perform physical I/Os on the same protected storage at the same time. This is planned for a future release of NSC.

AV 3-13

Features of Manageability in NSC

- Cluster-wide system administration
- Transparent Cluster-wide parallel TCP/IP networking
- Transparent Cluster-wide IPC
- Transparent Cluster-wide processes, migration, etc
- Automated application load balancing
- Applications, utilities, programmers, users and administrators often do not need to be cluster aware.
- Use of standard single node tools

key14

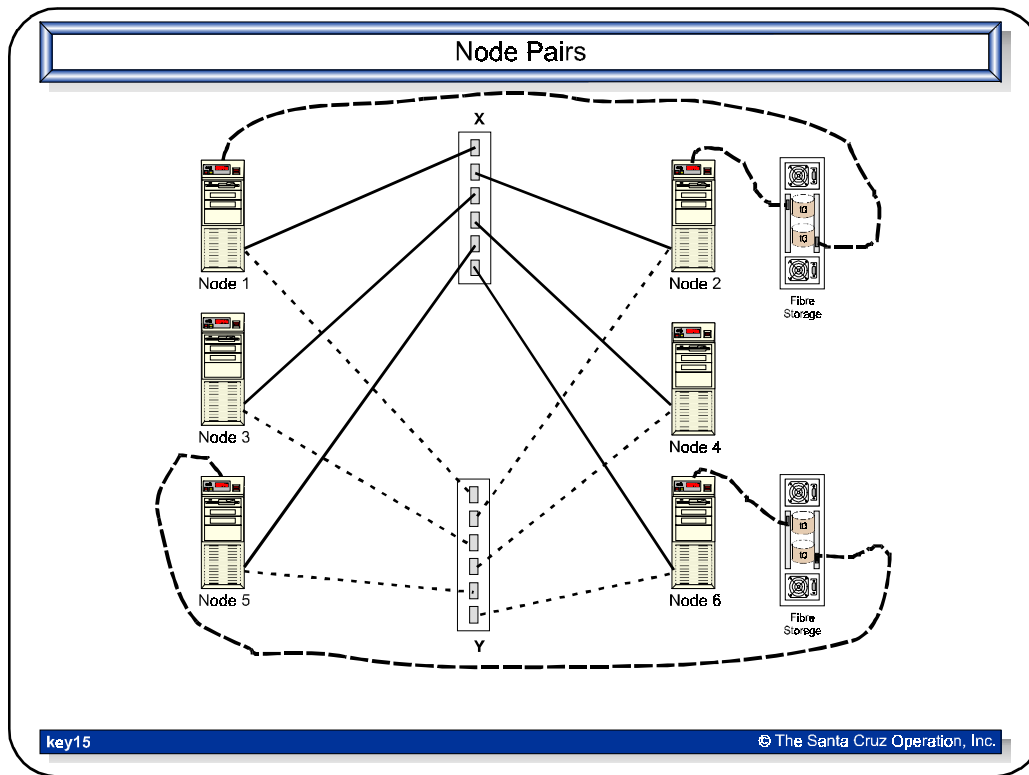
© The Santa Cruz Operation, Inc.

3.2.3 Manageability in NSC

The NSC SSI feature, combined with clusterized management tools, simplifies NSC administration. The degree of difficulty of NSC management is at the level of a single UnixWare host. Specifically:

- NSC has *clusterized* **SCOadmin** managers that perform their actions cluster wide. This is possible because of SSI, which stipulates that regardless of the size of the cluster, there is one password file, one file tree, one network configuration, one process list, and so on.
- Transparent cluster-wide parallel TCP/IP networking simplifies access to the cluster through a single IP address. This address, called the *Cluster Virtual IP* address (*CVIP*), acts like a network interface. The CVIP will failover from one node to another if the node with the CVIP fails. This provides uninterrupted service to clients.
- Using the **ps** command on any node lists all processes on the cluster. Any process can be signalled by using **kill** from any node without regard to the current location of the process. The administrator does not need to be aware of where the process is executing.
- Automated, configurable, load balancing relieves administrators of the task of constantly managing the work load.
- Applications, users, and administrators rarely need to be aware of the cluster. An application that runs on UnixWare 7 should run on NSC. A user might never be aware that their application is running on a cluster.
- Administrators and operators use standard UnixWare 7 commands to manage the cluster applications.

AV 3-14



3.2.4 Nodes in NSC

In addition to the concept of nodes discussed earlier, NSC has specific definitions of nodes. For a list of computers that have been certified for use as NSC nodes, see the SCO Clustering web site at URL:

<http://www.sco.com/products/clustering/uw7nschw.html>

3.2.4.1 Node Pairs

In NSC, a *node pair*:

- Consists of the two nodes that connect to any protected storage. In the current version of NSC, all protected storage must be protected by two nodes. Future releases may provide the ability for more nodes to participate in protecting storage.
- Contain nodes that are relatively similar in configuration to insure continued performance at the same level after a failover.
- Are designed using sequential odd-even numbered nodes. For example, the first node pair includes nodes 1 and 2, and the second (if present) includes nodes 3 and 4. This is illustrated by AV 3-14 on page 3-32.



3.2.4.2 Root node pair, potential root nodes, and current root node

The first two nodes are called the *root node pair* because:

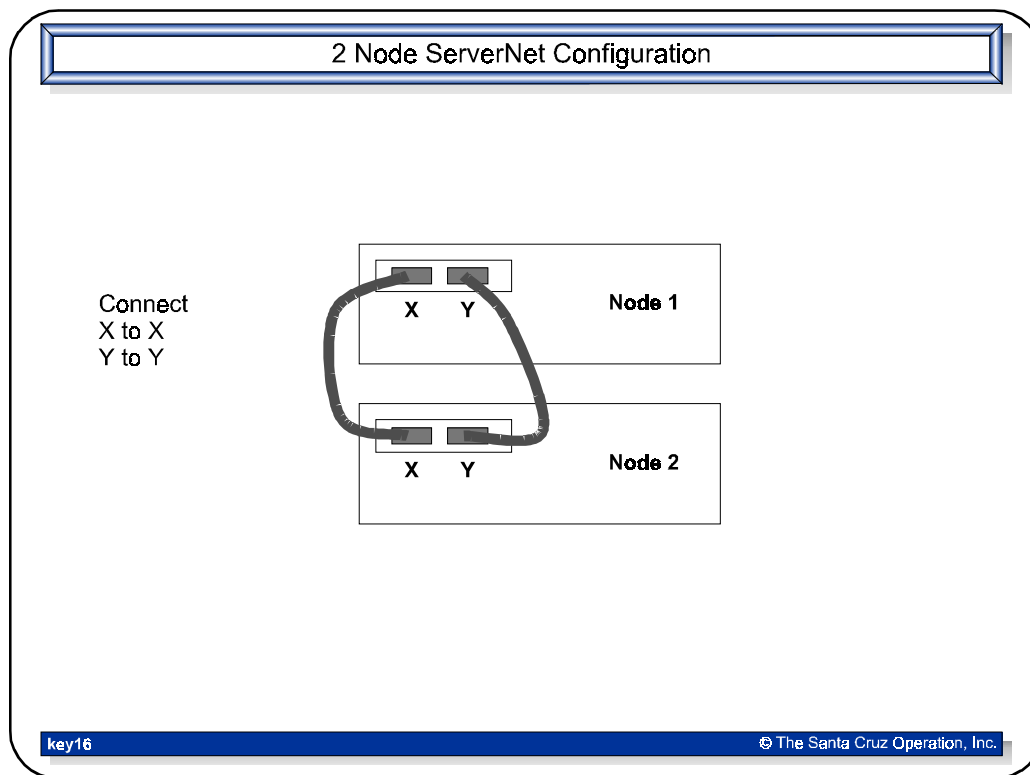
- When external storage is used for the *root* filesystem, one of these nodes physically manages the external storage of the *root* filesystem.
- When cross-node mirroring (CNM) is used, both nodes service read requests for the *root* filesystem. Writes are written to one disk and then mirrored to the other root node's disk.

Key properties of the root node pair include:

- One of these nodes must be in operation for the cluster to be available.
- The root node pair must have physical access to the disks with the root file system.
- They do not have to be the most powerful nodes in the cluster. Depending on the services provided by the cluster, the root node pair may be much less powerful than other nodes, with the powerful nodes running database applications and the root nodes providing less intensive services.
- Only one is acting as the root node at any time. This node is called the *current root node*.
- The other node, called the *potential root node*, while still capable of running applications, is standing by to assume the role of the current root node.
- Certain services only run on the current root node

In a future release of NSC, storage may be protected by more than two nodes. In that case, all nodes protecting the *root* filesystem will be *potential root nodes*. That is, any of those nodes could potentially become the root node.

AV 3-15



3.2.5 SAN options for NSC and the Virtual Interface Architecture (VIA)

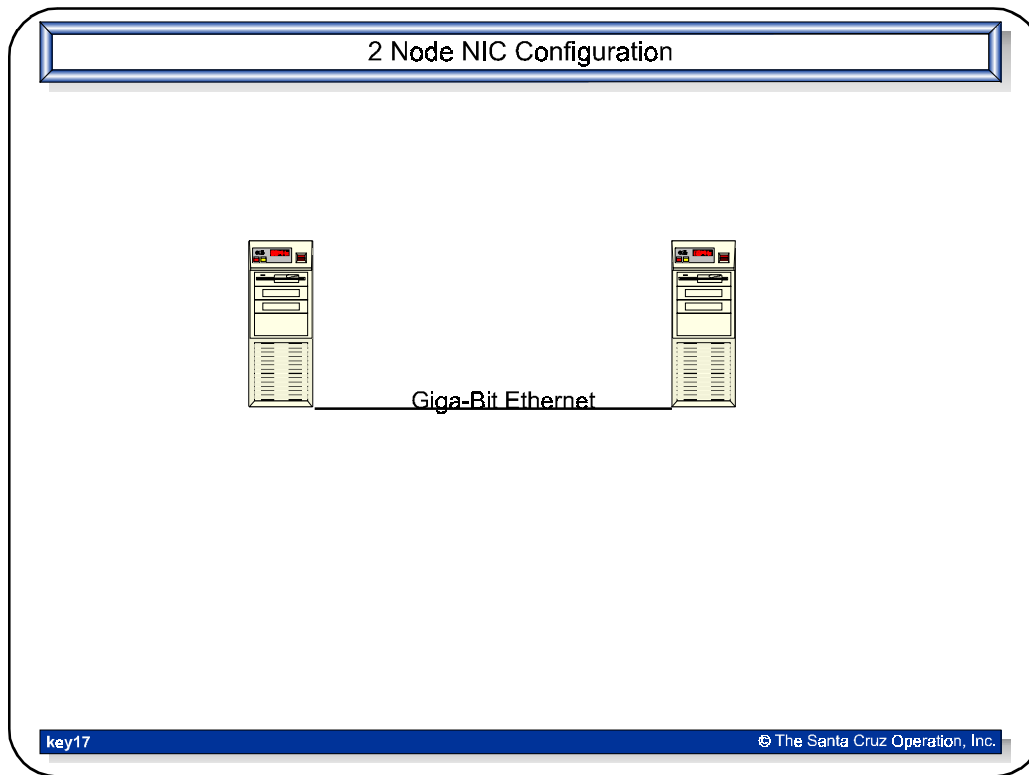
The options for implementing a SAN in NSC are being expanded rapidly. For a complete list of current options, see the SCO Clustering web site at URL:

<http://www.sco.com/products/clustering/uw7nschw.html>.

Projecting through the end of the year 2000, these SAN options are expected to be available:

- *ServerNet* is a proprietary solution from Compaq that:
 - Uses PCI cards that use few CPU resource because of the card's intelligence. They have two cable connections to allow transmitting and receiving messages on two interconnect fabrics, called X and Y.
 - Uses thick, many-pinned cables that are currently limited to 4 meters in length. This may be an issue for the physical location of nodes in a cluster.
 - Reaches higher maximum transfer rates because it does not use a standard networking protocol stack or physical layer
 - Has a transfer rate of 50MB/second (400Mb/second)
 - Has two possible configurations:
 - As illustrated in AV 3-15 on page 3-36, when only 2 nodes are present in a cluster, *ServerNet* uses only two cables, one from the X connector on one node's card to the X connector on the other node's card, and similarly for the Y.
 - As illustrated in AV 3-12 on page 3-28, when 3 or more nodes are present in a cluster, *ServerNet* requires a switch. It uses cables between the X and Y connectors on each node's card and the corresponding connector on the *ServerNet* switch.

AV 3-16



- Supported NICs. Releases 7.1.1b and later support a SAN architecture for only two nodes using Ethernet NICs. NICs up to gigabit capabilities will be included in the supported NICs. This implementation uses the network protocol stack to control traffic over the SAN and is illustrated in AV 3-16 on page 3-38.
- Virtual Interface Architecture (VIA). VIA was developed by Intel as a method for application software to access NICs directly, that is, without the need for kernel network protocol stacks or excessive user/kernel-mode transitions. VIA includes an API that must be implemented by library- and kernel-level routines.
- Servernet II. This is a follow-on to the current ServerNet technology. It will use VIA hardware as well.

3.2.5.1 Split Brain Avoidance

A *split brain* condition occurs when the potential root nodes all think that the other node is down and tries to become the root node for the cluster. In this scenario, one of three things occurs, all of which are cause problems and must be avoided:

- If you are using shared storage with an active/passive controller set, where each controller is attached to a potential root node, you will get *ping-pong panics*. In other words, one node will activate its controller, deactivating the other controller, causing the node that had the active controller to panic. When the panicked node reboots, it activates its controller, causing the other node to panic, and so on.
- If you are using JBOD, a type of dual-hosted SCSI external storage unit, data on the disks will likely be corrupted.
- If you are using CNM, both nodes update their own copy of the *root* filesystem. It will be almost impossible to merge the changes. Thus one set of changes will be lost, probably corrupting data. When a potential root node loses contact with the current root node, there are two possible reasons:
 - The root node has gone down. In this case, the potential root node must become the root node and reform the cluster.
 - The communication paths between the nodes have failed. In this case, the potential root node should take itself out of the cluster by panicking.

The same issues exist when the current root node loses contact with a potential root node. Either:

- The potential root node has gone down, in which case the root node must reform the cluster.
- The potential root node is up but can not be contacted. In this scenario, the current root node continues as the current root node assuming that the potential root node has made the same determination and panicked.

Certain SAN technologies can not distinguish between an inoperative connection and a node down. An example is a two node cluster using Compaq ServerNet without a ServerNet switch. When the potential root nodes lose contact and cannot distinguish between a node-down and an inoperable interconnect, a condition known as split-brain can occur.

To avoid a split brain condition when using such SAN technologies, you must configure *split-brain avoidance* by connecting a null-modem serial cable between the potential root nodes and setting up the NSC software to use this cable for split-brain avoidance.

3.2.6 Protected storage methods in NSC

NSC supports the following protected storage methods:

- *Cross-Node Mirroring (CNM)*. In a CNM cluster, the two root nodes use internal disks for the *root* filesystem. The two *root* filesystems are synchronized through Veritas Volume Manager. While this technique does not yield the performance of hardware RAID, the price is much lower. To improve performance, both nodes can read data from their internal disks, but only one node controls writes.
- *Shared storage*. This requires the use of external storage cabinets, populated with disks, connected to a node-pair. In a future release, NSC may support physically connecting protected storage to more than two nodes.

For a list of the protected storage components that have been certified for use with NSC, and their vendors, see the SCO Clustering web site at URL:

<http://www.sco.com/products/clustering/uw7nschw.html>..

3.2.7 Load balancing in NSC

In NSC, load balancing is turned off by default. When started, load balancing is available continuously to monitor CPU load and migrate processes among nodes. It is a highly-configurable subsystem, allowing for a great deal of flexibility in controlling process movement. For example, you can configure these metrics to control how load is balanced:

- What constitutes an *overloaded node*
- What constitutes an *underloaded node*
- How much CPU time a process running a candidate application must garner before it is eligible for migration
- The predicted effect on the load of the nodes affected by migration

Load balancing can greatly degrade performance of an application and the entire cluster in these circumstances:

- An application consists of multiple processes
- The processes perform a significant amount of interprocess communication

Databases, such as the non-parallel versions of Oracle, Informix, and Progress are all examples of applications that should not be subject to load-balancing.

The nodes of a cluster can be configured into *load balancing domains*. This allows you to configure which applications are balanced across which nodes. This allows you more fine-grained control over which applications are included in load-balancing among which nodes.

3.2.8 Physical and Virtual IP Network Addresses

Each NIC in a cluster has a *physical IP* address or *PIP*. All NICs can be on the same network or the cluster can have interfaces on multiple subnets. To simplify client access to the cluster, the cluster is assigned a *Cluster Virtual IP* address or *CVIP*. The CVIP is initialized on the first node to boot. The cluster sends out an ARP identifying the CVIP as having the MAC address of one of the NICs on the node to which the CVIP is currently assigned.

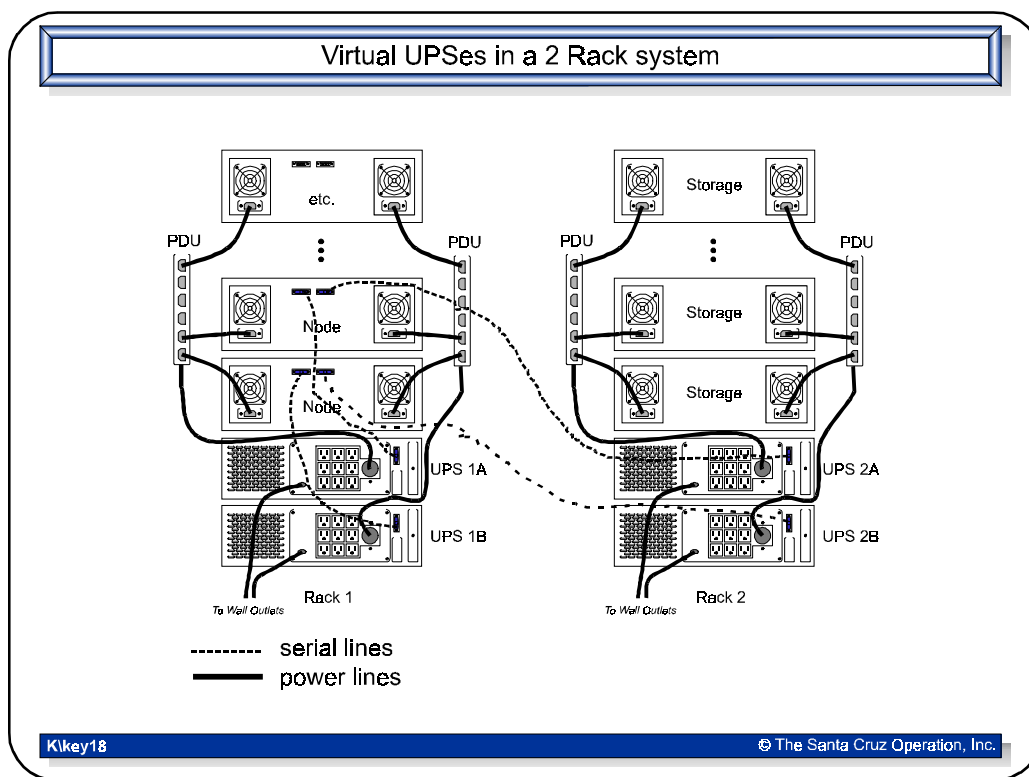
When clients need to access the cluster, they can access the CVIP or any of the PIPs in the cluster. The advantage of configuring clients to use the CVIP is that it will always be available if the cluster is running. A client that is configured to access a PIP will not connect if the node with the PIP is down.

Since there may be many NICs on different subnets, up to sixteen CVIPs can be configured, one per subnet. The primary CVIP is associated with the cluster name and should always be on a subnet that connects to the root nodes.

When the root node fails, the cluster will regroup and a potential root node takes the role of root node. Part of the regrouping process is to set up the CVIP on the new root node and send out the ARP broadcast to identify the MAC address of the CVIP. This insures clients have access to the cluster and do not have to reconfigure their telnet or ODBC attachments every time the CVIP is moved.

Data traffic emanating from the cluster can depart on any NIC. For this reason, any system to which the cluster sends data that is authenticated by IP address must consider the cluster as a multihomed host. For example, a print server must list the CVIP and all PIPs for the cluster in its *host.equiv* or *hosts.lpd* file.

AV 3-17



3.2.9 Virtual UPS

With single computers, the system is often configured to shutdown if the UPS detects a power failure. A cluster should have multiple UPSes for redundancy.

NSC allows you to specify which UPSes can shut down without shutting down the cluster. UPSes configured in this manner are called a *Virtual UPS*.

As shown in AV 3-17 on page 3-44, two UPSes (labelled 1A and 1B) could be configured to run the dual power supplies on a cluster's nodes. If one of the UPSes, 1A or 1B, shuts down, there is ample power on the remaining UPS to run the cluster. Consider UPSes 1A and 1B to be Virtual UPS 1.

Likewise, the external storage arrays can be powered by two UPSes, labelled 2A and 2B. If one of those UPSes fails, the external array can continue to run. Consider UPSes 2A and 2B to be Virtual UPS 2.

NSC allows you to configure the virtual UPSes with a control file. When power failure signals are received, the NSC virtual UPS software can examine the status of each UPS and make an intelligent decision on shutdown.

In the example above:

- If any one fails, the cluster will continue to run.
- If either of the internal UPSes in a virtual UPS fail, the cluster will continue to run
- If a virtual UPS loses power (all internal physical UPSes lose power), then the cluster must shut down.

Virtual UPSes provide an additional layer to the concept of no single point of failure. In reality, the UPSes within a virtual UPS should be connected to different generators and power grids. For example, UPS 1A and 2A can be connected to one wiring grid backed up by a generator, and UPS 1B and 2B can be connected to an alternate grid and generator.

3.2.10 Clusterizing

While the existence of a cluster is transparent to an application, you can choose to make your applications cluster-aware. This process is referred to as *clusterizing*.

Clusterizing applications is beyond the scope of this course. The **UnixWare® 7 NonStop® Clusters Administration** course provides basic information about clusterizing shell scripts, configuring applications for failover, and troubleshooting applications running in an NSC environment.

Additional information about optimizing applications for NSC is available from the SCO Clustering web site at URL:

<http://www.sco.com/products/clustering/optc/download.html>

3.3 Lesson Summary

This lesson presented key concepts generic to all clustering environments and NSC specifically.

Review questions

1. What does no single point of failure mean?
2. How does protected storage differ from a RAID array connected to a server?
3. How does single system image reduce the work of managing an NSC cluster?
4. What is the difference between a clusterized and non-clusterized command or **SCOadmin** manager?

Review Question Answers

This section contains the answers to the review questions on page 3-47.

1. No single point of failure means that each fault zone has redundant components.
2. Information stored on a disk that is accessible from multiple nodes it is considered *protected*. Storage must be protected to avoid single point of failure. A RAID array is connected to a single server.
3. The SSI reduces the work of managing a cluster because it provides a view of the cluster as a single system. Regardless of the size of the cluster, there is one password file, one file tree, one network configuration.
4. A clusterized command or SCOadmin manager is made aware of the cluster. For example, the **SCOadmin Process Manager** displays the list of processes that are running cluster-wide rather than the list of processes on a single node. A non-clusterized command is not aware of the cluster. For example, the **SCOadmin Video Configuration Manager** is not cluster aware and will only show the video configuration of the node that it is running on.

4

Cluster Comparisons



Lesson 4

Cluster Comparisons

Introduction

There are several products on the marketing that provide one or more aspects of clustering as defined in the previous lesson. Even if they only implement one of the aspects of clustering, they are referred to as a “cluster” product.

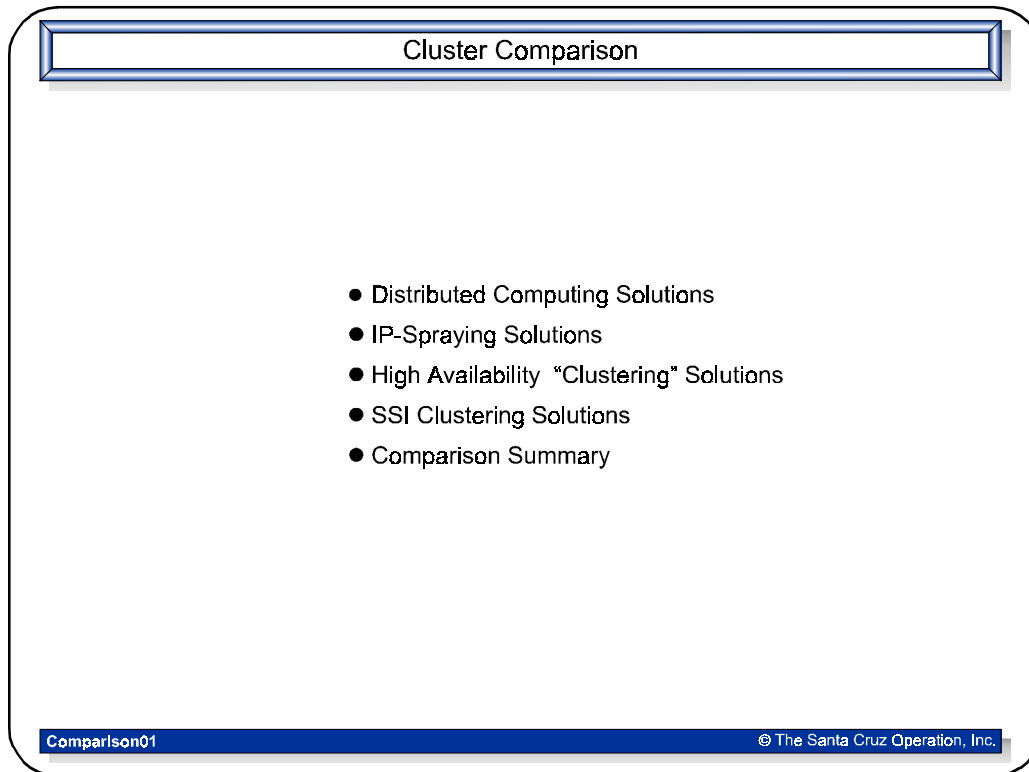
This lesson discusses some of the other “cluster” products and compares and contrasts them with UnixWare[®] 7 NonStop[®] Clusters (NSC).

Objectives

By the end of this lesson, you will be able to:

- List the names of some of the products with which NSC competes
- Indicate which products are truly cluster products
- Indicate which products are really not cluster products, and why
- Give at least three reasons why NSC is a better solution than its competitors

AV 4-1



Topics

In order to meet these objectives, this lesson discusses:

- Distributed Computing Solutions — products that claim to be clustering but are really just *distributed computing*
- IP-Spraying Solutions — products that claim to be clustering but are really a form of *IP-spraying*
- High Availability “Clustering” Solutions — products that exhibit many of the features of clustering, but do implement single system image (SSI)
- True Clustering Solutions — products that exhibit all of the features of clustering, including single system image (SSI).
- Comparison Summary — shows the differences between the various “clustering” products

4.1 Distributed Computing Solutions

The idea of using the CPUs of multiple computers to increase performance of applications has been around for many years. Many of these types of systems have been developed, both for commercial and experimental use.

However, these systems are not clustering solutions as defined by this course.

- They do not have any sort of single system image (SSI).
- They do not share a filesystem.
- Their only form of cooperation is sharing the answers to their computational problems.

Therefore, such systems are typically only useful for performing distributed computing or distributed number crunching.

Examples of these types of solutions include:

Beowulf This is a solution that uses many commodity, off-the-shelf computers, linked by an LAN, to solve specific computational problems. For more information on this solution, see <http://beowulf.gfsc.nasa.gov>.

SETI@home This is a project that is used to search through the data generated by the radio telescope run by the Search for Extra-Terrestrial Intelligence (SETI) project. The goal of the project is to find evidence of intelligent life in the radio telescope data. For more information on this solution, see <http://www.setiathome.ssl.berkeley.edu>.

Note The SETI@home project does not claim to be a clustering solution. However, it is an example of a distributed computing solution.

In each of the examples above, each computer is its own independent unit, with its own disks, user database, and so on. The only resemblance they have to clustering is that there are multiple computers involved and they are all cooperating on the same task.

4.2 IP-Spraying Solutions

The advent of the World Wide Web also generated a need for a single organization to be able to distribute the load of incoming HTTP requests across many computers. That is, when a request comes into an HTTP daemon on one machine, it can deflect the request to another server within a cooperating group. This group is referred to as a “cluster.”

This same approach can be used for other Internet-related services, but is primarily currently used only for HTTP servers, which serve most of the Web content.

An example of a solution that uses this approach is TurboLinux Turbo Clusters. However, these solutions are not clustering, as defined in this course, for several reasons:

- They do not have any sort of single system image (SSI).
- While they may NFS-mount the HTTP documents to be served, each machine has its own filesystem.
- There is no dynamic load balancing.

4.3 High Availability “Clustering” Solutions

There are a number of products that closely resemble a cluster, as defined in this course. The key feature they all lack is single system image (SSI) and dynamic load balancing. These two features are critical to cluster management and scalability.

Most of these high-availability solutions have shared filesystems and some form of failover. Examples include:

- Novell Cluster Server (NCS)
- Microsoft Wolfpack
- Windows 2000 Data Center Edition
- IBM HACMP
- SunCluster
- HP ServiceGuard
- DEC Truclusters
- SCO Reliant HA

4.4 True Clustering Solutions

There is currently only one clustering solution that meets the full definition of a cluster, as presented in this course; UnixWare® 7 NonStop® Clusters (NSC). It implements SSI, shared storage, failover of disk and applications, as well as dynamic load balancing of applications.

Because NSC implements SSI, standard UNIX administration skills can be directly applied to managing a UnixWare 7 NonStop Cluster.

AV 4-2

Comparison Summary								
Solution Type	Example Solution	SSI	Shared Disk	Disk Failover	Application Failover	Dynamic Load Balancing	Max Nodes	CPUs / Node
Distributed Computing	Beowulf	N	N	N	N	N	Unlimited	HW/OS Dependent
Distributed Computing	SETI@Home	N	N	N	N	N	Unlimited	HW/OS Dependent
IP Spraying	TurboLinux Turbo Clusters	N	HTTP Docs Only	N	Y	HTTP Requests Only	Unspecified	8
High Availability	Novell Cluster Server (NCS)	N	Y	Y	Y	N	32	8
High Availability	Microsoft Cluster Server (Wolfpack)	N	Y	Y	Y	N	2	4
High Availability	Microsoft Windows 2000 Data Center Edition	N	Y	Y	Y	N	4	8
High Availability	IBM HACMP	N	Y	Y	Y	N	32	24
High Availability	HP Serviceguard	N	Y	Y	Y	N	16	Unspecified
High Availability	DEC Trueclustes	N	Y	Y	Y	N	4	14
High Availability	SCO Reliant HA	N	Y	Y	Y	N	10	32
Real	UnixWare 7 NonStop Clusters	Y	Y	Y	Y	Y	30	32
Comparison02						© The Santa Cruz Operation, Inc.		

4.5 Comparison Summary

AV 4-2 on page 4-10 summarizes the different “clustering” solutions discussed in this lesson, grouped by type of solution. Note that only UnixWare 7 NonStop Clusters has a “Y” in each column indicating capabilities found in true clustering solutions.

4.6 Lesson Summary

This lesson discussed the various kinds of solutions on the market today that claim to be clustering solutions and explained why most of these solutions are not true clustering, as defined in this course.

Review questions

1. List three products on the market today that claim to be clustering solutions.
2. Provide two reasons why Microsoft Wolfpack is not true clustering, as defined in this course.
3. Provide two reasons why TurboLinux Turbo Clusters is not true clustering, as defined in this course.
4. Give three reasons why NSC is a better solution than its competitors.

Review Question Answers

1. The products on the market today that claim to be clustering solutions include:
 - Beowulf
 - TurboLinux Turbo Clusters
 - Novell Cluster Server (NCS)
 - Microsoft Cluster Server (Wolfpack)
 - Microsoft Windows 2000 Data Center Edition
 - IBM HACMP
 - HP ServiceGuard
 - DEC Truclusters
 - SCO Reliant HA
 - SCO UnixWare 7 NonStop Clusters (NSC)
2. The two reasons why Microsoft Wolfpack is not true clustering, as defined in this course, are that it lacks single system image (SSI) and dynamic load balancing of applications.
3. There are three reasons why TurboLinux Turbo Clusters is not a true clustering solution, as defined in this course. They include:
 - It does fully share disk storage between servers
 - It does not implement single system image (SSI)
 - It does not perform dynamic load balancing of applications
4. Three reasons why NSC is better than other solutions listed in this course include:
 - It has full single system image (SSI)
 - It is easier to administer
 - It can perform dynamic load balancing of applications, increasing scalability



5

Designing an NSC Cluster



Lesson 5

Designing an NSC Cluster

Introduction

A cluster is more than a collection of computers, hence it is more complex to design and build. Cluster design must include careful consideration of the overall system requirements, including performance goals, failover capability, and redundancy within fault zones.

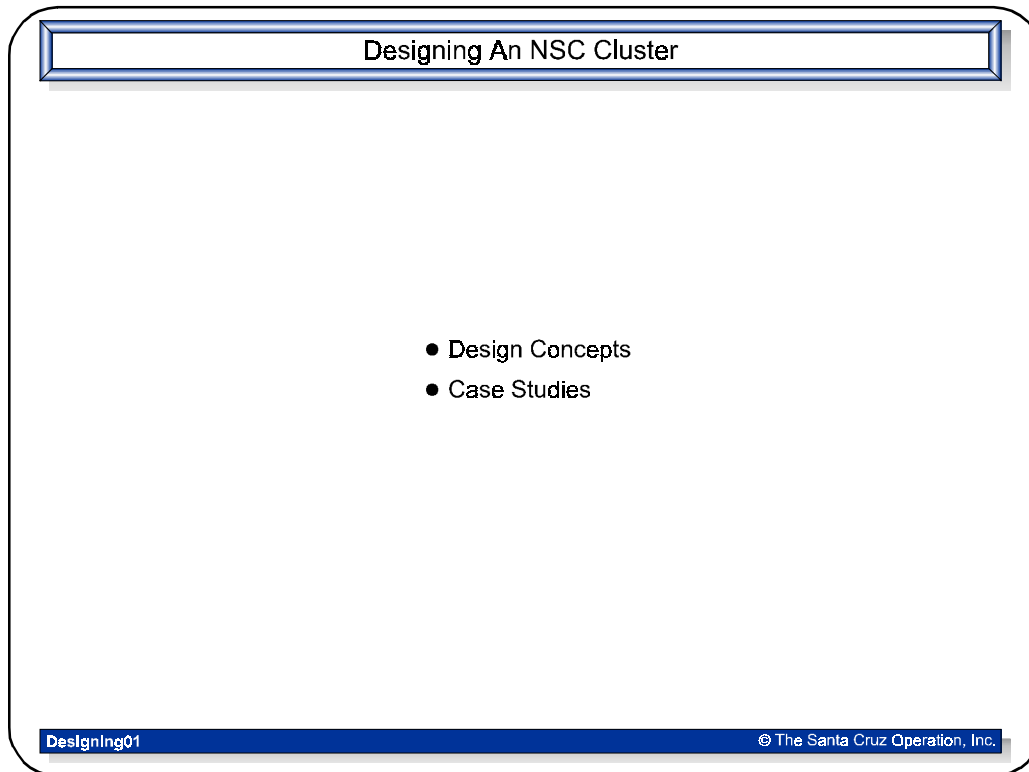
This lesson presents guidelines and rules for designing clusters.

Objectives

By the end of this lesson, you will be able to design a cluster based on a set of system criteria, including:

- I/O performance
- Vulnerability time
- Failover
- Load balancing
- Hardware capabilities, such as CPU cycles, amount of RAM, and disk space

AV 5-1



Topics

This lesson discusses:

- Design Concepts — using guidelines and rules for designing a cluster
- Case Studies — citing small, medium, and large systems with varying needs.

5.1 Design Concepts

When designing a cluster, carefully consider these criteria:

- Minimizing cluster vulnerability and avoiding single points of failure.
- Selecting and configuring nodes
- Deciding whether to use a tower or a rack
- Protecting your storage
- Protecting your power

Each of these issues is discussed more fully in this section.

5.1.1 Vulnerability and single point of failure

The two key design issues for clusters are *vulnerability* and *no single point of failure*.

No single point of failure is paramount to cluster design. You cannot over-emphasize redundancy in your cluster design. Customers that are extremely cost-conscious might want to trade cost savings for adequate design. *Resist these attempts*; you will certainly be blamed if the cluster fails, regardless of whether it was the customer who encouraged design changes that introduced points of failure.

Keep these points in mind:

- Plan for hardware failure and design the cluster around component failures
- Plan for electrical failures and design the power system accordingly. Use redundant generators supplying redundant power to the redundant UPSes.
- Plan for slow response time by component vendors and maintain a spare parts inventory. Geographical location, vendor response time, and other factors determine how many spares of each component to keep on hand. The more remote the location, the more components you should have on hand.

Include these components in your inventory:

- Keep enough disk drives on hand to withstand a failure of some percentage of disks. Consider the need for same or similar sized drives in RAID arrays.
- Consider keeping a complete spare node on hand. In a two node cluster, the vulnerability of the cluster is large as long as one node is down. A spare node can be used for daily non-essential tasks and added to the cluster when a node fails.
- Consider the need for spare cables. While cables do not fail frequently, pins can be bent when cabling repaired components.
- Check the condition of the batteries and keep replacements on hand. UPS batteries fail with age.

In addition to preventing a single point of failure, you must consider the vulnerability of the cluster when a single component fails.

If a component fails, the entire cluster is vulnerable to complete failure until the component is replaced. You should replace failed components as quickly as possible.

Consider CNM vulnerability during re-mirroring of the root hard drive. The cluster cannot withstand a failure of the current root node while the potential root node is being re-mirrored.

5.1.2 Node design

Node capacity is important to the overall performance of the cluster. Because nodes are usually used in pairs to protect disks and other resources, the two nodes in a node-pair are usually configured equally.

Determining node capacity requires an overall view of the cluster, its components, the workload, load balancing and the failover design. First consider the root node. It:

- Is required to access the root disk
- Might also control the disks used for non-root data storage
- Can perform tasks other than serving the *root* filesystem
- The root node and the failover nodes are approximately the same configuration. If the current root node fails, a potential root node assumes the role of current root node. The configuration for node 1 is used for other root node candidates.
- Might or might not have disks directly attached. Remember, all nodes have an internal disk for booting.
- Is sized according to the expected usage. The amount of RAM and the number and speed of processors are the main considerations.
- Should have ample network access. Plan to include one or more NICs per node. While each node has a complete TCP stack and can use the NICs on other nodes, the preferred path is through a local NIC. Use additional NICs to provide access to additional subnets.
- Should have enough PCI slots to handle the boards for disk controllers, NICs, SAN connection, serial and other connections
- Should have redundant tapes and CD-ROM drives.

With rack mounted systems that have external storage, do not choose nodes with large capacity (lots of slots) for internal disk drives. Because only one internal disk is normally used for booting, the unused disk slots in the server take up rack space. Further, the cost of nodes increases with the increased capacity for internal disks.

When sizing a particular node, consider the applications requirements:

- Is the application a single monolithic database?
- Will multiple applications (mixed workload) or multiple copies of a single program be run?
- Do all applications require maximized availability? Is downtime acceptable for certain applications?
- Is degraded performance acceptable when component failures occur?
- How do components of the application communicate?

Monolithic databases require heavy inter process communications and should not be spread across multiple nodes.

- What resources are required by the applications? Disk access? RAM? CPU?
- How do the applications scale?
 - For monolithic applications, add CPUs
 - For process replication, add nodes
- Do the applications use the disk primarily for reads or writes?

5.1.2.1 Cluster groups

Determine the resources required for each application or group of applications by:

1. Dividing the applications into groups
2. Analyzing the requirements of each group
3. Mapping the requirements to physical resources

Determine these requirements by calculating each application's resource needs:

- Disk space required for the applications and their data
- The disk I/O workload
- RAM amounts for each application running simultaneously
- Number of CPUs to handle the processing

After the physical requirements are established, consider the cluster requirements from an availability and scalability perspective. Is the configuration primarily for:

- High-availability—such as a small server that provides a critical business service
- Scalability—the cluster is expected to grow
- Both scalability and availability—you cannot afford downtime and need expansion capability

Another factor is the sensitivity to cost. A small business installing their only server has a different view of cost than an international company rolling out hundreds of replicated sites. Cost can be contained by using CNM instead of external disks, or by accepting some performance degradation if a node fails.

5.1.2.2 Mapping applications to nodes

After the computing requirements are determined, map the applications to nodes. Configure the nodes according to their work load. Keep in mind that external disks provide better performance than CNM.

If the total computing requirements easily fit into a single computer and the reason for clustering is high availability, a two node cluster is sufficient. With a two node cluster, other considerations are:

- Disk I/O requirements for the applications:
 - If the I/O requirements are low, then a CNM system provides a low cost hardware solution.
 - If disk I/O requirements are high, a high-performance external disk enclosure with RAID or JBOD is needed
- With a two node cluster, particularly with CNM, a pair of tower servers is less expensive than a rack mount system and cabling should not be a problem.
- If the cluster is expected to grow rapidly, a rack mount configuration should be considered.
- Allow for expansion to accommodate increases in computing requirements

If the applications fall into two or more distinct groups, decide if the groups can be run on different nodes, each with its own failover resources. For example, an e-commerce server might have a large, monolithic database that requires a node pair for the database and a web server running multiple copies on several other nodes.

Plan for the failover resources. In the above example, the database runs in an N+1 configuration: one server in use and one standing by to run the database in the event the first node fails. The multiple web server processes can run on servers in an N+0 configuration. If one node fails, the others share the work although there might be some performance degradation. To avoid degradation in an N+0 configuration, add an additional node to cover the loss of performance.

5.1.3 Node format—tower or rack mount

Clusters might be small enough to use two tower nodes connected by SAN cables. If the cluster grows beyond two nodes, a rack system must be considered. Racks provide an organized method of mounting the nodes and other equipment, as well as a mechanism for organizing cables.

Consider the cabling for a six node cluster using Compaq ServerNet, three external disk arrays, a keyboard, video, mouse (KVM) switch, and a pair of UPSes:

- Six nodes with two ServerNet cables and dual power supplies each requires 12 cables and 12 power cords
- The keyboard/video/mouse switch requires 21 cables—1 for each keyboard, video, and mouse, times 6 nodes is 18 cables, plus 1 for each keyboard, video, and mouse going to the actual keyboard, monitor, and mouse
- One power supply for the KVM switch requires one (1) power cord
- One power supply for the monitor requires one (1) power cord
- Three external drive arrays:
 - Connected to drive controllers requires 6 fiber cables
 - With dual power supplies requires 6 power cords
- Two UPSes connected to power outlets requires 2 power cords
- Two ServerNet switches require 2 power cords
- Additional cables needed for network switches or hubs, drive array hubs, external tape drives, and so on

For this 6 node cluster, a minimum of 39 cables and 24 power cords are required. These must be labelled and secured within the rack. Cables should be color coded or labelled by node number and additional coding for X or Y fabric. One method is to use color coded tape on each end with the X fabric cables having an additional white tape strip.

When planning for a rack mount system, the following must be taken into consideration:

- Where will the rack be located?
 - Will the floor hold the concentrated weight of the rack?
 - Is there adequate cooling?
 - Is there adequate power?
 - Is the area large enough to allow working around the rack?
- Will the rack arrive pre-configured?
 - How will the rack be moved from the loading dock to the computer room?
 - Is there sufficient clearance through doorways and halls to move the rack? Note: loaded racks should not be tipped over to go through doorways.
 - Are the floors level and smooth between loading dock and computer room?
- If the rack is configured on-site:
 - Is there adequate room for unboxing the components and assembling the rack?
 - Will there be enough help to assemble the rack? Some components are heavy and require two people for assembly.
 - Consider the layout of the rack ahead of time. Some cables might be limited in length, forcing the placement of components based on cable length. Power cables should not run between racks when multiple racks are being used. Place heavy components low in the rack.

5.1.4 Disk configuration

While all filesystems and disk devices are transparently accessible from all nodes, accessing data on disks attached to another node is slower because it requires moving the data across the SAN. While the SAN is a high speed connection, there is no advantage to moving heavy data traffic over it and adversely impacting the entire interconnect.

Therefore, where possible, disk drives are attached to the nodes that access the drive's data. For example, if a database is located on an external drive array and the database application is run on nodes three and four, consider attaching the external drive array to nodes three and four.

The two primary methods of configuring protected storage are external shared storage or CNM.

- Use external storage when:
 - Data storage is greater than 20 GB (consider re-mirror time)
 - Data access performance is critical
 - Storage expansion will be required
- Use CNM when:
 - The installation is cost sensitive
 - Limited storage requirements
 - Write performance is not an issue

5.1.4.1 CNM protected storage

CNM root disks are internal to the two root nodes and are mirrored using the **Veritas Volume Manager**. The advantage of CNM is low cost. CNM does not require an external disk cabinet nor does it require fibre array or other disk array controllers.

The two major disadvantages of CNM are:

- Write performance
- Vulnerability to failure during re-mirroring

When using NSC, all I/O must be performed by the node controlling the filesystem. In a CNM configuration, I/O requests originating on the non-controlling node must be sent over the SAN to the controlling node:

- In the event of a read request, the current root node returns the requested data to the requesting node over the SAN.
- In the event of a write request, the requesting node must send the data over the SAN with the write request, then the current root node must write the data to its local disk, *and* send the write request (with data) over to the potential root node.

This extra traffic for a write makes CNM less suitable for environments with a high volume of write requests.

The other major disadvantage of CNM is vulnerability to single point of failure. If either root node fails, upon rebooting, the *root* filesystem must be mirrored across the SAN before the restarted node is capable of performing as a root node. The re-mirror time is approximately 7 minutes per Gigabyte.

During the re-mirror time, the potential root node is available for work, but not to assume the role of current root node. Because of this window of vulnerability, you should limit the size of CNM disks to 20GB or less. The re-mirroring problem does not exist with the external shared solution.

5.1.4.2 External shared storage

External shared disks reside in disk cabinets. The cabinets themselves may be RAID enabled. These storage units:

- Can be attached to the nodes where primary access will occur
- Have no window of vulnerability for re-mirroring the *root* filesystem each time a root or potential-root node fails
- Are more expensive, more flexible in arrangement, and capable of more storage than CNM
- Should be used where high I/O and large data volumes are required

5.1.5 Electrical power and UPS

You must determine the cost-per-hour of potential downtime due to power loss and consider using virtual UPSes with generators as a backup power supply.

If generators are used, check them for proper configuration. Specifically check to ensure that the fuel pumps for the generators are not powered by commercial power, but by the generators themselves.

Test your UPS design before the cluster goes live.

5.2 Case Studies

The following case studies provide four different scenarios for clusters.

5.2.1 Small business or replicated site

This is the low end of the cluster spectrum. The requirements include:

- Less than 20 GB of storage is required.
- Applications easily run on a single server.
- Expansion of processing power is not a concern or can be resolved with CPU and RAM upgrades
- Low disk I/O load
- Extremely cost-sensitive customer
- Availability is very important
- Space is not a limitation

Exercise: Design the SMB replicated site cluster

Design a cluster that meets the specifications above. Include the number of nodes, protected storage option, and if it is a rack or tower configuration.

This is a good configuration for small businesses that cannot afford downtime. Businesses that are located in remote areas and require the computer to be available nights and weekends, can benefit from this approach.

5.2.2 Large storage, low processing

A step up from the previous system is the requirement for a large storage subsystem without an increase in processing power. The requirements include:

- Greater than 20 GB of storage required, perhaps into the hundreds of gigabytes. Ability to expand storage is a requirement.
- The application easily fits on a single server.
- Expansion of processing power is not a concern or can be resolved with CPU and RAM upgrades
- Low to heavy disk I/O load
- Cost sensitive installation and availability is very important

Exercise: Design the large-storage, low-processing cluster

Design a cluster that meets the specifications above. Include the number of nodes, the protected storage option and if is a rack or tower configuration.

This system can support a large database and provides failover in the event of node failure.

5.2.3 Low storage, high processing

In this scenario, the system has low storage and high processing power requirements. This is typical of a graphics-intensive or perhaps a compute-bound scientific application. The requirements include:

- Less than 20 GB of storage required
- The application(s) will not easily fit on a single server, very high processing power is required
- Expansion of storage is not an issue
- Expansion of processing power is a requirement
- Low disk I/O load
- Scalability and availability are important

Exercise: Design the low storage, high processing cluster

Design a cluster that meets the specifications above. Include the number of nodes, the protected storage option, and if it is a rack or tower configuration.

5.2.4 Heavy processing, heavy I/O

At the high of the spectrum are the large clusters, such as those used for heavy-volume e-commerce. These clusters have at least two application groups: web servers and databases. The requirement includes:

- Hundreds of gigabytes of storage
- The database application fits on a single server. Multiple copies of the web servers must be running
- Expansion of storage is an issue
- Heavy requirement for network access
- Heavy disk I/O load
- Performance-sensitive installation.
- Availability and scalability are very important.

Exercise: Design the high storage, high processing cluster

Design a cluster that meets the specifications above. Include the number of nodes, protected storage option and it is a rack or tower configuration.

5.3 Lesson Summary

This lesson described the considerations for designing a cluster. Several case studies were presented.

Review questions

1. List two considerations for designing a cluster disk configuration.
2. List two considerations for the number of nodes in a cluster
3. What is the maximum size of required storage that is recommended for CNM? Why?
4. When should a rack mount be used for a cluster?

5.4 Answers to Exercise and Review Questions

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Exercise answers

“Design the SMB replicated site cluster,” on page 5-17

A two-node CNM installation is sufficient. There is no requirement for expansion, SAN switches, or other external equipment to complicate the cabling. A pair of tower systems is a good choice.

Choose servers with adequate RAM and CPUs for the application load.

“Design the large-storage, low-processing cluster,” on page 5-18

The solution is a rack mount system with external shared storage. The need for a large amount of storage, combined with high-availability requirements, rules out the use of CNM because of the limits on internal storage and the time to re-mirror.

“Design the low storage, high processing cluster,” on page 5-19

Because of the large processing power required, use the maximum number of supported nodes. NSC 7.1.1 currently supports six.

Use CNM due to the low storage capacity and I/O load.

With more than two nodes, depending upon your SAN technology, SAN switches might be required, as well as a KVM switch.

With all the cable requirements, a rack mount would be the solution. Plan expansion room for additional nodes and external storage.

“Design the high storage, high processing cluster,” on page 5-20

For this cluster, use external storage and multiple nodes in a rack mount configuration.

Use one 1+1 node pair for each database served. Only combine the databases on the same node pair where performance allows.

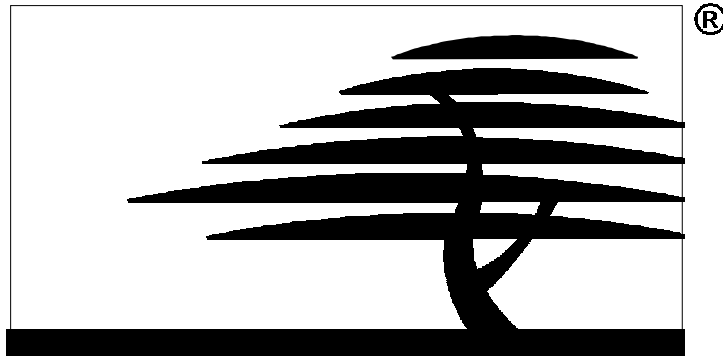
Use several nodes configured in a N+1 configuration for the web servers. Components will likely include a SAN switch, a KVM switch, external storage units attached to multiple node pairs, and possibly fibre channel hubs to connect multiple external storage units to one node pair.

Review question answers

This section contains the answers to the review questions on page 5-21.

1. Per Section 5.1, “Design Concepts,” on page 5-6, the issues you must consider when designing a cluster configuration include:
 - No single point of failure
 - Cluster format (tower or rack)
 - Power of machines including CPU speed, RAM, and disk size
2. Per Section 5.1.2, “Node design,” on page 5-8, the issues you must consider when deciding how many nodes to place in the cluster include:
 - Mapping of applications to nodes
 - Providing sufficient processing power to meet performance specifications
3. Per Section 5.1.4, “Disk configuration,” on page 5-14, the maximum amount of storage you should have on a CNM system is 20 MB, due to the time it takes to re-mirror a volume in the event of a node-loss.
4. Per Section 5.1.3, “Node format—tower or rack mount,” on page 5-12, you should consider using a rack-mounted system whenever the cabling or floor space required for a tower system becomes unmanageable. While the amount of cabling does not decrease, a rack provides better tools for managing the cables.





SCO

UnixWare® 7 NonStop Clusters® Installation

Student Guide

Course Release: 1.0

For Software: UnixWare® 7 NonStop® Clus-
ters Version 7.1.1

©2000 The Santa Cruz Operation, Inc. All rights reserved.

This publication is protected under copyright laws and international treaties.

Information in this document is subject to change without notice and does not represent a commitment on the part of The Santa Cruz Operation, Inc.

SCO documents are provided as is and may include technical inaccuracies or typographical errors. The Santa Cruz Operation, Inc. reserves the right to add, delete, change or modify SCO documents at any time without notice. SCO, the SCO logo, The Santa Cruz Operation, SCO OpenServer, Tarantella, the Tarantella logo and UnixWare are trademarks or registered trademarks of The Santa Cruz Operation, Inc. in the USA and other countries. IXI, X.desktop, Deskworks, Wintif, IXI Panorama, Deskterm, and IXI Desktop are trademarks or registered trademarks of IXI Limited, a subsidiary of The Santa Cruz Operation, Inc. Visionware, Devkit, Vision, Esprit, Kodon, Super.Vision, Visionware Super.Vision, Term.Vision, Visionware Direction, the Visionware logo, Visionware SQL-Retriever, and XVision are registered trademarks of Visionware Limited, a subsidiary of The Santa Cruz Operation, Inc. SuperVision, TermVision, X-Visionware, SQL-Retriever, and Vision Builder are trademarks of Visionware Limited. UNIX is a registered trademark of The Open Group in the United States and other countries. Cheyenne and ARCServe are registered trademarks of Cheyenne Software, Inc. Netscape, Netscape Navigator, Netscape Communications Server, Netscape Commerce Server, Netscape Proxy Server, Netscape FastTrack Server, Netscape Enterprise Server, Netscape SuiteSpot, Netscape Catalog Server, Netscape News Server, Netscape Mail Server, and Netscape Navigator Gold are trademarks of Netscape Communications Corporation. NFS was developed by Legent Corporation (formerly Lachman Associates, Inc.) based on LACHMAN SYSTEM V NFS. LACHMAN is a trademark of Legent Corporation. NFS is a trademark of Sun Microsystems, Inc. TCP/IP was developed by Legent Corporation (formerly Lachman Associates, Inc.) based on LACHMAN SYSTEM V STREAMS TCP, a joint development of Lachman Associates and Convergent Technologies. MPX was developed by Corollary, Inc. VP/ix is a product developed and licensed by Phoenix Technologies, Ltd/INTERACTIVE Systems Corporation. XRemote is a registered trademark of Network Computing Devices, Inc. Oracle is a registered trademark of Oracle Corporation, Redwood City, California. Java is a trademark of Sun Microsystems, Inc. Compaq, Tandem, Integrity, NonStop, NetFlex, ProLiant, ROMPaq, ServerNet, SmartStart, registered in United States Patent and Trademark Office. Intel and Pentium are registered trademarks, and Pentium II Xeon and Pentium III Xeon are trademarks of Intel Corporation. SCO, Veritas, VxFS, and VxVA are registered trademarks of Veritas Software Corporation. Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies. All other brand and product names are or may be trademarks of, and are used to identify products or services of, their respective owners.

The SCO software that accompanies this publication is commercial computer software and, together with any related documentation, is subject to the restrictions on US Government use as set forth below. If this procurement is for a DOD agency, the following DFAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the US Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

If this procurement is for a civilian government agency, this FAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: This computer software is submitted with restricted rights under Government Contract No. _____ (and Subcontract No. _____, if appropriate). It may not be used, reproduced, or disclosed by the Government except as provided in paragraph (g)(3)(i) of FAR Clause 52.227-14 alt III or as otherwise expressly stated in the contract. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

Table of Contents

Lesson 1: Course Introduction	1-1
Introduction.....	1-3
Course goal and purpose.....	1-3
Course Lessons	1-5
Target audience	1-5
Student Materials	1-6
The Student Guide	1-6
SCO documentation.....	1-6
Typographical Conventions	1-7
Lesson 2: General Tips for Building a Cluster	2-1
Tower and Rack Installations.....	2-7
Cable Management	2-8
Rack Operations.....	2-9
Rack planning	2-9
Rack assembly and loading.....	2-10
Rack cabling	2-13
Site Planning	2-14
Lesson Summary.....	2-15
Answers to Exercise and Review Questions.....	2-16
Lesson 3: Assembling and Initializing Cluster Hardware	3-1
Assembly Overview.....	3-7
Hardware Assembly.....	3-10
Assembling racks	3-11
Assembling UPSes and power strips	3-13
Assembling shared storage	3-15
Assembling nodes	3-17
Assembling SAN, LAN, and KVM switches or hubs	3-19
Assembling monitors	3-21
Hardware Initialization	3-23
Initializing nodes.....	3-23
Initializing shared storage.....	3-25
Installing the split-brain avoidance cable	3-25
Testing the SAN.....	3-26
LAN testing.....	3-26
Lesson Summary.....	3-27
Lab: Assembling Cluster Hardware.....	3-28
Cross-node mirroring procedure.....	3-29
Fibre Channel/JBOD/RSO procedure.....	3-33
Answers to Review Questions	3-37

Lesson 4: Installing SCO UnixWare 7	4-1
Understanding Software Installation	4-6
Special HBA driver considerations.....	4-7
Licensing Information.....	4-8
UnixWare 7 licenses	4-8
NSC licenses	4-9
Other licenses.....	4-10
Cluster Virtual Host Name.....	4-11
Understanding Disk and Filesystem Configuration.....	4-12
The cluster <i>root</i> filesystem.....	4-13
Online System Upgrade (OSU)	4-14
Planning non- <i>root</i> filesystems	4-15
Vendor-specific partitions	4-15
Deferring Network Configuration	4-16
Package Selection	4-17
Using EFSes and PTFs.....	4-18
Program Temporary Fixes (PTFs).....	4-18
Lesson Summary.....	4-19
Lab: Installing UnixWare 7.1 on the Initial Root Node.....	4-20
Continuing with the installation.....	4-26
Post-installation tasks	4-26
Answers to Exercise and Review Questions.....	4-27
 Lesson 5: Installing NSC Software	 5-1
Understanding NSC Software Installation.....	5-6
Preparing to Install NSC on the Initial Root Node	5-7
Applying Program Technical Fixes (PTFs).....	5-7
Encapsulating the <i>root</i> filesystem.....	5-8
Preparing shared storage.....	5-9
Removing SDI ghost devices.....	5-11
Installing NSC on the Initial Root Node.....	5-12
Selecting optional packages.....	5-12
Configuring <i>root</i> filesystem failover.....	5-13
Configuring the CVIP and PIP	5-14
Configuring optional packages	5-15
Post-Installation Tasks	5-16
Lesson Summary.....	5-17
Lab: Installing NSC on the Initial Root Node	5-18
Overview.....	5-18
Procedure for CNM clusters	5-19
Procedure for shared-storage clusters	5-27
Relevant Commands and Files	5-34
Answers to Review Questions	5-35

Lesson 6: Installing the Failover Root Node	6-1
Understanding NSC Dependent Node Installation	6-6
Installing the Failover Root Node.....	6-7
Creating dependent node boot diskettes	6-7
Booting the failover root node using the DNBDs.....	6-8
Adding the node license.....	6-8
Checking the fdisk table for CNM	6-9
Configuring external storage for dependent nodes	6-10
Configuring split-brain avoidance for CNM	6-10
Configuring networking on the dependent node.....	6-11
Installing the Mirroring license.....	6-11
Lesson Summary.....	6-12
Lab: Creating Dependent Node Boot Diskettes.....	6-13
Overview.....	6-13
Procedure: Using make_floppy to create DNBDs.....	6-14
Lab: Installing the Failover Root Node	6-15
Overview.....	6-15
Procedure: Installing the failover root node for CNM clusters	6-16
Procedure: Installing the failover root node for shared-storage clusters	6-20
Lab: Testing Failover	6-23
Overview.....	6-23
Procedure: Simulating root node failure	6-24
Relevant Commands and Files	6-26
Answers to Review Questions	6-27

1

Course Introduction



Lesson 1

Course Introduction

1.1 Introduction

This ***UnixWare 7 NonStop Clusters Installation*** course is designed for individuals who need to gain a firm understanding of how to install the hardware and software for NonStop Clusters, hereafter referred to as NSC. The target audience includes technicians that will install and configure NSC. The course may also be useful to system administrators that will maintain NSC systems on a daily basis and programmers who will develop cluster-aware applications.

1.1.1 Course goal and purpose

At the conclusion of this course, students should be able to:

- Assemble a cluster in either tower or rack form
- Perform hardware setup and diagnostics on cluster components including nodes, storage devices, and Server Area Network (SAN)
- Install UnixWare 7.1 on the initial root node in a cluster
- Install NSC on the first node in a cluster
- Install the failover root node

AV 1-1

NonStop Cluster Hardware and Software Installation

- Introduction
- General Tips for Building a Cluster
- Assembling Cluster Hardware
- Installing UnixWare 7.1 on the Initial Root Node
- Installing NSC Software on the Initial Root Node
- Installing Dependent Nodes

Intro01 © The Santa Cruz Operation, Inc.

1.1.2 Course Lessons

To achieve the course goals, this course contains the following lessons:

- Introduction (this lesson) — course goals, topics, and typographical conventions
- General Tips for Building a Cluster — tips for assembling, shipping, and preparing sites for tower and rack-mount clusters
- Cluster Hardware Assembly — assembling the individual components into a cluster
- Installing UnixWare 7.1 on Node 1 — installing UnixWare 7.1 in preparation for turning the computer into node 1 of an NSC cluster
- Installing Dependent Nodes — installing NSC on the failover root node

1.1.3 Target audience

This class is aimed at people who need to become NSC installation technicians. NSC System administrators, support technicians, and developers may also find this course useful. Students are likely to have a wide variety of backgrounds in UNIX[®], UnixWare, and SCO OpenServer[™]. They must have the following prerequisite skills before attending this course:

- Passed the SCO UnixWare 7 ACE certification tests
- Passed the **UnixWare[®] 7 NonStop Clusters[®] Concepts** course test

1.2 Student Materials

This student guide is yours to keep and write notes in. Your instructor might have other reference materials in the classroom for your use while attending this class. Please do not take or write in these materials as they will be needed again for later classes.

1.2.1 The *Student Guide*

The *Student Guide* contains:

- Outline of presentation
- Note-taking space
- Copies of all transparencies

1.2.2 SCO documentation

This course refers to the following SCO printed documentation:

- *SCO UnixWare NonStop Clusters Release Notes*
- *SCO UnixWare NonStop Clusters Installation Guide*

This course also refers to the online documentation using **scohelp**(X1).

1.3 Typographical Conventions

This publication presents commands, filenames, keystrokes, and other special elements as shown here:

lp or lp(1)	Commands, device drivers, programs, and utilities (names, icons, or windows); the number in parentheses indicates the reference manual section in which the command, driver, program, or utility is documented
<i>/new/client.list</i>	Files, directories, desktops (names, icons, or windows), system, network, and printer names
root	User names
<i>arg</i> or <i>arg</i>	Placeholders (replace with appropriate name or value)
[<i>arg</i>]	Surrounds an <i>option</i> or <i>cmdarg</i> that is not mandatory
<i>arg ...</i>	Indicates multiple occurrences of the <i>option</i> or <i>cmdarg</i>
\$	Command-line prompt for commands executable from the Bourne shell by any user
#	Command-line prompt for commands only executable by the root user from the root account
F2	Keyboard keys
CTRL SPACE	Press CTRL and SPACE keys simultaneously
CTRL ALT DEL	Press all three keys, CTRL , ALT , and DEL , simultaneously
CTRL d	Press CTRL key. With CTRL still depressed, press D .
ALT 0240	Press ALT key. With ALT still depressed, press and release in sequence 0 , 2 , 4 , and 0 .
CTRL Q + i	Press CTRL key. With CTRL still depressed, press Q . Release both keys and press I .
CTRL Q + I	Press CTRL key. With CTRL still depressed, press Q . Release both keys and press SHIFT I .
Exit program?	System output (prompts, messages)

tar tv	User input, such as on a command line
“Description”	Field names or column headings (on screen or in database)
Cancel	Button names
Edit	Menu names
Copy	Menu items
File ⇨ Find ⇨ Text	Sequences of menus and menu items
open or open(2)	Library routines, system calls, kernel functions, C keywords; the number in parentheses indicates the reference manual section in which the file is documented
\$HOME	Environment or shell variables
SIGHUP	Named constants, signals, and kernel parameters
“adm3a”	Data values
<i>employees</i>	Database names and tables
buf	C program structures
b_b.errno	Structure members

2

General Tips for Building a Cluster



Lesson 2

General Tips for Building a Cluster

Introduction

Implementing clusters requires careful planning and preparation. This lesson provides tips for simplifying assembly, installation, and subsequent management and maintenance of the cluster.

Objectives

By the end of this lesson, you will be able to:

- Describe advantages of using rack or tower forms for cluster hardware
- Define rules for labeling cables for easy identification of their type, source, and destination
- Describe the guidelines for placing components within and between racks, taking into consideration power requirements, weight distribution, and heat dissipation
- Select a location for housing a cluster, including power requirements, weights, and sizes of cluster components

AV 2-1

General Tips for Building a Cluster

- Tower and Rack Installations
- Cabel Labeling
- Rack Loading
- Site Planning

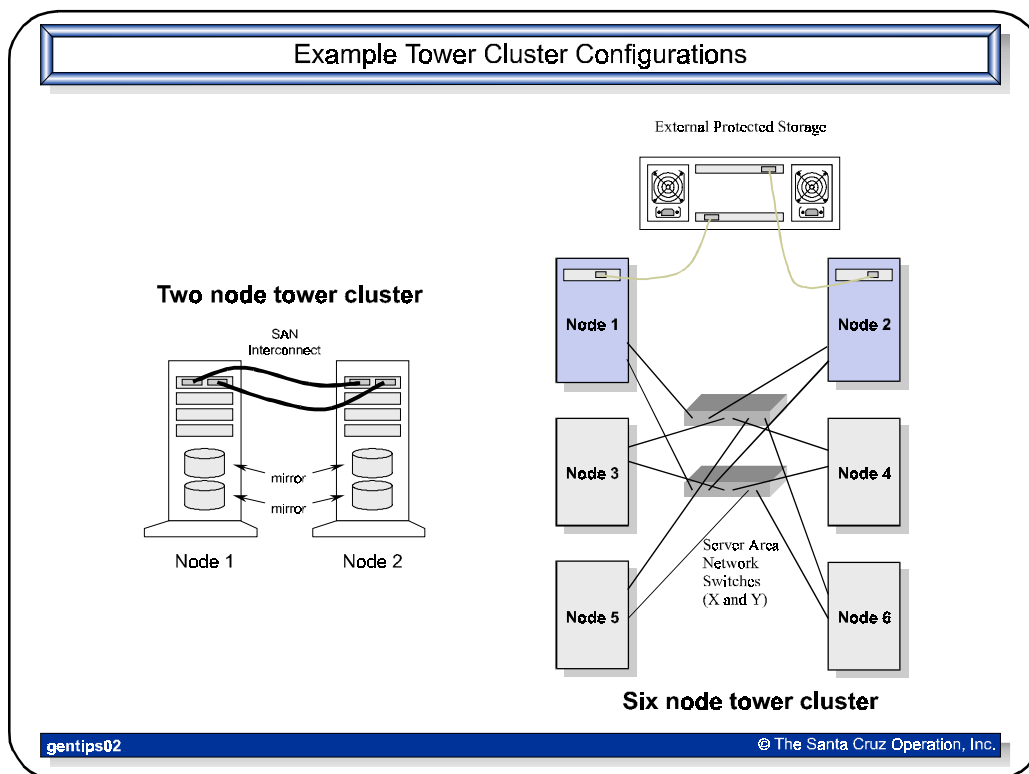
Gentips01 © The Santa Cruz Operation, Inc.

Topics

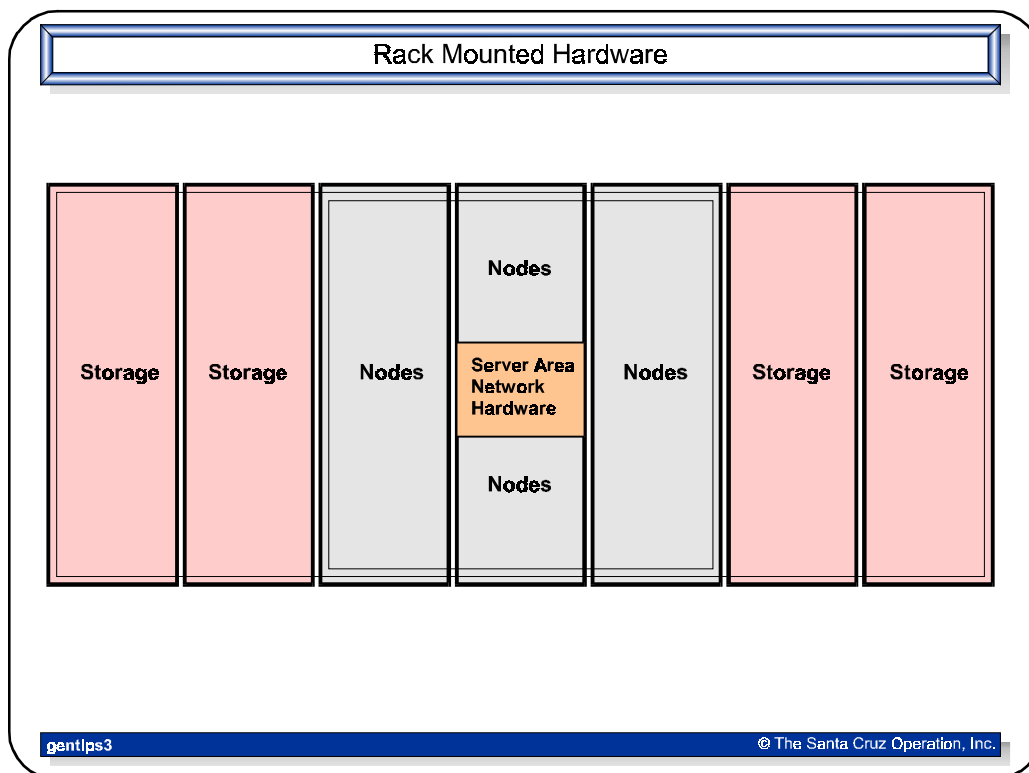
In order to meet these objectives, this lesson discusses:

- Tower and Rack Installations — deciding whether to use a rack or tower form
- Cable Management — placing components to meet cabling requirements, and how to label cables
- Rack Operations — placing components within a rack, lists issues for dealing with multi-rack clusters, and lists concerns you must be aware of when working with a rack
- Site Planning — issues that must be addressed before the cluster shows up at its final destination for final assembly and installation, including how to get the cluster to that location

AV 2-2



AV 2-3



2.1 Tower and Rack Installations

As discussed in the *UnixWare® 7 NonStop Clusters® Concepts* course, an NSC cluster contains many hardware components. Depending on the required capacity, a cluster includes some or all of this equipment:

- Two complete computers, known as nodes
- Protected storage housed in one or more external cabinets
- A Keyboard/Video/Mouse (KVM) switch and one keyboard, monitor, and mouse for the entire cluster; alternately, one keyboard, monitor, and mouse per node
- Two Server Area Network (SAN) switches
- Two or more UPSes and power strips
- Cables for all of the above

This equipment can either sit on the floor or be assembled into one or more racks, as illustrated in AV 2-2 and AV 2-3 on page 2-6.

There are advantages and disadvantages to each solution, as listed in the table below:

Table 1: Rack and Tower Advantages and Disadvantages

Issue	Tower	Rack
Expense	Less expensive	More expensive
Cables	Harder to manage	Easier to manage
Weight Distribution	Weight is more distributed	Weight is more concentrated
Floor Space	Need more	Need less
Cooling	Less of a concern	More of a concern
Tilting injuries	Not an issue	An important issue

2.2 Cable Management

Regardless of the format you choose, the many components require a phenomenal number of cables. Cable management is a critical to the smooth assembly and operation of your cluster. When a cable or component needs to be replaced, you must be able to quickly and easily locate and replace the affected cables. Two factors are critical:

- Adequate space—you must allow sufficient space to maneuver around cables
- Labels—you need to be able to identify both ends of the cable, and the node to which it belongs

Label each cable for easy identification of its type, source, and destination. Color-coded labels help you identify cables by node. These conventions are in common use for NSC cluster hardware, but any code you use consistently will work as well.

Table 2: Cable Labeling Standard

Node Number	Cable Tie Color
1	Pink
2	Orange
3	Yellow
4	Green
5	Purple

Add a white tie to all the cables that connect on the X fabric. For example, put a white tie on all the cables that attach to node 1. If you have more than two nodes, use other colored ties to indicate which cables connect to which nodes.

2.3 Rack Operations

Racks allow you to concentrate equipment in a single vertical space. It is critical that you understand how to work with rack equipment before you begin loading and cabling the rack.

Some vendors supply planning guides or software to assist you in designing racks to accommodate your specific needs. Check the *NSC Curriculum Hardware Supplement* for more information about the use of racks.

2.3.1 Rack planning

Before loading any equipment into a rack, plan how the rack will be loaded:

- Consider equipment and rack sizes. Racks are supplied in varying depths and heights to accommodate different types and amounts of equipment. The height of racks and equipment is measured in an industry-standard units known as a *U*. Each *U* is 1.75 inches, or approximately 4.49 centimeters.

For example, if you had three nodes of 5 *Us* each, a monitor of 5*Us*, and a keyboard/mouse tray and KVM switch that each take 1 *U*, then you need at least a 22*U* rack to hold this equipment.

- Make sure the rack is deep enough to hold your largest component. Be sure to consider the depth of the rails as well.
- Consider whether you need open or closed racks.
 - Open racks are less expensive
 - Closed racks provide better ventilation.
 - Closed racks require slightly more assembly, because they require that you install doors and sides.
- Consider whether to use convective- or forced-air-cooled racks.
 - Convective racks are less expensive.
 - Forced-air-cooled racks provide better cooling.

- If you have more equipment than will fit into a single rack, use multiple racks. In this case:
 - The racks should be of the same size whenever possible.
 - Locate components so that power cables do not cross racks.
 - Each rack needs its own UPS.
 - Racks should bolt together to prevent tipping and cable abrasion.
- Consider future expansion.
- Make sure your room is tall enough for the rack(s) you choose.
- Make sure the floor is strong enough to hold the weight you plan to put into the rack.
- If the rack is pre-loaded, make sure it fits through all doorways without tipping. If a rack is too tall to fit through a door, you must unload it before tipping it.
- Consider mechanisms provided to prevent tipping. These include:
 - Pull-out extension, or permanent, legs in front
 - Side legs or feet

2.3.2 Rack assembly and loading

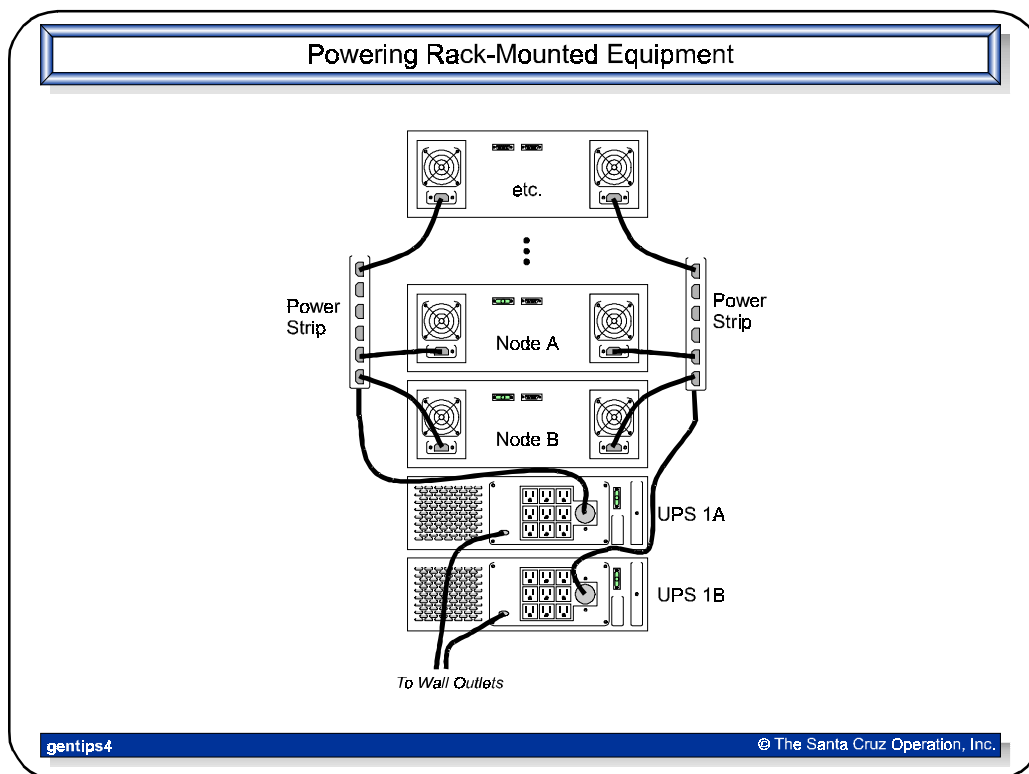
- Assemble the rack first. Always use the extension legs or stabilizer feet to prevent tipping, if they are provided with your rack.
- When loading the rack:
 - Assemble components as completely as possible, including rails, before you place the components into the rack
 - Keep the center of gravity as low as possible to avoid tipping the rack over:
 - Add components from the bottom up

- Put the heavier components closer to the bottom.

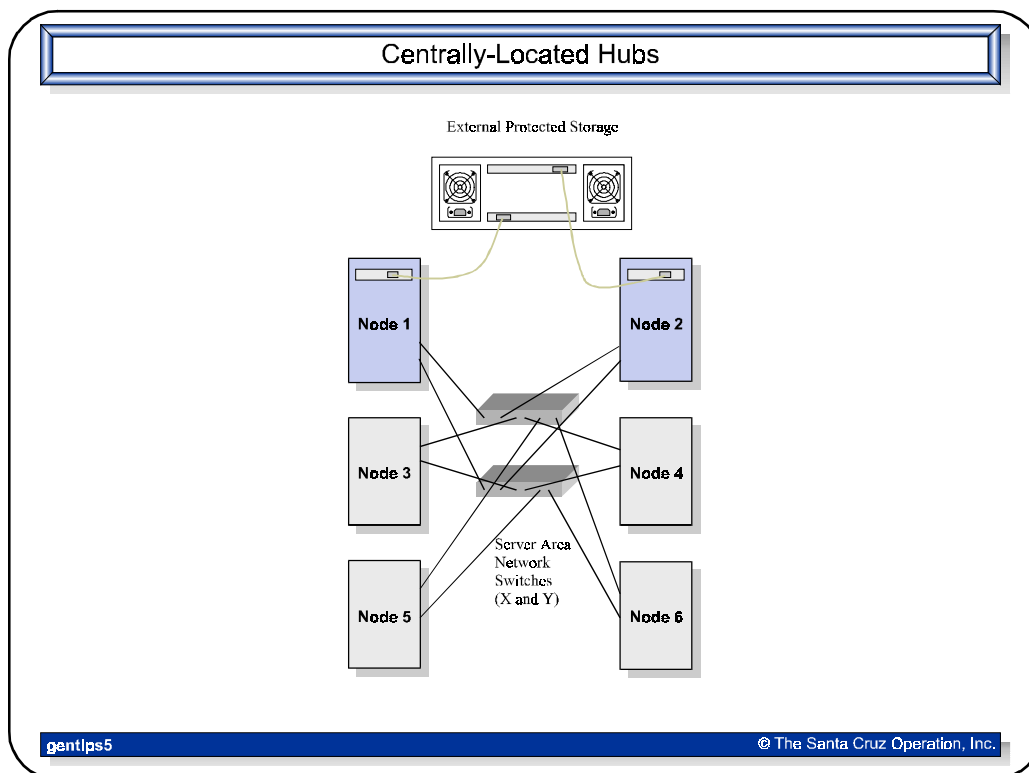
WARNING! A populated rack weighs enough to cause severe injury and possibly even death if it falls onto someone!

- Always place monitors at the top of a rack because they generate the most heat.
- If you must choose between weight and heat considerations, give weight highest priority, as you can also increase cooling capacity.
- Use at least two people to place heavier components in the rack. Because components can weigh hundreds of pounds, you need one person on each side of a component to lift the heavy components to the rack. If possible, have a third person guide the component into the rack and onto its rails. Always lift with your legs, never with your back.
- Do not extend more than one component at a time when removing equipment from rack.

AV 2-4



AV 2-5



2.3.3 Rack cabling

After all of the components are placed in the rack and the cables have all been labeled, you are ready to cable the components together.

The main issues when cabling are:

- Never run power cables between two racks. This creates a potential hazard and violates codes in many locales.
- Attach cables to the shortest components (depth-wise) first.
- Use cable-strain-relief ties where necessary. This also helps keep cables neatly bundled together.
- Use power strips designed for your rack between UPSes and other components. Place one power strip on each side of the cabinet, mid-height. Run power cables from the power strip to each power supply on the same side. This is shown in AV 2-4 on page 2-12.
- As shown in AV 2-5 on page 2-12, if using SAN switches or hubs, locate them centrally to the nodes.
- Be sure circuits are not overloaded if the rack is equipped with power strips.

WARNING! Do not overload the AC supply circuit that provides power to the rack to avoid the risk of fire, equipment damage, or personal injury.

2.4 Site Planning

Site planning is critical when deciding on a suitable location for the cluster equipment. Necessary attributes for a cluster site include (some of which have already been discussed):

- | | |
|---------------|--|
| Floor loading | Make sure the floor can handle the aggregate concentrated weight of the cluster equipment. This is especially important for clusters in racks, sites using raised floors, and sites using non-commercial class buildings. |
| Entry path | <p>Make sure the cluster components can be moved from the delivery vehicle to the cluster site.</p> <ul style="list-style-type: none">● Can racks make it under doorways without being tipped? Again, if this is not the case, you must empty the rack before it is tipped. In addition, make sure doors are wide enough to fit the widest piece of equipment.● Are the hallways wide and tall enough?● Is the floor is smooth and level with no gaps? |
| Power | Make sure the circuits bringing power to the cluster have enough capacity. |
| Cabling | Make sure there is sufficient space around and between the hardware to allow for cable management. As discussed, the cabling requirements for clusters are substantial. It is critical that you leave an adequate amount of space to manage the cables in an organized fashion that allows you to quickly and easily swap cables and components. |
| Cooling | Make sure the cooling system for the location is adequate to handle the heat generated by the cluster hardware. |

2.5 Lesson Summary

Some cluster configurations require a substantial amount of hardware. You might need to look for alternatives to traditional tower cabinet computers and storage devices. This lesson introduced rack-mount configurations and provided suggestions to help you plan for locating and assembling rack hardware.

Review questions

1. What are some advantages of using rack forms for cluster hardware?
2. What are the guidelines for labeling cables?
3. Describe the guidelines for placing components within and between racks.
4. What are some considerations for selecting a location for your cluster?

2.6 Answers to Exercise and Review Questions

This section contains the answers to review questions presented in this lesson.

Review Question Answers

This section contains the answers to the review questions on page 2-13.

1. Some advantages of using rack forms for cluster hardware are that they require less floor space and better facilitate cable management.
2. Label cables for easy identification of their type, source, and destination. Use color codes to indicate node membership. Add a white tie to uniquely identify the X fabric SAN cables.

3. Consider these guidelines for placing components within racks:

Assemble all components first, including rails

Place heaviest components at the bottom for stability

Place monitors at the top, for heat dissipation

Use two people to install heavy components

4. When planning a location for your cluster, consider power and cooling requirements, weight, cable space, and sizes of components.

3

Assembling and Initializing Cluster Hardware



Lesson 3

Assembling and Initializing Cluster Hardware

Introduction

You must follow a prescribed sequence of events in order to efficiently assemble a cluster and facilitate a successful software installation. Deviations from this established sequence will likely prove time-consuming and increase the likelihood that the software installation will not succeed.

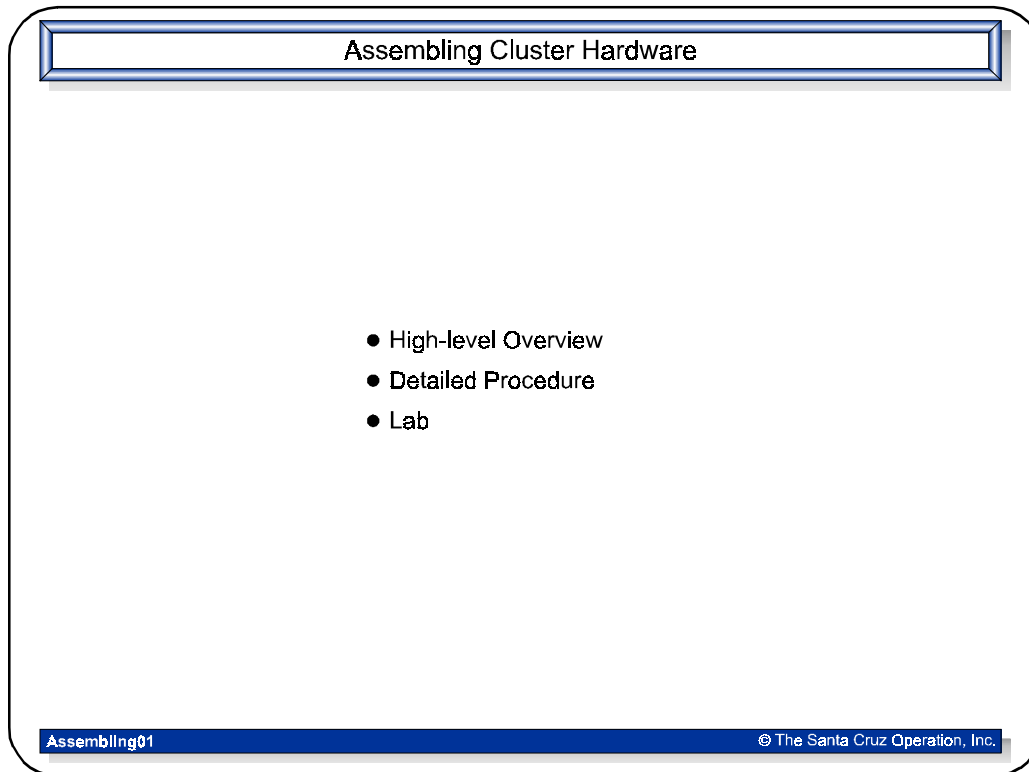
This lesson details the rules for assembling a cluster.

Objectives

By the end of this lesson, you will be able to:

- Describe the general cluster hardware assembly process
- Assemble cluster hardware
- Assemble racks and rack-mounted equipment
- Initialize hardware to prepare for software installation

AV 3-1



Topics

In order to meet these objectives, this lesson discusses:

- High-level Overview — an overview of the assembly process
- Hardware Assembly — physically assembling these components of a cluster:
 - Racks
 - UPSes and power strips
 - Nodes and controllers — the various components and controller cards that need to be present in a node, whether using CNM or protected storage
 - Switches and hubs — placing and cabling switches and hubs for LAN, SAN, and keyboard/video/mouse (KVM) switches
 - Shared storage — placing and cabling the various shared storage methods
- Hardware Initialization — running low-level hardware initialization on the components of a cluster. The subsections for the components are:
 - Nodes — running CMOS setup on various kinds of nodes, including how to insure that attributes of certain controllers are set to specific values.
 - SAN — running diagnostics on the SAN fabrics
 - Shared storage — creating hardware RAID arrays on certain shared storage solutions.

The last part of this lesson is a lab where you assemble the hardware for either a 2-node cluster using cross-node mirroring, or a cluster with more than 2 nodes using shared storage.



3.1 Assembly Overview

This procedure will help you assemble your hardware in an orderly fashion. It has been tested and proven effective and complete.

If you have expertise assembling hardware, and racks in particular, you might choose to perform some tasks in a different order. While it is not absolutely required that you strictly follow these steps in the order given, you are strongly recommended to use this procedure because it is known to work. If you are not using racks, disregard rack-specific tasks.

1. Assemble racks, if used. See Section 3.2.1, “Assembling racks,” on page 3-11.
2. Prepare UPSes. Place them in the bottom of the racks, if used. See Section 3.2.2, “Assembling UPSes and power strips,” on page 3-13.
3. Assemble shared storage units, if used. The type of shared storage used dictates any required assembly. In general:
 - Place the rails on the storage units.
 - Load the storage unit into the rack.

Remember that racks must be loaded with the heaviest components toward the bottom. If the weight of the storage units, including disks, is more than some of your nodes, place them in the rack before those nodes. Otherwise, place them in the rack after those nodes.

- Place disks in the storage units after they are placed in the rack.
- See Section 3.2.3, “Assembling shared storage,” on page 3-15 for additional information.

4. Assemble each node:

- Insert any cards that are not pre-installed. For example, many systems do not come with NICs or HBAs.
- Insert SAN cards. Some SAN technologies requires you to insert SAN cards into particular slots.
- Place the rails on the nodes and place the nodes in the rack, heaviest nodes toward the bottom.

For more information, see Section 3.2.4, “Assembling nodes,” on page 3-17.

5. Install rack-mounted keyboard trays and place the tray in the rack at a comfortable height.
6. Assemble SAN switches or hubs. If using racks, attach rails and place them into the rack.
7. Assemble LAN switches or hubs. If using racks, attach rails and place them into the rack.
8. Place the KVM switch in the center of the nodes. If using a rack, attach rails to the KVM switch and install it in the rack.
9. Place the monitor in an appropriate location. If using a rack, install the monitor tray and place it in the rack.
10. Cable the components together, starting with the shortest components. For example, LAN, SAN, and KVM switches are short. Cable these components first.

Label each cable as described in Section 2.2, “Cable Management,” on page 2-8.

Note If you must install a longer component on top of a shorter component in a rack, you might want to attach cables to these shorter components before placing the longer component on top of it. This helps prevent abrasions on your fingers when cabling equipment in the rack.

11. Power on components in this order:

- a.) Monitors
- b.) Switches or hubs
- c.) External storage cabinets
- d.) Nodes

Make sure each component has completed its firmware initialization before powering on the next component. For example, verify that all storage cabinets have completed their firmware initialization before powering on any nodes.

12. Run any necessary firmware initialization on each node or storage cabinet. The order in which you must perform these steps is vendor-dependent. See Section 1.8, “Initializing Nodes,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement***.
13. Run any necessary SAN verification software to verify that the SAN cards, cables, and switches or hubs are working properly.
14. Run any necessary LAN verification software to verify that the SAN cards, cables, and switches or hubs are working properly.

Once you have installed and initialized the hardware, you are ready to install the operating system as described in Lesson 4: “Installing SCO UnixWare 7.”

The next sections of this lesson provide greater detail and include references to vendor-specific information and procedures found in the ***NSC Curriculum Hardware Supplement***. The lab in Section 3.5, “Lab: Assembling Cluster Hardware,” on page 3-28 provides a complete procedure to use when assembling the classroom cluster hardware.

3.2 Hardware Assembly

Regardless of the vendor you chose for your cluster hardware, all have similar assembly procedures. The next sections describe common assembly tasks for cluster hardware, organized to reflect the order in which they are assembled and installed.

Tasks and information specific to each vendor's hardware components are described in the applicable section of the *NSC Curriculum Hardware Supplement*.

Use this checklist to insure that all components have been assembled and cabled before you continue to Section 3.3, "Hardware Initialization," on page 3-23.

Table 3: Assembly Checklist

Component Assembled	Component Cabled	Component Name
		Racks
		UPSes
		Power strips
		Nodes
		Shared storage
		SAN
		LAN
		KVM switch
		Monitor

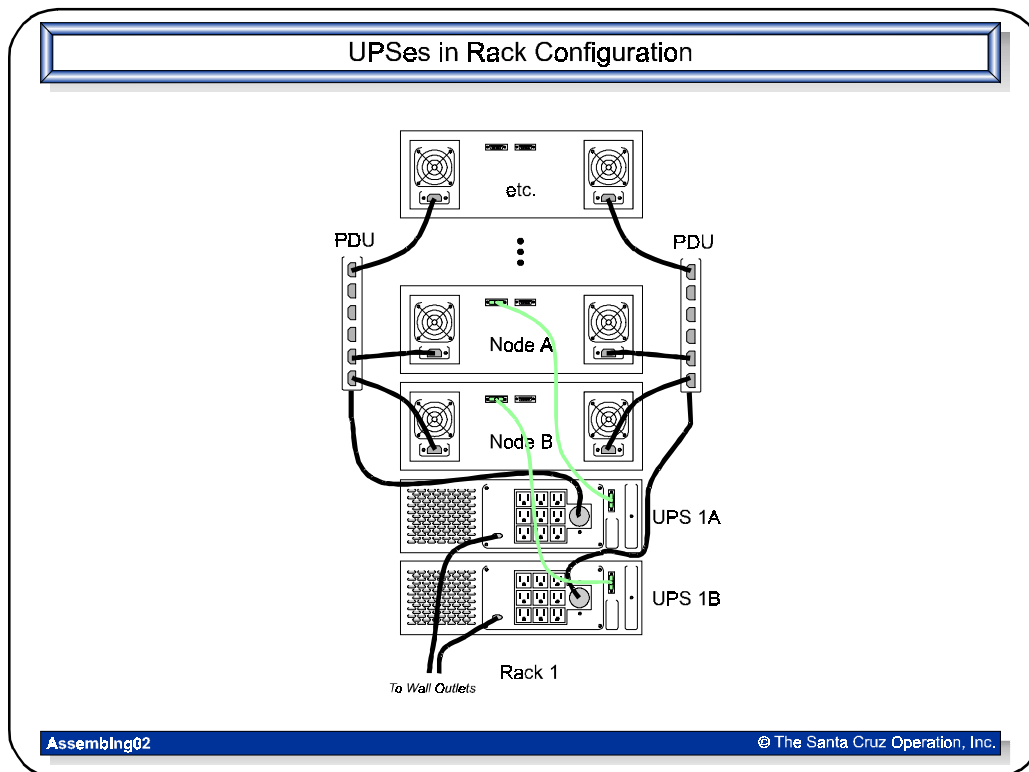
3.2.1 Assembling racks

If you are unfamiliar with racks, see Section 2.3, “Rack Operations,” on page 2-9 before continuing. To assemble a rack:

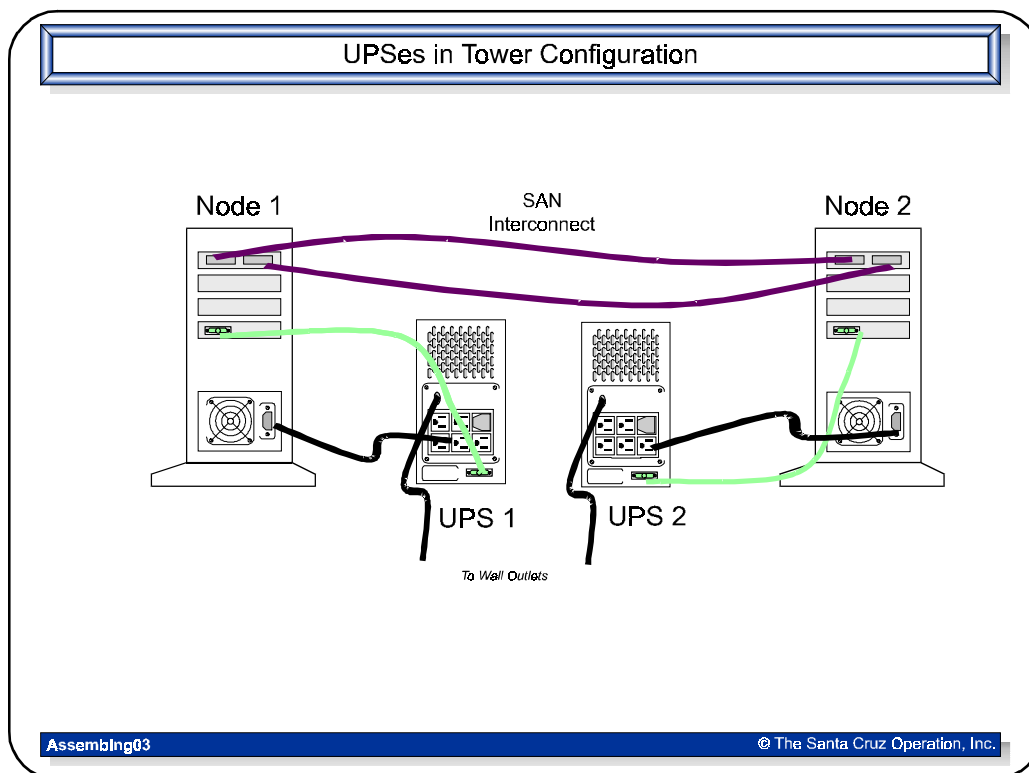
1. Perform a site survey to find an appropriate location.
2. Select appropriate rack model(s).
3. Assemble the rack(s). When using closed racks, leave the doors and sides off until the components have been installed.
4. Load the equipment.
5. Ground the rack.

For details on assembling cluster-certified racks, see Section 1.4, “Assembling Racks,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*.

AV 3-2



AV 3-3



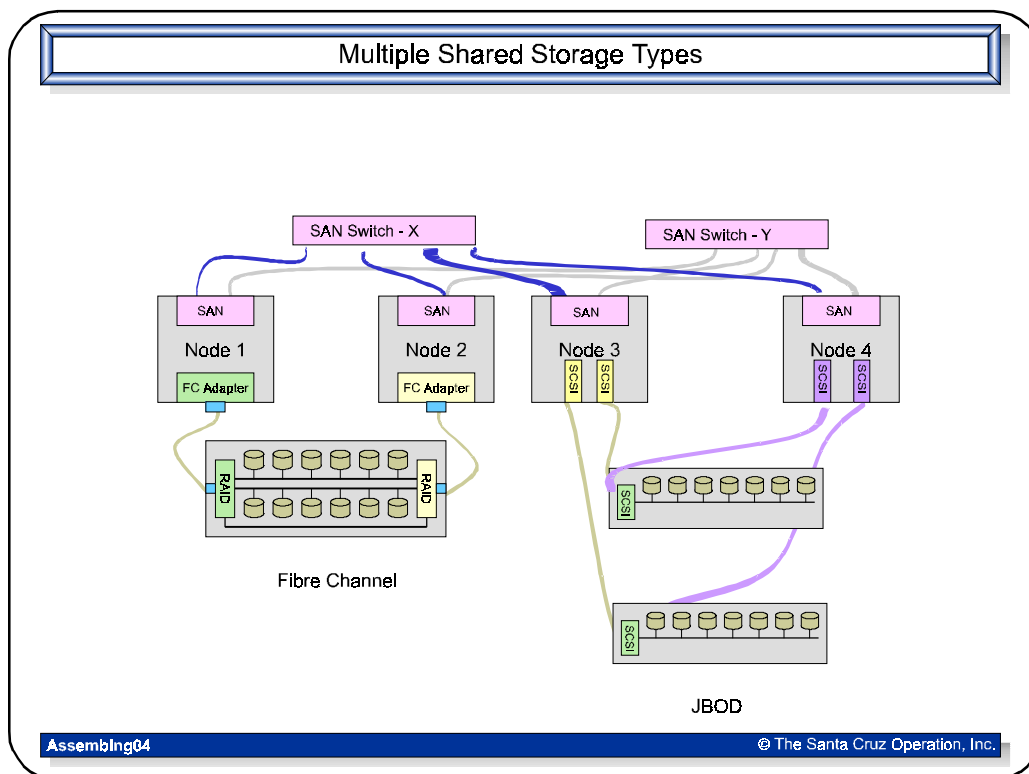
3.2.2 Assembling UPSes and power strips

Although UPSes vary widely, they have certain things in common. For each UPS device:

1. Follow the manufacturer's specifications for connecting the battery.
2. Attach rails and place them in the rack, if used.
3. Run power cords from each UPS to the power source.
4. Plug the power strips into each UPS.

When you are finished configuring your UPSes and power strips, your configuration should look similar to those shown in AV 3-2 and AV 3-3 on page 3-12.

AV 3-4



3.2.3 Assembling shared storage

There are several shared storage solutions you can use with your cluster. Further, you can employ multiple solutions within a single cluster. For example, as shown in AV 3-4 on page 3-14, you may have potential root nodes connected to a fibre solution and two other nodes connected to a JBOD solution.

The supported shared storage options and their required hardware are discussed in the applicable sections of the Section 1.6, “Assembling Shared Storage,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*.

To assemble shared storage:

1. Assemble the box(es).
2. Insert boxes into a rack, if used.
3. Populate the boxes with disks.
4. Cable the storage boxes to the nodes after nodes have been assembled.



3.2.4 Assembling nodes

When configuring a node:

1. Place HBAs in the node and connect them to appropriate disks.
 - You must have one HBA connected to one local disk that is used to boot the UnixWare 7 operating system on this node. This includes *boot*, *swap*, and *dump* slices.
 - If this node is a potential root node and you are using CNM, this will also be the disk used for the *root* filesystem.
 - If this node is protecting shared external storage, add one HBA for each external shared storage cabinet. There is an exception to this rule when using fibre storage with multiple fibre storage boxes attached to the node through fibre hubs.
2. Place SAN cards in the node. In many types of nodes, the bus and slot you choose for your SAN cards is critical. These are some common considerations for installing SAN cards:
 - Use the highest priority PCI slot
 - Use the fastest PCI bus
 - Do not put slower-speed PCI cards in the same bus with the SAN card; all cards then operate at the lowest-common speed

For example, Compaq ServerNet SAN cards have these requirements:

- Place the SAN card in the highest-priority PCI slot.
- Set the IRQ 15

If using other SAN technologies, install at least two SAN NICs per node.

See Section 1.5, “Assembling Nodes,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement*** for details specific to your vendor.

3. Place LAN cards in the node. As with host machines, if the node is on multiple subnets, it must have one LAN NIC per subnet.
4. Attach rails to the nodes and place them in the rack, heaviest nodes toward the bottom.

AV 3-5

Compaq 8-port K/M/M Switch



Assembling05

© The Santa Cruz Operation, Inc.

3.2.5 Assembling SAN, LAN, and KVM switches or hubs

To assemble switches:

1. Assemble the SAN switches.

Recall that most SAN technologies require you to use a switch to connect more than two nodes in a cluster. Some switches require assembly and configuration. For information on assembling SAN switches, see Section 1.7, “Assembling SAN Switches,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*.

2. Assemble LAN switches.

See your manufacturer’s documentation for information about installing and configuring LAN switches and hubs.

3. Cable the keyboard/video/mouse (KVM) switch, if used.

KVM switches are a single box, requiring no assembly. AV 3-5 on page 3-18 illustrates how to run cables between the nodes, the KVM, and the actual keyboard, mouse, and monitor in use.

Note Current KVM technology does not allow for redundant KVM switches. As such, the KVM can be viewed as a single point of failure for the cluster.

However, failure of the KVM can only cause nodes to fail if sufficient voltage is generated by the KVM failure. The damage to the node’s video, keyboard, or mouse circuitry caused by that voltage would have to be sufficient to subsequently damage the node itself.

This probability of this happening is so low that the KVM switch is not considered a potential single point of failure. If a KVM unit fails, it can be replaced without powering down the entire cluster.

4. Insert switches into the rack if used.

AV 3-6



3.2.6 Assembling monitors

To assemble a monitor:

1. Attach speakers to the monitor, if included.
2. Cable the speakers to the sound card in one of the nodes.
3. Sound card drivers are not yet clusterized and therefore represent a single point of failure, as do the speakers.
4. Attach the monitor base.
5. Place the monitor on a rack-mounted shelf, if used. As shown in AV 3-6 on page 3-20, monitors are usually placed at the top of a rack to allow heat to dissipate.
6. Attach power to the monitor.



3.3 Hardware Initialization

After you have assembled and cabled all of the hardware components, you must initialize certain components before they can be used in a cluster. This section discusses which components may need initialization, and how to perform that initialization. Where appropriate, you are referred to the *NSC Curriculum Hardware Supplement* for instructions specific to a particular make or model of a component.

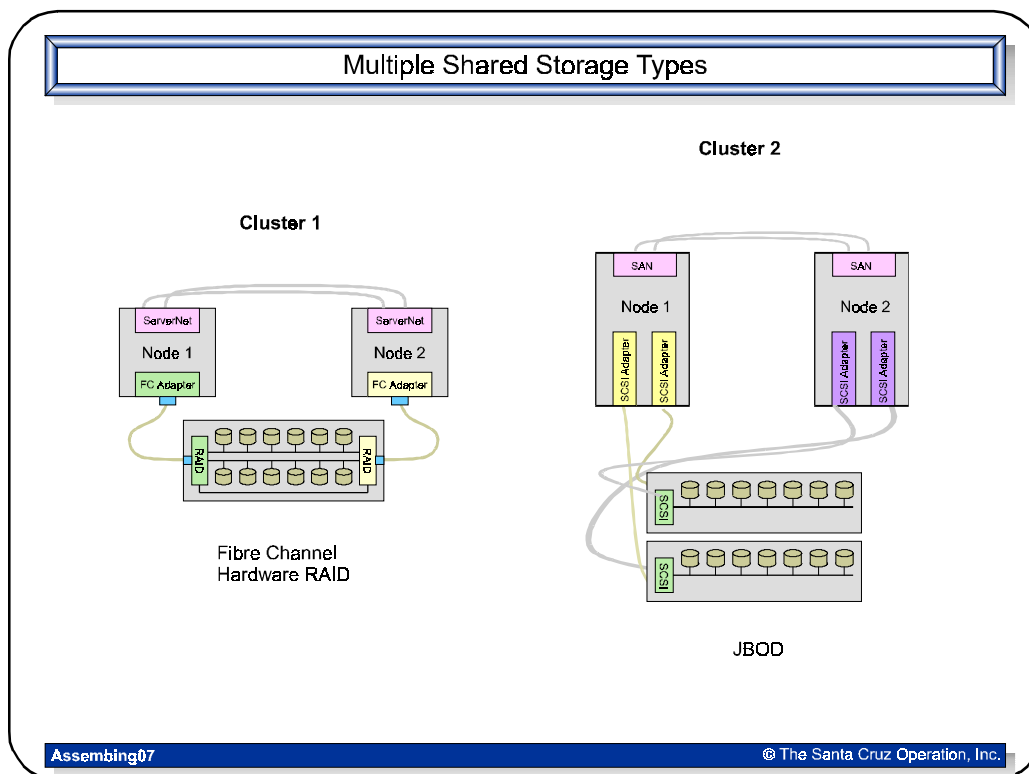
3.3.1 Initializing nodes

Whether you need to initialize a node depends on these factors:

Node manufacturer	For example, all Compaq nodes must be initialized with SmartStart to insure proper ordering of the storage controllers.
SAN type	Some nodes require that you use a specific interrupt number and type for the SAN card. For example, Compaq ServerNet I cards must be set to interrupt at IRQ15 with a trigger-level-type interrupt signal. On some nodes, you control this through PCI Plug and Play BIOS configuration common to many computers. On others, you must use proprietary setup software.

See Section 1.8, “Initializing Nodes,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement* for information on setting up a particular make and model of node.

AV 3-7



3.3.2 Initializing shared storage

Most protected storage schemes use RAID mirroring, performed either by hardware or software. AV 3-7 on page 3-24 illustrates two examples:

- Cluster 1 uses a sophisticated fibre storage solution with built-in RAID. The box contains dual SCSI busses, controllers, and power supplies, so that a single storage box can be used without a single point of failure. In this scenario, the disks on one physical bus are mirrored with disks on the other bus. The box then presents each of the mirrors to the system as single disks.
- Cluster 2 uses a simple JBOD storage solution with no hardware RAID assistance. Since there is no redundancy in either SCSI busses or controllers, two storage boxes are required. In this case, the NSC software must establish a mirror between the disks on box 1 and those on box 2.

Hardware initialization of shared storage solutions is only necessary where hardware RAID is being used, as in the Cluster 1 example.

See Section 1.9, “Initializing Shared Storage,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement*** for details on setting up specific shared storage solutions that utilize hardware RAID.

3.3.3 Installing the split-brain avoidance cable

Recall that a split-condition can exist when both nodes try to act as active root nodes. Certain SAN implementations that do not use SAN switches can experience this condition.

You can avoid this by configuring an alternate communication path using a null-modem cable between serial ports on two nodes. This prevents your *root* filesystem from being destroyed when the SAN communication is lost.

You are recommended to configure and test split-brain avoidance whenever you are installing a CNM cluster.

Attach a serial null-modem cable to a serial port on each node.

3.3.4 Testing the SAN

Before attempting to install the operating system software on a cluster, you must verify that the SAN components are functioning properly.

If using a vendor-specific SAN technology, such as Compaq ServerNet, see Section 1.11, “Testing the SAN Interconnect,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement*** for information on testing you SAN components.

If you are using LAN technology to implement the SAN, test these components just as you would other LAN components. Verify:

- Card-to-card connectivity
- Switch or hub connectivity

3.3.5 LAN testing

Test your LAN NICs and hubs using software utilities provided by your LAN hardware vendor. Your instructor will provide any necessary test or diagnostic tools for your classroom LAN hardware.

3.4 Lesson Summary

This lesson, in addition to the *NSC Curriculum Hardware Supplement* presented information on how to assemble the hardware needed to implement an SCO NSC cluster, including:

- A step-by-step overview of the process
- Assembly and cabling of cluster components
- Initialization of cluster components

Review questions

1. If using a rack, which component would you assemble and install into the rack first, UPSes or nodes? Why?
2. Which component is unnecessary in a two-node cluster, but must be included on a similar cluster with more than two nodes?
3. When installing a cluster with more than two nodes, which protected storage methods can be used?
4. How does the installation of a cluster with CNM differ from one with external storage?
5. If you are using shared storage for your *root* filesystem, can all of your disks be in the external cabinets?

3.5 Lab: Assembling Cluster Hardware

This lab guides you through assembling a cluster with two or more nodes using one of the supported protected storage techniques. The instructor will provide you with a complete list of equipment being used in your classroom and the type of cluster you will assemble.

Perform one of the labs below, based on the protected storage solution you are using. During the lab, you must assemble any components not already assembled (for example, the rack). You must also initialize and test all applicable hardware components.

The labs are finished when you can show your implementation, including passing applicable hardware tests, to the instructor. Check your progress with the instructor at key phases of the lab, such as completion of SAN hardware initialization.

3.5.1 Cross-node mirroring procedure

In this lab, you will assemble a two-node cluster using cross-node mirroring between nodes 1 and 2 to protect the *root* filesystem.

This procedure is based on the instructions presented in Section 3.1, “Assembly Overview,” on page 3-7. Where appropriate, you will be referred to the ***NSC Curriculum Hardware Supplement*** for detailed information about the hardware in your classroom.

If your classroom does not use all of the hardware devices mentioned in these steps, such as racks, hubs, or switches, disregard those tasks.

- Step 1. Assemble the rack according to the manufacturer’s instructions. See Section 1.4, “Assembling Racks,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement***.
- Step 2. Create a loading plan for the rack. See Section 1.4, “Assembling Racks,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement*** for details about your manufacturer’s rack loading requirements.
- Step 3. Assemble SAN, LAN, and KVM cables according to the color-coding guide in Section 2.2, “Cable Management,” on page 2-8.
- Step 4. Assemble UPSes:
 - a. Connect their batteries, if necessary.
 - b. Place them in the rack, one pair in each rack.
 - c. If not using a rack, place the UPSes on the ground in a position central to where the rest of the components will reside.
 - d. Plug the UPSes into the wall outlets.
 - e. Mount the power strips on the side of each side of the rack(s).
 - f. Plug the power strips into the UPSes.

- Step 5. Assemble each node. For this CNM-based cluster, each node must have:
- One disk attached to an HBA. This disk will be used for *boot*, *swap*, *dump*, and *root* filesystems.
 - One or two SAN cards, depending on the SAN technology used. Remember, if using ServerNet I, this card must be placed in the highest priority PCI slot.
 - One LAN card

See Section 1.5, “Assembling Nodes,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement* for more detail about the nodes used in your classroom.

If using racks, attach rails to the nodes and place them in the rack according to the loading plan created in Step 1.

Note If installing SAN, LAN, or KVM switches between nodes, assemble but do not install nodes above the switches until the switches themselves have been installed. This will prevent injury to your fingers.

If using towers, place the nodes symmetrically on either side of the UPSes.

- Step 6. Assemble SAN switches if your cluster has more than two nodes. See Section 3.2.5, “Assembling SAN, LAN, and KVM switches or hubs,” on page 3-19.

If using racks, attach rails to the switches and place them in the rack.

Note After installing the SAN switch, but before installing other devices above it, connect the SAN cables to the back of the SAN switch. Then install other devices above the SAN switch. Attaching cables to the switch at this time makes it easier to see and access the connection points.

If using a tower system, place the switches as centrally as possible to the nodes. A common location for the switches in tower clusters is on top of the nodes themselves.

- Step 7. Assemble LAN switches or hubs, if used.
- Place them in the rack, if used. Attach cables before adding any devices above them in the rack.
 - If using a tower system, place the LAN switches or hubs on top of the SAN switches or hubs. Attach cables.

- Step 8. Assemble KVM switches, if used.
- If using a rack, attach rails to the KVM switch and install it in the rack. Attach cables to the KVM switch before adding any devices above it in the rack.
 - If using a tower system, place the KVM switch on top of the network switches or hubs.

- Step 9. Connect the keyboard.
- Install the keyboard tray, if using a rack. Position it at a comfortable height to allow the average person to type while standing in front of the cluster.

Note Install remaining nodes within the rack above the keyboard tray and below the monitor mentioned next.
--

If you are not using a rack, place the keyboard on top of one of the nodes.

- Step 10. Connect the monitor.
- If using a rack and a monitor tray within the rack, place the monitor tray at the top of the rack. For racks less than full height, you can place the monitor on top of the rack itself.
- Step 11. Connect the LAN cables from the LAN hub or switch to the LAN cards in each node. Use the color-coded labels as a guide.
- Step 12. Connect the SAN cables. See Section 1.7, “Assembling SAN Switches,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement* for instructions for your classroom equipment.

Use the color coding to decide which cable to plug into which node.

- Step 13. Connect the KVM cables from the KVM switch to the keyboard, video, and mouse connectors on each node. Use the color coding to decide which cable to plug into which node.
- Step 14. Install the split-brain avoidance cable, if using a CNM cluster.
- Step 15. Power on the monitor.
- Step 16. Power on any switches or hubs.
- Step 17. Power on node 1.
- Step 18. Initialize the node. See Section 1.8, “Initializing Nodes,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement* for instructions specific to your classroom hardware.
- Step 19. Repeat Steps 15 and 16 for each node in the cluster.
- Step 20. Perform LAN diagnostics using software passed out by your instructor.
- Step 21. Perform SAN diagnostics. See Section 1.11, “Testing the SAN Interconnect,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement* for instructions specific to the SAN hardware in your classroom. Your instructor will provide any necessary software.

Your cluster hardware is now ready to install UnixWare 7 operating system software.

3.5.2 Fibre Channel/JBOD/RSO procedure

In this lab, you will assemble a two (or more)-node cluster using fibre channel between nodes 1 and 2 to protect the *root* filesystem.

These instructions are based on those presented in Section 3.1, “Assembly Overview,” on page 3-7. Where appropriate, you will be referred to the ***NSC Curriculum Hardware Supplement*** for a detailed list of hardware make and model specific instructions.

- Step 1. Assemble the rack according to the manufacturer’s instructions. See Section 1.4, “Assembling Racks,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement***.
- Step 2. Create a loading plan for the rack. See Section 3.2.1, “Assembling racks,” on page 3-11.
- Step 3. Label the SAN, LAN, and KVM cables according to the color-coding guide in Section 2.2, “Cable Management,” on page 2-8.
- Step 4. Assemble UPSes:
- Connect their batteries, if necessary.
 - Place them in the rack, one pair in each rack.
 - If not using a rack, place the UPSes on the ground in a position central to where the rest of the components will reside.
 - Plug the UPSes into the wall outlets
 - Mount the power strips on the side of each side of the rack(s)
 - Plug the power strips into the UPSes
- Step 5. Assemble the shared storage unit(s). See Section 1.6, “Assembling Shared Storage,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement***.

If you are using a tower system, locate the shared storage box(es) wherever it is convenient, so long as the fibre cables can reach between the storage box and its nodes.

If you are using a rack, place the shared storage box(es) in the rack according to your plan. However, do not populate the box(es) until they have been placed in the rack.

Step 6. Assemble each node. Since you are using external storage units for protected storage, each node must have:

- One internal disk attached to an HBA. This disk will be used for boot, swap, dump. It will not contain the root filesystem.
- One or more HBAs to attach to the storage box(es).
- One or two SAN cards, depending on the SAN technology used. Remember, if using Compaq ServerNet I, this card must be placed in the PCI slot with the highest priority.
- One LAN card

See Section 1.5, “Assembling Nodes,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement* for details on assembling your specific nodes.

If using racks, attach rails to the nodes and place them in the rack according to the loading plan created in Step 1.

Note If installing SAN, LAN, or KVM switches between nodes, assemble but do not install nodes above the switches until the switches themselves have been installed. This will prevent injury to your fingers.

If using towers, place the nodes symmetrically on either side of the UPSes.

Step 7. Assemble SAN switches if your cluster has more than two nodes. See Section 1.7, “Assembling SAN Switches,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*.

If using racks, attach rails to the switches and place them in the rack.

Note After installing the SAN switch, but before installing other devices above it, connect the SAN cables to the back of the SAN switch. Then install other devices above the SAN switch. Attaching cables to the switch at this time makes it easier to see and access the connection points.

If using a tower system, place the switches as centrally as possible to the nodes. A common location for the switches in tower clusters is on top of the nodes themselves.

- Step 8. Assemble LAN switches or hubs, if used.
- Place them in the rack, if used. Attach cables before adding any devices above them in the rack.
 - If using a tower system, place the LAN switches or hubs on top of the SAN switches or hubs. Attach cables.

- Step 9. Assemble KVM switches, if used.
- If using a rack, attach rails to the KVM switch and install it in the rack. Attach cables to the KVM switch before adding any devices above it in the rack.
 - If using a tower system, place the KVM switch on top of the network switches or hubs.

- Step 10. Connect the keyboard.
- Install the keyboard tray, if using a rack. Position it at a comfortable height to allow the average person to type while standing in front of the cluster.

Note Install remaining nodes within the rack above the keyboard tray and below the monitor mentioned next.
--

If you are not using a rack, place the keyboard on top of one of the nodes.

- Step 11. Connect the monitor.
- If using a rack and a monitor tray within the rack, place the monitor tray at the top of the rack. For racks less than full height, you can place the monitor on top of the rack itself.

- Step 12. Connect the LAN cables from the LAN hub or switch to the LAN cards in each node. Use the color-coded labels as a guide.

- Step 13. Connect the SAN cables.
- Use the color coding to decide which cable to plug into which node.

- Step 14. Connect the KVM cables from the KVM switch to the keyboard, video, and mouse connectors on each node. Use the color coding to decide which cable to plug into which node.

- Step 15. Cable the external shared storage unit, if used. See Section 1.6, “Assembling Shared Storage,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement***.
- Step 16. Power on the monitor.
- Step 17. Power on any switches or hubs.
- Step 18. Power on node 1.
- Step 19. Initialize the node. See Section 1.8, “Initializing Nodes,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement*** for instructions specific to your classroom hardware.
- Step 20. Repeat Steps 17 and 18 for each node in the cluster.
- Step 21. Initialize the disks and controllers in the shared storage units. See Section 1.9, “Initializing Shared Storage,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement*** for instructions specific to your classroom hardware.

Note Depending on the Fibre shared storage solution used, you might need to perform additional initialization tasks on node 2. If so, these steps are included in the applicable section of the <i>NSC Curriculum Hardware Supplement</i>.

- Step 22. Perform LAN diagnostics using software passed out by your instructor.
- Step 23. Perform SAN diagnostics. See Section 1.11, “Testing the SAN Interconnect,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement***. Your instructor will provide any necessary software.

Your cluster hardware is now ready to install UnixWare 7 operating system software.

3.6 Answers to Review Questions

This section contains the answers to review questions presented in this lesson.

Review Question Answers

This section contains the answers to the review questions on page 3-27.

1. According to the procedure on Section 3.1, “Assembly Overview,” on page 3-7 and the explanation given in Section 3.2.2, “Assembling UPSes and power strips,” on page 3-13, you should assemble and install UPSes before nodes because UPSes weigh more per U than nodes, and such items must be placed lower in the rack to keep the rack’s center of gravity low.
2. According to Section 1.7, “Assembling SAN Switches,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*, SAN switches are only required for clusters with more than two nodes. As such, in a cluster with only two nodes, said switch need not be included.
3. According to Section 1.2, “Supported Protected Storage Methods,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*, the protected storage method you use has nothing to do with the number of nodes in a cluster. As such, all supported protected storage methods are available on the cluster in question.
4. According to Section 3.2.4, “Assembling nodes,” on page 3-17, CNM systems only need HBAs for their internal disks, which will serve as both boot and root disks. For non-CNM systems, you must have an additional HBA per shared storage box connected to the node.
5. According to Section 3.2.4, “Assembling nodes,” on page 3-17, you must have a boot disk in addition to the external root disk for systems not using CNM.



4

Installing SCO UnixWare 7



Lesson 4

Installing SCO UnixWare 7

Introduction

Once you have assembled and initialized your cluster hardware, install UnixWare 7 on the initial root node.

To set up the proper environment for the subsequent installation of the NSC software, you must carefully navigate through the UnixWare 7 installation procedure, taking into account the NSC-specific considerations described in this lesson.

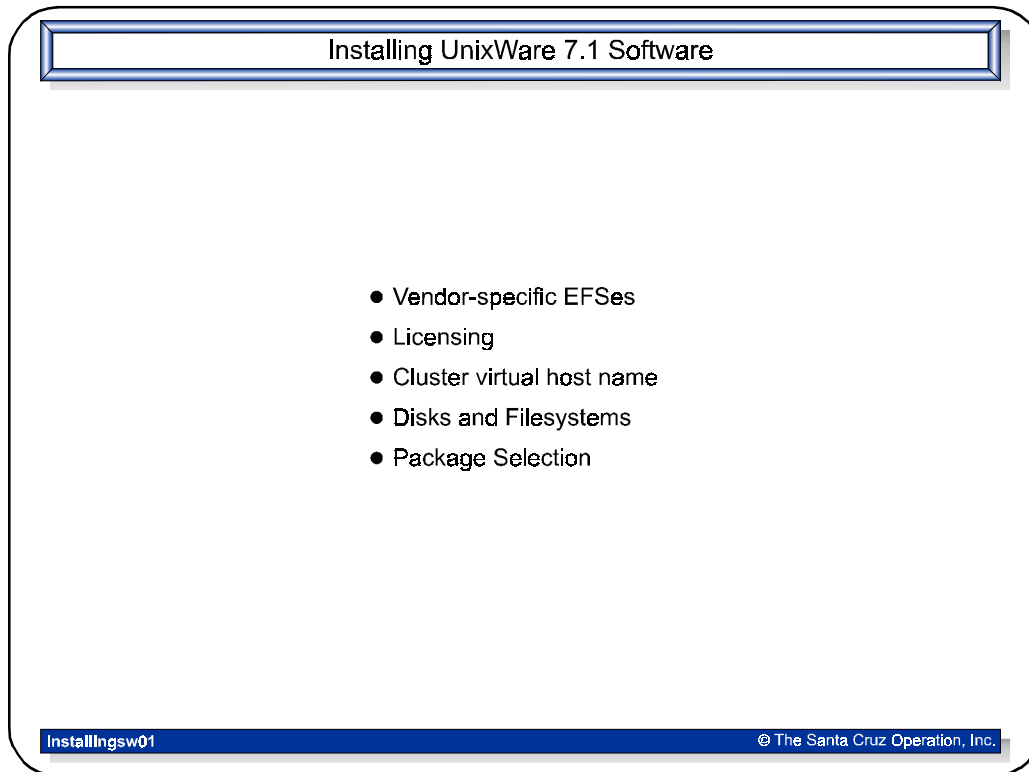
The lesson focuses on the unique NSC properties that influence the UnixWare 7.1 installation process.

Objectives

By the end of this lesson, you will be able to:

- Describe the installation process
- Prepare for the selections you must make during UnixWare 7 installation to support NSC functionality
- Perform any necessary pre-install tasks required by your cluster hardware vendor
- Install UnixWare 7 on the initial root node
- Perform post-installation tasks to prepare for NSC software installation

AV 4-1



Topics

In order to meet these objectives, this lesson discusses:

- Vendor-supplied EFSes — checking for hardware-specific drivers needed for installation
- Licensing — selecting appropriate UnixWare license editions for NSC
- Cluster Virtual Host Name — selecting appropriate node and cluster host names
- Disk and Filesystems — special considerations for NSC supported functionality
- Package Selection — deciding which packages are required, and which should be removed

4.1 Understanding Software Installation

After your hardware is assembled and initialized, your cluster software is assembled and installed in this sequence. This lesson covers Steps 1, 2, and 3.

1. Assemble operating system software. Create or obtain vendor-supplied EFS or HBA diskettes, if needed.
2. Install UnixWare 7.
3. Install vendor-supplied EFSes.
4. Prepare the initial root node for NSC installation
5. Install NSC on the initial root node
6. Install the failover root node
7. Install depending nodes.

Successful installation of a cluster requires careful preparation. Because installation of all nodes depends on the success of the installation and preparation of the initial root node, you must understand the configuration choices you will make during the UnixWare 7 installation process that are needed to support the NSC software.

This lesson describes steps 1, 2, and 3 of the software installation procedure. Each section gives you guidance about the selections you need to make during the UnixWare 7 installation process that affect the NSC software preparation and installation steps.

The lab in Section 4.10, “Lab: Installing UnixWare 7.1 on the Initial Root Node,” on page 4-20, provides the detailed procedure. Lesson 5: “Installing NSC Software” and Lesson 6: “Installing the Failover Root Node” describe the procedures for installing the cluster software.

4.2 Special HBA driver considerations

Hardware vendors supply HBA drivers for devices that are not included in the UnixWare 7.1 Media Kit. These are supplied in a number of ways:

- HBA diskettes supplied by the hardware manufacturer
- CD-ROMs provided by hardware vendors with HBA diskette images
- EFS diskettes sets which include one or more HBA diskettes
- Downloadable HBA diskette images

Be sure to read Section 1.12, “Vendor-Supplied Software,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement*** and use only those drivers specified by your hardware vendor. Be careful to install HBA disks in the order specified.

4.3 Licensing Information

There are two types of software licenses needed for a cluster:

- One UnixWare 7.1 license per node
- One NonStop Clusters license

Be sure you understand the license requirements and have the appropriate license information from the Certification of License Authority (COLA) card before beginning your UnixWare 7 installation.

You might also have other licenses for optional products or services.

4.3.1 UnixWare 7 licenses

Each node in the cluster must have a UnixWare 7 license. SCO provides licenses separately from the installation media. The UnixWare 7 Media Kit contains the software installation CD-ROM disks. A separate license package includes a COLA for each node with this information:

- License number
- License code
- License data

Each node must be installed with the same license UnixWare 7 *Edition*. Editions are collections of software targeted to a specific purpose. You must install each node with an Edition that includes a Veritas Online Data Manager (ODM) or Disk Mirroring license.

Alternately, you can add an optional ODM or Disk Mirroring license to an Edition that does not include mirroring or ODM.

ODM and Disk Mirroring licenses are included with these editions:

- Enterprise
- Data Center

An optional ODM or Disk Mirroring license can be purchased and used with these editions:

- Base
- Business
- Department

You cannot use the Free Unix, or the 60 Day Evaluation Editions because they do not support ODM.

If you install UnixWare 7 with a license Edition that does not include ODM or Disk Mirroring, you can add one of these licenses after UnixWare 7 installation using the **License Manager**. However, this *must* be added *before* NSC software is installed.

4.3.2 NSC licenses

In addition to the UnixWare 7 licenses required for each node, a separate NSC license is needed for the cluster software. At a minimum, a NonStop Clusters 2-Server Base License is required. If you add nodes, you must purchase NonStop Clusters Additional Server Licenses.

Note Do not enter your NonStop Clusters license during the UnixWare 7 operating system installation procedure.

You can install the NonStop Clusters software without a license for evaluation purposes. After 60 days, you must purchase and install NonStop Cluster licenses or remove the software from each node.

4.3.3 Other licenses

You might require additional licenses to support your specific cluster configuration. Some licenses apply to specific nodes, and some to the entire cluster.

For example, if you need to support 200 simultaneous users accessing the cluster, you must purchase additional Network User licenses. User licenses are applied to the cluster.

You might also require licenses for multiprocessor nodes, or to support nodes with large amounts of memory. CPU and RAM licenses apply to a single node.

All additional licenses should be installed using the **License Manager** after NSC is installed and all nodes are clusterized.

Consult the UnixWare 7 Installation Roadmap document supplied with the UnixWare 7 Media Kit to determine which licenses you need for your cluster.

4.4 Cluster Virtual Host Name

During UnixWare 7 installation, you are prompted to enter a node name. For a typical installation of a standalone server, this is the name by which the system is known locally and on the network. The commands **uname(1)** and **hostname(1)** return this name.

When installing the initial root cluster node, the node name you enter becomes the cluster virtual host name; that is, the collective name for all the nodes in the cluster. Therefore, you must supply the name that you want to call the cluster when you install UnixWare 7 on the initial root node.

Later, when you install the NSC software, you are prompted to enter the node name for the server itself.

4.5 Understanding Disk and Filesystem Configuration

NSC has unique requirements when configuring the filesystems. Be sure you understand these cluster-specific considerations before you start installing UnixWare 7 on the initial root node.

- Set */tmp*, */var*, and */var/tmp* to *off*. They are set to *memfs* by default, which only exist in the node's RAM and is therefore not available to other nodes. When set to *off*, these are created as directories within the *root* filesystem, and so they are *clusterized* and available to all nodes.
- Increase */stand* to a minimum of 40 MB. Additional copies of the kernel are stored here.
- Set up your *root* filesystem as recommended in Section 4.5.1, "The cluster root filesystem," on page 4-13.
- Do not create user filesystems such as */home*. Add these after installing NSC. Note that if you create user filesystems, they are not automatically *protected*; that is, mirrored.
- Add the values in the size column and verify that the space allocated is less than or equal to the size of the disk.

This table shows an example of the filesystem configuration table for an initial root node.

Table 4: Recommended Filesystem Setup

Name	Type	Size	Disk
/	vxfs	User configuration; minimum 2000, recommended 3000. Allow room for OSU	1
/stand	bfs	40	1
/dev/swap	slice	Increase calculated value	1
/dev/dump	slice	Calculated based on installed memory	1
/home	off	Do not create until after installing NSC	-
/home2	off	Do not create until after installing NSC	-
/var	off	-	-
/tmp	off	-	-
/var/tmp	off	-	-
/dev/volprivate	slice	Calculated; do not change	1
ALTS TABLE	slice	Calculated; do not change	1

4.5.1 The cluster *root* filesystem

The initial root node is the system on which you install UnixWare 7 and then perform the first installation of SCO NonStop Clusters software. Carefully consider how to configure your *root* filesystem before starting installation of the initial root node by familiarizing yourself with the unique aspects of the *root* filesystem in a cluster.

The *root* filesystem in a cluster is controlled by the initial root node, sometimes called the initial installation node. In the event of root node failure, the failover root node assumes control of the root filesystem.

- In a shared disk configuration, the *root* filesystem resides on an external disk shared by the root and failover root nodes.
- In a cross-node mirroring configuration, the *root* filesystem resides on the internal disk of the root node and is mirrored on the internal disk of the failover node.

The rest of this section provides guidelines for planning the size of the root filesystem to best balance your system requirements with the configuration of your cluster.

4.5.1.1 Planning *root* filesystem size

During installation, you are required to specify the size of the *root* filesystem. Calculate the size carefully. A very large *root* filesystem can significantly extend installation time. When determining the *root* filesystem size, different considerations apply to shared-disk clusters and others to those using cross-node mirroring.

Shared disk The NSC installation initially sets up the *root* filesystem on the internal drive of the initial installation node. Later during the installation, it moves the *root* filesystem to the external drive. Therefore, you cannot specify a *root* filesystem size that is larger than the internal disk of the initial installation node.

Balance operational needs and the time it takes for the *root* filesystem to be moved. Here are some examples:

- On systems where the initial installation node has a 9G internal drive, the *root* filesystem size should be a minimum of 2G.
- An average *root* filesystem size is around 3G.
- The *root* filesystem on node 1 must be smaller than the size of the drive in the external storage unit that will host the *root* filesystem.

If you plan to configure your system for future Online Software Upgrade (OSU) service, the drive that will eventually contain your *root* filesystem:

- Must be at least twice the size of the *root* filesystem
- Must also have room for any non-*root* filesystems you plan to store there

For example, if you have a 9G drive in your initial installation node and a 9G drive in your external array drive, and you plan to use OSU, the largest *root* filesystem you could specify would be limited to 4.5G in size; however, that would leave no room for any non-*root* filesystems on the external drive.

CNM The internal disk of the initial root node and that of the failover root node must each hold the *root* filesystem (and a spare copy of that filesystem if you plan to configure for OSU usage), as well as any non-*root* filesystems.

The *root* filesystem on node 1 must be smaller than the size of the internal drive in node 2.

4.5.2 Online System Upgrade (OSU)

During installation, you are prompted to set aside space for future Online System Upgrade (OSU) use. The OSU facility allows you to update UnixWare and NSC software while the cluster is fully-functional.

When you plan current filesystem space on the drive containing the *root* filesystem, include enough space for future OSU use. You must have room on the drive containing the *root* filesystem for a duplicate of that *root* filesystem.

Note Not all versions of NSC support OSU for all types of protected storage methods. Check the <i>SCO NonStop Clusters Release Notes</i>.
--

If you have additional non-*root* filesystems on the drive containing the *root* filesystem:

- Subtract the non-*root* filesystem sizes from the total space on the drive.
- The remaining space can be used for the *root* filesystem and the OSU set-aside space.
- Make sure your *root* filesystem is half the remaining space or less.

For example, if you have an 8GB drive, and non-*root* filesystems take up 1GB of the drive, you have 7GB of remaining space to use for the *root* filesystems. To enable the OSU feature, you should make the *root* filesystem no larger than 3.5GB.

4.5.3 Planning non-*root* filesystems

You can use any unused space on cross-node mirroring root volumes, or on shared-root external and internal drives for non-*root* files systems, such as */home* or */home2*. How you create non-*root* filesystems depends on the type of protected shared storage you use.

To simplify installation, do not configure any non-*root* filesystems until after UnixWare 7 and NSC have been installed and tested.

4.5.4 Vendor-specific partitions

Some vendors use a portion of the boot disk to store hardware- or vendor-specific utilities. Be sure to leave special partitions intact during the UnixWare 7 installation.

4.6 Deferring Network Configuration

Network configuration of the initial root node must be done in a particular sequence that is unique to NSC—after UnixWare 7 and supplements, but before NSC software.

Network configuration is deferred until after UnixWare 7.1 is running because there might be updated drivers that are installed later. For example, your hardware vendor might supply an EFS with NIC drivers that must be installed after UnixWare 7 is running.

The NIC must be configured before NSC because the IP address you assign becomes the CVIP during NSC installation. Likewise, the host name you configure for the first network interface becomes the cluster host name.

4.7 Package Selection

It is crucial that you select an NSC-compatible set of packages when installing UnixWare 7 on the initial root cluster node. Deselecting some packages can cause the NSC installation to fail. Conversely, installing some packages will cause problems on an NSC cluster node. It is strongly recommended that you adhere strictly to these guidelines:

- Select these packages from the UnixWare 7 Installation CD-ROM (1 of 3):
 - OS Multiprocessor Support—required *even* if you do not have multiprocessor server
 - Kernel Debugger
- Remove these packages from the UnixWare 7 Updates CD-ROM (2 of 3), if they are pre-selected:
 - Multipath IO (MPIO)
 - SCO NetWare
 - NIS
- Make sure the ODM package selection is FULL
- Defer installing the third UnixWare 7 CD-ROM until you have successfully completed the operating system installation.

Detailed instructions for making package selections are provided in Section 4.10, “Lab: Installing UnixWare 7.1 on the Initial Root Node,” on page 4-20.

4.8 Using EFSes and PTFs

Your hardware vendor provides Enhanced Functionality Supplements (EFSes) with the UnixWare 7.1 drivers you need to support your cluster hardware. These can be provided in several ways. For example:

- Floppy disk images created from a CD-ROM
- Floppy disk images downloaded from a suppliers web site

You might have a set of EFSes required to initially install UnixWare 7.1, and another to provide enhanced NSC-specific features. Refer to Section 1.12, “Vendor-Supplied Software,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*. Be sure to note whether vendor supplements are required to be installed in a particular order.

During the installation procedure provided in the Lab, you are directed to sections in the *NSC Curriculum Hardware Supplement* that contain specific information for the hardware used in your classroom.

4.8.1 Program Temporary Fixes (PTFs)

SCO and your hardware vendor provide Program Temporary Fixes (PTFs) which are applicable to specific versions of UnixWare 7 and NonStop Clusters.

Use only those PTFs specified in the installation procedures in Lesson 5: “Installing NSC Software.” Non-clusterized PTFs are not supported on UnixWare 7 systems running NSC.

Check the *SCO NonStop Clusters Release Notes* for guidance about using PTFs.

4.9 Lesson Summary

This lesson provided guidelines for installing UnixWare 7 on your initial root cluster node. The selections that impact the subsequent NSC installation process were highlighted and where necessary, specific directions were provided to help you navigate through critical sections of the installation process.

Review questions

1. In which order are HBA disks usually inserted during UnixWare 7 installation?
2. How much space should you allocate for the root filesystem if you have a 4 GB UNIX partition on the internal primary disk drive and want to use the OSU option?
3. Can you use Department Edition UnixWare 7 licenses on your cluster nodes?
4. If you create a */home* filesystem during UnixWare 7 installation, will it automatically be mirrored to the failover node when NSC is installed?

4.10 Lab: Installing UnixWare 7.1 on the Initial Root Node

This lab guides you through the UnixWare 7.1 installation for the initial root node of the classroom cluster to which you are assigned. The instructor will provide you with any configuration information you need.

The steps in the procedure are general with references to hardware- or vendor-specific sections of the *NSC Curriculum Hardware Supplement*. Be sure to read through the procedure and the referenced hardware-related material before beginning the lab.

The lab is finished when you can show your functional UnixWare 7.1 root node is running properly and connected to the classroom network. Check your progress with the instructor at key phases of the lab.

This lab assumes you have performed all hardware assembly and initialization labs in this course.

Procedure

Use this procedure to install UnixWare 7. The hardware- or vendor-specific information you need is included in the NSC Hardware Supplement. When necessary, these steps refer you to the application section of the Supplement.

Recall that UnixWare 7 installation is divided into several stages:

Setup	The installation prompts you with questions and records your answers. Your replies determine which software components are installed, how system hardware is configured, and the disk division layout.
Software loading	The installation program installs and configures your system using the parameters defined during setup. This stage is not interactive.
Initial hard disk boot	Configuring and testing your mouse.
Post-install setup	Adding EFSes and application software.

To install UnixWare 7 on the initial node:

- Step 1. Create any HBA disks required for your classroom hardware. See Section 1.12, “Vendor-Supplied Software,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement***.
- Step 2. Power off all nodes and external storage units using the procedure recommended by your hardware vendor. See Section 1.10, “Powering Down Cluster Hardware,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement***.
- Step 3. Insert the first UnixWare 7 installation diskette into the primary diskette drive of the initial root node.
- Step 4. Power on the node.

You see your system’s boot messages, and then SCO logo. It might take several minutes to get to this point.
- Step 5. Set the language, if prompted.
- Step 6. Insert the second UnixWare 7 installation diskette when prompted.

After software is extracted from this diskette, the **Welcome** window appears. Press **F10** to continue.

Step 7. Select the zone, locale, keyboard.

Step 8. Enter the UnixWare 7 license number, code, and data for node 1. See Section 4.3, “Licensing Information,” on page 4-8, for help understanding the license requirements for a cluster node.

Note that the NSC license is *not* entered here.

Step 9. Install Host Bus Adapter disk(s) when prompted.

- If you have a vendor-supplied HBA disk(s), read Section 1.12, “Vendor-Supplied Software,” in your vendor’s chapter in the **NSC Curriculum Hardware Supplement** to make sure you insert the HBA disk(s) in the required order.
- You must insert the SCO HBA diskette in addition to any vendor-supplied HBA diskette(s). You typically install vendor-supplied HBAs first.
- If you do not have a vendor-specific HBA disk, insert the SCO UnixWare 7 HBA disk.

Select “Proceed with Installation” and press **F10** after you have entered all necessary HBA diskettes.

Step 10. Select “Do not enter the DCU (autoconfigure drivers)” when prompted.

Step 11. Enter the name of the cluster when prompted for the system node name.

Note Be sure you enter a name by which the entire cluster will be known. You will enter an actual system node name for this server later when installing NSC.

See Section 4.4, “Cluster Virtual Host Name,” on page 4-11 for more information.

Step 12. Select “Install from CD-ROM” and press **F10**.

Step 13. Insert the first UnixWare 7 CD-ROM (1 of 3) when prompted and press **F10**.

Step 14. Select “Customize partitions” in the “Action” field for disk number 1. Press **F10** to continue.

The partition table for disk 1 is displayed. You might already have a partition on your primary disk. .

Note Some hardware vendors create a disk partition to store configuration utilities. Do not remove this partition

- Step 15. Add a UNIX partition. Scroll to an unused partition number and setting the disk options. Use **→** to move between fields and **F2** to select these values:
- Choose “UNIX System” in the “Type” field.
 - Choose “Active” in the “Status” field.
 - Use the remainder of the primary disk for the new UNIX partition.
 - Press **F10** to continue.
- Step 16. Configure additional internal disks to use the entire disk for UNIX, if present in your node.
- Step 17. Select “Customize filesystems and slices” and press **F10**.
- Make these changes:
- Set */tmp*, */var*, and */var/tmp* to off.
 - Increase */stand* to at least 40 MB.
 - Increase */swap*. It is automatically decreased when you turn off */tmp*, */var*, and */var/tmp*.
 - Set your *root* filesystem size according to the guidelines in Section 4.5.1, “The cluster root filesystem,” on page 4-13.
 - Do not create user filesystems such as */home* but leave space for adding them later, after installing NSC.
 - Add the values in the size column and verify that the space allocated is less than or equal to the size of the disk.
 - When finished, press **F10**.
- Step 18. Select “Customize disk options” and press **F10**.

- Step 19. Mark all choices “no” *except* “Install a new boot sector.” If not already set, change “Install a new boot sector” to “yes” and press **F10**.
- Step 20. Select “Customize Installation of Packages” when prompted to choose a system profile and press **F10**

WARNING! Do not deviate from the package selections documented here. This can cause your NSC installation to fail later.

- Step 21. Select “Core System Services” and press **F10**.
- Step 22. Select “Extended OS Utilities” and press **F2**.
- Step 23. Select “Select individual packages” and press **ENTER**. Select “Kernel Debugger.”
- Step 24. Page down and select “OS Multiprocessor Support (OSMP).”

You MUST install OSMP even if you have a single-processor system.

Note Do NOT select MPIO.

- Step 25. Press **ENTER** when finished with Extended OS Utilities selection.
- Step 26. Select “Networking Services” and press **F10**.
- Step 27. De-select “NIS” and press **F2**.
- Step 28. Page down and select “SCO Netware.” Press **F2** and select the option “Select no packages (OFF).” Press **ENTER**.
- Step 29. Verify that the “Online Virtual Disk Volume Administration” selection is set to “FULL” in the **Select/De-select Services** window.
- If it is not, you do not have an appropriate UnixWare 7 license. You can continue installation, but must add an ODM or Mirroring license after UnixWare 7 is install, but before NSC is installed.
- Step 30. Page down and select “SCO Netware.” Press **F2** and select the option “Select no packages (OFF).” Press **ENTER**.
- Step 31. Press **F10** when finished making these selections.

- Step 32. Press  in the **Current Selections** window to continue.
- Step 33. Select “Intel MP Specification” as the Platform Support Module *even if you do not have a multiprocessor system*. Press .
- Step 34. Select “Defer Network Configuration” and press .
- Step 35. Set the date and time and press .
- Step 36. Select a security level and press .
- Step 37. Enter the system owner information and press .
- Step 38. Set the **root** password and press .
- Step 39. Accept the license agreement and press .
- Step 40. Press  to start installing the software.
- Software loading is the longest stage of the installation and can proceed unattended. You can track the status of the installation by viewing the progress indicator.
- Step 41. Reinsert your UnixWare 7 HBA diskette when prompted.
- Step 42. Reinsert any vendor-supplied HBA diskette(s).
- Step 43. Remove your floppy disk and CD-ROM when the installation completes.
- Step 44. Press  to shut down and restart your system.

4.10.1 Continuing with the installation

After the restart of your system and several non-interactive configurations take place, you are prompted to configure your mouse. Follow these steps to configure your mouse.

- Step 1. Select the type of mouse. Press **F10**.
- Step 2. Confirm or change the mouse parameters. Press **F10**.
- Step 3. Press **ENTER** to begin the fifteen (15) second mouse test.
- Step 4. Click on your right mouse button. Press **ENTER** to continue.
- Step 5. Insert the UnixWare Upgrades CD-ROM (2 of 3) and press **F10**.
- Step 6. Scroll to “Enhanced Event Logging System” and press **SPACE** to deselect it. Press **ENTER** to continue.

Software is now loaded from this CD-ROM. You can track the status of the installation by viewing the progress indicator.

Press **F10** when complete, and remove the CD-ROM.

- Step 7. Press **F8** to defer installing the Optional Services CD-ROM (3 of 3). Wait while the kernel is rebuilt and the system reboots.

Your initial root node is now running UnixWare 7.

4.10.2 Post-installation tasks

After UnixWare 7 has been installed on your initial root node, and before you install NSC software:

- Step 1. Add an ODM or Mirror license, if the license Edition used during UnixWare 7 installation did not include one of these licenses.
- Step 2. Install any vendor-supplied EFSes or applications. See Section 1.12, “Vendor-Supplied Software,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement***.
- Step 3. Install all application software. It is unlikely you will do this in the classroom.
- Step 4. Reboot your system.

4.11 Answers to Exercise and Review Questions

This section contains the answers to review questions presented in this lesson.

Review Question Answers

This section contains the answers to the review questions on page 4-19.

1. EFSes supplied by hardware manufacturers are usually installed first. Then the HBA diskette supplied by SCO with the UnixWare 7 Media Kit is installed.
2. The *root* filesystem can be no larger than 2 GB since you must have space for two complete *root* filesystems in the UNIX partition.
3. You can use a Department Edition UnixWare 7 licenses for cluster nodes. However, you must add an ODM or Disk Mirroring license.
4. If you create a */home* filesystem it is not automatically configured as a mirrored filesystem. You should create non-*root* filesystems after NSC is installed and configured.



5

Installing NSC Software



Lesson 5

Installing NSC Software

Introduction

Once UnixWare 7.1 is installed in the prescribed manner on the initial root node (node 1) of a cluster, the Non-Stop Cluster (NSC) software is then added to that node. This process *clusterizes* node 1. That is, it converts node 1 from a stand-alone computer running UnixWare 7.1 to the first node in a cluster. Subsequent nodes can then be installed using a NSC-specific simplified procedure.

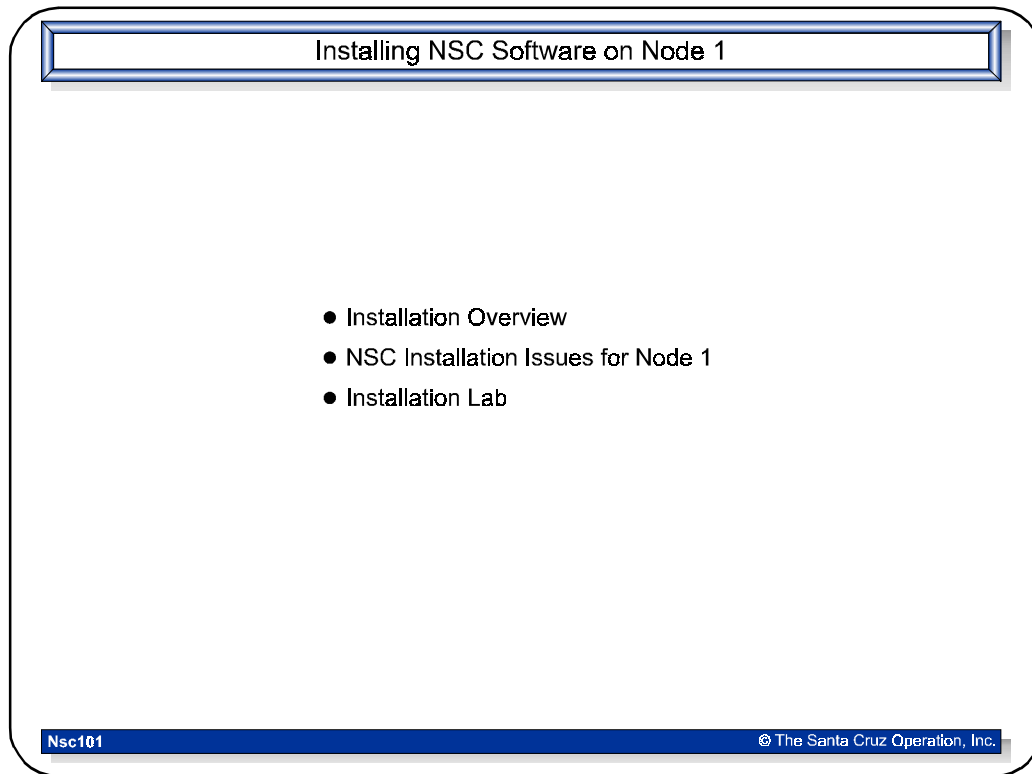
This lesson shows you how to install the NSC on the initial root node so that it is successfully clusterized and describes how other nodes are added to the cluster.

Objectives

By the end of this lesson, you will be able to:

- Perform NSC pre-installation preparation tasks
- Install NSC software and configure the initial root node
- Test the newly clusterized environment on node 1

AV 5-1



Topics

In order to meet these objectives, this lesson discusses:

- Node 1 Installation Procedure Overview — understanding the steps needed to install NSC on node 1, after the cluster hardware is all assembled and UnixWare 7 has been properly installed on node 1
- NSC Software Installation Information — navigating through the preparation steps and NSC installation prompts for all protected storage methods.
- Lab — installing NSC and related software to clusterize node 1

5.1 Understanding NSC Software Installation

Recall from the overview in Lesson 4: “Installing SCO UnixWare 7,” that your cluster software is assembled and installed in this sequence:

1. Assemble operating system software. Create or obtain vendor-supplied EFS or HBA diskettes, if needed.
2. Install UnixWare 7.
3. Install vendor-supplied EFSes.
4. Prepare the initial root node for NSC installation
5. Install NSC on the initial root node
6. Install the failover root node
7. Install depending nodes, if present.

Just as with the UnixWare 7 installation process, successful installation of the NSC software requires careful preparation. Because installation of all dependent nodes depends on the success of the installation and preparation of the initial root node, you must understand the configuration choices you will make during the NSC installation process before you begin.

This lesson describes Steps 4 through 6 of the software installation procedure. Each section gives you guidance about the selections you need to make during the NSC software preparation and installation steps.

The labs at the end of the lesson provide detailed procedures for installing the initial root node. There are separate procedures within the lab for CNM and shared-storage clusters.

The next lesson instructs you to install the failover root node and includes a lab that demonstrates the failover capability of your newly installed cluster.

5.2 Preparing to Install NSC on the Initial Root Node

Before installing the SCO UnixWare 7 NonStop Cluster software onto the UnixWare 7 initial root node you must perform several preparation tasks in a specific order prior to installing NSC.

This section provides information about the NSC-related pre-installation tasks you need to perform as part of the overall installation procedure included in Section 5.6, “Lab: Installing NSC on the Initial Root Node,” on page 5-18.

5.2.1 Applying Program Technical Fixes (PTFs)

Depending on the release of UnixWare and NSC you are installing, you might need to apply fixes to the operating system *before* installing NSC. For example, the NSC 7.1.1 CD-ROM includes a PTF for UnixWare 7.1.1 that must be installed before you install NSC. This is a package called UW7PTF and is installed using **pkgadd**.

Read the ***NonStop Clusters Release Notes*** for information about PTFs for the releases of UnixWare and NSC used in your classroom.

Note that there might also be fixes available from SCO that are more recent than those included on your NSC CD-ROM. For the latest information specific to your software version, see the SCO Support web site at URL:

<http://www.sco.com/Support/ftplists/nsclist.htm>

Be very careful to distinguish which fixes or supplements are required *before*, and which *after* NSC is installed.

5.2.2 Encapsulating the *root* filesystem

After applying any applicable supplements or PTFs for UnixWare, prepare for NSC installation by encapsulating the *root* filesystem on the internal drive of the initial root node.

The *root* filesystem failover configuration relies on the disk mirroring feature of the Veritas Volume Manager (**VxVM**) feature. Encapsulating the *root* filesystem brings it under Veritas control.

Section 5.6, “Lab: Installing NSC on the Initial Root Node,” on page 5-18 has detailed instructions for using `vxinstall` to encapsulate the *root* filesystem.

Do not install or encapsulate any other drives at this time. If you want to create additional filesystems on the root drive, or place other drives under **VxVM** management, do so after you have installed the NSC software. Adding non-*root* filesystems is discussed in the **UnixWare® 7 NonStop Clusters® Administration** course.

Note After you encapsulate the *root* filesystem, the system *must* be rebooted. Be sure to respond with **y** when prompted to reboot.

5.2.3 Preparing shared storage

When using shared storage units with SCO NonStop Clusters, you might be required to prepare the shared storage before you begin the NSC installation process.

These tasks do not apply to CNM clusters. If your classroom does not use shared storage units, skip this section and proceed to Section 5.3.1, “Selecting optional packages,” on page 5-12.

5.2.3.1 Stamping logical disks on shared storage units

Some shared storage units require you to stamp logical volumes with a unique identifier so they can be recognized by UNIX. This identifier is called a *nexus*.

The nexus is the name of the device in the `/dev/dsk` and `/dev/rdsk` directories and has the form `c#b#t#d#` where:

- The # after the ‘c’ is a controller number
- The # after the ‘b’ is a bus number
- The # after the ‘t’ is a target-id (sometimes called TID) number
- The # after the ‘d’ is a logical unit (sometimes called LUN) number

This must be done after UnixWare 7 is installed on the initial root node, but before the NSC software is installed.

If your cluster uses the Compaq FibreChannel Storage System, you must verify the array’s hardware connections and stamp its logical volumes with a SCSI nexus. Before beginning the NSC installation procedure, read Section 1.9.3, “Stamping disks in a Compaq Fibre array,” on page 65 in the ***NSC Curriculum Hardware Supplement*** for hardware set up information.

If your cluster uses a shared storage unit supplied by another vendor, check the Section 1.9, “Initializing Shared Storage,” in your vendor’s chapter in the ***NSC Curriculum Hardware Supplement*** to see if this requirement applies to your hardware.

You are referred to this section again at the appropriate step in the installation lab in Section 5.6.3, “Procedure for shared-storage clusters,” on page 5-27

5.2.3.2 Creating a UNIX partition and VTOC on shared storage

Some shared storage units require that you create a UNIX partition with a volume table of contents (VTOC) before installing NSC. Otherwise, the NSC installation will fail when trying to move the *root* filesystem from the internal drive to the external storage unit.

Use **diskadd(1M)** to create the UNIX partition with no (0) slices. This creates a valid VTOC table for the drive. The device name argument is of the form *c#b##t##d#*. For example, on a node with:

- An SCSI controller (with the internal root disk) as controller 0
- An IDE controller with a CD-ROM as controller 1
- A Fibre channel HBA connecting to the shared storage unit

This is an example command line for creating a UNIX partition on the first disk on the third controller on node 1:

```
# diskadd c2b0t0d0
```

Use the entire disk for UNIX, unless there is already a system, or vendor, partition on the disk. Do not remove these types of partitions. Use the remainder of the disk for UNIX.

Note Be sure specify 0 as the number of slices on the disk.

5.2.3.3 Controlling shared storage

Some shared storage solutions require that only one node actively controls the shared storage at any time. Since installation of the NSC software includes migrating the *root* filesystem from the internal disk (the one on which you installed UnixWare) to the external shared storage, node 1 must have control of the shared storage for the NSC installation to succeed.

If you are using a shared storage solution where all nodes have active control of the shared storage, such as JBOD or certain Fibre solutions, this is not an issue and you can skip this procedure.

To ensure that node 1 controls the external storage unit, you must power up the cluster hardware in a particular sequence. This is detailed in Section 5.6, “Lab: Installing NSC on the Initial Root Node,” on page 5-18, and summarized here:

1. Turn off all nodes sharing the storage other than node 1.
2. Shutdown node 1 with a normal, full system shutdown, not a shutdown with reboot.
3. Power off node 1.
4. Power off the shared storage devices.
5. Power on the shared storage devices. If the storage boxes and storage controllers are separate, follow the vendor’s instructions for power-on order.
6. Turn on and boot node 1. During the boot, node 1 will gain control of the shared storage.

5.2.4 Removing SDI ghost devices

SDI ghost devices can be created in certain circumstances on UnixWare systems. This occurs in multipath I/O environments—where two host bus adapters in a systems read and write to the same disk

You must remove any ghost devices that exist on your initial root node before installing NSC.

Section 5.6, “Lab: Installing NSC on the Initial Root Node,” on page 5-18 includes the steps used to check for, and remove, ghost devices.

5.3 Installing NSC on the Initial Root Node

Once all preparation tasks have been successfully accomplished, install the NSC software from the NSC CD-ROM using **pkgadd**.

This section provides an overview of all NSC-related information you need to follow the installation procedure provided in Section 5.6, “Lab: Installing NSC on the Initial Root Node,” on page 5-18.

5.3.1 Selecting optional packages

After you start the NSC installation procedure, you might be prompted to select optional packages. This depends on the version of NSC you are installing. Select all optional packages, unless directed otherwise by your instructor.

Read the ***UnixWare NonStop Clusters Release Notes*** for your version of NSC before you make selections to be sure you understand any restrictions or limitations.

5.3.2 Configuring *root* filesystem failover

One of the key features of NSC is the ability to have the cluster continue operation after losing the node controlling the *root* filesystem. This is possible only when there is another node that can take over control of the *root* filesystem.

How you configure failover depends on the type of protected storage method you are using. Regardless of whether you have a CNM or shared-storage cluster, you *must* configure *root* filesystem failover during the NSC installation process.

On CNM clusters:

- If node 1 has only one disk, the NSC installation program assumes you are performing a CNM install.
- For simplicity, do not configure OSU as part of the classroom lab later in this lesson. However, on live systems, it is recommended that you configure OSU.
- If node 1 has multiple disks, select the disk containing the *root* filesystem. For example, if the first disk on the first controller is used for the *root* filesystem, select “Drive 1 - c0b0t0d0” as the device.
- Enter **2** when prompted for the node number of the root failover node.
- Enter **0** when prompted to enter another cluster node number for root failover. This indicates there are no other nodes for root failover.
- The actual migration of the *root* filesystem to node 2 occurs when you install NSC on node 2.

On shared-storage clusters:

- Select the drive number that corresponds to the first external drive in the shared storage unit.
- For simplicity, do not configure OSU as part of the classroom lab later in this lesson. However, on live systems, it is recommended that you configure OSU.
- Enter the node number of the other node connected to the shared storage unit. In your classroom, this is generally node 2.
- Specify whether your shared storage unit is a RAID device.
- For simplicity, do not add additional disks for the *root* filesystem even if your classroom storage unit has multiple disks.
- The *root* filesystem is moved to the shared storage device when you complete this procedure and confirm your selections. This might take a while, depending on the size of your *root* filesystem.

5.3.3 Configuring the CVIP and PIP

During NSC installation, you are prompted configure the *Cluster Virtual IP (CVIP)* for the cluster. The NSC installation procedure automatically promotes the name and IP address of the network interface currently configured on this node to the name and IP address of the cluster.

Then you enter a new host name and IP address for the physical interface (NIC) on this node.

Be prepared with:

- A new host name for this node's network interface
- A new IP address for this node's network interface

The IP address should be on the same subnet as the CVIP.

5.3.4 Configuring optional packages

After you finish configuring CVIP information, the NSC installation program prompts you for any information needed to configure the optional packages you selected to install. The installation program does not actually install the packages until after it installs the NonStop Clusters software.

Follow any instructions that appear during optional package installation. If you are installing the CIM package, you are prompted for this Simple Network Management Protocol (SNMP) information. Enter appropriate values, or press to accept defaults for all fields.

For your classroom configuration, press through each field to accept default values.

5.4 Post-Installation Tasks

Before proceeding to install the dependent nodes in the cluster:

- Install application software you plan to have running on the root node. See the SCO NonStop Cluster Concepts course for help planning your cluster configuration to run applications.

In this course, it is unlikely you will install applications before continuing to Lesson 6: “Installing the Failover Root Node.”

- Remove and reinstall the ODM license using the **License Manager** so the license will apply cluster-wide.

5.5 Lesson Summary

This lesson has discussed how to prepare your initial root node for installing NSC software. An overview of the NSC installation process was provided to familiarize you with the types of configuration questions you need to be prepared to answer, whether your cluster uses CNM or shared storage units for protected storage.

The dependent node installation process was also presented.

Review questions

1. Which filesystems should you encapsulate before installing NSC?
2. How do you configure a NIC on a dependent node?
3. Is split-brain avoidance cabling needed for clusters using CNM?

5.6 Lab: Installing NSC on the Initial Root Node

This lab demonstrates how to install NonStop Cluster software on the initial root node and then test the configuration. Students are divided into two or more groups, one for each cluster.

5.6.1 Overview

Install your initial root node using the steps in the lab applicable to your cluster configuration. Properties of the cluster are established in Lesson 3: “Assembling and Initializing Cluster Hardware.” Review the properties of the cluster including its protected storage method, number of nodes, and SAN technology.

Before you begin, read the **SCO NonStop Clusters Release Notes**, if you have not already done so. You need the latest information applicable to your classroom release of NSC to install the correct PTFs and SLses.

5.6.2 Procedure for CNM clusters

To install NSC software and configure the initial root node:

- Step 1. Install any necessary updates to the operating system provided by SCO.

For example, if you are using UnixWare 7.1 or 7.1.1, install the PTF called UW7PTF from the NSC CD-ROM using **pkgadd**:

pkgadd -d cdrom1 UW7PTF

- Step 2. Configure the LAN NIC for node 1 using the **netcfg(1M)** command. Assign it the host name of the cluster and the IP address you are using as the CVIP.

- Step 3. Encapsulate the *root* filesystem.

- a. As **root**, enter:

vxinstall

A list of disk controllers found on the node is displayed with a description of the disk device-node naming conventions used by **VxVM**.

- b. Press to continue. An information screen is displayed describing the “Quick” and “Custom” options. Press to continue.
- c. Enter **2** to select a “Custom Installation.”
- d. Enter **y** when prompted to encapsulate the boot disk.

This encapsulates all filesystems on the boot disk, including the *root* filesystem. These filesystems will then be VxFS volumes when the node is rebooted.

- e. Accept the default name, *rootdisk*. Press .
- f. Verify the configuration displayed by **vxinstall**. Press to continue.
- g. Press and then enter **4** to leave the other disks unencapsulated.
- h. Press and then enter **4** if **vxinstall** detects other controllers. Leave all other disks on all controllers unencapsulated.

- i. Enter **y** when **vxinstall** displays a list of disks that will be encapsulated. It should only include *c0b0t0d0*.

If it does not, enter **q** to quit and restart **vxinstall**.

- j. Enter **y** when **vxinstall** prompts you to reboot.

WARNING! You *must* enter **y** to reboot after encapsulating the *root* filesystem. Continuing without rebooting the node causes the NSC installation to fail.

The system reboots multiple times. The number of reboots depends upon your system.

- Step 4. When the initial root node finishes booting, switch to the command-line login prompt by pressing the **CTRL** **ALT** **h** key combination, and log in as **root**.

Step 5. Check for SDI ghost devices. Enter:

/etc/scsi/sdighost -l

If this message is displayed, **sdighost** has found no ghost devices and you are finished with this procedure.

UX:sdighost:INFO:Currently, no names are in use as Ghost Names

Step 6. Remove SDI ghost devices, if found:

a. Enter:

/etc/scsi/sdighost -r

b. Shutdown the initial root node. Enter:

/etc/shutdown -i6 -g0 -y

c. Cycle power on the initial root node.

Step 7. When the initial root node finishes booting, switch to the command-line login prompt by pressing the **CTRL ALT h** key combination, and log in as **root**.

Step 8. Start the SCO NSC installation. Insert the NSC CD-ROM in the first CD-ROM drive on node 1 and enter:

pkgadd -d cdrom1 -q NSC

Step 9. Select all optional packages.

Step 10. Identify the node being installed. Since this procedure is always performed on the first node, enter **1** **ENTER**.

Step 11. Select the failover root node. For CNM:

- a. Enter the drive number of the disk that corresponds to the primary disk on the failover root node.
- b. Enter **n** to configure OSU, unless otherwise instructed.
- c. Enter **2** when prompted for the node number of the root failover node.
- d. Enter **0** when prompted to enter another cluster node number for root failover. This indicates there are no other nodes for root failover.

The actual migration of the root filesystem to node 2 will occur when you install NSC on node 2.

Step 12. Configure the CVIP.

- a. Enter **y** when prompted to configure the CVIP. The host name and IP address that was previously configured for the first NIC on node 1 is displayed. This will become the cluster name and CVIP.
- b. Enter the new host name for the NIC.
- c. Enter the new IP address for the NIC.
- d. Verify the values for the new interface to the NIC. Enter **y** if they are correct.

When you confirm the new network information for node 1, a message indicates that the information will be applied when the system is ready to run the NonStop Clusters software.

- Step 13. Configure SNMP information. Press  to accept defaults and continue.
- Step 14. Monitor the messages displayed by the installation program that provide status information about the tasks it is performing, such as copying NonStop Clusters files, mirroring the root filesystem, updating system files, and so on.
- Step 15. Remove the NSC CD-ROM and reboot the node. Enter:
- shutdown -i6 -y -g0**

AV 5-2

Cluster Commands

```
# cluster -v
1: UP
2: UP

# ifconfig -a
lo0: flags=4049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    inet 127.0.0.1 netmask ffffffff
    inet/perf: rcv size: 4096; send size: 8192; full-size frames: 1
    inet/options: rfc1323
cvip0: flags=84041<UP,RUNNING,MULTICAST,CVIP> mtu 16384
    inet 132.147.119.100 netmask fffffff0
    inet/perf: rcv size: 24576; send size: 24576; full-size frames: 1
    inet/options: rfc1323
net1.2: flags=4043<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 132.147.119.102 netmask fffffff0 broadcast 132.147.119.255
    inet/perf: rcv size: 24576; send size: 24576; full-size frames: 1
    inet/options: rfc1323
    ether 00:80:5f:bb:a9:6d
net0.1: flags=44043<UP,BROADCAST,RUNNING,MULTICAST,REMOTE> mtu 1500
    inet 132.147.119.101 netmask fffffff0 broadcast 132.147.119.255
    inet/perf: rcv size: 24576; send size: 24576; full-size frames: 1
    inet/options: rfc1323
    ether 00:80:5f:bb:c0:c6
```

Nsc102

© The Santa Cruz Operation, Inc.

AV 5-3

Cluster Commands (Cont'd)

```
# vxprint
Disk group: rootdg

TY NAME      ASSOC      KSTATE  LENGTH  PLOFFS  STATE  TUTILO
PUTILO
dg rootdg    rootdg      -        -        -        -        -
dm nscrootdisk nlc1b0t0d0s0 -      8373375 -        -        -
dm rootdisk  nlc0b0t0d0s0 -      8321670 -        -        -

pl rootvol-01 -          DISABLED 7839720 -        -        -
sd rootdisk-02 rootvol-01  ENABLED 7839720 0        -        -

v rootvol    root        ENABLED 7839720 -        ACTIVE -
pl nscrootvol-01 rootvol    ENABLED 7839720 -        ACTIVE -
sd nscrootdisk-01 nscrootvol-01 ENABLED 7839720 0        -        -

v standvol   fsngen      ENABLED 96390 -        ACTIVE -
pl standvol-01 standvol    ENABLED 96390 -        ACTIVE -
sd rootdisk-03 standvol-01 ENABLED 96390 0        -        -

v swapvol    swap        ENABLED 224910 -        ACTIVE -
pl swapvol-01 swapvol     ENABLED 224910 -        ACTIVE -

# mount
/ on /dev/root read/write/setuid on Mon Apr 17 01:34:23 2000
/dev/fd on /dev/fd read/write on Mon Apr 17 01:34:23 2000
/proc on /proc read/write on Mon Apr 17 01:34:23 2000
/stand/1 on /dev/stand.1 read/write on Mon Apr 17 01:34:24 2000
/system/processor on /processorfs read/write on Mon Apr 17 01:35:05 2000
/dev/tcp on /dev/tcp read/write on Mon Apr 17 01:35:05 2000
/stand/2 on /dev/stand.2 read/write on Mon Apr 17 01:38:59 2000
/stand/3 on /dev/stand.3 read/write on Mon Apr 17 01:39:27 2000
```

Nsc103

© The Santa Cruz Operation, Inc.

Step 16. Verify that the initial root node is successfully running the clusterized operating system. Although there are no other nodes installed yet, you can now see some of the differences in the NSC operating environment, as shown in AV 5-2 and AV 5-3 on page 5-24.

- a. Log into the cluster as **root**.
- b. Use these commands to view cluster-wide devices and information:

```
cluster -v  
ifconfig -a  
vxprint  
mount
```

Step 17. Install any Support Level Supplements (SLS) or PTFs required for NSC.

- a. Check the **SCO NonStop Clusters Release Notes** for any PTFs included on the NSC CD-ROM installation media.
- b. Check the SCO Support web site for additional SLses or PTFs. See URL:

<http://www.sco.com/Support/ftplists/nsclist.htm>

- c. Follow the instructions for installation in the accompanying documentation.

For example, if you are using UnixWare 7.1 or 7.1.1, install both the NSC SLS **nsc1006** from the CD-ROM, and NSC SLS **nsc1009**, which must be downloaded. These are both installed using **pkgadd**:

Follow the prompts to select and install SLS packages.

Reboot when finished:

```
shutdown -i6 -g0 -y
```

Step 18. Install any application software.

In the classroom setting, it is not likely you will be instructed to install additional software. However, in a real-world environment, you will need to do this before proceeding to install dependent nodes.

You have now completed installation of the initial root node.

5.6.3 Procedure for shared-storage clusters

To install NSC software and configure the initial root node in clusters using external shared storage units:

- Step 1. Install any necessary updates to the operating system provided by SCO.

For example, if you are using UnixWare 7.1 or 7.1.1, install the PTF called UW7PTF from the NSC CD-ROM using **pkgadd**. For example, enter:

pkgadd -d cdrom1 UW7PTF

- Step 2. Configure the LAN NIC for node 1 using the **netcfg(1M)** command. Assign it the host name of the cluster and the IP address you are using as the CVIP.

- Step 3. Encapsulate the *root* filesystem.

- a. As **root**, enter:

vxinstall

A list of disk controllers found on the node is displayed with a description of the disk device-node naming conventions used by **VxVM**.

- b. Press to continue. An information screen is displayed describing the “Quick” and “Custom” options. Press to continue.
- c. Enter **2** to select a “Custom Installation.”
- d. Enter **2** to select a “Custom Installation.”
- e. Enter **y** when prompted to encapsulate the boot disk.

This encapsulates all filesystems on the boot disk, including the *root* filesystem. These filesystems will then be VxFS volumes when the node is rebooted.

- f. Accept the default name, *rootdisk*. Press .
- g. Verify the configuration displayed by **vxinstall**. Press to continue.
- h. Press and then enter **4** to leave the other disks unencapsulated.

- i. Press **ENTER** and then enter **4** if **vxinstall** detects other controllers. Leave all other disks on all controllers unencapsulated.
- j. Enter **y** when **vxinstall** displays a list of disks that will be encapsulated. It should only include *c0b0t0d0*.

If it does not, enter **q** to quit and restart **vxinstall**.
- k. Enter **y** when **vxinstall** prompts you to reboot.

WARNING! You *must* enter **y** to reboot after encapsulating the *root* filesystem. Continuing without rebooting the node causes the NSC installation to fail.

The system reboots multiple times. The number of reboots depends upon your system.

- Step 4. When the initial root node finishes booting, switch to the command-line login prompt by pressing the **CTRL** **ALT** **h** key combination, and log in as **root**.
- Step 5. Stamp the logical disks with a SCSI *nexus* if you have a Fibre shared storage unit that does not have SCSI device addresses. See Section 1.9.2, “Stamping disks in the external disk subsystem,” in your vendor’s chapter in the **NSC Curriculum Hardware Supplement** for the procedure specific to your classroom hardware.
- Step 6. Initialize the logical disk to create a VTOC using **diskadd(1M)**. Create a UNIX partition with no slices. For example:

diskadd c2b0t0d0

- a. Enter **y****ENTER** to continue when warned about destroying data.
- b. Enter **y****ENTER** to accept the recommended partitioning of 100% for UNIX, unless your disk already has a vendor or system partition. In that case, allocate the rest of the disk to UNIX.
- c. Enter **n****ENTER** to skip surface analysis.
- d. Enter **0****ENTER** when prompted for the number of slices.
- e. Enter **y****ENTER** to confirm the disk allocation information.

- Step 7. Verify that node 1 has control of the shared storage devices:
- a. Turn off all nodes sharing the storage other than node 1.
 - b. Shutdown node 1 with a normal, full system shutdown, not a shutdown with reboot. For example:

cd /
shutdown -y -g0 -i0
 - c. Power off node 1.
 - d. Power off the shared storage devices.
 - e. Power on the shared storage devices. If the storage boxes and storage controllers are separate, follow the vendor's instructions for power-on order.
 - f. Turn on and boot node 1. During the boot, node 1 will gain control of the shared storage.
 - g. Switch to the home console screen using the   **h** key sequence.
 - h. Log in as **root**.

- Step 8. Check for SDI ghost devices. Enter:

/etc/scsi/sdighost -l

If this message is displayed, **sdighost** has found no ghost devices and you are finished with this procedure.

UX:sdighost:INFO:Currently, no names are in use as Ghost Names

- Step 9. Remove SDI ghost devices, if found:

- a. Enter:

/etc/scsi/sdighost -r

- b. Shutdown the initial root node. Enter:

/etc/shutdown -i6 -g0 -y

- c. Cycle power on the initial root node.

- Step 10. When the initial root node finishes booting, switch to the command-line login prompt by pressing the **CTRL ALT h** key combination, and log in as **root**.

- Step 11. Start the SCO NSC installation. Insert the NSC CD-ROM in the first CD-ROM drive on node 1 and enter:

pkgadd -d cdrom1 -q NSC

- Step 12. Select all optional packages.

- Step 13. Identify the node being installed. Since this procedure is always performed on the first node, enter **1** **ENTER**.

- Step 14. Select the failover root node. For shared storage clusters:
- Enter the drive number of the disk that corresponds to the shared storage device. If using RAID, this is a logical disk.
 - Enter **n** to configure OSU, unless otherwise instructed.
 - Indicate whether the selected disk is a hardware RAID device.
 - Enter **n** when prompted to configure additional disks for the *root* filesystem.
 - Indicate whether the configuration is correct when the current configuration is summarized.
 - Enter **2** when prompted for the node number of the root failover node.
 - Enter **0** when prompted to enter another cluster node number for root failover. This indicates there are no other nodes for root failover.
 - Wait while the *root* filesystem migrates to the shared storage unit. This takes several minutes.
- Step 15. Configure the CVIP.
- Enter **y** when prompted to configure the CVIP. The host name and IP address that was previously configured for the first NIC on node 1 is displayed. This will become the cluster name and CVIP.
 - Enter the new host name for the NIC.
 - Enter the new IP address for the NIC.
 - Verify the values for the new interface to the NIC. Enter **y** if they are correct.
- When you confirm the new network information for node 1, a message indicates that the information will be applied when the system is ready to run the NonStop Clusters software.
- Step 16. Configure SNMP information. Press  to accept defaults and continue.
- Step 17. Monitor these messages displayed by the installation program that provide status information about the tasks it is performing, such as copying NonStop Clusters files, mirroring the *root* filesystem, updating system files, and so on.
-

- Step 18. Remove the NSC CD-ROM and reboot the node. Enter:

shutdown -i6 -y -g0

- Step 19. Verify that the initial root node is successfully running the clusterized operating system. Although there are no other nodes installed yet, you can now see some of the differences in the NSC operating environment, as shown in AV 5-2 and AV 5-3 on page 5-24.

- a. Log into the cluster as **root**.
- b. Use these commands to view cluster-wide devices and information:

cluster -v
ifconfig -a
vxprint
mount

- Step 20. Install any Support Level Supplements (SLS) or PTFs required for NSC.

- a. Check the **SCO NonStop Clusters Release Notes** for any PTFs included on the NSC CD-ROM installation media.
- b. Check the SCO Support web site for additional SLses or PTFs. See URL:

<http://www.sco.com/Support/ftplists/nsclist.htm>

- c. Follow the instructions for installation in the accompanying documentation.

For example, if you are using UnixWare 7.1 or 7.1.1, install both the NSC SLS **nsc1006** from the CD-ROM, and NSC SLS **nsc1009**, which must be downloaded. These are both installed using **pkgadd**:

Follow the prompts to select and install SLS packages.

Reboot when finished:

shutdown -i6 -g0 -y

Step 21. Install any application software.

In the classroom setting, it is not likely you will be instructed to install additional software. However, in a real-world environment, you will need to do this before proceeding to install dependent nodes.

You have now completed installation of the initial root node.

5.7 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

cluster(1M)	Displays information about the cluster
fdisk(1M)	Manages UNIX disk partitions
format(1M)	Formats diskettes
ifconfig(1M)	Configures network interfaces
mount(1M)	Mounts filesystems
pkgadd(1M)	Installs software
sdghost(1Mtcp)	Displays ghost devices
shutdown(1M)	Changes system/cluster state
vxinstall(1M)	Encapsulates filesystems to put them under Veritas Volume Manager control
vxprint(1M)	Displays information about Veritas volumes and plexes

Relevant files

<i>/tmp</i>	Is the <i>/tmp</i> filesystem
<i>/var/tmp</i>	Is the <i>/var/tmp</i> filesystem
<i>/var</i>	Is the <i>/var</i> filesystem

5.8 Answers to Review Questions

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Review Question Answers

This section contains the answers to the review questions on page 5-17.

1. Encapsulate only the root filesystem before installing NSC. Configure other non-root filesystems you want to put under control of the Veritas Volume Manager after you have successfully installed NSC on the initial root node.
2. Network interfaces are always configured using **netcfg**, whether adding a NIC to a node, or a CVIP to the cluster.
3. Split-brain avoidance should be configured on all CNM clusters because they do not use SAN switches.



6

Installing the Failover Root Node



Lesson 6

Installing the Failover Root Node

Introduction

Once UnixWare 7.1 is installed in the prescribed manner on the initial root node (node 1) of a cluster, the Non-Stop Cluster (NSC) software is then added to that node. This process *clusterizes* node 1. That is, it converts node 1 from a stand-alone computer running UnixWare 7.1 to the first node in a cluster. Subsequent nodes can then be installed using a NSC-specific simplified procedure.

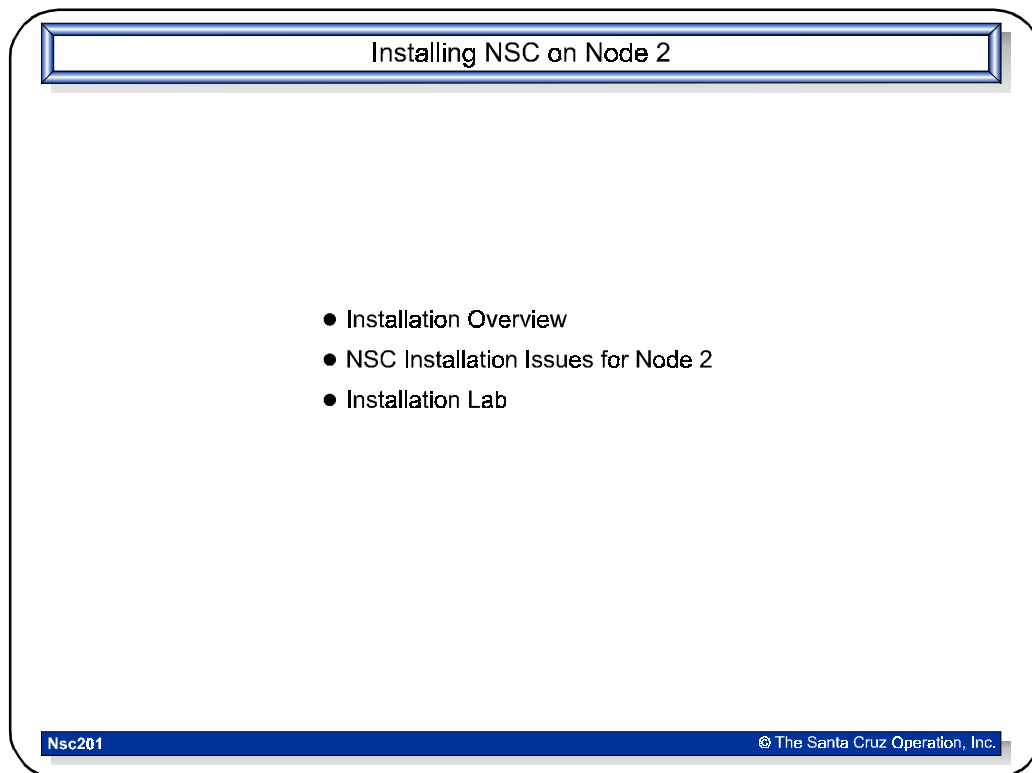
This lesson shows you how to install the NSC on the initial root node so that it is successfully clusterized and describes how other nodes are added to the cluster.

Objectives

By the end of this lesson, you will be able to:

- Perform NSC pre-installation preparation tasks
- Install NSC software and configure the failover root node
- Simulate node failure to demonstrate the NSC failover capability

AV 6-1



Topics

In order to meet these objectives, this lesson discusses:

- Node 2 Installation Procedure Overview — understanding the steps needed to install NSC on node 2, the failover root node
- NSC Dependent Node Installation Information — navigating through the preparation steps and NSC installation prompts
- Lab — installing NSC and related software on the failover root node and simulating node failure

6.1 Understanding NSC Dependent Node Installation

Recall from previous lessons that your cluster software is assembled and installed in this sequence:

1. Assemble operating system software. Create or obtain vendor-supplied EFS or HBA diskettes, if needed.
2. Install UnixWare 7.
3. Install vendor-supplied EFSes.
4. Prepare the initial root node for NSC installation
5. Install NSC on the initial root node
6. Install the failover root node
7. Install other depending nodes, if present.

After your initial root node is completely installed and you have performed all post-installation tasks described in Lesson 5: “Installing NSC Software,” you can install dependent nodes. This is the last step in the installation sequence.

The failover root node is always installed immediately after the initial root node. The installation process for all dependent nodes is simple and fast. Most of the configuration decisions affecting the installation of dependent nodes was made during the installation of the initial root node. There are three basic steps:

1. Create the dependent node boot diskettes (DNBD).
2. Boot from the floppies and respond to a series of prompts.
3. Reboot the node.

This lesson describes the dependent node installation process in detail. Each section gives you guidance about the selections you need to make during the NSC dependent node software preparation and installation steps.

The lab at the end of the lesson provides detailed procedures for installing the failover root node. There are separate procedures within the lab for CNM and shared-storage clusters.

The final lab demonstrates the failover capability of your newly-installed cluster.

6.2 Installing the Failover Root Node

This section provides an overview of all NSC-related information you need to perform the installation procedure provided in Section 6.4, “Lab: Creating Dependent Node Boot Diskettes,” on page 6-13.

6.2.1 Creating dependent node boot diskettes

When NSC is installed on your initial root node, a program called **make_floppy** is added. This is used to create pairs of dependent node boot diskettes used to install each of the dependent nodes in your cluster.

The **make_floppy** command

- Creates a bootable diskette image containing a minimal NSC kernel
- Creates a diskette image of a minimal NSC *root* filesystem
- Writes the disk images to floppy diskettes.

These disks are analogous to boot and root floppies used to start the general installation procedure for many version of UNIX.

Note Before you run **make_floppy**, you must prepare two formatted diskettes. These diskettes must be error-free or **make_floppy** will fail when attempting to write to the diskettes. If it fails, you must start again from the beginning and recreate the images. This is a time-consuming process. To help insure that **make_floppy** succeeds the first time, use the `verify` option to format.

Some versions of **make_floppy** have undocumented flags to store the DNBD images on the hard disk. Then, if the write process fails, you can restart from that point without having to recreate the images. View the **make_floppy** shell script to see if the version you are using supports these flags. If so, you are recommended to run **make_floppy** to save the images on the hard disk first. Then run **make_floppy** again to write the diskettes.

6.2.2 Booting the failover root node using the DNBDs

The installation of dependent nodes is started by booting the node using DNBD1 diskette. After you insert the DNBD2 diskette, the installation process starts automatically.

The next sections describe the information you need to respond to installation prompts.

6.2.3 Adding the node license

You must add a UnixWare 7 license when installing each dependent node. This should be the same type of license Edition as the root node, but a unique UnixWare 7 license for this node. The **License Manager** is automatically started during the NSC dependent node installation process. Add the license number, data, and node number in the **License Manager** form.

Be sure you enter a unique UnixWare 7 license. Do *not* enter the NSC license.

6.2.4 Checking the fdisk table for CNM

The disk in the failover root node that contains the failover *root* filesystem must have a valid partition table, if you have a two-node cluster using CNM. It must also have enough space for a UNIX partition large enough to hold a mirror of *root* filesystem on node 1.

Hence, this procedure must be done *from node 1*, just after adding the license during the dependent node installation process..

Note The timing for performing this step is critical. It must be performed after node 2 has booted and it's root disk device is available, but before the step in the NSC installation which configures the disk on node 2. Otherwise, the dependent node installation process will fail at this step.

6.2.5 Configuring external storage for dependent nodes

When using an external shared storage unit for your *root* filesystem, the NSC installation process configures the dependent node disks.

Select the option for automatic configuration and apply changes when prompted.

6.2.6 Configuring split-brain avoidance for CNM

Recall that a split-condition can exist when both nodes try to act as active root nodes. Certain SAN implementations that do not use SAN switches can experience this condition. If this occurs, your cluster can experience these problems:

Data corruption	Both the root and failover nodes write to the <i>root</i> filesystem on their local disks and the <i>root</i> filesystem becomes unsynchronized
Ping-pong panics	Each node believes it cannot take ownership of the root filesystem, because their mirror is not up to date, and panics

You can avoid this by configuring an alternate communication path using a null-modem cable between serial ports on two nodes. This prevents your *root* filesystem from being destroyed when the SAN communication is lost.

You are recommended to configure and test split-brain avoidance whenever you are installing a CNM cluster. You must have the serial cable attached to successfully complete the test. However, even if you do not have the cable attached, configure the NSC software for split-brain avoidance and test it later.

6.2.7 Configuring networking on the dependent node

During NSC installation, you are prompted to configure any network adapters present in the dependent node. The installation process automatically starts the **Network Configuration Manager**.

Configure networking for the NICs in the dependent node just as you would in any UnixWare system

Remember these points when configuring this NIC:

- If the dependent node has multiple NICs and they support the standard UnixWare 7 failover feature, you have the option to configure one as a *standby* interface. For NSC, do not configure NICs as standby. All NICs can be active in an NSC environment, and failover is achieved by way of the CVIP facility.
- The IP address for this NIC must be in the same subnet as the CVIP in order to support failover from another node.
- The host name should be unique.

Note The **Network Configuration Manager** automatically enters the cluster virtual host name in the TCP/IP configuration form. Change this to a unique name for this node.

6.2.8 Installing the Mirroring license

For CNM clusters, you must remove and reinstall the Veritas Mirroring or ODM license after the cluster is fully installed.

If the license is not applied cluster-wide, the root filesystem cannot be remirrored in some circumstances. This will cause ping-pong panics during boot when the root node has failed over.

Use the SCOadmin **License Manager** to remove and reinstalled the Veritas Mirroring or ODM license.

6.3 Lesson Summary

This lesson has discussed how to prepare your initial root node for installing NSC software. An overview of the NSC installation process was provided to familiarize you with the types of configuration questions you need to be prepared to answer, whether your cluster uses CNM or shared storage units for protected storage.

The dependent node installation process was also presented.

Review questions

1. Which filesystems should you encapsulate before installing NSC?
2. How do you configure a NIC on a dependent node?
3. Is split-brain avoidance cabling needed for clusters using CNM?

6.4 Lab: Creating Dependent Node Boot Diskettes

When the initial root node installation is finished, you are ready to install the cluster's failover root node. The installation procedure for dependent nodes is much less complex than the setup of node 1.

6.4.1 Overview

Follow this procedure to create the DNBDs needed to install each dependent node.

If your classroom cluster has more than two nodes, make a separate set for each additional dependent node.

After creating the boot diskettes, install node 2 as described in Section 6.5, "Lab: Installing the Failover Root Node," on page 6-15.

6.4.2 Procedure: Using `make_floppy` to create DNBDs

NonStop Clusters software includes a **make_floppy** command to make bootable versions of the single-system image containing all the information you provided in previous installation lessons. You then use these diskettes to install each dependent node in your cluster.

To make the dependent node boot diskettes:

- Step 1. Format two 1.44MB floppies using the **format(1M)** command. Although it is slow, use the verify option to insure that the floppies are error-free. Enter:

```
format -v /dev/rdsk/f03ht
```

- Step 2. Label the floppies *DNBD1* and *DNBD2*.

- Step 3. On node 1, insert DNBD1 into the diskette drive.

- Step 4. Run **make_floppy**. Enter:

```
make_floppy diskette1
```

Wait while the boot image is built. This will take several minutes.

- Step 5. When prompted, replace DNBD1 with DNBD2 and press .

- Step 6. When the **make_floppy** command completes, remove DNBD2 from the drive.

- Step 7. Write protect both diskettes.

When finished, install the failover root node according to the procedure in Lesson 6: “Lab: Installing the Failover Root Node.”

6.5 Lab: Installing the Failover Root Node

After creating the dependent node boot diskettes, you are ready to install the cluster's failover root node. The installation procedure for dependent nodes is much less complex than the setup of node 1.

Separate procedures are provided for CNM and shared storage clusters.

6.5.1 Overview

This lab has separate procedures for each type of cluster, CNM or shared storage. Use the lab applicable to your classroom configuration.

After installing and testing dependent nodes, verify that root failover is working by completing the final lab, Section 6.6, "Lab: Testing Failover," on page 6-23.

6.5.2 Procedure: Installing the failover root node for CNM clusters

To install the failover root node of a CNM cluster:

- Step 1. Boot the initial root node, if it is not already running.
- Step 2. Initialize the failover root node hardware, if you have not already done so. See Section 1.8, “Initializing Nodes,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*.
- Step 3. Boot the dependent node using the first DNBD diskette.
- Step 4. Insert the second DNBD when prompted and press **ENTER**.
- Step 5. Enter node number **2** for the failover root node, when prompted.
- Step 6. Ensure that the disk on this node has a valid **fdisk** table.
- Log in as **root** on node 1.
 - Change to the `/dev/dsk` directory.
 - View a listing of all disk devices on node 2. Enter:

ls n2*
 - Note the device node that has the lowest number after the letter c and the rest of the string containing `b0t0d0s0`. For example, `n2c0b0t0d0s0` or `n2c1b0t0d0s0`.
 - Use **fdisk**(1M) with the device you selected to see whether a UNIX partition exists on that device. For example:

fdisk n2c0b0t0d

If the disk has a UNIX partition, skip the rest of this procedure.
 - Use **diskadd**(1M) with the device you selected to create a UNIX partition with 0 slices on that device if a partition does not exist. For example:

diskadd n2c0b0t0d0

Notice that the `s0` portion is left off.

Be careful not to delete any system partition that may have been created by setup software proprietary to the vendor of the node. Other than that, the rest of the disk should be used for a single UNIX partition.

Step 7. Enter an appropriate UnixWare 7 license for this node.

- a. Enter **y** to start the **License Manager** and press **F1** or tab to access the menu bar.
- b. Select **License** ⇒ **Add**.
- c. Enter this node's UnixWare license number, code and data from the COLA that came in your *SCO UnixWare* media kit.

Recall, the dependent nodes must have the same *type* of license Edition as the initial root node, but the actual license number must be unique. If you are not sure which license is required, see Section 4.3.1, "UnixWare 7 licenses," on page 4-8.

Enter the node number for this dependent node.

- d. Tab to the **License** button and press **ENTER**.
- e. Press **ENTER** to select the **OK** button when prompted.

Notice the **License Manager** is clusterized, and shows all licenses for each node.

- f. Select **Host** ⇒ **Exit** to quit.

- Step 8. Configure split-brain avoidance, if using CNM:
- Enter **y** when prompted to configure split-brain avoidance.
 - Select the port on node 1 to which the serial cable is attached.
 - Select the port on node 2 to which the serial cable is attached.
 - Enter **y** when prompted to confirm your choices.
 - Attach the cable to the selected ports, if you have not already done so during hardware assembly.
 - Test the connection.
- Step 9. Configure the LAN NIC cards on the dependent node.
- Enter **y** when prompted to enter network information.

The **Network Configuration Manager** is started. It displays a list of all network interface controllers (NICs) and CVIP(s) already installed in the cluster.
 - Press  (tab) ⇨  to access the **Hardware** menu item.
 - Select **Add new LAN adapter**. The **Add new LAN adapter** window appears. This window contains the list of LAN adapters not yet configured.

Do not configure any NICs as *standby* if your node has multiple NICs.
 - Use your arrow keys to highlight the LAN NIC are configuring.
 - Press the  (tab) twice to highlight **Continue** and press . Wait while the network interface is added to the cluster.
 - If the driver for the NIC has special configuration capabilities, another window appears asking you to configure those capabilities. Configure the capabilities now, if needed.
 - The **Add Protocol** window appears with the **TCP/IP** protocol already selected (if it is not, select it).
 - Press  (tab) to select **Add**. This displays the **Internet Protocol Configuration** window.

- i. Enter the appropriate information for the node you are installing.

Note Be sure to enter a new host name. By default, the cluster name is entered in the “Host name” field. Change this to a node-specific name.

Choose an IP address on the same subnet as the CVIP and PIPs in the initial root node. Otherwise, the CVIP will not failover if the root node fails.

- j. Press  (tab) to select **OK**. A window displays a successful completion message.
- k. Press  to select **OK**. You are returned to the main **netcfg** window.
- l. Configure additional NICs, if this node has multiple network adapters.
- m. From the **Hardware** menu item, press  and select **Exit**.

Step 10. Remove any diskettes from the diskettes or CD-ROMs and then press  to reboot the cluster.

Step 11. Test the cluster. Log into the cluster as **root** and use these commands to view cluster-wide devices and information:

```
cluster -v  
ifconfig -a  
vxprint  
mount
```

Step 12. Run any vendor-supplied cluster test utilities. See Section 1.12, “Vendor-Supplied Software,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*.

Step 13. Use the SCOadmin **License Manager** to remove and reinstalled the Veritas Mirroring or ODM license.

6.5.3 Procedure: Installing the failover root node for shared-storage clusters

To install the failover root node for shared storage clusters:

- Step 1. Boot the initial root node, if it is not already running.
- Step 2. Initialize the failover root node hardware, if you have not already done so. See Section 1.8, “Initializing Nodes,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*.
- Step 3. Boot the dependent node using the first DNBD diskette.
- Step 4. Insert the second DNBD when prompted and press .
- Step 5. Enter node number **2** for the failover root node, when prompted.
- Step 6. Enter an appropriate UnixWare 7 license for this node.
 - a. Enter **y** to start the **License Manager** and press or tab to access the menu bar.
 - b. Select **License** ⇒ **Add**.
 - c. Enter this node’s UnixWare license number, code and data from the COLA that came in your *SCO UnixWare* media kit.

Recall, the dependent nodes must have the same *type* of license Edition as the initial root node, but the actual license number must be unique. If you are not sure which license is required, see Section 4.3.1, “UnixWare 7 licenses,” on page 4-8.

Enter the node number for this dependent node.
 - d. Tab to the **License** button and press .
 - e. Press to select the **OK** button when prompted.

Notice the **License Manager** is clusterized, and shows all licenses for each node.
 - f. Select **Host** ⇒ **Exit** to quit.

- Step 7. Set up the dependent node's disk.
- Select "Disk 1" when prompted.
 - Select "Automatic configuration."
 - Enter the number of the node you are installing, when prompted with the "Node X Disk Setup" screen.
 - Select "Apply changes to hard disk(s)."
 - Select "Apply changes and exit."
- Step 8. Configure the LAN NIC cards on the dependent node.
- Enter **y** when prompted to enter network information.

The **Network Configuration Manager** is started. It displays a list of all network interface controllers (NICs) and CVIP(s) already installed in the cluster.
 - Press  (tab) ⇨  to access the **Hardware** menu item.
 - Select **Add new LAN adapter**. The **Add new LAN adapter** window appears. This window contains the list of LAN adapters not yet configured.

Do not configure any NICs as *standby* if your node has multiple NICs.
 - Use your arrow keys to highlight the LAN NIC are configuring.
 - Press the  (tab) twice to highlight **Continue** and press . Wait while the network interface is added to the cluster.
 - If the driver for the NIC has special configuration capabilities, another window appears asking you to configure those capabilities. Configure the capabilities now, if needed.
 - The **Add Protocol** window appears with the **TCP/IP** protocol already selected (if it is not, select it).
 - Press  (tab) to select **Add**. This displays the **Internet Protocol Configuration** window.

- i. Enter the appropriate information for the node you are installing.

Note Be sure to enter a new host name. By default, the cluster name is entered in the “Host name” field. Change this to a node-specific name.

Choose an IP address on the same subnet as the CVIP and PIPs in the initial root node. Otherwise, the CVIP will not failover if the root node fails.

- j. Press  (tab) to select **OK**. A window displays a successful completion message.
- k. Press  to select **OK**. You are returned to the main **netcfg** window.
- l. Configure additional NICs, if this node has multiple network adapters.
- m. From the **Hardware** menu item, press  and select **Exit**.

Step 9. Remove any diskettes from the diskettes or CD-ROMs and then press  to reboot the cluster.

Step 10. Test the cluster. Log into the cluster as **root** and use these commands to view cluster-wide devices and information:

```
cluster -v  
ifconfig -a  
vxprint  
mount
```

Step 11. Run any vendor-supplied cluster test utilities. See Section 1.12, “Vendor-Supplied Software,” in your vendor’s chapter in the *NSC Curriculum Hardware Supplement*.

6.6 Lab: Testing Failover

This lab simulates a hardware failure that causes node 1 to become unusable and shows how control of the root filesystem migrates to node 2.

6.6.1 Overview

Node 1 will be powered down to simulate a hardware failure. You then log in to node 2 and confirm that the root filesystem is under control of node 2, the failover root node.

WARNING! If your cluster uses CNM, you must ensure that the failover root node filesystem mirror is current before you force a failure on node 1. Otherwise, node 2 will panic during boot.

6.6.2 Procedure: Simulating root node failure

This procedure can be used to simulate a hardware failure of the root node. This demonstrates the failover process where the failover root node takes control of the root filesystem.

- Step 1. Log into the cluster as **root**.
- Step 2. If your cluster uses CNM, verify that the *root* filesystem mirror on the failover root node is up to date using **vxinfo**. Enter:

```
vxinfo -p | grep IOFAIL
```

If the *rootvol-02* plex has **IOFAIL** as the mode, the failover *root* filesystem mirror is not up to date. **DO NOT PROCEED** until the mirror is updated. The **vxrecover** process updates the mirror. Make sure it has completed, and recheck the status using **vxinfo** before you continue.

WARNING! If you power off your root node before the *root* filesystem mirror is synchronized, your failover root node will panic when booted. Make sure you have removed and reinstalled the Mirroring or ODM license before proceeding.

- Step 3. Determine which node controls the *root* filesystem. This node is the root node. Enter:

```
where /
```

- Step 4. Log into the console on node 2.



- Step 5. Make sure the *root* filesystem mirror is up to date before proceeding. This is described in Step 2.

- Step 6. Run a command that will continue for several minutes. For example:

```
find / -name * print.
```

Step 7. Power off the root node. Note what happens on node 2. Your shell and viewer process continue to run, although it might pause during failover.

Step 8. Verify that the root filesystem is now controlled by the failover root node.
Enter:

where /

The new root node (formerly the failover root node) now has control of the *root* filesystem.

6.7 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

cluster(1M)	Displays information about the cluster
format(1M)	Formats diskettes
ifconfig(1M)	Configures network interfaces
make_floppy(1M)	Makes boot disks for installing dependent cluster nodes
mount(1M)	Mounts filesystems
pkgadd(1M)	Installs software
shutdown(1M)	Changes system/cluster state
vxinstall(1M)	Encapsulates filesystems to put them under Veritas Volume Manager control
vxprint(1M)	Displays information about Veritas volumes and plexes

6.8 Answers to Review Questions

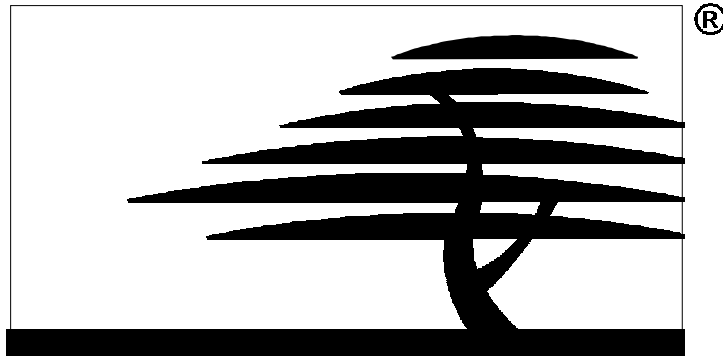
This section contains the answers to exercises, review questions, and labs presented in this lesson.

Review Question Answers

This section contains the answers to the review questions on page 6-12.

1. Encapsulate only the *root* filesystem before installing NSC. Configure other non-root filesystems you want to put under control of the Veritas Volume Manager after you have successfully installed NSC on the initial root node.
2. Network interfaces are always configured using **netcfg**, whether adding a NIC to a node, or a CVIP to the cluster.
3. Split-brain avoidance should be configured on all CNM clusters because they do not use SAN switches.





SCO

**UnixWare[®] 7 NonStop[®]
Clusters[®]
Administration**

Student Guide

Course Release: 1.0

For Software: UnixWare[®] 7 NonStop[®] Clus-
ters Version 7.1.1

©2000 The Santa Cruz Operation, Inc. All rights reserved.

This publication is protected under copyright laws and international treaties.

Information in this document is subject to change without notice and does not represent a commitment on the part of The Santa Cruz Operation, Inc.

SCO documents are provided as is and may include technical inaccuracies or typographical errors. The Santa Cruz Operation, Inc. reserves the right to add, delete, change or modify SCO documents at any time without notice. SCO, the SCO logo, The Santa Cruz Operation, SCO OpenServer, Tarantella, the Tarantella logo and UnixWare are trademarks or registered trademarks of The Santa Cruz Operation, Inc. in the USA and other countries. IXI, X.desktop, Deskworks, Wintif, IXI Panorama, Deskterm, and IXI Desktop are trademarks or registered trademarks of IXI Limited, a subsidiary of The Santa Cruz Operation, Inc. Visionware, Devkit, Vision, Esprit, Kodon, Super.Vision, Visionware Super.Vision, Term.Vision, Visionware Direction, the Visionware logo, Visionware SQL-Retriever, and XVision are registered trademarks of Visionware Limited, a subsidiary of The Santa Cruz Operation, Inc. SuperVision, TermVision, X-Visionware, SQL-Retriever, and Vision Builder are trademarks of Visionware Limited. UNIX is a registered trademark of The Open Group in the United States and other countries. Cheyenne and ARCServe are registered trademarks of Cheyenne Software, Inc. Netscape, Netscape Navigator, Netscape Communications Server, Netscape Commerce Server, Netscape Proxy Server, Netscape FastTrack Server, Netscape Enterprise Server, Netscape SuiteSpot, Netscape Catalog Server, Netscape News Server, Netscape Mail Server, and Netscape Navigator Gold are trademarks of Netscape Communications Corporation. NFS was developed by Legent Corporation (formerly Lachman Associates, Inc.) based on LACHMAN SYSTEM V NFS. LACHMAN is a trademark of Legent Corporation. NFS is a trademark of Sun Microsystems, Inc. TCP/IP was developed by Legent Corporation (formerly Lachman Associates, Inc.) based on LACHMAN SYSTEM V STREAMS TCP, a joint development of Lachman Associates and Convergent Technologies. MPX was developed by Corollary, Inc. VP/ix is a product developed and licensed by Phoenix Technologies, Ltd/INTERACTIVE Systems Corporation. XRemote is a registered trademark of Network Computing Devices, Inc. Oracle is a registered trademark of Oracle Corporation, Redwood City, California. Java is a trademark of Sun Microsystems, Inc. Compaq, Tandem, Integrity, NonStop, NetFlex, ProLiant, ROMPaq, ServerNet, SmartStart, registered in United States Patent and Trademark Office. Intel and Pentium are registered trademarks, and Pentium II Xeon and Pentium III Xeon are trademarks of Intel Corporation. SCO, Veritas, VxFS, and VxVA are registered trademarks of Veritas Software Corporation. Other product names mentioned herein may be trademarks and/or registered trademarks of their respective companies. All other brand and product names are or may be trademarks of, and are used to identify products or services of, their respective owners.

The SCO software that accompanies this publication is commercial computer software and, together with any related documentation, is subject to the restrictions on US Government use as set forth below. If this procurement is for a DOD agency, the following DFAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the US Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

If this procurement is for a civilian government agency, this FAR Restricted Rights Legend applies:

RESTRICTED RIGHTS LEGEND: This computer software is submitted with restricted rights under Government Contract No. _____ (and Subcontract No. _____, if appropriate). It may not be used, reproduced, or disclosed by the Government except as provided in paragraph (g)(3)(i) of FAR Clause 52.227-14 alt III or as otherwise expressly stated in the contract. Contractor/Manufacturer is The Santa Cruz Operation, Inc., 400 Encinal Street, Santa Cruz, CA 95060.

Table of Contents

Lesson 1: Course Introduction	1-1
Introduction.....	1-3
Course goal and purpose.....	1-3
Course Lessons	1-7
Target audience	1-8
Course roadmap	1-8
NSC Curriculum Hardware Supplement	1-8
Student Materials	1-10
The Student Guide	1-10
SCO documentation.....	1-10
Typographical Conventions	1-11
Sample Exam Items	1-13
 Lesson 2: Single System Image (SSI) and NSC Administration	 2-1
Overview of SSI in NSC and Current Limitations	2-7
Processes	2-10
File and Device Names	2-13
Networking and CVIP	2-15
Startup and Shutdown	2-16
Lesson Summary.....	2-17
Relevant Commands and Files	2-18
Answers to Exercises and Review Questions	2-19
 Lesson 3: Basic Process Administration	 3-1
Overview.....	3-6
The global process view	3-7
Process-related user-level NSC concepts	3-9
Cluster-Specific Process-Related Commands	3-11
The cluster (1) command	3-11
The onall (1) and onnode (1) commands.....	3-12
The node_self (1), clusternode_num (1), and where (1) commands	3-13
The fast (1) and fastnode (1) commands	3-15
The kill3 (1) and migrate (1) commands.....	3-16
Multi-Node Execution	3-18
IPC overhead.....	3-18
Migrating kernel objects	3-18
Lesson Summary.....	3-20
Relevant Commands and Files	3-21
Answers to Exercises and Review Questions	3-23

Lesson 4: Filesystem Administration**4-1**

Overview of Files, Filesystems, and Device nodes in NSC	4-7
Filesystem Administration in NSC	4-11
Understanding protected storage	4-11
Mount options in NSC	4-15
Mounting and unmounting filesystems.....	4-16
Configuring Filesystems for Failover	4-17
Configuring protected filesystems on hardware RAID arrays.....	4-18
Physically installing the drives	4-19
Creating logical volumes	4-19
Stamping volumes with a SCSI nexus.....	4-20
Configuring protected filesystems on CNM clusters.....	4-23
Physically installing the drives	4-24
Understanding Veritas Volume Management.....	4-25
Using the VERITAS Volume Manager.....	4-39
Lesson Summary.....	4-41
Relevant Commands and Files	4-42
Lab: Configuring a Non-Root Filesystem for Failover	4-43
Procedure for FibreChannel clusters.....	4-43
Procedure for CNM clusters	4-47
Answers to Exercises, Review Questions, and Labs	4-54

Lesson 5: Network Administration**5-1**

Overview of Networking in NSC	5-7
Adding NICs and PIPs.....	5-11
Understanding the CVIP	5-13
How the CVIP works.....	5-13
CVIP and name resolution.....	5-17
Adding CVIPs.....	5-19
IP Aliasing in NSC	5-20
Packet Routing in NSC	5-21
Running Multiple Server Daemons	5-23
PPP	5-24
Lesson Summary.....	5-25
Relevant Commands and Files	5-26
Answers to Exercises, Review Questions, and Labs	5-27

Lesson 6: Startup and Shutdown	6-1
---------------------------------------	------------

Cluster Startup	6-7
Cluster boot procedure.....	6-7
Booting CNM clusters	6-8
The startup sequence.....	6-9
Configuring Startup	6-10
The <i>boot(4)</i> file in NSC	6-10
The <i>inittab(4)</i> file and init(1M) in NSC	6-11
Shutdown For The Entire Cluster	6-14
Shutdown and Failure On Individual Nodes.....	6-15
Reforming the cluster.....	6-15
Controlled shutdown of an individual node.....	6-16
Reboot parameters	6-17
Lesson Summary.....	6-18
Lab: Rebooting the Cluster	6-19
Lab: Modifying <i>inittab(4)</i>	6-20
Relevant Commands and Files	6-21
Answers to Review Questions	6-22

Lesson 7: Load Leveling	7-1
--------------------------------	------------

Load Leveling Goals.....	7-7
Load Leveling Concepts	7-9
The Mechanics of Load Leveling	7-13
Configuring Load Leveling	7-15
The <i>parameters(4)</i> file	7-15
The <i>migrate_commands(4)</i> and <i>rexec_commands(4)</i> files	7-16
load_level flags	7-17
Starting and stopping the load leveling daemons	7-18
Starting load leveling from the command line.....	7-18
Starting load leveling using the keepalive subsystem.....	7-18
Restarting the load leveling daemons at boot time	7-19
Stopping load leveling started from the command line	7-19
Stopping load leveling started from keepalive	7-19
Overriding load leveling	7-20
Load Leveling Domains.....	7-21
Lesson Summary.....	7-23
Lab: Using Load-Leveling.....	7-24
Configuring load leveling	7-24
Optional lab: Setting up multiple load leveling domains	7-27
Answers to Exercises, Review Questions, and Labs	7-31

Lesson 8: Failover	8-1
Review of Failover Concepts.....	8-7
Understanding Failover.....	8-9
The keepalive (1M) daemon	8-9
The keepalive (1M) scripts.....	8-13
The spawndaemon (1M) configuration files	8-13
The spawndaemon (1M) command.....	8-15
Configuring Failover.....	8-16
Configuring failover for a multi-process application	8-17
Restarting Databases	8-19
The NCMS Keepalive Manager	8-21
Lesson Summary.....	8-25
Lab: Configuring Failover	8-26
Configuring a single-process application	8-27
Configuring a multi-process application.....	8-30
Relevant Commands and Files	8-33
Answers to Exercises, Review Questions, and Labs	8-34
 Lesson 9: Configuring NSC Behavior	 9-1
Overview of NCMS	9-7
Running NCSM	9-9
Configuring NSC	9-10
The Event Processor Subsystem	9-13
How EPS works	9-15
EPS Configuration Files	9-16
Configuring EPS using the NCMS Configuration Manager	9-17
Lesson Summary.....	9-19
Relevant Commands and Files	9-20
Answers to Review Questions	9-21
 Lesson 10: Troubleshooting Applications	 10-1
Detecting and Solving Typical Problems.....	10-6
Storing process IDs in a short integer	10-6
No <i>/stand/unix</i>	10-8
Non-clusterized device node names	10-11
The truss (1) Command	10-13
Other Resources	10-15
Lesson Summary.....	10-16
Relevant Commands and Files	10-17
Answers to Exercises, Review Questions, and Labs	10-18

Lesson 11: Tuning an NSC Kernel	11-1
--	-------------

Types of Kernel Tunables	11-6
Autotune.....	11-7
Functions, Interpolation, Extrapolation and Stepping	11-9
Overview of Auto-tuning configuration	11-13
A Real Example	11-17
Overriding Auto-Tuned Parameters.....	11-21
Applying This Knowledge.....	11-21
Performance Monitoring Tools	11-23
The real-time performance monitoring tool, rtpm (1M).....	11-23
The system activity report, sar (1M), program	11-25
NSC-Specific Kernel Tunables.....	11-27
Using id tune (1M) to Build an NSC Kernel	11-28
Lesson Summary	11-29
Relevant Commands and Files	11-30
Answers to Exercises and Review Questions	11-31

1

Course Introduction



Lesson 1

Course Introduction

1.1 Introduction

This ***UnixWare 7 NonStop Clusters Administration*** course is designed for individuals who need to gain a firm understanding of how to manage UnixWare® 7 NonStop® Clusters (NSC). The target audience includes day-to-day administrators of an NSC cluster. The course may also be useful to technicians who install NSC systems and programmers who will develop cluster-aware applications.

1.1.1 Course goal and purpose

At the conclusion of this course, students should be able to:

- Define the term *single system image* (SSI) and describe how it affects system administration
- Use the clusterized SCOadmin managers
- Use the non-clusterized administration tools in a cluster environment, applying them to specific nodes when needed
- Use NSC-specific commands to perform cluster-wide and node-specific administration tasks, including:
 - Starting and stopping individual nodes and the whole cluster, in either a controlled manner or as a result of a node failure (actual or simulated)
 - Migrating a running process from one node to another
 - Filesystem administration including mounting, unmounting, and maintaining protected storage
 - Network administration



- Configure and use the NSC load-leveling feature
- Configure and use the NSC failover daemons
- Configure other aspect of cluster behavior using NSC-specific tools
- Troubleshoot common problems encountered when running applications a clustered environment
- Explain how autotuning and general tuning is accomplished to effect the best possible cluster-wide performance

AV 1-1

The slide is titled "NonStop Cluster Administration" in a blue-bordered box at the top. Below the title is a bulleted list of topics. At the bottom, there is a blue footer bar containing the text "Intro01" on the left and "© The Santa Cruz Operation, Inc." on the right.

- Introduction
- Single System Image (SSI) and NSC Administration
- Basic Process Administration
- Filesystem Administration
- Network Administration
- Startup and Shutdown
- Load Leveling
- Failover
- Configuring NSC Behavior
- Troubleshooting Malfunctioning Applications
- Tuning an NSC Kernel

Intro01 © The Santa Cruz Operation, Inc.

1.1.2 Course Lessons

To achieve the course goals, this course contains the following lessons:

- Introduction (this lesson) — course goals, topics, lengths, and typographical conventions
- Single System Image (SSI) and NSC administration — reviews the concept of SSI and discusses how it affects system administration on a cluster
- Basic process administration — managing processes in a cluster environment; 32-bit PIDs, process migration and kernel objects, and NSC-specific process administration commands
- Filesystem administration — managing the cluster-wide file tree and device name space, the basics of using Veritas ODM for mirroring, and the implications of maintaining protected storage
- Network administration — distinguishing between a physical and virtual IP interface, multi-instance server daemons, and PPP
- Startup and shutdown — starting and shutting down the entire cluster and individual nodes
- Load leveling — balancing process load across the cluster using the load-leveling facility
- Failover — configuring software failover
- Configuring NSC behavior — using the NSC-specific configuration tools to configure certain aspects of NSC's behavior
- Troubleshooting malfunctioning applications — using UNIX tools to troubleshoot problems with applications running in an NSC environment
- Tuning an NSC kernel — configuring autotuning kernel parameters

1.1.3 Target audience

This class is aimed at people who need to become NSC system administrators. NSC installation and support technicians and developers may also find this course useful. Students are likely to have a wide variety of backgrounds in UNIX[®], UnixWare, and SCO OpenServer[™]. You must have the following prerequisite skills before attending this course:

- Passed the UnixWare 7 SCO ACE certification tests
- Passed the NSC Cluster Concepts test
- Have a working knowledge of UNIX shell programming

1.1.4 Course roadmap

UnixWare 7 NonStop Clusters Administration is the third course in a series on SCO NonStop Clusters. The other courses are:

- *UnixWare[®] 7 NonStop Clusters[®] Concepts*
- *UnixWare[®] 7 NonStop Clusters[®] Installation*

1.1.5 NSC Curriculum Hardware Supplement

The *NSC Curriculum Hardware Supplement* provides detailed information about hardware supported by UnixWare[®] 7 NonStop[®] Clusters. Separating the hardware-specific information into another book allows the NSC curriculum guides to remain vendor-independent.

This makes it possible to revise the hardware chapters on a schedule that accommodates changes in hardware support independent of the software release.

Each vendor has a separate chapter in the *NSC Curriculum Hardware Supplement*. The introductory chapter includes some generic hardware information, but is largely used as a placeholder for cross references into the supplement. These cross references point to sections in the supplement, rather than page numbers. This allows training centers to use different types of hardware in their classrooms. The instructor can direct students to the appropriate chapter of the supplement for their classroom configuration.

For example, when you are referred to the section titled “Initializing Nodes,” in your hardware vendor’s chapter in the *NSC Curriculum Hardware Supplement*, look for this section in the supplement table of contents in your vendor’s chapter.

The latest version of the *NSC Curriculum Hardware Supplement* can be found on the SCO web site. At any time after class, you can obtain the latest hardware-related information by downloading the latest version from the Internet.

1.2 Student Materials

This student guide is yours to keep and write notes in. Your instructor might have other reference materials in the classroom for your use while attending this class. Please do not take or write in these materials as they will be needed again for later classes.

1.2.1 The *Student Guide*

The *Student Guide* contains:

- Outline of presentation
- Note-taking space
- Copies of all transparencies

1.2.2 SCO documentation

This course refers to the online documentation using **scohelp**(X1) or a web browser.

1.3 Typographical Conventions

This publication presents commands, filenames, keystrokes, and other special elements as shown here:

lp or lp(1)	Commands, device drivers, programs, and utilities (names, icons, or windows); the number in parentheses indicates the reference manual section in which the command, driver, program, or utility is documented
<i>/new/client.list</i>	Files, directories, desktops (names, icons, or windows), system, network, and printer names
root	User names
<i>arg</i> or <i>arg</i>	Placeholders (replace with appropriate name or value)
[<i>arg</i>]	Surrounds an <i>option</i> or <i>cmdarg</i> that is not mandatory
<i>arg ...</i>	Indicates multiple occurrences of the <i>option</i> or <i>cmdarg</i>
\$	Command-line prompt for commands executable from the Bourne shell by any user
#	Command-line prompt for commands only executable by the root user from the root account
F2	Keyboard keys
CTRL SPACE	Press CTRL and SPACE keys simultaneously
CTRL ALT DEL	Press all three keys, CTRL , ALT , and DEL , simultaneously
CTRL d	Press CTRL key. With CTRL still depressed, press D .
ALT 0240	Press ALT key. With ALT still depressed, press and release in sequence 0 , 2 , 4 , and 0 .
CTRL Q + i	Press CTRL key. With CTRL still depressed, press Q . Release both keys and press I .
CTRL Q + I	Press CTRL key. With CTRL still depressed, press Q . Release both keys and press SHIFT I .
Exit program?	System output (prompts, messages)

tar tv	User input, such as on a command line
“Description”	Field names or column headings (on screen or in database)
Cancel	Button names
Edit	Menu names
Copy	Menu items
File ⇨ Find ⇨ Text	Sequences of menus and menu items
open or open(2)	Library routines, system calls, kernel functions, C keywords; the number in parentheses indicates the reference manual section in which the file is documented
\$HOME	Environment or shell variables
SIGHUP	Named constants, signals, and kernel parameters
“adm3a”	Data values
<i>employees</i>	Database names and tables
buf	C program structures
b_b.errno	Structure members

1.4 Sample Exam Items

These questions are representative exam items on the SCO ACE certification exam for this course. Being able to correctly answer these questions does not guarantee that you will be able to pass the actual exam.

1. Which of these is **NOT** a method used to migrate processes between nodes?

- A. **kill**
- B. **rfork**
- C. **migrate**
- D. **kill3**

2. Which command registers a program with **keepalive**?

- A. **spawndaemon**
- B. **keepalive**
- C. **ka_start**
- D. **keepalive_startup**

3. Which configuration file is used by **load_level**?

- A. */etc/cluster/load_level*
- B. */etc//load_level/parameters*
- C. */etc/cms/load_level*
- D. */etc/cluster/load_level/parameters*

Which statement regarding CVIPs is **TRUE**?

- A. You cannot configure the CVIP as a DHCP client.
- B. The CVIP interface can be aliased to only one virtual IP address.
- C. CVIP addresses must be on the same subnet as at least two NICs in the cluster.
- D. There is a limit of six CVIPs per cluster.

Answers to sample exam items

- 1. A
- 2. A
- 3. D
- 4. C

2

Single System Image (SSI) and NSC Administration



Lesson 2

Single System Image (SSI) and NSC Administration

Introduction

Recall that the concept of *single system image (SSI)* was defined and discussed in the **UnixWare® 7 NonStop Clusters® Concepts** course. In brief, SSI makes the cluster appear as a single computer to its users and administrators.

This lesson reviews the concept of single system image (SSI) and describes how SSI is implemented in NSC, focusing on the implications of SSI for you as an administrator of a cluster.

Objectives

By the end of this lesson, you will be able to:

- Explain the concept of SSI
- Describe how SSI effects namespaces, including process IDs, network addresses, and file names, including device names
- Define how SSI affects startup and shutdown of the entire cluster and of individual nodes
- Discuss NSC limitations resulting from the current implementation of SSI

AV 2-1

Single System Image (SSI) and NSC Administration

- Overview of SSI in NSC and Current Limitations
- Processes
- File and Device Names
- Networking and CVIP
- Startup and Shutdown

SSI01 © The Santa Cruz Operation, Inc.

Topics

In order to meet these objectives, this lesson discusses:

- Overview of SSI in NSC and current limitations — defining SSI in an NSC environment
- Processes — the single-PID space and implications for process management
- File and device names — how SSI affects the file tree and device names
- Networking and CVIP — reviews the concept of a single IP address to represent the entire cluster
- Startup and shutdown — how SSI affects startup and shutdown

Note These topics provide a high-level overview of the administrative issues. The remaining lessons in this course discuss the specific topics in detail.



2.1 Overview of SSI in NSC and Current Limitations

Cluster products that fully implement SSI present a view of the cluster as a single system to users and administrators. This provides several advantages:

- Users and administrators do not have to learn new cluster-related commands
- Programmers do not have to rewrite programs specifically for the clustered environment
- Administrative complexity is reduced

NSC implements SSI in almost every subsystem including processes, filesystems, networking, even startup and shutdown. That is, unless you seek cluster-specific information or services, the system appears to you—programmer, user, or administrator—as a single computer.

Although the SCO NSC implementation of SSI extends to almost every component of the operating system, there are a few limitations. SCO has an on-going effort to eliminate these remaining limitations. Check the ***UnixWare 7 NonStop Clusters Release Notes*** for the most current information. For NSC 7.1.1, these limitations apply:

Kernel object migration	When a program running in a process creates a kernel object, for example a socket or file descriptor, the object is created on the node where the object is created. If the process migrates to another node, the object remains on the node where it was created. If the node containing that object goes down, the object becomes unavailable. This can obstruct communication between network programs, cause file I/O to fail, and so on.
-------------------------	--

- IP only Currently, NSC only supports the IP protocol stack. This is generally not a problem as most programs either use IP directly, or use a protocol on top of IP such as NETBEUI or TCP. VisionFS is an example of an application that uses TCP/IP and therefore works with NSC.
- However, it does mean that IPX/SPX is not supported. Therefore, NetWare services are not supported by NSC.
- Only one CVIP per sub-net You can have only one CVIP per subnet to which the cluster connects. This means that all traffic inbound to a cluster on a particular subnet will be routed through a single LAN NIC. The LAN NIC serving the CVIP can be changed so there is no single point of failure. But, the LAN NIC currently acting as the CVIP receives all traffic targeted to the cluster for that subnet.
- This can be a bottleneck for network applications with heavy in-bound traffic.

Note Web and most database applications usually generate much more outbound (as apposed to inbound) network traffic. Therefore, this limitation does not generally affect those applications.

CVIPs are discussed further in lessons throughout this course.

- X Server only on the root node Currently, the X Server can run *only on the root node*.
- However, X clients can run on any node. For example, you can start multiple **xload**(X1) programs, each running on a different node. They can display the load on that node using an X Server running either on node 1, an independent machine, or via a browser and the webtop.

- | | |
|---------------------------|--|
| One node controls storage | Currently, storage can only be protected by two nodes. Further, of those two nodes, only one node performs all I/O for that storage. Thus, the protecting nodes are in an active/passive configuration.

With external shared storage, this means slight overhead because each I/O request must first go over the SAN. With CNM, this means even more overhead on each <i>write</i> because they must not only be performed on the current root node, but also on the other node protecting the storage. |
| VxVM on root nodes only | The Veritas Volume Manager can only be used to manage protected storage on the root nodes. It cannot be used to protect storage attached to other dependent nodes in a cluster. |

Exercise: Review Known Limitations

Read the “Known Limitations” section of the ***UnixWare 7 NonStop Clusters Release Notes*** for the version of NSC used in your classroom. These are either provided by your instructor, or viewed using a browsers from the NSC installation CD-ROM.

2.2 Processes

The purpose of SSI is to make the cluster appear as a single computing resource. But processes run on individual nodes in the cluster. In order to implement SSI for processes, NSC has these process-related customizations:

- Processes are assigned a cluster-wide process ID (PID) when they are created.
- The PID of a process includes the node number on which it was created (called the *origin node*). The PID is now a 32-bit number, with the high-order 16 bits indicating the origin node and the least significant 16-bits containing a node-unique number in the traditional manner of UNIX PIDs.

For example, on a non-NSC system, a process ID might be 1805 decimal, or 11100001101 binary. The equivalent PID on NSC started on node 2 would have a PID of 132877 decimal. That is, $1805 + 131072 (2^{17})$, because the 2nd of the high-order 16 bits is set for processes created on node 2.

- The PID does not change if a process migrates to another cluster. This maintains the UNIX paradigm that PIDs remain constant throughout the life of a process.
- The NSC kernel has a mechanism for finding any process on any node, regardless of its origin node, or where it has migrated, even if the origin node is lost.
- There is a cluster-wide view of all existing processes viewed with traditional UNIX tools such as the **ps(1)** command

Exercise: Viewing processes, cluster-wide

Use this exercise to observe how all processes are visible from all nodes, and how their PIDs are unique.

- Step 1. Login to the console of node 1 as **root**.
- Step 2. Change directory to `/labs/lab2`.
- Step 3. Start the **spinner** program on node 1. Enter:
- `./spinner`**
- Step 4. Login to the console of node 2 as **root**.
- Step 5. Change directory to `/labs/lab2`.
- Step 6. Start the **spinner** program on node 2. Enter:
- `./spinner`**
- Step 7. Login as **root** on a multiscreen on node 1.
- Step 8. Run **ps** to see your processes. Enter:
- `ps -ef | grep spinner`**
- Step 9. What are the PIDs of each process? What is the numeric difference between the PID of the two processes?
- Step 10. Kill the processes using the **kill(1)** command.

In addition to providing SSI for traditional processes manipulation, NSC also provides cluster-specific features for managing processes. These include:

- Failover — allows you to configure NSC to monitor an application and, if it fails (either because the node it was running on failed, the software aborted, or it was killed), restart it. Failover is somewhat like the **init**(1M) program. It is discussed in detail in Lesson 8: “Failover.”
- Load leveling — allows you to configure NSC to balance the load of selected applications among specific nodes. It is discussed in detail in Lesson 7: “Load Leveling.”
- Migration, pinning, and other generic process-related commands — which are discussed in detail in Lesson 3: “Basic Process Administration.”

2.3 File and Device Names

Since the goal of SSI is to make the cluster look like a single computing resource, and since disk, filesystems, and so on, are attached to at least two nodes, the NSC implementation of SSI for I/O has several aspects. These include:

- There is a single, cluster-wide file tree consisting of all filesystems mounted from all nodes cluster-wide.
- All device nodes for devices on all cluster nodes are accessible from every cluster node

This means device node names must be clusterized to ensure they are unique. For example, if you have two nodes, each with a parallel port, they can not both call their device nodes `/dev/lp0`.

As a rule, `.#` is appended to all device non-cluster device node names, where `#` is the number of the cluster node where the device is physically attached. For example, for two parallel ports, one each on nodes 1 and 2, the device names are `/dev/lp0.1` and `/dev/lp0.2`, respectively.

Exercise: Listing device nodes for the standard serial ports

Step 1. Log into the cluster as **root**.

Step 2. List the device nodes for the standard serial ports by entering:

```
ls -l /dev/term/*
```

Which names are used for the device nodes of the serial ports on node 1?

The exceptions are devices related to the storage device interface (SDI) kernel subsystem, including disks and HBAs. These devices have `n#` prepended to the non-cluster device node name, where `#` has the same meaning as above. For example, the first disk on the second node has a device node name of `n2c0b0t0d0s0`.

The */stand* directory is now a normal directory in the root filesystem. It contains several subdirectories, one for each node, named by node number, where the *stand* filesystems of the respective node is mounted.

Exercise: Listing the contents of /stand

Step 1. Log in as **root**.

Step 2. List the sub-directories of the */stand* directory. Enter:

ls -l /stand

How many subdirectories are listed?

Do all of the subdirectories with numbers for names have entries in the mount table as reported by the **mount(1M)** command?

Information on managing filesystems in a clustered environment is presented in Lesson 4: “Filesystem Administration.” Information on the impact of clusterized device node names is presented in Lesson 10: “Troubleshooting Applications.”

2.4 Networking and CVIP

To make an NSC cluster appear as a single computing resource on the network, the NSC implementation of SSI affects networking in these ways:

- NSC uses one CVIP for each subnet to which the cluster is attached.
- A CVIP creates a virtual network interface for a given subnet that:
 - Attaches itself to a physical LAN NIC on the subnet
 - Dynamically detaches from the current LAN NIC and attaches to a different LAN NIC (on the same subnet) if the node with the current LAN NIC is lost
 - Uses ARP to indicate to which physical LAN NIC it is currently attached
- If a cluster is attached to several subnets, you may configure one CVIP per subnet
- You do not have to use CVIPs. However, alternative schemes to present the cluster as a single resource to the network all have a single point of failure in their design.
- Network interfaces are visible cluster-wide. Their interface names follow the same naming convention as most device nodes in NSC. That is, the names have a `.#` appended to them, where `#` is the cluster node on which the interface physically exists.

Exercise: Viewing the network configuration

Step 1. Log in as **root**.

Step 2. List the network interfaces cluster-wide. Enter:

ifconfig -a

Step 3. List the MAC addresses of all network interfaces. Enter:

arp -a

How many nodes physical interfaces are there and what are their names? To which NIC is the CVIP mapped?

Detailed information on how CVIPs work and how they are managed is presented in Lesson 5: “Network Administration.”

2.5 Startup and Shutdown

Since the purpose of SSI is to make the cluster appear as a single computing resource, some commands have been modified to operate on the entire cluster rather than just on the node on which the command is run. Just as **ps(1)** displays a list of all processes cluster-wide, the **shutdown(1M)** command now shuts down all nodes, cluster-wide.

Use the cluster-specific command, **clusternode_shutdown(1M)**, to bring down a specific cluster node.

Similarly, the startup procedure has been significantly modified for clusters. For example:

- On a non-cluster system, there is only one **init(1M)** daemon running. This is also true of clusters.
- The boot procedure has been modified to only start **init** on the current root node.
- If the current root node fails, during cluster regroup, the **init** program is restarted on the new root node.
- Programs started by **init** that were running on a node that fails are restarted on the new root node.

Detailed information about managing the startup and shutdown process is provided in Lesson 6: “Startup and Shutdown.”

2.6 Lesson Summary

This lesson reviewed the concept of SSI and provided an overview of the implications of the NCS implementation of SSI to you as an administrator. Specifically, this lesson discussed SSI with respect to:

- Processes
- Files, filesystems, and device nodes
- Networking
- Startup and shutdown

This lesson also presented that current limitations of NSC's implementation of SSI.

Review questions

1. How do PID numbers differ for processes started on node 1 versus those started on node 2?
2. What happens if you run the **shutdown** command from a shell running on node 2?
3. What is the device node name for the first disk drive on controller 0 in node 4?
4. If you have a cluster with two subnets, how many CVIPs can you have?

2.7 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

clusternode_shutdown(1M)	Shuts down the specified cluster node
ifconfig(1Mtcp)	Displays and configures network interface parameters
init(1M)	Initializes processes during boot
kill(1)	Send a signal to a process
ls(1)	Lists the contents of a directory
mount(1M)	Mounts or unmounts filesystems
onnode(1)	Executes a command on a specific node
ps(1)	Reports process status
shutdown(1M)	Shuts down the cluster

Relevant files

<i>/stand</i>	Contains directories for each node in the cluster with the kernel files found in <i>/stand</i> on a standalone system
---------------	---

2.8 Answers to Exercises and Review Questions

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Exercise answers

“Review Known Limitations,” on page 2-9

Known limitations related to SSI in NSC 7.1.1 include:

- Only one CVIP per subnet
- X Server runs only on node 1
- Protected storage is controlled by only one node
- VxVM can only manage protected storage on root nodes

“Viewing processes, cluster-wide,” on page 2-11

The PIDs on node 1 are greater than those on node 2 by (2^{16} + a 16-bit number).

The PIDs on node 2 are always greater than 131072 (2^{17}) because the 2nd of the high-order 16 bits (that is, the 18th bit) is set to indicate the node number. So PIDs on node 2 are 2^{17} + the a 16-bit number..

“Listing device nodes for the standard serial ports,” on page 2-13

The device node names for the serial ports on node 1 are `/dev/term/00.1` and `/dev/term/00.2`.

“Listing the contents of /stand,” on page 2-14

By default, the `/stand` directory contains one directory for each node in your cluster.

“Viewing the network configuration,” on page 2-15

The **ifconfig** command displays:

- One loopback interface—`lo0`
- One entry for each PIP in the cluster—`net $n$ . $N$` , where n is the interface number and N is the node number
- One entry for each CVIP created—`cvip0` for the default CVIP interface created during installation

For example, on a two node cluster with one NIC in each node, these network interfaces are displayed by **ifconfig**:

```
lo0
cvip0
net0.2
net0.1
```

Review Question Answers

This section contains the answers to the review questions on page 2-17.

1. The PIDs on node 1 are always greater than 65536 because the 1st of the high-order bits (that is, the 17th bit) is set to indicate node number 1. So PIDs on node 1 are 2^{16} + the a 16-bit number.

The PIDs on node 2 are always greater than 131072 (2^{17}) because the 2nd of the high-order 16 bits (that is, the 18th bit) is set to indicate the node number. So PIDs on node 2 are 2^{17} + the a 16-bit number.

2. The **shutdown** command always shuts down the entire cluster. It must be run from the `/` directory.
3. The device node name for the first disk on controller 0 in node 4 is `n4c0b0t0d0s0`.
4. If you have a cluster with two subnets, you can have up to two CVIPs (as of release 7.1.1).

3

Basic Process Administration



Lesson 3

Basic Process Administration

Introduction

The NSC implementation of single system image (SSI) allows processes to start on, and move to, any node in a cluster. Sometimes it is desirable to start processes on specific nodes or prevent their migration to other nodes.

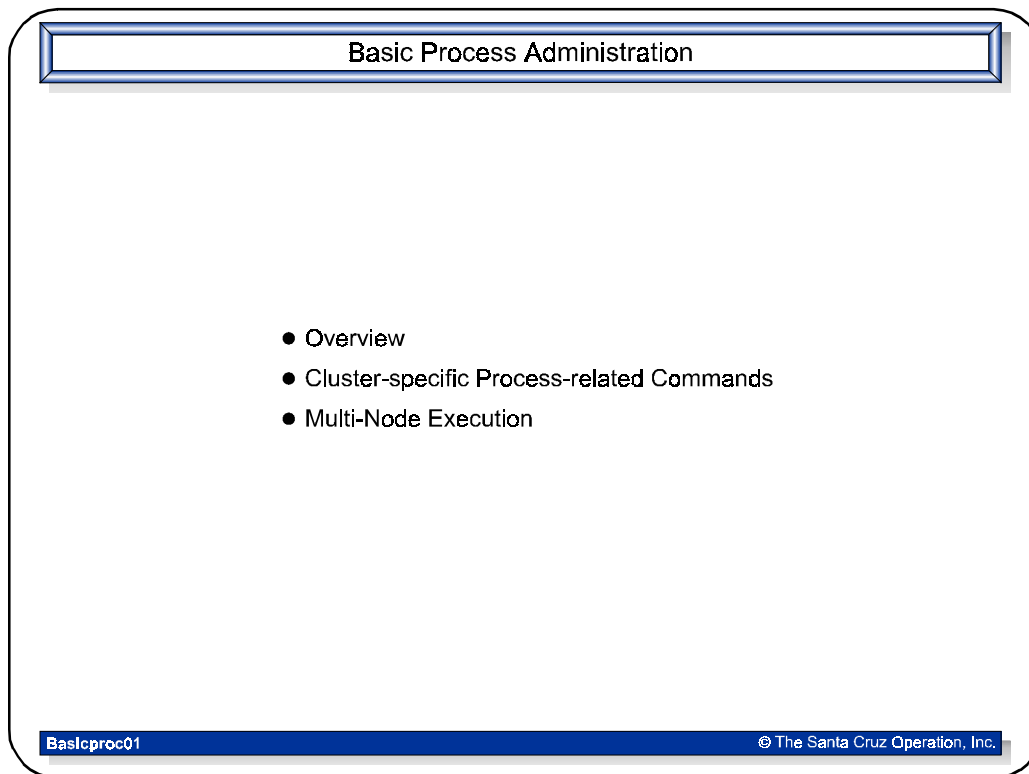
This lesson teaches you how to administer processes in this environment.

Objectives

By the end of this lesson, you will be able to:

- Define the terms *origin node* and *surrogate-origin node*
- Explain how the cluster presents a single, cluster-wide list of processes
- Name the NSC facility that automatically restarts failed processes
- Name the NSC facility that automatically migrates processes
- Identify and use the commands to manage processes on multiple nodes
- Explain the impact of process migration on kernel objects

AV 3-1



Topics

In order to meet these objectives, this lesson discusses:

- Overview — a general overview of processes in NSC
- Cluster-specific process-related commands — how to use commands to start, migrate, and kill processes in a cluster
- Multi-node execution — issues related to processes running on multiple nodes, IPC, migration, and kernel objects

3.1 Overview

One of the key differences between NSC and failover-only solutions such as Reliant HA or Microsoft Wolfpack, is that on NSC, all nodes run processes. Some other clustering solutions, such as Turbo Linux, allow processes to run on all nodes but you must manage processes on each node separately. The SSI implementation on NSC *provides a cluster-wide view of processes*.

In addition to allowing all nodes to run processes, NSC allows you to control where a process runs, either manually or automatically. In addition, the NSC failover subsystem allows you to have NSC monitor your processes and restart them if they fail for any reason, including a node loss.

These topics are discussed in more detail after you have learned the basics of process management on an NSC cluster.

- Section 3.2, “Cluster-Specific Process-Related Commands,” on page 3-11
- Lesson 7: “Load Leveling”
- Lesson 8: “Failover”

3.1.1 The global process view

Recall that on non-cluster UNIX systems, the kernel keeps a process table or process list with one entry per active process on the system. They often have a */proc* filesystem that provides application-level views of the process list.

NSC implements a clusterized */proc* filesystem. This means that utilities like **ps(1)** see all processes, cluster-wide, at all times.

To implement this cluster-wide view:

- The NSC kernel on each node keeps a *virtual process list* on each node. The virtual process list for a given node, *N*, contains:
 - An entry for every process started on node *N* that is still running. It can be running on any node in the cluster. For this process, node *N* is referred to as the *origin node*.
 - An entry for every process running on node *N*, regardless of which node originated the process.
 - An entry for every process related to one of the above, including parent, child, process-group leader, and so on.

The virtual process information on the origin node includes information about where the process is currently running.

- Process IDs (PIDs) now have two parts. The most significant portion contains the number of the origin node. The least significant portion contains a number unique to the node.

Processes are accessed in this manner (when using **kill**(1) to send a signal, for example).

- Look in the virtual process list to determine if the process is executing on the local node.
- If the process is not executing on the local node, use the PID to determine the origin node.
- Contact the origin node using a SAN message to determine where the process is running.
- Send a message to that node by way of the SAN

Whenever node *N* leaves the cluster:

- If *N* is the root node, A SAN event is trigger and causes election of a new root node.
- A *surrogate-origin node* (the current root node) gathers information from all remaining nodes for each currently-running process that originated on node *N*.
- The surrogate node makes one **vproc** entry in its own list for each of these processes which includes current information about where the process is currently running.
- The other nodes know that node *N* is down, and use the surrogate node instead of node *N* to obtain information about processes that originated at node *N*.

Whenever node *N* re-joins the cluster:

- The surrogate propagates the surrogate **vproc** entries back to node *N*.
- The other nodes in the cluster note that node *N* is back and to contact it for information about processes it created.

The creation and migration of **vproc** entries (during a node down and node up) occurs when a cluster is reformed. While reforming, no other process-related operations can occur. This is why the cluster appears to pause when a cluster goes down or comes back up.

3.1.2 Process-related user-level NSC concepts

There are several NSC-specific concepts related to controlling processes at the user/administrative level. Learn these concepts to better understand subsequent discussions about controlling processes in NSC.

- Migrate* When a process moves from node N to node M, it is said to *migrate* to node M. In the current version (UnixWare® 7 NonStop® Clusters), the whole process migrates together. That is, the light-weight processes (LWPs) of a process may not be executing concurrently on different nodes.
- In NSC, migration occurs through a NSC-specific signal, **SIGMIGRATE**, number 35. Unlike other signals, delivery of this signal must be accompanied by additional information, the node to which the process is migrated.
- Pinned* A process that ignores the **SIGMIGRATE** signal is said to be *pinned* to its current node. That is, it can not be migrated.
- Load* Several NSC tools place a process on the least *loaded* node, either at startup or through migration. In this context, *load* refers only to CPU load. It is not based on I/O, RAM utilization, or any other resource.
- Overloaded* A node that has a CPU load significantly greater than the other nodes in the cluster. What constitutes *significantly* is configurable in certain ways.
- Underloaded* The opposite of overloaded. That is, a node that has a CPU load significantly lower than the other nodes in the cluster.

AV 3-2

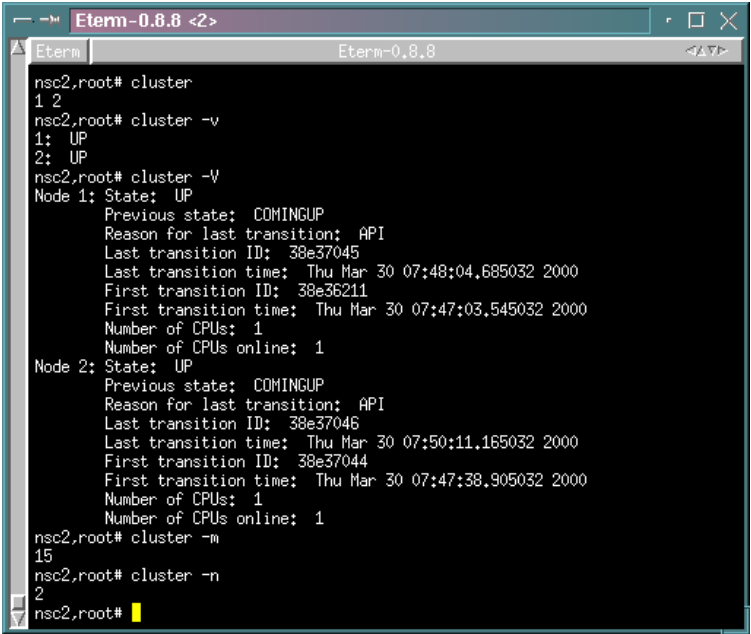
Cluster-Specific Process-Related Commands

- **cluster(1)**
- **onall(1)**
- **onnode(1)**
- **node_self(1)**
- **clusternode_num(1)**
- **where(1)**
- **fast(1)**
- **fastnode(1)**
- **kill3(1)**
- **migrate(1)**

Baslproc02© The Santa Cruz Operation, Inc.

AV 3-3

The `cluster(1)` Command



```
nsc2.root# cluster
1 2
nsc2.root# cluster -v
1: UP
2: UP
nsc2.root# cluster -V
Node 1: State: UP
    Previous state: COMINGUP
    Reason for last transition: API
    Last transition ID: 38e37045
    Last transition time: Thu Mar 30 07:48:04.685032 2000
    First transition ID: 38e36211
    First transition time: Thu Mar 30 07:47:03.545032 2000
    Number of CPUs: 1
    Number of CPUs online: 1
Node 2: State: UP
    Previous state: COMINGUP
    Reason for last transition: API
    Last transition ID: 38e37046
    Last transition time: Thu Mar 30 07:50:11.165032 2000
    First transition ID: 38e37044
    First transition time: Thu Mar 30 07:47:38.905032 2000
    Number of CPUs: 1
    Number of CPUs online: 1
nsc2.root# cluster -m
15
nsc2.root# cluster -n
2
nsc2.root#
```

Baslproc03© The Santa Cruz Operation, Inc.

3.2 Cluster-Specific Process-Related Commands

In addition to the traditional process-related UNIX commands, NSC has several commands that allow you to manually control where a process is executing within a cluster, or to determine on which node you want to execute a process.

Each of these commands are discussed in their own subsections below.

3.2.1 The **cluster(1)** command

Use the **cluster(1)** command as shown in AV 3-3 on page 3-10 to determine:

- The number of the nodes that are currently up
- The number and state of the nodes in the cluster
- Verbose information about each node, including state, last state, number of CPUs, times of transition, and so on
- The maximum number of nodes supported by the version of NSC in use
- The number of CPUs online in each node

This information can be useful in many ways, especially in shell scripts. For example:

- To distribute tasks to nodes based on the number of CPUs in each node
- To determine the state of a particular node, so you do not try to start processes on nodes that are not up

Exercise: Using the **cluster(1) command**

Determine when each of the nodes in your cluster made their last transition

3.2.2 The **onall(1)** and **onnode(1)** commands

Once you have determined on which node to execute a command, use the **onnode(1)** command to start a command on that node. The syntax is similar to the **nohup(1)** command. Enter the command you would normally use preceded with the **onnode** command. For example, to start a new Korn shell with **vi** history on node 2, enter:

```
onnode 2 ksh -o vi
```

You can also use **onnode** with the **-p** flag to *pin* a process to a node. This means that it will ignore signals from the NSC load-leveling daemon to move to another node. For example:

```
onnode -p 2 ksh -o vi
```

You can un-pin a pinned command with the **-P** flag to **onnode**.

Exercise: Using the **onnode(1) command**

Use the **onnode** command to start **ps(1)** and pin it to node 2.

Use the **onall(1)** command to run a command on all nodes within a cluster. This command provides you with considerable flexibility through various command-line arguments, as documented in the **onall(1)** manual page.

Exercise: Using the **onall(1) command**

After reviewing the **onall(1)** manual page, use the command to run a **sync(1)** command on all nodes of the cluster.

3.2.3 The `node_self(1)`, `clusternode_num(1)`, and `where(1)` commands

Use the **`node_self(1)`** or **`clusternode_num(1)`** commands to determine the node on which a process is currently running. This is useful for determining whether to migrate your process to another node. You might want to do this for several reasons:

- Migrate to the fastest node to get the best CPU performance. See Section 3.2.4, “The `fast(1)` and `fastnode(1)` commands,” on page 3-15.
- Migrate to the node that is serving a particular set of protected storage. You can find out which node is serving the storage using the **`where(1)`** command. This generally has the following effect:
 - It greatly increases the performance of I/O-intensive processes by eliminating calls across the SAN for I/O
 - It increases overall cluster performance by decreasing SAN traffic

Exercise: Using cluster commands together

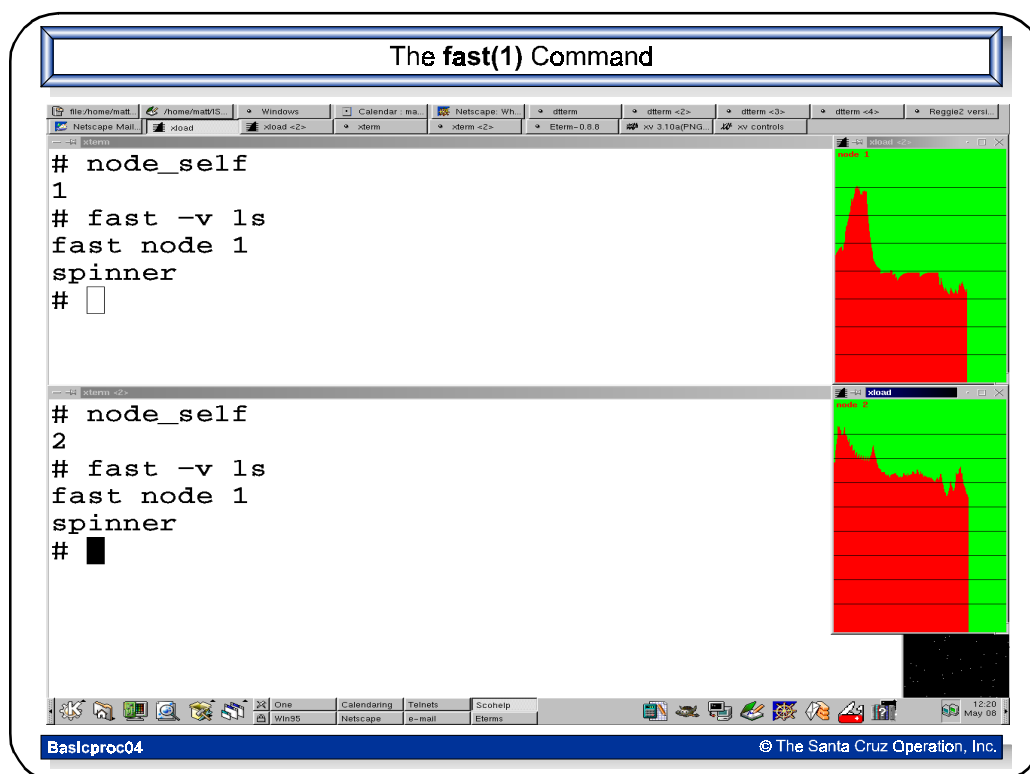
- | | |
|---------|--|
| Step 1. | Log in as <code>root</code> . |
| Step 2. | Determine which node is serving the <code>/labs/lab3</code> directory.
Enter:

<code>where /labs/lab3</code> |
| Step 3. | Determine the node on which you are running. Enter:

<code>node_self</code> |
| Step 4. | Use the <code>time(1)</code> , <code>dd(1)</code> , and <code>onnode(1)</code> commands to time the difference in reading the <code>/labs/lab3/meg10</code> file from the node that is serving the <code>/labs/lab3</code> directory and from a node that is not.

Send the output of <code>dd</code> to <code>/dev/null</code> and block the reads 1MB per block. |

AV 3-4



3.2.4 The **fast(1)** and **fastnode(1)** commands

Use the **fast(1)** and **fastnode(1)** commands to manually balance load across a cluster.

In order for these commands to function, the load-leveling daemon (**load_leveld**) must be running because these commands use information generated by this daemon to determine which node is currently the fastest.

Your instructor has already turned on **load_leveld** for this lesson. Lesson 7: “Load Leveling” discusses various ways of activating **load_leveld**.

AV 3-4 on page 3-14 shows a desktop and illustrates the effect of using the **fast** command. This slide shows:

- A terminal emulation window running on nodes 1 and 2. The output of the **node_self** commands shows that the terminal emulation window on top is node 1 and the bottom window is node 2.
- The **xload** client running on each node is shown next to their respective terminal windows. Note the multiple lines on the graph for node 2, indicating it has a greater load.
- Within each window, the **fast** command is used to run **ls**. The **-v** flag causes the **fast** command to indicate which node will run the command

Note Regardless of which node runs the fast command, it detects node 1 as the fastest (least CPU load) node, and runs the ls command on node 1.

For information on automatically balancing loads across a cluster, see Lesson 7: “Load Leveling.”

3.2.5 The **kill3(1)** and **migrate(1)** commands

As discussed in Section 3.1.2, “Process-related user-level NSC concepts,” on page 3-9, you can force process migration using the **SIGMIGRATE** signal (number 35). That is:

- You tell a process to migrate by sending it a **SIGMIGRATE** signal.
- The kernel migrates a process by performing default processing on the **SIGMIGRATE** signal.
- As with most signals, processes can ignore the **SIGMIGRATE** signal, preventing migration they do not initiate themselves.

Unlike other signals, **SIGMIGRATE** requires additional information—the node to which the process will migrate. Since the **kill(1)** command does not allow you to provide this additional information, NSC provides the **kill3(1)** command.

Use the **kill3** command to send any signal (currently only **SIGMIGRATE**) that requires an argument. The syntax is:

```
kill3 [-signame | -signum] argument pid ...
```

For example, to tell the process with PID 3030 to migrate to node 2, enter:

```
kill3 -SIGMIGRATE 2 3030
```

As an alternative to the **kill3** command, use the **migrate** command. The syntax is:

```
migrate node PID ...
```

For example, to tell the process with PID 3030 to migrate to node 2, enter:

```
migrate 2 3030
```

Each of these methods yield the same result

Exercise: Putting process-management commands together

The file in the `/labs/lab3/dd_on_tmp` is a shell script that uses most of the commands discussed thus far.

- Step 1. Log into the cluster as **root**.
- Step 2. Change to the `/labs/lab3` directory.
- Step 3. View the `dd_on_tmp` file using the utility of your choice. Step through the program with your instructor.
- Step 4. Run the program from shells running on various nodes within the cluster and observe the behavior.

3.3 Multi-Node Execution

The ability to run processes on multiple nodes concurrently is a powerful tool. To make most efficient use of this feature, you must understand the consequences of process migration. Then you can decide when you get the most benefit for the costs incurred when moving processes between nodes.

3.3.1 IPC overhead

Programs with multiple processes that make extensive use of UNIX System V inter-process communication (IPC), such as shared memory or semaphores, are not well suited to run on multiple nodes concurrently. This is because:

- IPC operations generate significant SAN traffic.
- IPC operations are synchronous. They take much longer—by orders of magnitude—to execute over the SAN versus locally.

Example applications of this type include database servers.

To determine if a multi-process application utilizes these facilities, enter:

```
egrep "shmget|semget" progname ...
```

3.3.2 Migrating kernel objects

Recall from Section 2.1, “Overview of SSI in NSC and Current Limitations,” on page 2-7 that *kernel objects* are data structures created by the kernel to keep track of resources used by a process. Kernel objects include:

- Semaphores
- System V shared memory segments
- File descriptors
- Sockets

For example, a web browser will create and use sockets. Sockets are kernel objects that keep track of:

- The two end-points of a conversation over the Internet (for example, the IP address of your computer and that of the web server being contacted)
- Which protocol the conversation is using (for example, TCP/IP or UDP/IP)

The kernel of each node in an NSC cluster holds the kernel objects that are created by processes that run on that node. If a process migrates, some kernel objects migrate with the program, some do not. This has two important implications:

- Use of non-migrating kernel objects after a process migrates involve SAN transactions. This increases the cost of using these kernel objects.
- If the node containing a non-migrating kernel object fails, the kernel object is lost. The process survives because it is executing on another node, but it cannot utilize the lost kernel objects, even if the lost node rejoins the cluster.

For example:

- Stop the **load_level** process using **kill**. If there are multiple **load_level** process, kill the one with PPID 1.
- Start a telnet session on node 2 to a host outside the cluster.
- Migrate the telnet session to node 1.
- Turn off node 2 to simulate a node loss.
- The telnet session would not be able to proceed, because the socket would not be able to access the socket left on node 2.

3.4 Lesson Summary

This lesson presented information on managing processes, both in terms of how the cluster allows utilities to work in a cluster-transparent manner, and in terms of the additional functionality provided by the NSC-specific process-related commands. Specifically, this lesson discussed:

- An overview of NSC processes including the global view of processes and NSC-specific process-related concepts
- NSC-specific process-related commands
- Issues related to running process on, and migrating processes to, multiple nodes

Review questions

1. Define the term surrogate origin node.
2. Which NSC facility automatically restarts failed processes?
3. Which NSC facility automatically migrates processes?
4. Give an example of an application you would *not* want to migrate and explain why.

3.5 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

cluster(1)	Displays information about nodes in the cluster
clusternode_num(1)	Displays the current node number
dd(1)	Converts and copies files
fast(1)	Executes a command on the least-loaded node
fastnode(1)	Determines the least-loaded node
kill3(1)	Sends SIGMIGRATE with a node number argument to a process or process group
migrate(1)	Sends SIGMIGRATE with a node number argument to a process or process group
node_self(1)	Prints the current execution node
nohup(1)	Run a command and ignore signals to hang-up
ps(1)	Reports process status
onall(1)	Executes a command on all nodes
onnode(1)	Executes a command on a specific node
sync(1M)	Updates the super block
time(1)	Times a command
where(1)	Determines the storage node of a specified file

Relevant files

<i>/dev/null</i>	Is a null file used as a placeholder for commands or pipes requiring a file name
<i>/proc</i>	Contains the special filesystem with information about active processes

3.6 Answers to Exercises and Review Questions

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Exercise answers

“Using the cluster(1) command,” on page 3-11

The answer to this exercise depends on how many nodes are in your cluster. Your instructor will provide you with that number.

“Using the onnode(1) command,” on page 3-12

The command is:

```
onnode -p 2 ps
```

“Using the onall(1) command,” on page 3-12

The command is:

```
onall sync
```

“Using cluster commands together,” on page 3-13

The commands used to perform this exercise depend on the node that is serving the `/labs/lab3` directory. The answers below are written as if node 1 is serving said directory:

```
onnode -p 1 time dd if=/labs/lab3/meg10 of=/dev/null bs=1024k
onnode -p 2 time dd if=/labs/lab3/meg10 of=/dev/null bs=1024k
```

“Putting process-management commands together,” on page 3-17

You should notice that, on all but one node, the program must migrate.

Review Question Answers

This section contains the answers to the review questions on page 3-20.

1. The surrogate origin is the node that takes the place of the origin node, if the origin node fails. The surrogate origin node is used to obtain information about processes that originated on that failed node.
2. The NSC failover facility automatically restarts failed processes by way of the **keepalive** daemon.
3. The NSC load level facility automatically migrates processes based on CPU load.
4. Any application that makes heavy use of kernel objects should not be subject to migration.

4

Filesystem Administration



Lesson 4

Filesystem Administration

Introduction

The *SCO UnixWare 7 Administration II* course presented information on how to administer a filesystem on a standalone host. Most of the information presented in that course is directly applicable to administering filesystems in a cluster.

This lesson discusses how filesystem administration is different between a stand-alone host and a cluster, and provides procedures for managing filesystems.

Objectives

By the end of this lesson, you will be able to:

- Describe how the **mknod(1M)** command has changed and is used to create clusterized device nodes
- Explain the mechanisms that enable failover-capable mounts of protected storage
- Provide basic information on using Veritas facilities for creating mirrored volumes between nodes

AV 4-1

Filesystem Administration

- Overview of Files, Filesystems, and Device Nodes in NSC
- Filesystem Administration In NSC

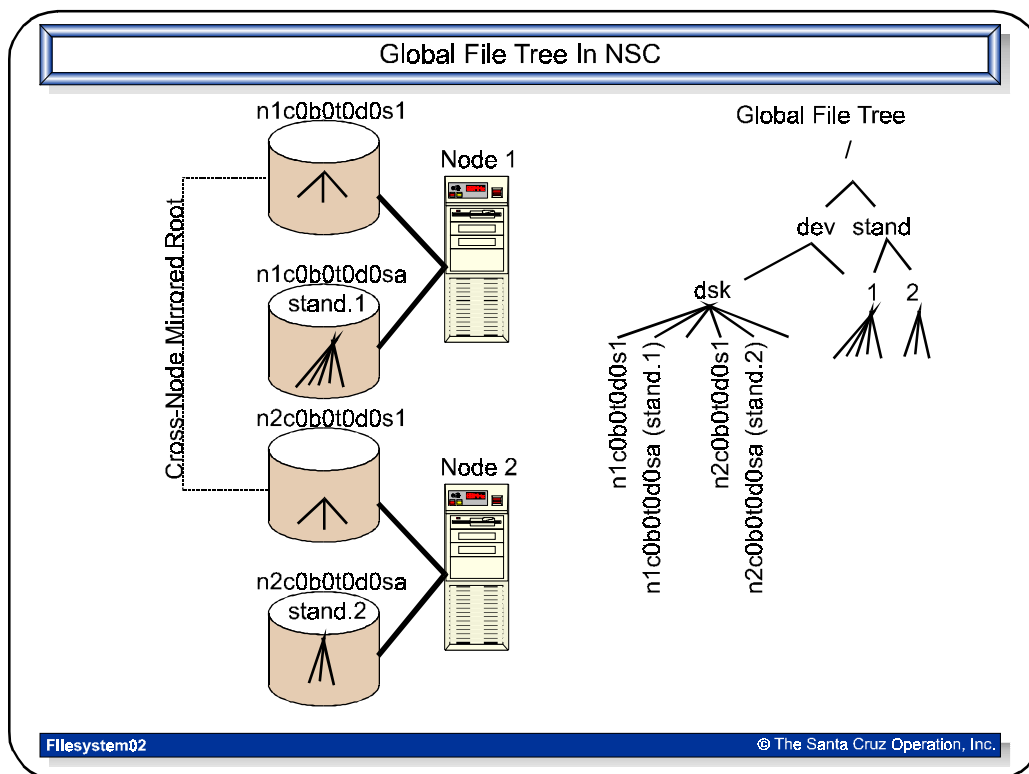
Filesystemc01 © The Santa Cruz Operation, Inc.

Topics

In order to meet these objectives, this lesson discusses:

- Overview of Files, Filesystems, and Device Nodes in NSC — how files, filesystems, and device nodes are different in NSC than in non-NSC environments
- Filesystems Administration in NSC — administering filesystems in a NSC environment
- Protected Filesystems — adding a protected user filesystem

AV 4-2



4.1 Overview of Files, Filesystems, and Device nodes in NSC

Three driving forces cause the administration of filesystems in NSC to be somewhat different from what you learned in the **UnixWare 7 Administration II** course. AV 4-2 on page 4-6 helps illustrate these points:

- The requirement for high availability implies redundancy. In terms of filesystems, this means you must have *protected storage*.

AV 4-2 on page 4-6 shows a typical CNM system where the *root* filesystem is mirrored between the two nodes. Note that the two nodes' *stand* slices are not protected. This is because each node must have its own *stand* slice, which contains its own copy of the UNIX kernel and node-specific boot parameters.

- The need for high availability also requires being able to perform online operating system upgrades while the cluster is running. This is accomplished with the Online System Upgrade (OSU) subsystem discussed in the **UnixWare® 7 NonStop® Clusters Administration Guide**.

Since OSU is not currently supported for CNM systems, there is no depiction of an OSU slice in the slide. However, if you want to use OSU, you must have a separate slice on the shared storage connected to the potential root node large enough to hold a copy of the *root* filesystem.

- The need for a single system image (SSI) requires that:
 - There is a single, unified file tree using all filesystems available on all nodes. This is discussed in more detail in Section 4.2, "Filesystem Administration in NSC," on page 4-11.

In AV 4-2 on page 4-6, you see a single file tree made up of all of the files and directories on all of the disks, including a mount point for each stand slice.

- All devices, regardless of the node to which they are physically attached, appear as a unified set of devices, accessible from every node in the cluster.

In AV 4-2 on page 4-6, you see each device node has a unique name, and all appear under the */dev/dsk* (and */dev/rdisk*) directory.

In addition to a unique device-node naming convention, the major/minor number scheme has changed to support NSC. Major/minor numbers are really viewed by the kernel as a single number.

- On non-NSC systems, it contains only two parts; the major and minor numbers.
- In NSC environments, it contains three parts; node , major, and minor numbers.

To reflect this change, the **mknod** command has been clusterized by adding an optional **node _number** argument at the end of the command line. Depending on the version of the NSC documentation, this may or may not be reflected in the **mknod(1M)** manual page, but is always available from the usage statement printed by the command. To view the syntax from the usage statement, enter (as **root**):

mknod

In order to use the new **mknod** command, you must know the device type and major and minor number of the device node you want to create. Then, invoke the **mknod** command with the appropriate arguments.

Exercise: Creating device nodes for extra floppy drives

In some versions of NSC, the installation does not create floppy device nodes for floppy drives on cluster nodes other than node 1. To remedy this situation and make the additional floppy drives available for your use:

Step 1. Login to the cluster as **root**.

Step 2. Find the major/minor number and device type of the equivalent device node on node 1. For example, to find the major/minor numbers of `/dev/rdisk/f03h`—the 3.5 1.44MB floppy drive 0) on cluster node 1—enter:

```
# ls -l /dev/rdisk/f03h
```

This is the output:

```
crw-rw-rw-  2  root sys  513,112 Apr  8 1998
/dev/rdisk/f03h
```

The numbers you want are where the size would normally be. In this case, they are 513, 112. The device type is the first character in the output of the command. In this case, **c**.

Step 3. Subtract 512 from the major number. This is the true major number of the device node.

Step 4. Make the new device node on cluster node 2. Enter:

```
mknod /dev/rdisk/f03h.2 c 1 112 2
```

Step 5. Set permissions to read and write for all:

```
chmod 666 /dev/rfd0135ds18.2
```

Step 6. Test the new floppy device node with **tar** or **cpio**.



4.2 Filesystem Administration in NSC

The NSC-specific filesystem administration issues are related to the requirement for protected storage. The unified view of the file tree is built into the NSC kernel implementation. Further, the protection of the *root* filesystem and any OSU slice is configured during installation. Thus, the only remaining issue is adding and configuring new storage.

4.2.1 Understanding protected storage

Regardless of which storage technology you are using, it must be protected. At a minimum, this means:

- The storage must be available from at least two nodes so that the loss of a node does not cause a loss of the storage
- The storage must be mirrored so that the loss of a single disk does not cause a loss of the storage

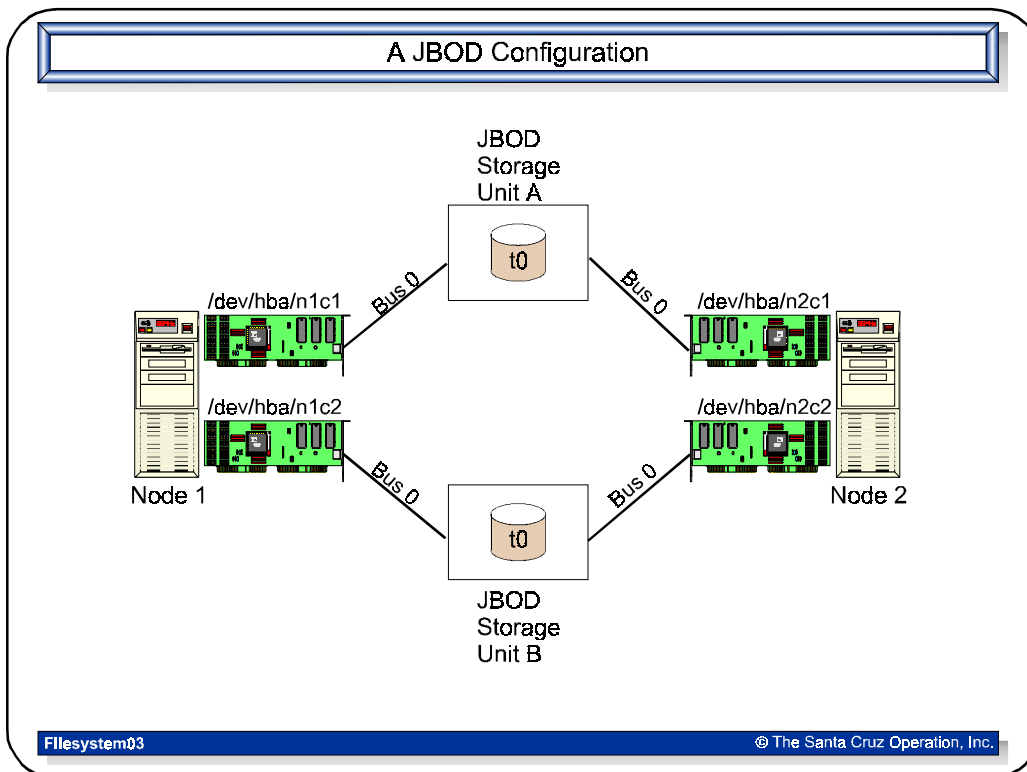
Protected storage implementations differ depending on the type of storage technology used:

Different storage technologies implement protected storage in different ways:

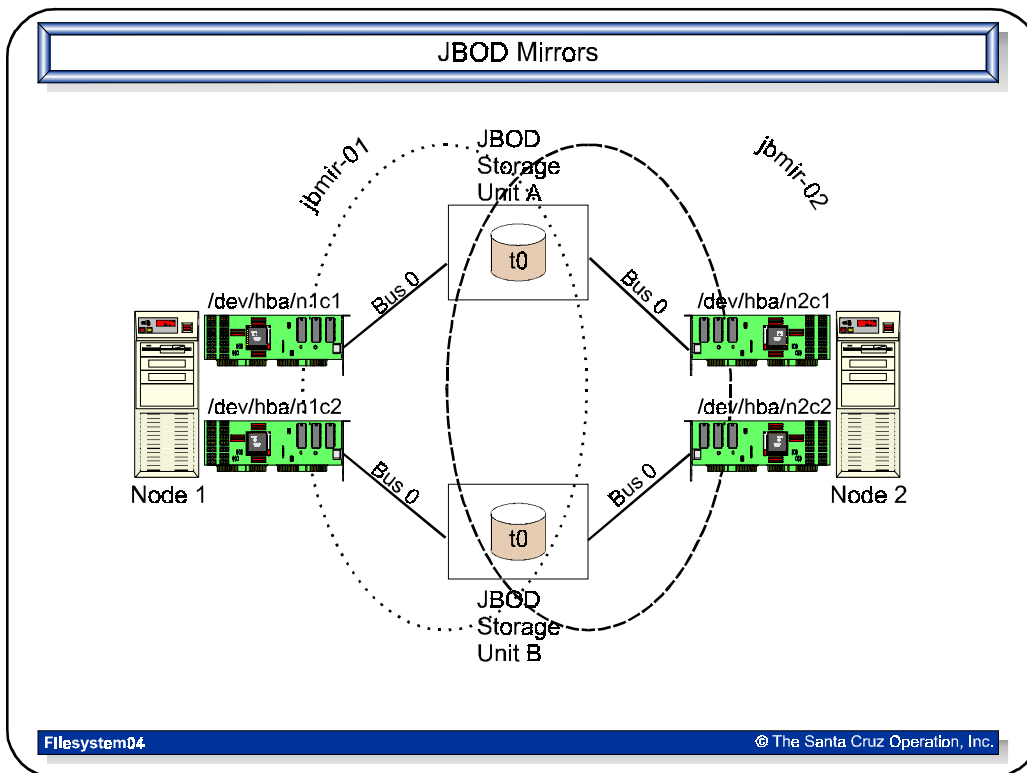
- On clusters using CNM, each node has its own disk and it is mirrored across nodes. At any given time, one node is actively serving I/O requests, and the other node is only mirroring writes.

If the node servicing the requests is lost, the other node continues processing. When the failed node returns, its mirror is updated or completely recovered by the Veritas software.

AV 4-3



AV 4-4



- JBOD (also known as dual-SCSI) is the most complicated. The hardware is shown in AV 4-3 on page 4-12. Each node has two SCSI controllers. Each SCSI controller is connected to a JBOD storage unit. That JBOD storage unit is also connected to a SCSI controller on the other node. In this example, if the LUN value for each disk is 0, the device names, relative to `/dev/dsk` or `/dev/rdisk` are:

- `n1c1b0t0d0`
- `n1c2b0t0d0`
- `n2c1b0t0d0`
- `n2c2b0t0d0`

As shown in AV 4-4 on page 4-12, a mirror is created between the two disks on each node. Thus, the two conditions are met because the disks are mirrored and available on each node. In the slide, the mirrors are called:

- `jbmir-01`, consisting of `/dev/dsk/n1c1b0t0d0` and `/dev/dsk/n1c2b0t0d0`
- `jbmir-02`, consisting of `/dev/dsk/n2c1b0t0d0` and `/dev/dsk/n2c2b0t0d0`
- For Fibre channel solutions are similar to, but less complicated than, JBOD. With this solution, each node has one fibre controller that is attached to the fibre storage box(es), possibly by way of fibre hubs or switches. Within the fibre storage box(es), hardware RAID is used to create a mirror of the disks. If the storage box only has one SCSI bus or power supply, then you must use two storage boxes.

However, if you are not using CNM to protect your storage, there is a NSC-specific issue you must address. The UNIX **mount(1M)** facility can only mount one device/filesystem to a mount point at a time. And, as discussed in the *UnixWare® 7 NonStop Clusters® Concepts* course, only one node controls the storage at a time. To handle the loss of the node that is currently controlling storage, you must configure the NSC-specific `/etc/cluster/fdevtab` file.

Note You are only required to configure additional non-*root* filesystems for failover. The *root* filesystem is automatically configured for failover during installation.

NSC uses the *fdevtab* and **cmount**(1M) facility to handle failover of non-root filesystems:

- The *fdevtab* file lists all of the names of the device nodes that access the same device/filesystem.
- The *vfstab*(4) file remains unchanged. It still lists the mount point, filesystem type, and mount options. However, the device node that is listed should be the first entry placed in *fdevtab* for this filesystem.
- In the event that the current path to the filesystem is lost, NSC looks in the *fdevtab* file for the next device-node to try, which is typically on a different node,
- Once a working path is found, the cluster node containing the device-node gains control of the hardware and I/O processing continues.

If NSC does not find a working path, the filesystem becomes unavailable. Note this will only happen if there are multiple simultaneous failures.

Referring again to AV 4-4 on page 4-12, an example *fdevtab* entry in the */etc/cluster/fdevtab* file for this system is:

```
fs 1:/dev/vx/dsk/jbmir-01 2:/dev/vx/dsk/jbmir-02
```

If the filesystem is being mounted on the */home2* directory, the corresponding *vfstab* entry in the */etc/vfstab* file is:

```
/dev/vx/dsk/jbmir-01 /dev/vx/rdisk/jbmir-01 /home2 vxfs 1 yes options
```

where *options* are discussed in Section 4.2.2, “Mount options in NSC,” on page 4-15.

Note If the example used Fibre instead of JBOD, the device names would not be Veritas mirror names. They would be directly under <i>/dev/dsk</i> and <i>/dev/rdsk</i> and conform to the convention described in the disk(7) manual page.
--

4.2.2 Mount options in NSC

The other issue related to mounting protected storage is the various mount options. These options are set in addition to any non-NSC *vxfs* options. They include:

csoft The *csoft* mount option is referred to as *data integrity level zero*. This level is analogous to NFS soft mounts. If the controlling node is lost, applications on other nodes in the cluster which are accessing that filesystem receive I/O errors. After the file system is failed over and remounted, applications on these other nodes continue processing normally.

chard The *chard* mount option is referred to as *data integrity level one*. This option causes the data to be write-cached on multiple nodes. This ensures that data for a completed write operation is not lost if the controlling node fails. In this case, data transfer and subsequent operations wait for the filesystem to fail over, after which the cached write operations are completed.

However, if the node that is hosting the writing process fails while data is in cache, the data may be lost.

The *chard* option does not guarantee that a write system call is *atomic* (performed completely or not at all). A node failure that occurs while a **write(2)** system call is in progress may result in a partial write to the file. Using *chard* only protects completed writes that are waiting in cache.

In some situations, modified data is cached on more than one node. Such an action is called backup caching. The performance of filesystems mounted with the *chard* option may be negatively impacted when a backup cache is not available. When a backup cache is not available, writes to a *chard*-mounted filesystem may need to be performed synchronously to disk to guarantee the data integrity associated with the *chard* mount option.

charddup The *charddup* mount option is referred to as *data integrity level two*. If the root node serving the filesystem goes down, any operations in progress are held and retried when the filesystem fails over. Data written to a file will survive any single node failure, including a failure of the node hosting the process that wrote the data and the root node,

Like the *chard* option, the *charddup* option only protects against the loss of data written by successfully completed **write()** calls. A node failure that occurs while a write system call is in progress may result in a partial write to the file.

The *charddup* option also supports backup caching, like *chard*.

4.2.3 Mounting and unmounting filesystems

The **cmount**(1M) command performs mount and failover remount operations according to cluster filesystem mount states. It executes automatically during node-up or node-down processing. The **cmount** command can list the mount state of all file systems, ensure all available file systems are mounted, or mount a specific file system.

The **cmount**(1M) command works similarly to the **mount**(1M) command. See the **cmount**(1M) manual page for more information.

Exercise: Displaying cluster filesystems

Run the **cmount** command without arguments to display information about the filesystems that are currently mounted within your classroom cluster.

4.3 Configuring Filesystems for Failover

This section describes the general procedure and considerations for adding filesystems in a cluster environment where the filesystem must be protected; that is, available in the event of a node failure. The detailed steps for configuring non-root filesystems is provided in the lab.

The two most common configurations are described:

CNM	Filesystems on internal disks in each node are mirrored in software under control of the Veritas Volume Manager (VxVM)
FibreChannel	Filesystems on disks in a Compaq FibreChannel storage unit are mirrored using RAID 1 hardware mirroring

This section describes how to add a protected filesystem in these two-node cluster configurations:

- A pair of disks is added to the cluster
 - For CNM, one disk is added to each of the potential root nodes
 - For RAID arrays, a pair of disks is added, one on each bus
- A single user filesystem is created on the new disk pair
- The *root* filesystem is on a separate disk

The remainder of the lesson describes how to add protected filesystems for CNM and FibreChannel clusters.

4.4 Configuring protected filesystems on hardware RAID arrays

For clusters using Compaq FibreChannel RAID arrays (RA4x00s), you must perform a series of array and logical disk operations to add drives and filesystems:

- Physically install the drives in the array
- Create logical volumes with the disks with the **nsraid1cfg**(1M) command
- Stamp additional volumes with a SCSI nexus to uniquely identify them, using the **casacfg**(1M) command
- Create device files for the disks using **sdmkdev**(1M)
- Bring the disks online (initialize **fdisk** partition, volume table of contents, and file systems) with the **diskadd**(1M) command
- Manually edit the failover device table in */etc/cluster/fdevtab* to assign device aliases for failover

The next sections provide more information about these tasks. A detailed procedure is provided in Section 4.8.1, “Procedure for FibreChannel clusters,” on page 4-43.

4.4.1 Physically installing the drives

Add a pair of drives RAID 1 mirroring:

- The disks must be in the same slot numbers in the same FC array chassis (for example, slot 1 on bus 0 and slot 1 on bus 1)
- The disks must be the same size

Refer to Installing the Hot-Pluggable Hard Drives in Chapter 2 of *Compaq FibreChannel Storage System User Guide* for physical details of module orientation and drive latch operation.

4.4.2 Creating logical volumes

The **nscraid1cfg** command allows you to initialize disk devices for use with NSC. **nscraid1cfg** provides a simple default RAID 1 mirror configuration, where pairs of physical disks are configured as a single logical disk.

Each logical disk is configured as a RAID 1 mirrored logical disk.

You must have **root** privileges to run the command, and it must be run from the node connected to the active FC array controller (not the standby array controller). Use the **-T** option to direct **nscraid1cfg** to the desired node.

4.4.3 Stamping volumes with a SCSI nexus

Certain storage system require that you verify the array's hardware connections and stamp its logical volumes with a SCSI nexus.

A nexus is a unique identifier for each disk. It is the name of the device in the `/dev/dsk` and `/dev/rdisk` directories and has the form `c#b#t#d#` where:

- The # after the 'c' is a controller number
- The # after the 'b' is a bus number
- The # after the 't' is a target-id (sometimes called TID) number
- The # after the 'd' is a logical unit (sometimes called LUN) number

Logical volumes in the Compaq RAID array are stamped using the **casacfg**.utility, which is installed as part of the Compaq EFS :

4.4.3.1 Bringing the disks online

Use the **diskadd(1M)** command to create a UNIX partition on the disk and add the filesystem slice. The device name argument is of the form *c#b#t#d#*. For example, on a node with:

- An SCSI controller (with the initial internal root disk) as controller 0
- An IDE controller with a CD-ROM as controller 1
- A FibreChannel HBA connecting to the shared storage unit

This is an example command line for creating a UNIX partition on the second disk on the third controller—that is, the disk added to the external storage unit:

```
# diskadd c2b0t1d0
```

Use the entire disk for UNIX.

4.4.3.2 Configuring the failover device table

The entries in the */etc/cluster/fdevtab* file specify aliases for block- special device names for mounting and failover remounting of file systems. These entries are processed automatically when a node joins, rejoins, or leaves a cluster.

Whenever one of the block-special device names in the *fdevtab* file corresponds to the special name for a manually mounted filesystem in the */etc/vfstab* or */etc/mnttab* files, the aliases listed in the *fdevtab* file control both mounting and failover remounting. Specifically, mounts and remounts for a given mount point are always attempted in the order specified in the *fdevtab* file.

Entries in the *fdevtab* file must comply with the following format rules:

- Lines beginning with the pound character (**#**) are treated as comments, and are ignored.
- Each line must begin with the specification of a filesystem type, followed by one or more space or tab characters. Currently, the only supported file system type is **fs**.
- Each line must contain two or more block-special device entries that begin with */dev/dsk/*.
- Entries are delimited with any number of semicolon (**;**), space, or tab characters, and by a line-terminating newline character.

- An entry can specify either a full block-special device path or can include a wild-card character (*) in place of any two-character component of the special name following the controller and bus. For example, `/dev/dsk/n1c0b0t0d0s2` identifies a partition on slice (partition) 2 of the first drive attached to node 1, while `/dev/dsk/n1c0b0t0d0*` specifies all slices on this device.
- Use a backslash character (\) to continue an entry on more than one line.

This is a sample *fdevtab* file for a filesystem on the second disk of an external storage unit attached to the third controller in each node:

```
#
#This file, /etc/cluster/fdevtab, specifies block-special device aliases
#for filesystems on multi-ported disk controllers.

#
fs      1:/dev/dsk/n1c3b0t0d1s0;          2:/dev/dsk/n2c3b0t0d1s0
fs      1:/dev/dsk/n1c3b0t0d1sa;          2:/dev/dsk/n2c3b0t0d1sa
fs      1:/dev/dsk/n1c3b0t0d1sb;          2:/dev/dsk/n2c3b0t0d1sb

fs      1:/dev/dsk/n1c3b0t0d1s*;          2:/dev/dsk/n2c3b0t0d1s*
fs      1:/dev/dsk/n1c3b0t0d1p*;          2:/dev/dsk/n2c3b0t0d1p*
fs      1:/dev/dsk/n1c3b0t0d1*;          2:/dev/dsk/n2c3b0t0d1*
fs      1:/dev/dsk/n1c3b0t0*;            2:/dev/dsk/n2c3b0t0*

#fs      1:/dev/dsk/n1c3b0*; 2:/dev/dsk/n2c0b0*
```

Refer to the *fdev(4)* reference manual page for details.

4.5 Configuring protected filesystems on CNM clusters

For clusters using cross-node mirroring, you must perform a series of array and logical disk operations to add drives and filesystems:

- Physically install the drives in the root node and failover root node.
- Put the disk under **Veritas Volume Manager** control with the **vxdiskadd(1M)** command
- Configure Veritas mirroring between the drives and add a Veritas filesystem
- Manually edit the failover device table in */etc/cluster/fdevtab* to assign device aliases for failover

The next sections provide more information about these tasks. A detailed procedure is provided in Section 4.8.2, “Procedure for CNM clusters,” on page 4-47.

Note Before adding and testing mirrored filesystems on a CNM cluster, remove and reinstall the Veritas Mirroring or ODM license using the SCOWadmin **License Manager**. This ensures that the license is properly applied cluster-wide. In some cases, mirroring does not function properly unless the licenses is removed and reinstalled.

4.5.1 Physically installing the drives

Add a drive into the root node and the failover root node:

- The disks should be configured similarly; same SCSI ID, same bus, etc.
- The disks should be the same size

Refer to your hardware vendor's manual for physical details of installing disks.

4.5.2 Understanding Veritas Volume Management

To protect filesystems in a CNM cluster, you must bring all physical disks under control of the Veritas **Volume Manager**. Protected storage is managed by a single instance of Veritas **Volume Manager**, known as the **VxVM**, running on the root node.

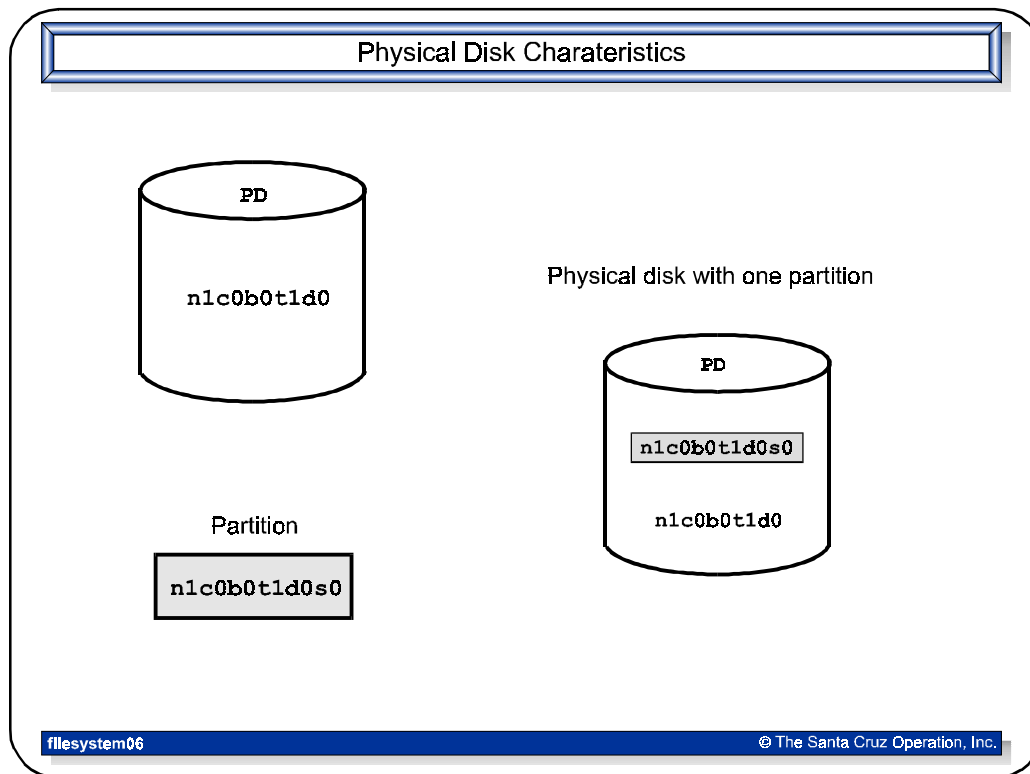
If the root node fails, **VxVM** is automatically started on the backup node and volumes are managed from the backup node now serving as the current root node. When the original root node returns to the cluster, **VxVM** re-establishes the mirror by copying current volume data from the backup root node to the original root node.

The **VxVM** builds virtual devices, called *volumes* on top of physical disks. Volumes are accessed by a UNIX filesystem, a database, or other applications in the same way physical disk partitions would be accessed.

Volumes are composed of other virtual objects that can be manipulated to change the volume's configuration. Volumes and their virtual components are referred to as **VxVM objects**.

To add filesystems in CNM clusters that are protected in the event of node failure, you must understand how the Veritas Volume Manager (**VxVM**) manages objects and the relationships between physical objects and Veritas objects.

AV 4-5



4.5.2.1 Physical Objects

To perform disk management tasks using the VxVM, you must understand two physical objects:

- Physical disks
- Partitions

4.5.2.1.1 Physical Disks

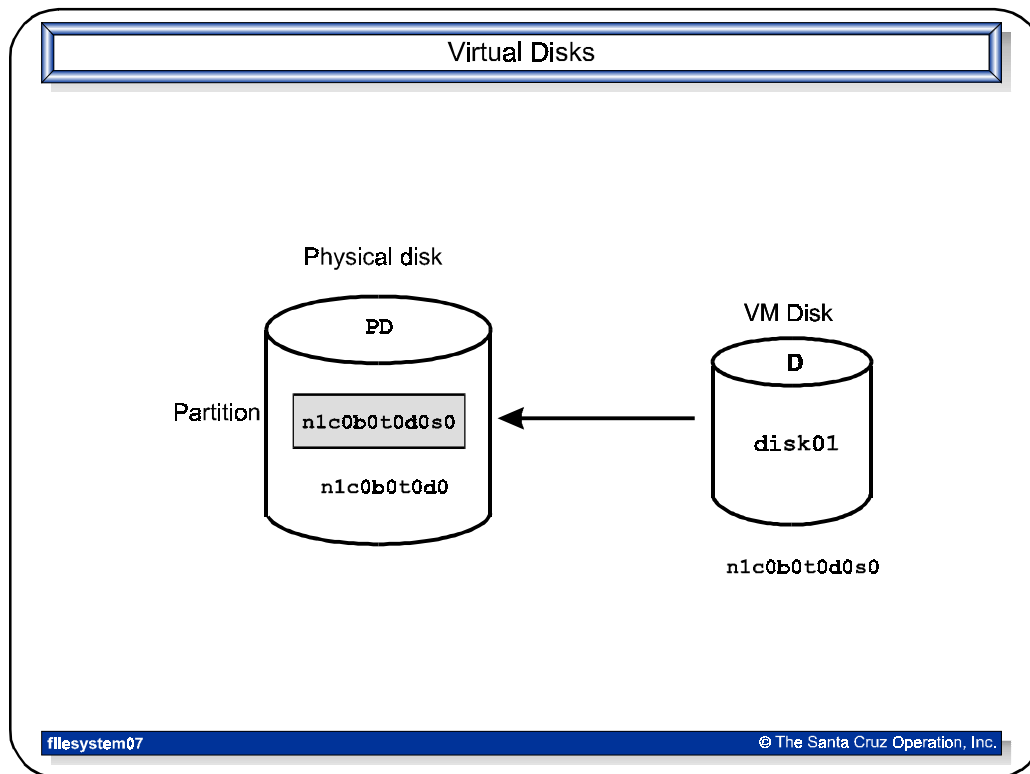
A physical disk is the underlying storage device (media). A physical disk can be accessed using a device name such as *c#b#t#d#*, where *c#* is the controller, *b#* is the bus, *t#* is the target ID, and *d#* is the disk number. The disk in AV 7-2 is disk number 0 with a target ID of 0, and it is connected to controller number 0 in the system.

4.5.2.1.2 Partitions

A physical disk can be divided into one or more partitions. The partition number, or *s#*, is given at the end of the device name. Note that a partition can take up an entire physical disk, such as the partition shown.

The relationship between physical objects and **VxVM** objects is established when you place a partition from a physical disk under **VxVM** control.

AV 4-6.



4.5.2.2 VxVM Objects

There are several **VxVM** objects that allow you to perform disk management tasks:

- VM disks
- Disk groups
- Subdisks
- Plexes
- Volumes

The **VxVM** objects are described in the sections that follow. Before you can use the **VxVM** to perform disk management tasks, you must understand the definitions of these objects and their relationships to each other and physical disk devices.

4.5.2.2.1 VM Disks

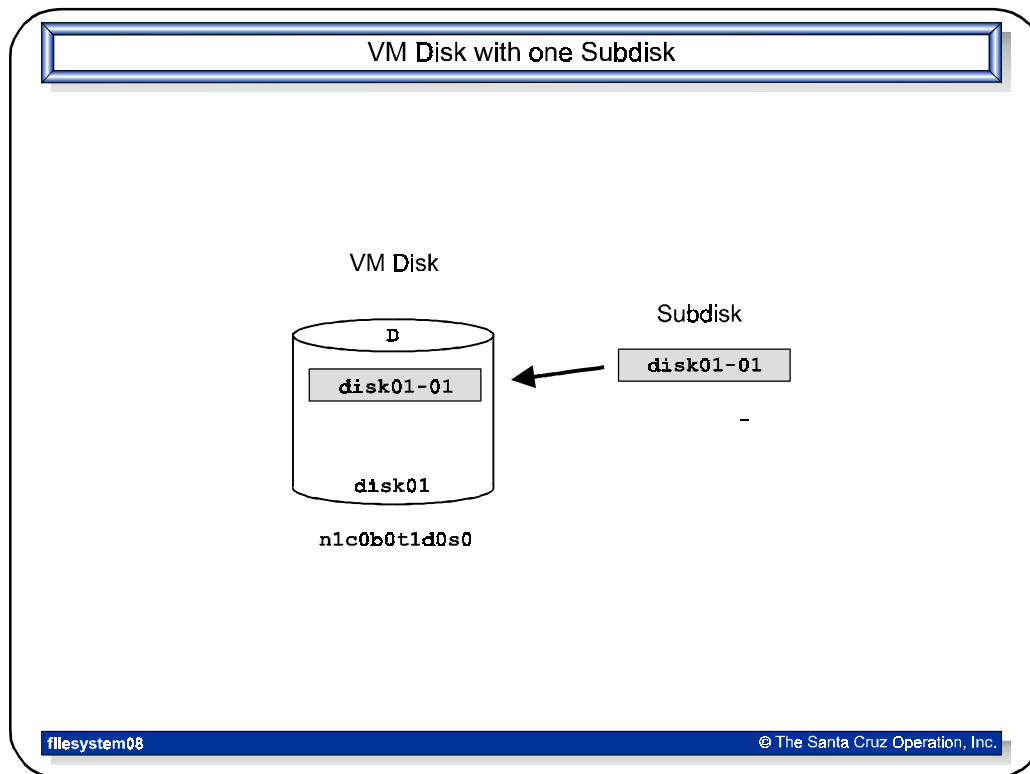
A VM disk is a contiguous area of disk space from which the **VxVM** allocates storage. When you place a physical disk under **VxVM** control, a VM disk is assigned to the disk.

VM Disk properties include:

- Each VM disk corresponds to at least one partition.
- A VM disk is composed of a public region (from which storage is allocated) and a private region (where configuration information is stored).
- A VM disk is accessed using a unique disk media name, which you can supply (or else the **VxVM** assigns one that typically takes the form *disk##*). AV 4-6 on page 4-28 shows a VM disk with a disk media name of *disk01* that is assigned to the partition *n1c0b0t0d0s0*.

With the **VxVM**, applications access volumes (created on VM disks) rather than partitions.

AV 4-7.



4.5.2.2.2 Disk Groups

A disk group is a collection of VM disks that share a common configuration.

A configuration consists of a set of records containing detailed information about existing **VxVM** objects, their attributes, and their relationships.

The default disk group is *rootdg* (the root disk group). Additional disk groups can be created, as necessary. Volumes are created within a disk group; a given volume must be configured from disks belonging to the same disk group.

Disk groups allow the administrator to group disks into logical collections for administrative convenience. A disk group and its components can be moved as a unit from one host machine to another.

4.5.2.2.3 Subdisks

A subdisk is a set of contiguous disk blocks. Subdisks are the basic units in which the **VxVM** allocates disk space. A VM disk can be divided into one or more subdisks. AV 4-7 on page 4-30 shows an example of a subdisk

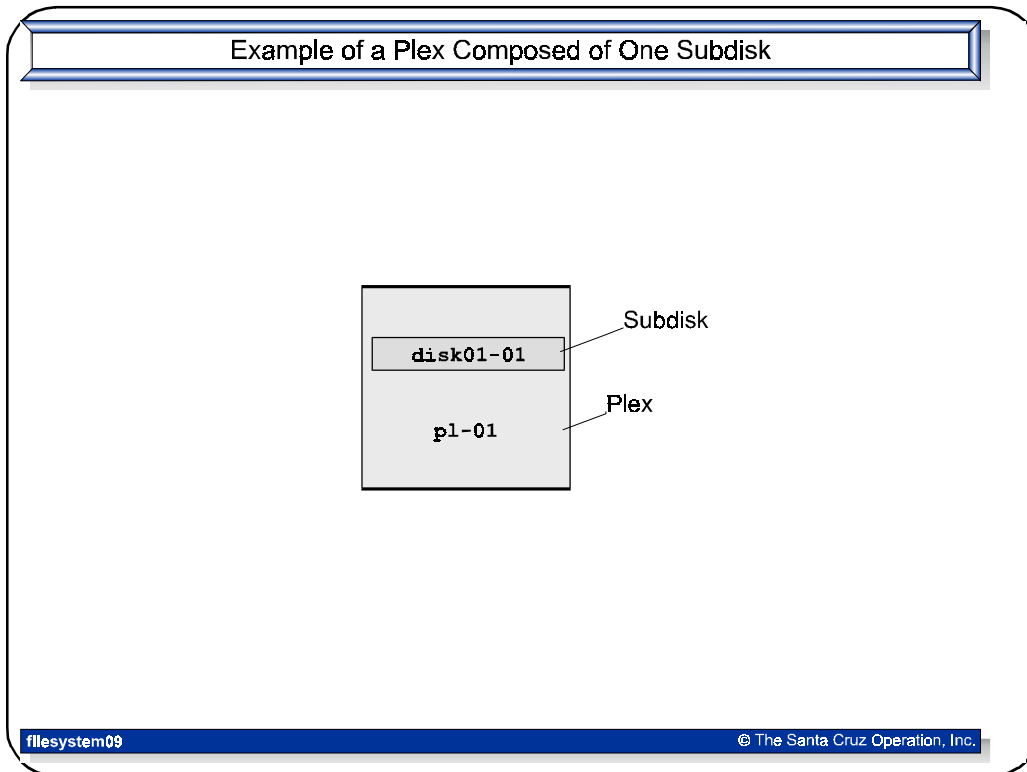
Any VM disk space that is not part of a subdisk is considered to be free space, which can be used to create new subdisks.

Each subdisk represents a specific portion of a VM disk, which is mapped to a specific region of a physical disk.

Since the default name for a VM disk is *disk##* (such as *disk01*), the default name for a subdisk is *disk##-##*. So, for example, *disk01-01* would be the name of the first subdisk on the VM disk named *disk01*.

A VM disk may contain multiple subdisks, but subdisks cannot overlap or share the same portions of a VM disk. AV 4-7 on page 4-30 shows a VM disk with one subdisk that is assigned to one partition.

AV 4-8

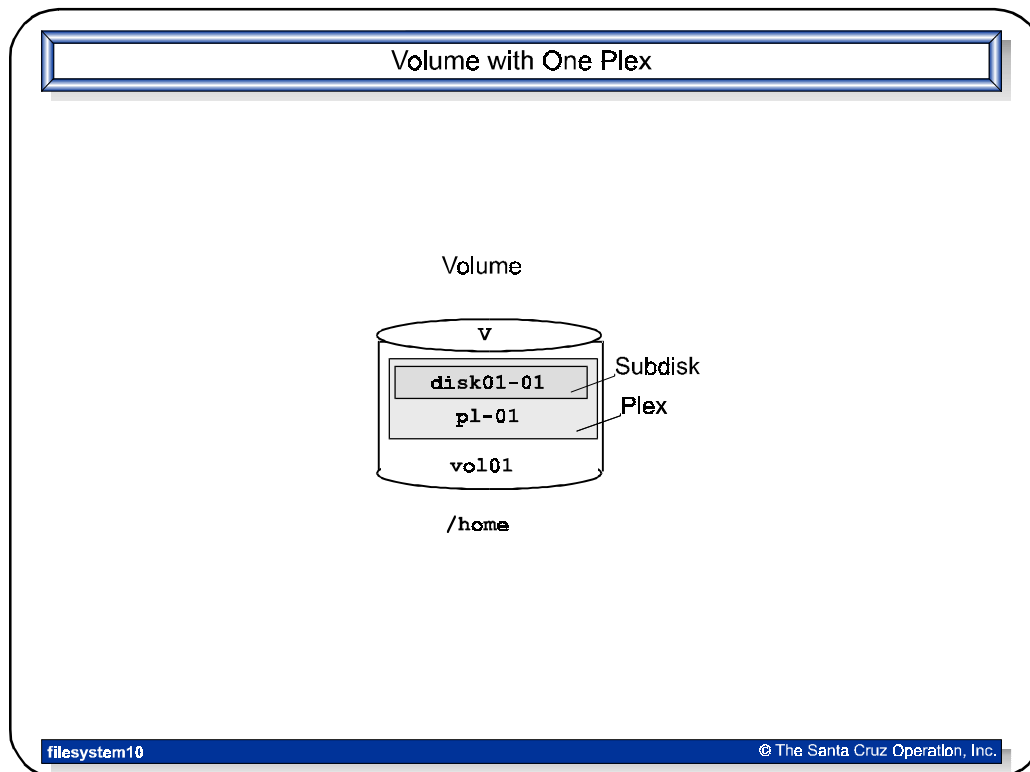


4.5.2.2.4 Plexes

The VxVM uses subdisks to build virtual entities called *plexes*, also referred to as mirrors. A plex consists of one or more subdisks located on one or more disks. In the NSC environment, data is organized as a RAID-1 mirror on the subdisks that constitute a plex.

The VxVM also supports a number of other ways to organize data, such as RAID-5 and RAID-0, but these are not applicable to the subject of adding protected (mirrored) filesystems.

AV 4-9



4.5.2.2.5 Volumes

A volume is a virtual disk device that appears to be a physical disk partition, but does not have the physical limitations of a physical disk partition.

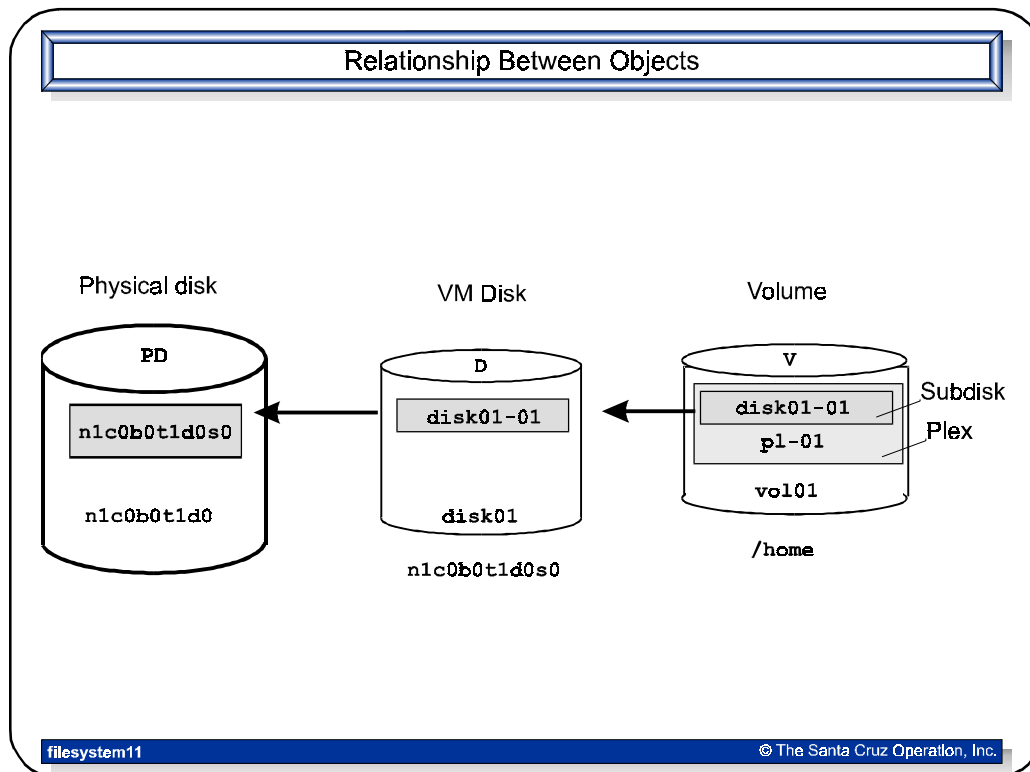
A volume consists of one or more plexes, each holding a copy of the data in the volume. Due to its virtual nature, a volume is not restricted to a particular disk or a specific area on a disk. The configuration of a volume can be changed, using the **VxVM** interfaces, without causing disruption to applications or filesystems that are using the volume.

For example, a volume can be mirrored on separate disks or moved to use different disk storage. A volume can consist of up to 32 plexes, each of which contains one or more subdisks. In order for a volume to be usable, it must have at least one associated plex with at least one associated subdisk. Note that all subdisks within a volume must belong to the same disk group.

The **VxVM** uses the default naming conventions of *vol##* for volumes and *vol##-##* for plexes in a volume. The example in AV 4-9 on page 4-34 uses the name *pl-01* to help distinguish the plex from the volume.

In a cluster environment, it is a good idea to use node designations in object names. For example, instead of using the **Volume Manager** default name of *disk01* or *disk02*, you can specify *n1disk01* or *n2disk02*, depending on which node the disk is installed.

AV 4-10



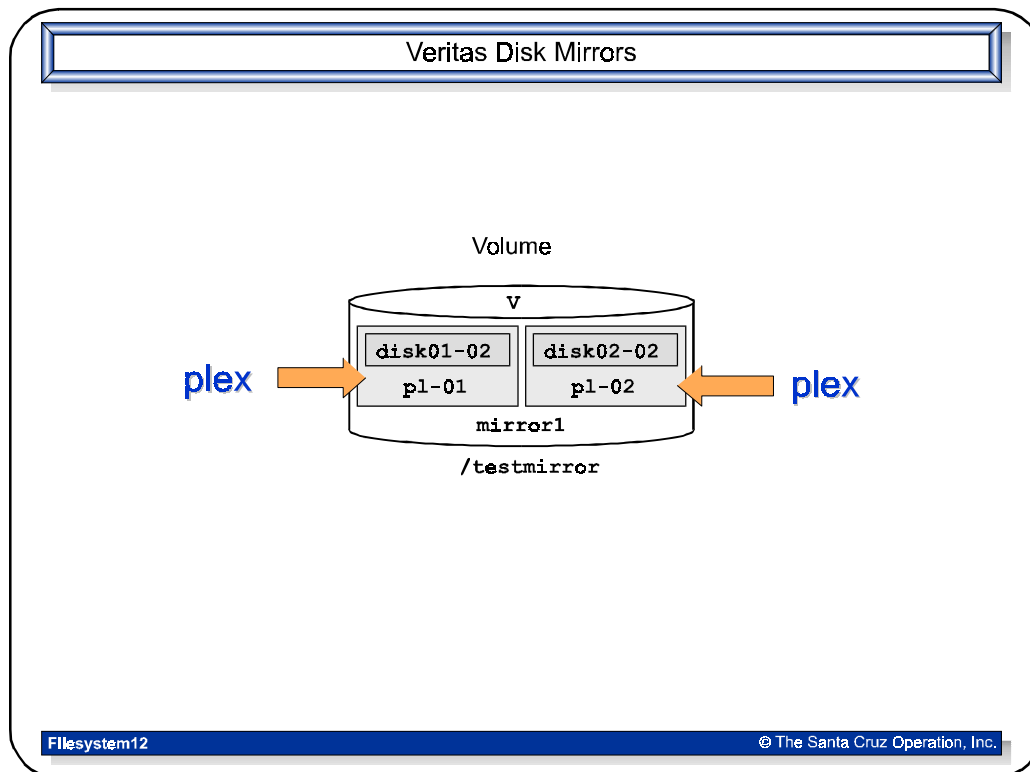
4.5.2.3 Relationships between VxVM objects

VxVM objects are of little use until they are combined to build volumes. VxVM objects generally have the following relationship:

- VM disks are placed under **VxVM** control and grouped into disk groups. Subdisks (each representing a specific portion of a disk) are combined to form plexes.
- A volume is composed of one or more plexes.

AV 4-10 on page 4-36 illustrates the relationships between (virtual) **VxVM** objects, as well as how they relate to physical disks. This illustration shows a VM disk, *disk01*, with volume with one plex, *pl-01*, comprised of one subdisk, *disk01-01*.

AV 4-11



4.5.3 Using the VERITAS Volume Manager

To set up cross-node mirroring for all filesystems, you must create the appropriate volumes, plexes and subdisks using the available space on the disk, and create user file systems that will be mirrored across both nodes.

AV 4-11 on page 4-38 is a visual representation of a CNM volume, *mirror1*. It includes two plexes, *pl-01* and *pl-02*, which are identical mirrors. In this example, each plex consists of one **Volume Manager** subdisk, each of which is mapped back to one physical disk attached to each node. For example, *pl-01* maps to physical disk *n1c0b0t1d0* on node 1 and *pl-02* maps to disk *n2c0b0t1d0* on node 2.

4.5.3.1 Bringing Physical Disks under Volume Manager Control

Adding physical disks to **Volume Manager** is the first step in creating mirrored volumes for user data. Bring the disk under **VxVM** management using the **vxdiskadd(1M)** command with the device name for the physical disk. For example, this command brings the second disk on the first controller on node 1 under **VxVM** control:

```
vxdiskadd n1c0b0t1d0
```

For simplicity, use the default values for the disk name and disk group name (*rootdg*).

4.5.3.2 Creating Veritas volumes and filesystems

Once Volume Manager disks are established, use the **VERITAS Volume Manager** to:

- Add a new disk
- Create a subdisk
- Create a plex
- Create a volume
- Create a mirrored filesystem within a volume
- View disk storage entity parameters

To run the **VxVA**, start SCAdmin and select **VERITAS Volume Manager**.

A detailed procedure for using the **VERITAS Volume Manager** is provided in Section 4.8.2, “Procedure for CNM clusters,” on page 4-47.

4.5.3.3 Configuring the Failover Device Table

Edit the `/etc/cluster/fdevtab` file to assign device aliases for file system failover. This file contains templates that are commented out, and may be used as samples for setting up the entries. Refer to the `fdevtab(4)` reference manual page for details.

This is a sample `fdevtab` file:

```
#
#This file, /etc/cluster/fdevtab, specifies block-special device aliases
#for filesystems on multi-ported disk controllers.

#
#fs      1:/dev/dsk/n1c0b0t1d0s0;          2:/dev/dsk/n2c0b0t1d0s0
#fs      1:/dev/dsk/n1c0b0t1d0sa;          2:/dev/dsk/n2c0b0t1d0sa
#fs      1:/dev/dsk/n1c0b0t1d0sb;          2:/dev/dsk/n2c0b0t1d0sb

#fs      1:/dev/dsk/n1c0b0t1d0s*;          2:/dev/dsk/n2c0b0t1d0s*
#fs      1:/dev/dsk/n1c0b0t1d0p*;          2:/dev/dsk/n2c0b0t1d0p*
#fs      1:/dev/dsk/n1c0b0t1d0*;          2:/dev/dsk/n2c0b0t1d0*
#fs      1:/dev/dsk/n1c0b0t1*;            2:/dev/dsk/n2c0b0t1*

#fs      1:/dev/dsk/n1c0b0*; 2:/dev/dsk/n2c0b0*

#fs      1:/dev/dsk/n1c0b0*; 2:/dev/dsk/n2c0b0*; 3:/dev/dsk/n3c0b0*; \
4:/dev/dsk/n4c0b0*; 5:/dev/dsk/n5c0b0*; 6:/dev/dsk/n6c0b0*
```

4.6 Lesson Summary

This lesson discussed the NSC-specific filesystem administration issues. Specific topics discussed include:

- Overview of files, filesystems, and device nodes in NSC
- Filesystems administration in NSC
- Adding protected filesystems.

Review questions

1. List and describe the different integrity levels added to the **mount** command for NSC.
2. What is the purpose of the *fdevtab* file?
3. What is the purpose of the OSU slice?

4.7 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

cmount (1M)	Mounts cluster filesystems
mknod (1M)	Makes a special file
mount (1M)	Mounts or unmounts filesystems
switchroot (1M)	Switches between available OSU root slices
write (2)	Performs a write to a file

Relevant files

<i>/dev/dsk</i>	Is the directory containing block-special disk device files. See <i>disk</i> (7).
<i>/dev/rdisk</i>	Is the directory containing character-special disk device files. See <i>disk</i> (7).
<i>/etc/cluster/fdevtab</i>	Is the table of aliases for filesystem failover devices. See <i>fdevtab</i> (4).
<i>/etc/vfstab</i>	Is the table of filesystem defaults. See <i>vfstab</i> (4).
<i>/dev/xv</i>	Is the directory containing Veritas devices

4.8 Lab: Configuring a Non-Root Filesystem for Failover

The next two labs show how to add a protected filesystem to a cluster demonstrating the two most common methods in use today:

- CNM using VERITAS Volume Manager to configure software mirroring
- External storage units (Compaq FibreChannel RA4x00) using RAID 0 hardware mirroring

The user filesystem is called *mirror1* and is mounted on the */testmirror* directory.

4.8.1 Procedure for FibreChannel clusters

For clusters using Compaq FibreChannel RAID arrays (RA4x00s), complete this procedure to add drives and filesystems:

- Step 1. Physically install a pair of drives in the array. Install disks in the corresponding slot in each of the two SCSI buses. See your hardware vendor's manual for installation instructions.
- Step 2. Update the firmware on the disks, if necessary. See Section 1.9, "Initializing Shared Storage," on page 15 in the *NSC Curriculum Hardware Supplement*. You must have firmware version 2.40, or dated 12/99 or later, to insure Compaq nodes and storage units work with NSC version 7.1.1.
- Step 3. Create logical volumes with the disks with the **nscraid1cfg(1M)** command. For example, if node 1 is the active node controlling the storage unit, enter:
- ```
nscraid1cfg -T 1
```
- Step 4. Stamp logical drives with a SCSI nexus. See Section 1.9.3, "Stamping disks in a Compaq Fibre array," on page 65 in the *NSC Curriculum Hardware Supplement*.
- Step 5. Make sure that the logical volume for the *root* filesystem is online and available by checking the */dev/rdisk* directory for device entries that correspond to the SCSI nexus of the logical volume you just created. For example, the disk device associated with nexus 2:0:1:0 is *n1c2b0t1d0*.
- Step 6. Create the device files using **sdimgdev(1M)** with the **-f** option. For example:
- ```
sdimgdev -f
```

- Step 7. Bring the disks online. Run the **diskadd(1M)** command from the `/dev/rdisk` directory. For example:

```
cd /dev/rdisk  
diskadd n1c2b0t1d0
```

- a. Enter **y** to continue when warned about destroying existing data.
- b. Enter **y** to use the entire disk for a UNIX partition.
- c. Enter **n** to skip surface analysis.
- d. Enter **1** when prompted for the number of filesystems. In this lab, the entire disk will be used for the filesystem.
- e. Enter **/testmirror** as the absolute pathname for the filesystem.
- f. Enter **vxfs** as the filesystem type.
- g. Press  to select the default block size (1024k).
- h. Press  to enable automounting.
- i. Enter the number of all available cylinders for `/mirror1`.
- j. Enter **y** when prompted to accept the allocation.
- k. Accept the default allocation of inodes, when prompted.
- l. The **df(1)** command displays all filesystems. The display should include a line showing `/mirror1` similar to this:

```
/testmirror  (/dev/dsk/c2b0t1d0s1): 100000 blocks
```

Step 8. Configure the failover device table.

a. Edit */etc/cluster/fdevtab*.

b. Create the entries for your cluster devices. For example:

```
fs      1:/dev/dsk/n1c2b0t1d0s*;      2:/dev/dsk/n2c2b0t1d0s*
```

Step 9. Test the filesystem failover. Shut down node 1 with this command:

```
clusternode_shutdown 1 -i0 -g0 -y
```

Step 10. On node 2, after the *root* filesystem has failed over, attempt to copy files to and from the */testmirror* directory.



4.8.2 Procedure for CNM clusters

To add a protected user filesystem to a CNM cluster:

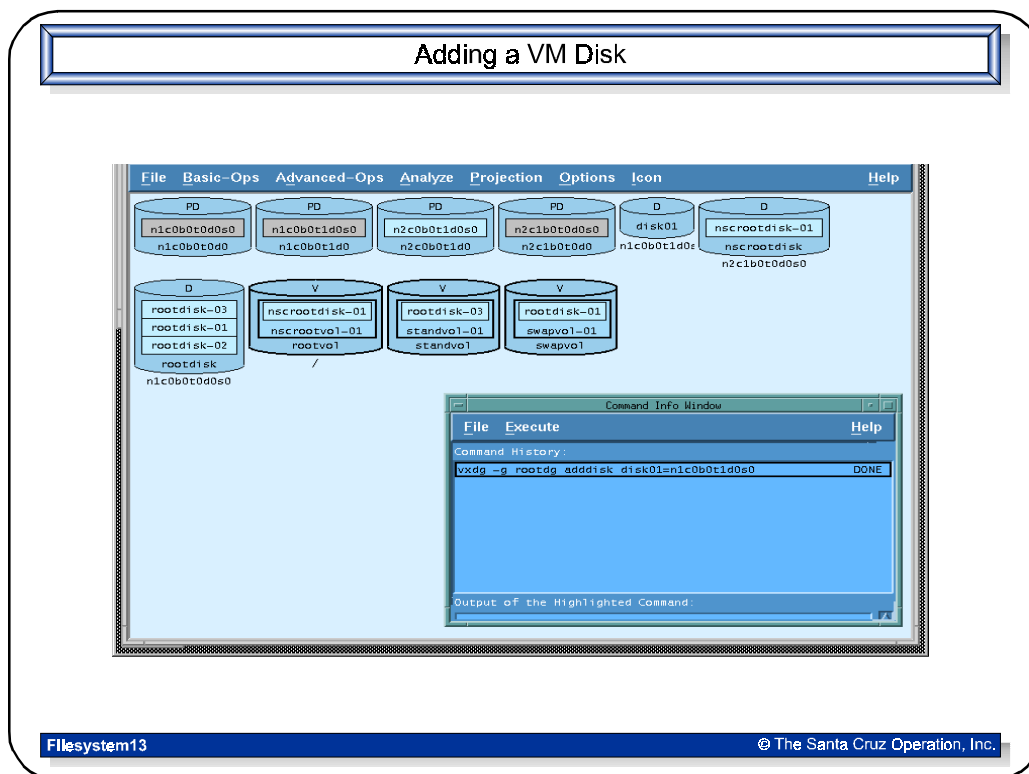
- Step 1. Install the physical disk for this filesystem, one in root node and one in the failover root node.
- Step 2. Log in as **root** on node 1. You must run **VxVM** in a graphical environment.
- Step 3. Reinstall the Veritas Mirroring or ODM license, if you have not done so after both cluster nodes were initially installed.
- Step 4. Bring the physical disk under **Volume Manager** control using **vxdiskadd**. This example assumes you have new disks that do not have an existing VTOC table. Otherwise you will be prompted to encapsulate the disk.

vxdiskadd n1c0b0t1d0

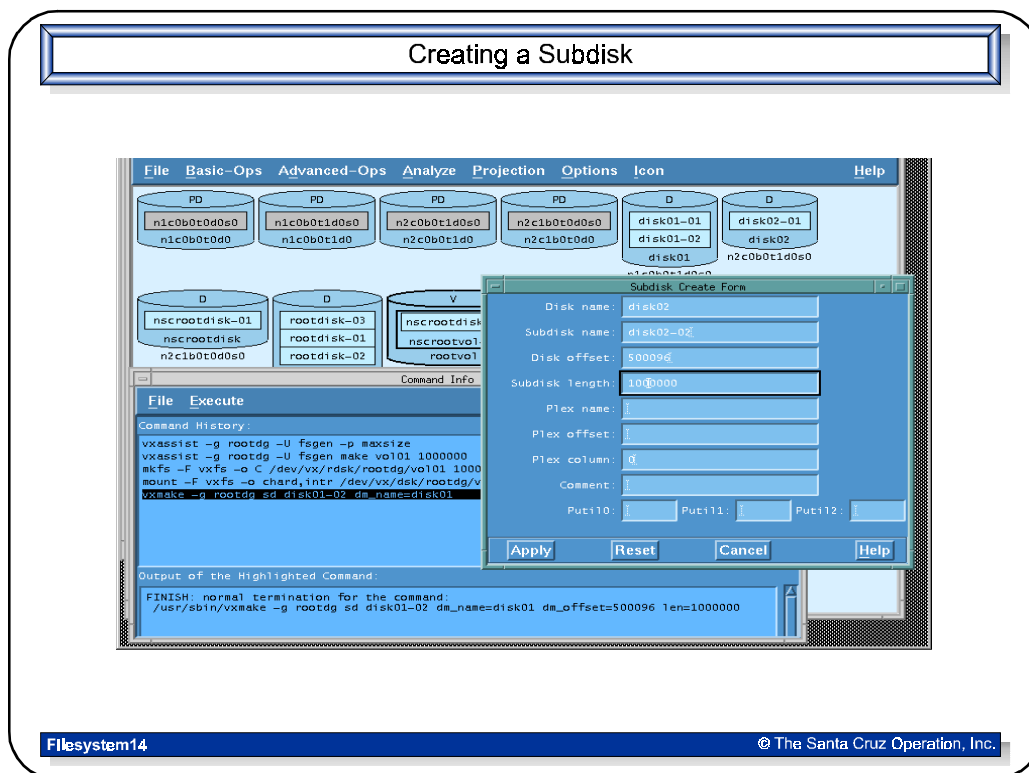
Respond the these prompts:

- a. Verify that you have entered the correct nexus.
- b. Press at the “Continue operation” prompt.
- c. Accept *rootdg* as the disk group name.
- d. Press to use the default name for the disk.
- e. Press to choose not to add this disk as a spare disk.
- f. Press at the “Continue operation” prompt.
- g. Enter the nexus number of the new disk.

AV 4-12



AV 4-13



Step 5. Add a VM disk.

- a. Start **VxVA**. Run **SCOadmin** and select **VERITAS Volume Manager**.
- b. Click the **World** button. The new logical disk, *disk01*, should be present in the **VxVM** World View if the **vxdiskadd** command was executed successfully in the previous step.
- c. Click on the **Basic-Ops** menu.
- d. Select **Disk Operations** ⇨ **Add Disks**.
- e. Enter the nexus number of the disk.
- f. Select **No** in the “Analyze disk” field.
- g. Enter **rootdg** in the “Disk group” field.
- h. Click on **Apply**.

The actual Veritas command executed is shown in the **Command Info** window if there is an error.

Step 6. Create two subdisks of the VM disk, *disk-01*:

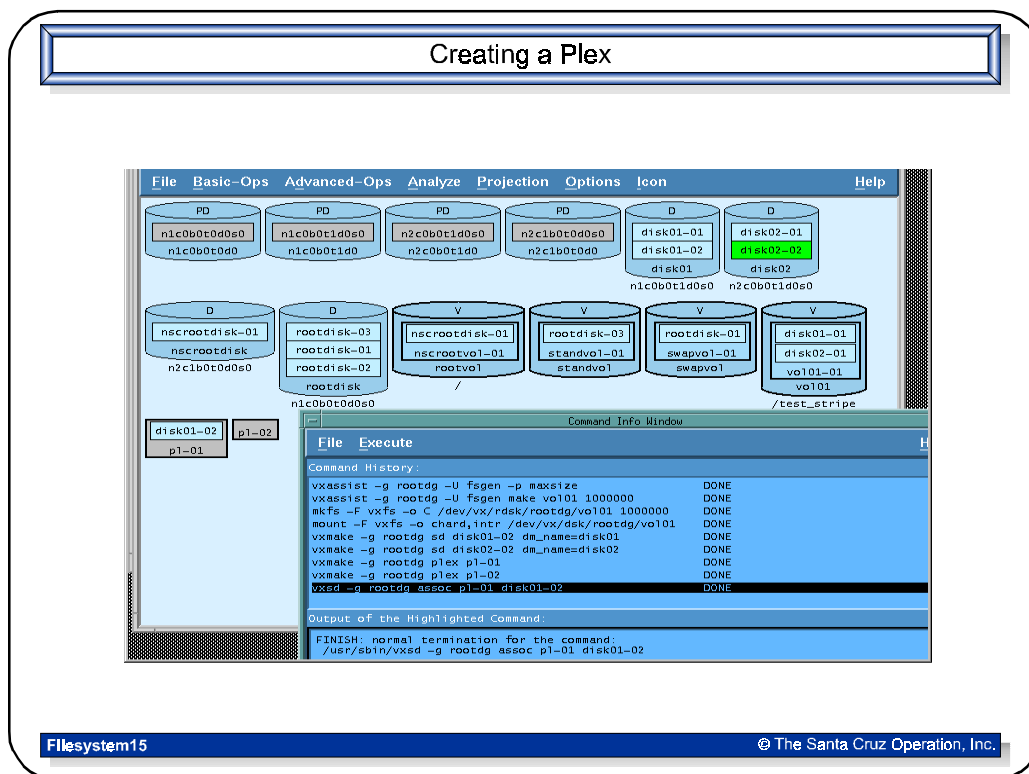
- a. In the **World View** display, click on disk *disk-01*.
- b. Select **Advanced-Ops** ⇨ **Subdisk** ⇨ **Create**.

The disk name will default to *disk01-01*. Do not change this.

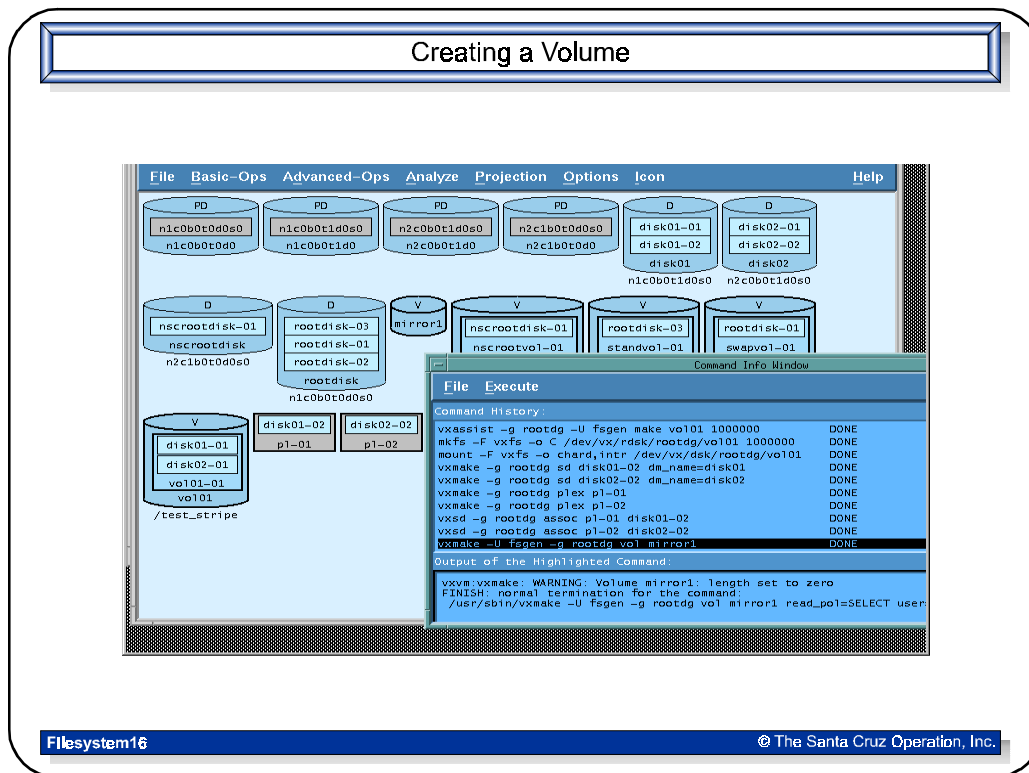
- c. Enter a subdisk length of 100000.
- d. Click the **Apply** button at the bottom of the window to create the subdisk.
- e. Repeat steps b - d to create *disk01-02*.

Note It is important to use the same subdisk length of 100000 for both disks, since you will be creating a mirrored set later in the lab.

AV 4-14



AV 4-15



- Step 7. Create two same-sized plexes.
- a. Select subdisk *disk01-01*. The subdisk will be highlighted in green.
 - b. Create a plex composed of the single subdisk, *disk01-01*, that was created in Step 5. Select **Advanced-Ops** ⇒ **Plex** ⇒ **Create**.

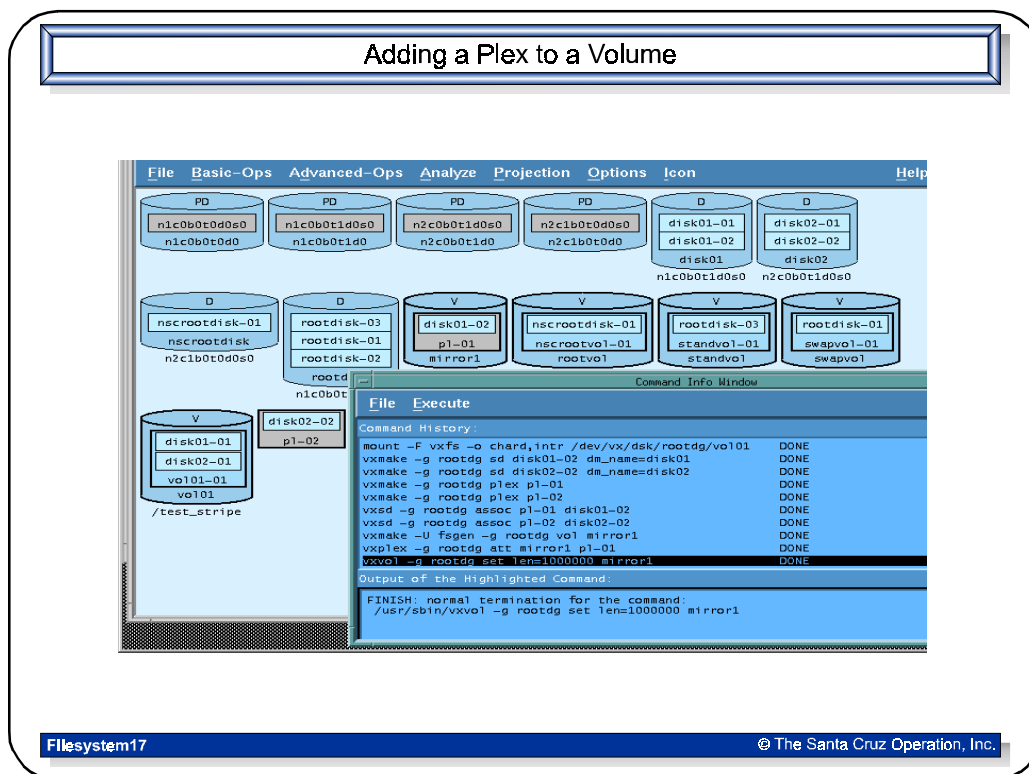
The **Plex Create Form** is displayed.
 - c. Enter **pl-01** as the plex name.
 - d. Click **Apply** to continue.

The **vxmake** command is invoked to create the plex. The **vxsd** command is then invoked to associate the subdisk with the plex.
 - e. The plex is created with name *pl-01*. It should now be visible in the **World View**.
 - f. Repeat step a - e to create another plex (*pl-02*) composed of one subdisk, *disk01-02*.

- Step 8. Create a volume.
- a. Select **Advanced-Ops** ⇒ **Volume** ⇒ **Create**.
 - b. Enter *mirror1* as the name for the new volume.
 - c. Select **Apply** to perform the volume creation.

The **vxmake** command is invoked to create the volume *mirror1*.

AV 4-16



- Step 9. Create a mirrored filesystem.
- Drag and drop plex *pl-01* into volume *mirror1*.
 - Drag and drop plex *pl-02* into volume *mirror1*.
 - Before a filesystem can be created on the volume, you must start the volume. Click on volume *vol1* to select it.
 - Start the volume. Select **Advanced-Ops** ⇒ **Volume** ⇒ **Start Volumes** ⇒ **Start**.
 - Enter *mirror1* as the name for the mirrored volume.
 - Select **Apply** to execute the volume creation.
 - Create a filesystem on the new volume. Click on the volume to select it.
 - Select **Basic-Ops** ⇒ **File System Operations** ⇒ **Make File System**.
 - Enter */testmirror* as the mount point.
 - Select **chard** as the mount option.
 - Accept all other defaults and click **Apply**.
 - Open a terminal window, and verify the correct operation of the new volume by attempting to copy some files to the */testmirror* directory.
- Step 10. Configure the failover device table.
- Edit */etc/cluster/fdevtab*.
 - Create the entry for your cluster device. For example:


```
fs      1:/dev/dsk/n1c0b0t0d0s0;      2:/dev/dsk/n2c0b0t0d0s0
```
- Step 11. Test the filesystem failover. Shut down node 1 with this command:
- clusternode_shutdown 1 -i0 -g0 -y**
- Step 12. On node 2, after the *root* filesystem has failed over, attempt to copy files to and from the */testmirror* directory.

4.9 Answers to Exercises, Review Questions, and Labs

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Review Question Answers

This section contains the answers to the review questions on page 4-41.

1. The options added to clusterize the **mount** command are:
 - csoft
 - chard
 - charddup
2. Entries in the *fdevtab* file controls failover for file systems other than the root file system.
3. The OSU slice is used to store an image of the *root* filesystem, so it can be upgraded while the cluster is running.

5

Network Administration



Lesson 5

Network Administration

Introduction

The ***UnixWare 7 Network Administration*** course presented information on how to administer IP-based networking facilities on a stand-alone host. Most of the information presented in that course is directly applicable to administering IP facilities in a cluster.

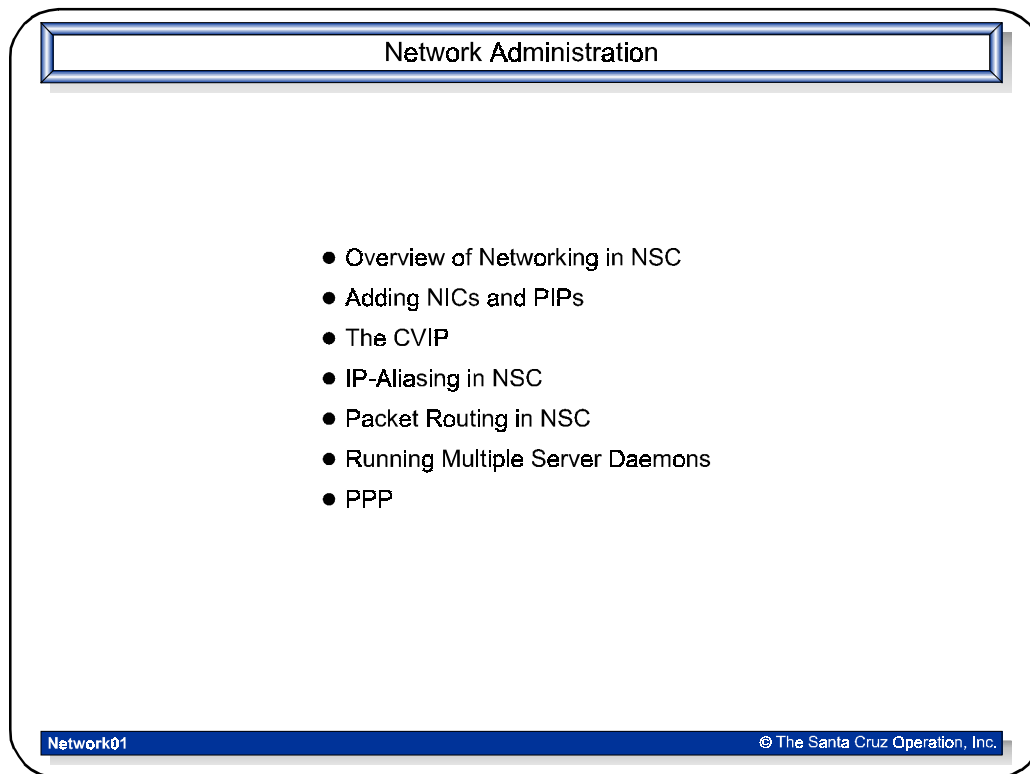
This lesson discusses how network administration is different between a stand-alone host and a cluster, and details procedures for performing that administration.

Objectives

By the end of this lesson, you will be able to:

- Explain how IP address and TCP and UDP port spaces are viewed within a cluster
- Manage NICs in a cluster
- Define the terms CVIP and PIP and describe their differences
- Describe the features and limitations of a CVIP
- Explain how traffic flows in and out of nodes in a cluster with a CVIP
- Create IP aliases for CVIP interfaces
- Describe how multiple instances of a server-daemons can be used in a cluster
- Describe PPP limitations within a cluster

AV 5-1

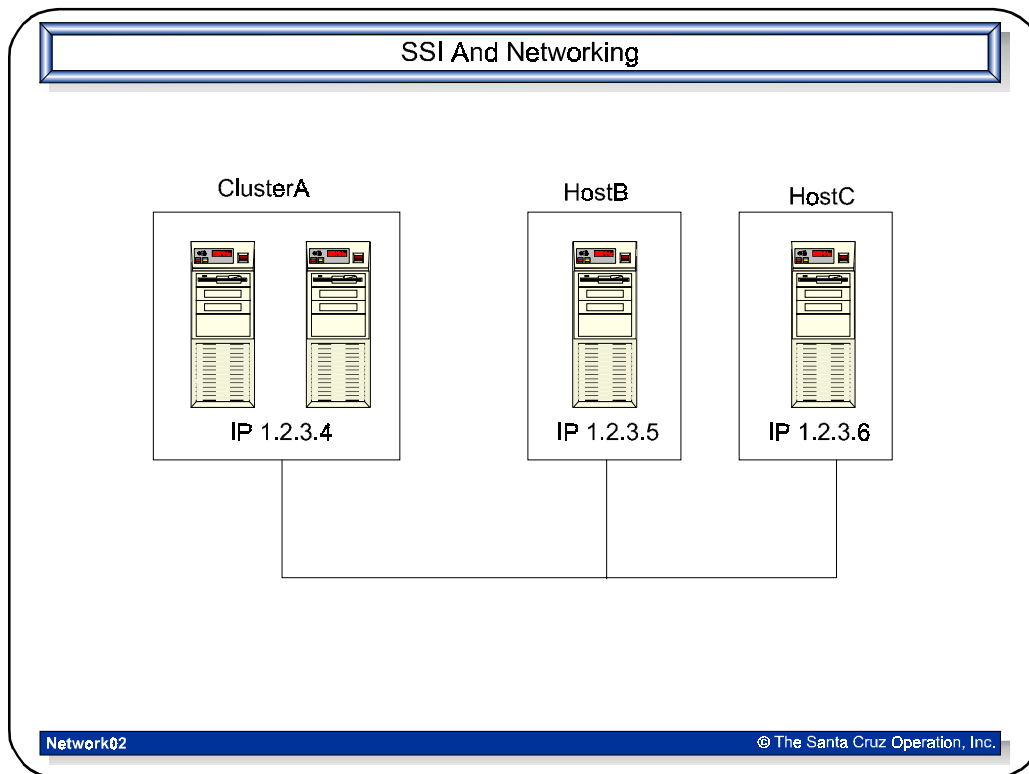


Topics

In order to meet these objectives, this lesson discusses:

- Overview of networking in NSC — the overall networking scheme used in NSC, including how the IP, TCP, and UDP address spaces are viewed
- Adding NICs and PIPs — how to manage physical network interfaces in a cluster
- The CVIP — use and limitations
- IP-aliasing in NSC — creating IP aliases to either PIPs or CVIPs, including advantages and disadvantages for each
- Packet routing in NSC — how packets are routed to and from cluster network interfaces, both physical and virtual
- Running multiple server daemons — why and how to run multiple server daemons
- PPP — how to configure PPP in NSC, including its features and limitations

AV 5-2



5.1 Overview of Networking in NSC

Recall from the *UnixWare® 7 NonStop Clusters® Concepts* course that the goal of single system administration (SSI) is to make a cluster appear as a single computing resource. With respect to networking, that means that the cluster should appear as a single host, as illustrated in AV 5-2 on page 5-6. This slide illustrates:

- The rest of the network sees the cluster by a single name, ClusterA
- The rest of the network sees the cluster by a single IP Address, 1.2.3.4
- For all intents and purposes, the cluster acts like it has only one LAN NIC, just like a host

Exercise: Connecting to a cluster using telnet

From a machine other than the cluster:

Step 1. Use a **telnet** client and connect to the IP address or name of the cluster.

Step 2. Once connected, login to the cluster as **root**.

Step 3. Enter the command:

ifconfig -a

How many interfaces are there on the same network?

Step 4. View the contents of the */etc/hosts* file.

What is the IP address of the cluster?

Which interface matches the IP address of the cluster?

Although the cluster acts as a single host, to prevent a single point of failure, the cluster must have at least two LAN NICs on each subnet to which the cluster attaches. Thus, in Step 3 in the preceding exercise, you saw multiple LAN NICs with IP addresses on the same subnet. Additional LAN NICs may also be added to increase network bandwidth from a cluster. This is described in Section 5.2, “Adding NICs and PIPs,” on page 5-11.



To make the cluster appear as a single node on the network, NSC uses an object known as a cluster virtual IP address (CVIP). The CVIP was also introduced in the **UnixWare® 7 NonStop Clusters® Concepts** course. In Step 3 of the preceding exercise, the `cvip0` interface is the CVIP for the first subnet to which the cluster is attached.

The CVIP is a flexible and versatile tool. You can alias other IP addresses to the CVIP to enable multihoming and host virtual domains on your cluster.

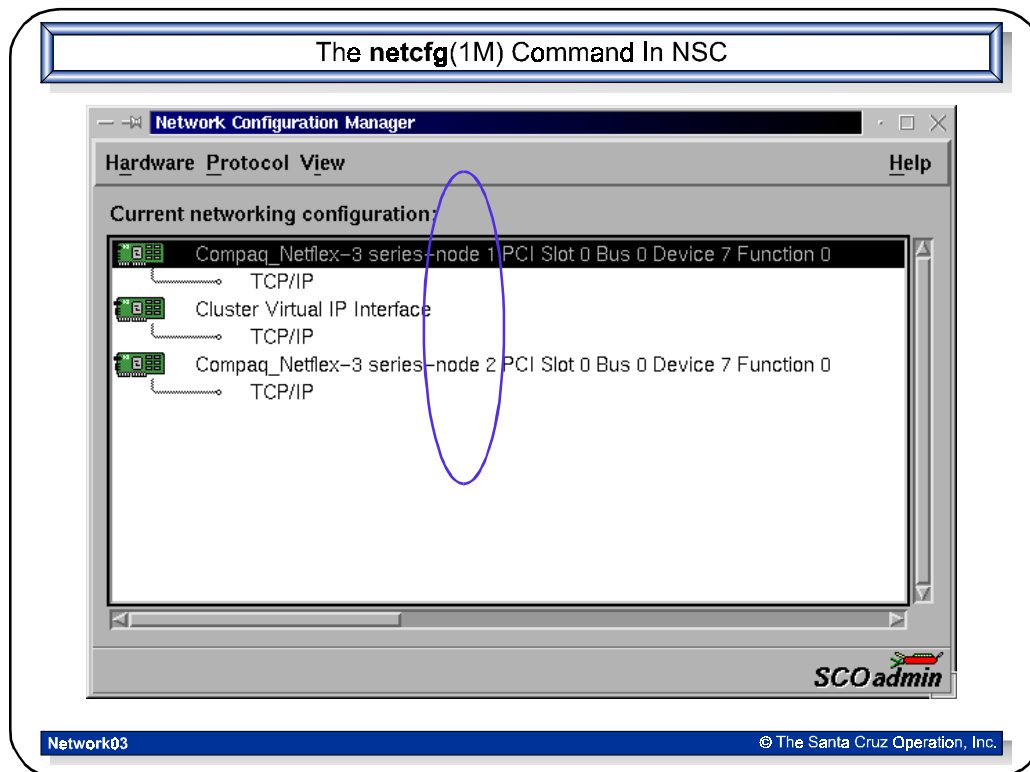
However, the CVIP and process migration can affect how packets are routed to network-related processes, especially during a node loss.

Since the CVIP uses a single IP address, and presents the cluster to the rest of the network as a single host, there are also issues related to running multiple copies of network-related processes on a cluster. Section 5.6, “Running Multiple Server Daemons,” on page 5-23 explores these issues.

Finally, there are limitations with PPP functionality in the current implementation of NSC.

Note For the remainder of this lesson, the term NIC refers to a LAN NIC, unless specifically stated otherwise.
--

AV 5-3



5.2 Adding NICs and PIPs

As a UNIX network administrator, there are two reasons you add a NIC to a host:

- For redundancy, using the failover NIC facility in UnixWare 7
- To attach the host to another subnet

Both of these reasons remain valid for adding NICs to nodes in a cluster. However, redundancy is best implemented in a cluster by making use of the CVIP facility, rather than the standard UnixWare NIC failover feature. In this case, the additional NIC is in active use, which has the added benefit of increasing bandwidth to the cluster.

You use the same utilities for managing NICs in NSC as you do in a non-NSC environment. That is you run **netcfg(1M)** or **scoadmin(1M)** ⇒ **Networking** ⇒ **Network Configuration Manager**. Either method brings up a window similar to the one shown in AV 5-3 on page 5-10. This slide was taken from a two-node cluster with one LAN NIC each. The CVIP was configured during the installation of the NSC software. Note:

- All network interfaces, cluster-wide, are displayed in the manager. You do not have to run **netcfg** separately on each node to manage their interfaces.
- The node to which each interface is attached is indicated in the detail line

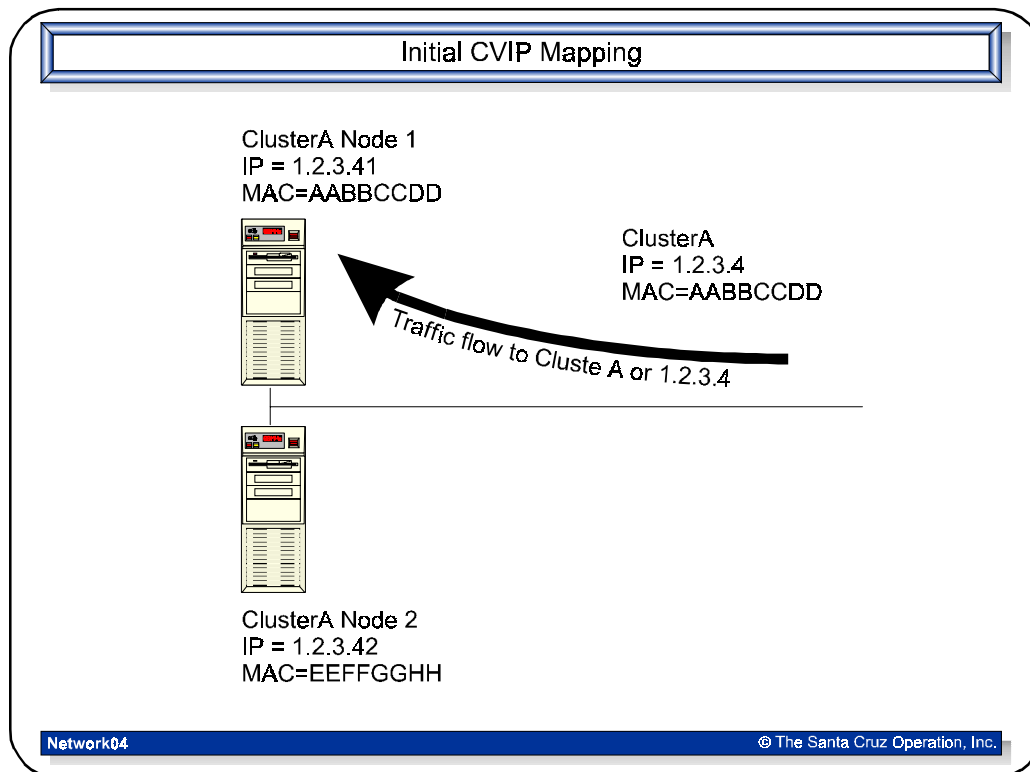
Note To distinguish CVIP and NIC interfaces, NIC interfaces are also referred to as physical IP (PIP) interfaces.

Use the same UnixWare 7 procedure to add a new NIC in an NSC environment. In the **Network Configuration Manager**, select **Hardware** ⇒ **Add New LAN Adapter**. When the list of NICs is displayed, the detail indicates the node to which the NIC is attached.

Exercise: Using netcfg

Start the **Network Configuration Manager**. Note the interfaces that are configured for the cluster.

AV 5-4



5.3 Understanding the CVIP

As previously mentioned, the goal of SSI with respect to networking is to allow the cluster to appear as a single host on the network, with a single IP address and host name. NSC uses the cluster virtual IP address (CVIP) to accomplish this.

As shown in the last section, the CVIP is just another network interface. This was illustrated by use of both the **ifconfig** and **netcfg** commands. The first CVIP is normally created when the NSC software is installed. You can create additional CVIPs, if needed.

This section describes the CVIP and related CVIP administration tasks in more detail..

5.3.1 How the CVIP works

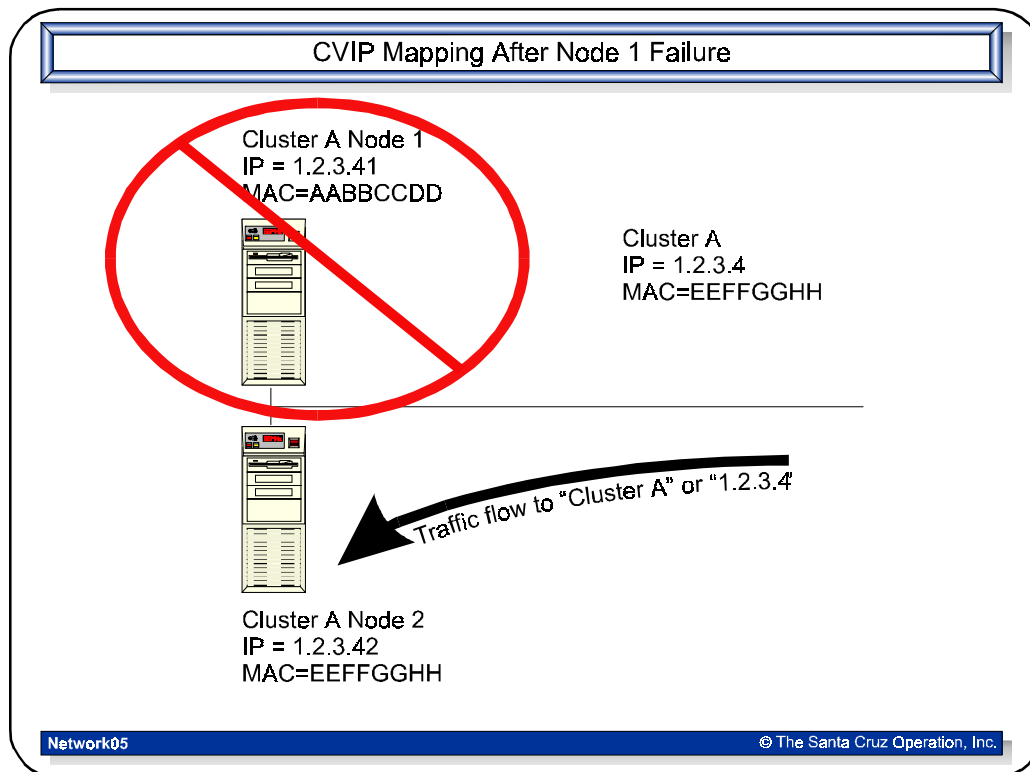
As discussed in the *UnixWare 7 Network Administration* course, users and network services specify a host name when connecting to another host. This name is then mapped to an IP address using name resolution, such as DNS, NIS, or */etc/hosts*. The IP address must then be mapped to the hardware address of the remote system. This is done by an *address resolution protocol (ARP)*.

A hosts sends a gratuitous ARP broadcast message to advertise its IP and MAC addresses to other hosts on a subnet of a LAN. Hosts on the network update their ARP cache entries with this address mapping.

When a cluster boots:

- One of the PIPs on each subnet is selected, and the IP address associated with the CVIP broadcasts the MAC address of that PIP. This effectively maps the CVIP to that PIP.
- Hosts on the network update their ARP cache with the IP address of the CVIP and MAC address of the associated PIP. This is illustrated in AV 5-4 on page 5-12.

AV 5-5



Hosts must also respond to incoming ARP requests. Since the CVIP is a virtual interface, not a physical piece of hardware, the CVIP to which the PIP is mapped provides proxy ARP service for that CVIP. That is, the PIP replies to ARP request asking for the MAC address associated with the CVIP's IP address using the PIP's own MAC address.

Exercise: Using arp(1Mtcp)

Run the **arp** command to see the address returned for the CVIP. For example, enter:

arp -a

To adhere to the principal that a cluster has no single point of failure, you must have at least two PIPs on each IP subnet. If one of the NICs fails, or a node fails, another PIP on the same subnet is selected and the CVIP migrates to the that PIP. This occurs automatically, as follows:

- The MAC address of the newly selected PIP is broadcast with the IP address of the CVIP as a gratuitous ARP
- All nodes on the subnet update their ARP tables with the new MAC address
- The newly selected PIP begins performing proxy-ARP for the CVIP
- The PIPs on the root and failover root nodes must be on the same subnet, along with the CVIP, to facilitate failover of the networking services

The complete list of conditions under which a CVIP can migrate include:

- A node with a network interface joins or leaves the cluster
- A CVIP address is added or removed
- A network interface is added or removed from one of the nodes in the cluster
- A node or network interface fails

The CVIP fails over automatically when the node or the network interface currently handling CVIP traffic fails. The interface monitor daemon, **ifmond**, performs the function of tracking the state of all network interfaces in the cluster and makes changes to the CVIP configuration as needed.

Whenever changes in the cluster network configuration are detected by **ifmond**, it executes a script that:

1. Creates a list of PIPs and CVIPs in the cluster
2. Determines whether the CVIP-to-MAC address mapping exists in the cluster's ARP table
3. Determines whether the assigned PIP is operational
4. Assigns a new PIP to the CVIP if the interface is not available

These criteria determine which interface is assigned to the CVIP:

- The interface must be on the same subnet as the CVIP
- PIPs on the root node are given preference
- If there are no other PIPs available on the root node, the next highest node number is given priority

Exercise: Observing CVIP Migration On Node Failure

- | | |
|---------|--|
| Step 1. | Login to the cluster as root . |
| Step 2. | Use the ifconfig command to determine which node contains the PIP to which the CVIP is currently mapped. |
| Step 3. | Use the node_self command to determine which node you are on. If it is the same node, logout and login explicitly to the another node on the cluster using telnet to connect to the PIP on the other node. |
| Step 4. | Simulate a failure the node with the CVIP by powering it off. |
| Step 5. | Run the ifconfig command again to determine which node contains the PIP to which the CVIP is now mapped. |

5.3.2 CVIP and name resolution

The cluster host name and CVIP address are usually assigned during UnixWare 7 and NSC software installation. After installation, you must configure your network's name resolution service with the cluster name and IP address of the CVIP if you want external hosts to connect to the cluster using the CVIP.

Whether you are using */etc/hosts*, Domain Name System (DNS), or another service, add the cluster name and CVIP address just as you would an individual network node.

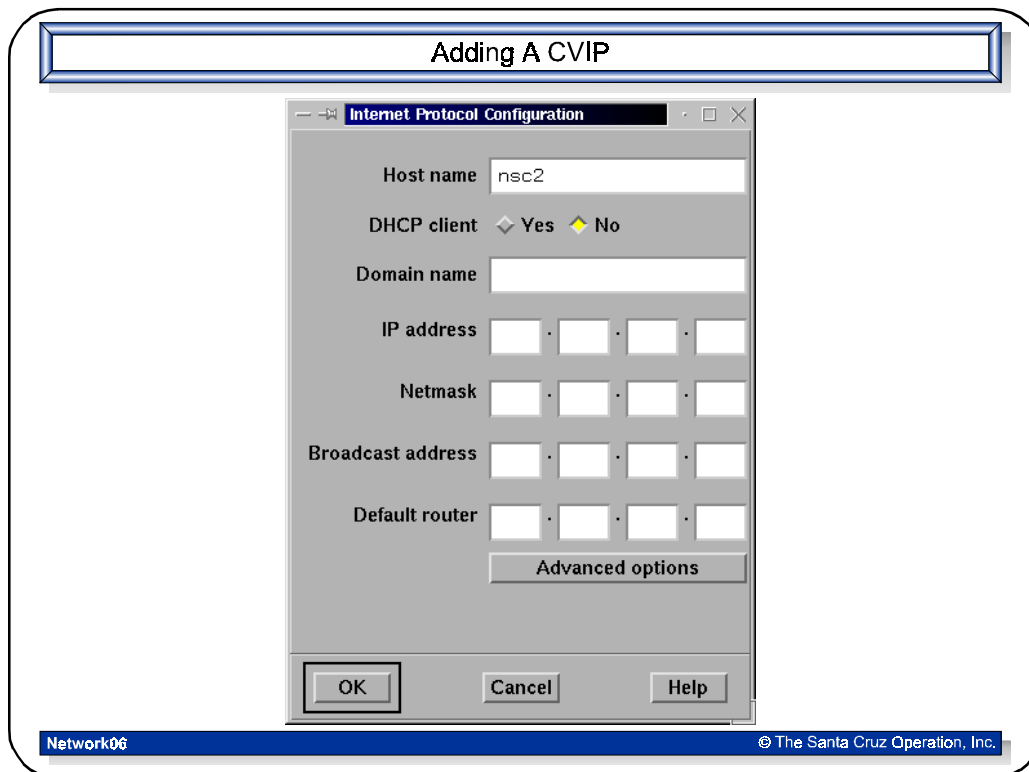
Note To ensure that Remote Procedure Calls (RPC) and other applications work after failover, the CVIP name must be set to the cluster name.

There are some cases where the PIPs within a cluster are registered with the name service, usually for performance benefits in certain environments. In general, only the CVIP and cluster name are made available to the name resolution service.

Optional Exercise: Configuring name resolution

Add the cluster name and CVIP to the name resolution service used in the classroom, as directed by your instructor.

AV 5-6



5.3.3 Adding CVIPs

If the cluster has two or more subnets, only the CVIP for the first subnet configured during software installation will be configured. Add a CVIP for each additional subnet to which the cluster is attached.

To add a CVIP:

- Step 1. Log into the cluster as **root**.
- Step 2. Run the **netcfg** command.
- Step 3. Make sure you are in the correct view; if you are uncertain, select **View** ⇨ **LAN**.
- Step 4. Select **Hardware** ⇨ **Add New CVIP**. This brings up a window similar to the one shown in AV 5-6 on page 5-18.
- Step 5. Fill out the form as you would for a new PIP on the subnet. Do *not* configure the CVIP as a DHCP client.
- Step 6. Click on **OK**. This adds the name/IP address pair to the */etc/hosts* file
- Step 7. The new CVIP appears as an icon in the **Network Configuration Manager** window.
- Step 8. Select **Hardware** ⇨ **Exit**.
- Step 9. If using another name resolution service, add the name/IP mapping information to that service.
- Step 10. Relink the kernel, if prompted when exiting **netcfg**.
- Step 11. Reboot the cluster if the kernel was relinked.

Note If you add a subnet, be sure to add two NICs on that subnet for redundancy.

5.4 IP Aliasing in NSC

Since a CVIP is a network interface supporting the IP protocol, you can create IP aliases to it. This is useful for hosting virtual domains, including web and mail services, just as you would on a non-NSC UnixWare 7 system.

For example, suppose you have the following:

- A new domain name of *newdom.com*
- A free IP address on your subnet of 1.2.3.100, which you are going to use for hosting the *newdom.com* domain
- The subnet of the IP address is the same subnet as that served by the *cvip0* interface

To create the IP alias, assign this IP address to the *cvip0* interface, enter:

```
ifconfig cvip0 alias 1.2.3.100
```

The alias will only last as long as the cluster remains up. To create the alias with every cluster boot, you must place an entry in the */etc/inet/config* file as discussed in the **UnixWare 7 Network Administration** course.

Since the IP alias is to a CVIP and not a PIP, the alias becomes fault tolerant and migrates to PIPs just as the CVIP does.

Exercise: IP alias failover

- | | |
|---------|---|
| Step 1. | Create an alias to your CVIP as indicated by your instructor. |
| Step 2. | Determine which NIC (PIP) is handling this new alias. |
| Step 3. | Force a failover of that node. |
| Step 4. | Use telnet to connect to the alias. |

Note that the connection succeeds, even though the PIP handling the CVIP is now on another node.

5.5 Packet Routing in NSC

Packet routing in a cluster occurs at two levels:

- Extra-cluster—between the cluster and external hosts
- Intra-cluster—between cluster nodes

Extra-cluster routing works essentially the same as between non-cluster hosts.

- The cluster routing daemons have been modified so that a PIP responds to ARP requests for the CVIP, as described in Section 5.3.1, “How the CVIP works,” on page 5-13
- Routing is configured in a cluster using the same tools and guidelines used for non-cluster UnixWare 7 systems

Intra-cluster routing is treated like a request to the loopback interface. If the source and target processes are on different nodes, requests are sent via the SAN to route the packets between said processes. Packets are routed between cluster nodes using two NSC-specific mechanisms:

- An NSC-specific driver, commonly referred to as the “glue” layer between the IP and TCP layers of the protocol
- Coordination of cluster-wide network operations by the *network registration database*, NeRD, which:
 - Maintains a global database that ensures connections are unique across the cluster
 - Manages state and location information for each network interface
 - Manages the routing table for the entire cluster

For incoming traffic, external hosts usually send packets to daemons listening on the IP address of the CVIP. The packets are passed to the node running the daemon.

For outgoing traffic, when a daemon sends response packets, NSC either:

- Routes the packets through a local NIC on the same subnet
- Sends a SAN message to transmit the packet through a NIC in the cluster on the same subnet.

In either of the above cases, the packet still claims to come from the IP address to which the request was originally sent.

Thus, all incoming traffic comes into the node which owns the CVIP, but outgoing traffic is distributed among the PIPs in the cluster.

WARNING! The kernel data structures used to communicate between TCP/IP-based services, *sockets*, are created on the node responding to the incoming request for a network service; that is, the node which owns the CVIP.

In UnixWare NSC 7.1.0, sockets are not migrated to other nodes when processes migrate. The implication of this limitation is this: If a connection is made to a process through the CVIP, the connection is lost if the node owning the CVIP fails. However, the CVIP migrates to another node, and new connections can be made through the new PIP mapping the CVIP.

5.6 Running Multiple Server Daemons

Each IP address has its own unique port space. That is, each IP address has a unique set of TCP and UDP ports used by client and server processes. This is true in both NSC and non-NSC environments.

Because NSC has several PIPs and one CVIP per subnet, it is possible to start server daemons on each of the PIPs as well as the CVIP. However, if hosts outside the cluster only connect to services within the cluster via the CVIP, this is counter-productive. You are creating extra CPU and RAM load for a process that will never be used.

However, you can still take advantage of NSC's parallel processing to run multiple copies of certain server daemons in one of two ways:

- Upon receiving a connection request, certain server daemons (for example **telnetd**) spawn a child process to accept the connection and service the client's requests. Configure the load leveling subsystem to distribute the load of these daemons. (see Lesson 7: "Load Leveling").
- If you are hosting virtual domains, and your web and/or mail servers allow this, you can configure separate server daemons for each IP address associated with each virtual domain. Each web/mail server can then be configured to be monitored by the failover daemon (see Lesson 8: "Failover") or load balanced (see Lesson 7: "Load Leveling") as you see fit.

Note Many Internet service daemons use a configuration file. The daemon must allow you to specify unique configuration file names for each instantiation in order for this scheme to work in NSC.

5.7 PPP

The PPP capabilities in NSC vary depending on the version.

NSC 7.1.0, 7.1.1

PPP can only be run on the current root node. It can only access devices on the node on which it runs. This has several consequences:

- If you configure PPP to run with clusterized device node names, for example `/dev/term/01m.1`, and the node with that device fails, PPP is unavailable for the entire cluster.
- Certain devices come with non-clusterized as well as clusterized device node names, such as `/dev/term/01m`. This non-clusterized device node name, refers (in this example) to COM1 on the current node. Using this device node name allows PPP to run on either of the potential root nodes. But you must have the associated equipment attached to each serial port.

Further, PPP connections do not survive failover.

NSC 7.1.1b

PPP can be run from any node. It can use hardware devices attached to any node. For redundancy, create multilink bundles with modems from multiple nodes. If the destination ISP does not support multilink PPP, set the maximum number of links in use to 1.

Regardless of the version you are running, the PPP daemon is monitored by the **keepalive**(1M) daemon. Therefore, you cannot stop the **pppd**(1M) daemon using the standard command:

ppptalk stop

Instead, you must run the shutdown script:

/etc/rc2.d/S71ppp stop

5.8 Lesson Summary

This lesson discussed the differences in performing network administration in a NSC environment. Topics discussed include:

- Overview of networking in NSC
- Adding NICs and PIPs
- The CVIP
- IP-aliasing in NSC
- Packet routing in NSC
- Running multiple server daemons
- PPP

Review questions

1. What events can cause a CVIP to migrate to another PIP?
2. If your cluster has 4 subnets, how many CVIPs should you configure? Why?
3. How many NICs should you have for every subnet in a cluster?
4. Give an example of how you could use CVIPs to host a virtual domain on your cluster. Why would you want to do this?
5. What are the limitations of PPP in the version of NSC used in your classroom?

5.9 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

arp (1Mtcp)	Provides address resolution protocol services
ifconfig (1Mtcp)	Configures network interfaces
keepalive (1)	Restarts failed processes in a cluster environment
netcfg (1M)	Configures network interfaces; can be invoked from the command line, or from scoadmin (1M)
node_self (1)	Displays the current node number
pppd (1M)	Establishes PPP connections
ppptalk (1Mtcp)	Starts and stops PPP connections
telnet (1Mtcp)	Connects to telnetd on the specified host

Relevant files

<i>/etc/hosts</i>	Is the local table for host name to IP address matching. See <i>hosts</i> (4tcp).
<i>/etc/rc2.d/S71ppp</i>	Starts and stops pppd (1)
<i>/etc/resolv.conf</i>	Is the name resolution configuration file. See <i>resolv.conf</i> (4tcp).

5.10 Answers to Exercises, Review Questions, and Labs

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Exercise answers

“Connecting to a cluster using telnet,” on page 5-7

The answers to this exercise are dependent upon your cluster configuration. Your instructor will provide you with the correct answers.

“Using arp(1Mtcp),” on page 5-15

The answers to this exercise are dependent upon your cluster configuration. Your instructor will provide you with the correct answers.

“Observing CVIP Migration On Node Failure,” on page 5-16

There is no answer associated with this exercise.

“IP alias failover,” on page 5-20

There is no answer associated with this exercise.

Review Question Answers

This section contains the answers to the review questions on page 5-25.

1. These events are monitored by **ifmond** and can cause a CVIP to migrate.
 - A node with a network interface joins or leaves the cluster
 - A CVIP address is added or removed
 - A network interface is added or removed from one of the nodes in the cluster
 - A node or network interface fails
2. If your cluster has 4 subnets you should have at least four CVIPs, one for each subnet. If you do not have a CVIP for a subnet, you will not have a failover solution in case a PIP on that subnet fails.
3. You must have at least two NICs for each subnet for redundancy. In a cluster environment, these can both be active NICs, as opposed to the active/backup NIC failover feature in non-NSC UnixWare environments.
4. You can create an IP alias to the CVIP and then register this IP address and a virtual domain name with the name resolution service. The new IP address and domain name are advertised through the name service, so network users and services can connect just they would to any host. Because this virtual domain is aliased to the CVIP, rather than a PIP, it remains accessible even in the event of NIC or node failure in the cluster.
5. PPP functionality depends on the release of NSC. See the **UnixWare 7 NonStop Clusters Release Notes** for information about PPP limitations.

6

Startup and Shutdown



Lesson 6

Startup and Shutdown

Introduction

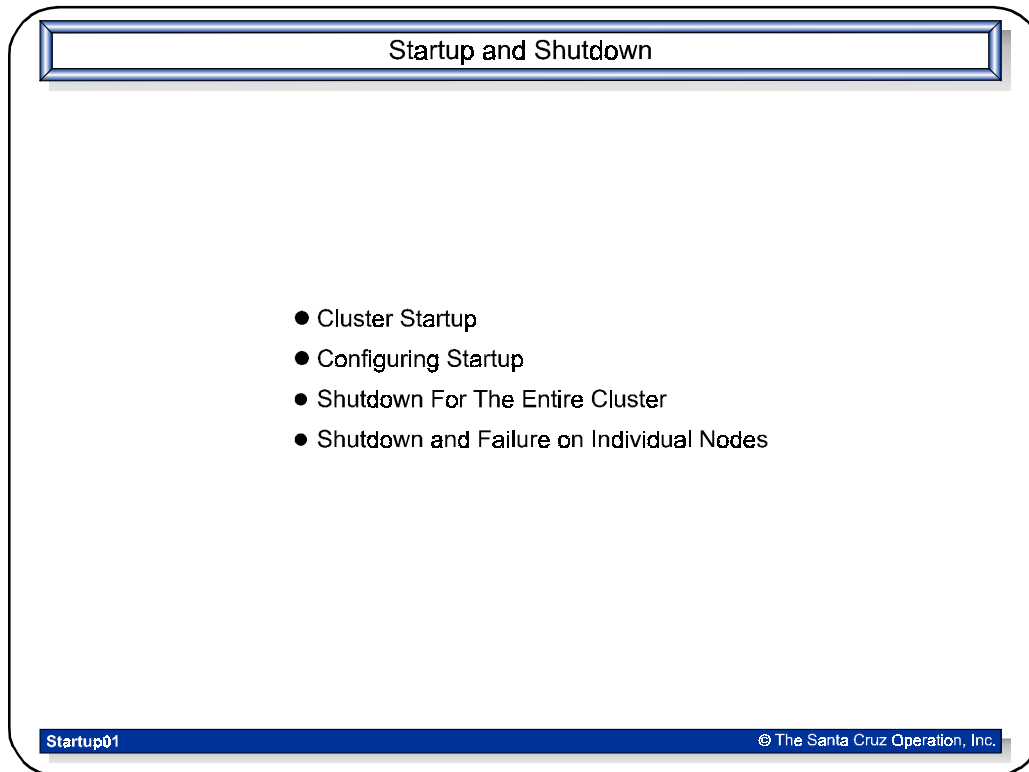
This lesson discusses how the startup and shutdown processes are different between a stand-alone host and a cluster. It also details procedures for how startups and shutdowns occur, either per node or cluster wide, and discusses node failures and recoveries.

Objectives

By the end of this lesson, you will be able to:

- Describe the startup process for a single node, noting:
 - Differences between the first and additional nodes to boot
 - Differences between potential root nodes and other nodes
- Specify the order in which nodes should be booted
- Explain what happens if, during startup, a node cannot find a root node, or join the cluster
- Perform an orderly shutdown of a single node and an entire cluster
- Describe how the cluster reforms when a node leaves the cluster

AV 6-1



Topics

In order to meet these objectives, this lesson discusses:

- Cluster startup — starting nodes in a cluster
- Configuring startup — new *boot(4)* and *inittab(4)* parameters, files, and scripts
- Shutdown for the entire cluster — shutting down the entire cluster
- Shutdown and failure on individual nodes — performing an orderly shutdown on any node in the cluster. It also discusses the ramifications of a node shutdown, cluster reformation, and failover.



6.1 Cluster Startup

On a standalone host system where storage is typically internal to the computer, you generally boot the system by powering on the computer. For NSC, the procedure is slightly more complex because of the existence of multiple computers and possibly external shared storage.

This lesson presents the general procedure for booting a cluster, as well as discussing what happens on the various nodes during the boot procedure.

6.1.1 Cluster boot procedure

The procedure for booting a cluster is:

- Step 1. Power on any SAN or LAN switches or hubs.
- Step 2. Power on any external storage devices, such as Fibre Array or JBOD cabinets, switches, or hubs.
- Step 3. Wait for the devices powered on in Steps 1 and 2 to fully initialize themselves. Often these devices have firmware that *boots* the device and then indicates success or failure through some LEDs.
- Step 4. Power on the node you want to be the root node. This is node 1, unless that node is currently inoperative.
- Step 5. If the cluster is using CNM, power on node 2 (the failover root node) within seconds of powering on node 1. This ensures that node 1 will be the root node, but that the *root* filesystem mirror between the nodes will not be considered stale and have to be re-mirrored. See Section 6.1.2, “Booting CNM clusters,” on page 6-8.
- Step 6. Regardless of the protected storage method used for the *root* filesystem, power on the remaining nodes in the cluster, one at a time. Because computers require more power for the first few seconds after power-on than they do to continue operating, space the node-powerups a few seconds apart to prevent overloading your facility’s power circuits.

Unless there is a catastrophic failure, each cluster node presents a login prompt when the sequence is completed. Section 6.1.3, “The startup sequence,” on page 6-9 detail this sequence so you can troubleshoot the boot process, if necessary.

6.1.2 Booting CNM clusters

When booting a cluster using CNM, the timing of the power-on of nodes 1 and 2 is critical:

- Power on node 1 before node 2 so that node 1 becomes the root node. This is important if you want to run certain things like X windows on the console, which can only be run on node 1, and then only if it is the root node in versions 7.1.0 and earlier. As NSC matures, the number of these dependencies decreases. But as of release 7.1.0, they are still an issue.
- Power on node 2 sufficiently *later than* node 1, so that node 1: 1) Finishes its power-on self test (POST); 2) Boots UnixWare 7; and 3) Elects itself as root node—before node 2 beats node 1 to the election.
- Power on node 2 sufficiently *soon after* node 1 so that node 1 knows that node 2 is joining the cluster and does not force a re-mirror of the *root* filesystem, during which the cluster is vulnerable to a single point of failure in node 1.

Timing constraints require that node 2 join the cluster after node 1 has elected itself as the root node but before it checks to see if node 2 is available to mirror the *root* filesystem. If nodes 1 and 2 are comparably configured (same make, model, number of CPUs, disk configuration, and the same amount of RAM), then powering on node 2 within two-five seconds of powering on node 1 should accomplish this goal.

To determine whether the timing was appropriate, check the following after cluster boot:

- Check to see if the *root* filesystem mirror is being updated.

If the **vxrecover**(1M) process is running with an argument of **rootvol-02** immediately after boot, you started node 2 too late. After the process finishes, indicating that the *root* filesystem is remirrored, shutdown the cluster and try again, starting node 2 earlier. If nodes 1 and 2 are configured differently, you may even need to start node 2 before node 1.

- Determine which node owns the *root* filesystem using the **where** / command line.

If **where** / returns node 2, you started node 2 too soon. If this is the case, node 1 may have not only been started too late to claim the root node, but also late enough to have NSC consider the root volume stale and in need of remirroring, indicated by **vxrecover** running with an argument of **rootvol-01**. Wait for the remirroring to complete and then shut down the cluster and try again, starting node 2 later.

6.1.3 The startup sequence

This is the boot sequence for cluster nodes. Use this as a guide for troubleshoot problems during node startup.

1. The kernel is loaded according to the procedure outlined in the **boot(1M)** manual page.
2. The kernel initializes itself, including statically-linked device drivers. Just after the driver for the SAN is initialized, it will test both fabrics and select one.
3. The node determines its own node number using information in the **boot(4)** file, and sends a SAN message indicating it is joining the cluster in the COMINGUP state.
4. The node contacts the Cluster Member Services (CMS) node (also found through the **boot** file) to locate the current root node.
5. If there is no current root node, and this node is a potential root node, it becomes the root node and starts the CMS, listing itself as root.

The node then mounts the *root* filesystem and creates the **init(1M)** process, which proceeds with a normal system startup. Skip to Step 9.

6. If the node is not the root node, it performs a cluster-filesystem mount of the *root* filesystem.

If a node is not a potential-root node, and there is no root node, or if a node is a potential root node and fails to mount the *root* filesystem, the node panics.

7. The CMS re-forms the cluster. See Section 6.4.1, “Reforming the cluster,” on page 6-15.
8. The **init** process runs the */etc/cluster/nodeup* script for this node.
9. The node is marked in the UP state. If this node performs CNM, and the mirror is stale, then the Veritas software will start the **vxrecover(1M)** process to rebuild the mirror.

Note Remember, if a mirror is being rebuilt on a potential root node, and the root node fails, the potential root node, and all other nodes in the cluster, panic.
--

6.2 Configuring Startup

The *UnixWare 7 System Administration II* course presented a substantial amount of information about how to configure the boot process by modifying the */stand/boot* and */etc/inittab* files. All of that information is applicable to NSC, with these exceptions and additions:

- Each node has its own *boot(4)* file residing on its *stand* slice on its internal boot disk. When node *N* joins the cluster, its *stand* filesystem is mounted under the */stand/N* directory.
- There are additional, NSC-specific parameters for the *boot(4)* file, as discussed in the next section.

6.2.1 The *boot(4)* file in NSC

The boot parameter file for each node is documented in the *boot(4)* manual page. There is also documentation on the NSC-specific additions to the file in **scohelp(1X) ⇒ NonStop Clusters Documentation ⇒ System Administrators Guide ⇒ Administering SCO UnixWare 7 NonStop Clusters Booting and Initialization ⇒ Understanding Boot File Extensions**.

The parameters of note include:

CLUSTER_NODENUM The node number of this computer when it joins the cluster. This value will be unique for each node.

WARNING! If you are using Compaq ServerNet switches, the number used for this entry must be one less than the port-number to which you attached this node's Servernet cables. This is because Servernet switches start numbering at 0 instead of 1.

CLUSTER_ROOTNODE A comma-separated list of nodes that are potential root nodes. The order the nodes appear in the list is the order the of secession in the event the current root node fails. This parameter must be the same on all nodes. See the CLMS parameter for details on how nodes find the current root node.

PARAMS The name of a file, relative to the stand slice, where the boot procedure gets additional parameters, discussed later in this section.

SBADEV If you are using a split-brain avoidance cable, this entry indicates the device node to which the cable is attached on this cluster-node.

Note The name of the driver for the COM1 and COM2 serial ports is **iasy**.

CLMS This parameter is a comma-separated list of nodes providing cluster membership services. By default, this is the same as the list of root nodes specified in the **CLUSTER_ROOTNODE** parameter. This parameter must be the same on all nodes.

During boot, potential root nodes check the CLMS nodes to determine if there is already a root node. If not, they mount the root filesystem themselves and become the root node, registering themselves with the CLMS.

ROOTDISK For non-CNM systems, this parameter specifies the name of the device node that contains the root filesystem. It follows the naming convention documented in the *disk(7)* manual page.

6.2.2 The *inittab(4)* file and *init(1M)* in NSC

There are several differences you need to understand about **init** in NSC.

- There is only one version of the **init** command running at any time within the cluster.
- The **init** daemon is now state-aware. That is, it keeps the state of the processes it is monitoring in a file in the *root* filesystem.
- The **init** daemon runs on the root node. In the event the root node fails, the **init** process is restarted on the new root node using the state information left by the previous **init** process. This is performed during the failover and cluster reforming procedure.
- As indicated in Section 6.1.3, “The startup sequence,” on page 6-9, **init** performs differently when it is first started on a root node then it does when it is started on a dependent node.
- To support this node-dependent behavior, the *inittab* file has been modified as documented in the *inittab(4)* manual page. These modifications are discussed in the rest of this section.

The following enhanced syntax options for the *inittab* file support **init** processing in a cluster operating environment:

- The special characters U, D, and R are now allowed in the run-state (2nd) field of an *inittab* entry. When present, they indicate that the command in the last field should only be run when a node enters or leaves the cluster, or after a current root node failover, respectively.

Note Each <i>inittab</i> entry can contain only one of the U, D, or R characters.
--

- The %n and %g special tokens can be used in the last (process) field. These tokens have special meaning:

Run Level	%n Token Expansion	%g Token Expansion
U	The number identifying the node that is joining the cluster	The number of times, from 0–15, that the node has joined the cluster during this boot cycle
D	The number identifying the node that is leaving the cluster	The number of times, from 0–15, that the node has left the cluster during this boot cycle
R	The number identifying the failed node	The number of times, from 0–15, that init (1M) processing has recovered during this boot cycle

- You can now place a hash mark (#) at the end of the last (process) field, followed by a list of node numbers. This list is then a node-restriction list, enabling you to target specific node(s) to run the command. See *inittab*(4) for further information.

6.2.2.1 Execution order requirements

Normally, commands in the *inittab* file are executed in the order in which they appear. However, when a node joins or leaves a cluster, a SIGCLUSTER signal sets the node's **init** processing state. This event causes **init** processing to enter a special transient run level during which only entries with a U, D, or R are processed.

Failover recovery also causes **init** to enter a special state, during which only entries with an R are processed.

During ordinary run levels, or when a child process dies, **init** executes all commands in the *inittab* file sequentially, except entries that include the following:

- D or R run levels
- U run level, unless the `respawn` keyword is in the `action` field

6.3 Shutdown For The Entire Cluster

Because NSC implements SSI, shutting down the entire cluster works the same as shutting down a single-computer system. That is, use the **shutdown**(1M) or **init**(1M) commands with appropriate arguments to shut down the entire cluster.

This will cause the dependent nodes to be shut down, according to the rules in the *inittab*(4) file. The root node will shut itself down after all dependent nodes are down.

If your cluster has external devices such as SAN or LAN switches or hubs or external shared storage, do not power these devices off until you see the message on the console of the root node indicating that it is okay to power off the system.

6.4 Shutdown and Failure On Individual Nodes

One of the primary benefits of using NSC is exceptional availability. This is achieved in several ways:

- You can take individual nodes out of the cluster to perform scheduled maintenance or upgrades.
- The cluster can survive the failure of any one node. It can even survive the failure of multiple nodes, as long as one root node continues to run.

6.4.1 Reforming the cluster

When a node leaves the cluster or rejoins the cluster, the CMS must reform the cluster. These tasks are performed when the cluster is reformed:

- If a node is leaving the cluster, all processes that started on that node, and have since migrated, have virtual processes entries made for them on the *surrogate-origin* node—typically the current root node. This allows other processes within the cluster to find the migrated process.
- If the node leaving the cluster was the current root node, a new root node must be selected from the potential root nodes. The new root node must then unmount the *root* filesystem it mounted through the cluster filesystem (cfs), and re-mount it, physically. It must also re-start the **init** program and run any *inittab*(4) entries with an R in their 2nd field.
- If a node is re-joining the cluster, virtual process entries moved to the surrogate-origin node, if any, must be moved back to the rejoining node.
- Further, any other protected filesystems that were served by this node must be remounted according to the information in the *fdevtab*(4) file, as described in Section 4.2, “Filesystem Administration in NSC,” on page 4-11.

6.4.2 Controlled shutdown of an individual node

To deliberately shutdown an individual node in the cluster, use either **clusternode_shutdown(1M)** or **node_down(1M)**.

6.4.2.1 The **clusternode_shutdown(1M)** command

The **clusternode_shutdown(1M)** command is logically equivalent to the **shutdown(1M)** command on a non-NSC system. That is, it performs an orderly shutdown of the node:

- Stop all processes currently running on the node.
- Process any *inittab(4)* entries with a **D** entry in their second field that are applicable to this node.
- Unmount any filesystems currently mounted by this node. If this is the current root node, a potential root-node is then engaged. If this node is serving non-root protected filesystems, they are unmounted from this node and remounted on one of the other protecting nodes.
- Prepare the system for power-off.
- Use the **-N** flag to specify the node number.

This command is the preferred method for shutting down an individual node.

6.4.2.2 The **node_down(1M)** command

In the event of an emergency, you can use the **node_down(1M)** command. This command is only slightly better than simply powering-off the node. The **node_down(1M)** manual page issues dire warnings about possible bad consequences of using this command. Use it at your own risk.

6.4.3 Reboot parameters

When a cluster node panics, it generates a memory image dump on the `/dev/dump.N` device (where *N* is a node number). During system boot you are prompted to save the dump image. The time, measured in seconds, that the system waits for a response depends upon the value of the `TIME` parameter in the `/etc/default/dump` file.

- Set the `TIME` parameter to 0 if you do not want to be prompted.
- Set the `TIME` parameter to a positive number to allow time to respond to the question. If you do not respond, the dump will not be saved and the boot proceeds automatically.
- Set the `TIME` parameter to a negative number or leave it out of the file entirely, to force the system to wait indefinitely.

The default is `TIME=15`.

6.5 Lesson Summary

This lesson presented issues related to starting and stopping an entire cluster and specific nodes within a cluster. The topics presented were:

- Cluster startup
- Configuring startup
- Shutdown for the entire cluster
- Shutdown and failure on individual nodes

Review questions

1. State the order in which nodes should be booted.
2. Why is it important to power up node 2 just after node 1 in a cluster using CNM?
3. Which node(s) have *boot* files? Where is the *boot* file located?
4. Does *init* run on every node in a cluster?
5. What actions are performed when a root node leaves a cluster?

6.6 Lab: Rebooting the Cluster

In this lab, you will use the **clusternode_shutdown(1M)** and **shutdown(1M)** commands to reboot individual nodes, and then the entire cluster.

Procedure

Use this procedure to reboot node 1, and shutdown node 2.

Step 1. Login to the cluster as **root**.

Step 2. Start a Korn shell on node 2. Enter:

```
onnode -p 2 ksh
```

Step 3. Reboot node 1. Enter:

```
clusternode_shutdown -N 1 -i 6 -g 0 -y
```

Step 4. Start a korn shell on node 1. Enter:

```
onnode -p 1 ksh
```

Step 5. WARNING: If the cluster uses CNM, wait until any re-mirroring is finished as discussed in Section 6.1.2, “Booting CNM clusters,” on page 6-8.

Step 6. Shutdown node 2. Enter:

```
clusternode_shutdown -N 2 -g 0 -y
```

Step 7. Reboot node 2.

Use this procedure to shut down the entire cluster.

Step 1. Login to the cluster as root.

Step 2. Enter:

```
shutdown -g0 -y
```

Reboot the cluster according to the procedure in Section 6.1, “Cluster Startup,” on page 6-7.

6.7 Lab: Modifying *inittab*(4)

In this lab, you will modify the */etc/inittab* file to announce that a node is coming up, going down, and when a new node is selected as the root node.

Procedure

Step 1. Login to the cluster as **root**.

Step 2. Edit the */etc/inittab* file. Add these lines at the end of the file:

```
l1:2345U:wait:/usr/sbin/onall echo "node %n up" >/var/adm/syslog
l2:2345D:wait:/usr/sbin/onall echo "node %n down" >/var/adm/syslog
l3:2345R:wait:/usr/sbin/onall echo "root node %d failedover" \
>/var/adm/syslog
```

Step 3. Tell **init** to re-read the *inittab* file. Enter:

telinit q

Step 4. Power off a non-root node.

Step 5. View the message in */var/adm/syslog*.

Step 6. Power on the node you powered off in Step 4.

Step 7. View the message in */var/adm/syslog*.

Step 8. Power off the root node.

Step 9. View the message in */var/adm/syslog*.

Step 10. Power the former root node back on.

Step 11. View the message in */var/adm/syslog*.

6.8 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

boot (1M)	Starts up a the system
clusternode_shutdown (1M)	Shuts down a node in a cluster
init (1M)	Initializes process control
nodedown (1)	Shuts down a node immediately
onnode (1)	Executes a command on a specific node
shutdown (1M)	Shuts down an entire cluster
telinit (1M)	Tells init to re-read the <i>inittab</i> file
vxrecover (1M)	Updates a Vertias disk mirror

Relevant files

<i>asyc</i> (7)	Controls serial ports COM1 and COM2
<i>disk</i> (7)	Describes device node naming conventions
<i>/etc/boot</i>	Is the configuration file for boot (1M). See <i>boot</i> (4).
<i>/etc/inittab</i>	Is the configuration file for init (1M). See <i>inittab</i> (4).
<i>/etc/utmpx</i>	Tracks user and accounting information
<i>/etc/wtmpx</i>	Tracks user and accounting information

6.9 Answers to Review Questions

This section contains the answers to review questions presented in this lesson.

Review Question Answers

This section contains the answers to the review questions on page 6-18.

1. Nodes should be booted in this order:
 - Power on the root node.
 - Power on the failover root node.
 - Power on remaining nodes.
2. It important to power up node 2 within seconds of node 1 in a cluster using CNM to ensure that node 1 will be the root node and the root filesystem mirror will be up to date.
3. Each node has a *boot* file located in its *stand11* slice on its internal boot/root disk.
4. The **init**(1M) process runs only on the root node.
5. When a root node leaves a cluster:
 - The CMS reforms by selecting a new root node from the list of potential root nodes.
 - The new root node unmounts the *root* filesystem from the cluster filesystem (cfs) and mounts it physically.
 - The new root node restarts **init** and runs *inittab* entries with R in the second field.

7

Load Leveling



Lesson 7

Load Leveling

Introduction

NSC is superior to simple failover solutions for many reasons. One reason is that it has multiple nodes always running the operating system and therefore ready to run applications. NSC increases this superiority by allowing administrators to balance the process work-load between these nodes. In NSC, this is called *load leveling*.

Load leveling can be done for only certain programs, for only certain nodes, or a combination of both.

This lesson discusses how to configure load leveling in an NSC cluster.

Objectives

By the end of this lesson, you will be able to:

- Turn load leveling on and off
- Configure load leveling
- Describe the daemons that implement load leveling
- Explain how to manually perform and override load leveling
- Describe the types of applications that should not be subjected to load leveling

AV 7-1

The slide is titled "Load Leveling" in a blue-bordered box at the top. Below the title is a bulleted list of seven topics. At the bottom, there is a blue footer bar containing the text "Loadlevelling01" on the left and "© The Santa Cruz Operation, Inc." on the right.

- Load Leveling Goals
- Load Leveling Concepts
- The Mechanics of Load Leveling
- Configuring Load Leveling
- Starting and Stopping the Load Leveling Daemons
- Overriding Load Leveling
- Load Leveling Domains

Loadlevelling01 © The Santa Cruz Operation, Inc.

Topics

In order to meet these objectives, this lesson discusses:

- Load Leveling Goals — how and when to use load leveling
- Load Leveling Concepts — reviewing the N+0 and N+1 concepts presented in the *UnixWare® 7 NonStop Clusters® Concepts* course
- The Mechanics of Load Leveling — understanding automatic load leveling
- Starting and Stopping Load Leveling Daemons — starting and stopping load leveling using the command line and the **keepalive** subsystem
- Configuring Load Leveling — understanding and modifying the various configuration files used by load leveling
- Overriding Load Leveling — excluding applications from load leveling
- Load Leveling Domains — grouping nodes for load leveling

AV 7-2



7.1 Load Leveling Goals

NSC includes a load leveling mechanism that attempts to distribute load evenly among the nodes in a cluster. This can be expressed mathematically:

Given N nodes, and a total load of L , the load on each node should be L/N

where the term *load* refers to CPU load only. Ideally, this results yields load graphs similar to those illustrated in AV 7-2 on page 7-6.

The NSC load leveling facility tries to balance the load among its nodes by:

- Migrating processes off of an overloaded node and onto an underloaded node
- Starting processes on underloaded nodes

While NSC's load leveling facility helps you achieve this goal, you will probably need to manually configure the load level mechanism to achieve better results for your particular mix of applications and user requirements. The default load-leveling configuration is often unable to effectively distribute your cluster load for these reasons:

- Processes do not exert an equal load on a node throughout their existence. That is, they spend some time waiting, some time running.
- Different programs running in processes exert different loads on the node. For example:
 - Accounting applications spend much of their time waiting on keyboard or disk I/O. When they do run, they are CPU-intensive for very short periods of time.
 - Graphics programs spend some of their time waiting on keyboard I/O. Once it is received, they are CPU-intensive for longer periods of time.

- Nodes may have varying numbers of CPUs. Moving a process to a node with fewer CPUs has a greater impact on that node than moving the same process to a node with more CPUs.
- You do not want to subject all process to load leveling. For example, you do not want to migrate the **init** daemon. Other examples are the **onnode** and **node_self** commands.
- Some process cannot be migrated by the load leveling mechanism. Load leveling is achieved by signal processes using SIGMIGRATE. Some processes are *pinned* to a node, which means they ignore SIGMIGRATE and therefore are not affected by load leveling.

7.2 Load Leveling Concepts

The NSC load leveling facility migrates process among nodes based on CPU load calculated for each node. CPU load is based on the number of processes waiting in the run queue averaged over a specified time period.

As a simple example, consider a four node cluster. At a given time, the run queues for each node are checked and the results are:

CPU	Run queue length
Node 1	1
Node 2	4
Node 3	5
Node 4	1

The load average for the cluster is calculated by adding the run queue length for all nodes and dividing that by the number of nodes. In this case, that is 11 divided by 4, or 2.3.

This cluster load average is then used to determine which nodes are over- and underloaded. How these values are calculated and used by the load leveling facility is determined by a set of configurable parameters defined in these files:

- */etc/cluster/load_level/parameters*
- */etc/cluster/load_level/rexec_commands*
- */etc/cluster/load_level/migrate_commands*

Before you attempt to configure load leveling, you need to understand these concepts and how they are implemented by the load-leveling mechanism.

Load information exchange	Nodes participating in load leveling must exchange their current load information with each other.
Exchange vector	The amount of information exchanged increases exponentially with the number of nodes. To minimize SAN traffic, NSC allows you to configure the number of nodes exchanging information at each iteration. The nodes that participate are then chosen at random at each exchange and are considered representative of the nodes in the cluster. The loads gathered from the nodes are called the load vector. Further, each node exchanges its load level averaged over the past 5, 30, and 60 seconds.
Load weights	NSC allows you to configure the importance of each of the time spans in determining the load of a node. Assign each time period a number between 0 and 100, with the total weight equal to 100.
Load averaging	The load of all nodes is added together and divided by the number of participating nodes. This number is called the cluster's load average.
Overloaded node	A node that has a load higher than the average is considered <i>overloaded</i> . NSC allows you to configure how overloaded a node must be before processes are migrated from the node.
Underloaded node	A node that has a load lower than the average is considered <i>underloaded</i> . NSC allows you to configure how underloaded a node must be before processes are migrated to the node.

Process weight	Once NSC has identified one or more overloaded and underloaded nodes, it will migrate one or more processes to effect a load that is within the specified parameters. To determine how many processes need to migrate, NSC approximates how much the load on a node will change due to a process migration. This number is called <i>the process weight</i>. NSC allows you to configure this number. Since you only have one number, it must represent the typical application on your cluster. Set this number to the percentage of time you believe the process spends running instead of sleeping. For example, if the average process spends 75% of its time sleeping, set this number to .25 (25% running).
Migration wait time	If NSC migrates processes that are not receiving any CPU time, then the migration may not help level the load. Therefore, NSC sets a minimum CPU time that a process must accumulate on a node before it is migrated. NSC allows you to configure this number.
rexec and rfork	The rexec(2) NSC system call is a derivative of the exec(2) non-NSC system call. It performs the same function as exec, except that it migrates the program to a specified node before performing the actual exec code. Similarly, the rfork(2) NSC system call is a derivative of the non-NSC fork(2) system call, except it creates the new process on the specified node. If a programmer wants to create a new process on a new node, they can either rfork then exec, or fork then rexec. The latter is more efficient because it generates less SAN traffic.

Load-leveling domains

When load leveling is activated, all nodes in the cluster participate by default. You can change this behavior in these ways:

- Prevent nodes from being migrated
- Group nodes to use separate load leveling parameters. A set of nodes is called a *load leveling domain*.

If you place all of your nodes in one or more load leveling domains, each domain is in an N+0 configuration. That is, in the event of a node failure, no resources are being held in reserve.

If you leave one node out of a load leveling domain, it is configured in a N+1 configuration. That is, in the event of a node failure, the node left out can be configured into the load leveling domain, keeping the performance roughly constant.

For absolute best use of your cluster investment, configure all nodes into one or more load leveling domains. Ensure that the performance of each domain exceeds requirements such that if one node is lost, the performance in the domain will at least meet minimum performance requirements.

7.3 The Mechanics of Load Leveling

The load-level daemon does its work in a completely distributed manner. After startup, one copy of the daemon runs on each processor node in the system. Each node's load leveler behaves identically; that is, there is no master/slave relationship nor is there a central repository for load-level data. Instead, load-level messages are passed periodically among the nodes.

Load leveling in NSC is accomplished as follows:

- At startup, the load-leveling daemon, **load_leveld**(1):
 - Reads the default configuration file, */etc/cluster/load_level/parameters*
 - Reads exclusion and inclusions lists in */etc/cluster/load_level/rexec_commands* and */etc/cluster/load_level/migrate_commands*, if these files are present
 - Starts one additional **load_leveld** for each node in the load-leveling domain
- The load-leveling daemons:
 - Exchange load information and add it into their *load vector* and calculate the load of each node according to the information in the *parameters* file
 - Balance load among nodes:
 - They start new processes on underloaded nodes, using **rexec**(2). This occurs only when **load_leveld** is running and the node starting the process is considered overloaded. Use the **-N** flag to prevent **load_leveld** from using **rexec**, if you do not want this behavior.
 - Periodically migrate processes to underloaded nodes using **SIGMIGRATE**. The period is determined by an entry in the *parameters* file. Use the **-n** flag to prevent **load_leveld** from performing migration, if you do not want the potential side-effects migration entails. See Section 3.3, “Multi-Node Execution,” on page 3-18 for more detail about the impacts of process migration.

The *parameters(4)* File

```
#
# This file specifies the parameters used by the load leveler daemon

send_timeout      2.0 # frequency in secs for sending load info.
redispatch_timeout 5.0 # frequency in seconds for re-dispatching procs
first_weight_factor 50 # 5 second weight factor
second_weight_factor 25 # 30 second weight factor
third_weight_factor 25 # 1 minute weight factor
number_vector_elements 8 # number of load vector elements
migrate_inclusion_flag 0 # 0 = exclusion list, 1 = inclusion list
rexec_inclusion_flag 0 # 0 = exclusion list, 1 = inclusion list
minimum_underload 1.0 # node is underloaded if underload <= parameter
minimum_overload 2.0 # node is overloaded if overload >= parameter
minimum_cputime 4.0 # minimum runtime of process for migration
per_process_load 1.0 # load attributed to a single process
nodes_to_use all # specify all nodes
rexec_bias 4.0 # per node amount to bias rexec
```


7.4 Configuring Load Leveling

You can start **load_leveld** from the command line, or it can be started by the keepalive subsystem. Start the daemon from the command line while you are trying different parameters to achieve the best configuration for your particular cluster requirements. Then, once you have established and tested the configuration, register **load_leveld** with keepalive so it will be restarted automatically in the event of node or process failure.

Before starting **load_leveld** you must understand how to modify the configuration files and use command line flags to make most effective use of the load leveling facility. After you make configuration changes described in this section, you can start the **load_leveld** as described in Section 7.5, “Starting and stopping the load leveling daemons,” on page 7-18.

7.4.1 The *parameters(4)* file

When **load_leveld** starts, it reads the */etc/cluster/load_level/parameters* file. The default *parameters* file contains the entries shown in on page 7-14.

Most of the entries in the file are documented in the *parameters(4)* manual page and in Section 7.2, “Load Leveling Concepts,” on page 7-9.

These require more explanation:

migrate_inclusion_flag and *rexec_inclusion_flag*

These entries indicate whether program names listed in the *migrate_commands* and *rexec_commands* files (respectively) are a list of commands to be subject to load balancing, nor *not* to be subject to load balancing.

rexec_bias

This entry indicates how much the load leveling daemon should use **rexec** instead of **SIGMIGRATE** to balance loads. It is less expensive—in terms of performance—to start a new process on a remote node than to move an existing process between nodes.

Exercise: The *parameters(4)* file

View the default *parameters* file on your cluster. Use the *parameters(4)* manual page to familiarize yourself with the default entries. You will need to modify these in a later exercise.

7.4.2 The *migrate_commands(4)* and *rexec_commands(4)* files

When **load_leveld**(1) starts, in addition to reading the *parameters(4)* file for each domain, it also reads two additional files, */etc/cluster/load_level/migrate_commands* and */etc/cluster/load_level/rexec_commands*. These files are referred to as the *migrate* and *rexec* files, respectively, and are documented in the *migrate_commands(4)* and *rexec_commands(4)* manual pages.

As previously mentioned, there are certain programs you might want to exempt from load leveling. These files allow you to specify the names of those programs.

By default, the list is an exclusion list. That is, the files list the names of the programs **load_leveld** must not load level.

By changing the *migrate_inclusion_flag* and *rexec_inclusion_flag* in the *parameters* file, these files can be inclusion instead of exclusion lists. That is, **load_leveld** will only load level the commands listed in the files.

7.4.3 load_leveld flags

There are several command line options that can be used to override default values for configuration files. These are the most common:

- p** Tells **load_leveld** to read the specified alternate configuration file instead of the default parameters file
- c** Tells **load_leveld** to read the specified file containing a list of commands that are exempt (or included) from migration
- r** Tells **load_leveld** to read the specified file containing a list of commands that are exempt (or included) from the invocation by way of **rexec**
- n** Disables migration; **load_leveld** will not attempt to reduce load by sending any processes the SIGMIGRATE signal
- N** Disables remote execution; **load_leveld** will not attempt to reduce load by performing an **rexec(2)** for processes using **exec(2)**
- v #** Specifies the level of logging information, where # is a number from 1 to 3

For a complete list of flags and options, see the **load_leveld(1M)** manual page.

7.5 Starting and stopping the load leveling daemons

There are two ways to start and stop the load leveling daemons:

- From a command-line
- Using the **keepalive** subsystem

Each is discussed below.

7.5.1 Starting load leveling from the command line

Using a command line for a shell requires **load_leveld** to have one additional process. This process starts all other **load_leveld** processes, then, like **init**(1), monitors them. If any die, the first process restarts the necessary replacements.

To start load leveling using this method, enter:

```
load_leveld [-b] [-c file] [-D] [-f] [-h] [-k] [-K directory] [-n]
[-N] [-p file] [-r file] [-R] [-v 1 | 2 | 3]
```

See the **load_leveld**(1M) manual page for further information.

7.5.2 Starting load leveling using the keepalive subsystem

The **keepalive** subsystem can be used to run the **load_leveld** processes. NSC comes with a default configuration that tells **keepalive** to start and monitor the **load_leveld** daemons.

To use the **keepalive** method, register **load_leveld** with **keepalive**. Enter:

```
/sbin/spawndaemon -o -r ka_load_leveld
```

WARNING! The **load_leveld**(1) manual page has similar instructions. However, it indicates that additional text is to be supplied on the command line. That manual page is incorrect. Use the command above instead.

7.5.3 Restarting the load leveling daemons at boot time

You can only start load leveling from the command line or keepalive subsystem while the cluster is running. If the cluster is rebooted for any reason, load leveling will not restart automatically.

To force load leveling to start every time the cluster boots, create shell scripts in the appropriate `/etc/rc*.d` directory. The contents of the script depends on how you are starting load leveling:

- If you are using the command line, the script must contain the commands you would enter on the command line as described in Section 7.5.1, “Starting load leveling from the command line,” on page 7-18.
- If you are using the **keepalive** subsystem, the script must contain the commands you would enter to register the daemons as described in Section 7.5.2, “Starting load leveling using the keepalive subsystem,” on page 7-18.

Depending on the run-level at which you start load leveling, you may want to create a shutdown script as well as a startup script.

7.5.4 Stopping load leveling started from the command line

To stop load leveling if it was started from the command line:

- Step 1. Find the PID of the **load_leveld** process with a parent process ID (PPID) of 1. To do this, enter:

```
ps -ef | grep load_leveld
```

The process you are looking for has a 1 in the third column.

- Step 2. Use **kill(1)** to send SIGTERM (15) to the process using the PID obtained in Step 1. This tells the **load_leveld** process to stop its children then exit.

7.5.5 Stopping load leveling started from keepalive

To stop load leveling if it was started from the **keepalive** subsystem, enter:

```
/sbin/spawndaemon -d -k /usr/sbin/load_leveld
```

This tells the **keepalive** subsystem to run the shutdown script associated with the **load_leveld** daemon and then unregister it.

7.6 Overriding load leveling

As previously mentioned, load leveling relies on the **rexec(2)** and **SIGMIGRATE** facilities of the NSC kernel. Further, it has a list of programs that are (by default) exempt from load leveling, including **onnode(1)** and **fast(1)**.

One way to start a program that is not subject to load leveling is to start it with the **onnode** or **fast** commands.

Further, if you do not want the program to be migrated, then use the **-p** flag to **onnode** or, make sure that the process explicitly ignores the **SIGMIGRATE** signal. Programs for which you do not have the source, create a shell script wrapper to ignore the **SIGMIGRATE** signal. For example, in a born shell script, or at the command line, enter:

```
trap '' 35
```

Because the action parameter is null in this command line, **SIGMIGRATE** (signal number 35) is ignored in all programs subsequently launched from that shell.

These are examples of the types of applications that are not well-suited to load leveling:

- Multiprocess applications that make heavy use of IPC. Performance could be substantially impacted because kernel objects are created on the origin node and do not migrate.
- Monolithic (single-process) applications. Subjecting large applications to load leveling could cause the program to *ping-pong* between nodes.

7.7 Load Leveling Domains

A load-leveling domain is a group of nodes in a cluster that use the same set of load-leveling parameters. If you want to have different sets of nodes using different parameters, you must set up separate load leveling domains.

For example, if you want one set of nodes to have a very close load level, and are willing to migrate and rexec more often than on another set of nodes, then you just set up two load leveling domains. You might want to do this when you have two sets of nodes running different types of applications.

To configure multiple domains:

1. Create a separate *parameters* file for each domain; for example, *params_domain1* and *params_domain2*.
2. Modify each *parameters* file and change the *nodes_to_use* field to specify the nodes in that domain.
3. Create separate *migrate_commands* files for each domain, if needed. For example, *migr_domain1* and *migrs_domain2*
4. Create separate *rexec_commands* files for each domain, if needed. For example, *rexec_domain1* and *rexec_domain2*
5. Start **load_leveld** once for each domain.

These example command lines show starting **load_leveld** is started for two domains:

```
load_leveld -c migr_domain1 -p parms_domain1 -r rexec_domain1
load_leveld -c migr_domain1 -p parms_domain2 -r rexec_domain2
```

To start load leveling for two separate domains from the keepalive subsystem:

1. Create two separate **keepalive** configuration files, for example, *ka_load_leveld1* and *ka_load_leveld2*.
2. Modify each file to use the **-p** flag followed by the name of the associated *parameters* file.
3. Register each of the configurations with the **keepalive** daemon using the **spawndaemon** command. For example:

```
/sbin/spawndaemon -o -r ka_load_leveld1  
/sbin/spawndaemon -o -r ka_load_leveld2
```

An example of configuring multiple domains is provided in Section 7.9.2, “Optional lab: Setting up multiple load leveling domains,” on page 7-27.

7.8 Lesson Summary

This lesson has discussed how automatic load leveling works and how to administer and defeat it. Specifically, this lesson discussed:

- Load leveling goals
- Load leveling concepts
- The mechanics of load leveling
- Starting and stopping load leveling
- Configuring load leveling
- Overriding load leveling
- Load leveling domains

Review questions

1. Is load leveling started by default when the cluster boots?
2. List the methods that can be used to start load leveling.
3. Which computer resources are considered when calculating the load level on a node?
4. How can you prevent an application from being migrated by **load_levelld** if you don't have the source code for the program?
5. How can you observe the results of load leveling in a cluster?

7.9 Lab: Using Load-Leveling

This lab is written in two parts. If you have a cluster with four or more nodes, do the second lab instead of the first.

7.9.1 Configuring load leveling

Follow the procedures outlined in this lesson to customize and then start load leveling for your cluster. Specifically, the load leveling daemons should:

- Exchange load information with all nodes at every exchange
- Exchange load information every 2 seconds
- Consider migrating processes every 2 seconds
- Consider a node overloaded when it has a load that is .3 greater than the average
- Consider a node underloaded when it has a load that is .3 less than the average
- Consider migrating processes with only 1 second of CPU time
- Be started from the command line
- Be stopped when the lab is finished

After configuring and starting the load leveling, place the cluster under node using the **spinner** program in */labs/lab9*. Start three instances of the **spinner** program on each node in the cluster. Start them all from the same node.

Start the **xload(1)** program on each node to verify that the load is being distributed among the nodes.

Procedure

- Step 1. Log into the cluster as **root** and run a graphical desktop.
- Step 2. Create a symbolic link between `/stand/1/unix` and `/stand/unix`. This is required to run **xload** in an NSC environment. Enter:

```
ln -s /stand/1/unix /stand/unix
```

- Step 3. Change to the `/etc/cluster/load_level` directory.
- Step 4. Make a copy of the `parameters` file to `parameters.orig` . Enter:

```
cp parameters parameters.orig
```

- Step 5. Edit the `parameters` file and set these values (if not already set):

```
send_timeout          1
redispatch_timeout    2
number_vectorelements number of nodes in your cluster
minimum_underload     .3
minimum_overload      .3
minimum_cputime        1.0
```

- Step 6. Start the load leveling daemons. Enter:

```
load_leveld
```

- Step 7. Allow foreign IP addresses to write to your display. Enter:

```
xhost +
```

- Step 8. Start and pin an **xload** program on each node in the cluster .Enter:

```
onnode -p N xload -bg green -fg red \  
-label nodeN -display $DISPLAY &
```

where *N* a node number. Enter this command once for each node, changing *N* each time to a different node number. Be sure to change it both after the **-p** flag and after the **-label** flag.

- Step 9. On node 1, start the **spinner** program 3 times for each node in the cluster. Enter (the appropriate number of times):
- /labs/lab9/spinner &**
- Step 10. Observe the load graphs displayed by **xload** for at least a minute to see that the loads eventually become approximately equal.
- Step 11. Stop the load leveling daemons by following the procedure in Section 7.5.4, “Stopping load leveling started from the command line,” on page 7-19.

7.9.2 Optional lab: Setting up multiple load leveling domains

This lab demonstrates the use of load-leveling domains. It requires a cluster with at least four nodes.

Divide your cluster into two sets of nodes. If there is an odd number of nodes, the odd one will not be load leveled.

Configure two load balancing domains, one for each set of nodes. Start load leveling from the command line for each domain. The configuration for each domain should be the same as that outlined in Section 7.9.1, “Configuring load leveling,” on page 7-24.

Place the separate domains under different loads. Verify that the loads are balanced within each domain, but different between domains.

Procedure

- Step 1. Log into the cluster as **root** and run a graphical desktop.
- Step 2. Create a symbolic link between `/stand/1/unix` and `/stand/unix`. This is required to run **xload** in an NSC environment. Enter:

```
ln -s /stand/1/unix /stand/unix
```

- Step 3. Change to the `/etc/cluster/load_level` directory.
- Step 4. Make a copy of the `parameters` file for the first domain. Enter:

```
cp parameters params1
```

- Step 5. Edit the `parameters` file and set these values (if not already set):

```
send_timeout          1
redispatch_timeout    2
number_vectorelements number of nodes in your cluster
minimum_underload     .3
minimum_overload      .3
minimum_cputime        1.0
```

- Step 6. Make a copy of the `parameters` file for the second domain. Enter:

```
cp parameters params2
```

- Step 7. Start the load leveling daemons for each domain. Enter:

```
load_leveld -p /etc/cluster/load_level/params1
load_leveld -p /etc/cluster/load_level/params2
```

- Step 8. Allow foreign IP addresses to write to your display. Enter:

```
xhost +
```

Step 9. Start and pin an **xload** program on each node in the cluster .Enter:

```
onnode -p N xload -bg green -fg red \  
-label nodeN -display $DISPLAY &
```

where *N* a node number. Enter this command once for each node, changing *N* each time to a different node number. Be sure to change it both after the **-p** flag and after the **-label** flag.

Step 10. Start the **spinner** program 2 times for each node in the first domain. From one of the nodes in the first domain, enter (the appropriate number of times):

```
/labs/lab9/spinner &
```

Step 11. Start the **spinner** program 3 times for each node in the second domain. From one of the nodes in the second domain, enter (the appropriate number of times):

```
/labs/lab9/spinner &
```

Step 12. Observe the load graphs displayed by **xload** for up to a minute to see that the loads eventually become approximately equal within each domain. The two domains have different load levels.

Step 13. Stop the load leveling daemons by following the procedure in Section 7.5.4, “Stopping load leveling started from the command line,” on page 7-19.

Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

exec(2)	Invokes subprocesses
fork(2)	Creates a new process
keepalive(1)	Displays network adapter driver statistics
load_level(1)	Monitors node load levels and migrates processes from overloaded to underloaded nodes
onnode(1)	Executes a command on a specified node
rexec(2)	Invokes subprocesses on a specified node
rfork(2)	Creates a new process on a specified node
spawndaemon(1)	Provides an interface to the keepalive daemon
xload(1X)	Displays a graphical view of the system load average

Relevant files

<i>/etc/cluster/load_level/parameters</i>	Contains configuration parameters for load_level(1) . See <i>parameters(4)</i> .
<i>/etc/rc*.d</i>	Contains startup directories and scripts

7.10 Answers to Exercises, Review Questions, and Labs

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Exercise answers

“The parameters(4) file,” on page 7-15

There is no answer associated with this exercise.

Review Question Answers

This section contains the answers to the review questions on page 7-23.

1. Load leveling is *not* started by default.
2. Load leveling can be started:
 - From the command line, using the `load_leveld` command
 - From an startup script during boot
 - From the **keepalive** subsystem
3. Only CPU usage is weighed when measuring the load on a node.
4. To prevent a program from migrating, you can start an application from a shell script that traps and ignores **SIGMIGRATE**. You can also enter the name of the executable file in `/etc/cluster/load_level/migrate_commands` and/or `/etc/cluster/load_level/rexec_commands`.
5. You can observe the results of load leveling in a cluster by running **xload(1X)** on all nodes in the cluster.



8

Failover



Lesson 8

Failover

Introduction

NSC is superior to simple failover solutions for many reasons. One reason is that it has multiple nodes always running the operating system and therefore ready to run applications. NSC increases this superiority by providing a facility that monitors applications running in the cluster and restarts them if they fail whether the application itself fails, or because the node running the application fails. In NSC, this facility is called *failover*.

The NSC failover facility is highly-configurable. You can add and remove programs, and change the way failover works.

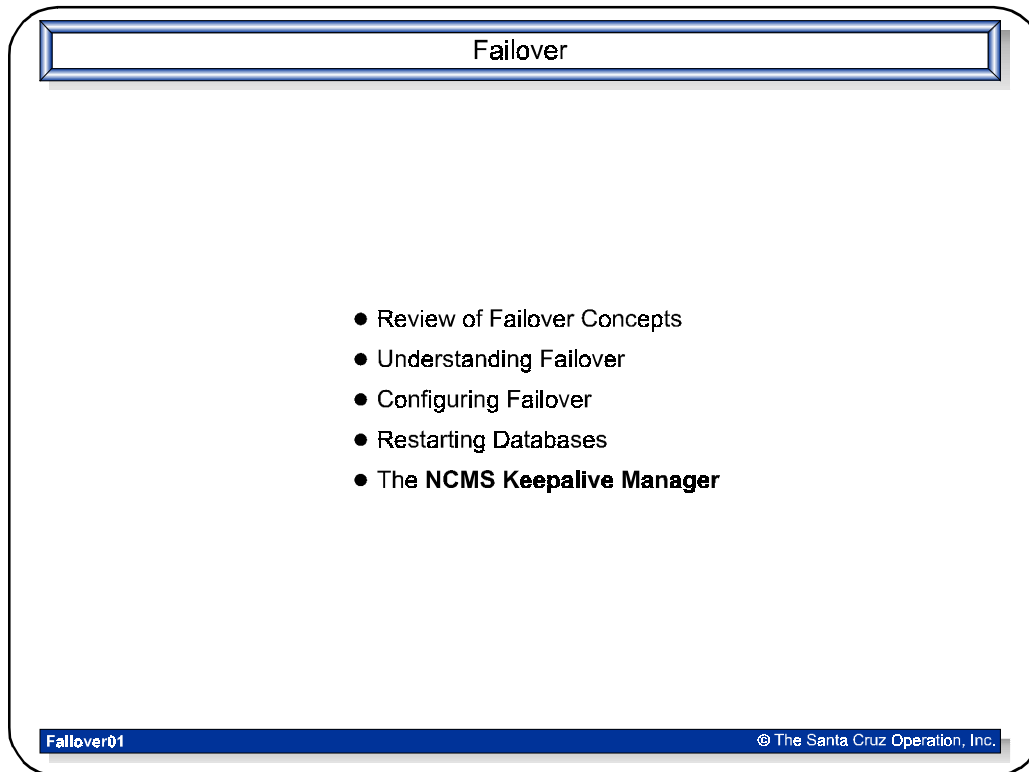
This lesson discusses how to configure failover in an NSC cluster.

Objectives

By the end of this lesson, you will be able to:

- Describe ways in which the NSC failover subsystem is superior to **init(1)** for monitoring and restarting processes
- Enable and disable failover
- Configure a single-process and multi-process application for failover
- Describe the daemon(s) used to implement failover
- Describe the command that submits applications for failover monitoring
- Write a restart script
- Describe the types of applications suited to monitoring by the failover facility

AV 8-1



Topics

In order to meet these objectives, this lesson discusses:

- Review of Failover Concepts — the 1+1 and N+1 concepts of failover presented in the *NonStop Cluster Concepts* course
- Understanding Failover — the daemons, commands, and files that are used to implement failover in NSC
- Configuring Failover — configuring failover to monitor single and multi-process applications
- Restarting Databases — using the **dbms_guard(1)** utility
- The **NCMS Keepalive Manager** — using this NSC-specific tool to monitor and restart daemons



8.1 Review of Failover Concepts

The **UnixWare® 7 NonStop Clusters® Concepts** course discussed different ways to design a cluster to be prepared to handle failure events. Included in the options were:

- 1+1 In this configuration, for every resource in the cluster, there is a spare resource to take its place in the event of a failure. For example, to configure a six-node cluster in 1+1 configuration, only three of the nodes are in service. The other three are held in reserve in the event one or all of the other three are lost.
- N+1 In this configuration, only one resource is held in reserve to replace one of possibly many resources that can fail. For example, to configure a six-node cluster in N+1 configuration, five of the nodes are in service. The other node is held in reserve in the event one of the other five nodes is lost. If more than one node is lost, cluster performance is degraded and the performance level specified for the expected workload might not be met.
- N+0 In this configuration, all nodes are pressed into service and no nodes are held in reserve. If a node is lost, cluster performance is degraded. However, if after a node is lost, the performance level still meets specifications, then this is a better use of your cluster investment than the N+1 or 1+1 configurations where equipment is not being put to work.

A recurring theme among these definitions is how many nodes are pressed into service and how many are held in reserve. All nodes in an NSC cluster are always booted, running at least system programs, and capable of running any number of user programs. The decision to press a node into service is defined by whether or not it is running user programs.

In addition to load leveling (see Lesson 7: “Load Leveling”), NSC provides the failover subsystem that allows you to tell NSC to monitor certain programs and restart them if they fail, or if the node they are running on is lost.

The failover subsystem is discussed in detail in the on-line NSC documentation. To view this documentation from **scohelp(1X)**, select **NonStop Clusters for SCO UnixWare 7 Documentation** ⇨ **System Administrator’s Guide** ⇨ **Monitoring and Restarting Processes**.



8.2 Understanding Failover

Failover is implemented in NSC by these four components:

keepalive (1M)	The daemon responsible for monitoring and restarting registered applications
<i>/etc/spawndaemon.d/ka_*</i>	Configuration files used by spawndaemon to register the failover behavior of an application with keepalive
<i>/etc/keepalive.d/*</i>	Startup, shutdown, and restart scripts used by keepalive to control applications
spawndaemon (1M)	The command used to direct the behavior of keepalive

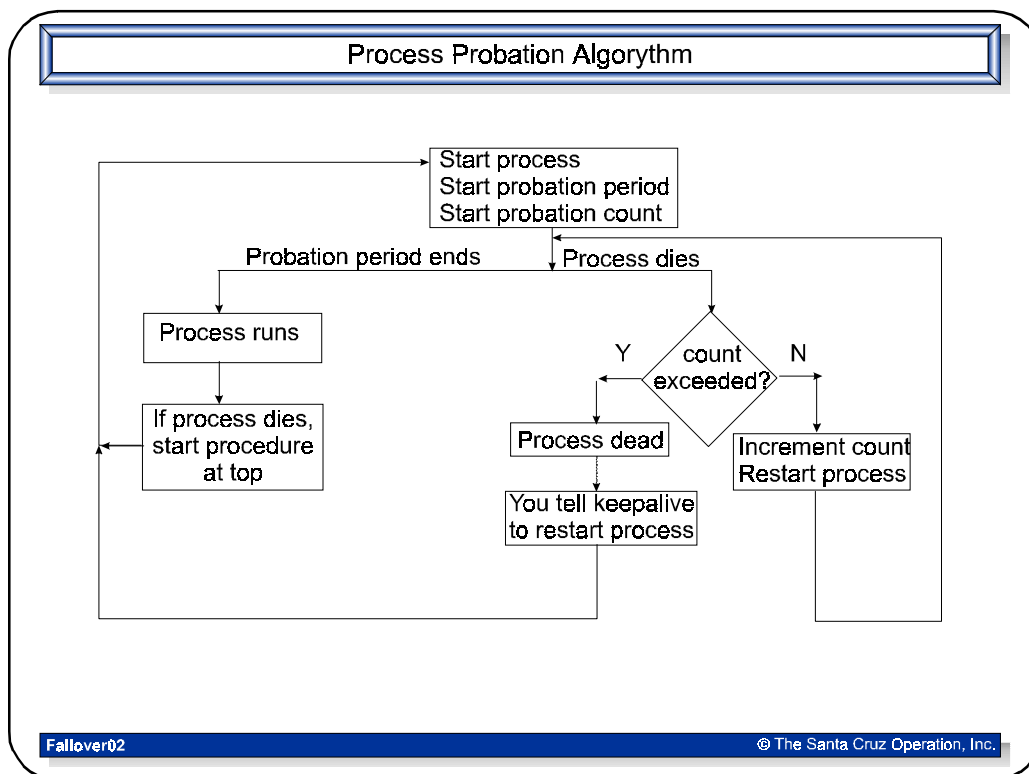
Once you understand the purpose of each component, you can configure applications to take advantage of the NSC failover feature. The labs in Section 8.7, “Lab: Configuring Failover,” on page 8-26 show the detail steps for configuring an application for failover.

8.2.1 The **keepalive**(1M) daemon

The **keepalive** daemon is similar to **init**(1M) in functionality when processing *respawn* entries in *inittab*(4). However, **keepalive** is more versatile than **init** in many ways:

- Whereas **init** can only monitor single processes, **keepalive** can be configured to monitor a group of processes. You specify an action that is initiated when a process within the group fails:
 - If a process in the group is critical, **keepalive** terminates all other processes in the group before restarting all processes in the group.
 - If a process is non-critical, the process is simply restarted.

AV 8-2



- You specify a probation period and a maximum number of retries within that period for a process. That is, when the process is first started, a timer is started. As illustrated in AV 8-2 on page 8-10, if the process dies within the probation period, it is restarted, up to the specified number of times:
 - If the process dies too many times within the specified probation period, the process is not restarted, unless you explicitly direct **keepalive** to restart it.
 - If the process does not die too many times within the specified probation period, the process continues to run. If it dies after the probation period, the probation period is re-started and the sequence continues.

With **init**, the probation period and number of times it will restart within the period are not configurable in any way.

- You can specify shell scripts to be run by **keepalive** in response to various events:

Startup	The script to start a process running; this script is mandatory
Shutdown	The script to be run when a process is shut down using the -k flag to the spawndaemon(1M) command (see below)
Process failure	The script to be run to recover from a process failure. Example items found in such scripts include removal of lock files.
Node failure	The script to be run to recover from the loss of the node running a process. This distinguishes the case between process failure (the process elected to exit) and node failure (the process was running fine but the node failed).
Process down	The script to be run when a process has failed too many times during its probation period

- You can specify how long to wait after a failure, before **keepalive** attempts a restart
- You can specify how long **keepalive** must wait when it is trying to shut down a process, before sending the process the **SIGKILL** signal.
- The standard output and error of each process is redirected by **keepalive** to a file under in the */var/adm/keepalive* directory.

Each application specifies its **keepalive** parameters in a separate configuration file in the */etc/spawndaemon.d* directory, described later in this section.

AV 8-3

The /etc/keepalive.d/load_level_d_restart File

```
#!/sbin/sh
DAEMON="/usr/sbin/load_level_d"
DAEMON_OPTS="-f -k -v"
if [ ! -x $DAEMON ]
then
    echo "load_level_d is disabled"
    exit 0
fi
exec $DAEMON $DAEMON_OPTS
```

Failover03

© The Santa Cruz Operation, Inc.

AV 8-4

The /etc/spawndaemon.d/ka_load_level_d File

```
:/usr/sbin/load_level_d:-f -k -v:60:root:sys::::load_level_d_restart:::::
```

Failover04

© The Santa Cruz Operation, Inc.

8.2.2 The **keepalive(1M)** scripts

The scripts used by **keepalive** to startup, shutdown, and restart an application are placed in the */etc/keepalive.d* directory.

These scripts have several common attributes:

- They all must reside in the */etc/keepalive.d* directory, with user and group of **adm**, and permissions of 755 (rwxr-xr-x)
- They are typically written in a shell scripting language, such as **ksh(1)**
- If the process does not daemonize itself, then it must be started as the last command in the script using the **exec** shell built-in command (see **sh(1)/ksh(1)/csh(1)** manual pages)

8.2.3 The **spawndaemon(1M)** configuration files

The files that control the failover behavior of an application are placed in the */etc/keepalive.d* directory. By convention, each file starts with the string *ka_*.

This file is read by **spawndaemon**, which uses the information to tell **keepalive** how to monitor and restart the application. The fields in this file are documented in the **spawndaemon(1)** manual page.

AV 8-3 on page 8-12 shows the *ka_load_level.d* **keepalive** configuration file. It includes these fields to control **load_level.d**:

- | | |
|--------------|---|
| process path | The full pathname to the process that will be started. This full pathname is used by the spawndaemon command when unregistering single-process applications. In the example, the pathname is <i>/usr/sbin/load_level.d</i> . |
| args | The arguments used to start this application. The arguments must be the in the same order as those used to start the application in the startup script. This is used to distinguish between multiple registrations of the same application. Each such registration must have a different set of arguments. In the example, the args are -f -k -v . |

For example, this is useful when using **keepalive** to monitor two **load_level.d** daemons, each with its own domain. To do this with a standard NSC system, after creating two **keepalive** configuration files, you must add a unique argument to each configuration file.

termwait	The number of seconds keepalive allows the shutdown script to run before keepalive issues a SIGKILL to try and stop the process. In the example, the value is 60 seconds.
uid	The user name or ID with which the process will run. In the example, the user is root .
gid	The group name or ID with which the process will run. In the example, the user is sys .
startup_script	The name of the script used to start the daemon. In the example, the script is inappropriately named <i>load_level0_restart</i> .

Only the process path, args, and startup_script fields are mandatory. See the **spawndaemon**(1) manual page for information on the fields not present in this example.

8.2.4 The `spawndaemon(1M)` command

The `spawndaemon(1M)` command is used to direct **keepalive** behavior:

Register	Tells keepalive to startup, and monitor the specified process. This is done with the -r or -R flag to spawndaemon .
Unregister	Tells keepalive to stop monitoring a specified process. This is done with the -d flag to spawndaemon .
Start	Tells keepalive to restart a process, even if it failed during its last probation period. This is done with the -x flag to spawndaemon .
Stop	Tells keepalive to shut down a process, using the shutdown script if one exists. This is done with the -k flag to spawndaemon .
Migrate	If you use the -c reject=# flag to spawndaemon , where # is a number from 0 to 255, and a process exists with an exit code equal to # , then keepalive will restart the process on a different node in the cluster, as specified by the -Z flag. This feature is called <i>node rejection</i> . It allows programs to tell keepalive that it should not run on the node it was assigned. For example, a node might not have sufficient CPU or RAM resources.

In addition to the arguments listed above, there are many other options documented in the `spawndaemon(1)` manual page.

Example uses of **spawndaemon** show how processes and process groups are registered and unregistered.

To register the process enter:

```
/sbin/spawndaemon -r ka_filename
```

where *ka_filename* is the name of the configuration file in the `/etc/spawndaemon.d` directory. Notice that the configuration file path is relative to the `/etc/spawndaemon.d` directory.

To unregister a processes, enter:

```
/sbin/spawndaemon -k -d full_pathname
```

where *full_pathname* refers to the executable program.

8.3 Configuring Failover

NSC comes with failover configurations for most system daemons. You might also want to add configurations for your applications.

There are three basic steps to configuring an application for failover:

1. Create any necessary startup, shutdown, or restart scripts for the application and place them in the */etc/keepalive.d* directory.
 - Script file names should end with *_startup*, *_shutdown*, or *_restart*
 - Set **adm** as the user and group owner
 - Set permissions to 755 permissions.
2. Create a process configuration file in the */etc/spawndaemon.d* directory.
 - The file should begin with the string *ka_*
 - Set **adm** as the user and group owner
 - Set permissions to 644 permissions.
3. Register the application using **spawndaemon**. For example:

```
/sbin/spawndaemon -o -r ka_filename
```

An example with detailed steps is provided in Section 8.7.1, “Configuring a single-process application,” on page 8-27.

8.3.1 Configuring failover for a multi-process application

Some applications use multiple processes. You can configure a multi-process application for failover by creating **keepalive** groups. Groups are established using a different format configuration file (the *ka_* file) in the */etc/spawndaemon.d* directory. In addition to the group configuration file, each process also has its own configuration file in */etc/spawndaemon.d*.

The configuration file for a group of processes differs from a single-process configuration file in these ways:

- The first line in the file contains this string:

```
<keepalive_group>:groupname
```

where *groupname* is a name you choose for the process group.

- Each subsequent line in the group configuration file specifies the configuration for a process within the group. These fields are included, separated by colons:

member_file	The name of the configuration file for the process. This is not the full pathname. It is the <i>ka_</i> name relative to the <i>/etc/spawndaemon.d</i> directory.
wait_time	The amount of time, in seconds, keepalive waits after starting this process before starting the next process in the group. This defaults to 0.
critical	A 1 or a 0, indicating whether or not (respectively) this process is critical to the group. If a process is critical to the group, and it dies, then keepalive terminates all processes in the group and restart the entire group.

In addition to the process group configuration file, each subprocess must have a configuration file in */etc/spawndaemon.d* to control the startup, shutdown, and restart of processes within a group.

The format of a configuration file for a process within a group is similar to individual process configuration files, except that the process group is always the first entry (before the first colon) in the file. These are also kept in */etc/keepalive.d*.

Startup, shutdown, and restart scripts for process within a group follow the same conventions as scripts for individual processes and are also kept in */etc/keepalive.d*.

The procedure for configuring a new multi-process application to be monitored by **keepalive** includes many of the steps for configuring a single process application. They are:

1. Create any necessary startup, shutdown, or restart scripts for each process in the group. Place them in the */etc/keepalive.d* directory. Change ownership and permissions on the scripts, as necessary.
 - Script file names should end with *_startup*, *_shutdown*, or *_restart*
 - Set **adm** as the user and group owner
 - Set permissions to 755 permissions.
2. Choose a unique name for the group of processes.
3. Create individual **keepalive** configuration files in the */etc/spawndaemon.d* directory for each process in the group.
 - The file should begin with the string *ka_*
 - Set **adm** as the user and group owner
 - Set permissions to 644 permissions.
4. Create a group **keepalive** configuration file in the */etc/spawndaemon.d* directory. As with the individual configuration files, the filename should begin with the string *ka_* and have 644 permissions with user and group set to **adm**.
5. Register the group using **spawndaemon**. Enter:

```
/sbin/spawndaemon -r ka_group_filename
```

where *ka__group_filename* is the name of the configuration file created in Step 3.

An example with detailed steps is provided in Section 8.7.2, “Configuring a multi-process application,” on page 8-30.

8.4 Restarting Databases

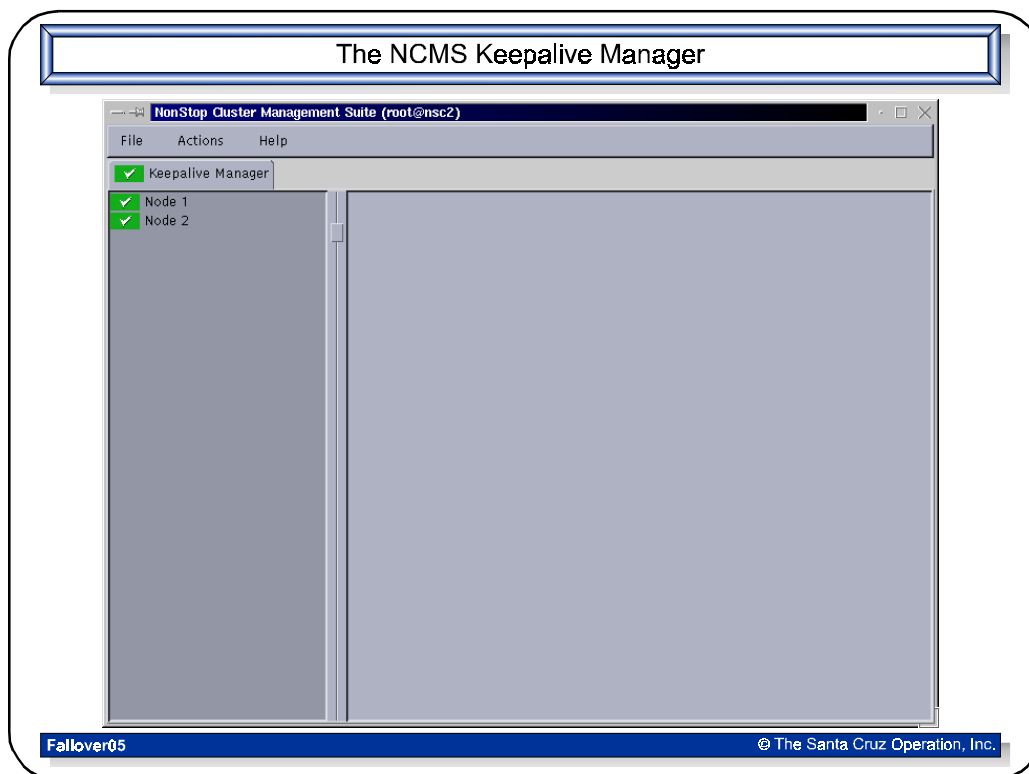
Server daemons suitable for configuring restart scripts include:

- Web servers, such as Apache or Netscape FastTrack
- Fax servers, such as VSI-Fax or Faximum
- Mail daemons, if using something other than **sendmail**(1M), which is already configured
- Databases, such as Oracle or Informix

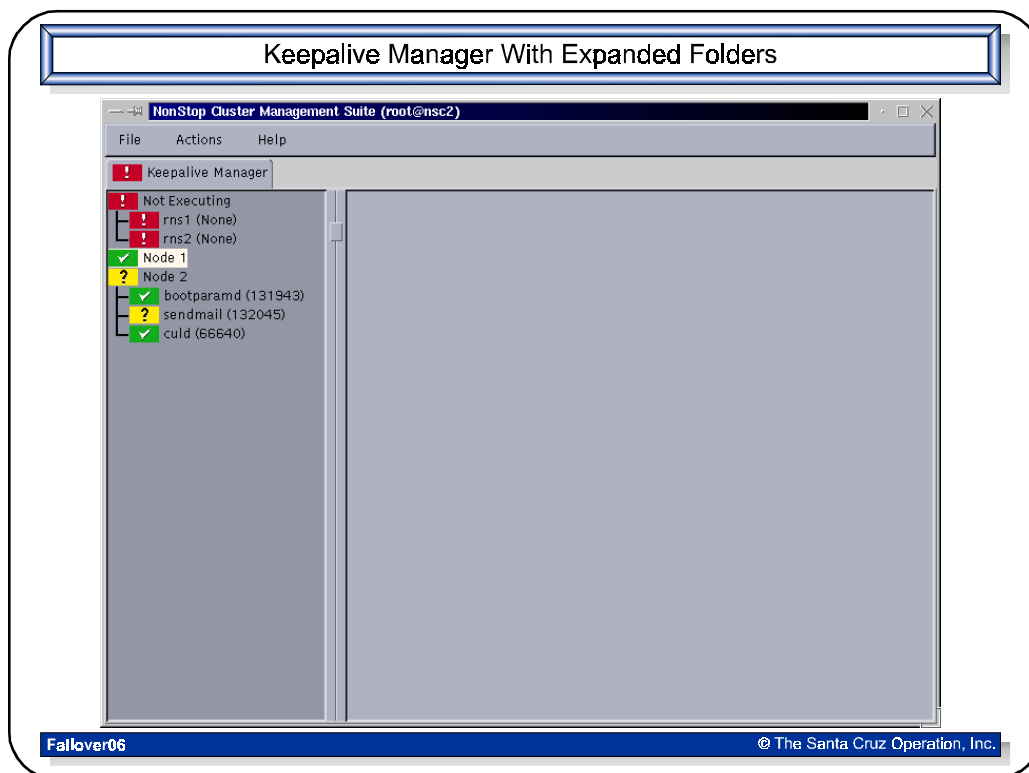
Because modern versions of Oracle 7 and Informix Online 5 have many server daemons with very specific startup ordering, NSC provides the **dbms_guard**(1) program to register these databases for monitoring.

See the manual page for further information.

AV 8-5



AV 8-6



8.5 The NCMS Keepalive Manager

Once you have configured the failover facility to monitor an application, you can use the **Keepalive Manager** to monitor and control the processes and daemons managed by the **keepalive** daemon. The **Keepalive Manager** is part of the NonStop Cluster Management Suite (NCMS)

To start the **Keepalive Manager**, select **scoadmin** ⇒ **Clustering** ⇒ **NCMS (Nonstop Cluster Management System) Keepalive Manager**. This displays a window similar to the one shown in AV 8-5 on page 8-20.

Note The **NCMS Keepalive Manager** is a Java application. It must be run in a supported environment, such as the UnixWare 7 X Windows environment.

AV 8-6 on page 8-20 shows the **Keepalive Manager** after expanding node 1 by double-clicking on the node 1 icon. Each of the additional items are processes running on the selected node being monitored by **keepalive**. Each entry has both the process name and ID.

Exercise: Starting the Keepalive Manager

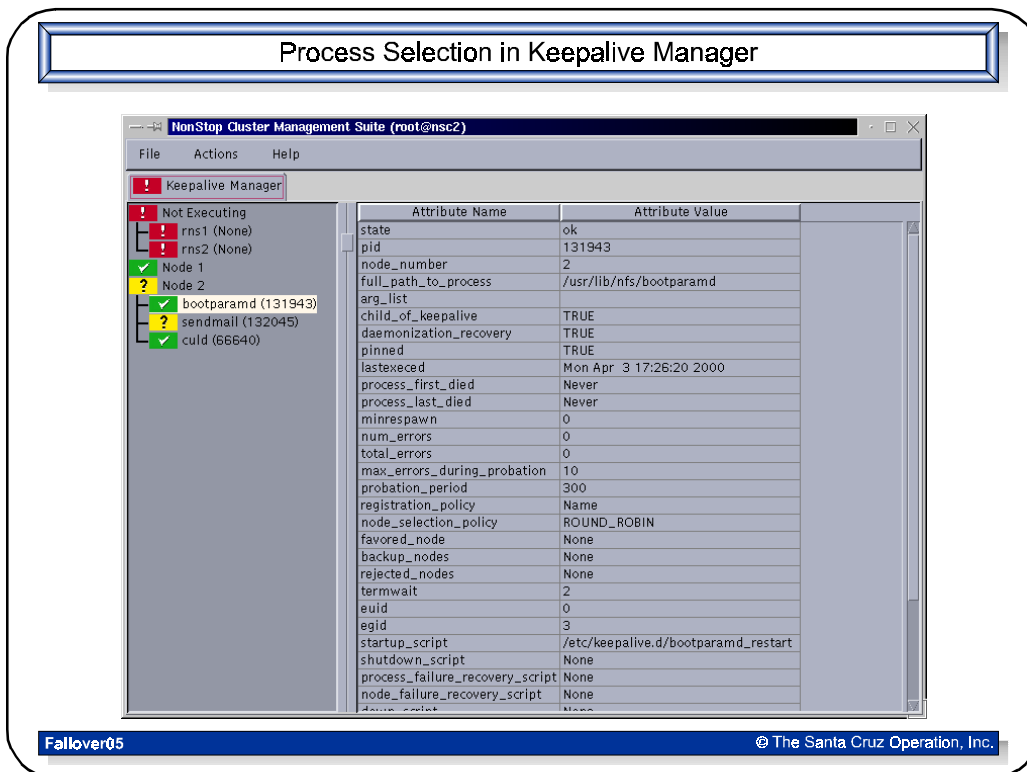
Start the **Keepalive Manager**.

Double-click on the label for node 1. You should see a window similar to that shown in AV 8-6 on page 8-20.

Notice the three colors of icons preceding the command names:

- | | |
|--------|---|
| Green | Means that the process has not failed since starting or counter reset |
| Yellow | Means the process has failed since starting or counter reset but is running |
| Red | Means the process is in the down state because of too many failures |

AV 8-7



To view the details of any particular process, single-click on the desired process. AV 8-7 on page 8-22 illustrates selection of the **bootparamd** program running on Node 2. Notice the scroll bar on the right side of the picture. This allows you to see all of the properties of the process, even though they do not fit within the viewing area of the window.

Exercise: Viewing properties of the sendmail process

From the **NCMS Keepalive Manager**:

- | | |
|---------|---|
| Step 1. | Expand the folder for Node 1. |
| Step 2. | Select the icon for the sendmail process. |
| Step 3. | Note the time the process was last exec'ed. |
| Step 4. | From a terminal emulation window, send signal 15 to the sendmail process using kill . |
| Step 5. | Select Actions ⇨ Update now from the main menu of the Keepalive Manager . |
| Step 6. | Note the new time that the process was last exec'ed, and that the icon has changed from green to yellow, both for the process and the node. |

These options are also available from the **Update Now** selection on the **Actions** menu:

- **Actions** ⇨ **Restart process** — Causes **keepalive** to stop and restart the process. The restart increments the processes failure count and, if necessary, restarts the probation period. If process was in the probation period and the allowed failure count is exceeded, the process is taken to the down state.
- **Actions** ⇨ **Clear probation error count** — Causes **keepalive** to reset the error count. This effectively restarts the probation period for the process.

- **Help** — Has submenus for various help topics on NCMS. The NCMS tools have their own help subsystem, currently separate from **scohelp**.

Exercise: Using NCMS Help

Step 1. Select **Help** ⇒ **On NCMS** to view the NCMS help subsystem.

Step 2. In the help window, from the navigation bar on the left, select **Keepalive Manager** ⇒ **Process Attribute Descriptions**.

According to the documentation, what is the difference between the “start” and “ok” states of a process?

- **File** ⇒ **Applications** — Causes another window to pop up showing the other NCMS managers that you can start. To start a specific application, select its check mark box and then select the **OK** button at the bottom of the window.
- **File** ⇒ **Close** — Causes the current NCMS application window to close. If other NCMS application windows are up, they remain up.
- **File** ⇒ **Exit** — Causes the entire NCMS application to exit. If other NCMS application windows are up, they exit as well.

8.6 Lesson Summary

This lesson discussed how NSC provides a failover mechanism for processes. Specifically, this lesson discussed:

- Review of failover concepts
- Failover mechanics
- Configuring failover
- Restarting databases
- The **NCMS Keepalive Manager**

Review questions

1. What tasks do you perform using the NCMS Keepalive Manager?
2. What is the purpose of the **keepalive** daemon?
3. What is the purpose of **spawndaemon**?
4. Why is a special program provided for configuring database failover?

8.7 Lab: Configuring Failover

This two part lab shows how to configure both single- and multiple-process applications for failover. You create the appropriate configuration files and shell scripts and then register the applications with **keepalive** using the **spawndaemon(1M)** command.

Then run the **NCMS Keepalive Manager** and view the effects of sending these signals to the PID(s) of the applications' processes.

1/SIGHUP	The process exits
2/SIGINT	The process displays the node on which it is running
35/SIGMIGRATE	The process alternates migrating between nodes 1 and 2

8.7.1 Configuring a single-process application

In this lab, you configure a web server for failover. The Netscape FastTrack web server is started by executing this command:

nsfast start

With no other arguments, **nsfast** starts **httpd** on port 80.

To perform this lab, create the appropriate files to start up **httpd**. Leave all optional parameters at their default settings.

After creating the configuration files and startup shell scripts, register the group for monitoring, and run the **NCMS Keepalive Manager**. Examine the entries for **httpd**.

Once the process is running, send the following signals to **httpd** observe the results:

- | | |
|---------------|--|
| 1/SIGHUP | The process will exit |
| 2/SIGINT | The process should announce on what node it is still running |
| 35/SIGMIGRATE | The process alternate migrating between nodes 1 and 2 |

Procedure

Use this procedure to configure the http daemon to be monitored by **keepalive**:

- Step 1. Log in as **root**.
- Step 2. Change to the `/etc/keepalive.d` directory.
- Step 3. Create a startup script in `/etc/keepalive.d`:
 - a. Copy the startup script from another file to create the startup script file with a name such as `httpd_startup`.
 - a. Set **adm** as the file owner and group using **chown**.
 - b. Set permissions to 755 (`rxr-xr-x`).
- Step 4. Create the configuration file for the process in the `/etc/spawndaemon.d` directory.
 - a. Create a new file with a name such as `ka_httpd`.
 - b. Set the user as the file owner and **adm** as group using **chown** and **chgrp**.
 - c. Set permissions to 644 (`rxr-xr-x`).
- Step 5. Register the process with the **keepalive** daemon, for non-daemonizing processes. Enter:

spawndaemon -r ka_httpd

- Step 6. Start the **NCMS Keepalive Manager** by starting **scoadmin(1M)** and selecting **Clustering** ⇒ **NCMS Keepalive Manager**.
- a. Double-click on each of the Node icons
 - b. Verify that **httpd** is present and have a green icon
 - c. Select (single-click) the sendmail process
 - d. Note the PID.
- Step 7. In a terminal emulation window, send signal 1 to the sendmail PID.
- Step 8. Refresh the **Keepalive Manager** by selecting **Actions** ⇒ **Update now**. The PID should change.
- Step 9. Send signals 2 and 35 to the PID. The PID should not change.
- Step 10. View the log files in `/var/adm/keepalive/httpd` to see the effects of sending the signals.

8.7.2 Configuring a multi-process application

In this lab, you configure a group of two process for failover. The files *rns1* and *rns2* are two programs that are part of an application called *rns*. These programs are located in the */labs/lab8* directory.

To perform this lab, create the appropriate files to start up the **rns** group of applications, keeping in mind that the **rns1** program is critical to the group. That is, if **rns1** dies, the entire group of processes must be restarted.

Specify all necessary parameters in the configuration files, and create only the necessary shell scripts. Leave all optional parameters at their default settings, except the criticality of **rns1** and **rns2**.

After creating the configuration files and startup shell scripts, register the group for monitoring, and run the **NCMS Keepalive Manager**. Examine the entries for **rns1** and **rns2** and verify that they are members of the **rns** group, and that **rns1** is critical and that **rns2** is not.

Once the processes are running, send the following signals to **rns1** and **rns2** and observe the results:

1/SIGHUP	The process will exit
2/SIGINT	The process should announce on what node it is still running
35/SIGMIGRATE	The process alternate migrating between nodes 1 and 2

Procedure

To configure failover for the group of processes comprising the **rns** application:

- Step 1. Log in as **root**.
- Step 2. Change to the */etc/keepalive.d* directory.
- Step 3. Create two startup scripts in */etc/keepalive.d*, one for each process:
 - a. Modify each script to run the appropriate program. Include the full path name.
 - a. Set **adm** as the file owner and group using **chown**.
 - b. Set permissions to 755 (rwxr-xr-x).
- Step 4. Create the group configuration file for **rns** in the */etc/spawnd daemon.d* directory.
 - a. Create a new file called *ka_rns*.
 - b. Modify the contents to specify the processes within the group.

HINT: It should contain three lines.
 - c. Set **adm** as the file owner and group using **chown**.
 - d. Set permissions to 644 (rwxr-xr-x).
- Step 5. Create two individual configuration files, one each for **rns1** and **rns2**.
 - a. Name the files *ka_rns1* and *ka_rns2*.
 - b. Modify the contents and specify the applicable criticality for each process. **rns1** is critical and **rns2** is not.

Remember to put **rns** as the text string before the first colon (so that they belong to the **rns** group)
 - c. Set **adm** as the file owner and group using **chown**.
 - d. Set permissions to 644 (rwxr-xr-x).

- Step 6. Register the process group with the **keepalive** daemon. Enter:
- ```
spawndaemon -r ka_rns
```
- Step 7. Start the **NCMS Keepalive Manager**. Run **scoadmin(1M)** and select **Clustering** ⇒ **NCMS Keepalive Manager**.
- Double-click on each of the Node icons.
  - Verify that **rns1** and **rns2** are present and have a green icon.
  - Select (single-click) each of the rns processes and verify that the **critical\_group\_process** value is TRUE for **rns1** and FALSE for **rns2**.

|                                                                                                                                             |
|---------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Note</b> If you see N/A for the <b>critical_group_process</b> value, you left off the group name in process's configuration file.</p> |
|---------------------------------------------------------------------------------------------------------------------------------------------|

- Step 8. Note the PID of both processes.
- Step 9. In a terminal emulation window, send signal 1 to the PID of **rns1**.
- Step 10. Refresh the **Keepalive Manager** by selecting **Actions** ⇒ **Update now**. The PID of each process should change.
- Step 11. In a terminal emulation window, send signal 1 to the PID of **rns2**. Repeat Step 6. However, only the PID of **rns2** should have changed.
- Step 12. Send signals 2 and 35 to each of the processes. Note that the PID of neither process changes.
- Step 13. View the log files in **/var/adm/keepalive/rns\*** to see the effects of sending the signals.

## 8.8 Relevant Commands and Files

---

The following commands and files were presented in this lesson.

### Relevant commands

---

|                          |                                                   |
|--------------------------|---------------------------------------------------|
| <b>dbms_guard(1)</b>     | Monitors and restarts databases                   |
| <b>keepalive(1)</b>      | Device Configuration Utility                      |
| <b>kill(1)</b>           | Sends signals to a processes                      |
| <b>load_level(1)</b>     | Migrates process between nodes based on CPU usage |
| <b>spawndaemon(1tcp)</b> | Display information about local and remote users  |

### Relevant files

---

|                           |                                                                              |
|---------------------------|------------------------------------------------------------------------------|
| <i>/var/adm/keepalive</i> | Contains <b>keepalive</b> log files                                          |
| <i>/etc/keepalive.d</i>   | Contains startup, shutdown, and restart scripts used by the keepalive daemon |

## 8.9 Answers to Exercises, Review Questions, and Labs

---

---

This section contains the answers to exercises, review questions, and labs presented in this lesson.

### Exercise answers

---

#### “Starting the Keepalive Manager,” on page 8-21

---

There is no answer associated with this exercise.

#### “Viewing properties of the sendmail process,” on page 8-23

---

There is no answer associated with this exercise.

#### “Using NCMS Help,” on page 8-24

---

The *start* state is for a process that keepalive is currently launching but is not yet up, whereas the *ok* state is for a process that is up and running.

## Review Question Answers

---

This section contains the answers to the review questions on page 8-25.

1. The **NCMS Keepalive Manager** allows you to monitor and restart applications.
2. The **keepalive** daemon is responsible for monitoring and restarting applications.
3. The **spawndaemon** program is used to register process and process groups with the **keepalive** daemon and to define the configuration parameters used for restarting the process/daemon.
4. The **dbms\_guard(1)** command is provided with NSC to configure failover for database applications. Because databases generally use many process which must be started in a specific sequence, configuring them for failover is complicated. The **dbms\_guard** program helps simplify failover configuration by providing sets of pre-configured startup, shutdown, and restart scripts.

## Lab answers

---

### Section 8.7.1, “Configuring a single-process application,” on page 8-27

---

These are the configuration files necessary for completing the lab in Section 8.7.1, “Configuring a single-process application,” on page 8-27.

**Table 1: /etc/spawndaemon.d/ka\_httpd**

|                                                               |
|---------------------------------------------------------------|
| <pre>:/usr/sbin/nsfast:::root:sys::::httpd_startup:::::</pre> |
|---------------------------------------------------------------|

**Table 2: /etc/keepalive.d/httpd\_startup**

|                                                   |
|---------------------------------------------------|
| <pre>#!/bin/sh  exec /usr/sbin/nsfast start</pre> |
|---------------------------------------------------|

**Section 8.7.2, “Configuring a multi-process application,” on page 8-30**

---

These are the configuration files necessary for completing the lab in Section 8.7.2, “Configuring a multi-process application,” on page 8-30.

**Table 3: /etc/spawndaemon.d/ka\_rnsgrp**

```
<keepalive_group>:rnsgrp
ka_rns1:1:1
ka_rns2:1:0
```

**Table 4: /etc/spawndaemon.d/ka\_rns1**

```
rnsgrp:/labs/lab8/rns1:::root:sys::::rns1_startup:::::
```

**Table 5: /etc/spawndaemon.d/ka\_rns2**

```
rnsgrp:/labs/lab8/rns2:::root:sys::::rns2_startup:::::
```

**Table 6: /etc/keepalive.d/rns1\_startup**

```
#!/bin/sh

exec /labs/lab8/rns1
```

**Table 7: /etc/keepalive.d/rns2\_startup**

```
#!/bin/sh

exec /labs/lab8/rns2
```





---

# 9

## *Configuring NSC Behavior*



**Lesson 9**

## **Configuring NSC Behavior**

### **Introduction**

---

In addition to the administration that can be performed via **scoadmin**(1M), NSC provides another tool for configuring NSC's behavior. This tool is called the NonStop Cluster Management Suite (NCMS).

Further, NSC also provides an enhanced, configurable, logging facility called the Event Processor Subsystem (EPS). Administrators can configure this subsystem to take specific actions based upon events. For example, EPS can call a pager if a node goes down.

This lesson provides an overview of the configuring NSC using the NCMS. It also provides an overview of EPS.

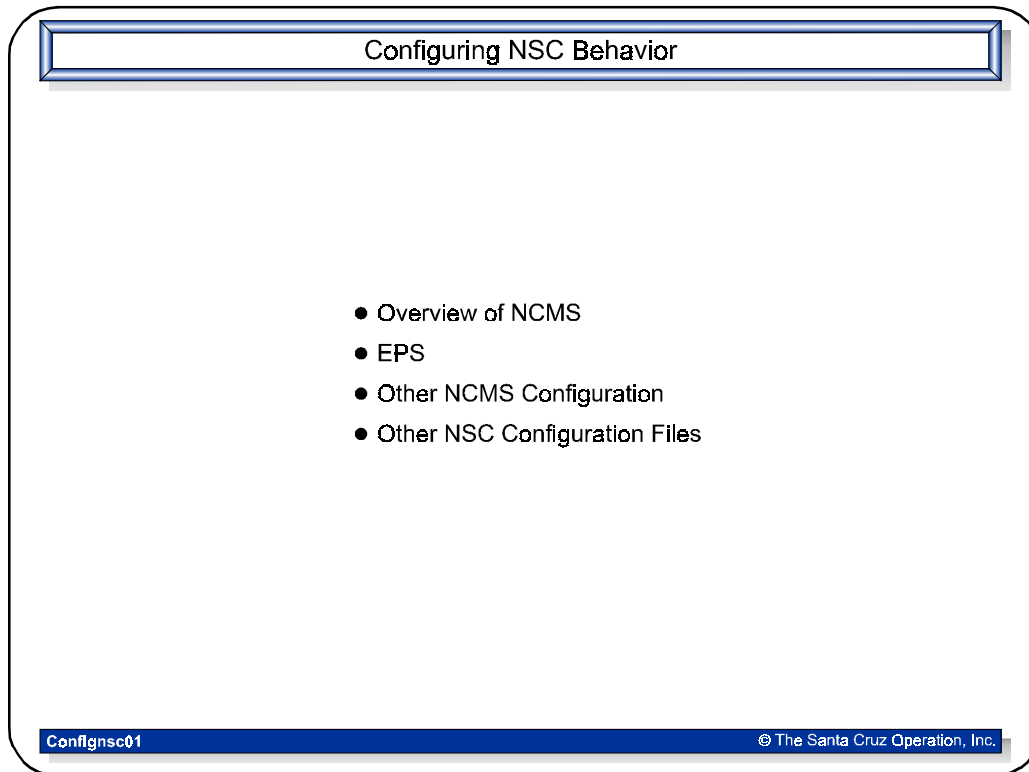
### **Objectives**

---

By the end of this lesson, you will be able to:

- Describe the three main functions of NCMS
- Describe the Event Processing Subsystem
- Enumerate the files used to configure EPS and describe their basic structure

AV 9-1



---

## Topics

---

In order to meet these objectives, this lesson discusses:

- Overview of NCMS — how the NonStop Clusters Management Suite is started and used to configure NSC
- NSC **Configuration Manager** — how to configure NCMS applications
- EPS — configuring EPS using the NCMS **Configuration Manager**



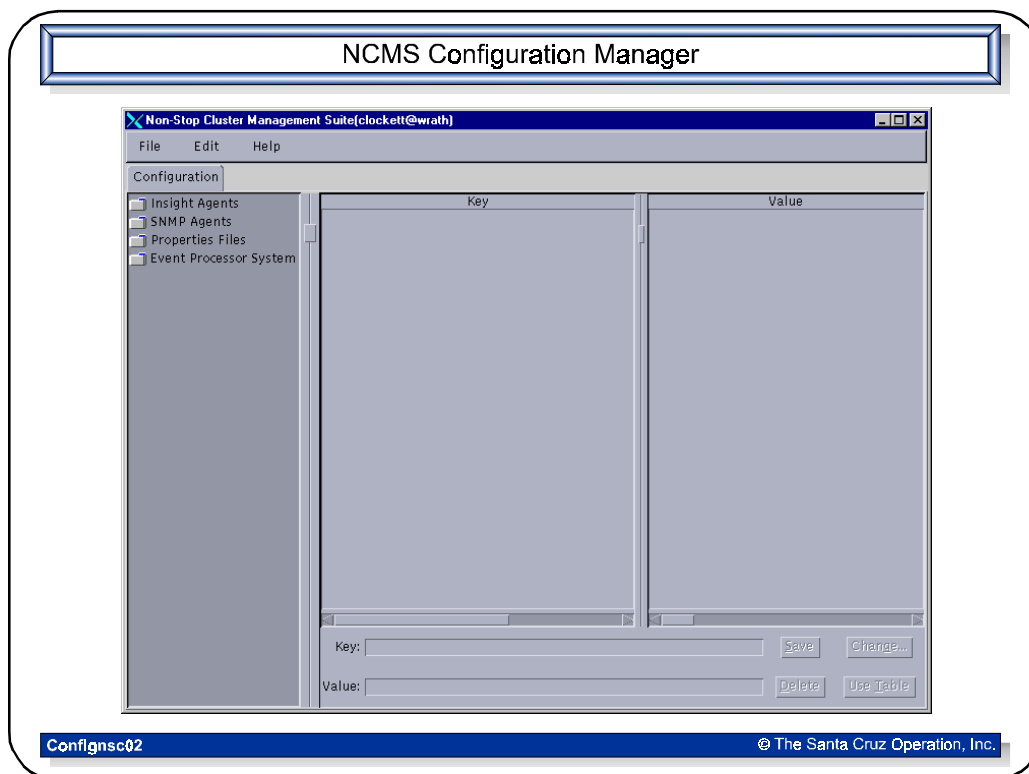
## 9.1 Overview of NCMS

---

The NonStop Cluster Management Suite (NCMS) provides a graphical user interface (GUI) to these administration applications:

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Configuration Manager</b> | <p>Serves as a specialized editor used to modify the contents of various configuration files.</p> <p>These types of files can be edited: Insight Manager agent initialization files, Simple Network Management Protocol (SNMP) agent initialization files, EPS configuration files, and Java property files used to configure operation of the NCMS applications.</p>                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Keepalive Manager</b>     | <p>Monitors and controls the processes and daemons registered with the <b>keepalive</b> daemon. All of the registered processes/daemons in the cluster are continually monitored for state changes. Icons indicate whether a process/daemon is down, has been restarted at least once, or is running and has not required a restart.</p> <p>A full set of state, status, and configuration attributes from the <b>keepalive</b> monitored process table can be displayed for each process/daemon. Control functions are available to restart a process/daemon and clear the number of process/daemon failures tracked by <b>keepalive</b>.</p> <p>See Lesson 8: “Failover,” for more information about configuring the <b>keepalive</b> daemon.</p> |
| <b>ServerNet SAN Manager</b> | <p>Monitors and controls the cluster ServerNet SAN on Compaq clusters. Monitoring functions include: state and status of the ServerNet SAN paths and fabrics, various I/O and error/recovery statistics associated with paths and SPAs, and SPA register contents. Control functions are available to designate the paths (X or Y) used between the nodes in the cluster and to reset SPA-related statistics. The ServerNet Manager also displays alert messages when events such as path failures or node removals occur.</p> <p>See the NSC online documentation for information on use the <b>ServerNet SAN Manager</b>.</p>                                                                                                                     |

AV 9-2





### 9.1.1 Running NCSM

---

NCMS is invoked with the **ncms**(1M) command from the operating system prompt:

**ncms**

When the **Select Applications** window appears, click on the applications that you want to run, then click on OK to start up the NCMS GUI.

Alternately, you can start NCMS from SCAdmin. Select **Clusters** ⇒ **NCMS**.

**Note** NCMS requires that your graphical environment supports Java.

#### 9.1.1.1 NCMS GUI with Help

---

The NCMS GUI has a menu bar across the top of the window and a set of tabs (one tab per application) running just below the menu bar. You select an application for use by clicking on its tab.

The **Help** menu is available for viewing overview and operational information on each of the applications in a separate help window.

When you select an item from the **Help** menu, the help window appears with two panes: a table of contents on the left and the help text for the item you selected from the **Help** menu on the right. You can move between help topics by clicking on the entries in the table of contents or taking links that appear in the help text. You can click on the plus (+) signs in the table of contents to expand a segment of the help topic tree. Clicking on a minus (-) sign collapses that segment of the tree.

## 9.1.2 Configuring NSC

---

The NCSM **Configuration Manager** is used to view and modify configuration files that control the appearance and behavior of the NCMS managers. They can also be edited manually.

The NCSM **Configuration Manager** operates on these files types:

- Compaq Insight Management (CIM) agent initialization files
- NSC SNMP agent initialization files
- NCMS properties files
- EPS configuration files

This simple example demonstrates how the NCSM **Configuration Manager** is used to modify configuration files. In this case, the NCSM property file is modified to change the location of the EPS configuration files.

Step 1. Start the NCSM **Configuration Manager**. Enter:

**ncms &**

Step 2. Select the **Configuration** tab, if it is not shown by default.

Step 3. Expand the **Properties Files** folder.

Step 4. Select **config\_prefs.properties**

A table showing property keys and values is displayed.

Step 5. Select the **config.eps.dir** property.

Step 6. Select **Edit** ⇨ **Change**.

Step 7. Change the value to */usr/local/lib/X11/app-defaults*.

Step 8. Click on **OK**.

Step 9. Select **File** ⇨ **Save**.

For more detailed information about configuring other subsystems, see the SCO UnixWare 7 NonStop Clusters Administration Guide..

**Exercise: Using the NCSM Configuration Manager**

---

Run the NCSM **Configuration Manager** and change the location of the EPS configuration files, as shown in the example above.



## 9.2 The Event Processor Subsystem

---

The Event Processor Subsystem (EPS) monitors event messages as they are added to the system log, searches for specific patterns in the messages, and takes a designated action when a pattern match is found.

The events to monitor are defined in EPS event filters. The actions available for each event type are:

- Sending a message (notification) to user-defined destination(s)
- Executing a UNIX command (or script)

Examples of user-defined destinations are the system console or an e-mail address.

By default, EPS is configured to send a notification message to the system console for:

- *ServerNet* system area network (SAN) path failure events
- Failure events produced by the **keepalive** system

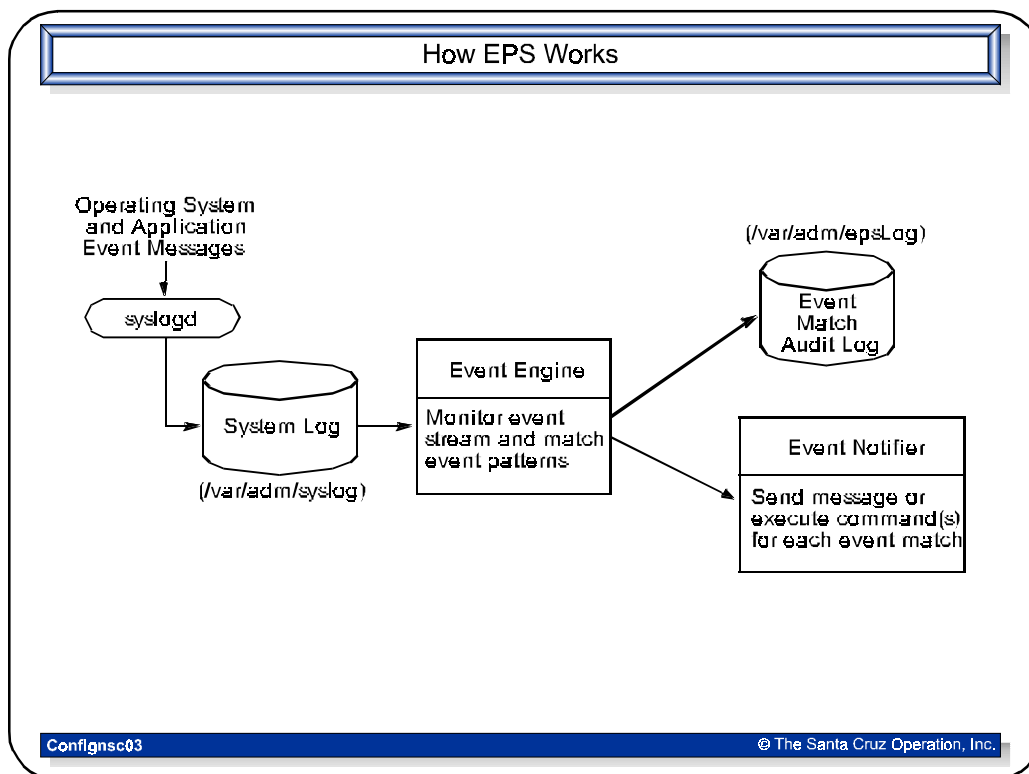
You can configure EPS to change these defaults and/or take actions on other event types.

The part of EPS responsible for sending notification messages, the Event Notifier, has an external command-line interface available to other applications that also need to send notification messages. This can also be useful for testing modifications to the EPS configuration. The command line is:

```
eNotify [-a appl] [-c crit] -D descr -d dest[,dest] -e event -h [-t time]
-m message
```

For a full description of the options and arguments, see the SCO UnixWare 7 NonStop Clusters Administration Guide.

AV 9-3



## 9.2.1 How EPS works

---

This is a synopsis of how the Event Processing System works:

1. System event messages generated by the NSC environment (operating system and applications) and logged by **syslogd**(1M) to the system log are monitored by the EPS event engine.

The EPS event engine is configured with event filters that define event strings to search for in the incoming event stream.

2. When an event message matching a pattern defined in a filter is encountered, the event engine:
  - Adds an entry to the audit log used to track events that are matched
  - Invokes the EPS Event Notifier to format and send a message to the destination(s) specified in the filter
3. EPS configuration files are used by the Event Notifier to format and send messages to:
  - UNIX devices, such as the system console */dev/console*
  - E-mail addresses—by way of **sendmail**(1M) and the Simple Mail Transfer Protocol (SMTP)
4. When a message is not delivered successfully (such as with SMTP) the first time, EPS configuration values are used by the Event Notifier to manage the retry attempts. If the retries fail, the Event Notifier is configured by default to log warning messages to the console.

## 9.2.2 EPS Configuration Files

---

EPS operation is configured using a collection of EPS configuration files. These files are similar to X (Server) resources files and are located in the `/usr/lib/X11/app-defaults` directory, by default.

|                        |                                                                                                                                                                                                                                                                                             |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Eps</i>             | Contains resources for general operation of EPS and the event notifier                                                                                                                                                                                                                      |
| <i>EpsFilters</i>      | Defines event filters for the event processor.                                                                                                                                                                                                                                              |
| <i>EpsEvents</i>       | Contains two basic types of resources: <ul style="list-style-type: none"><li>• Event definition parameters that determine the content and destination(s) of message notifications.</li><li>• Message retry parameters used when delivery of a message fails on the first attempt.</li></ul> |
| <i>EpsTemplates</i>    | Contains X-style resources that define the format of messages to be sent to various destinations.                                                                                                                                                                                           |
| <i>EpsSiteAdmin</i>    | Contains identification and contact information (such as name, phone, e-mail address, Internet Protocol (IP) address) specific to the site administrator(s) responsible for the system                                                                                                      |
| <i>EpsDestinations</i> | Contains the command lines used to invoke the message sending application or UNIX command/script for each destination                                                                                                                                                                       |

The files, and the resources they contain, are described in detail in the SCO UnixWare 7 NonStop Clusters Administration Guide.

These files can be manually edited if you are familiar with configuring X resources. Alternately, you can use the NCMS **Configuration Manager** to make changes to the EPS configuration files.



Whether you make changes manually, or using the NCMS **Configuration Manager**, you are recommended to make copies of the default EPS configuration files in a separate directory and use the copies to create custom site- and user-specific resources. This serves two purposes:

- You can keep site- and user-specific resources kept separate from the default resources to augment and override the defaults.
- Should future releases of EPS include configuration file changes, then your custom resources will be preserved.

A suggested location for customized resource files is `/usr/local/lib/X11/app-defaults`

### 9.2.3 Configuring EPS using the NCMS Configuration Manager

---

EPS is a comprehensive and flexible tool. As such, configuration can involve modifying numerous resource files. An example of a minor modification is provided to show how you can use the NCMS **Configuration Manager** to make changes to the EPS subsystem. For more complex examples, see the SCO UnixWare 7 NonStop Clusters Administration Guide.

To modify the EPS configuration to direct messages to the console on node 2:

Step 1. Create a new directory for the modified EPS resource files. Enter:

```
mkdir -p /usr/local/lib/X11/app-defaults
```

Step 2. Copy the original EPS resource files to the new directory. Enter:

```
cp /usr/lib/X11/app-defaults/Ep* \
/usr/local/lib/X11/app-defaults
```

**Note** The exercise “Using the NCSM Configuration Manager,” on page 9-11 changed the location of the EPS configuration files to `/usr/local/lib/X11/app-defaults`. If you did not perform this exercise, the NSMS **Configuration Manager** will not find the EPS configuration files and you will not be able to complete this exercise.

Step 3. Start the NCSM **Configuraton Manager**. Enter:

```
ncms &
```

Step 4. Select the **Configuration** tab, if it is not shown by default.

- Step 5. Expand the **Event Processing System** folder.
- Step 6. Select **EPS**.
- A table showing property keys and values is displayed. Select Table View, if it is not shown by default.
- Step 7. Click on **Use Table** in the lower right corner if the table view is not shown.
- Step 8. Select the **eps.problemDevice** property. By default, this property is set to */dev/console*.
- Step 9. Select **Edit** ⇨ **Change**.
- Step 10. Change the value to */var/adm/syslog*.
- Step 11. Click on **OK**.
- Step 12. Select **File** ⇨ **Update**.
- Step 13. Select **File** ⇨ **Exit**.

EPS messages are now sent to */var/adm/syslog*

### Exercise: Configuring EPS

---

- Step 1. Use the procedure above to change the device on which EPS messages are displayed so they are saved to a log file, such as the system log.
- Step 2. Generate an EPS message using the **eNotify** command to test the change. For example:
- ```
/opt/compaq/cms/bin/eNotify -a alpha \  
-s down
```
- Step 3. Check the log file to verify that the message was sent to the log file.

9.3 Lesson Summary

This lesson provided an overview of the NonStop Clusters Management Suite, with particular focus on the NCMS **Configuration Manager** and the Event Processing System.

Review questions

1. Why should you copy configuration files before making changes?
2. Which configuration file would you modify to change the format of messages sent by EPS?
3. Can you run NCMS in any graphical environment?

9.4 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

ncms(1M)	Runs the graphical NonStop Clusters Management Suite
eNotify(1M)	Sends a message about an event
eNotifyd(1M)	Monitors the system log file for EPS events
syslogd(1M)	Logs system messages

Relevant files

<i>/usr/lib/X11/app-defaults</i>	Is the location for EPS resource files
<i>/usr/lib/X11/app-defaults/Eps*</i>	Are the EPS resource configuration files
<i>/etc/syslogd</i>	Is the log file for system messages. See <i>syslogd(4)</i> .

9.5 Answers to Review Questions

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Review Question Answers

This section contains the answers to the review questions on page 9-19.

1. It is generally good practice to copy any configuration file before making modifications. If your modifications result in errors, you have the original files as a recovery mechanism.

In the case of EPS specifically, if you create a custom location for resource files, you prevent them from being overwritten when new resource files are written to the default location. This can happen when NSC updates are installed.

2. Modify the *EpsTemplates* file to change the format of EPS messages.
3. The NCMS GUI only runs in graphical environments that support Java, such as the UnixWare 7 Webtop or CDE desktop.



10

Troubleshooting Applications



Lesson 10

Troubleshooting Applications

Introduction

Most applications that run on non-clustered UnixWare 7 systems will run under NSC without any problems. However, applications that do not conform to that appropriate standards may have certain problems running in the NSC environment.

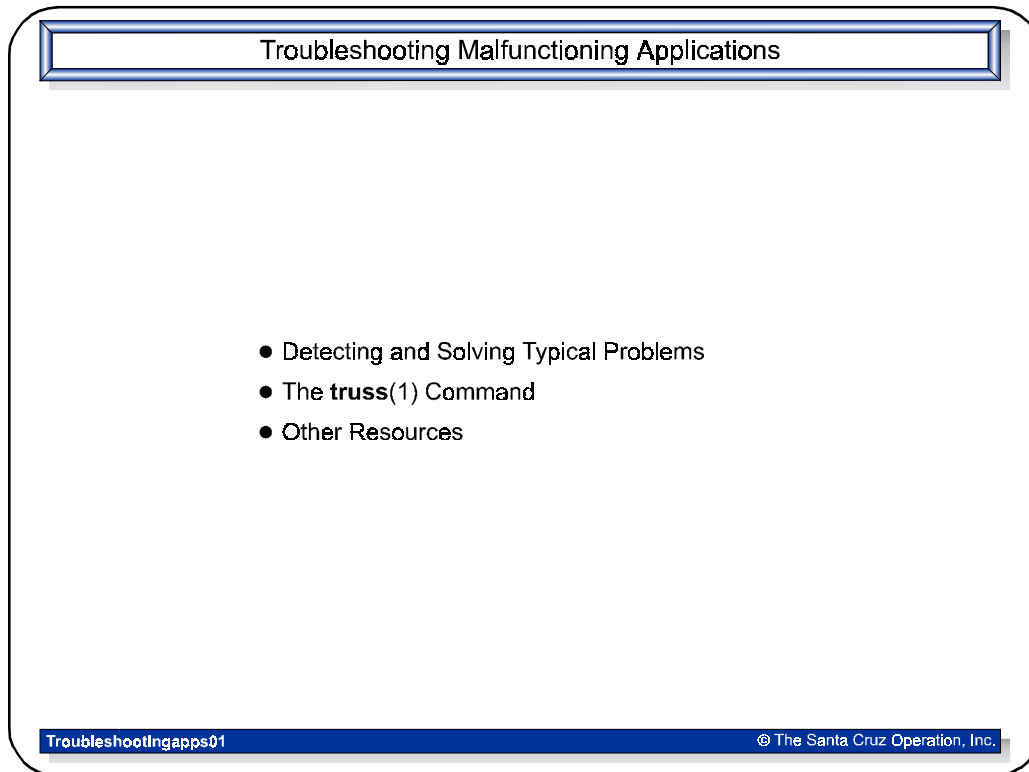
This lesson discusses typical problems applications encounter when they do not run properly under NSC and provides potential fixes or ways to work around them.

Objectives

By the end of this lesson, you will be able to:

- Describe the common symptoms exhibited by malfunctioning applications and the typical cause of these symptoms
- Explain how to use the **truss**(1) command to analyze a malfunctioning application
- Define workarounds or fixes necessary for some typical causes of malfunctioning applications

AV 10-1



Topics

In order to meet these objectives, this lesson discusses:

- Detecting and solving typical problems — the problems malfunctioning applications typically have in NSC and how they are solved
- The **truss**(1) command — how to use **truss** to find problems with malfunctioning applications
- Other resources — additional resources you can use to help diagnose and solve problems

10.1 Detecting and Solving Typical Problems

In the rare case that a program malfunctions when running on an NSC cluster, but works properly on a non-NSC UnixWare 7 system, the cause of the malfunction is probably known. This section discusses typical problems exhibited by malfunctioning programs and includes guidelines for detecting and solving them.

If the information provided in this lesson does not identify the problem, check <http://www.sco.com/ta>, the technical articles (TAs) at SCO's website.

10.1.1 Storing process IDs in a short integer

For some time, the POSIX standard has stated that the result of a **getpid(2)** system call, which gets a process ID (PID), is a 32-bit integer. However, for years before the POSIX standards, especially during development on 16-bit machines, programmers stored the result of the calls in a 16-bit (short) integer.

This code may still run well on many UNIX platforms today because said platforms never allow PIDs to rise above approximately 30,000, a number that fits within a 16-bit integer.

However, NSC actually requires programs to store PIDs in 32-bit numbers. Recall from Lesson 3: "Basic Process Administration," that this is because the high-order 16 bits of the 32-bit PID contain the number of the node on which the process was *started*.

Programs that store PIDs in a 16-, rather than 32-bit integer, will exhibit problems communicating with other processes, and even themselves, if such communication is based on the program learning their own or another processes PID. For example:

- Trying to send a signal to a process, the **kill(2)** system call returns -1 and sets `errno` to either:
 - 1 — EPERM, which means that the target process exists, but that the calling process does not have permission to send it a signal
 - 3 — ESRCH, which means there is no process by the ID it is using
- Trying to open a file where part of the file name includes the truncated PID

Problems of this type are identifiable with **truss(1)**. In truss output, look for calls to **getpid**. Observe what is returned. Then, look for calls like **open** or **kill** that do not use the appropriate PID.

Exercise: Observe PID Problems

In this exercise, you will observe the problem with storing PIDs in a short integer. NOTE: The first 3 steps may be performed for you by your instructor.

- Step 1. Log into a non-NSC system as **root**.
- Step 2. Change directory to */labs/lab11*.
- Step 3. Run the *malfunc* program. Notice that it is able to send itself a signal.
- Step 4. Log into a NSC system as **root**.
- Step 5. Change directory to */labs/lab11*.
- Step 6. Run the *malfunc* program. Notice that it is *not* able to send itself a signal.
- Step 7. Use **truss** on the *malfunc* program. What value is returned from the **getpid(2)** system call? What value is passed as the second parameter (process ID) to the **kill(2)** system call?

What is the likely cause of this program's malfunction?

The solution to this problem is to re-write and re-compile the program.

10.1.2 No */stand/unix*

Some programs open the file containing the kernel. On non-NSC systems, this file is stored in */stand/unix*. However, as discussed in Lesson 4: “Filesystem Administration,” in an NSC system, each node has its own stand filesystem mounted under */stand/#*, where # is the number of the node. For example, the kernel for node 1 is located in the */stand/1/unix* file.

It is generally obvious when programs cannot find */stand/unix* as they typically print a diagnostic message shortly before they terminate. If, on the other hand, the program simply terminates without any such warning, you can use **truss** to identify the problem. In the truss output, look for an **open(2)** system call where the first argument is */stand/unix* and the result is **ENOENT** (file does not exist).

A workaround for this problem is to create a symbolic link from */stand/unix* to one of the */stand/#/unix*, where # is the node of your choice.

There is only one problem with this workaround. If the node you selected goes down, your symbolic link does not point to a file that exists (as long as the node remains down). In that event, you can re-make the symbolic link to a node that is up, or wait for the down node to come back up.

Another workaround, which costs disk space, is to copy */stand/#/unix* to */stand/unix*. Since this file then sits in the root filesystem, which is protected in an NSC environment, the program will always be able to run, even if the current root node fails.

The one drawback to this solution is that, if you make a new kernel, you have to remember to re-copy the kernel from `/stand/#/unix` to `/stand/unix`. Alternatively, you can modify the program in the `/etc/conf/bin/idcpunix` file so that this copy happens automatically. This program is run every time a new kernel is built.

Exercise: Run xload(X1)

The **xload(X1)** program requires access to a kernel file to run. This exercise shows illustrates how to satisfy that requirement in an NSC environment.

- Step 1. Log into a NSC system in a graphical environment as **root**.
- Step 2. Run the **xload** program. Notice that it fails with a message indicating it “cannot get the namelist from `/stand/unix`.”
- Step 3. Make a symbolic link from `/stand/unix` to `/stand/1/unix` or `/stand/1/unix.1`.
- Step 4. Run the **xload** program again. It should succeed this time.



10.1.3 Non-clusterized device node names

Recall from Section 2.3, “File and Device Names,” on page 2-13 and Section 4.1, “Overview of Files, Filesystems, and Device nodes in NSC,” on page 4-7 that node names are generally clusterized for NSC, which means they are not the same as they are in a non-NSC environment.

Device nodes are often used by programs that utilize hardware, for example communications programs that use serial ports. Such programs have several options for specifying the device nodes they want to use:

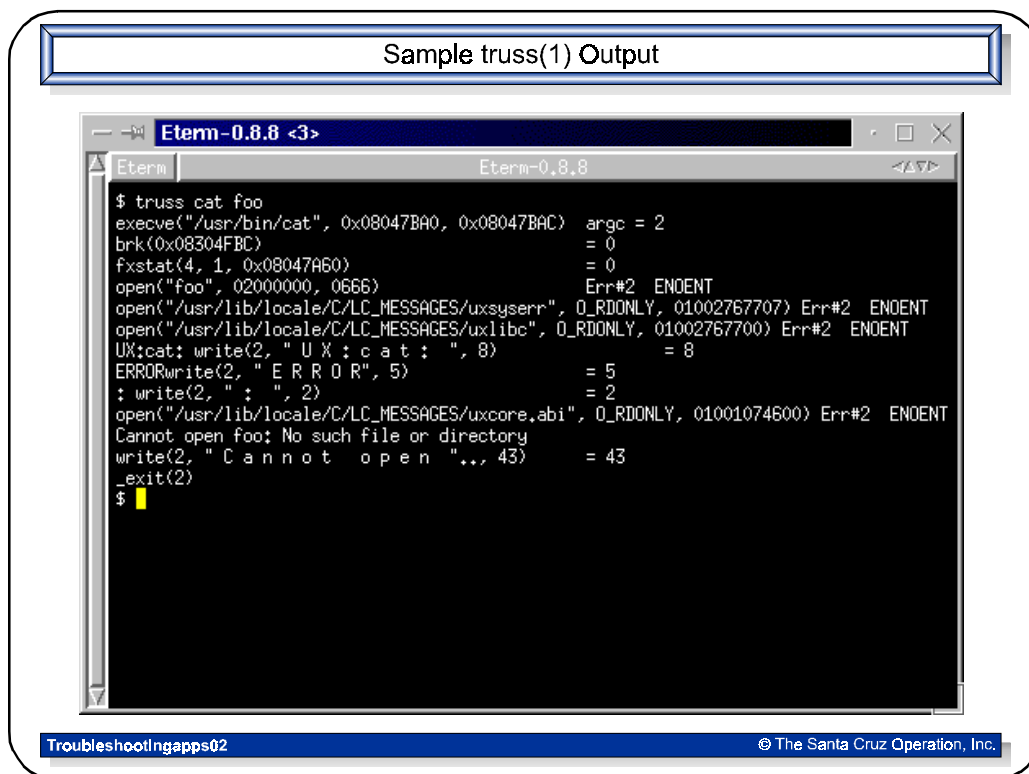
- They can hard-code the names of the device nodes into the program. This is poor programming practice and may render the program useless under NSC.
- They can allow command-line options to specify the names of the device nodes.
- They can utilize a configuration file which contains the names of the device nodes.

These scenarios call for the following actions:

- In the first case, the program will likely not run on NSC.
- In the second case, the user of the program must be sure to specify a clusterized device node name at program invocation. As an administrator, you must educate your user community about the clusterized device node names.
- In the third case, as an administrator, you must modify the configuration files for the programs to use clusterized device node names. If the configuration files are written automatically by a scoadmin manager, and the manager has been clusterized, the manager will use clusterized device node names.

Use the **truss** command to determine if a program is attempting to use a non-clusterized device node name. In the **truss** output, look for **open** or **access** system calls that return **ENOENT** and have a path name (the first argument) that starts with **/dev**.

AV 10-2



10.2 The *truss(1)* Command

UnixWare comes with a program called **truss(1)**, that allows you to trace the system call activities of any running program. In the event the process creates child processes (via the **fork(2)** system call), you control which process **truss** traces: the parent; the child; or both.

Use **truss** like **onnode** and **nohup**. That is, type **truss** followed by the command you want to trace. For example, to trace this command:

cat foo

enter the command:

truss cat foo

AV 10-2 on page 10-12 illustrates output of this command. The output contains:

- One line per system call made by the process, starting with the **execve(2)** of */usr/bin/cat*
- The parameters passed to each system call. By default, if the system call takes the address of a structure, the user-space virtual address (in hexadecimal) is displayed. If you would rather see the value of the members of the structure, enter this command line:

truss -v all cat foo

- The result of the system call. If the result was **-1**, indicating an error, **truss** also prints the value of **errno**, which indicates why the system call failed.

Note Each line is only printed after the system call has completed. Therefore **truss** can not be used to determine why a program is hung, if it is hung in a system call. Use **crash(1M)** in that case.

To determine the meaning of a particular error number for a given system call, read the manual page for that system call.

Note As a rule, you can ignore system calls that operate (or try to) on any object in the */usr/lib/locale* directory. These calls often fail because language localization is not installed.

In the example in AV 10-2 on page 10-12, the first system call to fail was **open(2)**, which was trying to open the *foo* file. The error number returned was 2, which is **ENOENT**. The manual page for **open(2)** says that **ENOENT** is returned when either:

- The **O_CREAT** flag is not set, and the object does not exist
- The **O_CREAT** flag is set but one or more of the components of the path name does not exist or is **NULL**.

O_CREAT is defined in the */usr/include/sys/fcntl.h* file as 0x100 (100, hexadecimal). It is a bit mask that, if desired, is bit-wise ored into the second parameter of the **open(2)** system call.

The illustration shows that the **O_CREAT** flag was not set. So, the reason for the failure of **open(2)**, in this case, was that the file *foo* does not exist.

10.3 Other Resources

There are a number of ways you can get help troubleshooting application compatibility problem. Use the web for fastest access to a wealth of information made available from SCO and your hardware vendor.

For accessing SCO NonStop Clustering information, start with URL:

<http://www.sco.com/clustering>

Also try searching through the SCO Support database of Technical Articles (TAs). The search form is available from URL:

<http://www.sco.com/ta/>

SCO also offers several application optimization services:

- The UnixWare NonStop Clusters Optimization Center is available through SCO Corporate Headquarters to help you write and test failover scripts. This is a no-cost service for independent software vendors. See URL:

<http://www.sco.com/products/clustering/optc>

- SCO Professional Services offers a fee-based service to customize your application to take full advantage of NSC features such as failover. See URL:

<http://www.sco.com/profservices/clustering/clustapp.html>

10.4 Lesson Summary

This lesson discussed the typical problems programs have in the NSC environment. Guidelines were provided for detecting and solving these common problems:

- The **truss**(1) command and how it can be used to detect certain problems
- Problems related to storing a PID in a 16-bit short integer
- Problems related to the absence of the */stand/unix* file
- Problems related to clusterized device node names

Pointers to additional troubleshooting information were also provided.

Review questions

1. What type of problems are easily detected using **truss**?
2. What is a good starting place for finding information to help you troubleshoot problems with applications running on NSC?

10.5 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

access(2)	Tests access to an object in a filesystem
crash(1M)	Analyzes dump files
getpid(2)	Gets the PID
kill(2)	Signals a process
open(2)	Opens a file
truss(1)	Traces programs

Relevant files

<i>/stand</i>	Is the directory that was the traditional mount point for the <i>stand</i> filesystem in non-NSC UNIX environments
<i>/stand/#/unix</i>	Are the directories (# is replaced with a node number) that are now the mount points for the <i>stand</i> filesystems of the nodes in an NSC system
<i>/stand/unix</i>	Is the traditional location of the UNIX kernel

10.6 Answers to Exercises, Review Questions, and Labs

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Exercise answers

“Observe PID Problems,” on page 10-7

The process ID returned by **getpid(2)** is dynamically assigned by the kernel. The second parameter to **kill(2)** is not the same as that returned by **getpid**. The reason is that the number returned by **getpid** was stored in a short (16-bit) integer, instead of a 32-bit integer.

Review Question Answers

This section contains the answers to the review questions on page 10-16.

1. The **truss** command is useful for detecting problems applications have accessing files. Two examples are programs that access device files using the non-cluster device node names, and programs that look for the UNIX kernel in */stand*.
2. See the SCO web site at URL <http://www.sco.com/ta/> for help troubleshooting application compatibility problems.

11

Tuning an NSC Kernel



Lesson 11

Tuning an NSC Kernel

Introduction

The *SCO UnixWare 7 Administration II* courses presented information on how to tune a kernel for optimal performance on a stand-alone host. In a cluster, there is a single kernel with a single set of tunable parameters that apply to the entire set of nodes.

Therefore, tuning a cluster kernel requires a deeper understanding of kernel parameters and a different methodology for gauging performance of each node and the cluster as a whole.

This lesson discusses how kernel tunable parameters work and the approach administrators use to tune a cluster kernel.

Objectives

By the end of this lesson, you will be able to:

- Describe autotune kernel tunables and explain how they differ from non-autotune parameters
- Override the value of an autotune kernel tunable with a fixed value
- Describe NSC-specific tunables
- Rebuilt and NSC kernel

AV 11-1

The slide is titled "Tuning an NSC Kernel" and contains a bulleted list of topics. The footer includes the identifier "Ktuning01" and the copyright notice "© The Santa Cruz Operation, Inc.".

Tuning an NSC Kernel

- Types of Kernel Tunables
- Autotune
- Performance Monitoring Tools
- NSC-Specific Kernel Tunables
- Building Kernels and **idtune**(1M) in NSC

Ktuning01 © The Santa Cruz Operation, Inc.

Topics

In order to meet these objectives, this lesson discusses:

- Types of kernel tunables — contrasts traditional static tunables and autotune tunables and discusses the methodology for tuning these different kinds of variables for a cluster.
- Autotune — configuring and overriding autotuned values
- Performance monitoring tools — monitoring system performance and detecting common problems
- NSC-specific kernel tunables — kernel tunable parameters specific to NSC kernels
- Building kernels and **idtune**(1M) in NSC — building kernels in the NSC environment

11.1 Types of Kernel Tunables

The UnixWare kernel is made up of a series of *modules*. Some of the modules are present at kernel boot. Others are loaded on demand. This has several advantages:

- The base kernel can be very small. This means that UnixWare can scale down to fit in a small footprint, like a cash register, as easily as it can scale up to handle a full-blown enterprise server load.
- The kernel can expand modularly and dynamically. In other words, new filesystems, scheduling classes, device drivers, and so on, can be added to the kernel without a full relink and reboot.

When writing a kernel modules, programmers often use many data variables. Sometimes, it is desirable to allow system administrators to set the value these variables. These variables are called *kernel parameters*, *kernel tunables*, or *kernel tunable parameters*.

Because the programmer may have to make certain rules about the value of a parameter, UnixWare sets boundaries on the variables in the form of a *minimum* and *maximum* values. They also have a *default* value for a parameter.

It is sometimes desirable to have the minimum, maximum, and default value of a tunable depend on the amount of memory that exists in the kernel at the time the system boots.

To allow for both constant and variable parameters, UnixWare offers two kinds of kernel tunables:

- *Traditional* kernel tunables where the value of the parameter has a default, minimum, and maximum value. The default value can be overridden using the **idtune(1M)** command.
- *Auto-tuning* parameters calculate the minimum and maximum values when the kernel is built based on the amount of memory present in the machine at that time. The default is determined at kernel boot time based on the amount of memory present at the time of boot.

11.2 Autotune

The value assigned to an auto-tuned kernel parameters is determined by a set of *rules*. These rules—equations, really—define a parameter’s value based on the amount of memory present, rather than assign a constant value. The rules make up a series of connected lines referred to as a curve.

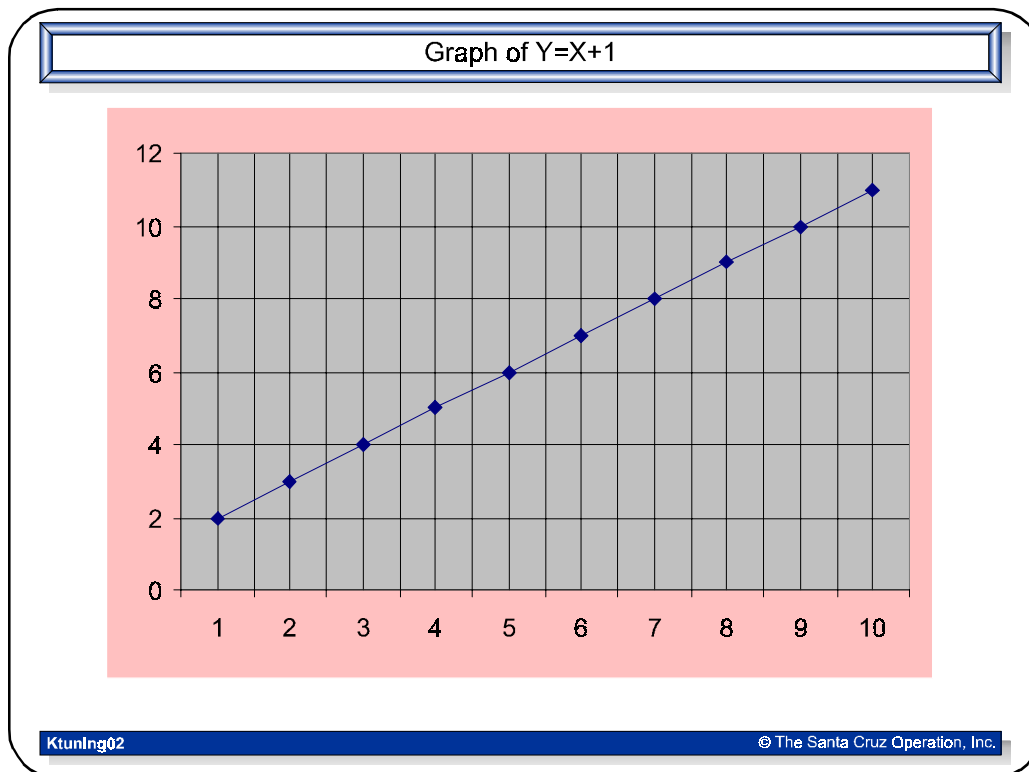
When a programmer decides to use one or more auto-tuning parameters in their module, they:

- Create a file in a specified format that contains the rules for determining the value of an auto-tuned parameter
- Call a routine in the kernel during module initialization to determine the value for a parameter. This routine makes its determination based upon the amount of memory present at the time the function is called.

Unfortunately, since a specific kernel function must be called to determine an auto-tuned parameter’s current value, you cannot, as an administrator, simply turn a traditional kernel parameter into an auto-tuned one.

The rest of this section discusses configuring auto-tuning parameters in detail.

AV 11-2



11.2.1 Functions, Interpolation, Extrapolation and Stepping

Before discussing how auto-tuning works, you must understand the terms *linear interpolation*, *extrapolation*, *function*, and *graph*.

11.2.1.1 Functions and graphs

The mathematical term *function* refers to a procedure always produces the same output given the same inputs. You often use *graphs* to visualize the effect of a function. For example, AV 11-2 on page 11-8 shows a graph of the function $y = x + 1$, where X ranges from 1 to 10.

That is, for a given value in the horizontal (X) direction, the corresponding value in the vertical (Y) direction is 1 greater. Functions whose graphs draw a line are *linear functions*.

Each square on the line in the slide represents a *point*—an intersection of the X and Y values. Points are often written as a parenthesized, comma-separated list in X and Y order. For example (6,7) would represent a point 6 units to the right and 7 units up.

11.2.1.2 Linear Interpolation

The mathematical process of finding a value along a line *between* two values is known as *linear interpolation*.

It is used when you:

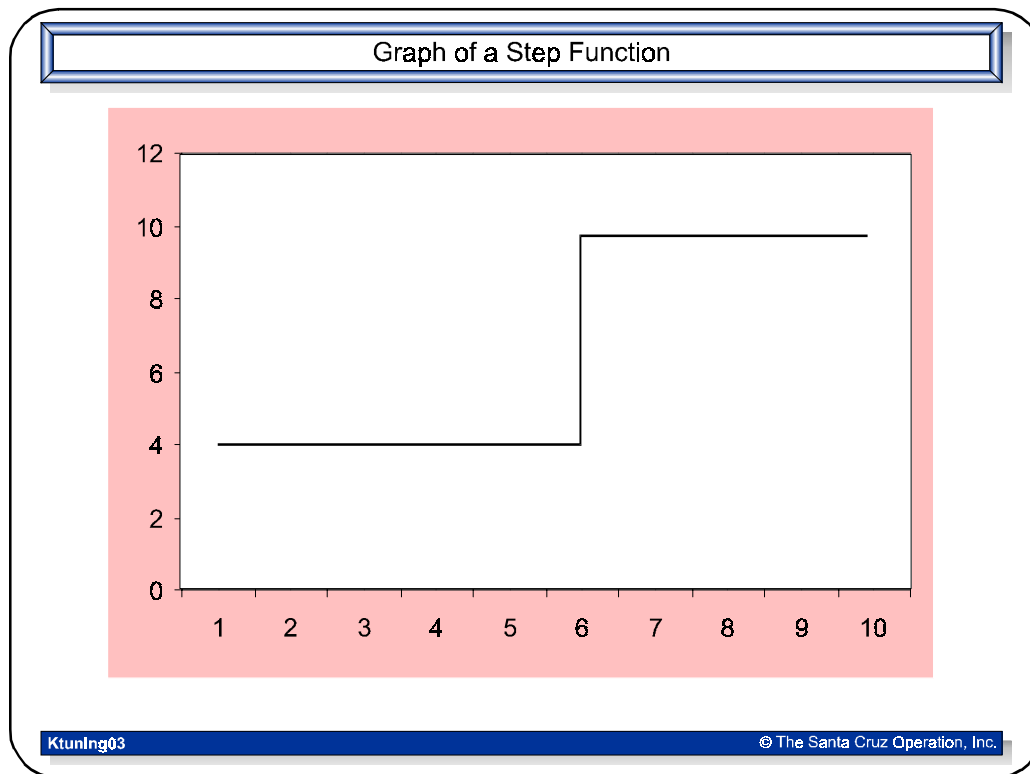
- Don't know the actual function, just that it is linear
- Are given two points
- Are asked for some value *between* the two points

As an example, consider how you would answer this question using the graph shown in AV 11-2 on page 11-8:

For a linear function with points at (1,2) and (10,11), what is the value of Y when X is 5?

The answer can be found using the graph. Find 5 on the horizontal (X) axis and follow the line to the Y axis. It intersects at 6, which is the solution to this equation.

AV 11-3



11.2.1.3 Linear Extrapolation

Whereas interpolation is the process of finding a point on a line *between* two points, *extrapolation* is the process of finding the value of a function at a point *outside* of two points by extending the line as far as necessary.

Again, referring to AV 11-2 on page 11-8, answer these questions:

- What is the value of Y when X is 11. The answer is 12.
- What is the value of Y when X is 100. The answer is 101.

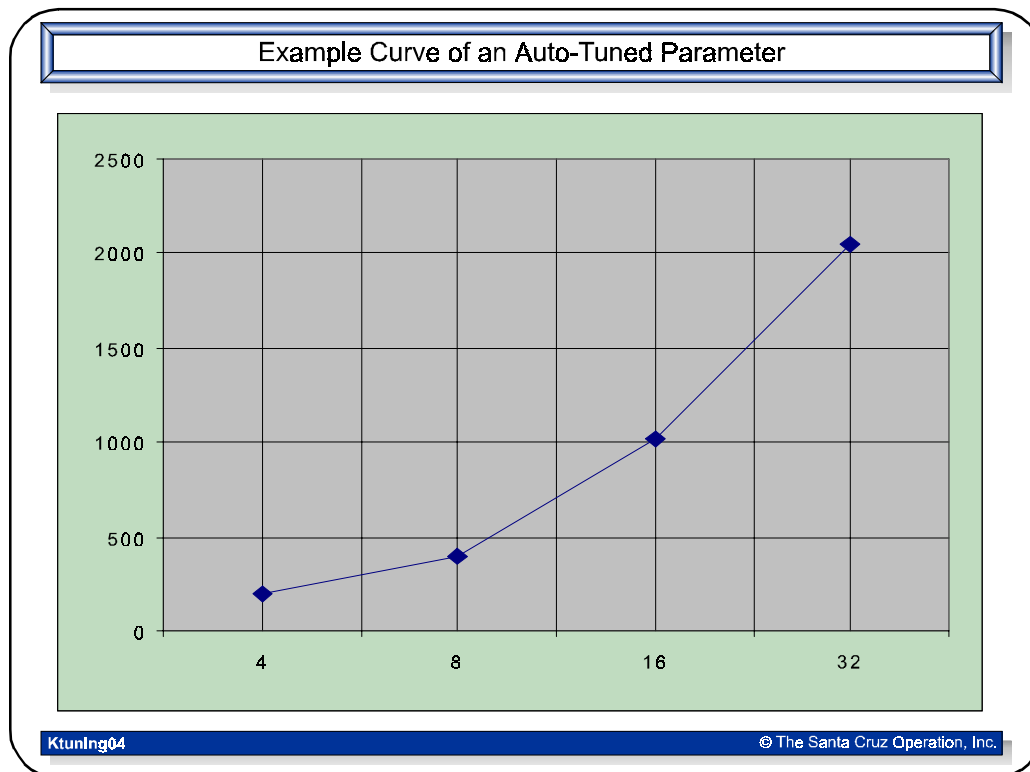
To use a scientific approach, either use an extrapolation formula or plot it on graph paper, extend the line, and determine the intersection. The computer uses a formula.

11.2.1.4 Stepping

Not all functions are linear. Among other possibilities, the function may be a *step* function. That is, the Y value may remain flat until X goes up to a certain amount, and then the Y value takes some kind of a jump. The graph in AV 11-3 on page 11-10 illustrates such a function.

In AV 11-3 on page 11-10, the Y value remains constant for all X values from 1 to 6. Then, when X is 6, the Y value suddenly jumps to 10 and stays there for all values of X from 6 to 10. There is nothing magical about the Y value being 10 when X goes to 6. Those are just numbers chosen for this example.

AV 11-4



11.2.2 Overview of Auto-tuning configuration

Auto-tuning variables are configured by writing a set of rules for the value of a tunable, based upon the amount of memory present in the system. The rules are called *tuning curves*. The curves are really a series of one or more linear or step functions, with the X values being the amount of memory present, and the Y values being the value of a parameter.

The graph in AV 11-4 on page 11-12 illustrates a sample curve for the default value of some kernel auto-tuned parameter. This shows that the value of the tunable differs dependent on the amount of system memory.

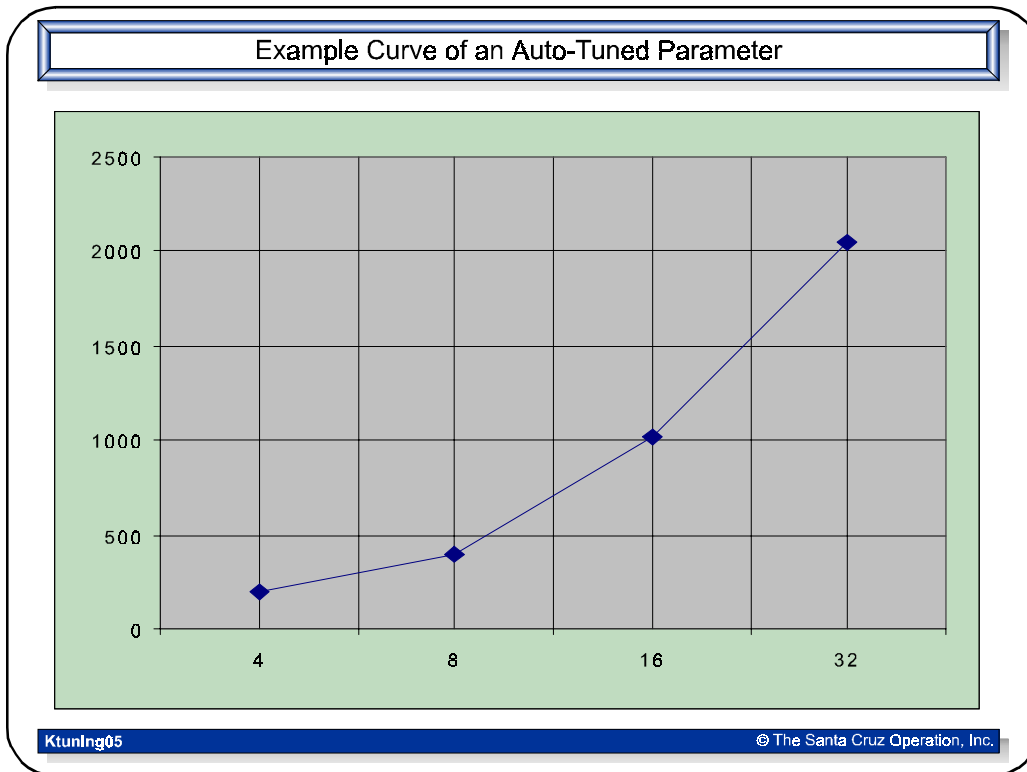
- 4 MB, the value of the tunable should be 200
- 8MB, the value of the tunable should be 400
- 16MB, the value of the tunable should be 1024
- 32 MB, the value of the tunable should be 2048

If the amount of memory present is between any of these numbers, it will take on a value along the line drawn between the numbers. For example, for:

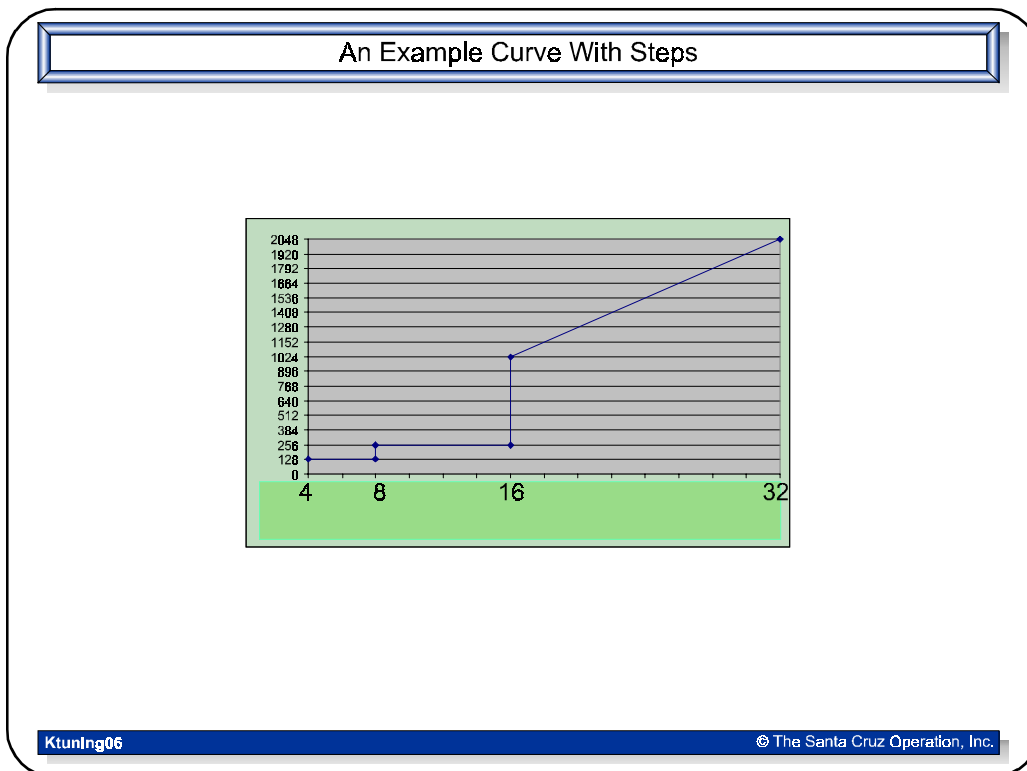
- 6 MB of memory (halfway between 4 and 8), then the value is 300 (halfway between 200 and 400)
- 12MB of memory (halfway between 8 and 16), then the value is 712
- 24MB of memory (halfway between 16 and 32), then the value is 1536

UnixWare determines the value *anywhere* along the lines using the end-points and *linear interpolation* (discussed earlier).

AV 11-5



AV 11-6



Sometimes the memory on a system is greater than the highest value specified as an end-point for a line. For example, referring again to AV 11-4 on page 11-12, if the system has more than 32MB of memory, UnixWare determines the value of the parameter using *linear extrapolation*.

When it uses extrapolation, it always extrapolates along the last line formed by the last two end-points.

Continuing the example, AV 11-5 on page 11-14 illustrates UnixWare extrapolating the value of the parameter along the last line to a memory value of 64MB (the dashed line is the extrapolation line extending the line formed by the last two end-points given in the configuration of the curve).

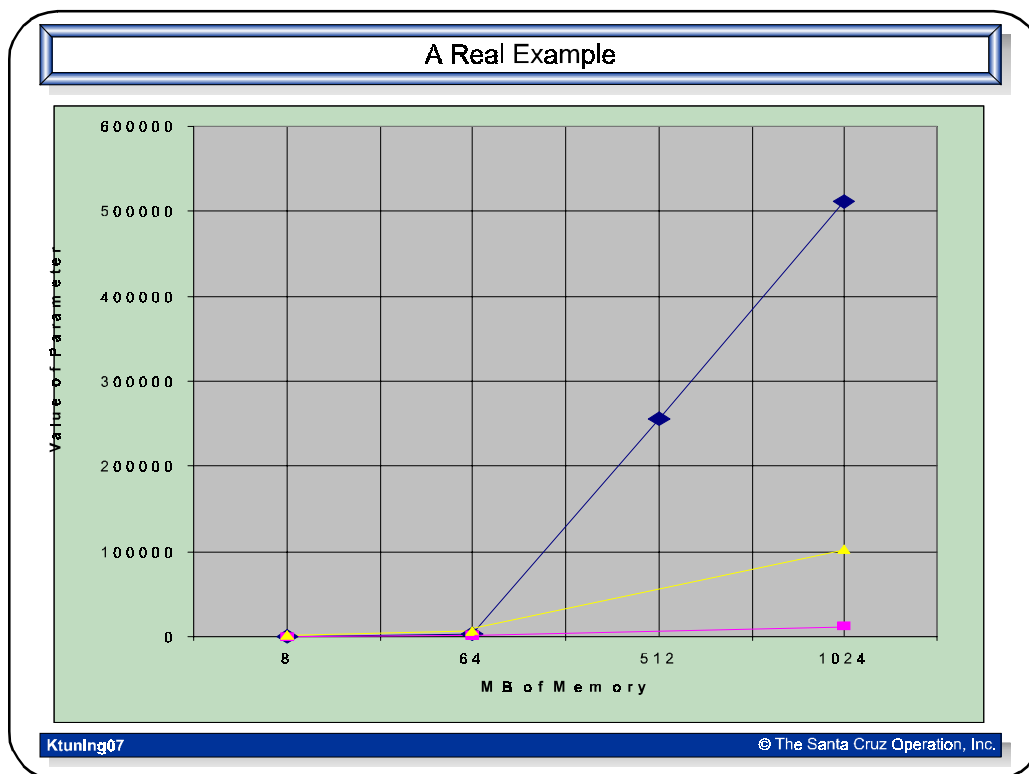
Sometimes, a parameter's curves includes step functions, not just linear ones as you have seen so far. In this case, determining the value of a parameter when the amount of memory present does not exactly match a data point is a bit more involved. Take for example, the following graph for determining the value of a parameter.

In AV 11-6 on page 11-14, the value of the parameter takes several jumps. If the amount of memory is:

- Up to 8MB, the value is 128
- Between 8MB to 16MB, the value is 256
- At 16MB, the value is 1024
- Between 16MB and 32MB, the value is interpolated between the values 1024 and 2048
- Greater than 32MB, the value is extrapolated along the line created by the (16,1024) and (32,2048) points.

Note Before trying to verify the information presented in this lesson by checking the auto-tuned configuration for your system, remember that the <i>memory</i> value being used in these curves is not total physical RAM, but the amount of physical memory less kernel memory; that is, pageable RAM. Later you will see how this memory value is determined.
--

AV 11-7



11.2.3 A Real Example

As previously mentioned, all kernel parameters must have a default, minimum, and maximum value. Therefore, although the graphs have only shown one *curve*, there are really three for each parameter; one each for the minimum, default, and maximum values. AV 11-7 on page 11-16 illustrates a full set of curves for the KV_RESV_MAX as they are configured within the default UnixWare 7 system. The graph is generated from this data:

Table 8:

CURVE	POINT TYPE	MB	VALUE
DEF	LINEAR	8	400
DEF	LINEAR	64	3200
DEF	LINEAR	512	256000
DEF	STEP	1024	512000
MIN	LINEAR	8	100
MIN	LINEAR	64	800
MIN	STEP	1024	12800
MAX	LINEAR	8	800
MAX	LINEAR	64	6400
MAX	STEP	1024	102400

Between the data in the table and the graph, observe that all values are linearly interpolated. This is because all points for each curve, except the last points on each curve, are of type LINEAR. The last points on each curve are of type STEP. When the last point on a curve is a STEP point, this means that there will be no linear extrapolation on that curve.

Notice that the lines keep getting farther apart as megabytes increase. When this happens between two or more curves, the curves are said to *diverge*. It is desirable to either have the curves diverge (as shown) or to have them parallel each other. Curves should never *converge* (come together), unless they do so at a STEP end-point. Otherwise, you may have a scenario where the minimum is supposed to be more than the default, maximum, or both.

11.2.3.1 The Mechanics of Auto-tuning (How Does It Work)

These concepts are needed to describe the step-by-step procedure for determining the value of an auto-tuned parameter.

1. A kernel module/driver writer creates an Autotune file according to the format described in the *Autotune(4dsp)* manual page.
2. The **idinstall(1M)** command copies the Autotune file for the module/driver into the */etc/conf/autotune.d* directory with a file name that matches the module's/driver's name.
3. At any time thereafter, you can modify this file. The modifications must meet all of the rules of logic presented here, such as non-convergence, and the rules of syntax described by the *Autotune(4dsp)* manual page.
4. At install or modification time, the kernel's tuning files are rebuilt by the **idbuild(1M)** command. Use the **-B** flag so you can observe the build and check the results of the changes before a reboot.
5. The **idbuild** command runs a command called **idreadauto** (no manual page available) and a few other commands which read the *Autotune(4dsp)* files and build the curves. The result of this action is that:
 - The default, minimum, and maximum values for each parameter are configured according to the amount of memory in the system at build time. These values are written into the */etc/conf/cf.d/mtune* file.
 - A check is made to determine that $MAX > DEF > MIN$ for each parameter. If this rule is broken, a warning message is displayed during the build.
6. The tuning curve for the default value is turned into a C structure and stored in the file */etc/conf/pack.d/modname/tune.c*, where *modname* is the name of the module from which the autotuning variables were read. As part of the kernel build, each of these files is compiled into a corresponding file called *tune.o*.
7. During kernel boot, the amount of memory is calculated and stored in a variable called *tunemem*. You can view the value of this variable using **crash(1M)**. Enter:

od -d tunemem
8. As each module is initialized, either at kernel boot or when the module is dynamically loaded, the code in that module will call **tune_calc**, passing a pointer to the C structures and a count of how many elements are in the structure.

9. The **tune_calc** routine looks at each point on the curve. Remember, it is only the default curve, until it either:
 - Finds a point for more memory than the **tunemem**, then it backs up the prior entry,OR
 - Runs out of points (it may have to invoke extrapolation), and then backs up to the last entry
10. If the first point is found, it is returned regardless of its type (STEP or LINEAR). This means that you are always guaranteed that a tunable will have at least the lowest value specified in its configuration file.
11. Once it has an entry-point, if that point is:
 - A STEP value, then the value of that point is returned
 - A LINEAR value, then either interpolation or extrapolation is used between this point and the previous one.

WARNING! There is currently no runtime verification that the resulting value does not violate the maximum or minimum values. If the default curves are modified so that they converge with linear extrapolation, and memory is added to a system without a kernel rebuild, you can end up with violations of minimum and maximum values with no warning being generated, because these checks are only made at kernel build time!

If you allow this to occur, there is a good chance that the kernel module or driver will either panic the system or, worse, corrupt data undetected.

11.2.3.2 Notes About *Autotune*(4dsp)

This section provides supplemental information not found in the *Autotune*(4dsp) manual page.

11.2.3.2.1 PHYS Entries

The `tunemem` variable, by default, is set to the amount of memory in the *page pool*—also known as pageable memory—that is not part of the kernel. There are times when a parameter's value should be based on the entire amount of memory (RAM size) in a system. To calculate the value of a parameter based on the size of physical size of RAM, precede the variable's rules with a single entry in the *Autotune* file that is the name of the variable in the first field and the text `PHYS` in the second field. No other fields are needed on this line.

11.2.3.2.2 Curve Rules and Enforcement

As previously stated, curves should not converge, as a rule. This is not stated in the manual page. Remember that convergence is only tested at the time of a kernel build. So, if you

1. Build a kernel on a system with a given amount of RAM and the `MAX > DEF > MIN` rule is ok for that value, and then
2. Add RAM to the system such that you now have converging curves,

you will have a default value that violates a `MIN` or `MAX` value and you will not be warned about it. Again, this may cause a kernel panic or worse, undetected data corruption.

The manual page implies that all values (`MIN`, `MAX`, and `DEFAULT`) are calculated at run-time, but this is only true for the default.

Furthermore, if you want linear interpolation or extrapolation, you must have at least two points. If you want extrapolation, the last point must be `LINEAR`. This is not documented in the manual page.

11.2.4 Overriding Auto-Tuned Parameters

You may need to fix the value of a parameter at constant value, or to set the system to use has a different amount of memory than it really has. You might want to do this to check that rules are generating the values you want during interpolation or extrapolation.

Use **id tune**(1M) to fix an auto-tuned parameter's value, as root, type the following at a command prompt:

```
/etc/conf/bin/id tune PARM value
```

When you do this, **id tune** warns you that this is an auto-tuned parameter and prompt you to verify that you want to override this characteristic. If you say yes, the parameter's *curves* are created in the appropriate *tune.c* file with only one parameter. Recall from the discussion of the mechanics of auto-tuning, that when there is only point for a curve, that value for that point is always used for the parameter's value.

To build the curves based on an amount of memory other than that present on the system where the kernel is built, use the **-t** flag to the undocumented **id readauto** command. For example, to see what values parameters would have on a system with 1GB of memory, as **root**, run:

```
/etc/conf/bin/id readauto -t 1024
```

The output is written to standard out in the same format as the */etc/conf/cf.d/mtune* file.

11.2.5 Applying This Knowledge

Unfortunately, the kernel does not keep all auto-tuning parameters in on convenient location, or use a single naming convention. In order to use utilities like **crash**(1M) or **kdb**(1M) to look at the values of auto-tuned parameters requires some additional knowledge.

For simplicity, most of the rest of this section only uses **crash**. You must be **root** to perform these exercises.

As previously stated, the kernel keeps the amount of memory used in calculating the value of tunables in the kernel variable called **tunemem**. You can display the value of **tunemem** from **crash** by entering:

```
od -d tunemem
```

This displays the value of **tunemem** in decimal.

Many of the tunables have their values stored in the `v` variable, which is actually a structure that contains many variables, all of which are tunables, many of which are auto-tuned. The **crash** utility has the `v` command to show the values of these parameters. Of the values displayed:

- `v_call` contains the calculated value of `NCALL`
- `v_maxup` contains the calculated value of `MAXUP`
- `v_bufhwm` contains the calculated value of `BUFHWM`
- `v_proc` contains the calculated value of `MAXPROC`

There is another variable, called `tune`, that is a structure (also called `tune`) that holds more of the values of the auto-tuned parameter. Unfortunately, **crash** does not have a convenient command for displaying the members of this variable's structure. However, you can teach **crash** about the `tune` structure and then display the structure's contents. To do this, within **crash**, enter these commands:

```
addstruct tune /usr/include/sys/tuneable.h
```

```
od -f tune tune
```

Exercise: Using crash(1M)

Identify which of the members of the `tune` structure are from auto-tuned parameters configured in files in the `/etc/conf/autotune.d` directory.

11.3 Performance Monitoring Tools

As an administrator, you have the same UnixWare 7-provided performance monitoring tools available to you in a cluster as you do on a non-cluster system, including **rtpm(1M)** and **sar(1M)**. These tools are discussed in the sections below.

11.3.1 The real-time performance monitoring tool, **rtpm(1M)**

The **rtpm(1M)** command is a standard UnixWare 7 tool that allows you to monitor the same items as **sar**, plus several more. It also allows you to graph two items at once, and to display several of the parameters on the screen in a format that is more condensed than that of **sar**.

The **rtpm** command has not been clusterized for NSC. However, you can still use it to analyze one node at a time. To do this, run **rtpm** with the **onnode(1)** command. For example, to use **rtpm** to analyze node 2, as **root**, enter:

```
onnode 2 rtpm
```

To analyze several nodes at once, start several X terminal emulators and run the **rtpm** with **onnode** one each node to be analyzed.

Exercise: Analyzing two nodes using rtpm

- Step 1. Log in as **root** in a graphical environment.
- Step 2. Start three terminal emulators (**xterm**, **dtterm**, or **Eterm**).
- Step 3. In one of the terminal emulator windows, enter:
- onnode 1 rtpm**
- Step 4. In another terminal emulator window, enter:
- onnode 2 rtpm**
- Step 5. In the third terminal emulator window, start the **spinner** program located in */labs/lab10* on node 1. Enter:
- onnode 1 /labs/lab10/spinner &**
- Record the PID of the process. Notice the CPU utilization on node 1. It should increase. Also take note of the CPU utilization on node 2.
- Step 6. Force the **spinner** program to migrate to node 2. Enter:
- migrate 2 pid**
- where *pid* is the pid of the process recorded in Step 5. Notice that the CPU utilization goes up on node 2 and down on node 1.
- Step 7. Close the terminal emulation windows.

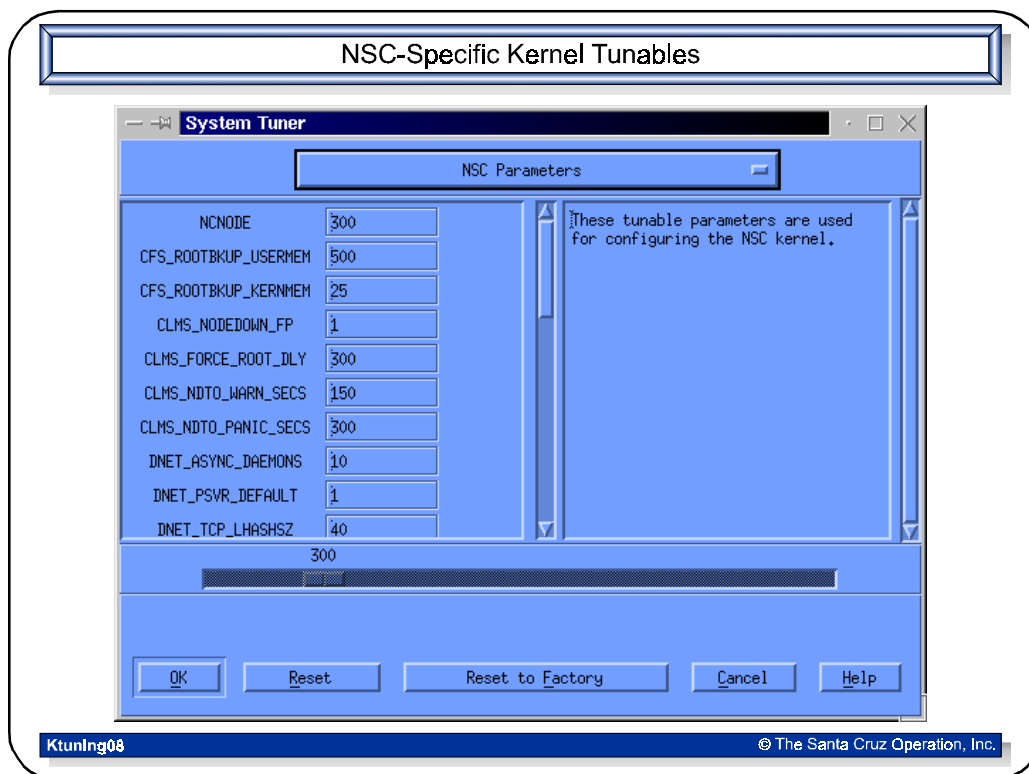
11.3.2 The system activity report, **sar(1M)**, program

The sar utility has been clusterized to provide either:

- Cluster-wide performance statistics
- Node-specific performance statistics

See the **sar(1M)** manual page for details on how to get cluster-wide or node-specific information.

AV 11-8



11.4 NSC-Specific Kernel Tunables

The NSC kernel adds several new tunables to the kernel. Each tunable has its own description which you can view through the **System Tuner** application. To do this:

- Step 1. Login to the cluster as **root** using an environment supporting X-windows.
- Step 2. Start **scoadmin**.
- Step 3. Open the **System** folder.
- Step 4. Double-click on **System Tuner**.
- Step 5. Select **NSC Parameters** in the pull-down menu at the top of the screen. This displays a window similar to the one shown in AV 11-8 on page 11-26.
- Step 6. Select any of the parameters on the left and a corresponding description is displayed on the right.

Exercise: Viewing NSC kernel parameters

Start the **System Tuner** and view the NSC tunable parameters.

11.5 Using idtune(1M) to Build an NSC Kernel

The **idbuild**(1M) and **idtune**(1M) commands that build and tune kernels were discussed in significant detail in the *Unix System Administration II Using UnixWare 7* course. In general, they work the same in NSC environments as they do in a non-NSC environment.

The NSC-specific issues you must consider are:

- The **-c** flag to the **idtune** command changes the running kernel. Since each node is running their own copy of the kernel, you must **idtune -c** in conjunction with the **onall**(1) command. For example, to change the value of the NAUTOUP parameter to 10 on all running kernels, enter:

```
onall /etc/conf/bin/idtune -c NAUTOUP 10
```

- New kernels are made by **idbuild** and placed in the */etc/conf/cf.d/unix* file.

Normally, the */etc/conf/bin/idcpunix* program is run when the node next shuts down to make a copy of the kernel in the *stand* slice. This command has been clusterized to copy the kernel to the stand slice of every node that is *up*.

If a node is down during the kernel rebuild, it does not get a copy of the new kernel in its *stand* slice. To remedy this, when the new node starts, copy the kernel from */etc/conf/cf.d/unix* to */stand/N/file*, where *N* is the node name and *file* is the name specified by the **bootprog** parameter for that node, typically *unix.1*. For example, to refresh node 2, enter:

```
copy -mo /etc/conf/cf.d/unix /stand/2/unix.1
```

After copying over the new *unix.N*, reboot the refreshed node.

11.6 Lesson Summary

This lesson has discussed how performance tuning on a cluster differs from that of a non-cluster computer. Specifically, it discussed:

- How to calculate auto-tuning parameters
- How to configure auto-tuning parameters
- Performance monitoring tools available in NSC
- Building kernels in an NSC environment

Review questions

1. How to auto-tuning parameters differ from other kernel parameters?
2. What is the purpose of an *Autotune* file?
3. What are the considerations for building new kernels in an NSC environment?

11.7 Relevant Commands and Files

The following commands and files were presented in this lesson.

Relevant commands

id tune (1M)	Changes the values of kernel tunables
id build (1tcp)	Builds a UNIX kernel
crash (1M)	Analyzes a kernel dump file
kdb (1M)	Debugs the UNIX kernel
id readauto (1M)	Reads the Autotune (4dsp) files and builds the tuning curves
on node (1)	Runs a process on a specified node
rtpm (1M)	Monitors cluster performance
sar (1M)	Monitors system performance

Relevant files

<i>/etc/conf/cf.d/unix</i>	Contains the UNIX kernel on non-NSC systems
<i>/etc/conf/cf.d/mtune</i>	Defines MIN, MAX, and DEFAULT values for UNIX kernel tunables
<i>/etc/conf/cf.d/pack.d</i>	Contains C structures which define tuning curves for kernel parameters
<i>/stand/N/unix.n</i>	Contains the UNIX kernel on NSC nodes
<i>/etc/conf/autotune.d</i>	Contains Autotune files for kernel drivers. See <i>Autotune</i> (4dp).

11.8 Answers to Exercises and Review Questions

This section contains the answers to exercises, review questions, and labs presented in this lesson.

Review Question Answers

This section contains the answers to the review questions on page 11-29.

1. Autotuning parameters have minimum and maximum values based on the amount of memory on the system when the kernel is built. The default value is set when the system is booted, based on the amount of memory present.

Traditional kernel values have defined minimum, maximum, and default values that do not depend on other system configuration values.

2. The *Autotune* file defines the rules for determining the values assigned to autotuning parameters.
3. If a node is down during a kernel rebuild, the **idcpunix** program does not copy the new kernel to the *stand* slice on that node. You must copy the kernel from */etc/conf/cf.d/unix* to */stand/N/unix.N*, where N is the node number.

