

HP Serviceguard I

H6487S L.00



Student guide — Part 1

Use of this material to deliver training without prior written permission from HP is prohibited.

HP Serviceguard I

H6487S L.00



Student guide — Part 1

Use of this material to deliver training without prior written permission from HP is prohibited.

© Copyright 2007 Hewlett-Packard Development Company, L.P.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

This is an HP copyrighted work that may not be reproduced without the written permission of HP. You may not use these materials to deliver training to any person outside of your organization without the written permission of HP.

Pentium®, Intel Itanium™ Logo: Intel, Intel Inside, and Itanium registered trademarks of Intel Corporation in the U.S. and other countries and are used under license.

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Microsoft®, Windows®, and MS Windows® are U.S. registered trademarks of Microsoft Corp.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of The Open Group.

Export Compliance Agreement

Export Requirements. You may not export or re-export products subject to this agreement in violation of any applicable laws or regulations.

Without limiting the generality of the foregoing, products subject to this agreement may not be exported, re-exported, otherwise transferred to or within (or to a national or resident of) countries under U.S. economic embargo and/or sanction including the following countries:

Cuba, Iran, North Korea, Sudan and Syria.

This list is subject to change.

In addition, products subject to this agreement may not be exported, re-exported, or otherwise transferred to persons or entities listed on the U.S. Department of Commerce Denied Persons List; U.S. Department of Commerce Entity List (15 CFR 744, Supplement 4); U.S. Treasury Department Designated/Blocked Nationals exclusion list; or U.S. State Department Debarred Parties List; or to parties directly or indirectly involved in the development or production of nuclear, chemical, or biological weapons, missiles, rocket systems, or unmanned air vehicles as specified in the U.S. Export Administration Regulations (15 CFR 744); or to parties directly or indirectly involved in the financing, commission or support of terrorist activities.

By accepting this agreement you confirm that you are not located in (or a national or resident of) any country under U.S. embargo or sanction; not identified on any U.S. Department of Commerce Denied Persons List, Entity List, US State Department Debarred Parties List or Treasury Department Designated Nationals exclusion list; not directly or indirectly involved in the development or production of nuclear, chemical, biological weapons, missiles, rocket systems, or unmanned air vehicles as specified in the U.S. Export Administration Regulations (15 CFR 744), and not directly or indirectly involved in the financing, commission or support of terrorist activities.

Printed in the USA

HP Serviceguard I

Student guide — Part 1

April 2007

Contents

Overview

Student Performance Objectives	1
Student Profile and Prerequisites	6
Agenda	7

Module 1 — Introduction to High Availability

1-1. SLIDE: What Causes a System to Go Down?	1-2
1-2. SLIDE: Causes of Failures	1-4
1-3. TEXT PAGE: Causes of Downtime	1-6
1-4. SLIDE: Not a Big Deal? You Tell Me!	1-8
1-5. SLIDE: Average Cost per Hour of Downtime	1-9
1-6. SLIDE: What Is High Availability?	1-10
1-7. SLIDE: Computer System Availability	1-11
1-8. SLIDE: Network Availability	1-13
1-9. SLIDE: Application Availability	1-15
1-10. SLIDE: Three Pillars of High Availability	1-17
1-11. SLIDE: 5nines Support Partnerships	1-18
1-12. SLIDE: High Availability Terms	1-19
1-13. SLIDE: High Availability Percentages	1-21
1-14. SLIDE: Availability Continuum Hierarchy	1-22
1-15. SLIDE: For More Information	1-23
1-16. SLIDE: Course Agenda	1-24
1-17. LAB: Prepare your Systems for Later Modules: Configure IP Addresses	1-25

Module 2 — Introduction to Serviceguard

2-1. SLIDE: Introducing Serviceguard	2-2
2-2. SLIDE: High Availability with Serviceguard	2-3
2-3. SLIDE: Features and Benefits of Serviceguard	2-4
2-4. SLIDE: How Serviceguard Works	2-6
2-5. SLIDE: Serviceguard Packages	2-7
2-6. SLIDE: Redistributing Application Packages	2-8
2-7. SLIDE: Minimizing Planned Downtime	2-9
2-8. SLIDE: Serviceguard Bundle/Products	2-10
2-9. LAB: Installation of Serviceguard Software	2-11

Module 3 — High Availability Planning

3-1. SLIDE: What Are the Risks?	3-2
3-2. SLIDE: Reducing the Risk (Redundant Data)	3-4
3-3. SLIDE: Reducing the Risk (Minimizing Downtime)	3-5
3-4. SLIDE: Reducing the Risk (Network Connectivity)	3-6
3-5. SLIDE: Reducing the Risk (Asymmetric Clusters and Other Issues)	3-7
3-6. SLIDE: Disk Configurations	3-9
3-7. SLIDE: LVM Mirrored JBODs	3-10
3-8. SLIDE: Disk Array Configuration	3-11
3-9. SLIDE: Highly Available SANs	3-12
3-10. SLIDE: Network Configurations	3-13
3-11. SLIDE: Redundant LANs	3-14
3-12. SLIDE: Normal Network Flow	3-16
3-13. SLIDE: Network Flow during Package Failover	3-17

Contents

3-14.	SLIDE: Routed Networks and Serviceguard Clusters	3-18
3-15.	SLIDE: Configuration Roadmap	3-19
3-16.	LAB: Hardware Inventory	3-20
3-17.	LAB: Configuring and Managing LVM Boot Disks on Integrity-Based Systems (Optional).....	3-28
3-18.	LAB: Configuring and Managing LVM Boot Disks on PA-Risc-Based Systems (Optional).....	3-35

Module 4 — LVM for Serviceguard

4-1.	SLIDE: Review of LVM Concepts.....	4-2
4-2.	SLIDE: Serviceguard Configuration Using Mirrored Disks	4-4
4-3.	SLIDE: Step 1: Configure Volume Group on First System	4-5
4-4.	SLIDE: Step 2: Import Volume Group to Second System	4-7
4-5.	SLIDE: VG Export and Import -s Option.....	4-9
4-6.	SLIDE: LVM Issues for Using Serviceguard.....	4-10
4-7.	LAB: Building a Shared Volume Group	4-12
4-8.	LAB Solution: Building a Shared Volume Group	4-17

Module 5 — Cluster Concepts and Configuration

5-1.	SLIDE: Definition of a Cluster.....	5-2
5-2.	SLIDE: Major Components of a Cluster	5-3
5-3.	SLIDE: Network Interface Configuration	5-4
5-4.	SLIDE: Cluster Lock Configuration Using LVM Disks	5-6
5-5.	SLIDE: Cluster Lock Configuration Using a Quorum Server	5-7
5-6.	SLIDE: Heartbeat Configuration	5-9
5-7.	SLIDE: cmc ld Process.....	5-10
5-8.	SLIDE: Cluster Formation Requirements	5-11
5-9.	SLIDE: Steps to Configure a Cluster.....	5-13
5-10.	SLIDE: Step 1: Build Cluster Configuration ASCII File	5-14
5-11.	SLIDE: Cluster Management Options.....	5-17
5-12.	TEXT PAGE: The cmquerycl command.....	5-19
5-13.	TEXT PAGE: Sample Cluster Configuration ASCII File Using LVM Disks for Cluster Lock Services.....	5-21
5-14.	TEXT PAGE: Sample Cluster Configuration ASCII File Using Quorum Server Technology for Cluster Lock Services	5-25
5-15.	SLIDE: Step 2: Compile and Distribute Binary File	5-29
5-16.	TEXT PAGE: The cmcheckconf Command	5-30
5-17.	TEXT PAGE: The cmapplyconf Command	5-31
5-18.	SLIDE: Step 3: Start the Serviceguard Daemons.....	5-32
5-19.	TEXT PAGE: The cmrunc l Command.....	5-33
5-20.	SLIDE: Cluster Configuration Procedure.....	5-34
5-21.	SLIDE: Viewing the Cluster — cmviewcl Command	5-36
5-22.	SLIDE: Checking the Cluster Log.....	5-38
5-23.	LAB: Build a Cluster	5-39
5-24.	LAB: Cluster Lock Using A Quorum Server	5-40
5-25.	LAB: Cluster Lock Using LVM Disks	5-51
5-26.	LAB Solution: Cluster Lock Using A Quorum Server	5-59
5-27.	LAB Solution: Cluster Lock Using LVM Disks.....	5-71

Module 6 — Additional Cluster Features

6-1.	SLIDE: Additional Cluster Features	6-2
6-2.	SLIDE: Serviceguard Volume Groups	6-3
6-3.	SLIDE: Marking Volume Groups for Use in Serviceguard.....	6-5
6-4.	SLIDE: Exclusive Mode Volume Group Activation.....	6-7
6-5.	SLIDE: Cluster Formation and Reformations.....	6-8
6-6.	SLIDE: Ways to Initially Form the Cluster	6-9
6-7.	SLIDE: Node Failures and Node Joins.....	6-10
6-8.	SLIDE: Cluster Reformation Example	6-12
6-9.	SLIDE: Local LAN Card Failover	6-14
6-10.	SLIDE: Normal Network Flow	6-15
6-11.	SLIDE: Network Flow to Standby LAN Card	6-16
6-12.	LAB: Shared Disks/Shared Volume Groups	6-17
6-13.	LAB: Exiting and Joining the Cluster	6-21
6-14.	LAB: Local LAN Card Failover (Using Remote Equipment)	6-24
6-15.	LAB: Local LAN Card Failover (Using Local Equipment)	6-26
6-16.	LAB Solution: Shared Disks/Shared Volume Groups.....	6-29
6-17.	LAB Solution: Exiting and Joining the Cluster	6-33
6-18.	LAB Solution: Local LAN Card Failover (Using Remote Equipment).....	6-36
6-19.	LAB Solution: Local LAN Card Failover (Using Local Equipment).....	6-38

Module 7 — Packages and Services

7-1.	SLIDE: Packaging Concepts.....	7-2
7-2.	SLIDE: Sample Package Configuration	7-3
7-3.	SLIDE: Sample Configuration after Node Failure	7-4
7-4.	SLIDE: Package Switching	7-6
7-5.	SLIDE: Viewing Package Status.....	7-8
7-6.	SLIDE: Modifying Package Status	7-9
7-7.	SLIDE: Step 1: Build Package Configuration File	7-10
7-8.	TEXT PAGE: Sample Package Configuration File	7-14
7-9.	SLIDE: Step 2: Build Package Run/Halt Script	7-21
7-10.	TEXT PAGE: Sample Control Script File	7-24
7-11.	SLIDE: Step 3: Recompile and Distribute Binary File.....	7-34
7-12.	SLIDE: Package Scripts	7-35
7-13.	SLIDE: Package Owner and State	7-37
7-14.	SLIDE: Package Configuration Procedure	7-39
7-15.	SLIDE: The Package Script Log File	7-41
7-16.	SLIDE: Review of Commands for Controlling a Cluster.....	7-42
7-17.	REVIEW: Check Your Understanding.....	7-44
7-18.	LAB: Package Configuration #1: logger Package	7-47
7-19.	REVIEW Solution: Check Your Understanding	7-52
7-20.	LAB Solution: Package Configuration #1: logger Package.....	7-55

Module 8 — Package Policies

8-1.	SLIDE: Package Policies.....	8-2
8-2.	SLIDE: Failover Policies.....	8-4
8-3.	SLIDE: FAILOVER_POLICY: CONFIGURED_NODE	8-5
8-4.	SLIDE: FAILOVER_POLICY: MIN_PACKAGE_NODE	8-6
8-5.	SLIDE: Rotating Standby	8-7
8-6.	SLIDE: Failback Policies	8-8

Contents

8-7.	SLIDE: Example of Automatic Failback	8-9
8-8.	SLIDE: Access Control Policies	8-10
8-9.	SLIDE: Node Fail Fast and Service Fail Fast.....	8-12
8-10.	LAB: FAILBACK POLICY	8-15
8-11.	LAB: Access Control Policy	8-16
8-12.	LAB: SERVICE_FAIL_FAST_ENABLED (Optional).....	8-18
8-13.	LAB: NODE_FAIL_FAST_ENABLED (Optional)	8-20
8-14.	LAB Solution: FAILBACK POLICY.....	8-23
8-15.	LAB Solution: Access Control Policy	8-25
8-16.	LAB Solution: SERVICE_FAIL_FAST_ENABLED (Optional)	8-27
8-17.	LAB Solution: NODE_FAIL_FAST_ENABLED (Optional)	8-29

Module 9— Application Monitoring Scripts

9-1.	SLIDE: Actions Performed by Run Script.....	9-2
9-2.	SLIDE: Control Script.....	9-3
9-3.	SLIDE: Rules for Service Processes	9-4
9-4.	SLIDE: Application Monitoring Script.....	9-6
9-5.	SLIDE: Example: The logprog Program.....	9-8
9-6.	LAB: logprog Package Monitoring Script	9-9
9-7.	LAB Solution: logprog Package Monitoring Script.....	9-12

Module 10 — Best Practices

10-1.	SLIDE: Best Practices.....	10-2
10-2.	SLIDE: Monitoring the sys log File.....	10-3
10-3.	SLIDE: Package Script Troubleshooting.....	10-5
10-4.	SLIDE: Package Script Log — Example 1.....	10-6
10-5.	SLIDE: Package Script Log — Example 2.....	10-8
10-6.	SLIDE: Common Cluster Configuration Issues	10-10
10-7.	SLIDE: Common Package Configuration Issues	10-12
10-8.	SLIDE: Testing Cluster Operations	10-14
10-9.	SLIDE: Useful Troubleshooting Commands.....	10-17
10-10.	SLIDE: The Built-In Safety Net.....	10-20
10-11.	SLIDE: Lost Cluster Lock Disk (LVM Disks)	10-21
10-12.	SLIDE: Monitoring Cluster and Package Resources.....	10-23
10-13.	SLIDE: Notification for Package Failure.....	10-25
10-14.	LAB: Package Monitoring with Email Notification.....	10-26
10-15.	LAB: Using the cmset log Command.....	10-28
10-16.	LAB: Package Monitoring with Email Notification.....	10-29
10-17.	LAB: Using the cmset log Command.....	10-31

Overview

The *Hands On with Serviceguard* course (H6487S) is an advanced course that teaches system administrators how to increase system availability with Serviceguard.

This course describes the concepts of high availability and fault tolerance, and discusses hardware solutions to achieve optimal availability. Other topics discussed within the framework of high availability are networking with Serviceguard, performance management, automated monitoring, backups, and monitoring Serviceguard clusters.

The course is NOT designed for software developers who will be writing applications to run in an HP Serviceguard environment. Although the course will give the software developers a good overview, they need to reference the white paper, "Designing Highly Available Cluster Applications," available through this web site:

<http://gsslweb.cup.hp.com/ServiceGuard>

Student Performance Objectives

Module 1 — Introduction to High Availability

- Define terms like standard availability, high availability, and fault tolerant.
- Identify single points of failure for a computer system.
- Identify causes of downtime and strategies for reducing downtime.
- Configure additional non-routable IP addresses

Module 2 — Introduction to Serviceguard

- Describe three features and benefits of Serviceguard.
- Describe on a high level how Serviceguard works.
- List the components of a Serviceguard package.

Module 3 — High Availability Planning

- Identify specific actions that can be taken to improve High Availability.
- Identify multiple risks (of SPOF) and list a couple of options for reducing the risks.
- Identify different disk technology for a High Availability environment.
- Identify different network configuration for a High Availability environment.
- Mirror your root volume group.

Overview

Module 4 — LVM for Serviceguard

- Understand how to configure a shared volume group for a Serviceguard environment.
- Understand how to export and import volume groups without having to specify disk devices.
- Understand how to activate a shared volume group in exclusive read/write access mode.
- Understand the list of operating system files that need to be modified to support shared volume groups in a Serviceguard environment.

Module 5 — Cluster Concepts and Configuration

- Describe the differences between heartbeat LAN cards, stationary LAN cards, and standby LAN cards.
- Configure a Serviceguard cluster using a cluster lock disk or using Quorum Server technology.
- Configure the heartbeat-polling interval for a Serviceguard cluster.
- Describe three functions performed by the Serviceguard cluster manager daemon.
- Describe the major requirement needed to bring up a Serviceguard cluster.
- Describe Access Control Policies for a cluster
- Implement the systematic procedure for defining, configuring, and bringing up a Serviceguard cluster.

Module 6 — Additional Cluster Features

- Mark a volume group as belonging to a Serviceguard cluster.
- Activate a shared volume group in an exclusive read/write access mode.
- Configure and test the local LAN failover capability of the Serviceguard cluster.
- Describe the differences between heartbeat LAN cards and standby LAN cards.
- List two methods in which a node can *join* an existing cluster.
- List two methods in which a node can *leave* an existing cluster.

Module 7 — Packages and Services

- Configure a basic package to run in a Serviceguard environment.
- Describe the fields in a package configuration file.
- Describe the variables in a package control script file.
- Provide the command to re-enable a package for switching.
- Interpret the output of `cmviewcl -v` as it relates to packages.
- Provide the full path name of a package-specific log file.

Module 8 — Packages Policies

- Use `FAILOVER_POLICY` to determine which node a package starts up on after the package fails.
- Use the `FAILBACK_POLICY` to determine which node a package starts up on after a node fails.
- Determine proper Access Control Policies for a package
- Determine when `NODE_FAIL_FAST` or `SERVICE_FAIL_FAST` should be enabled.

Module 9 — Application Monitoring Scripts

- Configure and implement an application monitor.
- List actions performed by the package control script.
- List three rules for Serviceguard service processes.

Module 10 — Best Practices

- View and interpret current activities within the cluster.
- View cluster-related activities over time.
- View activities performed by the `run` and `halt` scripts.
- Use contributed Serviceguard commands to perform troubleshooting.
- Understand the Safety Time Register.
- Replace a failed LVM lock disk.

Overview

Module 11 — Cluster and Package Online Reconfiguration

- Change the cluster configuration while the cluster is running.
- Add a node to the cluster while the cluster is running.
- Remove a node from the cluster while the cluster is running.
- Add a package to a running cluster.
- Delete a package from a running cluster.
- Modify specific package attributes while the package is running.
- Modify a down package while the cluster and other packages are running.
- Perform the above operations by using the HP-UX command line.

Module 12 — Highly Available NFS

- Use the Serviceguard NFS toolkit.
- List files in the NFS toolkit.
- Set up a Serviceguard package using the NFS toolkit.

Module 13 — The Highly Available Oracle Database

- List different Serviceguard toolkits.
- Use Oracle toolkit and cookbook.
- Configure the Oracle 10g Database to run in a Serviceguard environment.

Module 14 — EMS Resources and Serviceguard Packages

- List the five EMS monitoring daemons.
- Use the `res ls` command to list the available EMS resources.
- Set up a Serviceguard package to use EMS resources.

Module 15 — High Availability Networking

- Connect a second network to a Serviceguard cluster with multiple levels of redundancy.
- Set up a Serviceguard node to provide LAN card redundancy.
- Configure a highly available network using redundant hubs.

- Configure a highly available network using redundant routers.
- Configure a highly available network using redundant networks.

Module 16 — Rolling Upgrade Issues

- Explain why rolling upgrades are carried out.
- List provisions for rolling upgrades.
- Know when rolling upgrades will fail.
- Carry out a rolling upgrade.

Module 17 — LVM Maintenance for Packages

- Add / remove a disk to/from a volume group owned by a package.
- Add / remove a logical volume to/from a volume group owned by a package.
- Add / remove a volume group to/from a package.
- Make a logical volume and file system (in a volume group owned by a package) larger or smaller.

Module 18 — ServiceGuard Manager

- Discuss the components of the Serviceguard Manager.
- Navigate the Serviceguard Manager GUI.

Module 19 — Troubleshooting Scenarios and Final Review

- Perform troubleshooting activities to resolve nine different Serviceguard configuration problems.
- Successfully complete 15 final review questions.

Overview

Student Profile and Prerequisites

This course is designed for HP-UX system administrators who develop, design, implement, and monitor Serviceguard clusters.

In order to understand the material in the Serviceguard course, to work through the labs in the course, and to work with the product in a production environment, it is essential that the student have a background in the following four areas:

1. Shell Programming — the entire process of configuring user applications to run in highly available, mission critical, environments involves writing shell scripts to start, halt and monitor the applications. A background with shell programming is essential. *POSIX Shell Programming* (H5888S), is recommended to improve in shell programming.
2. HP-UX System Administration —this course focuses on how the administration of a HP-UX server changes when it is part of a Serviceguard cluster. This course assumes the student is already familiar with standard HP-UX administration. *HP-UX System and Network Administration I* (H3064S), or *HP-UX System and Network Administration for Experienced UNIX Administrators* (H5875S), is recommended to improve in HP-UX system administration.
3. TCP/IP and basic networking — Serviceguard is designed for client/server applications and relies heavily on reassigning IP addresses from one LAN card to another, moving IP addresses from one system to another, and ARP broadcasts. *HP-UX System and Network Administration II* (H3065S), is recommended to improve in TCP/IP and basic networking.
4. LVM — The administration of Serviceguard volume groups is distinctly different from the administration of volume groups in a non-Serviceguard environment. A thorough knowledge of LVM is required to understand and confidently perform administrative tasks for volume groups in a Serviceguard clustered environment. Even as simple a task as adding a disk to a volume group in order to expand the file system/database, if not performed correctly for a clustered environment (and the standard non-clustered process is not correct), will break the cluster and leave the data vulnerable to corruption. *Hands On with Logical Volume Manager and MirrorDisk/UX* (H6285S), is highly recommended for students with limited experience with LVM. For new HP-UX administrators, H3064S and H5875S do not provide enough detail on LVM to prepare the student for Serviceguard.

Agenda for this 5-day class

Day 1

Module 1	Introduction to High Availability
Module 2	Introduction to Serviceguard
Module 3	High Availability Planning
Module 4	LVM for Serviceguard

Day 2

Module 5	Cluster Concepts and Configuration
Module 6	Additional Cluster Features
Module 7	Packages and Services
Module 8	Package Policies

Day 3

Module 9	Application Monitoring Scripts
Module 10	Best Practices
Module 11	Cluster and Package Online Reconfiguration
Module 12	Highly Available NFS

Day 4

Module 13	The Highly Available Oracle Database
Module 14	EMS Resources and Serviceguard Packages
Module 15	High Availability Networking
Module 16	Rolling Upgrade Issues

Day 5

Module 17	LVM Maintenance
Module 18	Serviceguard Manager
Module 19	Troubleshooting Scenarios and Final Review

Overview

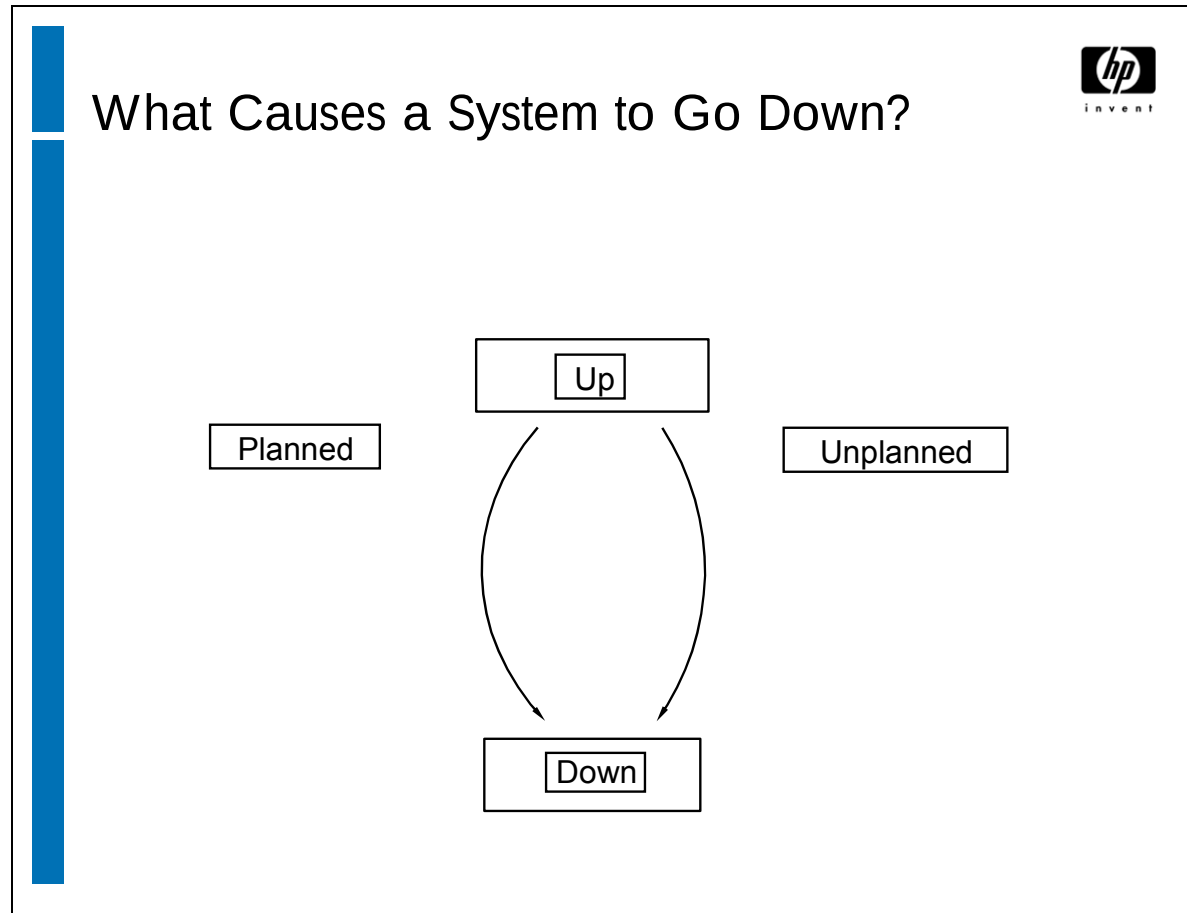
Module 1 — Introduction to High Availability

Objectives

Upon completion of this module, you will be able to do the following:

- Define terms like standard availability, high availability, and fault tolerant.
- Identify single points of failure for a computer system.
- Identify causes of downtime and strategies for reducing downtime.
- Configure additional non-routable IP addresses

1-1. SLIDE: What Causes a System to Go Down?



Student Notes

One objective of any system administrator is to keep the computer systems up and running. This objective becomes more important during business hours. It becomes even more important when the application is "mission critical" and millions of dollars are at stake when the computer system goes down.

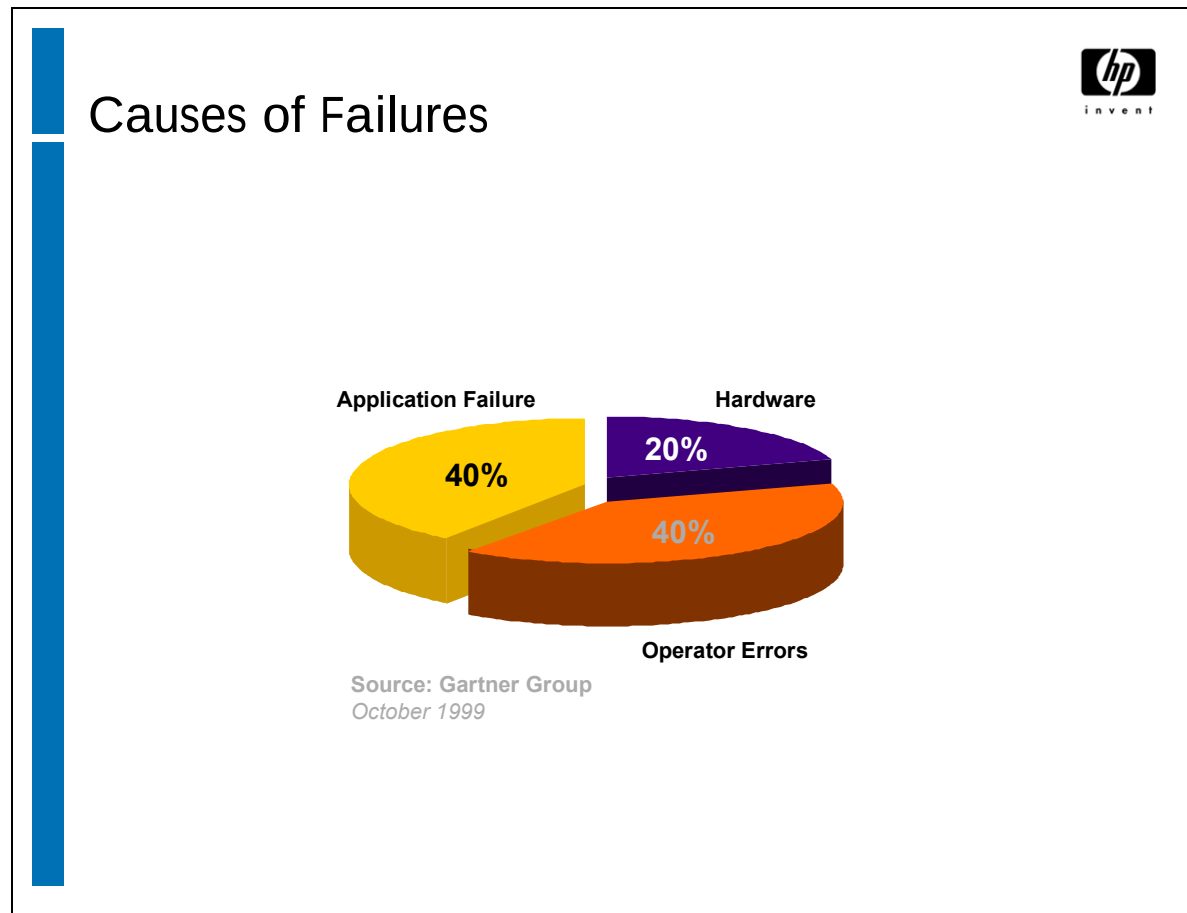
The following are some *planned* reasons why an application might become unavailable:

- To reconfigure the kernel.
- To apply patches.
- To perform software upgrades.
- To perform hardware upgrades.
- To perform disk/lvm maintenance.
- To perform full system backups.
- To perform system maintenance.
- To perform database maintenance.

The following are some *unplanned* reasons why a computer system may go down:

- Hardware failures
- System panics
- Application errors
- Power failures
- User errors
- Software defects
- Natural disasters

1-2. SLIDE: Causes of Failures



Student Notes

Recent research analyzing causes of system failures show three broad categories of failures: hardware, application, and user or operator errors.

Operator and User Errors

The bulk of system outages today are the result of operator and user errors. Processes and procedures are just as important to high availability as the technology. The best technology will not reduce downtime if users and operators are not properly trained. Operator errors can be reduced through better tools, increased training, easier to use applications, and centralized administration.

Application and Software Failures

With software releases rolling out at ever increasing rates, the likelihood of software bugs also increases. Recent statistics indicate over 40 percent of system failures are software related. Though many of these bugs have patches, unless the patches are applied, the system becomes vulnerable to software failures.

Hardware Failures

Hardware has traditionally been the most common cause of system failures, but recent research indicates that only 20 percent of today's system failures are hardware related. This statistic reflects the fact that hardware reliability has been increasing over the years.

Hardware failures would include such components as CPU boards, memory cards, disk controller cards, disk drives, and network controller cards.

1–3. TEXT PAGE: Causes of Downtime

Planned Downtime

Causes of Downtime

Backups

Hardware/software/application
reconfiguration/upgrade

Install patch

Database maintenance/conversion

Disk/LVM maintenance

Strategies for Minimizing Downtime

Hot backups and database hot backups.

Rolling upgrade with Serviceguard
and self-terminating SCSI cable.

Rolling upgrade with Serviceguard.

Typically requires downtime.

On-line JFS, hot swappable disks, and RAID
devices (like HP's XP Array Series).

Unplanned Downtime

Causes of Downtime

Hardware failure, SPOF
(Single Point of Failure)

CPU, memory

Disks

SCSI/Fibre Channel cards

Network card

Network infrastructure component

Client (PC)

Second hardware failure (if
redundant component also fails)

Power failure

Software defects

Human error

Strategies for Minimizing Downtime

SuperDome systems (or those systems with
redundant hardware components).

SuperDome systems, Serviceguard clusters, and
Serviceguard clusters using Oracle 9i or Oracle
10g, with or without RAC.

LVM mirroring and disk arrays.

PV links and LVM mirroring.

Backup network card through Serviceguard,
and/or auto port aggregation.

Redundant network infrastructure and dynamic
routing.

Multiple PCs.

Constant, on-going monitoring using scripts,
SGManager, EMS, and/or OpenView products
(like Node Manager and OVO)

UPSs and diesel generators

Rigorous testing procedures; change control

Training, security policies, and written
procedures

Natural disasters (fire, flood, etc.)	Campus cluster, metro cluster, continental cluster
Poor performance (for some applications, poor performance is as unacceptable as downtime)	PRM, scripting to automatically move/stop applications, OVPA/OVPM, GlancePlus, and EMS

In general, the strategy for protecting against hardware failure is to build in redundant components. If the primary component fails, the redundant component automatically takes over. If the redundant component fails before the primary component is repaired, we have unplanned downtime. A failed primary component may go unnoticed without constant, ongoing monitoring.

5 Nines:5 Minutes is an HP strategy incorporating strategic alliances with Oracle, Cisco and others which for very specific environments would ensure 99.999% uptime or no more than five minutes downtime per year.

See <http://hp.com/go/ha> for more details.

- Auto port aggregation allows multiple ports (LAN cards) to be combined into a single link aggregate having a single IP address. This provides greater bandwidth, load balancing, and high availability.
- Campus cluster is based on Fibre Channel hubs.
- Continental cluster is based on database replication across a WAN.
- Metrocluster is based on EMC SRDF, XP Continuous Access, or Continuous Access EVA.

1–4. SLIDE: Not a Big Deal? You Tell Me!



Not a Big Deal? You Tell Me!

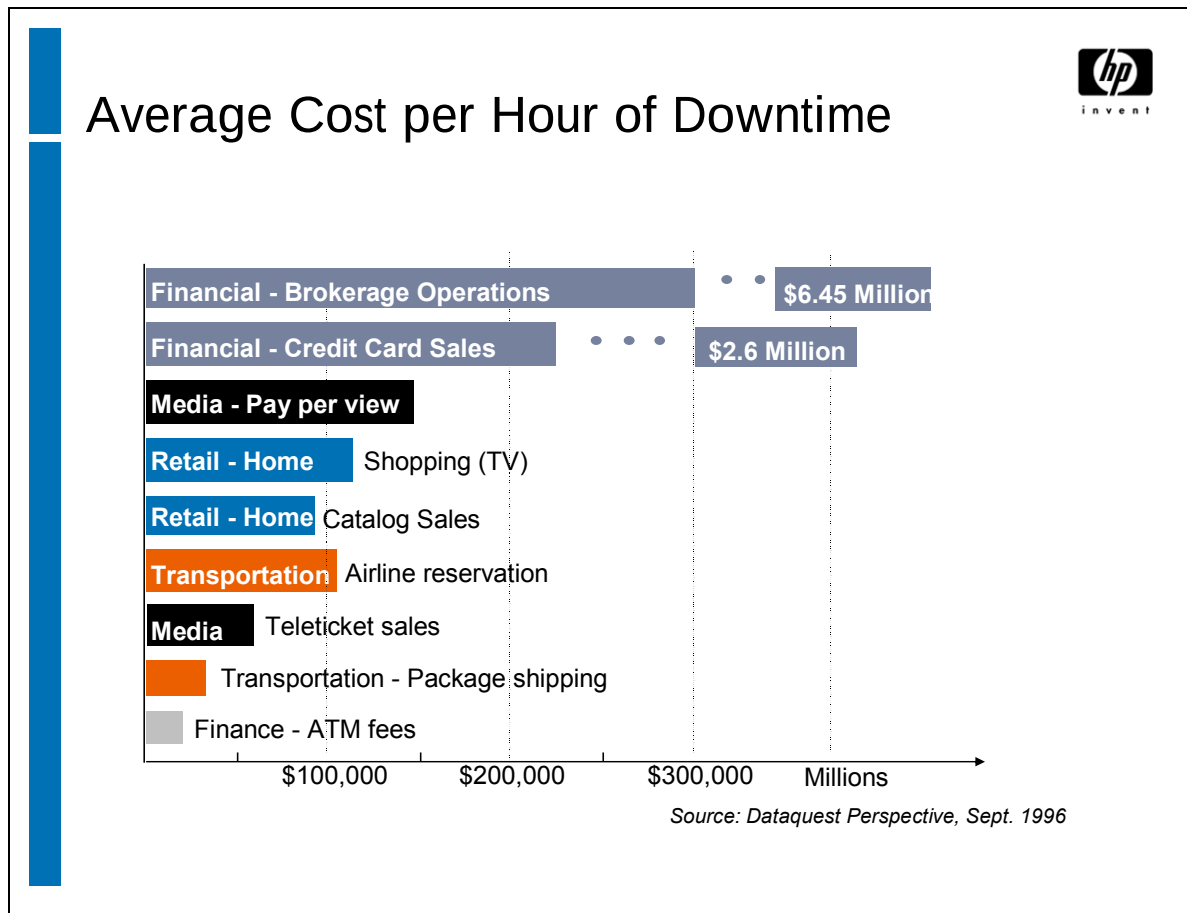


Student Notes

In today's business environment, mission critical applications simply must be available all the time — or the company will end up paying the consequences in lost production, customers, and revenue.

Businesses cannot afford downtime when the competition is just a click away with e-commerce. Downtime can generate unwanted headlines and significantly affect sales revenue, customer loyalty, and stock price.

1-5. SLIDE: Average Cost per Hour of Downtime



Student Notes

The cost of downtime is already high and quickly rising, making it easier than ever to justify investments in high availability (HA). By moving an application uptime from 99.90% uptime to 99.95% uptime, a company would eliminate nearly 4.5 hours of downtime per year. At the cost of downtime per hour as shown on the above slide, those four hours could save hundreds of thousands of dollars, depending on the company.

1–6. SLIDE: What Is High Availability?

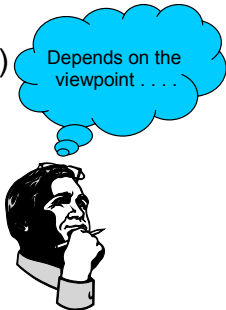


What Is High Availability?

A **system** is highly available if a single component or **resource** **failure interrupts** the system for only a **brief time**.

What is a system ?	(Computer? Network? Application?)
What is a resource ?	(Hardware? Software? OS? Database?)
What is a failure ?	(Disk crash? Too many packets? Full file system?)
What is an interruption ?	(Reboot? User reconnect? Poor performance?)
What is a brief time ?	(Minutes? Hours? Days?)

HIGH AVAILABILITY IS A DESIGN!



Student Notes

The above definition of high availability (HA) is very general and requires expansion of the terms: **system**, **resource**, **failure**, **interrupts**, and **brief time**. These definitions will vary, depending on the viewpoint. For example, system and resource tend to mean **hardware** to an administrator. To an application user, however, the **killing** of a database process is a failure resulting in loss of availability of a **software** resource.

The term **system**, in many cases, includes more than just the hardware. Depending on the “system” being kept highly available, it could include operating system resources, application processes, databases, and the ability of the users to access and connect to those resources.

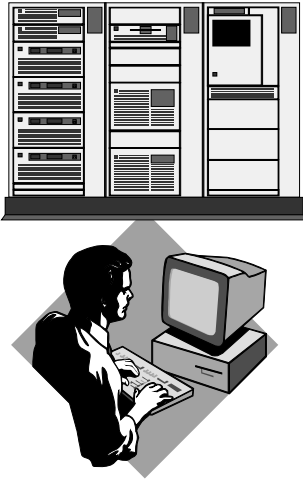
Some failures can be handled transparently without any interruption (for example, disk failure in a RAID array, single bit memory error). Other failures can result in complete loss of service, and a restart. The priority of an HA system is to minimize the duration of the interruption (initially minutes) and reduce it to zero for most types of failures.

The focus in a high availability environment is *time*. It is being able to recover from any single point of failure in as brief a period as possible.

1-7. SLIDE: Computer System Availability

Computer System Availability

System:	Computer
Resources:	CPU Memory Disk
Failures:	System crash Disk failure
Interruption:	System reboots Replace failed hardware
Outage Time:	Minutes to days



Student Notes

Computer system availability is the concern of the system administrator. The primary objective in *computer system availability* is ensuring the “computer” stays up without any failed hardware resources.

In looking at the HA terms as it relates to the *computer availability*, the above slide gives examples for each term:

- The “system” being kept highly available is the computer.
- The “resources” upon which a computer depends are predominantly hardware. These resources include CPUs, memory, disk drives, controller cards, power, etc. Failure of any of these resources would cause the computer to become unavailable.
- The “failures” include event that would cause the resources to become unavailable. Examples of possible failures include CPU failures, disk drive crash, double-bit memory parity errors, power outages, etc.

Introduction to High Availability

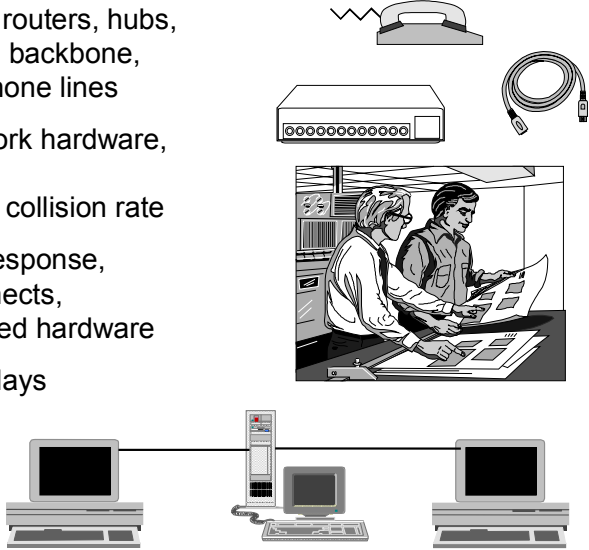
- The “interruptions” caused by the above failures almost always require a system reboot. In many cases, the interruption also includes the replacement of the failed hardware component.
- The “outage time” could be as short as minutes if the interruption is just a system reboot. Or it could take days if a field engineer needs to come on site to replace a failed hardware component.

1-8. SLIDE: Network Availability



Network Availability

System:	Network
Resources:	Computers, routers, hubs, LAN cables, backbone, Modems, phone lines
Failures:	Failed network hardware, Bad cables, High packet collision rate
Interruption:	Slow user response, User reconnects, Replace failed hardware
Outage Time:	Minutes to days



Student Notes

Network availability is the concern of the network administrator. The primary objective in *network availability* is ensuring computers and other network devices can communicate with each other.

In looking at the HA terms as they relate to *network availability*, the slide gives examples for each term:

- The “system” being kept highly available is the network.
- The “resources” upon which the network depends include routers, bridges, hubs, network interface cards (NICs), etc. Failure of any of these resources would cause the network to potentially become unavailable.
- The “failures” include any resource failure. These include disconnected cables, power surges and power outages, equipment being accidentally powered off, etc.
- The “interruptions” caused by the above failures almost always results in a temporary loss of node-to-node communications.

Introduction to High Availability

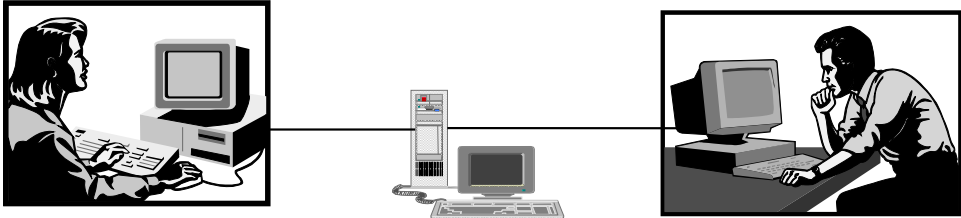
- The “outage time” could be as short as seconds if a secondary route existed between the two nodes. Or it could take hours if an engineer needs to come on site to fix/replace a network component and only a single route between two nodes is available.

1-9. SLIDE: Application Availability



Application Availability

System:	Application
Resources:	Computers, networks, operating system resources
Failures:	System crash, Network component failure, Full file system, performance paralysis
Interruption:	Slow response time, system reboots, Replace failed hardware
Outage Time:	Minutes to days



Student Notes

Application availability is the concern of the end user, the system administrator, and the network administrator. The primary objective in *application availability* is ensuring the end user has continuous access to the application (often a client/server network-based application).

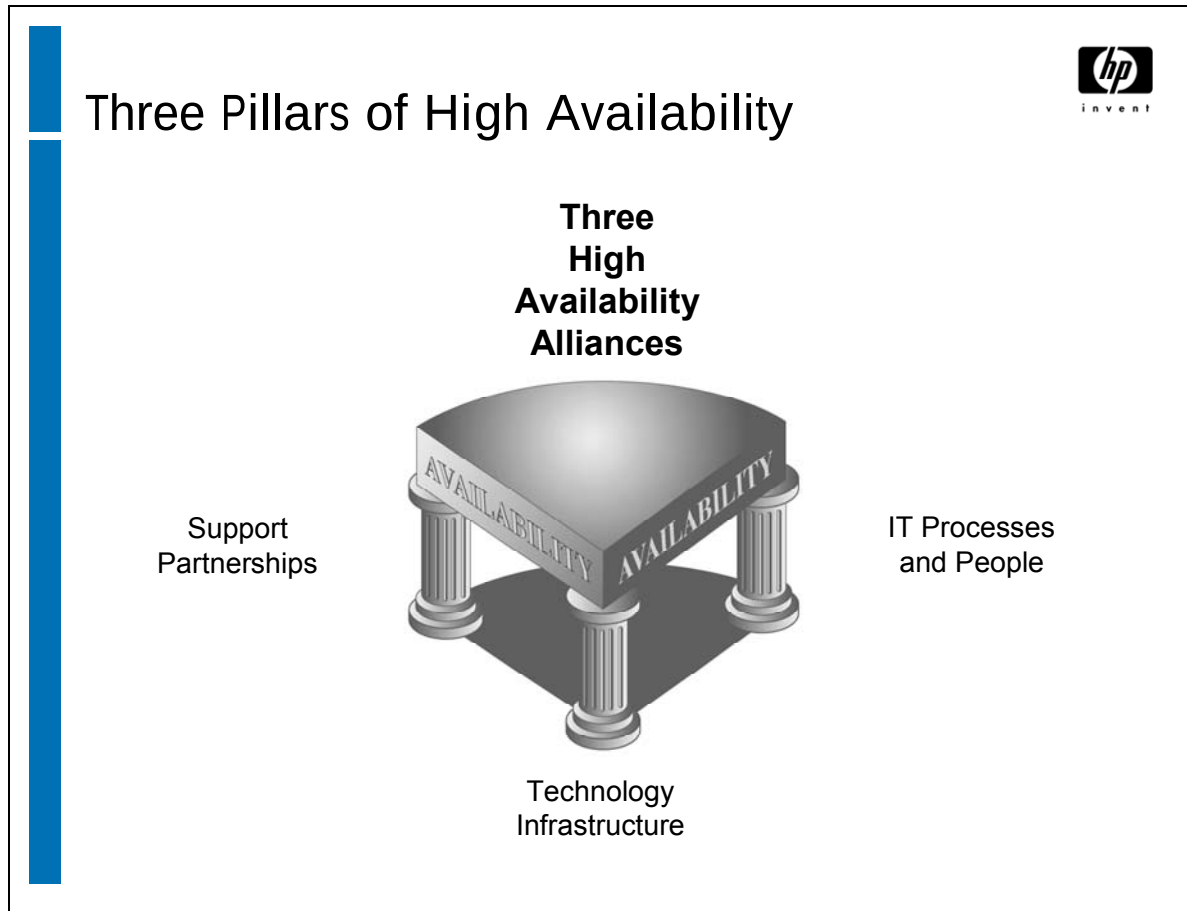
The slide gives examples of each HA term as it relates to *application availability*.

- The “system” being kept highly available is the application.
- The “resources” upon which the application is dependent are software and hardware resources. These resource computers, networks, OS and application resources. Failure of any of these resources would cause the application to potentially become unavailable to the end user.
- The “failures” include any resource failure that would prevent the end-user from properly accessing the application. These include any previously discussed system and/or network failures, or the loss of free disk space (as in a full filesystem) or the unplanned termination of an application process.

Introduction to High Availability

- The “interruptions” caused by the above failures almost always results in the application becoming unavailable to the end-user. In many cases, the interruption will require the end-user to reconnect to the application (i.e. log back in).
- The “outage time” could be as short as minutes or much longer, depending on the failure.

1-10. SLIDE: Three Pillars of High Availability



Student Notes

Recognizing that only a balanced and comprehensive approach will successfully keep businesses up and running, a solid HA strategy should be built on the following three pillars, which collectively address all the major areas of downtime.

- **Technology Infrastructure.** Both hardware and software products must be available to allow the recovery of any single component failure. Often, this is the pillar which is most focused on, but it is NOT the only pillar of high availability.
- **Support Partnerships.** No one vendor will be able to provide all the HA solutions for all companies. Some vendors will specialize in hardware HA solutions, other software HA solutions, and others network HA solutions. Being able to take advantage of multiple HA solutions concurrently from multiple vendors, and having the different solutions work cooperatively together, is a key pillar of high availability.
- **IT Processes.** Even when all the technology is configured correctly and all single-points-of-failure have been properly addressed, a company may still be unable to achieve their HA goals if the solutions and HA technology is not maintained properly. Recognizing that operator and user errors are the greatest cause of outages, without good IT processes (like documented procedures and training) high availability goals cannot be achieved.

1-11. SLIDE: 5nines Support Partnerships



Student Notes

The 5nines:5minutes initiative encompasses several industry leaders, including HP, BEA, Cisco, EMC, Oracle, and SAP, all working together to deliver premier high availability solutions. The 5nines:5minutes initiative addresses the entire IT solution, not just individual IT components, to deliver true end-to-end availability.

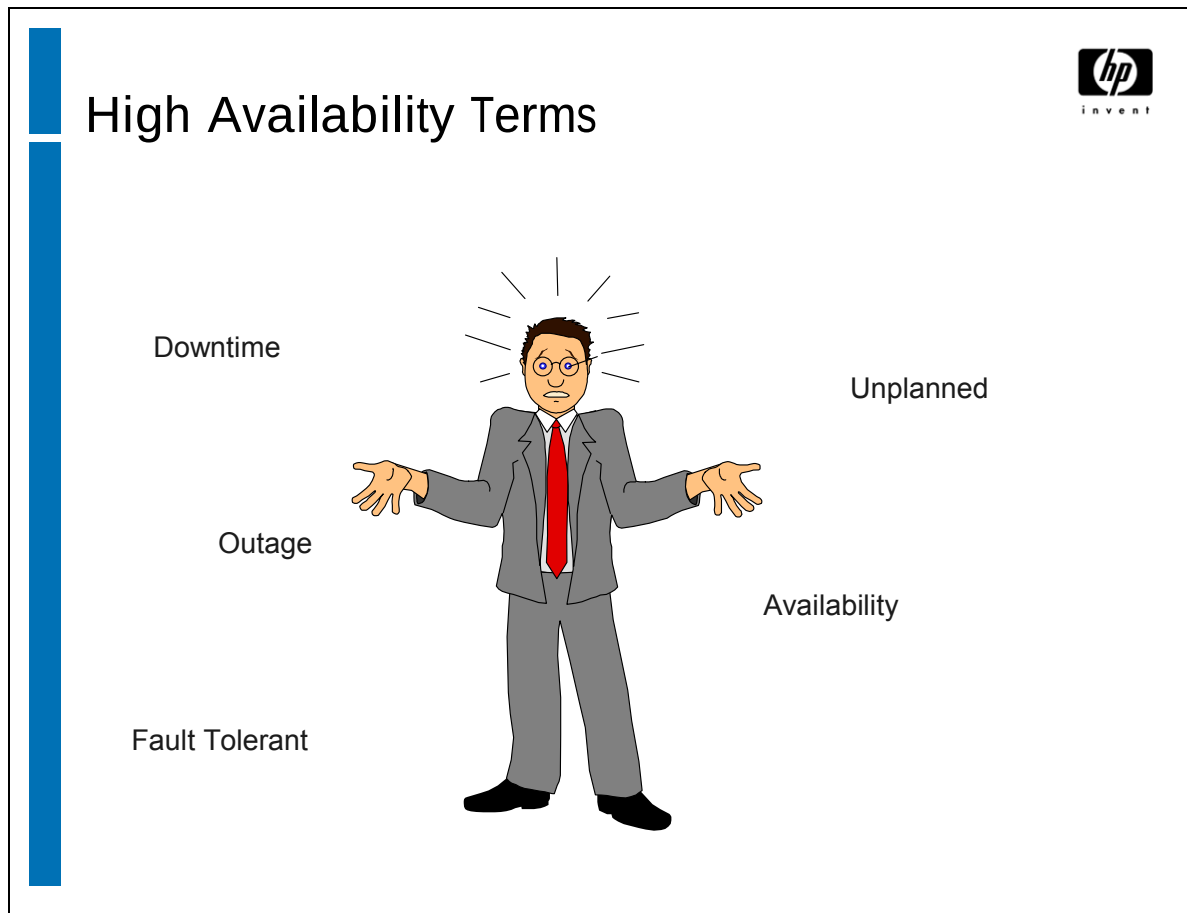
5nines:5minutes offers two solutions to address different customer needs. One, known as Mission Critical Server Suites (MCSS), delivers 99.95% availability, equating to only 4.5 hours of downtime per year. The other solution, for which 5nines:5minutes was named, will deliver 99.999% availability, or just 5 minutes of downtime per year.

All these solutions are for very specific applications and very specific configurations.

For more details on the 5nines:5minutes initiative, see:

<http://www.hp.com/products1/unix/highavailability>

1-12. SLIDE: High Availability Terms



Student Notes

High availability terms and definitions:

- | | |
|-----------|--|
| 24 x 7 | An environment that must have its systems up and running 24 hours a day, 7 day a week. This environment's systems cannot be down for any reason, including maintenance or backups. |
| 24 x 6.75 | An environment in which systems are down for 6 hours a week (Sunday morning, from midnight to 6:00 am, for example) to do general hardware or software maintenance.

In reality, most customers recognize there must be some downtime for these types of activities. |
| 16 x 5 | An environment in which systems are up 16 hours a day, 5 days a week (for example, Monday through Friday, 4:00 am to 8:00 pm). These systems may have lots of scheduled downtime. In the example cited, 8 hours a day during the week and all day on the weekends. |

Module 1
Introduction to High Availability

Downtime	Any amount of time when the application is unavailable (planned or unplanned). In the case of the 16 x 5 environment, there are 88 hours of downtime a week which is planned.
Planned	Downtime that is planned. This is time that the customer <i>plans</i> to bring down the system.
Unplanned	Downtime that is unplanned. This is time that the customer expects the application to be available, but it is not available due to an unplanned event or outage.
Outage	An occurrence that renders an application unavailable when it is expected to be available. An outage can occur because of hardware, software, user, or environmental problems.
Availability	The time that the application can be used during times when it is expected to be usable. Availability ignores planned or scheduled downtime and is expressed as a percentage.
Fault tolerant	A system that can tolerate any hardware failure and still keep the application available to the users. This does <i>not</i> mean that the system hardware never fails; it means the system has multiple hardware components that function concurrently, duplicating all of the computation, I/O, and so on. These systems protect against hardware failures by providing totally redundant hardware in a single system.
Highly available	A system that can recover quickly from all or most resource failures. The application may become unavailable, but only for a short period of time. Highly available systems typically contain data redundancy through mirroring or RAID, some kind of backup system hardware, a UPS for power outage protection, and multiple LANs.
Standard reliability	A system that relies only on basic hardware; there are no additional precautions taken to protect against an outage. Fault tolerant vendors claim that non-fault tolerant basic reliability results in an availability of 95 percent. HP's hardware typically provides availability in the 97–98 percent range without using any HA techniques.

1-13. SLIDE: High Availability Percentages



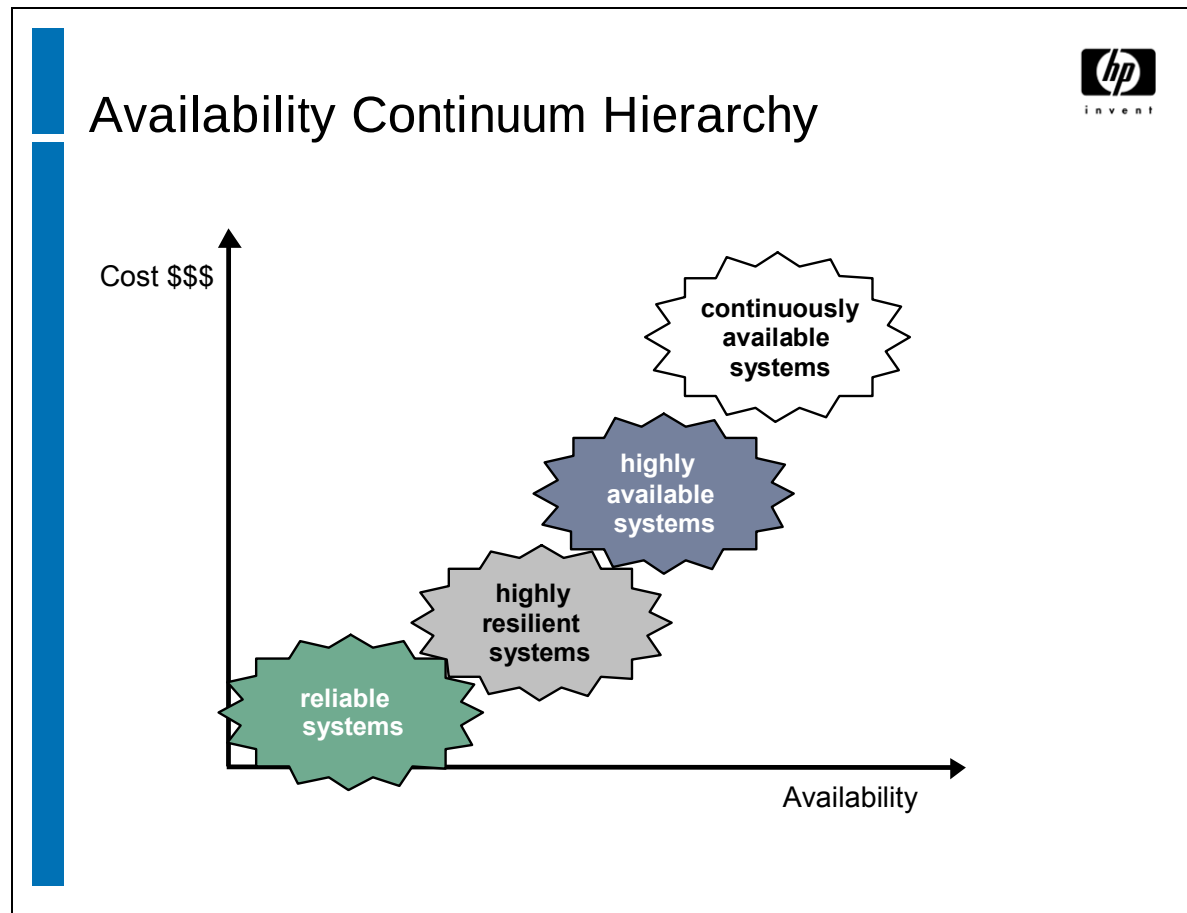
<u>Availability</u>	<u>Yearly Total Down Time</u>	<u>Type of System</u>
99.999	5 minutes	Fault Tolerant
99.99	50 minutes	
99.95	4.3 hours	Top High Availability
99.90	8.8 hours	
99.86	12 hours	Median High Avail
99.73	24 hours	
99.00	3.6 days	
98	7.2 days	HP Standard Avail
97	10.8 days	
96	14.4 days	
95	18 days	Most Standard Avail

Student Notes

The slide shows the amount of downtime corresponding to the amount of availability desired. The median for highly available systems is 99.86 percent or 12 hours of downtime per year.

HP's standard reliability (98 percent availability) has downtime of 7 days per year. The Serviceguard product helps to reduce downtime from *days* to *hours*.

1-14. SLIDE: Availability Continuum Hierarchy



Student Notes

The above slide shows there are a number of different levels of availability. The lowest level of availability, reliable systems, implies that the reliability of the hardware determines the availability of the system.

Reliable systems typically do not recover from many single-points-of-failure (SPOF). By providing additional technology and procedures, like mirroring data and/or adding a UPS, the systems can be upgraded to "highly resilient".

A "highly resilient" system can be made "highly available" by adding the ability to recover from any SPOF, like a failed CPU or a memory failure. The Serviceguard product is designed to accomplish this objective.

Finally, "continuously available" systems allow for recovery from multiple-points-of-failure, like an earthquake destroying an entire data center. An example of a "continually available" system is HP's Continental Clusters product.

1–15. SLIDE: For More Information



Additional Resources

- <http://docs.hp.com/en/ha.html>
- <http://www.hp.com/go/ha>
- <http://www.hp.com/products1/unix/highavailability>
- <http://www.itrc.hp.com>

Student Notes

The following websites can provide further information on high availability.

<http://docs.hp.com/hpux/ha>

<http://hp.com/go/ha>

<http://www.hp.com/products1/unix/highavailability>

<http://www.itrc.hp.com>

1-16. SLIDE: Course Agenda



Course Agenda

Day 1:

- Introduction to High Availability
- Introduction to Serviceguard
- High Availability Planning
- LVM for Serviceguard

Day 2:

- Cluster Concepts and Configuration
- Additional Cluster Features
- Packages and Services

Day 3:

- Package Policies
- Application Monitoring Scripts
- Best Practices
- Cluster and Package Online Reconfiguration
- Highly Available NFS

Day 4:

- The Highly Available Oracle Database
- EMS Resources and Serviceguard Packages
- High Availability Networking
- Rolling Upgrade Issues

Day 5:

- LVM Maintenance for Packages
- Serviceguard Manager
- Troubleshooting Scenarios and Final Review

Student Notes

1-17. LAB: Prepare your Systems for Later Modules: Configure IP Addresses

Directions

This lab will prepare us for later modules. We will configure an additional IP address for each system being used in your training class. This IP address will be used in later modules when we configure our Serviceguard cluster. Additional IP addresses will also be used in later modules when we configure Serviceguard packages.

Your success in future modules will depend directly on your success in this lab!

Preliminary Steps

1. Just in case something goes wrong during this lab, make a backup copy of all of your network configuration files. There is a shell script in your `/labs` directory designed specifically for this purpose. The shell script will save a tar archive backup of your network configuration files in the file you specify. Add the `-l` option to verify your backup.

Execute the following commands on both systems:

```
# cd /labs
# ./netfiles.sh -s ORIGINAL
# ./netfiles.sh -l           ← This is the letter “ell”, not the number “one”
# ./netfiles.sh -l ORIGINAL ← This is the letter “ell”, not the number “one”
```

Part 1: Checking the Current LAN Card Configuration

1. **On both nodes**, check the current configuration of the LAN cards. How many LAN cards does your system have?

```
# lanscan
```

Part 2: Configuring the New LAN Card Configuration

The goal of this portion of the lab exercise is to configure an additional IP address for each of the systems we will be using.

1. What IP addresses are currently assigned to your systems? Write down those values in the table below. In most cases, the `lan0` interface is unconfigured, and will remain so.

In many cases, the `lan0` interface is either unconfigured or used for an internal network. In many cases, the `lan1` interface will be configured with an IP address on the 10-net. Additional interfaces (eg. `lan2`, `lan3`, `lan4`, ...) are available for our use.

Remember that the following chart should contain only the IP addresses currently configured. There will likely be a few blank rows in your chart once you have finished this step.

```
# netstat -in
```

Just mark "none" for non-existent lan cards, and mark "nc" for lan cards not currently configured.

Lan Interface	First system	Second system
lan0		
lan1		
lan2		
lan3		
lan4		
lan5		

2. Next, add another IP address to the existing network configuration, using the next-available interface card. (NOTE: Even if `lan0` is unconfigured, do not use this card.)

Ask your instructor for the appropriate IP address to use and fill in the same chart as you used previously, but this time, also write in the new IP addresses you will be using.

Lan Interface	First system	Second system
lan0		
lan1		
lan2		
lan3		
lan4		
lan5		

3. **On both nodes**, edit the `/etc/rc.config.d/netconf` file to configure this IP address. NOTE: If `lan0` is unconfigured, follow the steps below. If `lan0` is already configured, skip to the “IF `lan0` IS CONFIGURED” section on the next page. Be sure to check the entire `/etc/rc.config.d/netconf` file for `lan` entries before proceeding.

IF `lan0` IS UNCONFIGURED

Locate the appropriate section below and make the suggested changes.

```
INTERFACE_NAME[0]="lan1"
IP_ADDRESS[0]="10.10.x.x"
SUBNET_MASK[0]="255.255.0.0"
BROADCAST_ADDRESS[0]=""
INTERFACE_STATE[0]=""
DHCP_ENABLE[0]="0"
INTERFACE_MODULES[0]=""
```

Copy these seven lines to a spot immediately below these seven lines. After copying, change the index value from “[0]” to “[1]” for each of the seven lines. Also, make the following changes ...

- Change the `INTERFACE_NAME[1]` entry from its current value to the next available `lan` card interface (eg. “`lan2`”).
- Change the `IP_ADDRESS[1]` entry from its current value to the new IP address on the 10-net, as was obtained in step 2 above.
- Change the `SUBNET_MASK[1]` entry to `255.255.255.0`.
- Leave the `BROADCAST_ADDRESS[1]=""` entry undefined.
- Leave the `INTERFACE_STATE[1]=""` entry undefined.
- Leave the `DHCP_ENABLE[1]="0"` entry as “0”.
- Leave the `INTERFACE_MODULES[1]=""` entry undefined.

The final version of your `/etc/rc.config.d/netconf` file might then look like:

```
INTERFACE_NAME[0]="lan1"
IP_ADDRESS[0]="10.10.x.x"
SUBNET_MASK[0]="255.255.0.0"
BROADCAST_ADDRESS[0]=""
INTERFACE_STATE[0]=""
DHCP_ENABLE[0]="0"
INTERFACE_MODULES[0]=""
```



```
INTERFACE_NAME[1]="lan2"
IP_ADDRESS[1]="10.10.y.y"
SUBNET_MASK[1]="255.255.255.0"
BROADCAST_ADDRESS[1]=""
INTERFACE_STATE[1]=""
DHCP_ENABLE[1]="0"
INTERFACE_MODULES[1]=""
```

```
# vi /etc/rc.config.d/netconf
```

IF lan0 IS CONFIGURED

Locate the appropriate section below and make the suggested changes.

```
INTERFACE_NAME[1]="lan1"
IP_ADDRESS[1]="10.10.x.x"
SUBNET_MASK[1]="255.255.0.0"
BROADCAST_ADDRESS[1]=""
INTERFACE_STATE[1]=""
DHCP_ENABLE[1]="0"
INTERFACE_MODULES[1]=""
```

Copy these seven lines to a spot immediately below these seven lines. After copying, change the index value from "[1]" to "[2]" for each of the seven lines. Also, make the following changes ...

- Change the INTERFACE_NAME[2] entry from its current value to the next available lan card interface (eg. "lan2").
- Change the IP_ADDRESS[2] entry from its current value to the new IP address on the 10-net, as was obtained in step 2 above.
- Change the SUBNET_MASK[2] entry to 255.255.255.0.
- Leave the BROADCAST_ADDRESS[2]="" entry undefined.
- Leave the INTERFACE_STATE[2]="" entry undefined.
- Leave the DHCP_ENABLE[2]="0" entry as "0".
- Leave the INTERFACE_MODULES[2]="" entry undefined.

The final version of your /etc/rc.config.d/netconf file might then look like:

```
INTERFACE_NAME[0]="lan0"
IP_ADDRESS[0]="192.168.x.x"
SUBNET_MASK[0]="255.255.0.0"
BROADCAST_ADDRESS[0]=""
INTERFACE_STATE[0]=""
DHCP_ENABLE[0]="0"
INTERFACE_MODULES[0]=""

INTERFACE_NAME[1]="lan1"
IP_ADDRESS[1]="10.10.x.x"
SUBNET_MASK[1]="255.255.0.0"
BROADCAST_ADDRESS[1]=""
INTERFACE_STATE[1]=""
DHCP_ENABLE[1]="0"
INTERFACE_MODULES[1]=""

INTERFACE_NAME[2]="lan2"
IP_ADDRESS[2]="10.10.y.y"
SUBNET_MASK[2]="255.255.255.0"
BROADCAST_ADDRESS[2]=""
INTERFACE_STATE[2]=""
DHCP_ENABLE[2]="0"
INTERFACE_MODULES[2]=""
```

```
# vi /etc/rc.config.d/netconf
```

4. **On both nodes**, activate your changes.

Special Note

- If you are using remote equipment, then access your remote equipment via the Management Processor. Once logged into the Management Processor, connect to the Unix system or partition to which you have been assigned. From there, execute the following two steps to activate your changes.
- If you are using local equipment, then immediately execute the following two steps to activate your changes.

```
# /sbin/init.d/net stop  
# /sbin/init.d/net start
```

If you receive the following error, ignore it.

ERROR: Failed to add route entry because interface is not yet initialized. May need to add route entry with route command after the interface is up:
Add net default: gateway 10.99.x.x: Network is unreachable

5. **On both nodes**, verify that your new IP address was set properly? How can you find out?

```
# netstat -in
```

6. **On both nodes**, modify your `.rhosts` files as follows:

- Do not remove any entries from your `.rhosts` file.
- To establish trust between the nodes of your cluster, add the hostname of the “other” node in your cluster.
- Ask your instructor for the hostname of the class Quorum Service machine, and add this hostname to your `.rhosts` files if it is not already there.

Add these hostnames to your `.rhosts` files.

```
# vi ~/.rhosts
```

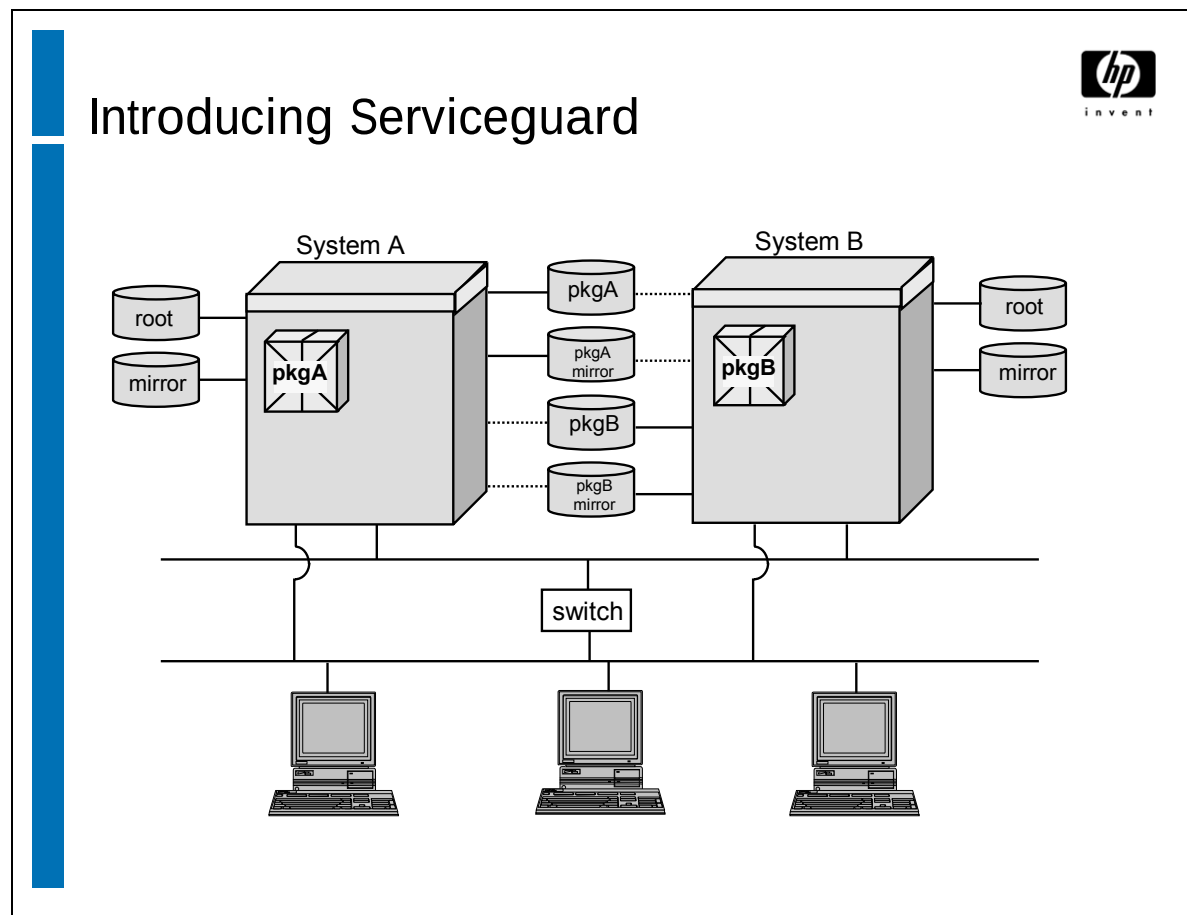
Module 2 — Introduction to Serviceguard

Objectives

Upon completion of this module, you will be able to do the following:

- Describe three features and benefits of Serviceguard.
- Describe on a high level how Serviceguard works.
- List the components of a Serviceguard package.

2-1. SLIDE: Introducing Serviceguard



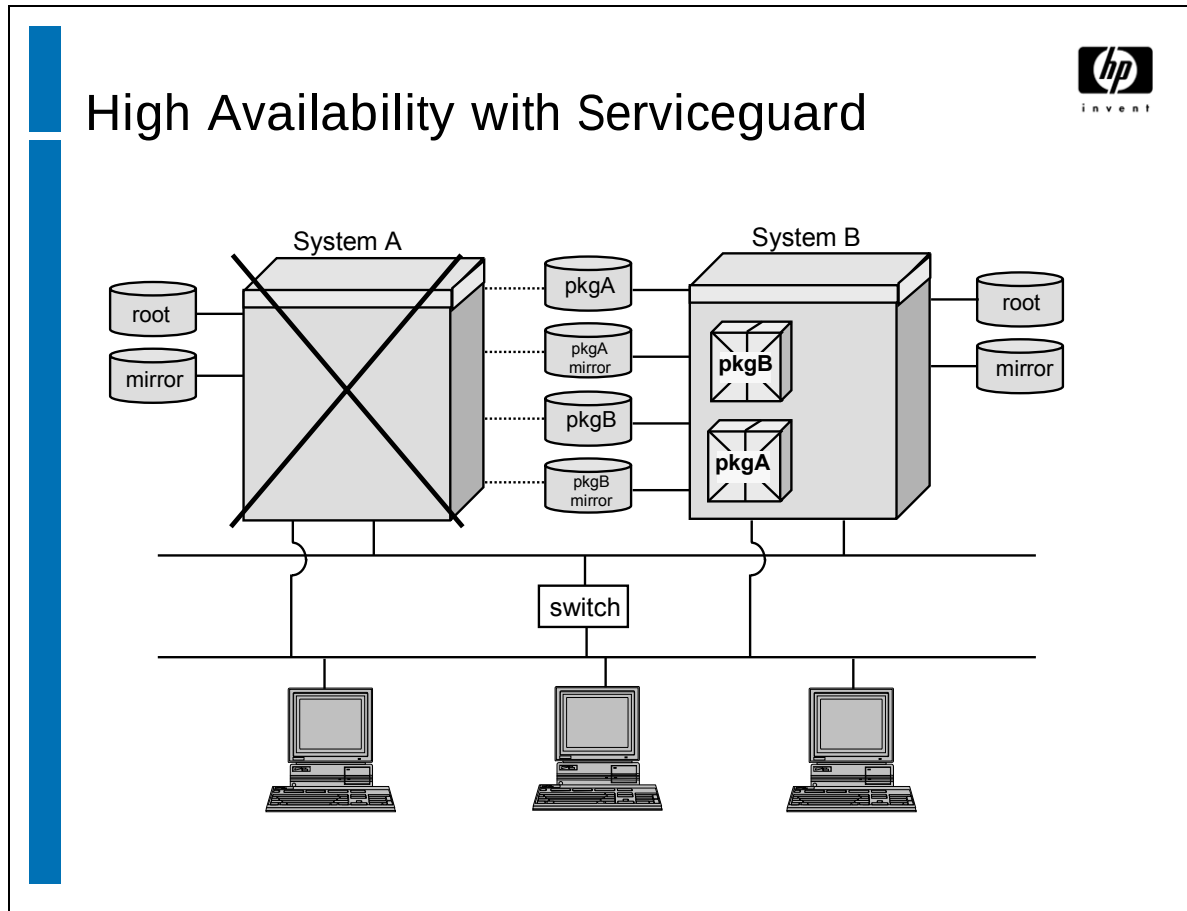
Student Notes

Serviceguard allows you to create **high availability** clusters of HP server systems. A highly available Serviceguard environment allows applications to continue in spite of a hardware or software failure. Highly available systems protect users from software failures as well as failures from hardware failures, like failures of CPU, disk, or network components. In the event that one component fails, the redundant component takes over.

A Serviceguard cluster is a networked grouping of HP business servers having sufficient redundancy of software and hardware that a single-point-of-failure will not significantly disrupt service. Applications resources are grouped together in **packages**, and in the event of a failure of an application/package resource, Serviceguard can automatically transfer control of the application to another node within the cluster.

In the above slide, SystemA is running PackageA and SystemB is running PackageB. Each package has a separate group of disks associated with it, containing data needed by the package's applications, and a mirror copy of the data. Note that both nodes are physically connected to both groups of mirrored disks. In the slide, SystemA is shown with exclusive write access to the top disks (solid lines) and System B is shown as connected without access to the top disks (dotted lines). Similarly, SystemB has exclusively access to the bottom disks and SystemA is connected, but does not have access to the bottom disks.

2-2. SLIDE: High Availability with Serviceguard



Student Notes

Under normal conditions, Serviceguard simply monitors the health of the cluster and package components. When Serviceguard detects a node or package failure, it will transfer control of the package to the next available node in the cluster. This is the situation shown on the above slide.

Once a package is transferred to a second node, it remains there as long as the second node continues running (by default). However, the package can be configured to automatically return to its primary node as soon as the primary node comes back online. Or, the system administrator can manually transfer the package back to the primary node at any appropriate time.

Serviceguard is designed to work with other high availability (HA) products to provide even higher levels of availability. Some of the other HA products include:

- **MirrorDisk/UX:** eliminates SPOF of the disk subsystem
- **Event Monitoring Service (EMS):** allows monitor and detection of non-Serviceguard resources
- **Disk Arrays:** provides RAID level data protection
- **UPSs:** eliminates failures related to power outages

2-3. SLIDE: Features and Benefits of Serviceguard



Features and Benefits of Serviceguard

Features	Benefits
• Highly Available Cluster (applications recover to alternate nodes in < 60 seconds -- often in < 10 seconds) • LAN failure protection (fast local switch to standby LAN adapter inside same node) • Application Packages allow all resources for a package to be defined in one place • Automatic cluster reconfiguration after a node failure • Intelligent cluster reconfiguration after a node failure • No idle resources • Facilitates online hardware and software updates • Any mix of PA-RISC and / or Integrity Servers	• Applications remain available to users, even after a hardware or software failure • LAN card failures do not cause an application outage • Applications can be moved easily and transparently without client reconfiguration • No manual user intervention is needed to recover from a node failure • Data integrity is preserved during a node failure (i.e. no "split-brain" syndromes) • Every node runs a production application • Applications available during hardware and software upgrades • Flexible load balancing during failover

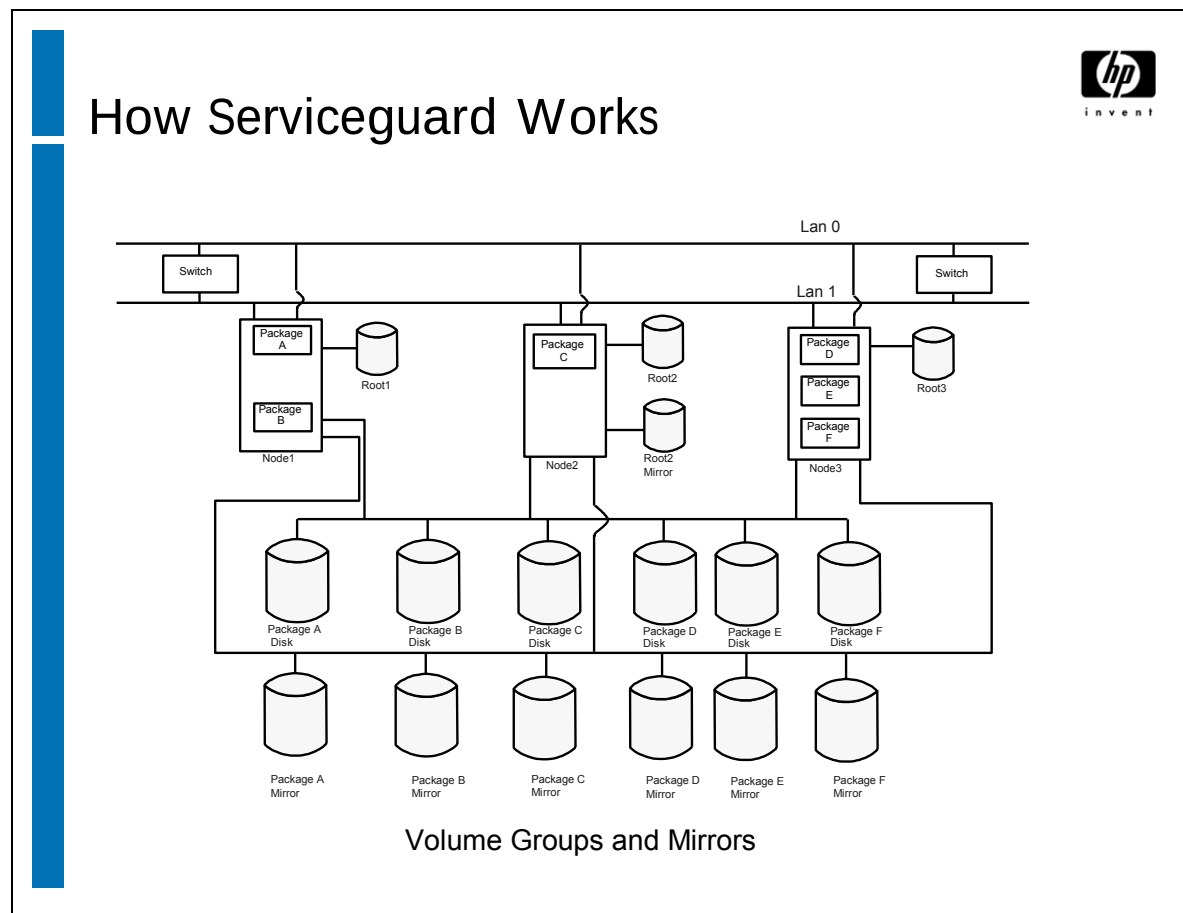
Student Notes

Features and benefits of Serviceguard include the following:

- *High cluster availability:* Potentially, any node in the cluster can run the shared applications. If a node goes down, all shared applications on the node will move to the other available nodes within the cluster.
- *LAN failure protection:* Every LAN interface card (LANIC) is capable of having a corresponding standby LANIC. If a primary LANIC fails, all IP addresses configured on that card would move to the standby card.
- *Application transparency:* Clients (workstations) will not be able to tell on which node an application is running.
- *Fast application switching:* The amount of time between when an application initially fails and when the application is restarted on another node is under 60 seconds. The Journal File System (JFS) is highly recommended in order to optimize the failover time by reducing the amount of time needed to repair the file system.

- *Easy and flexible application management:* Applications can be easily stopped and restarted on nodes within the cluster. Tools to monitor the state of the cluster and applications within the cluster exist in both a GUI and character-based mode.
- *Intelligent reconfiguration:* In the event of a failure, the cluster will automatically reconfigure itself to make the application available. No user interaction is required for reconfiguration.
- *Flexible load balancing:* Upon node failure, the workload on the failed node can be distributed across surviving nodes in the cluster. For example, if a node had three applications and the node failed, the applications could be spread across three different systems within the cluster.
- *No idle resources:* Every system is a production system, every system within the cluster can be running mission critical applications. No system sits idly, waiting for another system to fail.
- *Online software update:* Systems within the cluster can now have their software updated without having their applications unavailable to the users. The applications could run on alternate nodes while the system is being software upgraded.
- *Any mix of PA-RISC and / or Integrity Servers:* A cluster can have a mix of business server types, both PA-RISC and Integrity. For example, one or more partitions from a large Superdome system could be in the same cluster as an rp8400.

2-4. SLIDE: How Serviceguard Works



Student Notes

A **Serviceguard cluster** is a networked group of nodes (hosts), which monitor each other, in order to ensure that interruptions to the availability of selected services running on these nodes are kept small.

The state of any node or network interface within a cluster can be monitored from any node within the cluster. For example, if `/dev/lan0` has failed on node 1, that information will be obtained by each node within the cluster. The Serviceguard daemon monitors the health of the cluster, all nodes in the cluster, and all network resources.

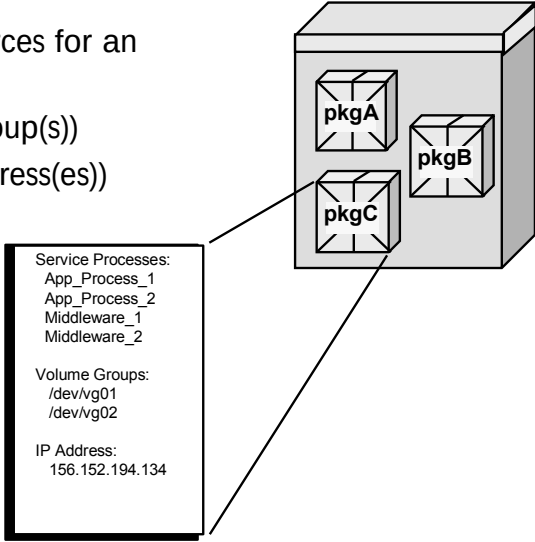
A cluster may contain up to 16 nodes. The original node is the first node on which a package is running before Serviceguard initiates a fail-over. The adoptive node is a node to which Serviceguard could transfer control of a package. A package can have several adoptive nodes. The node on which a package is currently running is said to have package custody.

2-5. SLIDE: Serviceguard Packages



Serviceguard Packages

- Simple, flexible
- Packages group all the resources for an applications, including:
 - Disk resources (volume group(s))
 - Network resources (IP address(es))
 - Service process
- Packages are the atomic entities that are managed and moved within a Serviceguard cluster environment



The diagram illustrates a Serviceguard package structure. A large box represents the package, containing three smaller boxes labeled pkgA, pkgB, and pkgC. To the left of this box is a configuration box with the following text:

```
Service Processes:
App_Process_1
App_Process_2
Middleware_1
Middleware_2

Volume Groups:
/dev/vg01
/dev/vg02

IP Address:
156.152.194.134
```

Lines connect the configuration box to the main package box and its sub-packages.

Student Notes

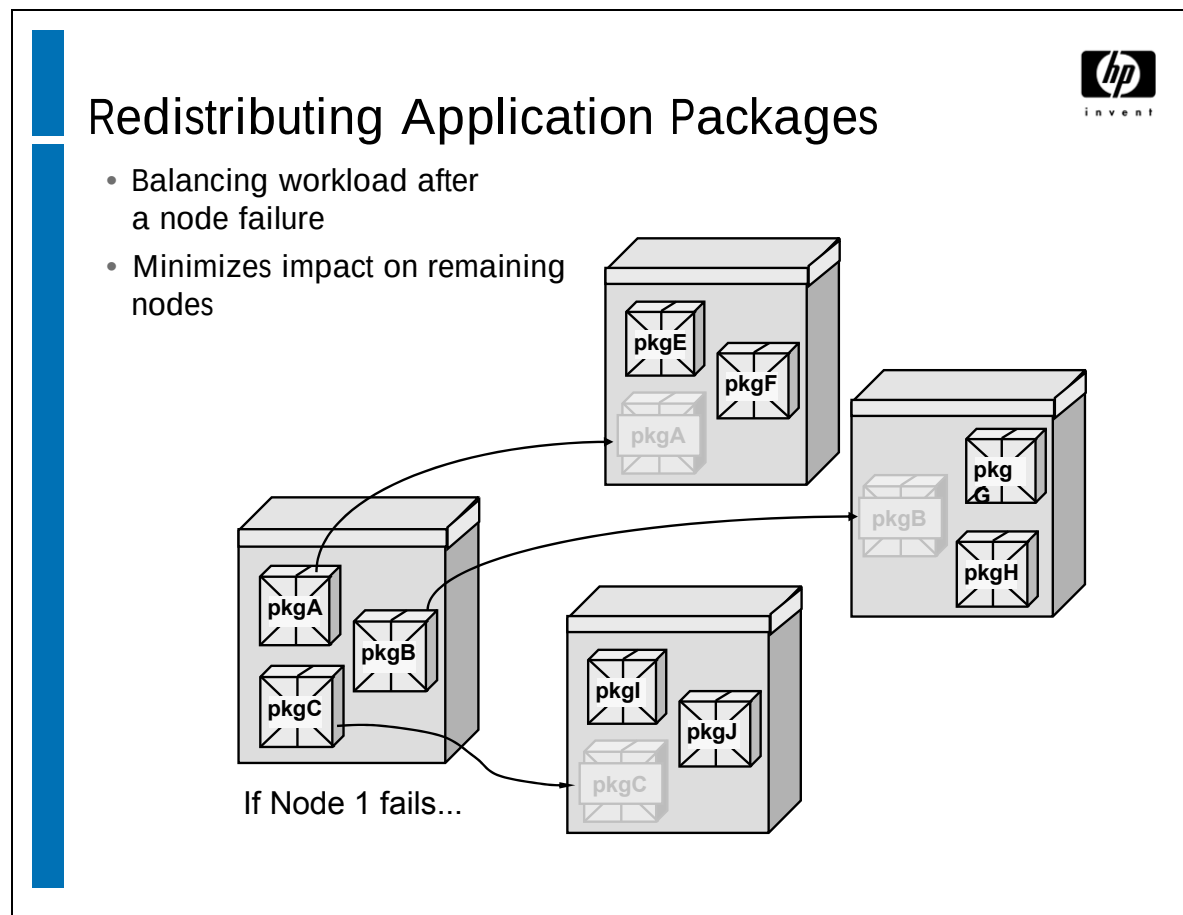
A Serviceguard package defines an application along with its programs and resources. It is dependent on such resources as disks, network addresses, files, and services that may be related to an application. All of these resources would move together to another system in the event of a failure on the original system or network. The package configuration file lists all the nodes that the package is able to run on within the cluster. Packages provide the mechanism for switching a highly available service (and all it is dependent on) as a unit between nodes in response to failures. If a node has multiple packages running on it, the separate packages can be switched to separate nodes, spreading the load of the original node.

No more than one package can depend on a volume group. You can specify if you want control of a package to be transferred or if you want the package to be halted in the event of a failure. The adoptive SPU must have access to all files and services needed to run the application.

Packages, therefore,

- may use disk resources,
- may use network resources,
- must include at least one process, and commonly include many processes.

2-6. SLIDE: Redistributing Application Packages



Student Notes

In the event of a node failure, an application package will be adopted by another node in the cluster. Later, the failed node is able to rejoin the cluster. Application packages can remain running on the adoptive node or can be moved back (by the Serviceguard administrator using either the command line or Serviceguard Manager) to the original node after the original node is back on line.

2-7. SLIDE: Minimizing Planned Downtime

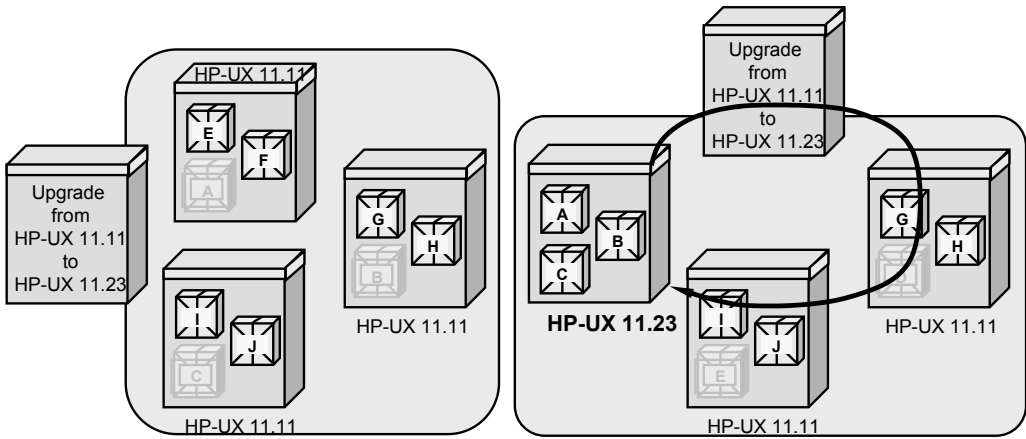


Minimizing Planned Downtime

- Move packages to alternate nodes.
- Perform hardware or software upgrades, maintenance.
- Move packages back.

Backward compatibility:

- OS
- Serviceguard



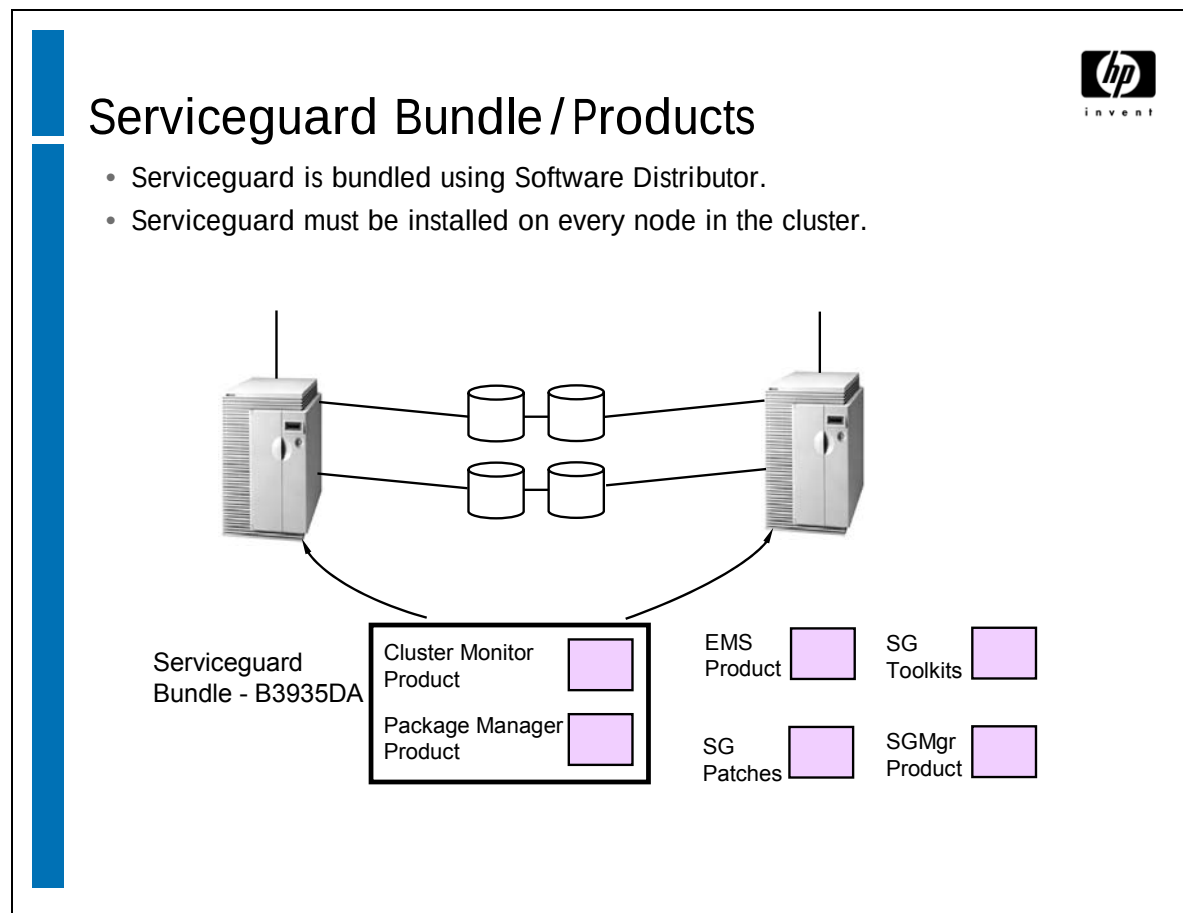
Student Notes

We will spend minimal time in this discussion right now. In a later module, we will discuss this “rolling upgrade” technique in detail.

For now, suffice it to say that we do the following to minimize planned downtime with Serviceguard:

1. Move packages to alternate nodes.
2. Perform hardware or software upgrade/maintenance.
3. Move packages back and roll through cluster.

2-8. SLIDE: Serviceguard Bundle/Products



Student Notes

The Serviceguard product is located on the HP-UX 11.00 and 11i Application media.

The core Serviceguard products are bundled together and are automatically installed when the administrator installs the HP-UX 11i Mission Critical Operating Environment. The core products are Cluster Monitor and Package Manager. Additional software can supplement these products, including the EMS product, various Toolkits, and related Serviceguard patches.

In addition, a standard Serviceguard installation should also include:

- OnLineJFS
- LVM MirrorDisk/UX
- PRM (Product Resource Manager) for managing system performance
- EMS (Event Monitoring Service)
- Serviceguard Manager

All these products (Serviceguard and all its supplemental products) must be installed on each system in the cluster.

2-9. LAB: Installation of Serviceguard Software

Directions

Special Note

When these systems were installed for our Serviceguard class, HP-UX 11i, including the "Mission Critical Operating Environment", was installed. This Operating Environment includes the "OnLine JFS" software and the "MirrorDisk/UX" software. Of course, it also includes the Serviceguard software itself. This Operating Environment also includes most (but not all) of the additional software needed for this class. This lab asks you to install the Serviceguard Manager bundle, as well.

1. List all existing software products on the system. Verify that Serviceguard is installed.

Answer:

```
# swlist -l product | grep -i guard
```

2. Run the Software Distributor `swinstall` command.

Answer:

```
# swinstall
```

3. At the Specify Source window, accept the default host name and default depot name, which is `/var/spool/sw`.

4. At the Software Selection window, select the following products:

- B8325BA Serviceguard Manager (version A.05.00)

Module 2
Introduction to Serviceguard

5. Once the software has been selected, go to the Action menu and select Install.
6. When the analysis phase completes, the status of the installation will change to READY. Select OK to continue with the installation.
7. When the installation phase completes, the status of the installation will change to COMPLETE. Select Done to exit the window.
8. Select EXIT from the File menu to exit Software Distributor.
9. Verify that the above software bundle has indeed been properly installed.

Answer:

```
# swlist -l bundle | grep -i guard
```

← For Serviceguard Manager

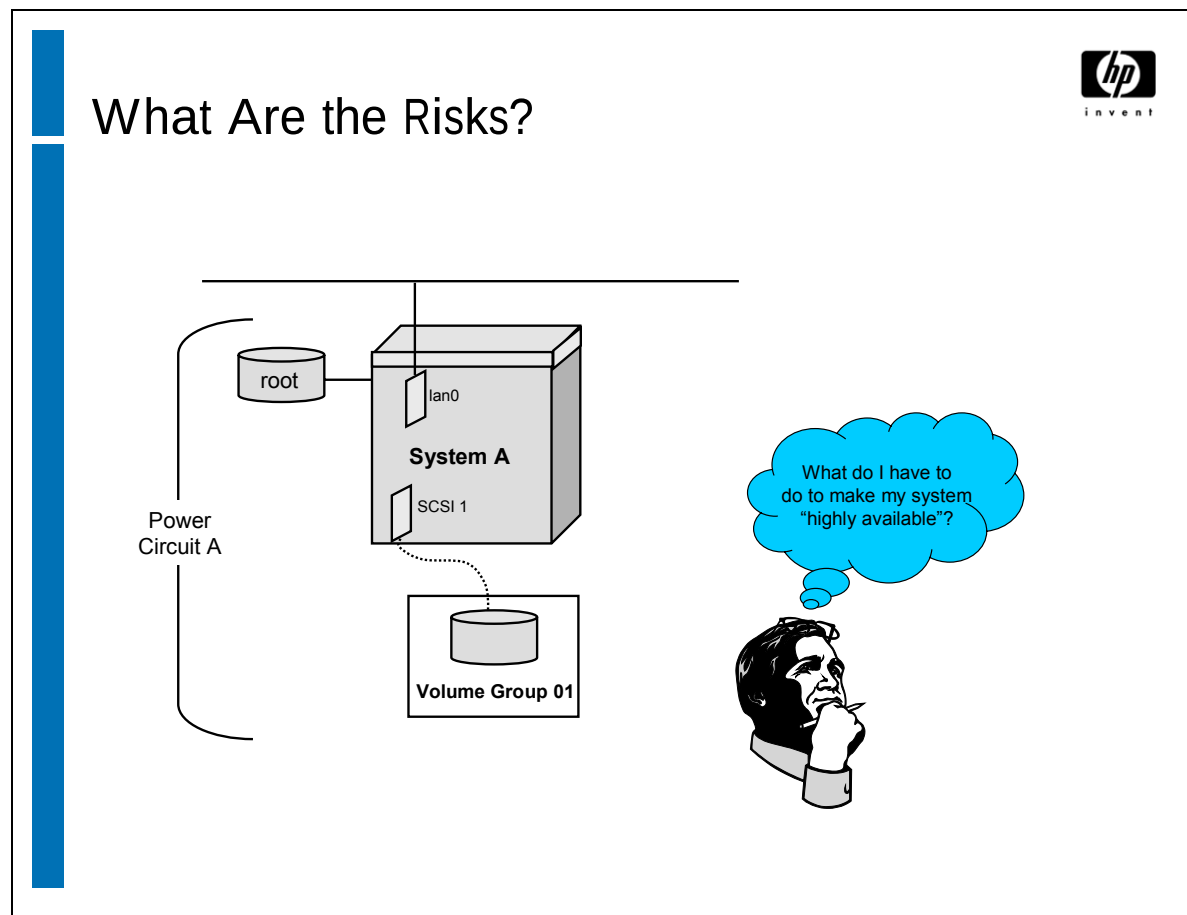
Module 3 — High Availability Planning

Objectives

Upon completion of this module, you will be able to do the following:

- Identify specific actions that can be taken to improve high availability (HA).
- Identify multiple risks (SPOF's) and list options for reducing the risks.
- Identify different disk technology for a high availability environment.
- Identify different network configuration for a high availability environment.
- Mirror your root volume group.

3-1. SLIDE: What Are the Risks?



Student Notes

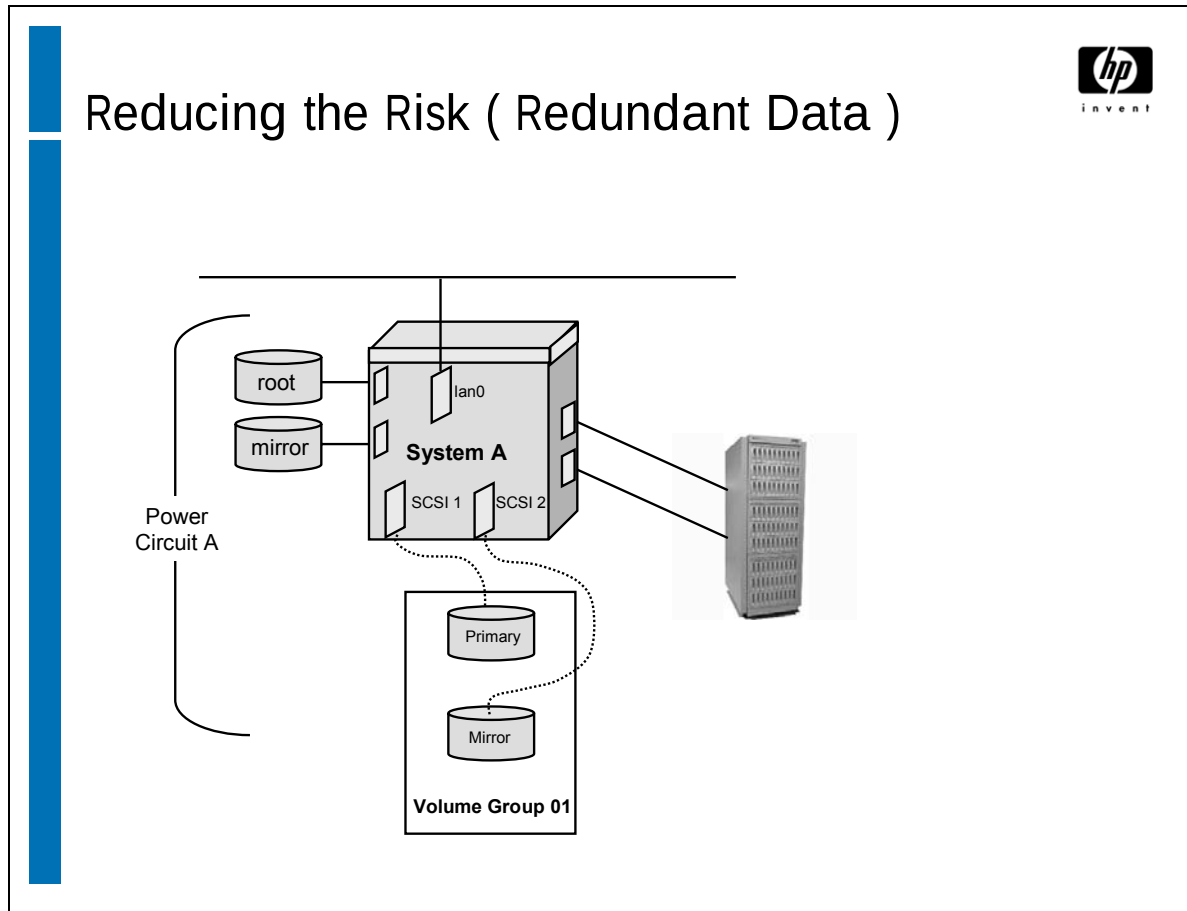
The simple drawing on the above slide illustrates several common Single-Points-Of-Failure (SPOFs). A common perception of SPOFs is that they are only hardware related. Items often listed as SPOF include:

- Disk drives
- Controller cards
- LAN cards and cables
- CPU
- Power supply

But, SPOFs can also include bugs in the operating system, poorly set kernel parameters, and/or unknowledgeable system administration personnel. Other aspects to consider are not only the availability of the system or application(s) but response times and performance. The performance of a server may be poor because of heavy demand or bad configuration. From the viewpoint of end-users and client applications, it might as well be unavailable.

HA technologies and plans that effectively use these technologies can prevent these situations. HA design should consider the possible management and control potential, as well as the *reactive to failure* elements.

3-2. SLIDE: Reducing the Risk (Redundant Data)



Student Notes

The first HA consideration is usually related to *data* and disk drives. A design incorporating some form of hardware RAID technology or software mirroring is common. We will compare the features and benefits of these two approaches later.

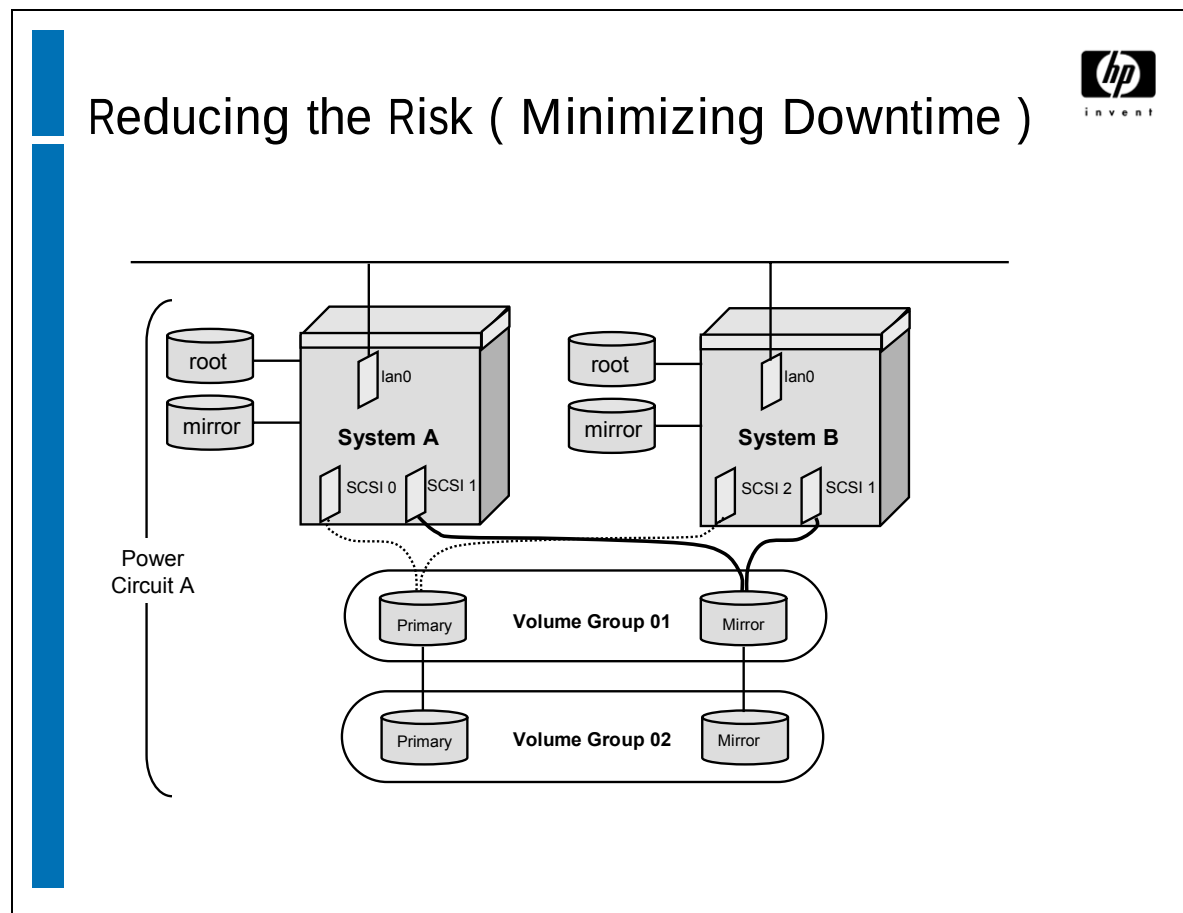
From both an HA and performance perspective, the design should include multiple controller paths and cables. LVM PV links are only used in the event of a primary path timeout, and do not load share. Good RAID array design and configuration will utilize all controllers, each acting as PV link paths for the other, resulting in HA and load sharing.

What type of RAID array and level to use cannot be taken in isolation. We must consider other technologies. (Serviceguard and Serviceguard SGeRAC edition, for example, can require specific *lock disk* types under certain circumstances.)

Other pertinent questions which should be asked relative to protecting data:

- Is online backup a requirement?
- How easy is it to replace a failed disk drive?
- Is it necessary to mix RAID levels, and tune for performance?

3-3. SLIDE: Reducing the Risk (Minimizing Downtime)



Student Notes

If there is only one computer, then there is a SPOF. Utilizing Serviceguard (or, in some circumstances, Serviceguard SGeRAC), an HA cluster can be designed to allow other systems to provide a support for the applications and services. These additional nodes will require access to the disk drives and networks. The HA software *monitors* the health of the nodes and services, and takes appropriate action on failures.

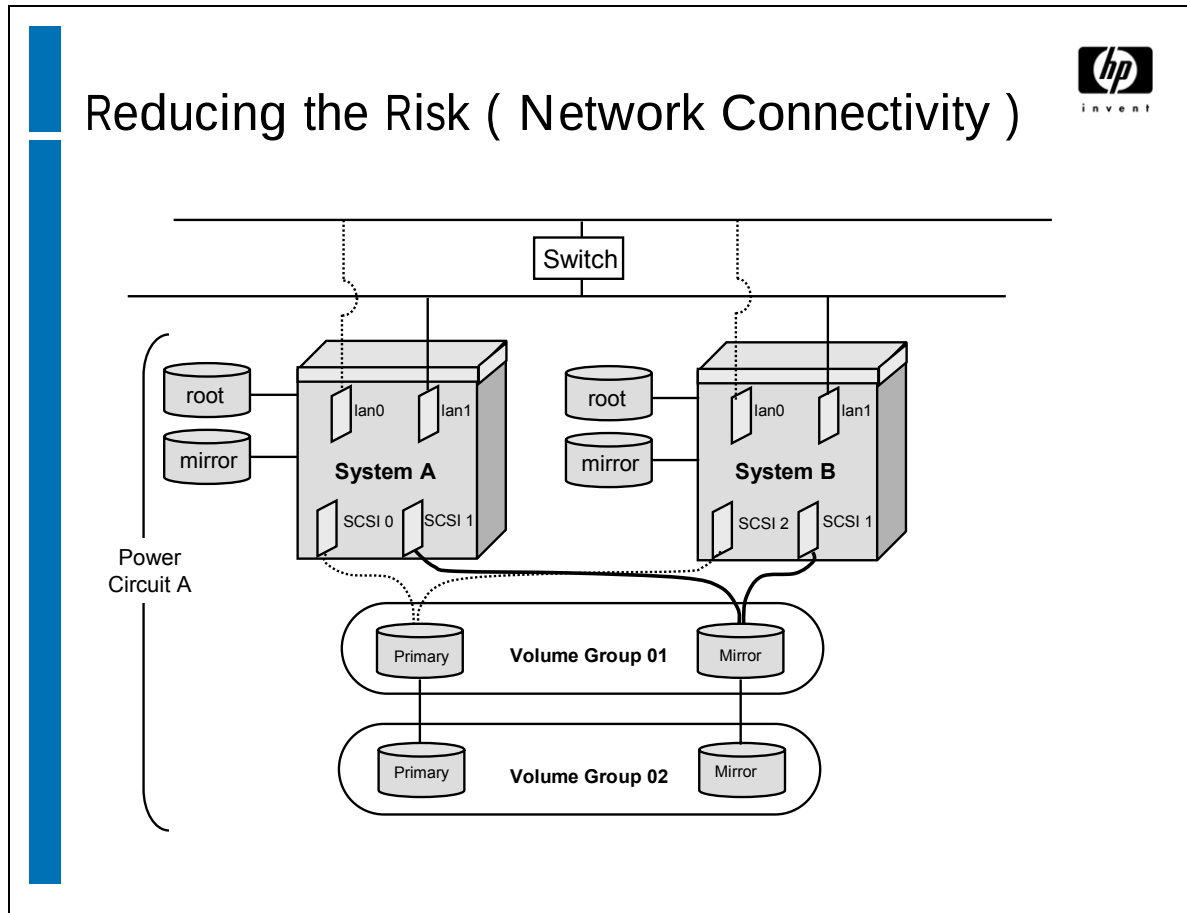
These failures can be:

- Hardware – e.g. CPU failure.
- Software – e.g. termination of a process.
- Monitored resource event – e.g. task wait time exceeds a defined value.

This design also facilitates the easy *movement* of applications between systems, allowing system upgrades (both hardware and OS software) to be performed with minimum planned downtime. (The downtime would just be the time taken to close the application on one system and restart it on the other.)

Depending on the shared disk technology, it is also possible to perform backups from the second system.

3-4. SLIDE: Reducing the Risk (Network Connectivity)



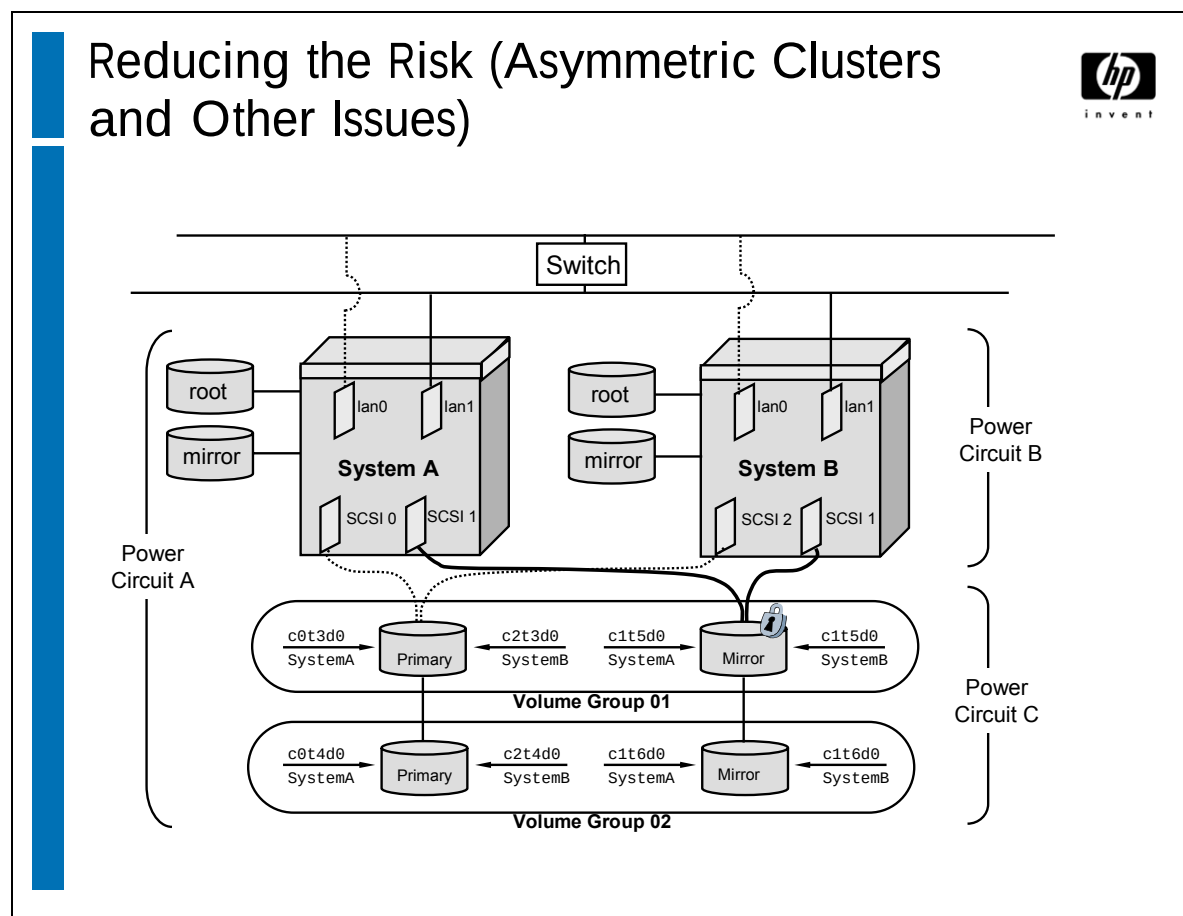
Student Notes

In an HA cluster, clients will access the services via the network. This is another SPOF that can be accommodated by the cluster software. Using multiple LAN controller cards, or LANICs, bridges, switches, and cabling, etc., a system can switch rapidly to a *standby* LANIC without any perceived interruption by clients.

Applications and services can also be *linked* to network availability such that they will relocate to another system if such a standby is not available and the *monitored* network fails on this system.

Always consider the availability from the client through to the *service*, including redundant *routes*.

3-5. SLIDE: Reducing the Risk (Asymmetric Clusters and Other Issues)



Student Notes

In addition to reducing physical risks, a solid high availability plan must also address:

- Potential power outages.
- HA training for operators and support personnel.
- Agreed upon procedures for recovering from various failures.
- The availability of documentation for the HA products.
- A well configured cluster which follows the "best practices" for cluster configuration.

The above slide provides an example of what can happen when a cluster is *not* configured using "best practices." One of these "best practices" is to make the cluster symmetrical. One part of symmetry is that device file names used to reference the shared disks should be the same across all nodes in the cluster.

Because the two systems use two different instances of SCSI controllers to share a SCSI bus, the cluster is asymmetric. The device file names are different for these shared disks, where the device file name is dependent on the system from which the disk is accessed. For example, If a disk is accessed from SystemA, the device file name is `c0t3d0`, but if it is accessed from SystemB, the same disk has a device file name of `c2t3d0`.

3–6. SLIDE: Disk Configurations



Disk Configurations

Possible disk configurations for a Serviceguard cluster include:

- LVM Mirrored JBODS
- Disk Array Configuration
- Storage Area Networks

Student Notes

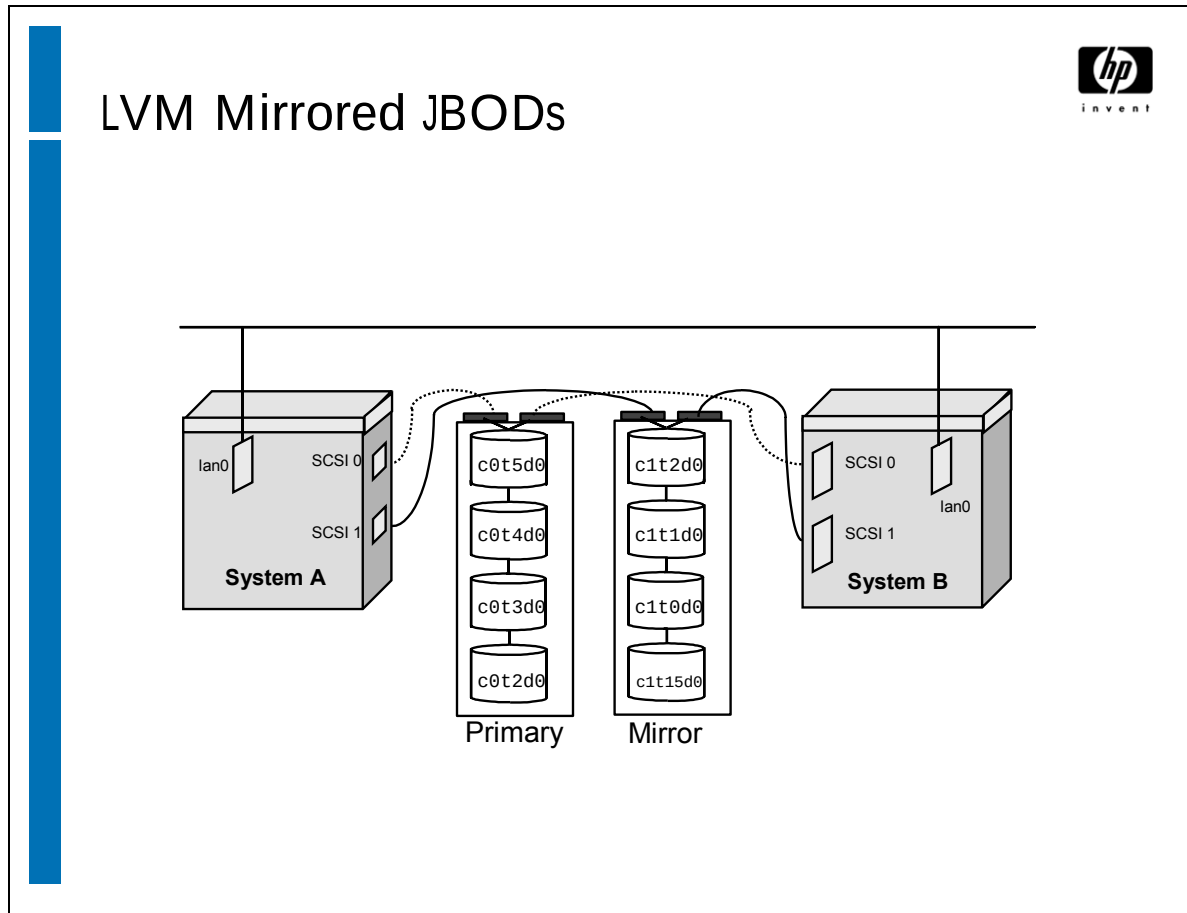
The above slide outlines three different disk technologies/configurations supported by Hewlett Packard in a Serviceguard environment.

Each disk technology has its own set of advantages and disadvantages. These include:

- Performance
- Price
- Capacity
- Ease-of-use

The advantages and disadvantages of each of these disk technologies are discussed in the following pages.

3-7. SLIDE: LVM Mirrored JBODs

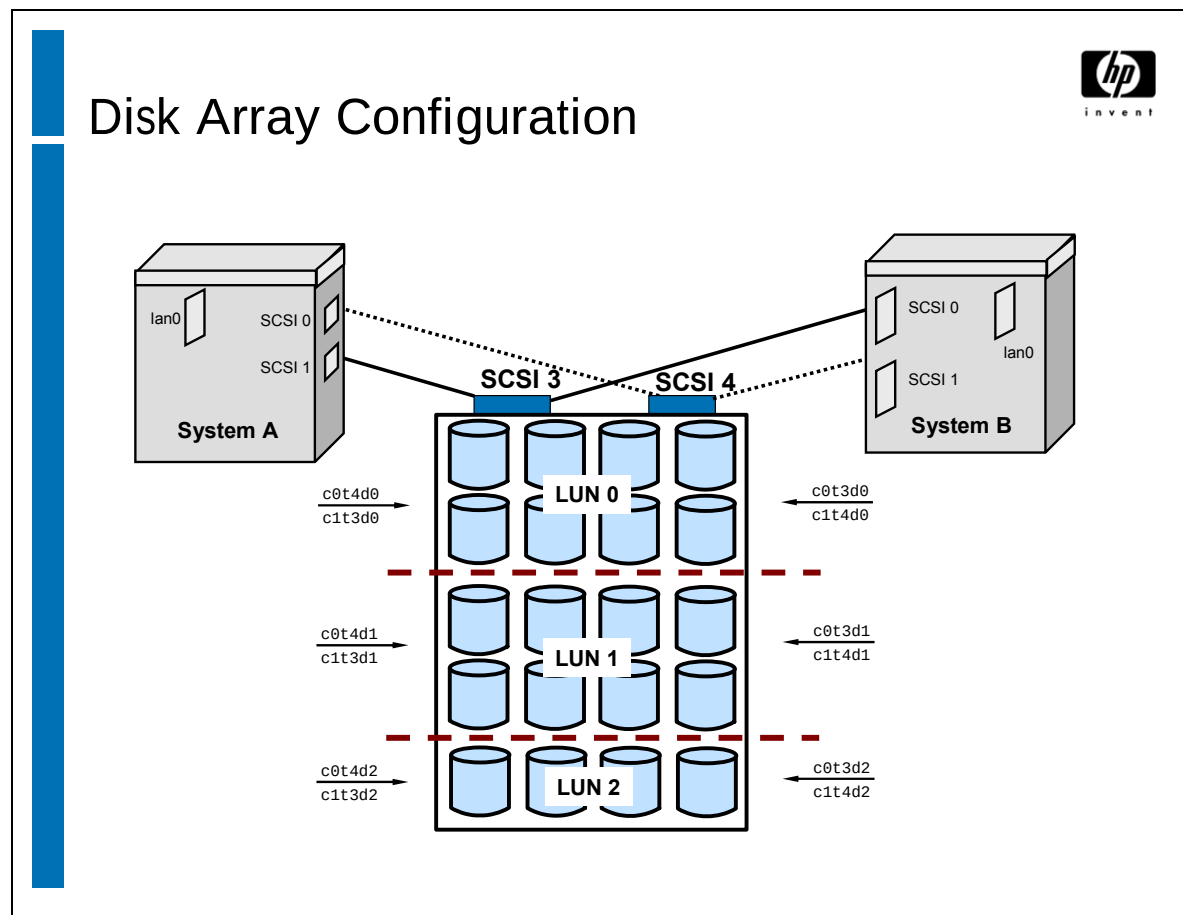


Student Notes

The disk technology that uses multiple standalone disk drives is also known as *just a bunch of disks* (JBODs).

High Availability Disk Enclosures can rack-mount multiple disk drives in storage cabinets, each disk drive being hot-pluggable. Every component is highly available (dual components) including the power supplies, power cords, cooling fans, and internal Fibre Channel or SCSI buses.

3-8. SLIDE: Disk Array Configuration



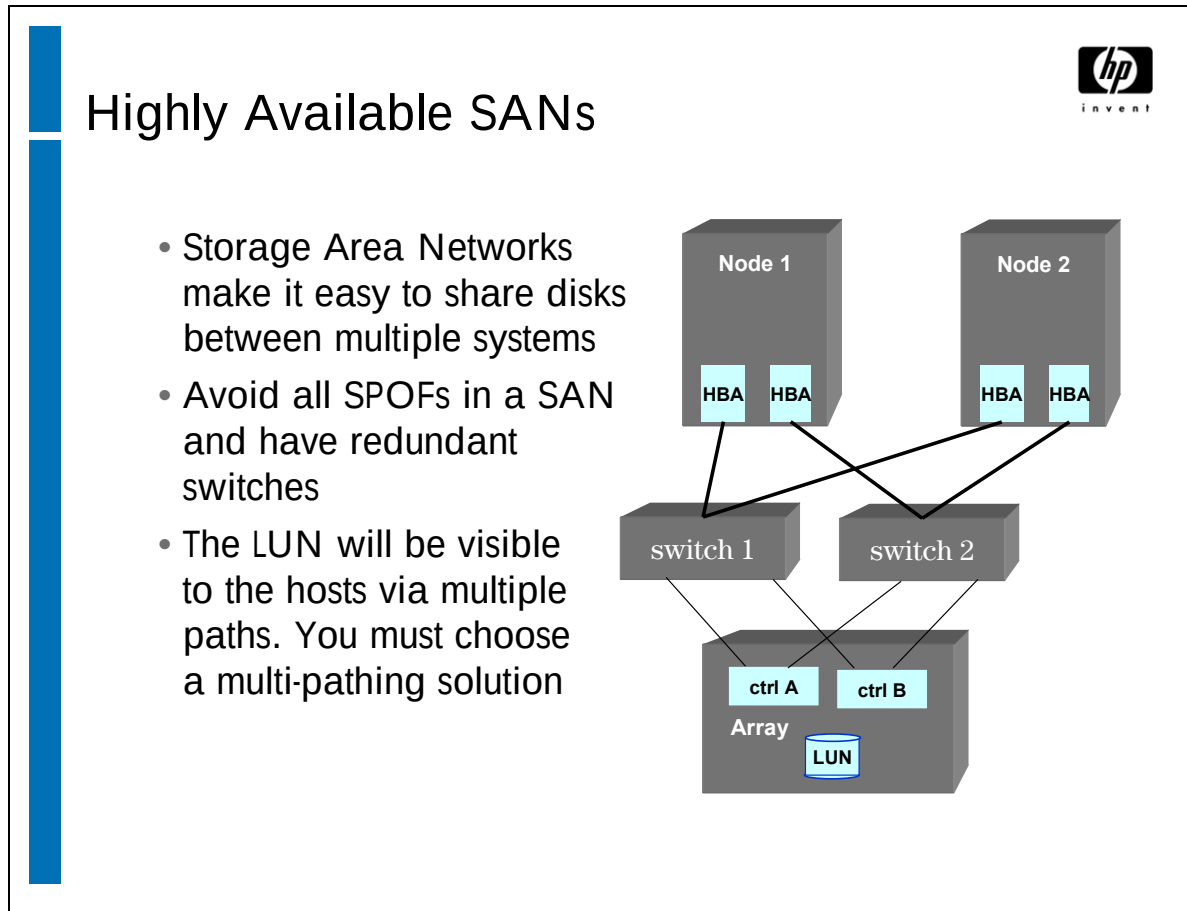
Student Notes

High Availability Disk Arrays

High availability disk arrays are only supported in a Serviceguard environment in which dual service processors (SPs) are used. Each SP is assigned a SCSI address and appears to the HP-UX operating system as one large disk (or several large disks using SCSI LUNs). Only the SPs are assigned SCSI addresses; the individual disk drives are not.

If SCSI logical unit numbers (LUNs) are used, each SCSI LUN will reference one or more disk drives within the array. Within each LUN, a different RAID level can be defined. The example on the slide shows three LUNs, with two LUNs containing eight disks and one LUN containing four disks. All of the LUNs can be configured to be accessible by both SPs, providing high availability at the SP level.

3-9. SLIDE: Highly Available SANs



Highly Available SANs

- Storage Area Networks make it easy to share disks between multiple systems
- Avoid all SPOFs in a SAN and have redundant switches
- The LUN will be visible to the hosts via multiple paths. You must choose a multi-pathing solution

Student Notes

Storage Area Networks (SANs) make it easier to share storage between multiple systems. A separate network is created for the storage. LUNs can be made visible to any hosts connected to the SAN. As with other types of disk configurations, single points of failure must be avoided.

Multi-pathing Solutions

In the example on the slide each host will see the LUN twice. To HP-UX the one LUN will look like two different devices. A multi-pathing solution must be used in this situation.

One choice is to use LVM PVlinks. This is a no-cost solution, but one path will be the primary and the other path will only be used in case of a failure. Also, you must manually configure both paths.

Another choice is to purchase multi-pathing software such as Veritas Dynamic Multi-Pathing (DMP), EMC Power-path, or HP Secure Path. All of these products support load balancing across the paths and have the ability to dynamically discover the multiple paths.

3-10. SLIDE: Network Configurations



Network Configurations

Possible network configurations for a Serviceguard cluster include:

- Redundant LANs
- Normal Network Flow
- Network Flow during Package Failover
- Routed Networks and Serviceguard Clusters

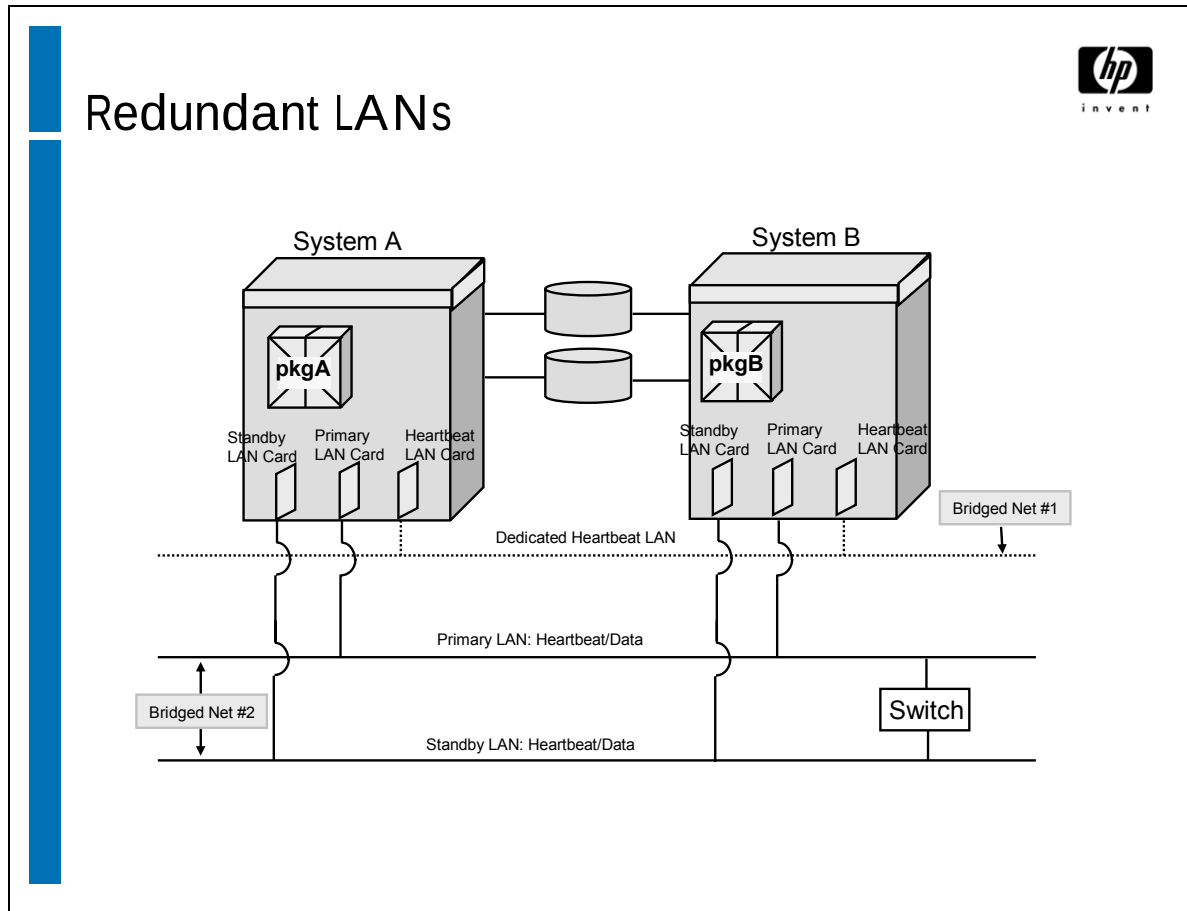
Student Notes

To eliminate single points of failure for networking, redundant network interfaces, redundant cables, and redundant LANs are required.

There are many possible network configurations to achieve network redundancy. Some of the configurations are supported, and others are not supported in a Serviceguard environment.

The following slides provide a sample of supported and unsupported network configurations for a Serviceguard environment.

3-11. SLIDE: Redundant LANs



Student Notes

An example of redundant LAN interfaces to the same network is shown on the above slide. In the slide, each Serviceguard node is connected to two "bridged networks." The first connection is a "dedicated" connection to a dedicated Heartbeat LAN subnet. This is a standard type connection with no redundant LAN connections. If the heartbeat LAN card fails, then the node will be unable to get the heartbeat LAN subnet.

The second connection is to a Heartbeat/DATA subnet. In this case, each node has two connections to the subnet: a primary LAN card connection and a standby LAN card connection. Note the primary and standby LAN segments are connected by a switch (it could also be a hub or a bridge) to provide redundancy between the two segments of the Heartbeat/DATA subnet. In the case of a primary LAN card failure for the Heartbeat/DATA subnet, Serviceguard will perform a local switch of the IP addresses on the primary LAN card to the standby LAN card.

The reason for not having a standby LAN card for the dedicated Heartbeat LAN: redundancy is already being provided by sending heartbeat packets across both LAN subnets. If a LAN card to the dedicated heartbeat LAN subnet is lost, the cluster can continue as long as the heartbeat can get through to each node on the Heartbeat/DATA subnet. Serviceguard only requires one of the heartbeat packets to get through, not both.

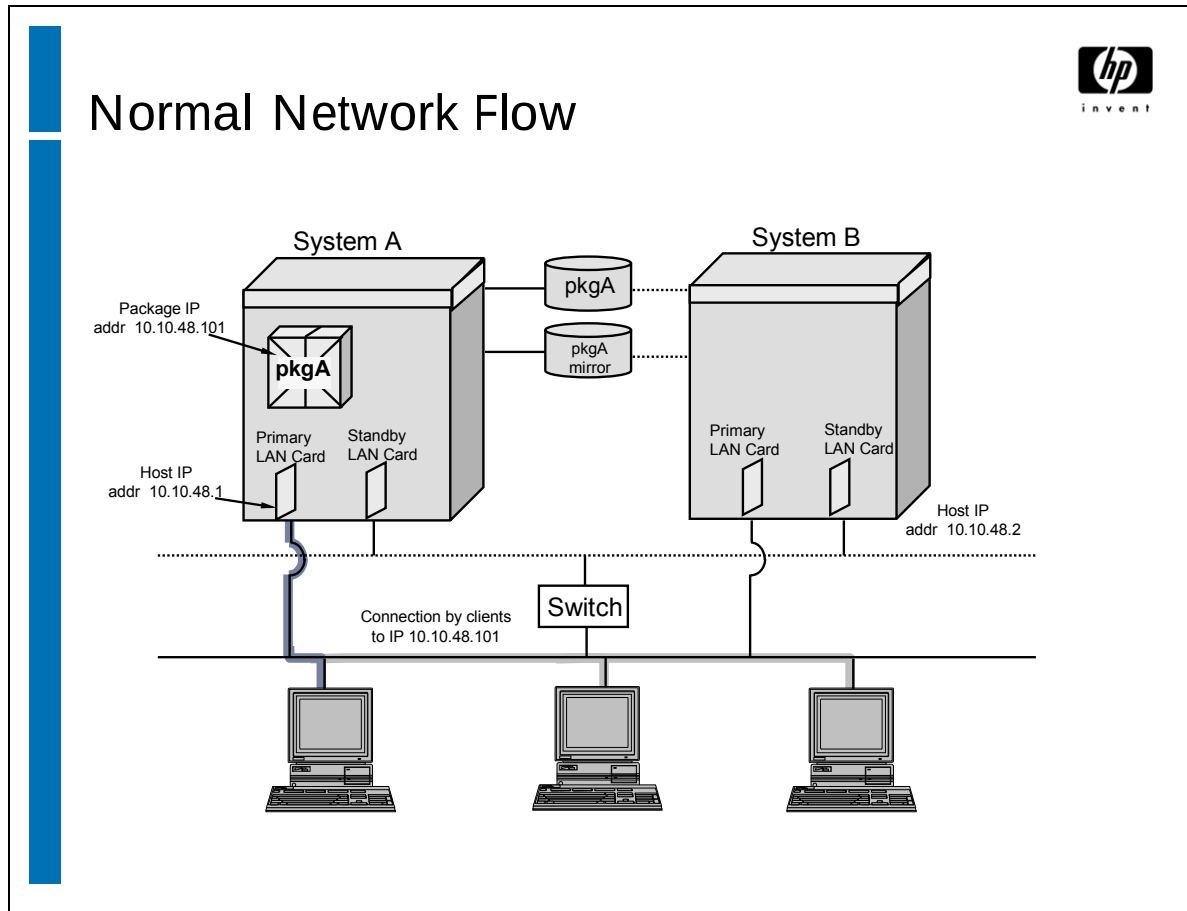
Standby LAN Cards

Standby LAN cards are configured automatically by Serviceguard during the cluster setup/configuration phase. The two requirements to be a standby LAN card are:

- The LAN card must be unconfigured (it must not have an IP address assigned).
- The standby LAN card must be on the same physical network as the primary LAN card. In other words, the standby LAN card must be able to communicate (send packets) to the same nodes on which the primary LAN card communicates.

The configuration shown on the slide is a supported network configuration for a Serviceguard environment. The bridge is a key component, because it combines the two physical LAN segments into one continuous bridged network.

3-12. SLIDE: Normal Network Flow



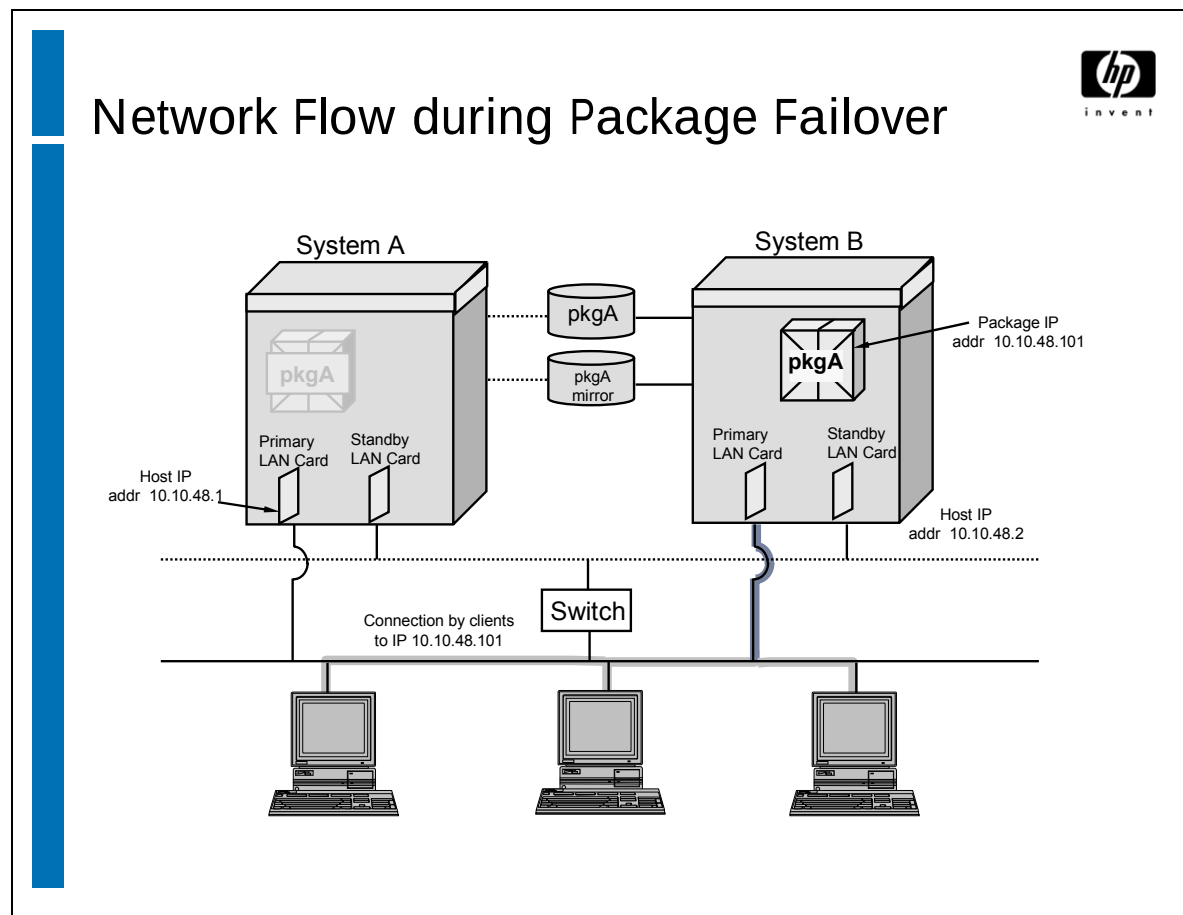
Student Notes

The above slide shows how clients connect to HA applications in a Serviceguard environment. From the outside, it *appears* as the client connects with the server hosting the HA application (as normal). But, internally, something quite different is happening.

While SystemA still contains an IP address that represents the host computer, it also contains a "floating" IP address that represents the application. Recall that all resources associated with the application are combined into an entity called a "package." When an application starts up, one of the resources for the package, namely the IP address, is added to the LAN card of the host on which it is running. Therefore, once the application starts, there are two IP addresses associated with the primary LAN card on SystemA: the IP address of the system itself (10.10.48.1) and the IP address of the application (10.10.48.101).

When a client connects to the application in a Serviceguard environment, it does *not* connect with the IP address of the system hosting the application; instead, it connects with the IP address of the application itself! In the above slide, notice the clients are connected to IP address 10.10.48.101, which is the IP address of PackageA.

3-13. SLIDE: Network Flow during Package Failover



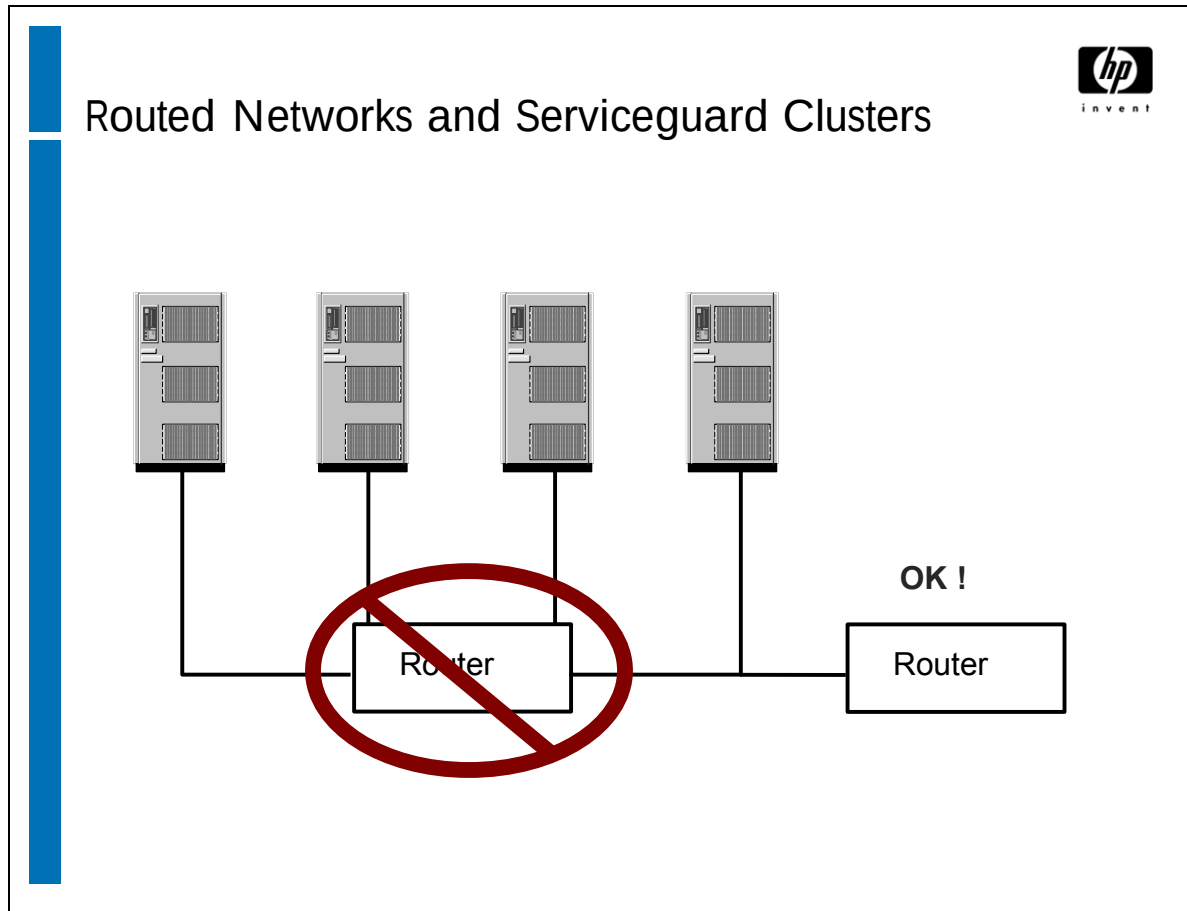
Student Notes

The above slide shows the impact on the network flow when a package moves from one host to another host in a cluster.

The main impact of moving a package is the IP address of the application moves when the package moves. On the previous slide, we saw the clients were sending their network packets to SystemA, because that was where the IP address of PackageA was located. During a package failure or package move, the IP address of the application is de-configured off the primary system's LAN card, and is configured onto the secondary system's LAN card. As a result, all network packets destined to the IP address of the application are now routed to the secondary system, since that is the new location of the package's IP address.

The major benefit of having clients connect to the "floating" IP address of the application is the clients do not need to be re-configured when the application moves. If there were no IP address associated with the application, then the clients would all need to be reconfigured when the application started on a new node. But, because Serviceguard supports floating IP addresses, client modifications are not needed when an application moves to another node in the cluster.

3-14. SLIDE: Routed Networks and Serviceguard Clusters

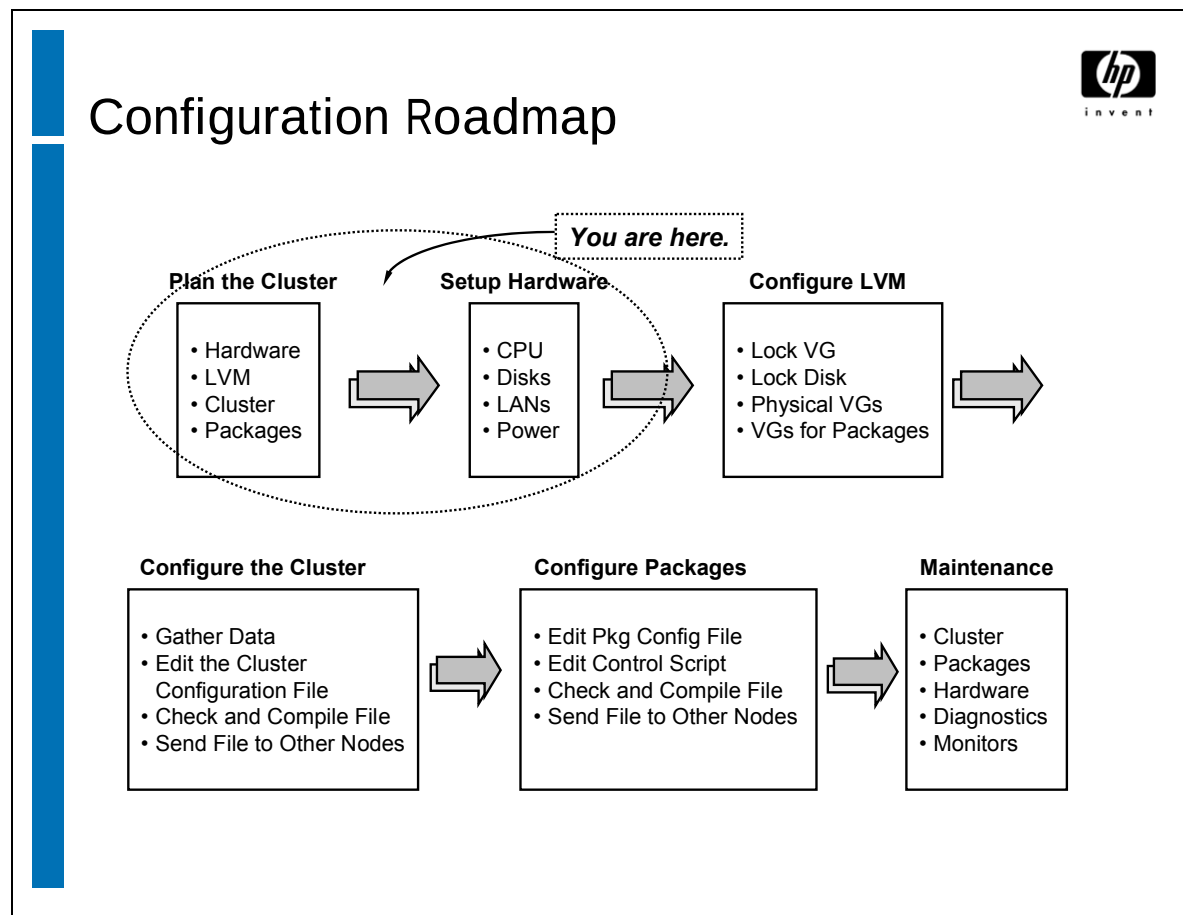


Student Notes

There are a number of rules to remember when working with routers in a Serviceguard environment:

- The members of an HA Cluster may not be connected to each other via a router. All nodes must be on the same segment in a routed network. Therefore, the Enterprise Cluster can connect to a routed network.
- Although the heartbeat uses IP, cluster formation uses link level addressing (streams DLPI- Data Link Provider Interface). The complicated "state machine" and timing concerns at cluster formation means that a short deterministic algorithm must be used. Sockets, which are associated with higher-level protocols, would take too long to perform this function. Therefore, heartbeats do not communicate via sockets.
- These requirements also prevent HA Clusters from working in a WAN environment.

3-15. SLIDE: Configuration Roadmap



Student Notes

The above slide provides a good "roadmap" of activities needed to configure and maintain a Serviceguard environment.

At this point in the class, you are about to complete the "Plan the Cluster" and "Setup Hardware" phases by performing the "Hardware Inventory" lab exercise at the end of this module.

The next steps, as shown on the slide, will be to:

- Configure a share volume group using LVM
- Configure/Setup the Cluster
- Configure Package to execute in the Cluster
- Perform Cluster and Package maintenance activities.

3-16. LAB: Hardware Inventory

Directions

Planning is an essential component of any successful Serviceguard implementation. Part of that planning involves identifying all hardware that will be used in the cluster and identifying what it will be used for. This includes servers, LAN cards, and disks.

In this lab, we will examine some of the hardware requirements of Serviceguard and identify the servers, LAN cards, and disks we will be using in our classroom implementation of a Serviceguard Cluster. We will also mirror the root disk.

Part 1: Identify the Servers

Data to Record	Command Used	Server1	Server2
Node Name	hostname uname -a		
System Model	model		
OS Rev	uname -r		
Memory	dmesg		
Date / Time / Timezone	date		
Serviceguard Revision	swlist -l product \ grep -i guard		

NOTE: The servers do not need to be identical in a Serviceguard cluster. Packages containing applications will failover more smoothly if date and time are synchronized and if the adoptive node has adequate resources (CPU, memory, etc.) to run the application.

Part 2: Identify the LAN Cards

Each server in a Serviceguard cluster must have at least two LAN cards.

In a production environment, HP strongly recommends at least three LAN cards per server with at least one LAN card per server used exclusively for heartbeats (as a dedicated heartbeat LAN).

In our classroom cluster, we will not have a separate, dedicated heartbeat LAN card on each server. Instead, each server will have a primary LAN card and a standby LAN card. Some network traffic will use the public LAN cards. Other network traffic (heartbeats and client/server data) will use the private LAN cards (to be configured later). If a LAN card fails, all traffic will be sent via the standby LAN card.

The primary LAN cards must be configured with an IP address. The standby LAN cards must not have an IP address. Serviceguard will automatically assign the primary's IP address to the standby if the primary fails. The primary and standby LAN cards on both servers must all reside on the same bridged-subnet. Let's verify all of this.

First, consider the <u>Primary</u> LAN Card	Command	Server1	Server2
IP Address	<code>ifconfig lanX (often lan0) netstat -in</code>		
Subnet	<code>netstat -in</code>		
Hardware Path	<code>lanscan ioscan -fC lan</code>		
Station Address	<code>lanscan</code>		
Net-Interface Name / PPA	<code>lanscan</code>		
Description (from ioscan)	<code>ioscan -fC lan</code>		

Now, consider any <u>Standby</u> LAN Card	Command	Server1	Server2
IP Address	ifconfig lanY (often lan1) netstat -in	These LAN cards <u>must not</u> have an IP address assigned.	
HW Path	lanscan ioscan -fC lan		
Station Address	lanscan		
Net-Interface Name / PPA	lanscan		
Description (from ioscan)	ioscan -fC lan		

1. If you are using local equipment, make certain you know which LAN card is in which slot in the backplane of each server. If you are using remote equipment, continue with the next step.
2. Are all LAN cards physically attached to the same bridged-subnet? Let's test.

Using the `linkloop` command, verify that, on each server, you have connectivity between all of the available `lan` cards on that server **AND** on the other server.

NOTE: The `linkloop` command can test connectivity of a LAN card that has no IP address. `linkloop` takes the station address as an argument.

NOTE: The `linkloop` command, without options, assumes you intend to send a test packet out of the first LANIC in the system. Therefore, if you are using another LANIC as the origination LANIC, it will be necessary to use the `-i` option with the appropriate interface. The default for the `-i` option is `-i 0`. Thus, if your currently configured system LANIC is `lan1`, then you would enter:

```
# linkloop -i 1 0x_destination_station_address
```

On Server 1:

```
# linkloop [ -i N ] 0x_1st_lan_card_server1_station_address
# linkloop [ -i N ] 0x_2nd_lan_card_server1_station_address
# linkloop [ -i N ] 0x_3rd_lan_card_server1_station_address
# linkloop [ -i N ] 0x_4th_lan_card_server1_station_address

# linkloop [ -i N ] 0x_1st_lan_card_server2_station_address
# linkloop [ -i N ] 0x_2nd_lan_card_server2_station_address
# linkloop [ -i N ] 0x_3rd_lan_card_server2_station_address
# linkloop [ -i N ] 0x_4th_lan_card_server2_station_address
```

On Server 2:

```
# linkloop [ -i N ] 0x_1st_lan_card_server2_station_address
# linkloop [ -i N ] 0x_2nd_lan_card_server2_station_address
# linkloop [ -i N ] 0x_3rd_lan_card_server2_station_address
# linkloop [ -i N ] 0x_4th_lan_card_server2_station_address

# linkloop [ -i N ] 0x_1st_lan_card_server1_station_address
# linkloop [ -i N ] 0x_2nd_lan_card_server1_station_address
# linkloop [ -i N ] 0x_3rd_lan_card_server1_station_address
# linkloop [ -i N ] 0x_4th_lan_card_server1_station_address
```

If any `linkloop` command fails, we must correct the LAN connectivity problem before proceeding with building the cluster.

Answer:

All LAN cards should be able to communicate with each other. If there are connectivity issues, notify your instructor.

Part 3: Identify the Disks

Recorded Data	Command	Server1	Server2
Number of (a) sharable SCSI cards or (b) sharable bus adapters	(a) <code>ioscan -fC ctl</code> or (b) <code>ioscan -fC ba</code>		
Hardware Path of (a) sharable SCSI cards or (b) sharable bus adapters (e.g. 8/0)	(a) <code>ioscan -fC ext_bus</code> or (b) <code>ioscan -fC ba</code>		
Initiators Hardware address of (a) the SCSI card itself, or (b) hardware address of sharable bus adapters (e.g. 8/0.7.0)	(a) <code>ioscan -fC ctl</code> or (b) <code>ioscan -fC ext_bus</code>		
Hardware Paths of sharable Disks (e.g. 8/0.5.0)	<code>ioscan -fC disk</code>		
Device file names of sharable disks (e.g. c0t5d0)	<code>ioscan -fnC disk</code>		

High Availability Planning

1. If you are using local equipment, determine which disks are private and which are shared (you may be able to trace the cables). If you are using remote equipment, ask your instructor for this information

2. **Special note:** If you are using local equipment, then continue with this question. If you are using remote equipment, then skip this question completely, and go immediately to question 4 below.

Identify the physical location of each specific disk. How do you know which disk is in which slot in which cabinet?

Answer:

A simple way of locating a disk is to issue the following command, and watch which disk's access light illuminates.

```
# dd if=/dev/rdisk/disk_in_question of=/dev/null
```

3. What volume groups currently exist?

Answer:

```
# strings /etc/lvmtab
```

4. Which disks are currently in use and owned by these existing volume groups?

Answer:

```
# vgdisplay -v
# strings /etc/lvmtab
```

Informational only: Does each shared disk have the same device file name regardless of which server we use to access the disk? Would it be possible for the same disk to have a different device file name on server1 vs. server2? Would Serviceguard work in this environment?

When the nodes of a cluster are all identical in their configurations (including device file names for shared disks), this cluster is referred to as a "Symmetric Cluster."

For more on maintaining symmetric clusters, please continue with Part 4 on the next page.

Part 4: Information on Maintaining Symmetric Clusters

Note: This section of the lab is informational only. There is no work that must be completed in this section of the lab.

If a disk is simultaneously attached to SCSI card instance 1 on server1 and SCSI card instance 2 on server2, the disk will have a different device file name on server1 vs. server2. While Serviceguard will work in this environment, it may be less confusing if this situation is avoided.

Most Serviceguard administrators would prefer a cluster that is symmetric (where the disk device files for shared volume groups are identical between systems) as it is easier to administer. And while it is possible to alter device file names through the procedure below, this concept is quickly becoming a moot point with the introduction of SAN's and other disk arrays into Serviceguard environments. With SAN's and certain arrays, it is nearly impossible to create identical disk device file names between systems.

In a SCSI disk environment, it is possible to alter instance numbers in order to create identical device file names for disks. The procedure follows:

Begin by creating a file that maps H/W addresses to instance numbers and then run **ioinit** against this file.

For example, if the current instance number assignment on server1 looks like this:

8/0	ext_bus	0
8/4	ext_bus	1
8/8	ext_bus	2

and the current instance number assignment on server2 looks like this:

8/0	ext_bus	2
8/4	ext_bus	1
8/8	ext_bus	0

we can use the following method to make server2 look like server1:

To give the 8/0 SCSI card an instance of 0, and the 8/8 SCSI card an instance of 2, create a file called `/tmp/mappings` which contains:

8/0	ext_bus	0
8/4	ext_bus	1
8/8	ext_bus	2

Then, execute `ioinit -f /tmp/mappings -r` on server2. The system will reboot with the new mapping. This must be done BEFORE the shared volume groups are created!

3-17. LAB: Configuring and Managing LVM Boot Disks on Integrity-Based Systems (Optional)

Special Note

- a. Lab 3-17 is for those using Integrity, or Itanium-based, systems.
- b. If you are using PA-RISC systems, skip this lab and proceed to Lab 3-18.
- c. If you are not sure which equipment you have, ask your instructor.

Overview

In this hands-on lab, you will create mirrors for all of the logical volumes that comprise the root volume group, `vg00`. **These steps should be performed on both systems in the cluster to improve the availability of the root disk on each system of the cluster.**

Carefully read and follow the instructions below.

Part 1: Mirroring an Integrity Boot Disks: Configuring Partitions

In order to improve system stability, many administrators choose to mirror the system boot disk. Doing so guarantees that your system continues running, even if one of the boot disks or SCSI interface cards on your system is damaged.

1. First, use the `lvlnboot` command to determine the location of your current boot disk. After that, use the `ioscan` and `diskinfo` commands to identify a disk large enough to serve as a mirror.

```
# lvlnboot -v
# ioscan -funC disk
# diskinfo /dev/rdisk/cxtxdx
```

Module 3
High Availability Planning

2. Use the `idisk` command to create a 500 MB system partition, an OS partition, and a 400 MB service partition on your new boot disk.

```
# vi /tmp/idf
```

```
3
EFI 500MB
HPUX 100%
HPSP 400MB
```

```
# idisk -wf /tmp/idf /dev/rdisk/cxtxdx
```

Note that the screen output from this `idisk` command is fairly large. Down about 21 lines from the top of the screen output, you should see something like the following, proving that the three partitions have been created:

```
Primary Partition Table (in 512 byte blocks):
Partition 1 (EFI):
  Partition Type GUID      = c12a7328-f81f-11d2-ba4b-00a0c93ec93b
  Unique Partition GUID    = cca5f42a-96e8-11d9-8002-d6217b60e588
  Starting Lba             = 0x22
  Ending Lba               = 0xfa021
Partition 2 (HP-UX):
  Partition Type GUID      = 75894c1e-3aeb-11d3-b7c1-7b03a0000000
  Unique Partition GUID    = cca5f448-96e8-11d9-8003-d6217b60e588
  Starting Lba             = 0xfa022
  Ending Lba               = 0x430e6fb
Partition 3 (HPSP):
  Partition Type GUID      = e2a1e728-32e3-11d6-a682-7b03a0000000
  Unique Partition GUID    = cca5f466-96e8-11d9-8004-d6217b60e588
  Starting Lba             = 0x430e6fc
  Ending Lba               = 0x43d66fb

Legacy MBR (MBR Signatures in little endian):
  MBR Signature = 0x46f1a5cc

Protective MBR
```

3. Create device files for the new partitions.

```
# insf -eC disk
```

4. Use the `idisk` command again, this time to verify the partition table on the new boot disk.

```
# idisk /dev/rdisk/cxtxdx
```

Once again, in the middle of the `idisk` screen output, you should see three partitions.

5. Verify that the partition device files were created properly. There should be block **and** raw device files (`cxtxdxs1`, `cxtxdxs2`, and `cxtxdxs3`) for the new boot mirror.

```
# ioscan -func disk | grep cxtxdx
```

Part 2: Mirroring the Boot Disk: Configuring the System Partition

After you create the partition table, you can initialize and populate the EFI system partition.

1. First, populate the `/efi/hpux` directory in the new EFI partition.

```
# mkboot -e -l /dev/rdisk/cxtxdx          ← Takes about 5 seconds to execute
```

2. Change the `auto` file **on both boot disks** so the host can boot without quorum. Change the boot string in the file to `boot vmunix -lq`. Be sure to specify section `s1`, the EFI partition in your `efi_cp` command! Begin by creating a temp file to store the boot string. Note: `cxtxdx` below is your new boot disk and `cytydy` is the original boot disk.

```
# vi /tmp/auto
```

```
boot vmunix -lq
```

```
# efi_cp -d /dev/rdisk/cxtxdxs1 /tmp/auto /efi/hpux/auto
# efi_cp -d /dev/rdisk/cytydys1 /tmp/auto /efi/hpux/auto
```

3. Verify the contents of the system partition. Again, be sure to specify the `s1` partition!

```
# efi_ls -d /dev/rdisk/cxtxdxs1
```

4. Verify the contents of the `auto` file in the system partition.

```
# efi_cp -d /dev/rdisk/cxtxdxs1 -u /efi/hpux/auto /tmp/auto.contents
# cat /tmp/auto.contents
```


Part 3: Mirroring an 11i v2 Boot Disk: Configuring the OS Partition

Now, configure the OS partition. Recall that the OS partition is essentially an encapsulated LVM boot disk.

1. Initialize and add the new OS partition to vg00. Be sure to specify section s2, the OS partition!

```
# pvcreate -fB /dev/rdisk/cxtxdxs2
# vgextend vg00 /dev/dsk/cxtxdxs2
# lvmboot -v
```

← To see your newly initialized (though still incomplete) boot disk.

2. Mirror *all* of the logical volumes in vg00. Depending on the number, and size, of your logical volumes in vg00, this step could take up to 15 minutes to complete.

```
# for i in /dev/vg00/lvol*
> do
> echo "Now mirroring $i ...\\n\\n"
> lvextend -m 1 $i
> done
```

3. Be sure to add a new line to the /stand/bootconf file so that SDUX knows which disks are boot disks. Be sure to specify section s2, the OS partition!

```
# vi /stand/bootconf
```

l /dev/dsk/cxtxdxs2	← The first character in this line is an "ell", not a "one"
l /dev/dsk/cytydys2	← The first character in this line is an "ell", not a "one"

4. Verify that the disk was added to vg00, and that the logical volumes are synchronized. For each logical volume, the LV Status field should report "available/syncd".

```
# vgdisplay -v vg00 | more
```

5. Verify that the BDRA was updated properly. You should see references to both disks in the boot, root, and swap portions (but not the dump portion) of the output.

```
# lvmboot -v
```

Part 4: Configuring an HP-UX 11i v2 Boot Disk: Configuring the EFI Boot Manager Menu

Finally, add the new mirrored disk to the EFI boot manager menu. This ensures that the EFI will automatically attempt to boot the mirrored boot disk if the primary boot disk fails.

1. Ensure that your initial boot disk is configured as your primary boot source, and add your new boot disk to the EFI boot manager as the second (i.e. high availability alternate) boot source.

If needed, replace the hardware paths below with the hardware paths for your first and second boot disks.

# setboot	← To see the current configuration
# setboot -p 0/1/1/0.1.0	← If needed, to set the primary boot path
# ioscan -funC disk	← Obtain hardware path for the mirror disk
# setboot -h 0/1/1/1.2.0	← Set the "High Availability Alternate" (or "HAA") boot path.

2. Verify that your new boot path was recorded properly in NVRAM.

```
# setboot
```

Part 5: Mirroring the Boot Disk: Testing the New Mirror

Now for the real test: Can you successfully boot from the alternate disk? Try it!

Special Note

- a. If you are using remote equipment, remember to log into Unix using the GSP/MP. That way, you will be able to watch the boot sequence. You will also be prepared to press the “down arrow” key mentioned in step 1 below.
- b. If you are using local equipment, and are logged in to Unix directly on the console, proceed immediately with step 1 below.

1. Use the `shutdown` command to initiate the boot sequence. When given an opportunity to interrupt the EFI autoboot sequence, do so by pressing the down arrow key.

```
# shutdown -ry 0
```

When the EFI Boot Manager menu appears, press the “down arrow” key.

2. Select your mirrored disk from the boot manager selection menu and proceed with the boot process.

```
EFI Boot Manager ver 1.10 [14.60] Firmware ver 1.61 [4241]
Please select a boot option
  HP-UX Primary Boot
  HP-UX HA Alternate Boot
  EFI Shell [Built-in]
```

3. When the boot process finishes, check the `syslog.log` file to verify which disk/kernel you booted from. You should see something like the following:

```
# grep "Boot device's HP-UX HW path" /var/adm/syslog/syslog.log
vmunix: Boot device's HP-UX HW path is: 0.0.0.0.1.0
```

4. Congratulations on mirroring your boot disk on an Integrity server! At this point, you can either stay booted from your mirror copy, or do a `shutdown -ry 0` and let the system auto-boot using the primary copy.

```
# shutdown -ry 0    ← Optional
```

3–18. LAB: Configuring and Managing LVM Boot Disks on PA-Risc-Based Systems (Optional)

Special Note

- a. Lab 3-18 is for those using PA-RISC systems.
- b. If you are not sure which equipment you have, ask your instructor.

Overview

In this hands-on lab, you will create mirrors for all of the logical volumes that comprise the root volume group, vg00. **These steps should be performed on both systems in the cluster to improve the availability of the root disk.**

Carefully read and follow the instructions below. Perform the following steps on each node that will be used for Serviceguard:

Directions

1. First, find the names of the disks you will use. These instructions will use the following disks:

/dev/dsk/c0t5d0	(already part of vg00, by default)
/dev/dsk/c0t8d0	(second internal disk on the same controller)

If the systems you are using have different device file names, then please change the above device file names accordingly.

2. Ensure that the second internal disk is available for use in this mirroring lab.

```
# strings /etc/lvmtab
```

If this disk (/dev/dsk/c0t8d0) is not available, please notify your instructor.

3. Now, initialize a disk as a boot device for use with LVM (be sure to use the raw device file).

```
# pvcreate -B [ -f ] /dev/rdisk/c0t8d0
```

4. Extend the vg00 volume group to include the new disk (use the block device file).

```
# vgextend vg00 /dev/dsk/c0t8d0
```

5. Now, let's check the disk space on the newly added disk to verify that there is sufficient space to add the boot programs to the disk. If the disk at `/dev/dsk/c0t8d0` has less space than the disk at `/dev/dsk/c0t5d0`, ask your instructor for assistance.

```
# diskinfo /dev/dsk/c0t5d0
# diskinfo /dev/dsk/c0t8d0
```

6. Copy the boot programs to the LVM disk.

```
# mkboot /dev/rdisk/c0t8d0
```

7. Verify that the LIF programs were copied to the disk.

```
# lifls -l /dev/rdisk/c0t8d0
```

8. Verify the LVM configuration on the root volume group.

```
# lvlnboot -v
```

9. Using the following script, mirror each of the logical volumes, in order, onto the second disk. Depending on the number, and size, of your logical volumes in `vg00`, this step could take up to 15 minutes to complete.

```
# for i in /dev/vg00/lvol*
> do
> echo "Now mirroring $i ... \n\n"
> lvextend -m 1 $i
> done
```

10. Verify that the mirrored volumes are registered correctly in the BDRA for `vg00`.

```
# lvlnboot -v
```

11. Now, add a new line to the `/stand/bootconf` file so that SDUX knows which disks are boot disks.

```
# vi /stand/bootconf
```

l /dev/dsk/cxtxdx	← The first character in this line is an “ell”, not a “one”
l /dev/dsk/cytydy	← The first character in this line is an “ell”, not a “one”

12. Display the contents of the LIF file named **AUTO**, which is the “Autoexecute” file used by the HP-UX secondary loader at boot time. This file normally contains the “hpux” startup string. The command shown below copies the **AUTO** file to **STDOUT**.

```
# lifcp /dev/rdisk/c0t5d0:AUTO -
```

13. Modify the **AUTO** file to allow the system to boot without LVM quorum. This should be performed on both boot disks in the root volume group.

```
# mkboot -a "hpux -lq" /dev/rdisk/c0t5d0
```

```
# mkboot -a "hpux -lq" /dev/rdisk/c0t8d0
```

14. Check the primary and alternate boot paths in use on the system.

```
# setboot
```

15. Finally, change the alternate boot path to be the second disk, if the current value is something other than the second disk.

```
# setboot -a 8/0.8.0
```

16. If you have time available, you may test your mirroring by ...

- rebooting the system
- interrupt the boot sequence ("press any key")
- SEArch for potential boot devices
- boot from the mirror copy

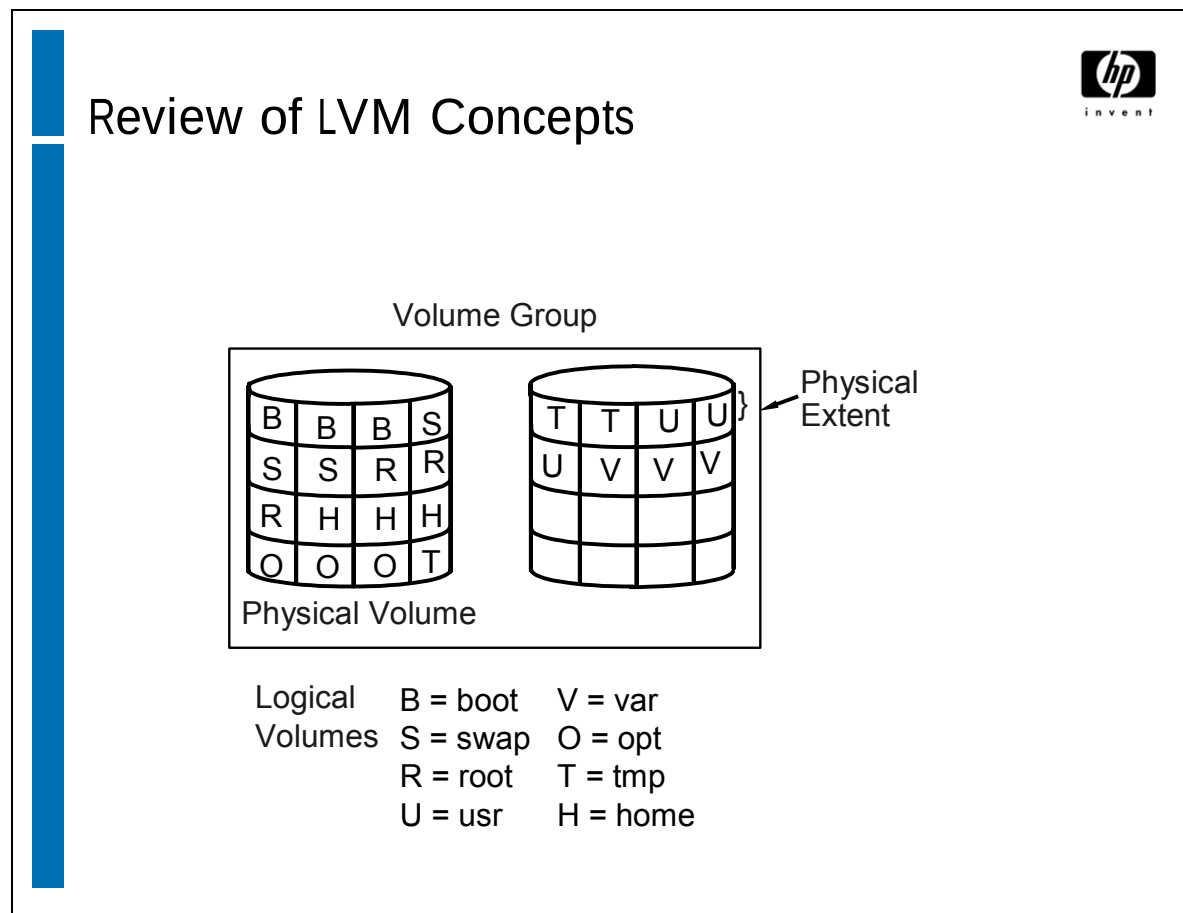
Module 4 — LVM for Serviceguard

Objectives

Upon completion of this module, you will be able to do the following:

- Understand how to configure a shared volume group for a Serviceguard environment.
- Understand how to export and import volume groups without having to specify disk devices.
- Understand how to activate a shared volume group in exclusive read/write access mode.
- Understand the list of operating system files that need to be modified to support shared volume groups in a Serviceguard environment.

4-1. SLIDE: Review of LVM Concepts



Student Notes

Logical Volume Manager (LVM) is the default disk management subsystem replacing the traditional hard disk sectioning used on HP 9000 systems. It allows the system manager much greater flexibility in controlling the size of the various disk partitions or logical volumes. As with traditional disk sections, logical volumes can be used for file systems, raw data, swap locations, or dump locations. A logical volume can be used in place of a disk section or a whole disk in an HP-UX administration command.

LVM Terminology

There are four basic LVM terms upon which this module is based: volume groups, physical volumes, logical volumes, and physical extents.

Volume Groups

The volume group is the most fundamental concept in LVM. In simple terms, it is a group of disk drives (also called physical volumes). The volume group is comprised of all the space on all the disks. When space is allocated from the volume group, it is done without concern for which disk the space is allocated on.

Physical Volumes

The physical volume is the physical disk drive. When configuring a disk for use under LVM, the entire disk must be allocated to a volume group. Once the disk is in the volume group, we refer to the disk as a physical volume, indicating the disk is under the control of LVM.

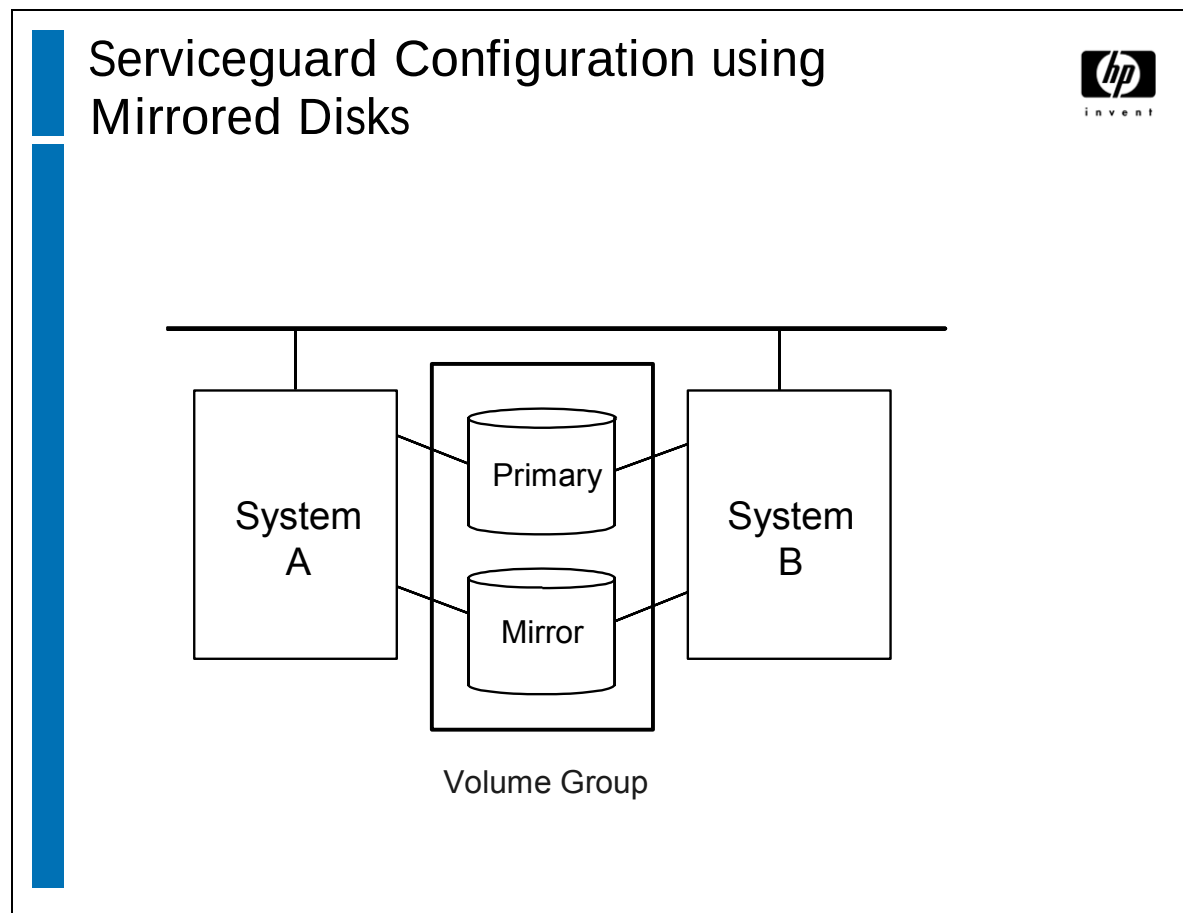
Logical Volumes

The logical volume is the unit of space which behaves like a disk section. A logical volume can be used to store a file system or it can be referenced as raw disk space. The main difference between logical volumes and disk sections is logical volumes do *not* have to be contiguous; logical volumes can be spread across multiple disks. Logical volumes can be grown dynamically.

Physical Extents

The physical extent is the increment by which space is allocated. All logical volumes and volume groups are measured in physical extents. The default size of a physical extent is 4 megabytes; however, the size can be changed when a volume group is created. The current range is 1–256 megabytes.

4-2. SLIDE: Serviceguard Configuration Using Mirrored Disks



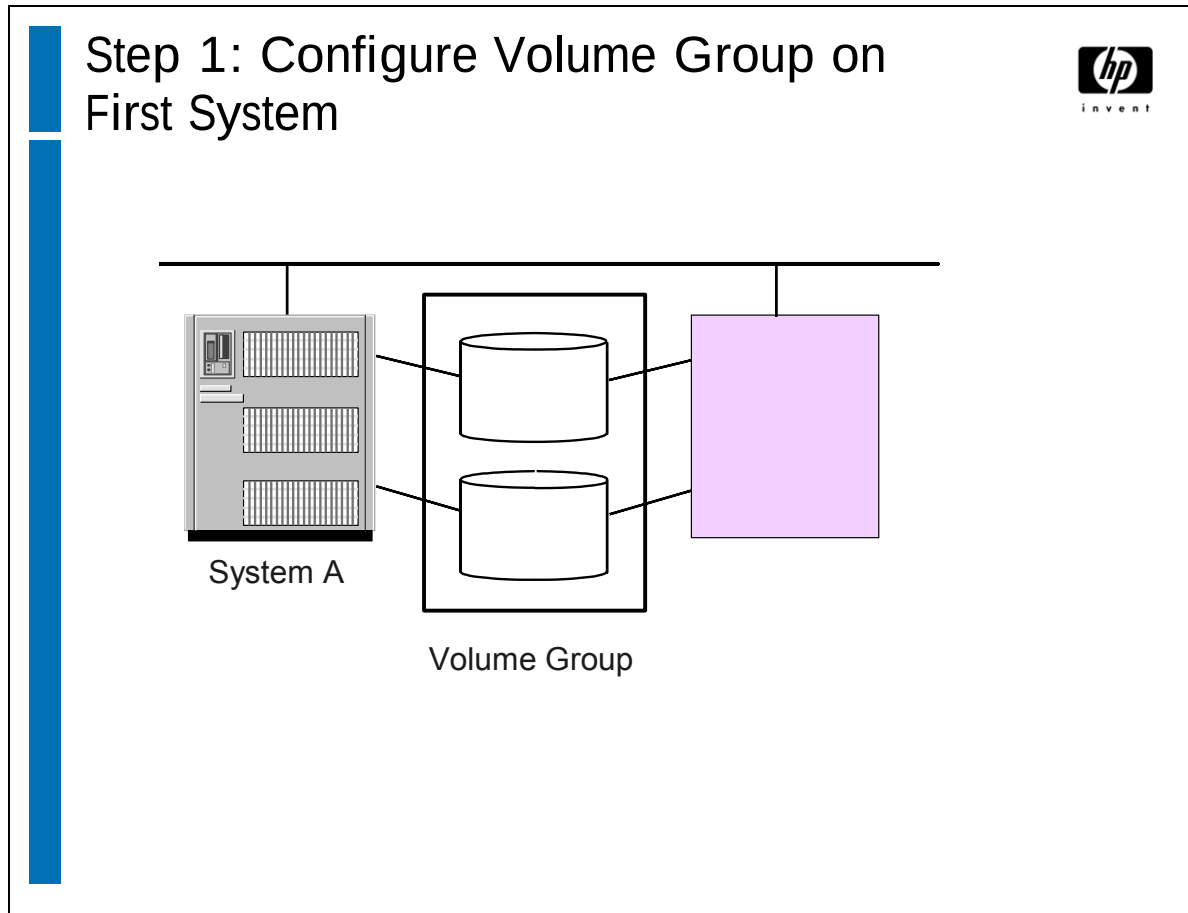
Student Notes

The slide shows a simple Serviceguard configuration for a two-node cluster. This is the simplest possible configuration for two nodes wishing to operate in a high availability Serviceguard mode.

Notice that the volume group is being shared by system A and system B. Both nodes have equal access to the volume group. The primary disk drive in the volume group is mirrored to provide protection should the primary drive fail, and the mirror is placed on a second SCSI controller card to provide protection in the event of a SCSI controller card failure.

With the above configuration, the system would be able to survive any single failure of a SCSI controller card or a SCSI drive as it relates to the volume group.

4-3. SLIDE: Step 1: Configure Volume Group on First System



Student Notes

There are two steps when configuring a shared volume group between two systems. The first step is to create the shared volume group on the first system. Examples of the commands to create the shared volume group on the first system are shown below:

Set Up the Volume Group

1. Make a directory for the volume group.

```
# mkdir /dev/vg01
```

2. Make the volume group control file (always called "group").

```
# mknod /dev/vg01/group c 64 0x010000
```

3. Prepare the disks for LVM.

```
# pvcreate [ -f ] /dev/rdisk/c_t_d0  
# pvcreate [ -f ] /dev/rdisk/c_t_d0
```

4. Create the volume group.

```
# vgcreate /dev/vg01 /dev/dsk/c_t_d0 /dev/dsk/c_t_d0
```

5. Once the shared volume group is completed, a mirrored logical volume with a file system can be created in the volume group.

Set Up a Mirrored Logical Volume and File System

1. Create a logical volume in the shared volume group.

```
# lvcreate -L 100 -n mcsg /dev/vg01
```

2. Mirror the logical volume.

```
# lvextend -m 1 /dev/vg01/mcsg
```

3. Create a journaled file system in the logical volume.

```
# newfs -F vxfs /dev/vg01/rmcsg
```

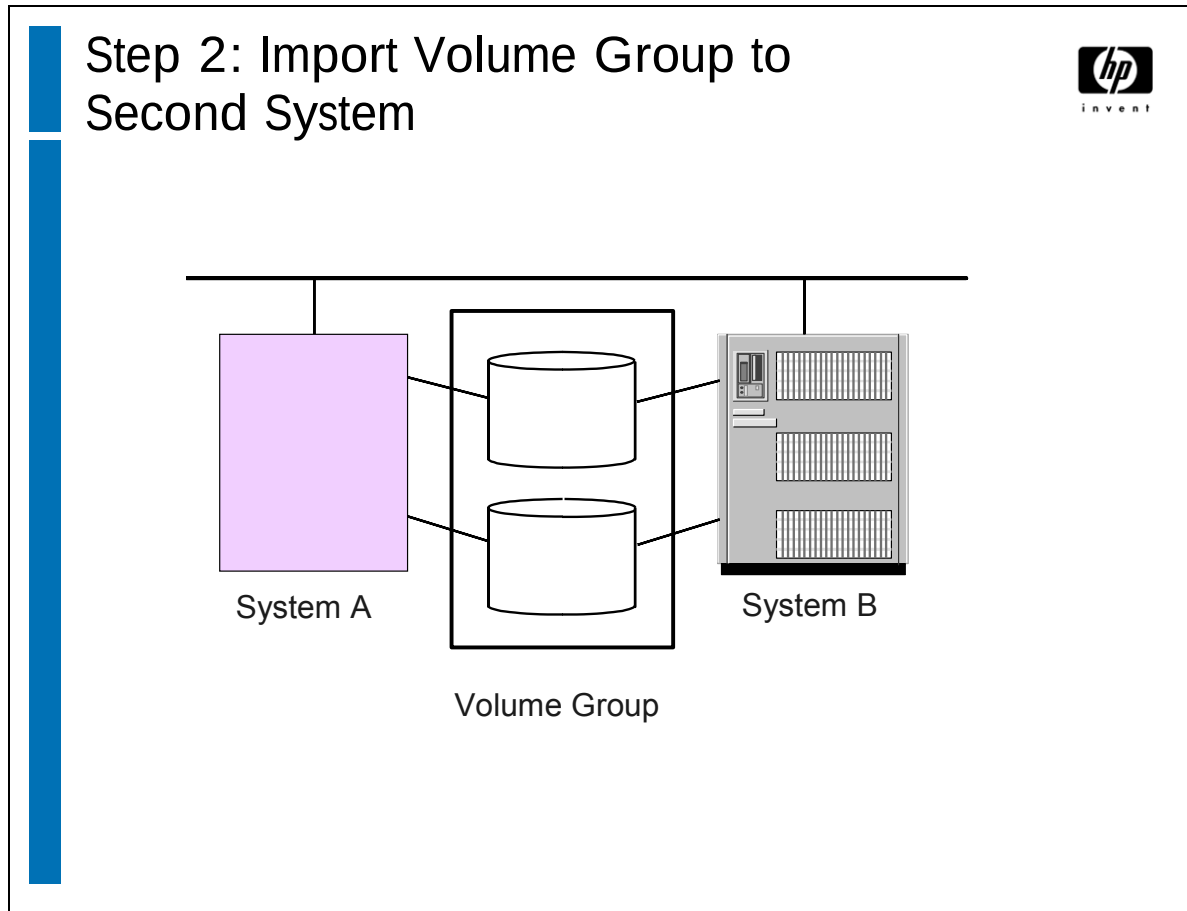
4. Make a mount point directory for the file system.

```
# mkdir /mcsg
```

5. Mount the file system to the mount point directory.

```
# mount -F vxfs /dev/vg01/mcsg /mcsg
```

4-4. SLIDE: Step 2: Import Volume Group to Second System



Student Notes

The second step, once the shared volume group has been successfully created, is to *import* the volume group to the second system in the cluster.

Export LVM Configuration to the Second System

1. Create a LVM map file for the volume group.

```
[System A] # vgexport -v -p -m /tmp/vg01.mapfile /dev/vg01
```

2. Copy the LVM map file over to the other nodes in the cluster.

```
[System A] # ftp System_B  
ftp> put /tmp/vg01.mapfile
```

3. Create a directory on the other nodes for the volume group.

```
[System B]# mkdir /dev/vg01
```

4. Create the group file on the other nodes for the volume group.

```
[System B]# mknod /dev/vg01/group c 64 0x010000
```

5. Import the volume group onto the other systems.

```
[System B]# vgimport -v -m /tmp/vg01.mapfile /dev/vg01 \  
/dev/dsk/c_d_t0 /dev/dsk/c_d_t0
```

6. The volume group must be activated to perform the command in step 7.

```
[System B]# vgchange -a y vg01
```

7. Create a configuration backup file

```
[System B]# vgcfgbackup /dev/vg01
```

8. Be sure to deactivate your volume group, as we don't want it active on both nodes simultaneously.

```
[System B]# vgchange -a n vg01
```

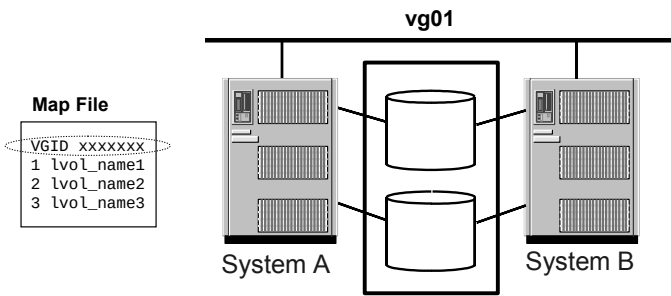
4-5. SLIDE: VG Export and Import -s Option



VG Export and Import -s Option

```
vgexport -v -p -s -m <mapfilename> <vg_name>
    create mapfile without removing the VG,
    and save the VGID for the vgimport command
```

```
vgimport -v -s -m <mapfilename> <vg_name>
    scan for disks that have the same VGID as in the mapfile
```



```
vgexport -v -p -s -m \
    /etc/lvmconf/vg01.map vg01
```

```
vgimport -v -s -m \
    /etc/lvmconf/vg01.map vg01
```

Student Notes

The `-s` option was introduced to the `vgexport/vgimport` commands with HP-UX 10.10. This option allows a volume group to be imported without having to specify the disk device file names.

This option is useful when working with large disk arrays, like the XP1024, where determining the disk device file names to use during a `vgimport` can be complex and difficult.

The `vgexport` command with the `-s` option causes the VGID of the volume group being exported to be saved in the map file. Storing the VGID in the map file allows the volume group to be easily imported without specifying disk device file names.

The `vgimport` command with the `-s` option uses the saved VGID in the map file to scan the disks for matches. Physical volumes containing a matching VGID are automatically included during the `vgimport`. This eliminates the need to explicitly specify which physical volumes are being imported.

4-6. SLIDE: LVM Issues with Serviceguard



LVM Issues with Serviceguard

Some important issues when using LVM within Serviceguard:

- The `/etc/lvmrc` file needs to be modified:
`AUTO_VG_ACTIVATE=0`
`custom_vg_activation { }`
- The Volume Group and Logical Volume major and minor numbers **MUST** be consistent
- The Cluster only needs to be halted to modify the cluster lock configuration and a very few other tasks

Student Notes

The `/etc/lvmrc` File

When using volume groups within Serviceguard, the volume groups should *not* be activated automatically at boot time. Serviceguard will activate a package's defined volume group when the package is started on the node where the package is running. Remember, only one node can activate a volume group at a time.

The default behavior of the boot process is to activate all known volume groups at boot time. This needs to be changed for cluster marked volume groups. To modify the default behavior at boot time, edit `/etc/lvmrc` and make the following changes:

- Set `AUTO_VG_ACTIVATE=0`. This will turn off the automatic activation of all volume groups at boot time.

Specify selected volume groups to be activated at boot time in the `custom_vg_activation` routine. All volume groups (except `vg00`) to be activated when the system boots should be listed here. These volume groups do not include cluster marked volume groups.

Consistent Major and Minor numbers

The major and minor numbers used to configure cluster volume groups must be the same across all systems in the cluster.

For example, if system A configures a shared volume group with minor number 0x010000 and system B configures the same volume group with minor number 0x020000, then a cross link error will occur when the cluster comes up.

All major and minor numbers for both volume groups and logical volumes must be consistent across all machines in the cluster.

Adding and Modifying LVM Configurations

Most LVM modifications can be made while the cluster is up and running. The only modifications that require the cluster to be brought down are modifying the configuration of the cluster lock disk and a few other very specific occurrences. All other LVM modifications, even modifications which require a `vgimport` again, can be done while the cluster is up.

When does `vgimport` need to be re-executed? The `vgimport` command needs to be re-executed any time new LVM device files are created. For example, adding a new logical volume or physical volume to an existing shared volume group would require a `vgexport` and `vgimport` on the other systems.

If the modification only changes an LVM characteristic, such as adding three-way mirroring or extending a logical volume, then no `vgimport` is necessary because no new device files were created.

For example, assume `vg01` is exclusively activated on system A, and a new logical volume is created in `vg01`. System B will have no device files for this new logical volume, so system B will have to `vgexport vg01` and then `vgimport vg01` in order to see the new device files for the newly created logical volume. The `vgimport` and `vgexport` are done while `vg01` is exclusively activated on system A. Note that to run `vgimport` and `vgexport`, the volume group never needed to be activated on system B.

4-7. LAB: Building a Shared Volume Group

Directions

The concepts in this lab were discussed in detail in the “Hands-On with Logical Volume Manager” class.

This lab will use these concepts to build a shared volume group that will later be used in a Serviceguard cluster.

Start here by identifying systems to be node1 and node2 .

node1: _____ (Please write the hostname here)

node2: _____ (Please write the hostname here)

Part 1: Building the Volume Group on Node1

1. On what will hereafter be designated **node1**, list all existing volume groups so that we do not attempt to use an already-assigned disk.
2. Next, make a directory for a new volume group called vg01 .
3. Create a control file for the new volume group.

4. Select two disks not already assigned to any other volume groups. These disks must be sharable between the nodes (*i.e. not internal disks*). These are the disks we will use for our new volume group.

```
# ioscan -fnC disk
```

When you have selected your disks, write the names of their device files here:

first disk, block file name

first disk, raw file name

second disk, block file name

second disk, raw file name

5. Prepare these two disks to be used within the new volume group.
6. Now create the new volume group.

Part 2: Setting Up a Logical Volume for Serviceguard

1. Continuing on node1, create a 200MB logical volume named `sglvol1`.
2. Mirror the new `sglvol1` logical volume.
3. Next, create a journal file system in the `sglvol1` logical volume.
4. Create a mount point directory called `/sgdata1`.
5. Mount the new file system to the mount point directory.
6. Finally, for demonstration purposes, copy some files into the file system so that, during a package failover, we can see these files "arrive" on the adoptive node. After copying the files, list the directory contents to see which files were copied.

Part 3: Importing LVM Configurations to a Second System

1. While still on **Node1**, export the LVM configuration information using "preview" mode.

NOTE: The following message is just a warning; notice that the mapfile is created anyway.

Warning: Volume group vname is still active.

2. Use ftp or rcp to transfer the map file over to the second system, **Node2**. If you choose rcp, you will likely need to add appropriate entries to your ~root/.rhosts files first.
3. Now, on **Node2**, create the directory for the volume group (to be imported).
4. Continuing on **Node2**, create the control file for the volume group.

5. Import the volume group.

NOTE: You can ignore the warning messages about different IDs.

Create a mount point directory. This must be the same as on Node1. After creating the mount point directory, activate the volume group in "read-only" mode, mount the file system in "read-only" mode, and view the files previously placed there, thus proving that the volume group / logical volume / file system really is shared. Also, create a backup copy of the LVM configuration. After viewing the files, unmount the file system and deactivate the volume group.

6. Congratulations on successfully building a shared volume group. We will use this volume group when we build our cluster in a later module.

4–8. LAB Solution: Building a Shared Volume Group

Directions

The concepts in this lab were discussed in detail in the “Hands-On with Logical Volume Manager” class.

This lab will use these concepts to build a shared volume group that will later be used in a Serviceguard cluster.

Start here by identifying systems to be node1 and node2 .

node1: _____ (Please write the hostname here)

node2: _____ (Please write the hostname here)

Part 1: Building the Volume Group on Node1

1. On what will hereafter be designated **node1**, list all existing volume groups so that we do not attempt to use an already-assigned disk.

Answer:

```
# strings /etc/lvmtab
```

2. Next, make a directory for a new volume group called vg01 .

Answer:

```
# mkdir /dev/vg01
```

3. Create a control file for the new volume group.

Answer:

```
# mknod /dev/vg01/group c 64 0x010000
```

Module 4
LVM for Serviceguard

4. Select two disks not already assigned to any other volume groups. These disks must be sharable between the nodes (i.e. not internal disks). These are the disks we will use for our new volume group.

```
# iotop -fnC disk
```

When you have selected your disks, write the names of their device files here:

first disk, block file name

first disk, raw file name

second disk, block file name

second disk, raw file name

5. Prepare these two disks to be used within the new volume group.

Answer:

```
# pvcreate [ -f ] /dev/rdisk/c_t_d_  
# pvcreate [ -f ] /dev/rdisk/c_t_d_
```

6. Now create the new volume group.

Answer:

```
# vgcreate vg01 /dev/dsk/c_t_d_ /dev/dsk/ c_t_d_
```


Part 2: Setting Up a Logical Volume for Serviceguard

1. Continuing on node1, create a 200MB logical volume named sglvol1.

Answer:

```
# lvcreate -L 200 -n sglvol1 vg01
```

2. Mirror the new sglvol1 logical volume.

Answer:

```
# lvextend -m 1 /dev/vg01/sglvol1
```

3. Next, create a journaled file system in the sglvol1 logical volume.

Answer:

```
# newfs -F vxfs /dev/vg01/rsglvol1
```

4. Create a mount point directory called /sgdata1.

Answer:

```
# mkdir /sgdata1
```

5. Mount the new file system to the mount point directory.

Answer:

```
# mount -F vxfs /dev/vg01/sglvol1 /sgdata1
```

6. Finally, for demonstration purposes, copy some files into the file system so that, during a package failover, we can see these files "arrive" on the adoptive node. After copying the files, list the directory contents to see which files were copied.

Answer:

```
# cp /sbin/lv* /sgdata1/.  
# ll /sgdata1
```

Part 3: Importing LVM Configurations to a Second System

1. While still on **Node1**, export the LVM configuration information using "preview" mode.

NOTE: The following message is just a warning; notice that the mapfile is created anyway.

Warning: Volume group vname is still active.

Answer:

```
# vgexport -v -p -s -m /tmp/vg01.map vg01
```

2. Use ftp or rcp to transfer the map file over to the second system, **Node2**. If you choose rcp, you will likely need to add appropriate entries to your `~root/.rhosts` files first.

Answer:

```
# rcp /tmp/vg01.map <Node2>:/tmp/.
```

3. Now, on **Node2**, create the directory for the volume group (to be imported).

Answer:

```
# mkdir /dev/vg01
```

4. Continuing on **Node2**, create the control file for the volume group.

Answer:

```
# mknod /dev/vg01/group c 64 0x010000
```

5. Import the volume group.

NOTE: You can ignore the warning messages about different IDs.

Answer:

```
# vgimport -v -s -m /tmp/vg01.map vg01
```

6. Create a mount point directory. This must be the same as on Node1. After creating the mount point directory, activate the volume group in "read-only" mode, mount the file system in "read-only" mode, and view the files previously placed there, thus proving that the volume group / logical volume / file system really is shared. Also, create a backup copy of the LVM configuration. After viewing the files, unmount the file system and deactivate the volume group.

Answer:

```
# mkdir /sgdata1
# vgchange -a r vg01
# mount -F vxfs -o ro /dev/vg01/sglvol1 /sgdata1
# ll /sgdata1
# vgcfgbackup vg01    ← Create a backup copy of the LVM configuration
# umount /sgdata1
# vgchange -a n vg01
```

7. Congratulations on successfully building a shared volume group. We will use this volume group when we build our cluster in a later module.

Module 5 — Cluster Concepts and Configuration

Objectives

Upon completion of this module, you will be able to do the following:

- Describe the differences between heartbeat LAN cards, stationary LAN cards, and standby LAN cards.
- Configure a Serviceguard cluster using a cluster lock disk or using Quorum Server technology.
- Configure the heartbeat-polling interval for a Serviceguard cluster.
- Describe three functions performed by the Serviceguard cluster manager daemon.
- Describe the major requirement needed to bring up a Serviceguard cluster.
- Describe Access Control Policies for a cluster
- Implement the systematic procedure for defining, configuring, and bringing up a Serviceguard cluster.

5-1. SLIDE: Definition of a Cluster



Definition of a Cluster

- A Cluster is a collection of nodes which cooperate together to improve the availability of applications running on the nodes.
- Two major benefits of a cluster during downtime events ensure:
 - High Availability when another node takes over. (Unplanned)
 - The cluster can be used for rolling upgrades. (Planned)

Student Notes

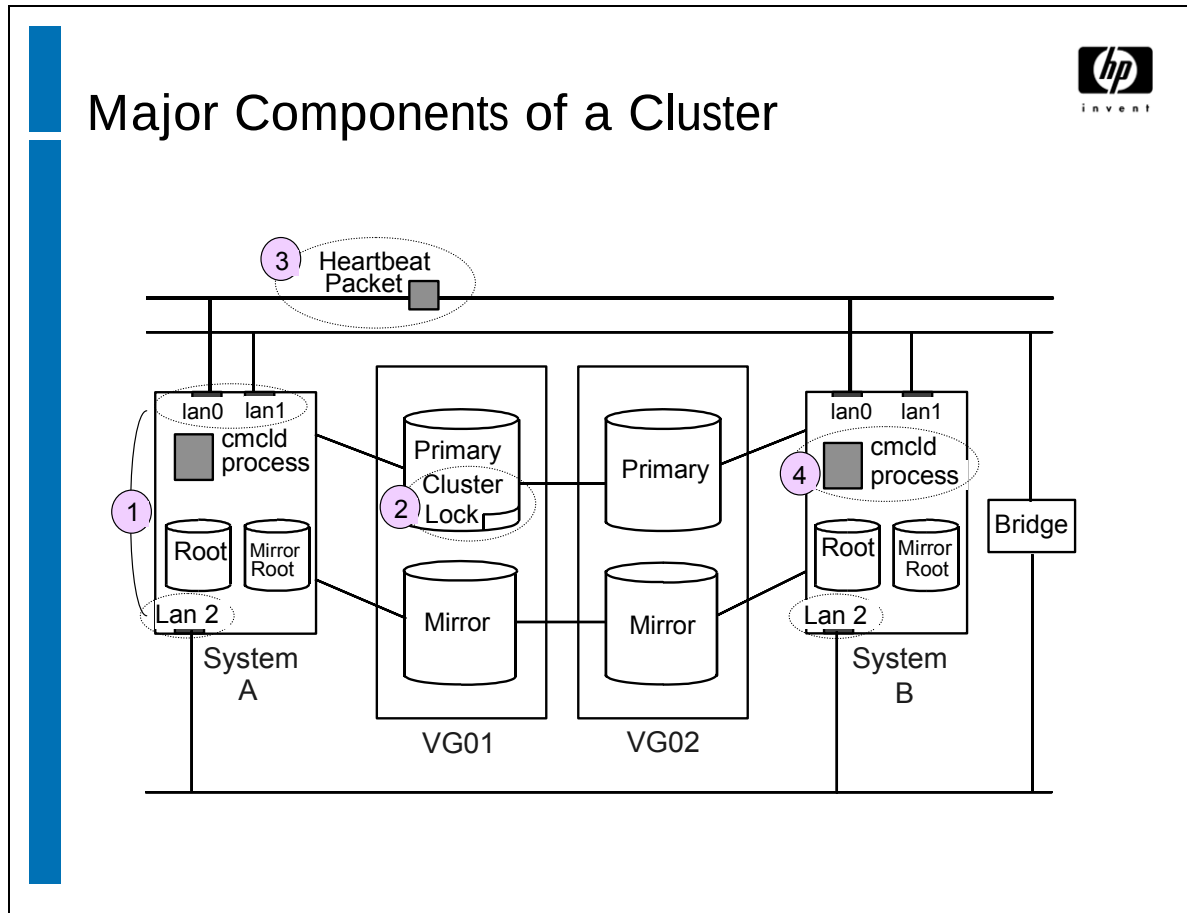
A Serviceguard cluster is a group of networked HP 9000 business servers (nodes) having sufficient redundancy of software and hardware that a single point of failure will not significantly disrupt service. HP 9000 workstations cannot be clustered nodes.

In order to provide a high level of availability, a typical cluster uses redundant system components, like two LANs, two LAN cards, two SCSI cards, and mirrored or RAID disks.

In addition to hardware redundancy, the software must support the capability to transfer applications to another system in the event of a failure. Serviceguard provides the following support:

- In the case of a system failure (CPU, memory, kernel hangs), applications are transferred from the failed system to a functioning system within the cluster in a minimum amount of time.
- In the case of LAN failure, the IP addresses are transferred to a standby LAN card.
- For software failures, the application can be restarted on the same node or switched to another node within the cluster.

5-2. SLIDE: Major Components of a Cluster

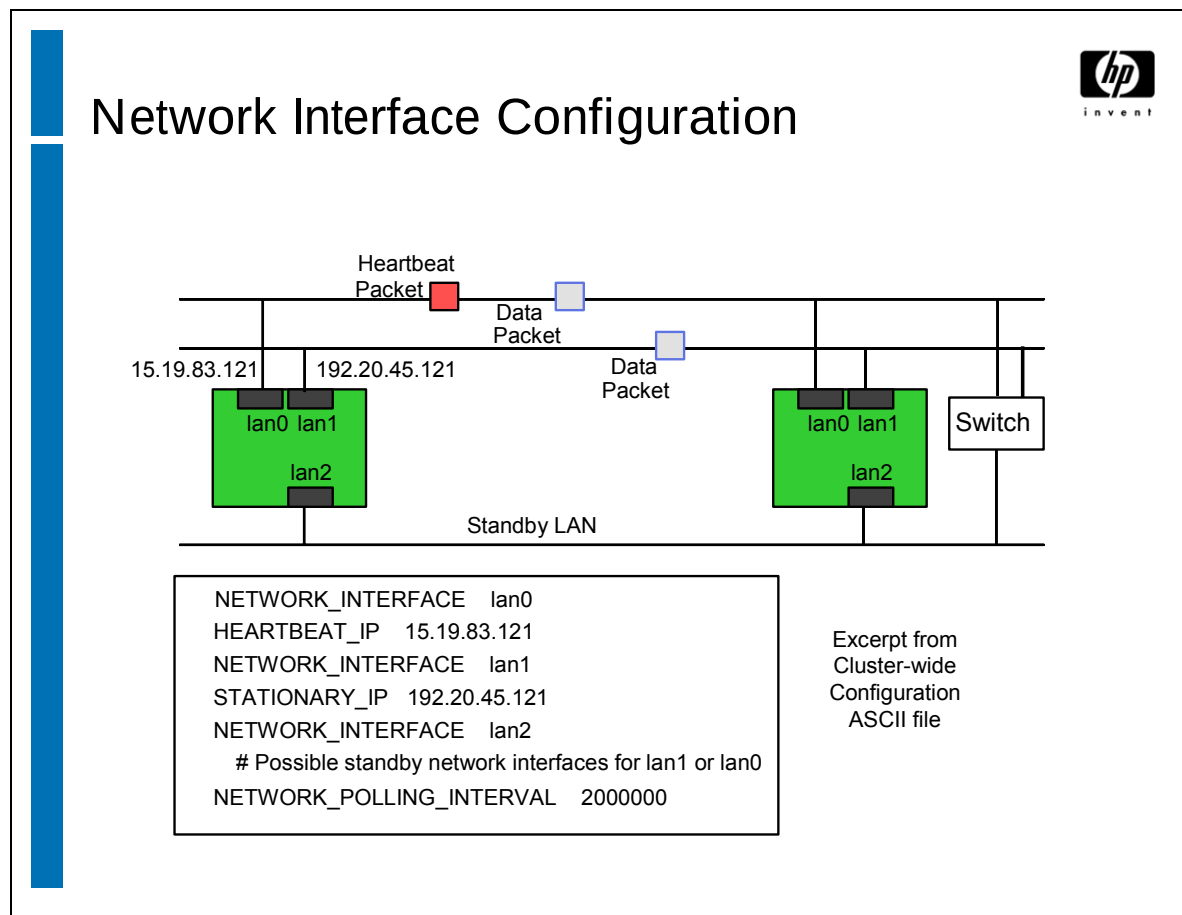


Student Notes

The slide shows a sample cluster along with all the major components which make up the cluster. The major components discussed in this module include the following:

1. **LAN interface cards** can be used for data traffic, heartbeat traffic, or for standby purposes.
2. The **cluster lock disk** (shown here), or a Quorum Server (discussed later) is used only during cluster reformation and only for specific situations.
3. **Heartbeat packets** are sent between the nodes to maintain status information.
4. The **cmcld** process is the main Serviceguard process. It is most often referred to as the cluster daemon and it runs on every node within the cluster.
5. Each system in the cluster contains its own root disk. The root disk is mirrored on both systems adding additional high availability.

5-3. SLIDE: Network Interface Configuration



Student Notes

The slide shows the three different ways a LAN card can be configured within the Serviceguard environment.

The bottom of the slide shows what the configurations would look like in the cluster configuration file.

HEARTBEAT_IP

LAN cards configured as HEARTBEAT_IP are used for heartbeat traffic and potentially for data traffic. A minimum of one HEARTBEAT_IP must be defined for each node in the cluster.

A card configured with a heartbeat IP address will be used by the `cmcl` process to communicate with other nodes within the cluster. One such communication is heartbeat messages. These messages are sent by each node in the cluster to the other nodes to indicate they are still alive.

Conversely, each node looks for these heartbeat messages from the other nodes. If heartbeat messages are not received within a prescribed amount of time, the cluster will try to reform minus the node not sending heartbeat messages.

Note that IPv6 heartbeat is not supported, but an IPv6 LAN IS supported within a cluster.

STATIONARY_IP

LAN cards configured as STATIONARY_IP are used for data traffic only. Serviceguard monitors these LAN cards; however, no heartbeats or cluster communication packets will travel across the card.

It is recommended that STATIONARY_IP addresses only be used when two or more heartbeat IP addresses already exist. For redundancy of heartbeat messages, it is better to configure a LAN card as HEARTBEAT_IP to provide redundancy and to protect against network congestion on other LANs.

The term **stationary** is used to indicate that the IP address will never leave the node. However, the IP address could failover to a standby card within the node.

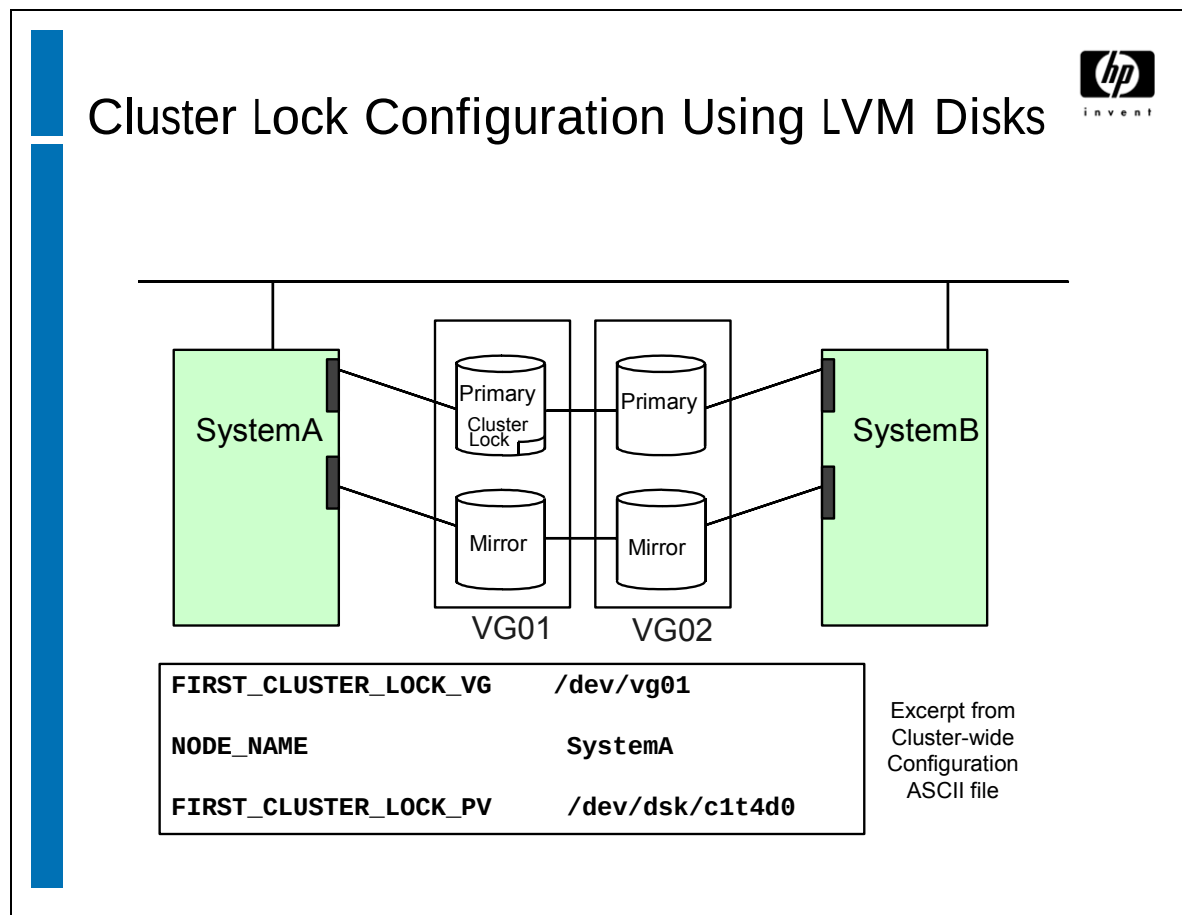
Standby (Indicated by a Blank Definition)

LAN cards with no defined configuration will automatically be used as standby LAN cards. The term standby indicates the card will be used should one of the configured LAN cards (HEARTBEAT_IP or STATIONARY_IP) fail.

In the event of a failure, all the IP addresses on the failed LAN card will be moved over to the standby LAN card. Standby LAN cards must be on the same physical network as the LAN card they are backing up. This is to allow all other systems to continue communications with the local system after the IP addresses fail over.

In example on the slide, lan0 is being backed up by lan2 on both systems. The standby LAN card cannot be a backup for lan1, because lan2 and lan1 are not on the same physical network.

5-4. SLIDE: Cluster Lock Configuration Using LVM Disks



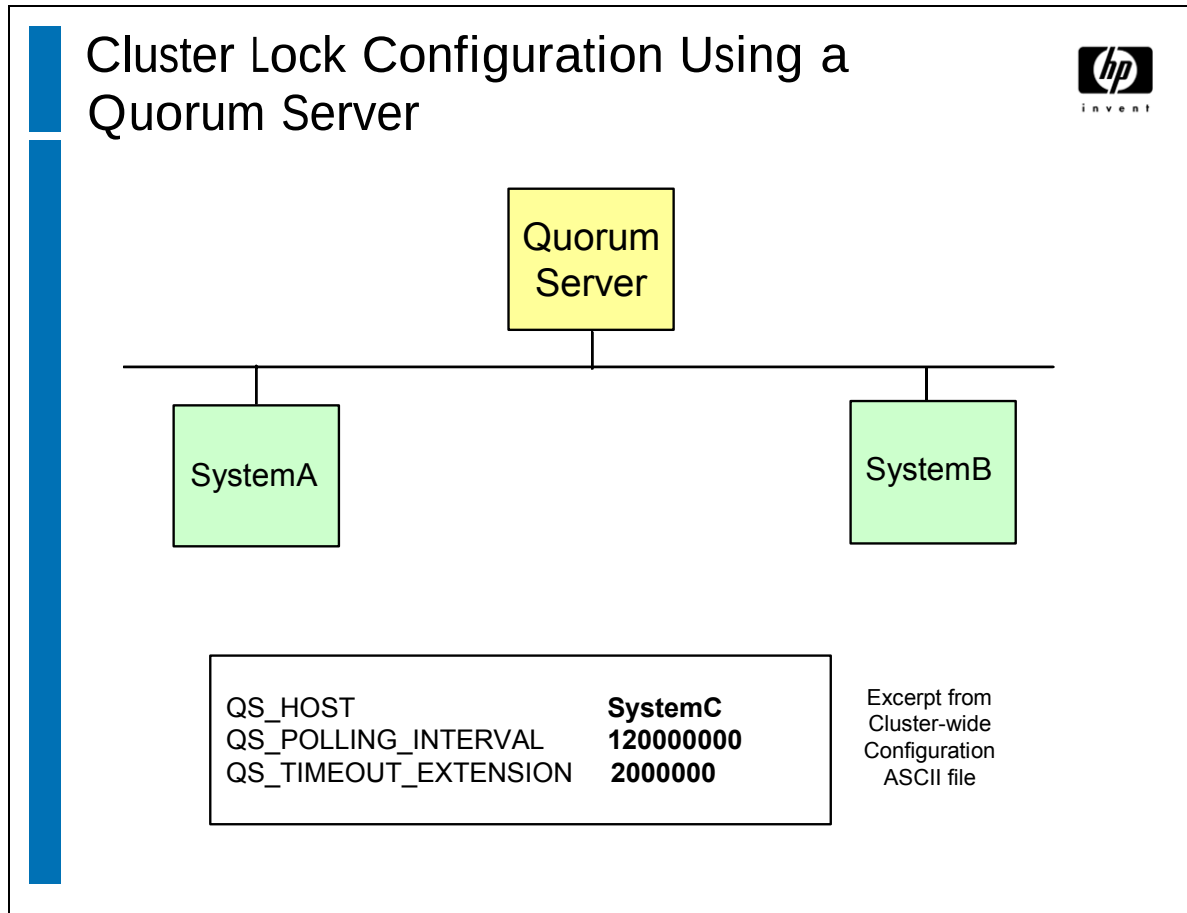
Student Notes

The cluster lock is a disk located in a volume group shared by all nodes in the cluster. The cluster lock volume group and physical volume are identified in the cluster configuration file.

The cluster lock is used as a tiebreaker only for situations in which the running cluster fails and, in the attempt to reform the cluster, the cluster is split into two subclusters of equal size. Each subcluster will attempt to acquire the cluster lock. The subcluster that gets the cluster lock will form the new cluster. The subcluster that does not get the cluster lock will panic in order to allow existing applications to failover to the new cluster.

In the example on the slide, assume that communications between SystemA and SystemB fails. The failure of the cluster would yield two one-node subclusters. Because each subcluster contains 50 percent of the nodes, both subclusters would try to obtain the cluster lock disk as a means of breaking the tie to determine which subcluster would get to form the new cluster. The subcluster that does not get the cluster lock disk panics.

5-5. SLIDE: Cluster Lock Configuration Using a Quorum Server



Student Notes

An available system, external to the Serviceguard Cluster, could be used to provide Cluster Lock services instead of the traditional LVM disks. This external system is configured as a "Quorum Server" by first using `swinstall` to install the Quorum Server software, and then configuring the system. We will setup a Quorum server in the lab for this module.

Starting with Quorum Server version 2.0, the software will run on both HP-UX and Linux clusters. Additionally, the Quorum Server can be configured as a package in a cluster different than the cluster which is requesting Quorum Server functionality.

Other than the above discussion, all facets of Cluster Lock services are the same as discussed on the previous page.

When configuring your cluster to use a quorum server, the following lines are added to the cluster ascii file:

```
QS_HOST                quorum_server_hostname
QS_POLLING_INTERVAL    120000000
QS_TIMEOUT_EXTENSION    2000000                # optional
```

The `QS_POLLING_INTERVAL`, in microseconds, is the frequency in which the cluster nodes check for TCP connectivity to the Quorum Server

The `QS_TIMEOUT_EXTENSION` relates to the amount of time needed to access the Quorum Server when polling and when a lock is needed. If your environment has added network latency, then it may be necessary to configure this parameter to allow more time to access the Quorum Server before the request (a TCP connectivity check or a lock request) times out. By default, there is a Quorum Server timeout that is calculated using the current heartbeat and node timeout values. You would use the `QS_TIMEOUT_EXTENSION` parameter if you need more than the default timeout. Extending this value will increase the cluster reformation time.

Configuring a Quorum Server

1. Choose a system (not within the cluster) to be the Quorum Server.
2. Install Quorum Server software (Product B8467BA) on this server.
3. Create an authorization file, which allows specific hosts to obtain quorum services. This file must be named `/etc/cmcluster/qs_authfile` and will contain the list of all cluster nodes that will access quorum services from the quorum server. Use one line per node.
4. By default, quorum service run-time messages go to `stdout` and `stderr`, so you may want to redirect these messages to a logfile. Create the following directory for this logfile: `/var/adm/qs`
5. Edit the `/etc/inittab` file and add the following line at the bottom:

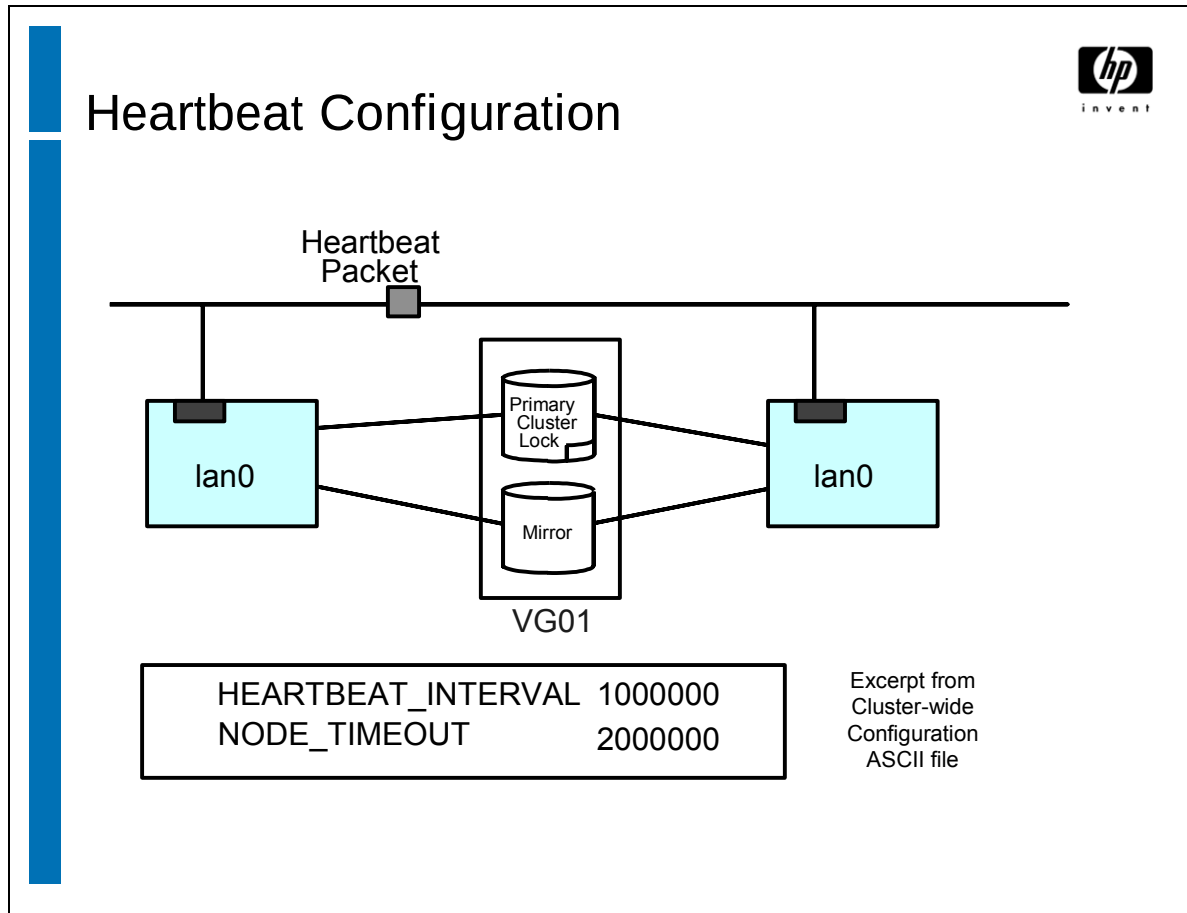
```
qs:345:respawn:/usr/sbin/qs  >> /var/adm/qs/qs.log 2>&1
```
6. Start the Quorum Server as follows: `# init q`

Note: After starting the quorum server software, if changes are made to `/etc/cmcluster/qs_authfile`, the file will need to be re-read.

This is done by executing the following command: `# /usr/sbin/qs -update`

More information on the `/usr/sbin/qs` command can be found in the man pages (on the system running the Quorum Server software).

5-6. SLIDE: Heartbeat Configuration



Student Notes

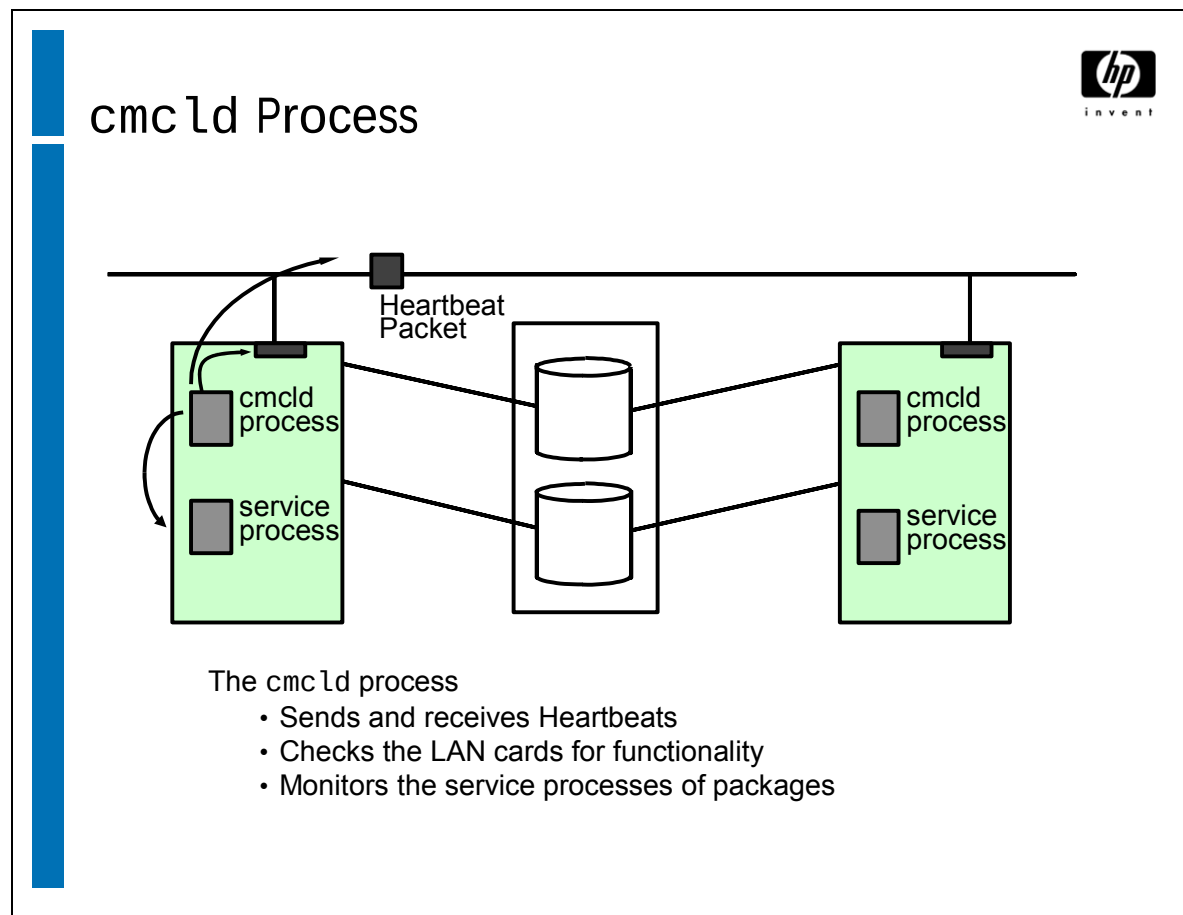
Heartbeat packets are the primary mechanism used by nodes within a cluster to communicate status information.

Each node sends a heartbeat message to the cluster coordinator at every HEARTBEAT_INTERVAL number of microseconds specified in the cluster configuration file. The cluster coordinator looks for this message from each node, and if it is not received within NODE_TIMEOUT time, then the cluster is reformed minus the node no longer sending heartbeat messages.

The NODE_TIMEOUT parameter defaults to 2 seconds. However, for a significant portion of installations, a setting of 5 to 8 seconds is more appropriate. The maximum value recommended for NODE_TIMEOUT is 30 seconds.

If heartbeat and data are sent over the same LAN subnet, data congestion may cause heartbeat packets not to reach the cluster coordinator. This could cause a cluster reformation even though all nodes in the cluster are healthy. To prevent this situation, it is recommended that multiple HEARTBEAT_IPs be configured within a cluster.

5-7. SLIDE: cmcld Process



Student Notes

The cluster manager daemon (cmcld) is the primary Serviceguard process. This process initializes the cluster, monitors the health of the cluster, and recognizes when a failure within the cluster occurs. This process also performs the reformation of the cluster when a node joins or leaves the cluster.

Every node in the cluster will have a cmcld process. During the formation of the cluster, one of the cmcld processes will be selected to be the **cluster coordinator**. The cluster coordinator has the additional responsibility to be the central focal point for all heartbeat messages.

5-8. SLIDE: Cluster Formation Requirements



Cluster Formation Requirements

Cluster Lock Mechanism

- Required for 2 nodes (Quorum Server or LVM Disks)
- Recommended for 3 or 4 nodes (Quorum Server or LVM Disks)
- Optional for 5 or more nodes (Quorum Server only)
- Should be on separate power sources from other systems in cluster

Node Attendance

- By default, 100% node attendance is required for initial cluster formation

Network Requirements

- All nodes in the cluster must be on same subnet
- Configured LAN cards in the same system MUST be on separate subnets
- At least one HEARTBEAT_IP network is configured

Student Notes

Cluster Lock Disk

A cluster lock disk is required for the formation of a two-node cluster. Without a cluster lock disk, the cluster would not be able to reform should one node fail, causing the entire cluster to go down. Cluster lock functionality may be implemented by using either LVM disks or using a Quorum server.

With three or four nodes, a cluster lock is recommended but not required. For three nodes, there would never be a situation where the nodes are divided 50/50. However, if a node is pulled out of the cluster for maintenance purposes, then the cluster lock would be used should one of the remaining two nodes in the cluster fail.

For four nodes, the cluster lock disk would be used when the nodes are separated into two, 2-node sub clusters. The lock is not required, because the chance of being split into two, equally sized halves is very small for most environments.

For five or more nodes, a cluster lock using LVM disks is not supported. While a Quorum Server could be used, it would be optional.

100 Percent Node Attendance

To initially bring up any cluster of any size, 100 percent node attendance is required. Without this requirement, the same cluster could potentially form twice with two different sets of nodes.

When a request is made by one of the nodes to form/bring up the cluster, all nodes in the cluster are notified of the request. If one of the nodes already has the cluster up, the other nodes will join the already existing cluster. If all nodes return confirmation of the request and no nodes have already formed a cluster, a new cluster formation will be performed. If however, all nodes do not return confirmation of the request, then the formation of the cluster will wait for the outstanding confirmations to arrive. The amount of time that the nodes wait for outstanding confirmations is `AUTO_START_TIMEOUT`, which is defined in the cluster configuration file.

Network Requirements

In order to form a new cluster, every node must have at least one `HEARTBEAT_IP` address configured. These `HEARTBEAT_IP` addresses must be on the same subnet (if a node has two `HEARTBEAT_IP` addresses, then only one has to be on the common subnet).

If a node has multiple LAN cards configured, the LAN cards must all be on different subnets. Routing protocols get confused when two LAN cards in the same system are configured for the same subnet.

5-9. SLIDE: Steps to Configure a Cluster



Steps to Configure a Cluster

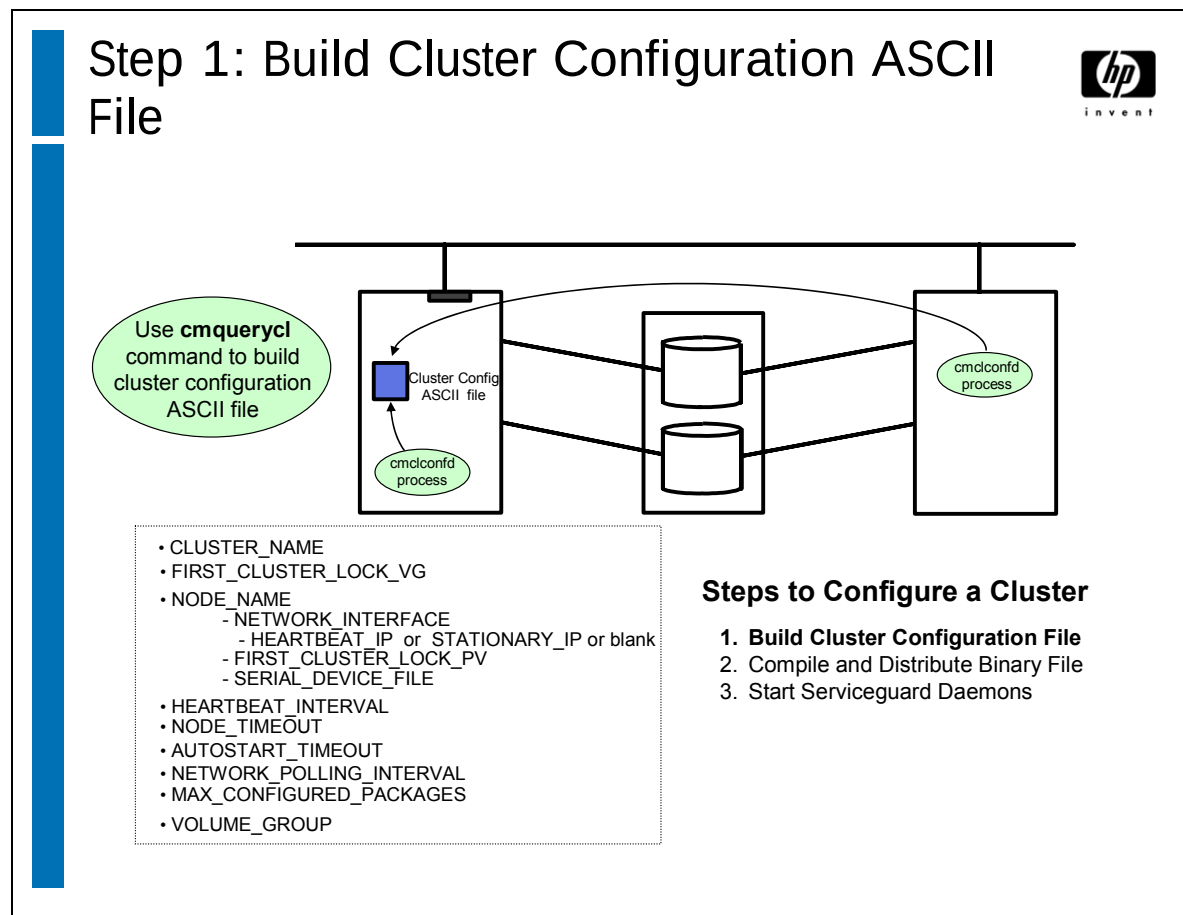
1. Build cluster configuration file
`cmquerycl -v [-q QS_HOST] -C cmclconfig.ascii \`
`-n node1 -n node2...`
2. Compile and distribute binary file
`cmcheckconf -v -C cmclconfig.ascii`
`cmapplyconf -v -C cmclconfig.ascii`
3. Start the Serviceguard daemons
`cmrunc1 -v`
4. If necessary, halt the cluster
`cmhaltcl -v`

Student Notes

To configure a Serviceguard cluster, the three basic steps are:

1. **Build the cluster configuration ASCII file.** The file that the user creates defines the nodes, the disks, the LAN cards, and any other resources that are to be part of the cluster. The command used to help build the cluster configuration ASCII file is called `cmquerycl`. If using a Quorum Server to provide cluster lock services, be sure to have configured the QS software on another host. Then, in the `cmquerycl` command, include the `-q` option, specifying the `QS_HOST`.
2. **Compile and distribute the cluster binary file.** Once the cluster configuration ASCII file is created, it needs to be compiled into a binary format that the Serviceguard daemons can access efficiently. This compiled file is then distributed automatically to all other nodes in the cluster. The command used to compile and distribute the binary configuration file is called `cmapplyconf`.
3. **Start the Serviceguard daemons.** Once the binary file is distributed, the Serviceguard daemon, `cmclld`, can be started. This daemon *cannot* be started directly, and should be started only through the `cmrunc1` command.

5-10. SLIDE: Step 1: Build Cluster Configuration ASCII File



Student Notes

The first step in building the cluster is to create a “cluster-wide configuration ASCII file” which describes the overall contents of the cluster. The cluster-wide configuration ASCII file should be created in the `/etc/cmcluster` directory.

The `cmquerycl` command is used to gather the node-specific information and to create a template. The syntax for the `cmquerycl` command is:

```
# cmquerycl -v -C cmclconfig.ascii -n node1 -n node2
```

Once the template is created, it can easily be edited and customized with any editor. At a minimum, the `CLUSTER_NAME` and `NODE_TIMEOUT` variables should be modified in the template. Other values should be modified as appropriate.

The fields in the cluster configuration template ASCII file are described below:

<i>CLUSTER_NAME</i>	The name of the cluster as it will appear in the output of <code>cmviewcl</code> and other commands. This name must be unique. The cluster name can contain up to 40 characters.
<i>FIRST_CLUSTER_LOCK_VG</i>	The volume group from which the cluster lock disk is selected.
<i>NODE_NAME</i>	The hostname of a system that will be a node in the cluster. This field must be defined for each system in the cluster. The node name can contain up to 31 characters.
<i>NETWORK_INTERFACE</i>	The network interface as listed by <code>lanscan</code> (e.g. <code>lan1</code>).
<i>HEARTBEAT_IP</i>	The IP address of the LAN interface, which will carry heartbeat messages. A minimum of one <code>HEARTBEAT_IP</code> must be defined.
<i>FIRST_CLUSTER_LOCK_PV</i>	The name of the physical volume within the <code>FIRST_CLUSTER_LOCK_VG</code> that will have the cluster lock written on it. The physical volume group identifier can contain up to 40 characters.
<i>SERIAL_DEVICE_FILE</i>	The name of the device file that corresponds to the RS232 serial port that will carry the serial heartbeat that will be used only in the case where the heartbeat communications is interrupted. The device name can contain up to 40 characters.
<i>HEARTBEAT_INTERVAL</i>	The interval between transmissions of heartbeat messages from one node to the cluster coordinator. The default value is 1000000 microseconds (one second).
<i>NODE_TIMEOUT</i>	The time (in microseconds) after which a node may decide another node has become unavailable and thereby initiate a cluster reformation. The default value is 2000000 microseconds (2 seconds). A good rule of thumb is to have 2-3 heartbeats within 1 <code>NODE_TIMEOUT</code> .
<i>AUTOSTART_TIMEOUT</i>	The amount of time a node waits for other nodes to become available when bringing up the cluster. All nodes must be present in order for the cluster to come up. The default value is 600000000 microseconds (10 minutes)

<i>NETWORK_POLLING_INTERVAL</i>	The frequency at which the cluster-identified LAN cards are checked by the cluster manager daemon (cmclld). The default is 2000000 microseconds (2 seconds).
<i>NETWORK_FAILURE_DETECTION</i>	Determines how LAN card failures are handled. If set to INONLY_OR_INOUT, the LAN card will be considered down when the inbound message count stops increasing –or– when both inbound & outbound message counts stop increasing. If set to INOUT, both inbound and outbound message counts must stop increasing for the LAN card to be considered down. INOUT is the default.
<i>MAX_CONFIGURED_PACKAGES</i>	The maximum number of packages that can be configured in the cluster. The default is 150, as is the absolute maximum.
<i>Access Control Policy</i>	By default, Serviceguard configuration can only be performed by the system administrator. Access control policies allow the root user to delegate certain tasks to non-root users. The cluster ascii file has entries for USER_NAME, USER_HOST, and USER_ROLE. USER_NAME is the user to whom certain access will be granted. USER_HOST is where USER_NAME can issue Serviceguard commands (not necessarily a node in the cluster). USER_ROLE defines USER_NAME's privileges (FULL_ADMIN, PACKAGE_ADMIN, or MONITOR). Note: None of the users granted access control are allowed to configure the cluster; this activity is limited only to the root user.
<i>VOLUME_GROUP</i>	The volume groups which the Serviceguard cluster uses. The volume group name can contain up to 39 characters.

5-11. SLIDE: Cluster Management Options



Cluster Management Options

Role-Based Access Control

- By default, only the root user can access/control the cluster
- Other users (optionally from other hosts) can be allowed certain privileges

Access Control Policy Parameters

- USER_NAME ANY_USER –or– a list of specific users
- USER_HOST Where the user can execute SG commands
- USER_ROLE MONITOR, PACKAGE_ADMIN, or FULL_ADMIN

Limitations

- Only the root user can configure a cluster or packages

Student Notes

Beginning with Serviceguard A.11.16, non-root users can manage/monitor a cluster and/or the packages within it. In the cluster ascii file, there are several options for Access Control. Specifically, USER_NAME, USER_HOST, and USER_ROLE.

For each of these, the options follow:

The first line must be USER_NAME, second USER_HOST, and third USER_ROLE.

1. USER_NAME can either be ANY_USER, or a maximum of 8 login names from the /etc/passwd file on user host.
2. USER_HOST is where the user can issue Serviceguard commands. If using Serviceguard Manager, it is the COM (Cluster Object Manager) server. Choose one of these three values: ANY_SERVICEGUARD_NODE, or (any) CLUSTER_MEMBER_NODE, or a specific node. For node, use the official hostname from domain name server, and not an IP address or fully qualified name.

Cluster Concepts and Configuration

3. USER_ROLE must be one of these three values:
 - MONITOR: read-only capabilities for the cluster and packages
 - PACKAGE_ADMIN: MONITOR, plus administrative commands for packages in the cluster
 - FULL_ADMIN: MONITOR and PACKAGE_ADMIN plus the administrative commands for the cluster.

Access control policy does not set a role for configuration capability. To configure a cluster or a package, a user must log on to one of the cluster's nodes as root (UID=0). Access control policy cannot limit root users' access.

MONITOR and FULL_ADMIN can only be set in the cluster configuration file, and they apply to the entire cluster. PACKAGE_ADMIN can be set in the cluster or a package configuration file. If set in the cluster configuration file, PACKAGE_ADMIN applies to all configured packages. If set in a package configuration file, which will be discussed in a later module, PACKAGE_ADMIN applies to that package only.

Conflicting or redundant policies will cause an error while applying the configuration, and will stop the process.

Example: to configure a role for user oracle from node rx26u891 to administer a cluster and all its packages, enter:

```
USER_NAME  oracle
USER_HOST  rx26u891
USER_ROLE  FULL_ADMIN
```

Note: Access Policies are not supported in a mixed revision cluster. If updating from an earlier version of Serviceguard, and if /etc/cmcluster/clcmnode list was used to give non-root users access, the update process will convert old cmclnode list entries into Access Control Policies when every nodes in the cluster is finished updating. Additionally, all are given the role of MONITOR in the Cluster ASCII file.

5-12. TEXT PAGE: The cmquerycl command

Special Note

Here is the actual screen output resulting from the cmquerycl command.
This output was obtained while using **version 11.17** of Serviceguard.

```
root@rx26u230 [/etc/cmcluster]
# cmquerycl -v -C cmclconfig.ascii -n rx26u230 -n rx26u231
Looking for other clusters ... Done
Gathering storage information
Found 79 devices on node rx26u230
Found 79 devices on node rx26u231
Analysis of 158 devices should take approximately 9 seconds
0%----10%----20%----30%----40%----50%----60%----70%----80%----90%----100%
Found 2 volume groups on node rx26u230
Found 2 volume groups on node rx26u231
Analysis of 4 volume groups should take approximately 1 seconds
0%----10%----20%----30%----40%----50%----60%----70%----80%----90%----100%
Note: Disks were discovered which are not in use by either LVM or VxVM.
      Use pvcreate(1M) to initialize a disk for LVM or,
      use vxdiskadm(1M) to initialize a disk for VxVM.
Gathering network information
Beginning network probing
Completed network probing
Network interface lan0 on node rx26u230 couldn't talk to itself.
Network interface lan0 on node rx26u231 couldn't talk to itself.
```

```
Node Names:    rx26u230
               rx26u231
```

Bridged networks (local node information only - full probing was not performed):

```
1      lan1      (rx26u230)
      lan2      (rx26u230)
      lan3      (rx26u230)

2      lan1      (rx26u231)
      lan2      (rx26u231)
      lan3      (rx26u231)
```

IP subnets:

IPv4:

```
10.10.0.0      lan1      (rx26u230)
               lan1      (rx26u231)
```

IPv6:

Possible Heartbeat IPs:

Module 5
Cluster Concepts and Configuration

10.10.0.0	10.10.48.130	(rx26u230)
	10.10.48.131	(rx26u231)

Possible Cluster Lock Devices:

/dev/dsk/c20t0d7	/dev/vg01	66 seconds
------------------	-----------	------------

LVM volume groups:

/dev/vg00	rx26u230
-----------	----------

/dev/vg01	rx26u230
	rx26u231

/dev/vg00	rx26u231
-----------	----------

LVM physical volumes:

/dev/vg00		
/dev/dsk/c2t1d0s2	0/1/1/0.1.0	rx26u230

/dev/vg01		
/dev/dsk/c20t0d7	255/255/0/0.7	rx26u230

/dev/dsk/c20t0d7	255/255/0/0.7	rx26u231
------------------	---------------	----------

/dev/vg00		
/dev/dsk/c2t1d0s2	0/1/1/0.1.0	rx26u231

LVM logical volumes:

Volume groups on rx26u230:

/dev/vg00/lvol1	FS MOUNTED	/stand
/dev/vg00/lvol2		
/dev/vg00/lvol3	FS MOUNTED	/
/dev/vg00/lvol4	FS MOUNTED	/home
/dev/vg00/lvol5	FS MOUNTED	/ora1
/dev/vg00/lvol6	FS MOUNTED	/tmp
/dev/vg00/lvol7	FS MOUNTED	/opt
/dev/vg00/lvol8	FS MOUNTED	/usr
/dev/vg00/lvol9	FS MOUNTED	/var

Volume groups on rx26u231:

/dev/vg00/lvol1	FS MOUNTED	/stand
/dev/vg00/lvol2		
/dev/vg00/lvol3	FS MOUNTED	/
/dev/vg00/lvol4	FS MOUNTED	/home
/dev/vg00/lvol5	FS MOUNTED	/ora1
/dev/vg00/lvol6	FS MOUNTED	/tmp
/dev/vg00/lvol7	FS MOUNTED	/opt
/dev/vg00/lvol8	FS MOUNTED	/usr
/dev/vg00/lvol9	FS MOUNTED	/var

Writing cluster data to cmclconfig.ascii.

root@rx26u230 [/etc/cmcluster]

#

5-13. TEXT PAGE: Sample Cluster Configuration ASCII File Using LVM Disks for Cluster Lock Services

The following is the cluster-wide ASCII configuration file
using Serviceguard version A.11.17, without a Quorum Server
(i.e., using LVM disks for cluster lock functionality)

```
# *****  
# ***** HIGH AVAILABILITY CLUSTER CONFIGURATION FILE *****  
# ***** For complete details about cluster parameters and how to *****  
# ***** set them, consult the Serviceguard manual. *****  
# *****
```

```
# Enter a name for this cluster. This name will be used to identify the  
# cluster when viewing or manipulating it.
```

```
CLUSTER_NAME          cluster1
```

```
# Cluster Lock Parameters  
# The cluster lock is used as a tie-breaker for situations  
# in which a running cluster fails, and then two equal-sized  
# sub-clusters are both trying to form a new cluster. The  
# cluster lock may be configured using only one of the  
# following alternatives on a cluster:  
#         the LVM lock disk  
#         the quorum server  
#  
# Consider the following when configuring a cluster.  
# For a two-node cluster, you must use a cluster lock. For  
# a cluster of three or four nodes, a cluster lock is strongly  
# recommended. For a cluster of more than four nodes, a  
# cluster lock is recommended. If you decide to configure  
# a lock for a cluster of more than four nodes, it must be  
# a quorum server.  
  
# Lock Disk Parameters. Use the FIRST_CLUSTER_LOCK_VG and  
# FIRST_CLUSTER_LOCK_PV parameters to define a lock disk.  
# The FIRST_CLUSTER_LOCK_VG is the LVM volume group that  
# holds the cluster lock. This volume group should not be  
# used by any other cluster as a cluster lock device.  
  
# Quorum Server Parameters. Use the QS_HOST, QS_POLLING_INTERVAL,  
# and QS_TIMEOUT_EXTENSION parameters to define a quorum server.  
# The QS_HOST is the host name or IP address of the system  
# that is running the quorum server process. The  
# QS_POLLING_INTERVAL (microseconds) is the interval at which  
# Serviceguard checks to make sure the quorum server is running.  
# The optional QS_TIMEOUT_EXTENSION (microseconds) is used to increase  
# the time interval after which the quorum server is marked DOWN.  
#  
# The default quorum server timeout is calculated from the  
# Serviceguard cluster parameters, including NODE_TIMEOUT and  
# HEARTBEAT_INTERVAL. If you are experiencing quorum server  
# timeouts, you can adjust these parameters, or you can include  
# the QS_TIMEOUT_EXTENSION parameter.
```

Cluster Concepts and Configuration

```
#
# The value of QS_TIMEOUT_EXTENSION will directly effect the amount
# of time it takes for cluster reformation in the event of failure.
# For example, if QS_TIMEOUT_EXTENSION is set to 10 seconds, the cluster
# reformation will take 10 seconds longer than if the QS_TIMEOUT_EXTENSION
# was set to 0. This delay applies even if there is no delay in
# contacting the Quorum Server. The recommended value for
# QS_TIMEOUT_EXTENSION is 0, which is used as the default
# and the maximum supported value is 30000000 (5 minutes).
#
# For example, to configure a quorum server running on node
# "qshost" with 120 seconds for the QS_POLLING_INTERVAL and to
# add 2 seconds to the system assigned value for the quorum server
# timeout, enter:
#
# QS_HOST qshost
# QS_POLLING_INTERVAL 120000000
# QS_TIMEOUT_EXTENSION 2000000
```

```
FIRST_CLUSTER_LOCK_VG          /dev/vg01
```

```
# Definition of nodes in the cluster.
# Repeat node definitions as necessary for additional nodes.
# NODE_NAME is the specified nodename in the cluster.
# It must match the hostname and both cannot contain full domain name.
# Each NETWORK_INTERFACE, if configured with IPv4 address,
# must have ONLY one IPv4 address entry with it which could
# be either HEARTBEAT_IP or STATIONARY_IP.
# Each NETWORK_INTERFACE, if configured with IPv6 address(es)
# can have multiple IPv6 address entries(up to a maximum of 2,
# only one IPv6 address entry belonging to site-local scope
# and only one belonging to global scope) which must be all
# STATIONARY_IP. They cannot be HEARTBEAT_IP.
```

```
NODE_NAME                rx26u230
NETWORK_INTERFACE       lan1
HEARTBEAT_IP            10.10.48.130
NETWORK_INTERFACE       lan2
NETWORK_INTERFACE       lan3
FIRST_CLUSTER_LOCK_PV /dev/dsk/c20t0d7
# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE     /dev/tty0p0
```

```
# Possible standby Network Interfaces for lan1: lan2,lan3.
```

```
NODE_NAME                rx26u231
NETWORK_INTERFACE       lan1
HEARTBEAT_IP            10.10.48.131
NETWORK_INTERFACE       lan2
NETWORK_INTERFACE       lan3
FIRST_CLUSTER_LOCK_PV /dev/dsk/c20t0d7
# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE     /dev/tty0p0
```

```
# Possible standby Network Interfaces for lan1: lan2,lan3.
```

```
# Cluster Timing Parameters (microseconds).
```

```
# The NODE_TIMEOUT parameter defaults to 2000000 (2 seconds).
# This default setting yields the fastest cluster reformations.
# However, the use of the default value increases the potential
# for spurious reformations due to momentary system hangs or
# network load spikes.
# For a significant portion of installations, a setting of
# 5000000 to 8000000 (5 to 8 seconds) is more appropriate.
# The maximum value recommended for NODE_TIMEOUT is 30000000
# (30 seconds).
```

```
HEARTBEAT_INTERVAL          1000000
NODE_TIMEOUT              2000000
```

```
# Configuration/Reconfiguration Timing Parameters (microseconds).
```

```
AUTO_START_TIMEOUT         600000000
NETWORK_POLLING_INTERVAL   2000000
```

```
# Network Monitor Configuration Parameters.
# The NETWORK_FAILURE_DETECTION parameter determines how LAN card failures are
# detected.
# If set to INONLY_OR_INOUT, a LAN card will be considered down when its inbound
# message count stops increasing or when both inbound and outbound
# message counts stop increasing.
# If set to INOUT, both the inbound and outbound message counts must
# stop increasing before the card is considered down.
```

```
NETWORK_FAILURE_DETECTION    INOUT
```

```
# Package Configuration Parameters.
# Enter the maximum number of packages which will be configured in the cluster.
# You can not add packages beyond this limit.
# This parameter is required.
```

```
MAX_CONFIGURED_PACKAGES      150
```

```
# Access Control Policy Parameters.
```

```
#
# Three entries set the access control policy for the cluster:
# First line must be USER_NAME, second USER_HOST, and third USER_ROLE.
# Enter a value after each.
#
# 1. USER_NAME can either be ANY_USER, or a maximum of
#    8 login names from the /etc/passwd file on user host.
# 2. USER_HOST is where the user can issue Serviceguard commands.
#    If using Serviceguard Manager, it is the COM server.
#    Choose one of these three values: ANY_SERVICEGUARD_NODE, or
#    (any) CLUSTER_MEMBER_NODE, or a specific node. For node,
#    use the official hostname from domain name server, and not
#    an IP addresses or fully qualified name.
# 3. USER_ROLE must be one of these three values:
#    * MONITOR: read-only capabilities for the cluster and packages
#    * PACKAGE_ADMIN: MONITOR, plus administrative commands for packages
#      in the cluster
#    * FULL_ADMIN: MONITOR and PACKAGE_ADMIN plus the administrative
#      commands for the cluster.
#
# Access control policy does not set a role for configuration
# capability. To configure, a user must log on to one of the
# cluster's nodes as root (UID=0). Access control
# policy cannot limit root users' access.
#
# MONITOR and FULL_ADMIN can only be set in the cluster configuration file,
```

Cluster Concepts and Configuration

```
# and they apply to the entire cluster. PACKAGE_ADMIN can be set in the
# cluster or a package configuration file. If set in the cluster
# configuration file, PACKAGE_ADMIN applies to all configured packages.
# If set in a package configuration file, PACKAGE_ADMIN applies to that
# package only.
#
# Conflicting or redundant policies will cause an error while applying
# the configuration, and stop the process. The maximum number of access
# policies that can be configured in the cluster is 200.
#
# Example: to configure a role for user john from node noir to
# administer a cluster and all its packages, enter:
# USER_NAME   john
# USER_HOST   noir
# USER_ROLE    FULL_ADMIN
```

```
# List of cluster aware LVM Volume Groups. These volume groups will
# be used by package applications via the vgchange -a e command.
# Neither CVM or VxVM Disk Groups should be used here.
# For example:
# VOLUME_GROUP      /dev/vgdatabase
# VOLUME_GROUP      /dev/vg02
VOLUME_GROUP      /dev/vg01
```

5-14. TEXT PAGE: Sample Cluster Configuration ASCII File Using Quorum Server Technology for Cluster Lock Services

The following is the cluster-wide ASCII configuration file
using Serviceguard version A.11.17, with a Quorum Server
(i.e. an external system) for cluster lock functionality

```
# *****  
# ***** HIGH AVAILABILITY CLUSTER CONFIGURATION FILE *****  
# ***** For complete details about cluster parameters and how to *****  
# ***** set them, consult the Serviceguard manual. *****  
# *****
```

```
# Enter a name for this cluster. This name will be used to identify the  
# cluster when viewing or manipulating it.
```

```
CLUSTER_NAME          cluster1
```

```
# Cluster Lock Parameters  
# The cluster lock is used as a tie-breaker for situations  
# in which a running cluster fails, and then two equal-sized  
# sub-clusters are both trying to form a new cluster. The  
# cluster lock may be configured using only one of the  
# following alternatives on a cluster:  
#     the LVM lock disk  
#     the quorum server  
#  
#  
# Consider the following when configuring a cluster.  
# For a two-node cluster, you must use a cluster lock. For  
# a cluster of three or four nodes, a cluster lock is strongly  
# recommended. For a cluster of more than four nodes, a  
# cluster lock is recommended. If you decide to configure  
# a lock for a cluster of more than four nodes, it must be  
# a quorum server.  
  
# Lock Disk Parameters. Use the FIRST_CLUSTER_LOCK_VG and  
# FIRST_CLUSTER_LOCK_PV parameters to define a lock disk.  
# The FIRST_CLUSTER_LOCK_VG is the LVM volume group that  
# holds the cluster lock. This volume group should not be  
# used by any other cluster as a cluster lock device.  
  
# Quorum Server Parameters. Use the QS_HOST, QS_POLLING_INTERVAL,  
# and QS_TIMEOUT_EXTENSION parameters to define a quorum server.  
# The QS_HOST is the host name or IP address of the system  
# that is running the quorum server process. The  
# QS_POLLING_INTERVAL (microseconds) is the interval at which  
# Serviceguard checks to make sure the quorum server is running.  
# The optional QS_TIMEOUT_EXTENSION (microseconds) is used to increase  
# the time interval after which the quorum server is marked DOWN.  
#  
# The default quorum server timeout is calculated from the  
# Serviceguard cluster parameters, including NODE_TIMEOUT and  
# HEARTBEAT_INTERVAL. If you are experiencing quorum server  
# timeouts, you can adjust these parameters, or you can include
```

Cluster Concepts and Configuration

```
# the QS_TIMEOUT_EXTENSION parameter.
#
# The value of QS_TIMEOUT_EXTENSION will directly effect the amount
# of time it takes for cluster reformation in the event of failure.
# For example, if QS_TIMEOUT_EXTENSION is set to 10 seconds, the cluster
# reformation will take 10 seconds longer than if the QS_TIMEOUT_EXTENSION
# was set to 0. This delay applies even if there is no delay in
# contacting the Quorum Server. The recommended value for
# QS_TIMEOUT_EXTENSION is 0, which is used as the default
# and the maximum supported value is 30000000 (5 minutes).
#
# For example, to configure a quorum server running on node
# "qshost" with 120 seconds for the QS_POLLING_INTERVAL and to
# add 2 seconds to the system assigned value for the quorum server
# timeout, enter:
#
# QS_HOST qshost
# QS_POLLING_INTERVAL 120000000
# QS_TIMEOUT_EXTENSION 2000000
```

```
QS_HOST 10.51.206.148
QS_POLLING_INTERVAL 300000000
```

```
# Definition of nodes in the cluster.
# Repeat node definitions as necessary for additional nodes.
# NODE_NAME is the specified nodename in the cluster.
# It must match the hostname and both cannot contain full domain name.
# Each NETWORK_INTERFACE, if configured with IPv4 address,
# must have ONLY one IPv4 address entry with it which could
# be either HEARTBEAT_IP or STATIONARY_IP.
# Each NETWORK_INTERFACE, if configured with IPv6 address(es)
# can have multiple IPv6 address entries(up to a maximum of 2,
# only one IPv6 address entry belonging to site-local scope
# and only one belonging to global scope) which must be all
# STATIONARY_IP. They cannot be HEARTBEAT_IP.
```

```
NODE_NAME rx26u230
NETWORK_INTERFACE lan1
HEARTBEAT_IP 10.10.48.130
NETWORK_INTERFACE lan2
NETWORK_INTERFACE lan3
# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE /dev/tty0p0
```

```
# Possible standby Network Interfaces for lan1: lan2,lan3.
```

```
NODE_NAME rx26u231
NETWORK_INTERFACE lan1
HEARTBEAT_IP 10.10.48.131
NETWORK_INTERFACE lan2
NETWORK_INTERFACE lan3
# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE /dev/tty0p0
```

```
# Possible standby Network Interfaces for lan1: lan2,lan3.
```

```
# Cluster Timing Parameters (microseconds).
```

```
# The NODE_TIMEOUT parameter defaults to 2000000 (2 seconds).
```

```
# This default setting yields the fastest cluster reformations.
# However, the use of the default value increases the potential
# for spurious reformations due to momentary system hangs or
# network load spikes.
# For a significant portion of installations, a setting of
# 5000000 to 8000000 (5 to 8 seconds) is more appropriate.
# The maximum value recommended for NODE_TIMEOUT is 30000000
# (30 seconds).
```

```
HEARTBEAT_INTERVAL          1000000
NODE_TIMEOUT              2000000
```

```
# Configuration/Reconfiguration Timing Parameters (microseconds).
```

```
AUTO_START_TIMEOUT         600000000
NETWORK_POLLING_INTERVAL   2000000
```

```
# Network Monitor Configuration Parameters.
# The NETWORK_FAILURE_DETECTION parameter determines how LAN card failures are
# detected.
# If set to INONLY_OR_INOUT, a LAN card will be considered down when its inbound
# message count stops increasing or when both inbound and outbound
# message counts stop increasing.
# If set to INOUT, both the inbound and outbound message counts must
# stop increasing before the card is considered down.
```

```
NETWORK_FAILURE_DETECTION    INOUT
```

```
# Package Configuration Parameters.
# Enter the maximum number of packages which will be configured in the cluster.
# You can not add packages beyond this limit.
# This parameter is required.
```

```
MAX_CONFIGURED_PACKAGES      150
```

```
# Access Control Policy Parameters.
```

```
#
# Three entries set the access control policy for the cluster:
# First line must be USER_NAME, second USER_HOST, and third USER_ROLE.
# Enter a value after each.
#
# 1. USER_NAME can either be ANY_USER, or a maximum of
#    8 login names from the /etc/passwd file on user host.
# 2. USER_HOST is where the user can issue Serviceguard commands.
#    If using Serviceguard Manager, it is the COM server.
#    Choose one of these three values: ANY_SERVICEGUARD_NODE, or
#    (any) CLUSTER_MEMBER_NODE, or a specific node. For node,
#    use the official hostname from domain name server, and not
#    an IP addresses or fully qualified name.
# 3. USER_ROLE must be one of these three values:
#    * MONITOR: read-only capabilities for the cluster and packages
#    * PACKAGE_ADMIN: MONITOR, plus administrative commands for packages
#      in the cluster
#    * FULL_ADMIN: MONITOR and PACKAGE_ADMIN plus the administrative
#      commands for the cluster.
```

```
# Access control policy does not set a role for configuration
# capability. To configure, a user must log on to one of the
# cluster's nodes as root (UID=0). Access control
# policy cannot limit root users' access.
```

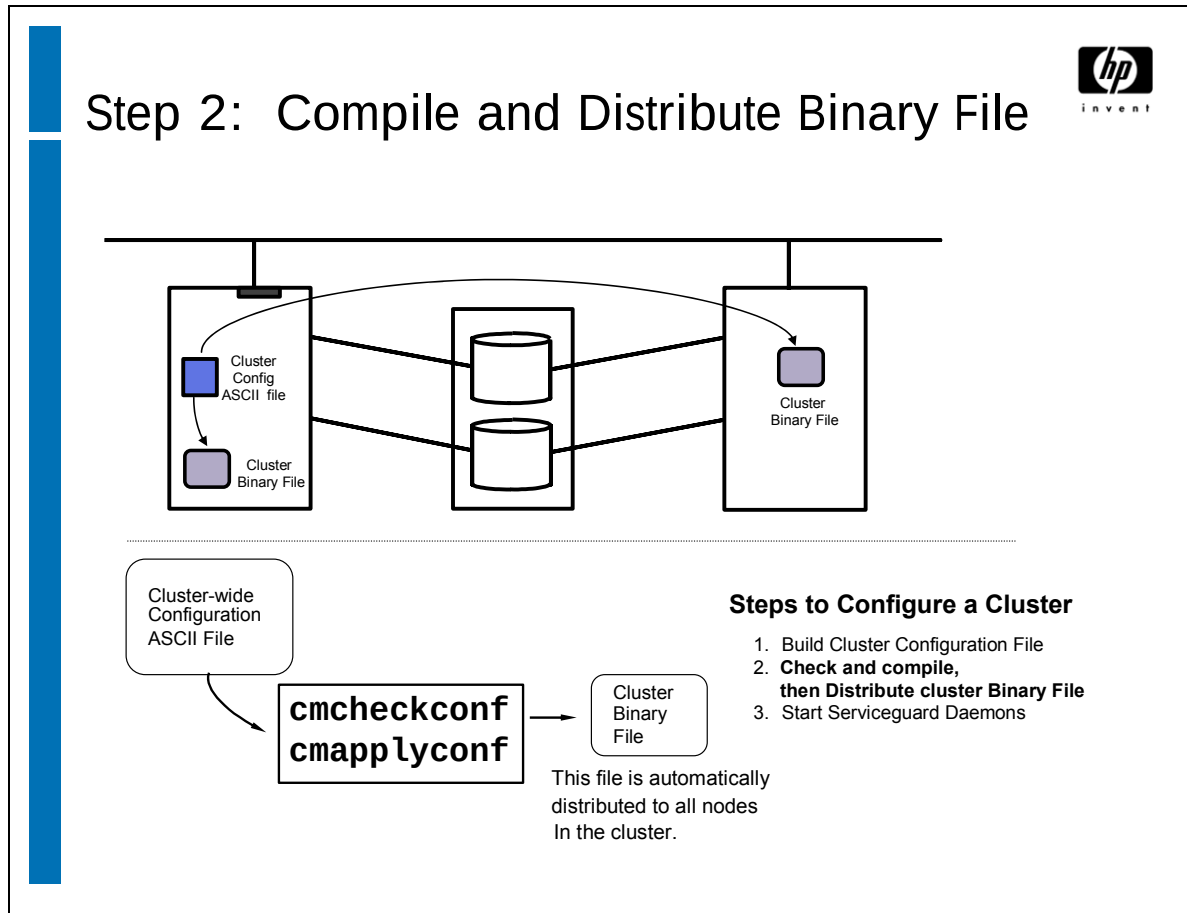
```
#
# MONITOR and FULL_ADMIN can only be set in the cluster configuration file,
# and they apply to the entire cluster. PACKAGE_ADMIN can be set in the
```

Cluster Concepts and Configuration

```
# cluster or a package configuration file. If set in the cluster
# configuration file, PACKAGE_ADMIN applies to all configured packages.
# If set in a package configuration file, PACKAGE_ADMIN applies to that
# package only.
#
# Conflicting or redundant policies will cause an error while applying
# the configuration, and stop the process. The maximum number of access
# policies that can be configured in the cluster is 200.
#
# Example: to configure a role for user john from node noir to
# administer a cluster and all its packages, enter:
# USER_NAME  john
# USER_HOST  noir
# USER_ROLE  FULL_ADMIN

# List of cluster aware LVM Volume Groups. These volume groups will
# be used by package applications via the vgchange -a e command.
# Neither CVM or VxVM Disk Groups should be used here.
# For example:
# VOLUME_GROUP      /dev/vgdatabase
# VOLUME_GROUP      /dev/vg02
```


5-15. SLIDE: Step 2: Compile and Distribute Binary File



Student Notes

Once the cluster configuration ASCII file is edited, the file should be checked for errors. The `cmcheckconf` command can be used to perform these checks:

```
# cmcheckconf -v -C cmclconfig.ascii
```

After the file has been verified as containing no errors, the `cmapplyconf` command is used to create and distribute the cluster binary file.

```
# cmapplyconf -v -C cmclconfig.ascii
```

See the following two pages for the actual screen output resulting from the `cmcheckconf` and `cmapplyconf` commands.

5-16. TEXT PAGE: The cmcheckconf Command

Special Note

Here is the actual screen output resulting from the cmcheckconf command.
This output was obtained while using **version A.11.17** of Serviceguard.

```
root@rx26u230    [/etc/cmcluster]
# cmcheckconf -v -C cmclconfig.ascii
Checking cluster file: cmclconfig.ascii
Checking nodes ... Done
Checking existing configuration ... Done
Gathering storage information
Found 2 devices on node rx26u230
Found 2 devices on node rx26u231
Analysis of 4 devices should take approximately 1 seconds
0%----10%----20%----30%----40%----50%----60%----70%----80%----90%----100%
Found 2 volume groups on node rx26u230
Found 2 volume groups on node rx26u231
Analysis of 4 volume groups should take approximately 1 seconds
0%----10%----20%----30%----40%----50%----60%----70%----80%----90%----100%
Gathering network information
Beginning network probing (this may take a while)
Completed network probing
Checking for inconsistencies
Adding node rx26u230 to cluster cluster9
Adding node rx26u231 to cluster cluster9
cmcheckconf: Verification completed with no errors found.
Use the cmapplyconf command to apply the configuration.

root@rx26u230    [/etc/cmcluster]
#
```

5-17. TEXT PAGE: The cmapplyconf Command

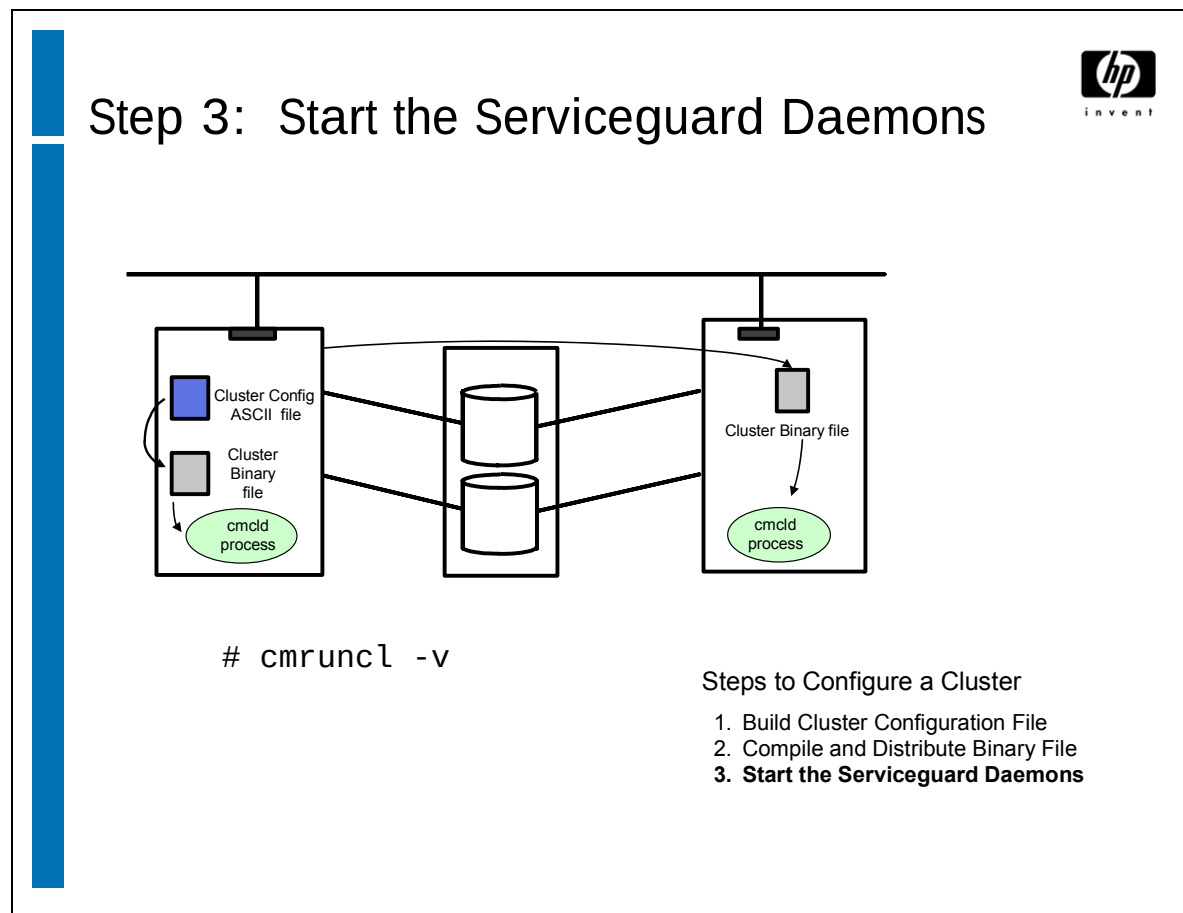
Special Note

Here is the actual screen output resulting from the cmapplyconf command.
This output was obtained while using version A.11.17 of Serviceguard.

```
root@rx26u230  [/etc/cmcluster]
# cmapplyconf -v -C cmclconfig.ascii
Checking cluster file: cmclconfig.ascii
Checking nodes ... Done
Checking existing configuration ... Done
Gathering storage information
Found 2 devices on node rx26u230
Found 2 devices on node rx26u231
Analysis of 4 devices should take approximately 1 seconds
0%-----10%-----20%-----30%-----40%-----50%-----60%-----70%-----80%-----90%-----100%
Found 2 volume groups on node rx26u230
Found 2 volume groups on node rx26u231
Analysis of 4 volume groups should take approximately 1 seconds
0%-----10%-----20%-----30%-----40%-----50%-----60%-----70%-----80%-----90%-----100%
Gathering network information
Beginning network probing (this may take a while)
Completed network probing
Checking for inconsistencies
Adding node rx26u230 to cluster cluster9
Adding node rx26u231 to cluster cluster9
Marking/unmarking volume groups for use in the cluster
Completed the cluster creation

root@rx26u230  [/etc/cmcluster]
#
```

5-18. SLIDE: Step 3: Start the Serviceguard Daemons



Student Notes

Once the cluster binary file has been distributed to all nodes in the cluster, the `cmruncl` command can be used to bring up the cluster.

```
# cmruncl -v
```

This command launches the appropriate Serviceguard daemons, including the `cmcl` daemon that monitors the cluster resources and sends or receives heartbeat packets.

5-19. TEXT PAGE: The cmruncl Command

Special Note

Here is the actual screen output resulting from the `cmruncl` command.
This output was obtained while using version A.11.16 of Serviceguard.

```
root@rx26u230    [/etc/cmcluster]
# cmruncl -v
cmruncl: Validating network configuration...
cmruncl: Network validation complete
Waiting for cluster to form ..... done
Cluster successfully formed.
Check the syslog files on all nodes in the cluster to verify that no
warnings occurred during startup.

root@rx26u230    [/etc/cmcluster]
#
```

5-20. SLIDE: Cluster Configuration Procedure



Cluster Configuration Procedure

1. Set up trusted hosts within cluster systems:

```
# vi /etc/cmcluster/cmclnodelist
```

this_node_hostname	root
other_node_hostname	root
2.

```
# cd /etc/cmcluster
```
3.

```
# cmquerycl -v [-q QS_HOST] -C cmclconf.ascii \
```

```
-n node1 -n node2
```
4.

```
# vi cmclconf.ascii
```

Modify CLUSTER_NAME field
Modify timing parameters
5.

```
# cmcheckconf -v -C /etc/cmcluster/cmclconf.ascii
```
6.

```
# cmapplyconf -v -C /etc/cmcluster/cmclconf.ascii
```
7.

```
# cmruncl -v
```
8.

```
# cmviewcl -v
```

Student Notes

The above slide summarizes the setup procedure for configuring a cluster. The details of each step are provided below:

(Step 1) Set Up Trusted Hosts

Prior to Serviceguard version A.11.16, the `/etc/cmcluster/cmclnodelist` file was used to configure all inter-host access for Serviceguard commands.

As of A.11.16, this file is only used to configure the cluster. Once the cluster is configured, an internal access list is used. Details will be discussed in a later module.

(Steps 2 – 4) Create the Cluster Configuration ASCII File

The cluster configuration ASCII file should be created in the `/etc/cmcluster` directory. The `cmquerycl` command can be used to gather the node-specific information and to create a template that can be easily customized for the cluster.

At the very least, the `CLUSTER_NAME` variable should be modified in the template. Other values should be modified as appropriate.

(Steps 5 and 6) Compile and Distribute the Binary File

Once the cluster configuration ASCII file is edited, the file should be checked for syntax and logic errors. The `cmcheckconf` command performs the following checks:

- Network addresses and connections.
- Cluster lock disk connectivity (if using a cluster lock disk).
- Validity of configuration parameters for the cluster and packages.
- Uniqueness of names.
- Existence and permission of scripts specified in the command line.
- If all the nodes specified are in the same heartbeat subnet.
- If you specify the wrong configuration filename.
- If all nodes are accessible.
- No more than one `CLUSTER_NAME`, `HEARTBEAT_INTERVAL`, `AUTO_START_TIMEOUT`, and `FIRST_CLUSTER_LOCK_VG` are specified.
- The value for `HEARTBEAT_INTERVAL` $\geq$ 1 second; the value of `NODE_TIMEOUT` is at least twice the value of `HEARTBEAT_INTERVAL`.
- The value of `AUTO_START_TIMEOUT` are integers $\geq$ 0.
- At least 1 and no more than 16 `NODE_NAME`'s (Serviceguard OPS edition or Serviceguard) are specified.
- Each Node is connected to each heartbeat network.
- All heartbeat networks are of the same type of LAN.
- The network interface device files specified are valid LAN devices.
- There is no more than one serial (RS232) port connection per node.
- `VOLUME_GROUP` entries are not currently marked as cluster-aware.

After the file has been verified as containing no errors, the `cmapplyconf` command is used to create and distribute the cluster binary file.

(Step 7) Bring Up the Cluster

The `cmruncl` command is used to bring up the cluster.

(Step 8) Check the status of the cluster

```
# cmviewcl -v
```

5-21. SLIDE: Viewing the Cluster – `cmviewcl` Command

Viewing the Cluster — `cmviewcl` Command



```
# cmviewcl -v

CLUSTER    STATUS
cluster9   up

  NODE    STATUS  STATE
  rx26u230 up      running

Cluster_Lock_LVM:
VOLUME_GROUP  PHYSICAL_VOLUME  STATUS
/dev/vg01      /dev/dsk/c20t0d7  up

Network_Parameters:
INTERFACE  STATUS  PATH      NAME
PRIMARY    up      0/1/2/0    lan1
STANDBY    up      0/2/1/0    lan2

  NODE    STATUS  STATE
  rx26u231 up      running

Cluster_Lock_LVM:
VOLUME_GROUP  PHYSICAL_VOLUME  STATUS
/dev/vg01      /dev/dsk/c20t0d7  up

Network_Parameters:
INTERFACE  STATUS  PATH      NAME
PRIMARY    up      0/1/2/0    lan1
STANDBY    up      0/2/1/0    lan2
```

Student Notes

The `cmviewcl` command with the `-v` (verbose) option displays information about the status of the cluster and nodes within the cluster.

Cluster States

The *status* of a cluster may be one of the following:

Up At least one node has a running cluster daemon (`cmcl`).

Down No cluster daemons are running on any nodes.

Reforming The cluster is in the process of determining its active membership.

Node States

The *status* of a node is either up or down, depending on whether its cluster daemon is running or not. A node may also be in one of the following states:

Initializing A node sees itself in this state after its daemon has started, but before it is ready to communicate with other nodes' daemons.

Failed	A node never sees itself in this state. Other active members of the cluster will see a node in this state.
Cluster reforming	A node in this state is running the protocols which ensure that all nodes agree to the new membership and the new coordinator of the cluster.
Running	A node in this state has completed all required activity for the last reformation and is operating normally.
Halted	A node never sees itself in this state. Other nodes will see it in this state after the node has gracefully left the cluster.

New in version A.11.17 is an option to `cmviewcl` to put the output in a format that is more easily parsed by a shell script.

```
# cmviewcl -f line
```

```
name=cluster9
id=1416255277
status=up
state=stable
coordinator=rx26u230
node:rx26u230|name=rx26u230
node:rx26u230|status=up
node:rx26u230|state=running
node:rx26u230|os_status=up
node:rx26u230|id=1
node:rx26u231|name=rx26u231
node:rx26u231|status=up
node:rx26u231|state=running
node:rx26u231|os_status=unknown
node:rx26u231|id=2
```

5-22. SLIDE: Checking the Cluster Log



Checking the Cluster Log

```
# tail -f /var/adm/syslog/syslog.log

May 2 15:30:45 systemA cmcld[2048]:      cmruncl
May 2 15:30:47 systemA cmcld[2048]:      Starting cluster protocols
May 2 15:30:48 systemA cmcld[2048]:      Attempting to form a new cluster
May 2 15:30:50 systemA cmcld[2048]:      Clearing the Cluster Lock
May 2 15:30:45 systemA cmcld[2048]:      Turning on safety time protection
May 2 15:30:50 systemA cmcld[2048]:      Two nodes have formed a new cluster
May 2 15:30:51 systemA cmcld[2048]:      The new active membership is:
                                          systemA, systemB
May 2 15:30:52 systemA cmcld[2048]:      clvmd initialized successfully
```

Student Notes

The slide shows entries from the `/var/adm/syslog/syslog.log` file when the `cmruncl` command was being executed.

In general, the `syslog.log` file is an excellent source of information regarding the activities that have taken place on the cluster as a whole. The `cmcld` daemon logs cluster messages to the HP-UX message daemon `syslogd`, using the daemon facility. Check `/etc/syslog.conf` for `syslogd` configuration on your systems.

5-23. LAB: Building a Cluster

Special Note:

When building a cluster in this class, you must choose whether you wish to use a Quorum Server –or- an LVM Lock Disk. Please advise your instructor of your preference when asked.

Quorum Server for Cluster Lock Services:

If you wish to use a Quorum Server to provide cluster lock services, **notify your instructor immediately** and then turn the page and begin LAB 5-24.

LVM Lock Disk:

If you wish to use an LVM Lock Disk instead, turn to LAB 5-25 and begin configuring your cluster. Volume group vg01, previously configured in Module 4, is the volume group that will provide LVM Cluster Lock disk functionality.

5-24. LAB: Cluster Lock Using A Quorum Server

Directions

Before you build your cluster, install the Quorum Server software on a system designated by the instructor. Then configure, startup, and test your quorum server.
Then configure, startup, and test your cluster.

Part 1: Identify the Depot location from which to swinstall the Quorum Server software

Special Note

In this lab, we will configure one quorum server for the entire class.
Multiple Serviceguard clusters can all use the same Quorum Server.

1. Your instructor may ask for a volunteer to configure the Quorum Server.
Ask your instructor for the name of the person who will complete all of Part 2 of this lab, "Configure the Quorum Server", and write that person's name here:

Name of person completing Part 2

2. Ask your instructor for the depot location of the Quorum Server software. If the software is located on a different server, note that as well. Write the depot location here:

Server and Depot Location

Part 2: Configure the Quorum Server

1. Ask your instructor which system will be used as the Quorum Server. Write its hostname here:

Quorum Server

The student whom the instructor designated to install the quorum server software should login to the quorum server now to perform the next several steps.

2. On the designated Quorum Server system, the student will install the Quorum Server software, product number B8467BA. Install this software using `depotserver` and the software depot (*noted above in Part I, item 2*). We install this software only on the system that will become our Quorum Server. The Quorum Server software is not installed on individual systems which later will be part of the cluster.

```
# swinstall -s depotserver:/pathname \*
```

3. Now, with the software installed on the Quorum Server system, we need to create the authorization file to allow specific host systems to obtain quorum services. The *required* pathname for this authorization file is `/etc/cmcluster/qs_authfile`.

In this file, enter the hostnames of all cluster nodes that will access quorum services from this Quorum Server. Use one line per node, as in the following example:

```
r213c2  
r213c3  
r213c4  
r213c5
```

4. By default, quorum service run-time messages go to `stdout` and `stderr`. Instead, let's redirect these messages to a logfile. For this purpose, we will create the new subdirectory `/var/adm/qs`.

5. Now, edit the `/etc/inittab` file so that the Quorum Server will be started any time the system restarts or reboots. We also redirect `stdout` and `stderr` to a file in the `/var/adm/qs` subdirectory.

```
# vi /etc/inittab
```

and, at the bottom of the file, add a line that reads:

```
qs:345:respawn:/usr/sbin/qs >> /var/adm/qs/qs.log 2>&1
```

6. Next, start the Quorum Server software by telling `init` to re-read its configuration file.

```
# init q
```

7. Verify that the Quorum Server is running properly by checking the `/var/adm/qs/qs.log` file.

```
# cat /var/adm/qs/qs.log
```

The log file should contain entries that look something like the following, which indicates that the Quorum Server has properly started:

```
Nov 12 15:02:51:0: Starting Quorum Server
Nov 12 15:02:51:0: Total allocated: 466492 bytes, used: 20896 bytes, unused 429224 bytes
Nov 12 15:02:54:0: Server is up and waiting for connections at port 1238
```

Part 3: Configure, Startup, and Test your Serviceguard Cluster

1. First, list the nodes that will comprise your Serviceguard cluster. Write their hostnames here:

Node1

Node2

2. Now, choose one node to be the "primary node" for your cluster. This is the node from which almost all cluster administration will be done. Write the hostname for your primary node here:

Primary Node

Special Note

For the remainder of the week, your primary node will be the node that is generally used for all cluster administration, unless specifically directed otherwise.

3. Next, on both nodes, and before we start our actual configuration, edit the /etc/lvmrc file. We do this so that volume groups used within the cluster will be activated by Serviceguard. We do not want these volume groups activated by default whenever the node reboots. To accomplish this, edit the file and set the AUTO_VG_ACTIVATE line equal to 0 (zero).

AUTO_VG_ACTIVATE=0

4. Now, on your primary node, set up the special file used by Serviceguard that enables you to initially configure your cluster. We do this on the primary node by creating a new file called `/etc/cmcluster/cmclnodelist`. **Important:** All nodes in the cluster must be listed in this file. **The file is identical on all nodes**, and therefore we can easily `ftp` or `rcp` the file to every node in the cluster. The format for the file looks something like the following:

<Node1>	root
<Node2>	root
<Node3>	root
<Node4>	root

5. Now, we create the cluster-wide ASCII configuration file. This file is where all cluster-wide information will be stored.

Note that we must specify: (a) the name of our Quorum Server, (b) the name of the cluster-wide ASCII file, and (c) the names of each node in our (*soon-to-be*) cluster.

6. Examine the cluster-wide ASCII configuration file built by the `cmquerycl` command *in the previous step*. Identify the following information:

```
# cd /etc/cmcluster
# more cmclconfig.ascii
```

To what value is the name of the cluster ("`CLUSTER_NAME`") set? _____

What is the hostname of the Quorum Server ("`QS_HOST`")? _____

To what value is the Quorum Server polling interval ("`QS_POLLING_INTERVAL`") set? _____

To what value is the HEARBEAT_INTERVAL set? _____

To what value is the NODE_TIMEOUT set? _____

To what value is the maximum number of packages ("`MAX_CONFIGURED_PACKAGES`") set? _____

LAN Interfaces	Node1	Node2
Which lan interface card will be used as HEATBEAT_IP?		
Which lan interface card, if any, will be used as STATIONARY_IP?		
Which lan interface card, if any, will be used as a standby card (if you have an extra card)?		
Which lan interface card, if any, will be used as another standby card (if you have a second extra card)?		

7. Edit the cluster-wide ASCII configuration file to change some of these values (*seen above*):
 - a. Change your cluster name to whatever your instructor asks. Ask your instructor for this new name.
 - b. Set the QS_POLLING_INTERVAL to 120000000 microseconds (*120 seconds*).
 - c. Set the HEARTBEAT_INTERVAL to 3000000 microseconds (*3 seconds*).
 - d. Set the NODE_TIMEOUT value to 6000000 microseconds (*6 seconds*).
 - e. Set the AUTO_START_TIMEOUT value to 800000000 microseconds (*800 seconds*).
8. Let's add a non-root user to the list of users who can execute certain Serviceguard commands. In the cluster ascii file, give the oracle user Access Control rights that will enable him/her to monitor packages within the cluster. Allow this user to do this from either node in your cluster.
9. Now, on your primary node, use the cmcheckconf command to check the cluster-wide ASCII configuration file for errors. Fix any errors and re-check if necessary:
10. In order to see what files are created by the cmapplyconf command (*see step 11 below*), view the contents of the /etc/cmcluster directory as it is now, before we do step 10.
11. Next, use the cmapplyconf command to apply the configuration. The cmapplyconf command will always (1) compile the ASCII file to binary form, and (2) distribute this binary file to all nodes in the cluster.

12. Once again, view the contents of the `/etc/cmcluster` directory.

```
# ll /etc/cmcluster
```

Write the names of the file(s) created by the `cmapplyconf` command?

Files in the <code>/etc/cmcluster</code> directory		

13. Now, start cluster services on all nodes (*that is, start the cluster*). While it is a good practice to administer your cluster from one specific node, it is possible to start your cluster from any node (in the cluster).

14. Using the log file for `syslogd` (`/var/adm/syslog/syslog.log`), verify that the cluster has properly formed on all nodes in the cluster.

15. Now, view the complete status of the cluster. In the screen output, how many references do you see to "Quorum Server"?

Number of references seen to "Quorum Server" → _____

Part 4: Test your Serviceguard Cluster

1. For our first test, from your primary node, halt the cluster.
2. Now, with the cluster halted, shutdown and power off any one node except your primary node. In Part 4 of this lab, we will call this node (*the node to be shutdown*) "Node2".

Note that, if you are using remote equipment, the shutdown `-hy 0` command will automatically cause a `<power -off>`.

3. Now, on your primary node, attempt to restart the cluster. Note that we are starting the cluster from the primary node only, but the `cmrunc1` command will apply to all the nodes in the cluster.

As the cluster is trying to start, watch your screen carefully, and answer the following questions:

- a. Does the cluster form properly? Why or why not?
- b. How can we restart the cluster if one node has become inoperable? See the manpage for `cmrunc1` and note the `-n` option. Note that it is possible to start the cluster with less than 100% node attendance.

```
# man cmrunc1
```

Start the cluster with this option.

4. Now, turn the power back on for Node2, and boot it up to run-level 3. If you are using remote equipment, issue a `pc` command from the Management Processor Command Menu.

Note: In order to interact with the system at the MP prompt, you will need to login to the MP on the system you shut down. If you do not know the MP hostname, ask your instructor.

To login to the MP, you'll need to supply an MP username and password. Once logged in, and at the MP prompt, type `CM` to enter the Command Menu. This is where you'll find the `pc` command, which is used to control power to your system.

After powering up the system, and while still at the `MP:CM>` prompt, type `MA` to return to the MP prompt. From here, type `C0` to access the system console and to login to unix once the system is back up.

Answer:

`<power -on>` (Using local equipment)

or

`MP:CM> pc` (Using remote equipment)

5. After Node2 is fully booted, check to see if it has automatically rejoined the cluster.
6. Examine the `/etc/rc.config.d` directory. Can you determine the name of the file in `/etc/rc.config.d` that controls whether or not a node automatically rejoins the cluster? **Hint:** you may be able to identify the file visually by listing the directory or, since all files in this directory are text files, a `grep` command might be handy.

```
# cd /etc/rc.config.d
# ll
```

The name of this file is _____.

Cluster Concepts and Configuration

7. Since Node2 has not automatically rejoined the cluster (*step 5 above*), use the `cmrunnode` command to bring Node2 back into the running cluster. Afterward, check again to ensure that Node2 has rejoined the cluster and the cluster is now behaving correctly.

Congratulations on successfully starting and testing your Serviceguard cluster using a Quorum Server!

5-25. LAB: Cluster Lock Using LVM Disks

Directions

In a previous module, we configured a shared volume group. Cluster Lock Disk functionality will be obtained from a disk in this volume group automatically. As such, if you elect to use an LVM Lock Disk in an I.T. environment, it is important to build the shared volume group first before building the cluster. Let's proceed with building our cluster.

Part 1: Configuring your Serviceguard Cluster

1. First, list the nodes that will comprise your Serviceguard cluster. Write their hostnames here:

Node1

Node2

2. Now, choose one node to be the "primary node" for your cluster. This is the node from which almost all cluster administration will be done. Write the hostname for your primary node here:

Primary Node

Special Note

For the remainder of the week, your primary node will be the node that is generally used for all cluster administration, unless specifically directed otherwise.

3. Next, on both nodes, and before we start our actual configuration, edit the /etc/lvmrc file. We do this so that volume groups used within the cluster will be activated by Serviceguard. We do not want these volume groups activated by default whenever the node reboots. To accomplish this, edit the file and set the AUTO_VG_ACTIVATE line equal to 0 (zero).

AUTO_VG_ACTIVATE=0

4. Now, on your primary node, set up the special file used by Serviceguard that enables you to initially configure your cluster. We do this on the primary node by creating a new file named `/etc/cmcluster/cmclnodelist`. **Important:** All nodes in the cluster must be listed in this file. The file is identical on every node in our cluster, and therefore we can easily `ftp` or `rcp` the file to every node in the cluster. The format for the file looks something like the following:

Node1	root
Node2	root
Node3	root
Node4	root

5. Now, we create the cluster-wide ASCII configuration file. This file is where all cluster-wide information will be stored.

Note that we must specify the name of the cluster-wide ASCII file and the names of each node in our (*soon-to-be*) cluster.

6. Examine the cluster-wide ASCII configuration file built by the `cmquerycl` command (*in the previous step*). Identify the following information:

To what value is the name of the cluster ("CLUSTER_NAME") set?

To what value is the HEARTBEAT_INTERVAL set?

To what value is the NODE_TIMEOUT set?

To what value is maximum number of packages ("MAX_CONFIGURED_PACKAGES") set?

LAN Interfaces	Node1	Node2
Which lan interface card will be used as HEARTBEAT_IP?		
Which lan interface card, if any, will be used as STATIONARY_IP?		
Which lan interface card, if any, will be used as a standby card (if you have an extra card)?		
Which lan interface card, if any, will be used as another standby card (if you have a second extra card)?		

7. Edit the cluster-wide ASCII configuration file to change some of these values (*seen above*):
 - a. Change your cluster name to whatever your instructor asks. Ask your instructor for this new name and write it here _____.
 - b. Set the HEARTBEAT_INTERVAL to 3000000 microseconds (*3 seconds*).
 - c. Set the NODE_TIMEOUT value to 6000000 microseconds (*6 seconds*).
 - d. Set the AUTO_START_TIMEOUT value to 800000000 microseconds (*800 seconds*).
8. Let's add a non-root user to the list of users who can execute certain Serviceguard commands. In the cluster ascii file, give the oracle user Access Control rights that will enable him/her to monitor packages within the cluster. Allow this user to do this from either node in your cluster.
9. Now, on your primary node, use the `cmcheckconf` command to check the cluster-wide ASCII configuration file for errors. Fix any errors and re-check if necessary:
10. To see what files are created by the `cmapplyconf` command (*see step 11 below*), view the contents of the `/etc/cmcluster` directory as it is now, before we do step 10.

11. Next, still on your primary node, use the `cmapplyconf` command. The `cmapplyconf` command will always: (a) compile the `cmclconfig.ascii` file to binary, and (b) distribute the binary file to all nodes.
12. Once again, view the contents of the `/etc/cmcluster` directory.
Write the names of the file(s) created by the `cmapplyconf` command?

Files in the <code>/etc/cmcluster</code> directory		

13. Now, start cluster services on all nodes (*that is, start the cluster*). While it is a good practice to administer your cluster from one specific node, it is possible to start your cluster from any node (in the cluster).
14. Using the log file for `syslogd` (`/var/adm/syslog/syslog.log`), verify that the cluster has properly formed on all nodes in the cluster.
15. Using `cmviewcl -v`, verify that the cluster has properly formed on all nodes in the cluster.

Part 2: Test your Serviceguard Cluster

1. For our first test, from your primary node, halt the cluster.
2. Next, with the cluster halted, shutdown and power off any one node except your primary node. In this part of the lab, we will call this node "Node2" (*the node to be shutdown*).

Note that, if you are using remote equipment, the shutdown -hy 0 command will automatically cause a <power-off>.

3. Now, on your primary node, attempt to restart the cluster. Note that we are starting the cluster from the primary node only, but the `cmrunc1` command will apply to all the nodes in the cluster.

As the cluster is trying to start, watch your screen carefully, and answer the following questions:

- a. Does the cluster form properly? Why or why not?
- b. How can we restart the cluster if one node has become inoperable? See the manpage for `cmrunc1` and note the `-n` option. Note that it is possible to start the cluster with less than 100% node attendance.

man cmrunc1

Start the cluster with this option.

4. Now turn the power back on for Node2, then boot Node2 up to run-level 3. If you are using remote equipment, issue a `pc` command from the Management Processor Command menu.

Note: In order to interact with the system at the MP prompt, you will need to login to the MP on the system you shut down. If you do not know the MP hostname, ask your instructor.

To login to the MP, you'll need to supply an MP username and password. Once logged in, and at the MP prompt, type `CM` to enter the Command Menu. This is where you'll find the `pc` command, which is used to control power to your system.

After powering up the system, and while still at the `MP:CM>` prompt, type `MA` to return to the MP prompt. From here, type `C0` to access the system console and to login to unix once the system is back up.

5. After Node2 is fully booted, check to see if Node2 has automatically rejoined the cluster?
6. Now, examine the `/etc/rc.config.d` directory. Can you discover the name of the file in `/etc/rc.config.d` that controls whether or not a node automatically rejoins the cluster? **Hint:** you may be able to identify the file visually by listing the directory or, since all files in this directory are text files, a `grep` command might be handy.

The name of this file is _____.

Cluster Concepts and Configuration

7. Since Node2 has not automatically rejoined the cluster (*item 5 above*), use the `cmrunnode` command to bring Node2 back into the running cluster. Afterward, check again to ensure that Node2 has rejoined the cluster and the cluster is now behaving correctly.

Congratulations on successfully configuring, starting, and testing your Serviceguard cluster using an LVM Lock Disk.

5-26. LAB Solution: Cluster Lock Using A Quorum Server

Directions

Before you build your cluster, install the Quorum Server software on a system designated by the instructor. Then configure, startup, and test your quorum server.
Then configure, startup, and test your cluster.

Part 1: Identify the Depot location from which to swinstall the Quorum Server software

Special Note

In this lab, we will configure one quorum server for the entire class.
Multiple Serviceguard clusters can all use the same Quorum Server.

1. Your instructor may ask for a volunteer to configure the Quorum Server.
Ask your instructor for the name of the person who will complete all of Part 2 of this lab, "Configure the Quorum Server", and write that person's name here:

Name of person completing Part 2

2. Ask your instructor for the depot location of the Quorum Server software. If the software is located on a different server, note that as well. Write the depot location here:

Server and Depot Location

Part 2: Configure the Quorum Server

1. Ask your instructor which system will be used as the Quorum Server. Write its hostname here:

Quorum Server

The student whom the instructor designated to install the quorum server software should login to the quorum server now to perform the next several steps.

2. On the designated Quorum Server system, the student will install the Quorum Server software, product number B8467BA. Install this software using `depotserver` and the software depot (*noted above in Part I, item 2*). We install this software only on the system that will become our Quorum Server. The Quorum Server software is not installed on individual systems which later will be part of the cluster.

```
# swinstall -s depotserver:/pathname \*
```

3. Now, with the software installed on the Quorum Server system, we need to create the authorization file to allow specific host systems to obtain quorum services. The *required* pathname for this authorization file is `/etc/cmcluster/qs_authfile`.

In this file, enter the hostnames of all cluster nodes that will access quorum services from this Quorum Server. Use one line per node, as in the following example:

```
r213c2
r213c3
r213c4
r213c5
```

Answer:

```
# cd /etc/cmcluster
# vi qs_authfile
```


4. By default, quorum service run-time messages go to `stdout` and `stderr`. Instead, let's redirect these messages to a logfile. For this purpose, we will create the new subdirectory `/var/adm/qs`.

Answer:

```
# mkdir /var/adm/qs
```

5. Now, edit the `/etc/inittab` file so that the Quorum Server will be started any time the system restarts or reboots. We also redirect `stdout` and `stderr` to a file in the `/var/adm/qs` subdirectory.

```
# vi /etc/inittab
```

and, at the bottom of the file, add a line that reads:

```
qs:345:respawn:/usr/sbin/qs  >>  /var/adm/qs/qs.log  2>&1
```

6. Next, start the Quorum Server software by telling `init` to re-read its configuration file.

```
# init q
```

7. Verify that the Quorum Server is running properly by checking the `/var/adm/qs/qs.log` file.

```
# cat /var/adm/qs/qs.log
```

The log file should contain entries that look something like the following, which indicates that the Quorum Server has properly started:

```
Nov 12 15:02:51:0: Starting Quorum Server
Nov 12 15:02:51:0: Total allocated: 466492 bytes, used: 20896 bytes, unused 429224 bytes
Nov 12 15:02:54:0: Server is up and waiting for connections at port 1238
```

Part 3: Configure, Startup, and Test your Serviceguard Cluster

1. First, list the nodes that will comprise your Serviceguard cluster. Write their hostnames here:

Node1

Node2

2. Now, choose one node to be the "primary node" for your cluster. This is the node from which almost all cluster administration will be done. Write the hostname for your primary node here:

Primary Node

Special Note

For the remainder of the week, your primary node will be the node that is generally used for all cluster administration, unless specifically directed otherwise.

3. Next, on both nodes, and before we start our actual configuration, edit the /etc/lvmrc file. We do this so that volume groups used within the cluster will be activated by Serviceguard. We do not want these volume groups activated by default whenever the node reboots. To accomplish this, edit the file and set the AUTO_VG_ACTIVATE line equal to 0 (zero).

AUTO_VG_ACTIVATE=0

Answer:

```
# vi /etc/lvmrc
```

4. Now, on your primary node, set up the special file used by Serviceguard that enables you to initially configure your cluster. We do this on the primary node by creating a new file called `/etc/cmcluster/cmclnodelist`. **Important:** All nodes in the cluster must be listed in this file. **The file is identical on all nodes**, and therefore we can easily `ftp` or `rcp` the file to every node in the cluster. The format for the file looks something like the following:

<code><Node1></code>	<code>root</code>
<code><Node2></code>	<code>root</code>
<code><Node3></code>	<code>root</code>
<code><Node4></code>	<code>root</code>

Answer:

```
# cd /etc/cmcluster
# vi cmclnodelist
# chmod 444 cmclnodelist
# rcp cmclnodelist <Node2>:/etc/cmcluster/.    (to transfer the file to Node2
                                                and also to all other nodes)
```

5. Now, we create the cluster-wide ASCII configuration file. This file where all cluster-wide information will be stored.

Note that we must specify: (a) the name of our Quorum Server, (b) the name of the cluster-wide ASCII file, and (c) the names of each node in our (*soon-to-be*) cluster.

Answer:

```
# cd /etc/cmcluster
# cmquerycl -v -q quorum_server -C cmclconfig.ascii -n <Node1> \
                                                    -n <Node2>
```

Cluster Concepts and Configuration

6. Examine the cluster-wide ASCII configuration file built by the `cmquerycl` command *in the previous step*. Identify the following information:

```
# cd /etc/cmcluster
# more cmclconfig.ascii
```

To what value is the name of the cluster ("CLUSTER_NAME") set? _____

What is the hostname of the Quorum Server ("QS_HOST")? _____

To what value is the Quorum Server polling interval ("QS_POLLING_INTERVAL") set? _____

To what value is the HEARTBEAT_INTERVAL set? _____

To what value is the NODE_TIMEOUT set? _____

To what value is the maximum number of packages ("MAX_CONFIGURED_PACKAGES") set? _____

LAN Interfaces	Node1	Node2
Which lan interface card will be used as HEARTBEAT_IP?		
Which lan interface card, if any, will be used as STATIONARY_IP?		
Which lan interface card, if any, will be used as a standby card (if you have an extra card)?		
Which lan interface card, if any, will be used as another standby card (if you have a second extra card)?		

7. Edit the cluster-wide ASCII configuration file to change some of these values (*seen above*):
 - a. Change your cluster name to whatever your instructor asks. Ask your instructor for this new name.
 - b. Set the QS_POLLING_INTERVAL to 120000000 microseconds (*120 seconds*).
 - c. Set the HEARTBEAT_INTERVAL to 3000000 microseconds (*3 seconds*).
 - d. Set the NODE_TIMEOUT value to 6000000 microseconds (*6 seconds*).
 - e. Set the AUTO_START_TIMEOUT value to 800000000 microseconds (*800 seconds*).

Answer:

```
# vi /etc/cmcluster/cmclconfig.ascii
```

8. Let's add a non-root user to the list of users who can execute certain Serviceguard commands. In the cluster ascii file, give the oracle user Access Control rights that will enable him/her to monitor packages within the cluster. Allow this user to do this from either node in your cluster.

Answer:

```
# vi /etc/cmcluster/cmclconfig.ascii and add the following:
```

```
USER_NAME    oracle
USER_HOST    CLUSTER_MEMBER_NODE
USER_ROLE    MONITOR
```

9. Now, on your primary node, use the `cmcheckconf` command to check the cluster-wide ASCII configuration file for errors. Fix any errors and re-check if necessary:

Answer:

```
# cmcheckconf -v -C cmclconfig.ascii
```

10. In order to see what files are created by the `cmapplyconf` command (*see step 11 below*), view the contents of the `/etc/cmcluster` directory as it is now, before we do step 10.

Answer:

```
# cd /etc/cmcluster
# ll
```

Cluster Concepts and Configuration

11. Next, use the `cmapplyconf` command to apply the configuration. The `cmapplyconf` command will always (1) compile the ASCII file to binary form, and (2) distribute this binary file to all nodes in the cluster.

Answer:

```
# cd /etc/cmcluster
# cmapplyconf -v -C cmclconfig.ascii
```

12. Once again, view the contents of the `/etc/cmcluster` directory.

```
# ll /etc/cmcluster
```

Write the names of the file(s) created by the `cmapplyconf` command?

Files in the <code>/etc/cmcluster</code> directory		

13. Now, start cluster services on all nodes (*that is, start the cluster*). While it is a good practice to administer your cluster from one specific node, it is possible to start your cluster from any node (in the cluster).

Answer:

```
# cmruncl -v
```

14. Using the log file for `syslogd` (`/var/adm/syslog/syslog.log`), verify that the cluster has properly formed on all nodes in the cluster.

Answer:

```
# cd /var/adm/syslog
# tail -75 syslog.log | more
```

15. Now, view the complete status of the cluster. In the screen output, how many references do you see to "Quorum Server"?

Answer:

```
# cmviewcl -v
```

Number of references seen to "Quorum Server" → _____

Part 4: Test your Serviceguard Cluster

1. For our first test, from your primary node, halt the cluster.

Answer:

```
# cmhaltcl -v
```

2. Now, with the cluster halted, shutdown and power off any one node except your primary node. In Part 4 of this lab, we will call this node (*the node to be shutdown*) "Node2".

Note that, if you are using remote equipment, the shutdown -hy 0 command will automatically cause a <power-off>.

Answer:

```
# cd /                                (on Node2)
# shutdown -hy 0                      (on Node2)
< power off >                        (on Node2)
```

3. Now, on your primary node, attempt to restart the cluster. Note that we are starting the cluster from the primary node only, but the cmruncl command will apply to all the nodes in the cluster.

As the cluster is trying to start, watch your screen carefully, and answer the following questions:

- c. Does the cluster form properly? Why or why not?

Answer:

No. We do not have 100% of the cluster nodes.

- d. How can we restart the cluster if one node has become inoperable? See the manpage for cmruncl and note the -n option. Note that it is possible to start the cluster with less than 100% node attendance.

```
# man cmruncl
```

Start the cluster with this option.

Answer:

```
# cmruncl -v -n <Node1>
```


- Now, turn the power back on for Node2, and boot it up to run-level 3. If you are using remote equipment, issue a `pc` command from the Management Processor Command Menu.

Note: In order to interact with the system at the MP prompt, you will need to login to the MP on the system you shut down. If you do not know the MP hostname, ask your instructor.

To login to the MP, you'll need to supply an MP username and password. Once logged in, and at the MP prompt, type `CM` to enter the Command Menu. This is where you'll find the `pc` command, which is used to control power to your system.

After powering up the system, and while still at the `MP:CM>` prompt, type `MA` to return to the MP prompt. From here, type `C0` to access the system console and to login to unix once the system is back up.

Answer:

<code><power -on></code>	(Using local equipment)
or	
<code>MP:CM> pc</code>	(Using remote equipment)

- After Node2 is fully booted, check to see if it has automatically rejoined the cluster.

Answer:

```
# cmviewcl -v
```

- Examine the `/etc/rc.config.d` directory. Can you determine the name of the file in `/etc/rc.config.d` that controls whether or not a node automatically rejoins the cluster? **Hint:** you may be able to identify the file visually by listing the directory or, since all files in this directory are text files, a `grep` command might be handy.

```
# cd /etc/rc.config.d
# ll
```

The name of this file is _____.

Answer:

```
# cd /etc/rc.config.d
# ll
# grep -i cluster *
```

Cluster Concepts and Configuration

7. Since Node2 has not automatically rejoined the cluster (*step 5 above*), use the `cmrunnode` command to bring Node2 back into the running cluster. Afterward, check again to ensure that Node2 has rejoined the cluster and the cluster is now behaving correctly.

Answer:

```
# cmrunnode -v <Node2>
# cmviewcl -v
```

Congratulations on successfully starting and testing your Serviceguard cluster using a Quorum Server!

5-27. LAB Solution: Cluster Lock Using LVM Disks

Directions

In a previous module, we configured a shared volume group. Cluster Lock Disk functionality will be obtained from a disk in this volume group automatically. As such, if you elect to use an LVM Lock Disk in an I.T. environment, it is important to build the shared volume group first before building the cluster. Let's proceed with building our cluster.

Part 1: Configuring your Serviceguard Cluster

1. First, list the nodes that will comprise your Serviceguard cluster. Write their hostnames here:

Node1

Node2

2. Now, choose one node to be the "primary node" for your cluster. This is the node from which almost all cluster administration will be done. Write the hostname for your primary node here:

Primary Node

Special Note

For the remainder of the week, your primary node will be the node that is generally used for all cluster administration, unless specifically directed otherwise.

3. Next, on both nodes, and before we start our actual configuration, edit the /etc/lvmrc file. We do this so that volume groups used within the cluster will be activated by Serviceguard. We do not want these volume groups activated by default whenever the node reboots. To accomplish this, edit the file and set the AUTO_VG_ACTIVATE line equal to 0 (zero).

AUTO_VG_ACTIVATE=0

Answer:

```
# vi /etc/lvmrc
```

4. Now, on your primary node, set up the special file used by Serviceguard that enables you to initially configure your cluster. We do this on the primary node by creating a new file named `/etc/cmcluster/cmclnodelist`. **Important:** All nodes in the cluster must be listed in this file. The file is identical on every node in our cluster, and therefore we can easily ftp or rcp the file to every node in the cluster. The format for the file looks something like the following:

Node1	root
Node2	root
Node3	root
Node4	root

Answer:

```
# cd /etc/cmcluster
# vi cmclnodelist
# chmod 444 cmclnodelist
# rcp cmclnodelist Node2:/etc/cmcluster/.    (to transfer the file to Node2
                                              and also to all other nodes)
```

5. Now, we create the cluster-wide ASCII configuration file. This file is where all cluster-wide information will be stored.

Note that we must specify the name of the cluster-wide ASCII file and the names of each node in our (*soon-to-be*) cluster.

Answer:

```
# cd /etc/cmcluster
# cmquerycl -v -C cmclconfig.ascii -n <Node1> -n <Node2>
```

6. Examine the cluster-wide ASCII configuration file built by the `cmquerycl` command (*in the previous step*). Identify the following information:

To what value is the name of the cluster ("CLUSTER_NAME") set?

To what value is the HEARTBEAT_INTERVAL set?

To what value is the NODE_TIMEOUT set?

To what value is maximum number of packages ("MAX_CONFIGURED_PACKAGES") set?

LAN Interfaces	Node1	Node2
Which lan interface card will be used as HEARTBEAT_IP?		
Which lan interface card, if any, will be used as STATIONARY_IP?		
Which lan interface card, if any, will be used as a standby card (if you have an extra card)?		
Which lan interface card, if any, will be used as another standby card (if you have a second extra card)?		

Answer:

```
# cd /etc/cmcluster
# more cmclconfig.ascii
```

Module 5
Cluster Concepts and Configuration

7. Edit the cluster-wide ASCII configuration file to change some of these values (*seen above*):
- Change your cluster name to whatever your instructor asks. Ask your instructor for this new name and write it here _____.
 - Set the HEARTBEAT_INTERVAL to 3000000 microseconds (*3 seconds*).
 - Set the NODE_TIMEOUT value to 6000000 microseconds (*6 seconds*).
 - Set the AUTO_START_TIMEOUT value to 800000000 microseconds (*800 seconds*).

Answer:

```
# cd /etc/cmcluster
# vi cmclconfig.ascii
```

8. Let's add a non-root user to the list of users who can execute certain Serviceguard commands. In the cluster ascii file, give the oracle user Access Control rights that will enable him/her to monitor packages within the cluster. Allow this user to do this from either node in your cluster.

Answer:

```
# vi /etc/cmcluster/cmclconfig.ascii and add the following:

USER_NAME      oracle
USER_HOST      CLUSTER_MEMBER_NODE
USER_ROLE      MONITOR
```

9. Now, on your primary node, use the `cmcheckconf` command to check the cluster-wide ASCII configuration file for errors. Fix any errors and re-check if necessary:

Answer:

```
# cd /etc/cmcluster
# cmcheckconf -v -C cmclconfig.ascii
```

10. To see what files are created by the `cmapplyconf` command (*see step 10 below*), view the contents of the `/etc/cmcluster` directory as it is now, before we do step 10.

Answer:

```
# cd /etc/cmcluster
# ll
```

11. Next, still on your primary node, use the `cmapplyconf` command. The `cmapplyconf` command will always: (a) compile the `cmclconfig.ascii` file to binary, and (b) distribute the binary file to all nodes.

Answer:

```
# cd /etc/cmcluster
# cmapplyconf -v -C cmclconfig.ascii
```

12. Once again, view the contents of the `/etc/cmcluster` directory.
Write the names of the file(s) created by the `cmapplyconf` command?

Answer:

```
# cd /etc/cmcluster
# ll
```

Files in the <code>/etc/cmcluster</code> directory		

13. Now, start cluster services on all nodes (*that is, start the cluster*). While it is a good practice to administer your cluster from one specific node, it is possible to start your cluster from any node (in the cluster).

Answer:

```
# cmruncl -v
```

Cluster Concepts and Configuration

14. Using the log file for `syslogd` (`/var/adm/syslog/syslog.log`), verify that the cluster has properly formed on all nodes in the cluster.

Answer:

```
# tail -75 /var/adm/syslog/syslog.log | more
```

15. Using `cmviewcl -v`, verify that the cluster has properly formed on all nodes in the cluster.

Answer:

```
# cmviewcl -v
```


Part 2: Test your Serviceguard Cluster

1. For our first test, from your primary node, halt the cluster.

Answer:

```
# cmhaltcl -v
```

2. Next, with the cluster halted, shutdown and power off any one node except your primary node. In this part of the lab, we will call this node "Node2" (*the node to be shutdown*).

Note that, if you are using remote equipment, the shutdown -hy 0 command will automatically cause a <power-off>.

Answer:

```
# cd /                                (on Node2)
# shutdown -hy 0                      (on Node2)
<power-off>                          (on Node2)
```

3. Now, on your primary node, attempt to restart the cluster. Note that we are starting the cluster from the primary node only, but the `cmrunc1` command will apply to all the nodes in the cluster.

As the cluster is trying to start, watch your screen carefully, and answer the following questions:

- a. Does the cluster form properly? Why or why not?

Answer:

No. We do not have 100% of the cluster nodes.

- b. How can we restart the cluster if one node has become inoperable? See the manpage for `cmrunc1` and note the `-n` option. Note that it is possible to start the cluster with less than 100% node attendance.

```
# man cmrunc1
```

Start the cluster with this option.

```
# cmrunc1 -v -n <Node1>
```

Cluster Concepts and Configuration

4. Now turn the power back on for Node2, then boot Node2 up to run-level 3. If you are using remote equipment, issue a `pc` command from the Management Processor Command menu.

Note: In order to interact with the system at the MP prompt, you will need to login to the MP on the system you shut down. If you do not know the MP hostname, ask your instructor.

To login to the MP, you'll need to supply an MP username and password. Once logged in, and at the MP prompt, type `CM` to enter the Command Menu. This is where you'll find the `pc` command, which is used to control power to your system.

After powering up the system, and while still at the `MP:CM>` prompt, type `MA` to return to the MP prompt. From here, type `C0` to access the system console and to login to unix once the system is back up.

Answer:

```
<power -on>
```

(Using local equipment)

or

```
MP:CM> pc
```

(Using remote equipment)

5. After Node2 is fully booted, check to see if Node2 has automatically rejoined the cluster?

Answer:

```
# cmviewcl -v
```

6. Now, examine the `/etc/rc.config.d` directory. Can you discover the name of the file in `/etc/rc.config.d` that controls whether or not a node automatically rejoins the cluster? **Hint:** you may be able to identify the file visually by listing the directory or, since all files in this directory are text files, a `grep` command might be handy.

The name of this file is _____.

Answer:

```
# cd /etc/rc.config.d
```

```
# ll
```

```
# grep -i cluster *      (the file is /etc/rc.config.d/cmcluster)
```

7. Since Node2 has not automatically rejoined the cluster (*item 5 above*), use the `cmrunnode` command to bring Node2 back into the running cluster. Afterward, check again to ensure that Node2 has rejoined the cluster and the cluster is now behaving correctly.

Answer:

```
# cmrunnode -v <node2>  
# cmviewcl -v
```

Congratulations on successfully configuring, starting, and testing your Serviceguard cluster using an LVM Lock Disk.

Module 6 — Additional Cluster Features

Objectives

Upon completion of this module, you will be able to do the following:

- Mark a volume group as belonging to a Serviceguard cluster.
- Activate a shared volume group in an exclusive read/write access mode.
- Configure and test the local LAN failover capability of the Serviceguard cluster.
- Describe the differences between heartbeat LAN cards and standby LAN cards.
- List two methods in which a node can *join* an existing cluster.
- List two methods in which a node can *leave* an existing cluster.

6-1. SLIDE: Additional Cluster Features



Additional Cluster Features

In this module, we will discuss in detail ...

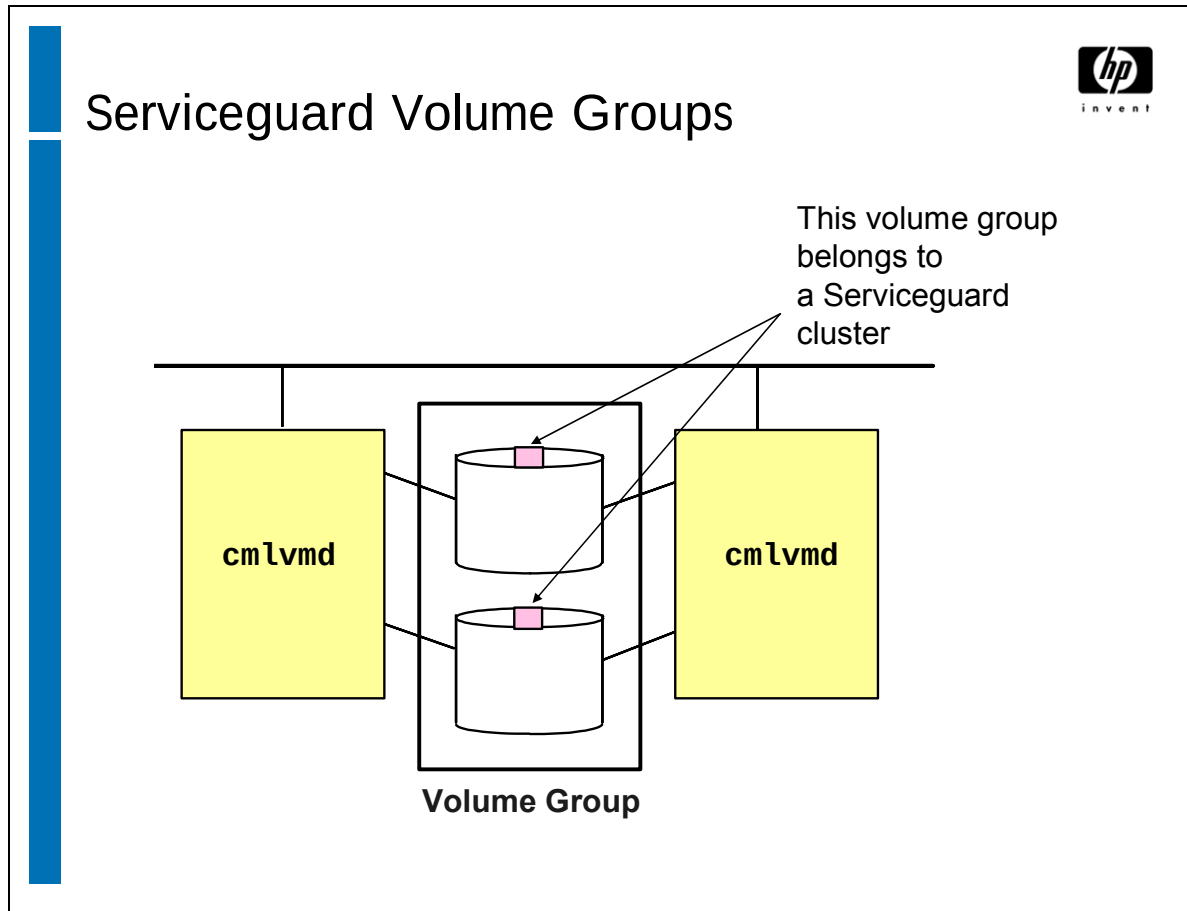
- Serviceguard Volume Groups
- Node Failures, Node Exits, and Node Joins
- Local LAN Card Failover

Student Notes

This module focuses on additional features of a Serviceguard cluster which have not yet been highlighted. These features include:

- LVM enhancements specifically for Serviceguard that greatly improve the integrity data stored in a Serviceguard high availability environment.
- Commands that allow control over node "exits" and node "joins", which provide flexibility when managing a Serviceguard cluster.
- Local LAN Card failover which provides redundancy protection for the local LAN cards.

6-2. SLIDE: Serviceguard Volume Groups



Student Notes

When LVM was designed in the late 1980s, the situation in which two systems would be cabled to the same volume group was not considered. As a result, if two systems cabled to the same volume group try to access the shared volume group concurrently using standard LVM functionality, LVM will allow it.

This means that both systems could activate the volume group concurrently using standard LVM. Both systems could mount the shared file system concurrently using standard LVM, and both systems could try to create a file in the exact same spot of the file system concurrently using standard LVM. Because each system contains its own view of the file system and that view would be written out to disk every 30 seconds by flushing the buffer cache, the shared file system would become corrupted very quickly.

To remedy this situation, the standard LVM functionality was modified to indicate when a volume group belonged to a Serviceguard cluster. If a volume group is part of a Serviceguard cluster, only one node will be allowed to access the volume group at a time. If a second system tries to access the volume group while the first system has it activated, an error will be returned to the second system indicating that the volume group is already active by another node in the cluster.

How Do Serviceguard Volume Groups Work?

In the LVM header structure, a single bit (which was previously unused with standard LVM) is turned on to indicate membership with a Serviceguard cluster. If the bit is turned on, the volume group is assumed to be part of a Serviceguard cluster and only one node will be allowed to activate the volume group at a time.

In addition, the cluster ID is written to the volume group headers so that no other cluster can activate the volume group.

The enforcement of single node access to a volume group is done through the `cm_lvmd` daemon process. This process must always be running on every node while the cluster is active. When a node tries to activate a volume group with the `vgchange` command, the request is passed to the `cm_lvmd` daemon for approval. The daemon determines activation by broadcasting a message to all other `cm_lvmd` daemon processes on other systems, asking if they currently have the volume group in question active. If a node already has the volume group active, then its corresponding `cm_lvmd` daemon will respond indicating the volume group is already active, causing the attempted activation request to fail.

A volume group can only be activated if all other `cm_lvmd` daemons respond with a status of not active for the volume group attempting to be activated.

6-3. SLIDE: Marking Volume Groups for use in Serviceguard

	
Marking Volume Groups for use in Serviceguard	
Marking Volume Group for Serviceguard	<code>vgchange -c y VGName</code>
Marking Volume Group as non-Serviceguard	<code>vgchange -c n VGName</code>
Standard Volume Group Activation (classical LVM)	<code>vgchange -a y VGName</code>
Standard Volume Group Deactivation (classical LVM)	<code>vgchange -a n VGName</code>
Exclusive Volume Group Activation	<code>vgchange -a e VGName</code>
Exclusive Volume Group Deactivation (same as classical)	<code>vgchange -a n VGName</code>

Student Notes

New arguments and options have been added to the `vgchange` command to support Serviceguard marked volume groups.

Marking a Volume Group for Serviceguard

The `-c` option has been added to the `vgchange` command to enable you to mark a volume group for clustering. When a volume group is marked for clustering, only one node in the cluster at a time will be allowed to activate the cluster.

The `vgchange -c y VGName` command marks a volume group as part of a cluster. The `vgchange -c n VGName` command removes an existing mark, indicating that the volume group is no longer part of a cluster. These options are applied automatically by the `cmapplyconf` command when the volume group is listed in the cluster-wide ASCII file.

Volume Group Activation Modes

There are two ways to activate a volume group for write access: shared access and exclusive access:

Shared Read/Write Activation

The `vgchange -a y VGName` command activates a volume group for *shared* read/write access.

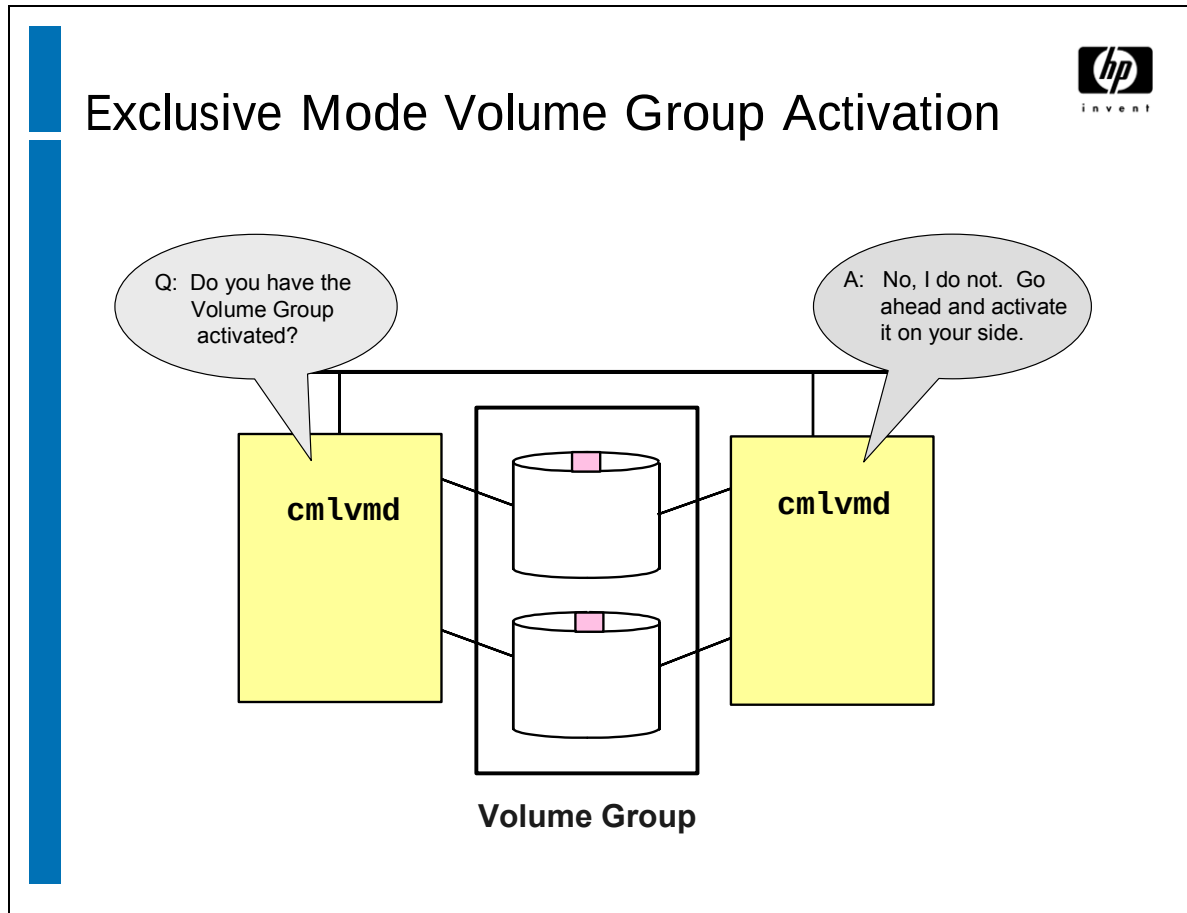
When activated in shared read/write mode, both nodes can write to the file system concurrently. This is standard LVM functionality. This mode of activation is not supported for Serviceguard volume groups. This mode of activation can only be used on non-marked volume groups.

Exclusive Read/Write Activation

The `vgchange -a e VGName` command activates a volume group for *exclusive* read/write access.

When a node activates a volume group in exclusive read/write mode, the node will be the only node to have write access to the volume group. This mode of activation only works if no other node in the cluster has already activated the volume group. This mode of activation is the only supported activation mode for Serviceguard marked volume groups.

6-4. SLIDE: Exclusive Mode Volume Group Activation



Student Notes

Exclusive volume group activation means only one node will have write access to the volume group.

There are two requirements for exclusive activation:

- The volume group must be marked as a clustered volume group. The cluster ID on the volume group must match the cluster ID stored in the `/etc/cmcluster/cmclconfig` file. If the volume group has not been marked, then exclusive activation is not allowed.

The `cm1vmd` daemon process must be up and running. If `cm1vmd` is not running, there is no way to check if another node has already activated the volume group. The only way to start `cm1vmd` is to bring up the Serviceguard cluster. HP does not support running `cm1vmd` outside of Serviceguard.

6-5. SLIDE: Cluster Formation and Reformations


invent

Cluster Formation and Reformations

Two concepts and an example of forming or reforming the cluster:

- Ways to initially form the cluster
- Node failures, node exits, and node joins
- Cluster reformation example

Student Notes

Another feature of Serviceguard clusters is the flexibility to allow nodes to easily exit and join existing clusters.

The next few slides highlight the cluster formation and reformation processes.

6-6. SLIDE: Ways to Initially Form the Cluster



Ways to Initially Form the Cluster

Two ways to initially form the cluster:

1. **Manually**
Execute the `cmruncl` command.
2. **Automatically at System Startup**
Edit the `/etc/rc.config.d/cmcluster` file.

AUTOSTART_CMCLD=1

Student Notes

Manual Cluster Startup

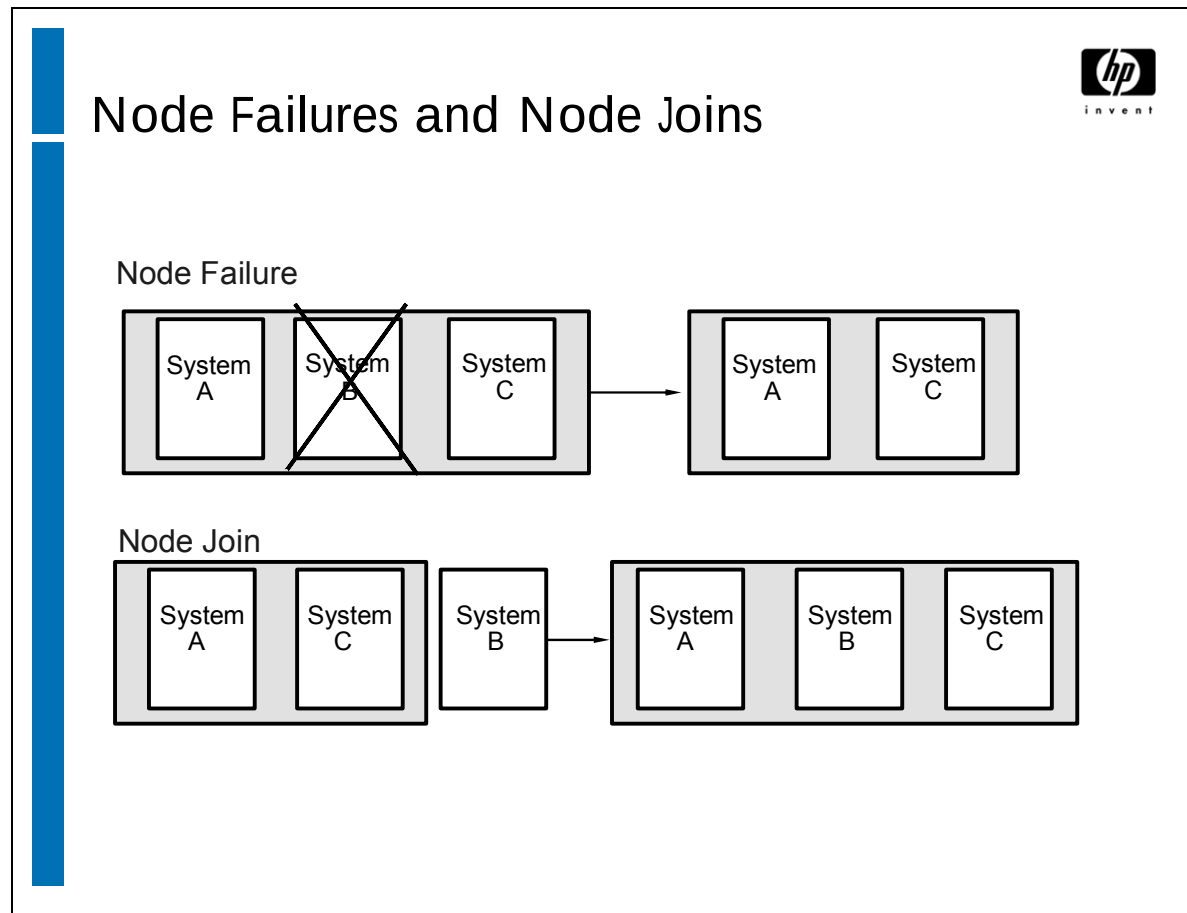
The `cmruncl` command is used to manually initiate the cluster formation. Any node configured as part of the cluster can run this command if the cluster is not already up. This command is typically used the first time the cluster is brought up or after any cluster maintenance operation (or modification) that requires the cluster be brought down.

Automatic Cluster Startup on Each Node

To enable each node in the cluster to automatically start or join the cluster at boot time, edit the `/etc/rc.config.d/cmcluster` file and set the value of `AUTOSTART_CMCLD` to 1.

At boot time, if the cluster already exists, the node will try to join it; if no cluster exists, the node will attempt to start the cluster, provided all other nodes are available. If all other nodes are not yet available, the node will wait up to the period (as defined in the Cluster ASCII file) to form the cluster. If all nodes do not become available by the end of the `AUTO_START_TIMEOUT` period, the cluster will not form automatically.

6-7. SLIDE: Node Failures and Node Joins



Student Notes

The cluster reforms every time a change is made to cluster membership. The three methods by which membership changes are **node failure**, **node exit**, and **node join**.

Node Failure and Node Exit

Anytime a node leaves the cluster, a reformation takes place to exclude the exiting node. Nodes could leave the cluster for the following reasons:

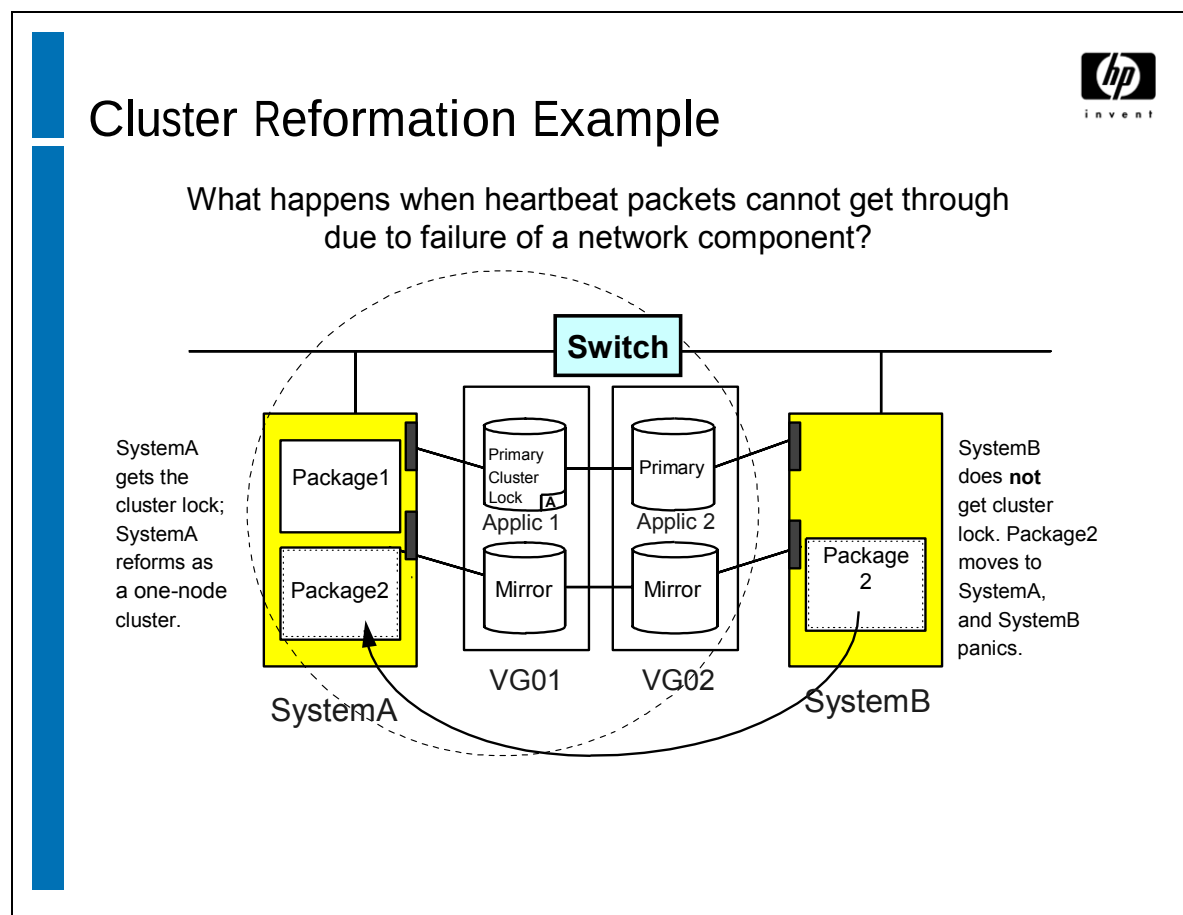
- The `cmhaltnode` command was run to manually remove the node from the cluster.
- A hardware failure occurred causing the node to crash.
- Heavy network traffic prohibits the heartbeat messages from being received causing the cluster coordinator to think the node was unhealthy.

Node Join

Anytime an inactive node wants to join an existing cluster, a reformation takes place. Nodes join a cluster for the following reasons:

- The `cmrunnode` command was run to manually add the node to the cluster.
- A node reboots after a panic and is configured to automatically join the cluster at boot time.

6–8. SLIDE: Cluster Reformation Example



Student Notes

The slide illustrates what happens during a cluster reformation in which both nodes receive exactly 50 percent of the vote.

The Situation

Assume a two-node cluster, with Package1 running on SystemA and Package2 running on SystemB. Volume group vg01 is exclusively activated on SystemA, volume group vg02 is exclusively activated on SystemB. In addition, package IP addresses are assigned to SystemA and SystemB respectively (this is not shown on the slide).

The Failure

Only one LAN has been configured for both heartbeat and data traffic. Failure of network equipment (such as a hub, switch, cable, or LAN card) would prevent heartbeat packets from getting through.

Since SystemA does not receive heartbeat messages from SystemB, SystemA attempts to reform as a one-node cluster. Likewise, since SystemB does not receive heartbeat messages from SystemA, SystemB also attempts to reform as a one-node cluster. During the election protocol, each node votes for itself, yielding both nodes with 50 percent of the vote. At this

point, a true race condition exists between SystemA and SystemB for the cluster lock disk. Because both nodes have 50 percent of the vote, both nodes go for the cluster lock. Only one will get it.

The Outcome

Assume SystemA gets the cluster lock. SystemA reforms as a one-node cluster. After reformation, SystemA will make sure all applications configured to run on an existing clustered node are running. When SystemA discovers Package2 is not running in the cluster, it will try to start Package2 if Package2 is configured to run on SystemA.

SystemB recognizes that SystemA has begun reforming as a one node cluster because SystemA has the cluster lock. To release all resources related to Package2 (such as exclusive access to volume group vg02 and the Package2 IP address) as quickly as possible, SystemB will perform a transfer of control (TOC) panic.

6-9. SLIDE: Local LAN Card Failover


invent

Local LAN Card Failover

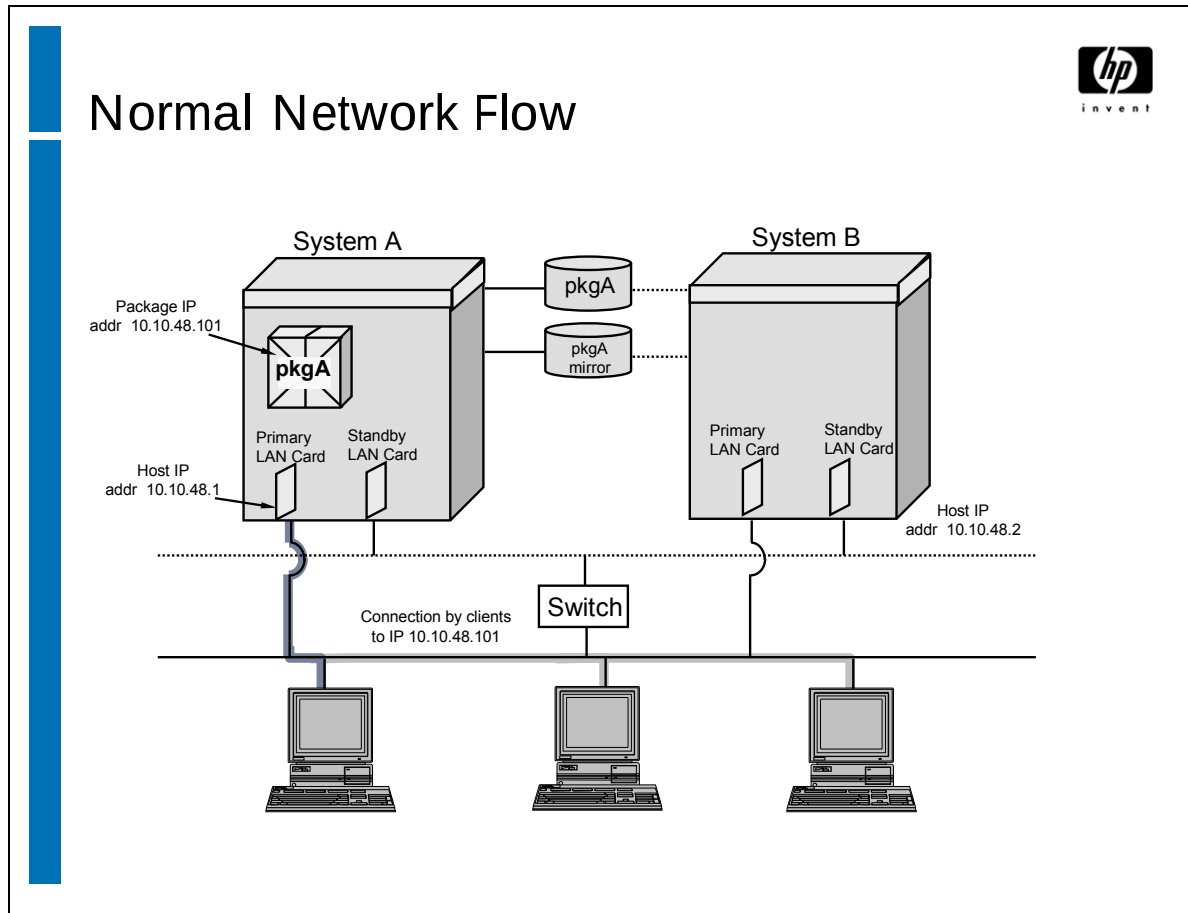
- Normal network flow through primary LAN card
- Network flow through standby LAN card

Student Notes

The "local LAN card failover" feature allows every LAN card to have a corresponding standby LAN card. If the LAN card fails, all IP addresses on that LAN card would move to the standby LAN card, preventing an application outage from occurring.

The details of the local LAN card failover feature are illustrated on the next two slides.

6-10. SLIDE: Normal Network Flow

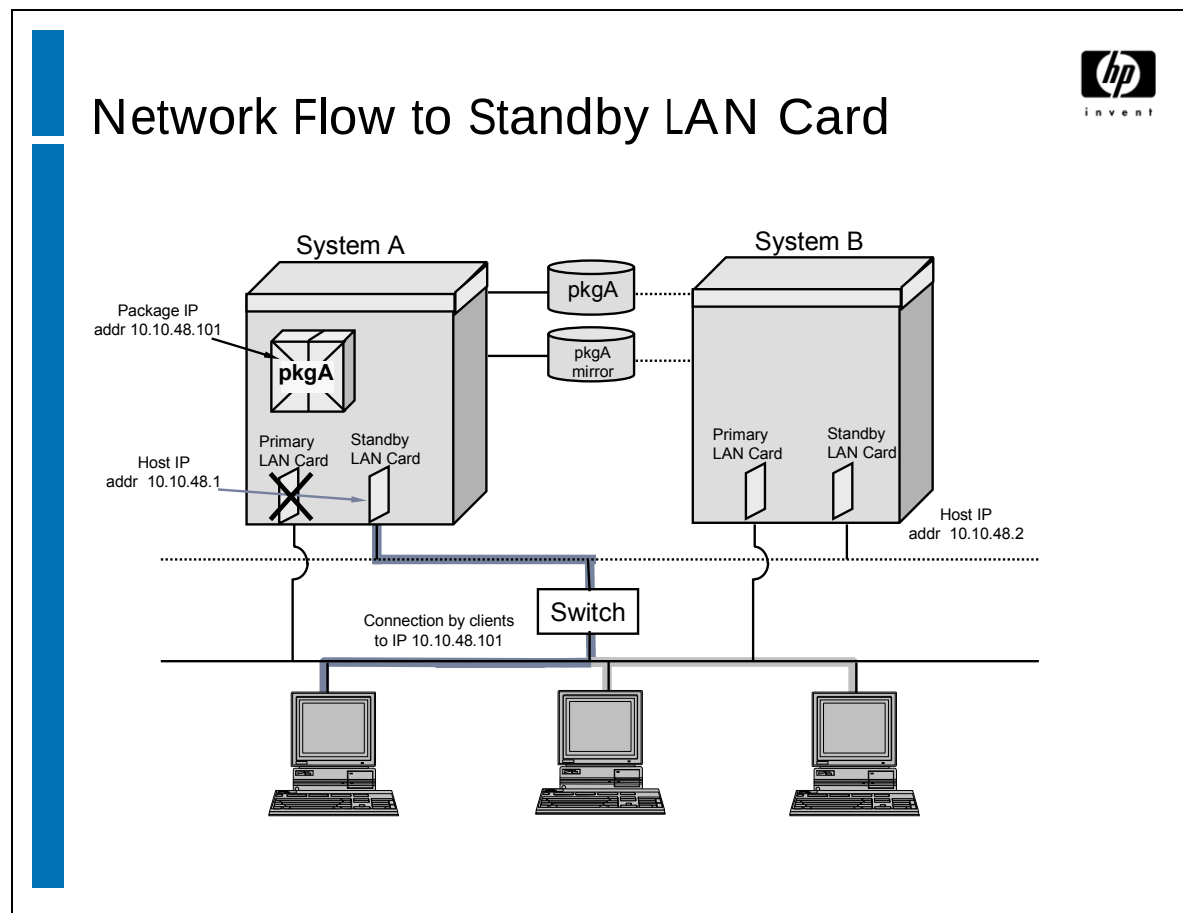


Student Notes

The above slide shows two Serviceguard nodes connected to a single "bridged network", where the bridged network consists of two LAN segments (an upper and lower segment) joined by a hub. SystemA and SystemB are communicating with each other and the client systems over the lower LAN segment. The upper LAN segment in the slide is considered the "standby" segment.

Before turning to the next page, try to answer the following question: *"What would happen if the primary LAN card in SystemA were to fail?"*

6-11. SLIDE: Network Flow to Standby LAN Card



Student Notes

In the above slide, we see the answer to the question: "What would happen if the primary LAN card in SystemA were to fail?"

When the primary LAN card fails, the Serviceguard daemon, `cmcl`, detects the failure within the "NETWORK_POLLING_INTERVAL", and moves all IP addresses configured on the primary LAN card over to the standby LAN card. The switch is transparent at the TCP/IP level and all applications on SystemA continue to run on SystemA.

6-12. LAB: Shared Disks/Shared Volume Groups

Directions

1. First, verify that the file system (/sgdata1) is not mounted on either node.
(umount /sgdata1 if necessary)
2. Verify that vg01 is not activated on either node.
3. Ensure the cluster is up with both nodes as members.
4. Ensure that vg01 is marked as a "**cluster-aware**" volume group.
5. Try the following on node1:

```
# vgchange -a y vg01
```


What happens?

6. Again, test: on node1, issue the following command:

```
# vgchange -a e vg01
```

What happens?

7. Now test node2. On node2, issue the following command:

```
# vgchange -a e vg01
```

What happens?

8. Back on node1, issue the following command:

```
# vgchange -a n vg01
```

9. Now, bring down the cluster with a `cmhaltcl` command.

```
# cmhaltcl -v
```

10. With the cluster in a halted state, on node1, issue the following command:

```
# vgchange -a y vg01
```

This still does not work.

11. Now, on node1, issue the following command:

```
# vgchange -a e vg01
```

What happens?

12. Can we access our shared data in any way with the cluster down and the daemons not running? For example, we might want to run a database conversion or maintenance program with the cluster down. Can we do this?

13. On node1, issue the following commands:

```
# vgchange -c n vg01  
# vgchange -a y vg01  
# mount -F vxfs /dev/vg01/sglvol1 /sgdata1
```

14. Try accessing the data on the file system; use the traditional `ll` command. After that, unmount the filesystem.

15. Deactivate volume group `vg01`.

NOTE: You must return `vg01` to the cluster with `vgchange -c y vg01`. This, however, will not work without the Serviceguard daemons `cmlvmd` and `cmclld` running.

16. Try to mark the volume group `vg01` for cluster use. Does this command work properly?

17. Restart the cluster.

18. Mark the volume group `vg01` for cluster use now that cluster has been restarted. This command should work properly now!

6-13. LAB: Exiting and Joining the Cluster

Directions

1. With the cluster up, both nodes as members, and
tail -f /var/adm/syslog/syslog.log running on both systems, on node1, issue
the cmhaltnode command.

```
# cmhaltnode                ← on <node1>
```

2. Check the following: Are both nodes still part of the cluster?
3. On node1, issue a cmrunnode command.
4. Next, configure **both** nodes to automatically join the cluster at boot time.

Additional Cluster Features

5. Now, shutdown and power off one of the nodes. Let us choose node2 for this purpose. Remember: if you are using local equipment, just physically turn-off the power switch. If you are using remote equipment, the shutdown command will automatically power off the system.
6. On the node that is still up, view the status of the cluster.
7. Next, halt the cluster.
8. With the cluster halted, attempt to start the cluster while one node is down.
9. Did the cluster start? Why or why not?
10. Force the cluster to start without the failed node. You will receive a warning. Reply "y" to continue to bring up the cluster.

Additional Cluster Features

11. Power-on node2, and allow it to come to run level 3. On remote equipment, go to the GSP / MP command menu and issue a pc command.

6-14. LAB: Local LAN Card Failover (Using Remote Equipment)

Special Note

- a. Lab 6-14 is for those using remote equipment (and do not have the ability to physically disconnect a LAN cable).
- b. If you have local equipment (and the ability to physically disconnect a LAN cable), do not do this lab. Instead, Turn to Lab 6-15.
- c. If you are not sure which equipment you have, ask your instructor.

1. Execute the commands from the indicated nodes below. From node2, ping node1. Display the arp cache on node2. Make a note of the MAC and IP addresses of the LAN cards on Node1 using lanscan and netstat -in.

Answer:

```
<node2># ping <node1>
<node2># arp -a | grep <node1>
```

Node1's IP Address: _____ Node1's MAC Address: _____

```
<node1># lanscan ; netstat -in
```

LAN IP _____	LAN MAC Address _____
LAN IP _____	LAN MAC Address _____
LAN IP _____	LAN MAC Address _____

2. On node1, and with the cluster up and with both nodes in the cluster, execute `tail -f /var/adm/syslog/syslog.log`.
3. Open another session on node1 and run the following script, which will simulate the failure of a LAN card. Run `/labs/mod_06/simulate_lan_card_failure.sh`. Allow this script to continue to run as you proceed to the next step.

4. What do you notice in the `syslog.log` file?
5. Open another session and view the cluster information.
6. Did Serviceguard detect the failure?
7. Did it automatically switch to the standby LAN card?
8. Verify the LAN failover. Issue the commands `lanscan` and `netstat -in` on `node1` and compare the output with what you found from step #1. Also, try another `cmviewcl -v`. This confirms that the system uses the backup LAN card when the primary LAN is down.
9. Terminate the `simulate_lan_card_failure.sh` script with a `<ctrl-c>`. Then, using the commands from step 8 above, verify that Serviceguard automatically fails back to the primary LAN card.

6-15. LAB: Local LAN Card Failover (Using Local Equipment)

Directions

Special Note

- Lab 6-15 is for those using local equipment where you have the ability to physically disconnect a LAN cable.
- If you are using remote equipment, please do lab 6-14.
- If you are not sure which equipment you have, ask your instructor.

- Execute the commands on from the indicated nodes below. From node2, ping node1. Display the arp cache on node2. Make a note of the Ethernet and IP addresses of the LAN cards on Node1 using lanscan and netstat -in.

Answer:

```
<node2># ping <node1>
<node2># arp -a | grep <node1>
```

Node1's IP Address: _____ Node1's MAC Address: _____

```
<node1># lanscan ; netstat -in
```

LAN IP _____	LAN MAC Address _____
LAN IP _____	LAN MAC Address _____
LAN IP _____	LAN MAC Address _____

- On node1, and with the cluster up and with both nodes in the cluster, and in the Putty window, execute `tail -f /var/adm/syslog/syslog.log`. Back in node1's Graphical window, issue the following:

```
<node1># xclock -update 1 -display __ (see comment below) __ &
```

To fill in the correct value in the blank space in the above command, use your ReflectionX window on node2 and capture the output of the “`echo $DISPLAY`” command. Place this output in the blank in the above command. Also, do not forget to place this `xclock` process in the background.

3. With the second hand moving on our `xclock` process, disconnect the primary LAN card on `node1` from the network.

4. What happens to the second hand on our `xclock`?

5. View the cluster information.

6. Did Serviceguard detect the failure?

7. Did it automatically switch to the standby LAN card?

8. Issue an `"arp node1"` from the command line on `node2` and note the IP address or Ethernet address of `node1`.

```
# arp node1
```

Additional Cluster Features

9. Verify the LAN failover. Issue the commands `lanscan` and `netstat -in` and `ifconfig lan0` and `ifconfig lan1` on `node1`. Also, try another `cmviewcl -v`. This confirms that the system uses the backup LAN card when the primary is disconnected. Compare the output with what you found in step #1 above.

10. Reconnect the primary LAN card on `node1` and, using the commands from step 9 above, verify that Serviceguard automatically fails back to the primary LAN card.

Finally, identify the `xc lock` process ID. Use the PID to kill the process.

6-16. LAB Solution: Shared Disks/Shared Volume Groups

Directions

1. First, verify that the file system (/sgdata1) is not mounted on either node.
(umount /sgdata1 if necessary)

Answer:

```
# bdf
# umount /sgdata1    ← Only if needed.
```

2. Verify that vg01 is not activated on either node.

Answer:

```
# vdisplay vg01          ← on Node1 and on Node2
# vgchange -a n vg01     ← if needed on either / both nodes
```

3. Ensure the cluster is up with both nodes as members.

Answer:

```
# cmviewcl
```

4. Ensure that vg01 is marked as a "**cluster-aware**" volume group.

Answer:

```
# vgchange -c y vg01
```

5. Try the following on node1:

```
# vgchange -a y vg01
```

What happens?

Answer:

The activation fails: "Conflicts with configured mode."

Additional Cluster Features

6. Again, test: on node1, issue the following command:

```
# vgchange -a e vg01
```

What happens?

Answer:

Succeeds, since we used the "exclusive" option.

7. Now test node2. On node2, issue the following command:

```
# vgchange -a e vg01
```

What happens?

Answer:

Fails, since node1 has the exclusive activation on vg01.

8. Back on node1, issue the following command:

```
# vgchange -a n vg01
```

9. Now, bring down the cluster with a cmhaltctl command.

```
# cmhaltctl -v
```

10. With the cluster in a halted state, on node1, issue the following command:

```
# vgchange -a y vg01
```

This still does not work.

11. Now, on node1, issue the following command:

```
# vgchange -a e vg01
```

What happens?

Answer:

Fails because the cluster is not running.

12. Can we access our shared data in any way with the cluster down and the daemons not running? For example, we might want to run a database conversion or maintenance program with the cluster down. Can we do this?

Answer:

Yes.

13. On node1, issue the following commands:

```
# vgchange -c n vg01
# vgchange -a y vg01
# mount -F vxfs /dev/vg01/sglvol1 /sgdata1
```

14. Try accessing the data on the file system; use the traditional `ll` command. After that, unmount the filesystem.

Answer:

```
# ll /sgdata1
# umount /sgdata1
```

15. Deactivate volume group `vg01`.

Answer:

```
# vgchange -a n vg01
```

NOTE: You must return `vg01` to the cluster with `vgchange -c y vg01`. This, however, will not work without the Serviceguard daemons `cmclmd` and `cmclld` running.

16. Mark the volume group `vg01` for cluster use. Does this command work properly?

Answer:

```
# vgchange -c y vg01
```

No.

17. Restart the cluster.

Answer:

```
# cmrunc1 -v
```

18. Mark the volume group `vg01` for cluster use now that cluster has been restarted. This command should work properly now!

Answer:

```
# vgchange -c y vg01
```

6-17. LAB Solution: Exiting and Joining the Cluster

Directions

1. With the cluster up, both nodes as members, and
tail -f /var/adm/syslog/syslog.log running on both systems, on node1, issue
the cmhaltnode command.

```
# cmhaltnode ← on <node1>
```

2. Check the following: Are both nodes still part of the cluster?

Answer:

```
# cmviewcl  
No, <node1> is no longer in the cluster.
```

3. On node1, issue a cmrunnode command.

Answer:

```
# cmrunnode ← on <node1>
```

4. Next, configure **both** nodes to automatically join the cluster at boot time.

Answer:

```
# cd /etc/rc.config.d  
# vi cmcluster ← On both nodes !
```

```
AUTOSTART_CMCLD=1
```

5. Now, shutdown and power off one of the nodes. Let us choose node2 for this purpose. Remember: if you are using local equipment, just physically turn-off the power switch. If you are using remote equipment, the shutdown command will automatically power off the system.

Answer:

```
# cd /  
# shutdown -hy 0          ← On node2  
<power-off>              ← Only with local equipment
```

6. On the node that is still up, view the status of the cluster.

Answer:

```
# cmviewcl -v
```

7. Next, halt the cluster.

Answer:

```
# cmhaltcl -v
```

8. With the cluster halted, attempt to start the cluster while one node is down.

Answer:

```
# cmruncl -v
```

9. Did the cluster start? Why or why not?

Answer:

No, since we have less than 100% of the nodes present.

10. Force the cluster to start without the failed node. You will receive a warning. Reply "y" to continue to bring up the cluster.

Answer:

```
<node1> # cmruncl -n <node1>
```

11. Power-on node2, and allow it to come to run level 3. On remote equipment, go to the GSP / MP command menu and issue a pc command.

Answer:

<power-on> ← Using local equipment.

MP> cm ← Using remote equipment.
MP:CM> pc

12. Did node2 successfully rejoin the cluster? Briefly, please discuss why or why not.

Answer:

<Node2> rejoins the cluster at boot because we set AUTOSTART_CMCLD=1 in its /etc/rc.config.d/cmcluster file.

6-18. LAB Solution: Local LAN Card Failover (Using Remote Equipment)

Special Note

- d. Lab 6-14 is for those using remote equipment (and do not have the ability to physically disconnect a LAN cable).
- e. If you have local equipment (and the ability to physically disconnect a LAN cable), do not do this lab. Instead, Turn to Lab 6-15.
- f. If you are not sure which equipment you have, ask your instructor.

1. Execute the commands from the indicated nodes below. From node2, ping node1. Display the arp cache on node2. Make a note of the MAC and IP addresses of the LAN cards on Node1 using lanscan and netstat -in.

Answer:

```
<node2># ping <node1>
<node2># arp -a | grep <node1>
```

Node1's IP Address: _____ Node1's MAC Address: _____

```
<node1># lanscan ; netstat -in
```

LAN IP _____	LAN MAC Address _____
LAN IP _____	LAN MAC Address _____
LAN IP _____	LAN MAC Address _____

2. On node1, and with the cluster up and with both nodes in the cluster, execute `tail -f /var/adm/syslog/syslog.log`.
3. Open another session on node1 and run the following script, which will simulate the failure of a LAN card. Run `/labs/mod_06/simulate_lan_card_failure.sh`. Allow this script to continue to run as you proceed to the next step.

Answer:

```
<node1># /labs/mod_06/simulate_lan_card_failure.sh
```

4. What do you notice in the `syslog.log` file?

Answer:

`/var/adm/syslog/syslog.log` reflects the LAN card failover.

5. Open another session and view the cluster information.

Answer:

```
# cmviewcl -v
```

6. Did Serviceguard detect the failure?

Answer:

Yes. See entries in the `syslog.log` file and note that the primary lan card for node1 shows a status of “down” in the `cmviewcl -v` output.

7. Did it automatically switch to the standby LAN card?

Answer:

Yes. See entries in the `syslog.log` file.

8. Verify the LAN failover. Issue the commands `lanscan` and `netstat -in` on node1 and compare the output with what you found in step #1. Also, try another `cmviewcl -v`. This confirms that the system uses the backup LAN card when the primary LAN is down.

Answer:

```
# lanscan
# netstat -in
# cmviewcl -v
```

9. Terminate the `simulate_lan_card_failure.sh` script with a `<ctrl-c>`. Then, using the commands from the step above, verify that Serviceguard automatically fails back to the primary LAN card. Note that it may take 10-15 seconds for the “failback” to complete.

Answer:

```
< ctrl-c >
# lanscan
# netstat -in
# ifconfig lan0
# cmviewcl -v
```

6–19. LAB Solution: Local LAN Card Failover (Using Local Equipment)

Directions

Special Note

- d. Lab 6-15 is for those using local equipment where you have the ability to physically disconnect a LAN cable.
- e. If you are using remote equipment, please do lab 6-14.
- f. If you are not sure which equipment you have, ask your instructor.

1. Execute the commands on from the indicated nodes below. From node2, ping node1. Display the arp cache on node2. Make a note of the Ethernet and IP addresses of the LAN cards on Node1 using `lanscan` and `netstat -in`.

Answer:

```
<node2># ping <node1>
<node2># arp -a | grep <node1>
```

Node1's IP Address: _____ Node1's MAC Address: _____

```
<node1># lanscan ; netstat -in
```

LAN IP _____	LAN MAC Address _____
LAN IP _____	LAN MAC Address _____
LAN IP _____	LAN MAC Address _____

2. On node1, and with the cluster up and with both nodes in the cluster, and in the Putty window, execute `tail -f /var/adm/syslog/syslog.log`. Back in node1's Graphical window, issue the following:

```
<node1># xclock -update 1 -display __ (see comment below) __ &
```

To fill in the correct value in the blank space in the above command, use your ReflectionX window on node2 and capture the output of the “`echo $DISPLAY`” command. Place this output in the blank in the above command. Also, do not forget to place this `xclock` process in the background.

3. With the second hand moving on our `xclock` process, disconnect the primary LAN card on `node1` from the network.

Answer:

Unplug the LAN cable connected to the primary LAN card.

4. What happens to the second hand on our `xclock`?

Answer:

The second hand pauses (reflecting the lan card failover), then skips forward again to reflect the correct time.

Additionally, `/var/adm/syslog/syslog.log` reflects the lan card failover.

5. View the cluster information.

Answer:

```
# cmviewcl -v
```

6. Did Serviceguard detect the failure?

Answer:

Yes. See the `syslog.log` file.

7. Did it automatically switch to the standby LAN card?

Answer:

Yes. See the `syslog.log` file.

8. Issue an `"arp node1"` from the command line on `node2` and note the IP address or Ethernet address of `node1`.

```
# arp node1
```

9. Verify the LAN failover. Issue the commands `lanscan` and `netstat -in` and `ifconfig lan0` and `ifconfig lan1` on node1. Also, try another `cmviewcl -v`. This confirms that the system uses the backup LAN card when the primary is disconnected.

Answer:

```
# lanscan
# netstat -in
# ifconfig lan0
# cmviewcl -v
```

Compare the output with what you found in step #1 above.

10. Reconnect the primary LAN card on node1 and, using the commands from step 9 above, verify that Serviceguard automatically fails back to the primary LAN card.

Answer:

```
# lanscan
# netstat -in
# ifconfig lan0
# cmviewcl -v
```

11. Finally, identify the `xclock` process ID. Use the PID to kill the process.

Answer:

```
# ps -ef | grep xclock
# kill <pid>
```

Module 7 — Packages and Services

Objectives

Upon completion of this module, you will be able to do the following:

- Configure a basic package to run in a Serviceguard environment.
- Describe the fields in a package configuration file.
- Describe the variables in a package control script file.
- Provide the command to re-enable a package for switching.
- Interpret the output of `cmviewcl -v` as it relates to packages.
- Provide the full path name of a package-specific log file.

7-1. SLIDE: Packaging Concepts



Packaging Concepts

- A package is an application and all the application's resources required to execute properly.
- Resources for packages may include:
 - Disk Resources (Volume Group(s), Disk Group(s))
 - Network Resources (IP Address(es))
 - Service Processes
- Packaging Files include:
 - Package Configuration File (pkg.conf)
 - Package Control Script (pkg.cntl)

Student Notes

Applications that run in a Serviceguard high availability environment must be configured with all their related resources into a **package**.

The information needed to run a package in a Serviceguard cluster is contained in the ...

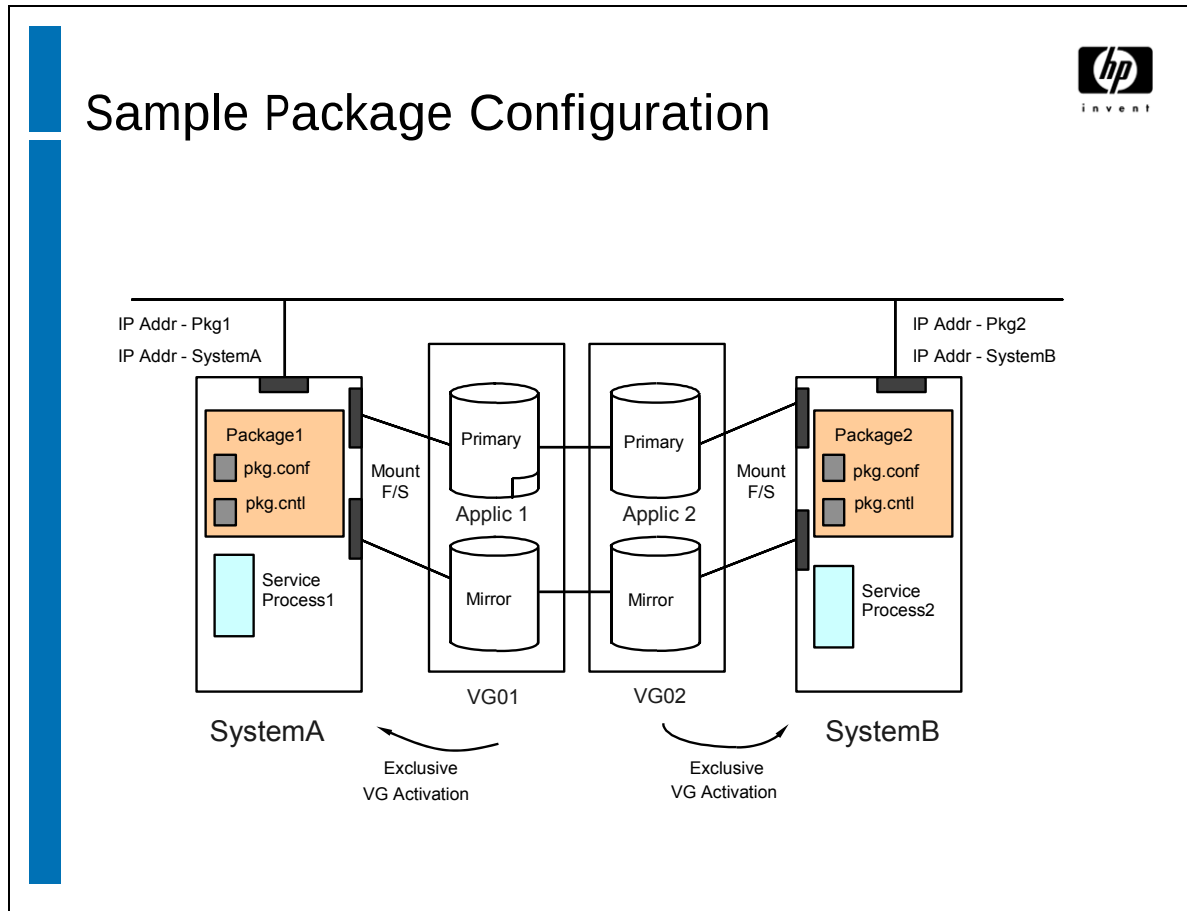
- **package configuration file** and
- **package control script**.

One configuration file and one control script will need to exist for each package.

The package configuration file defines the dependencies for the application, like a subnet or a service process. Also defined in the configuration file are package attributes and characteristics.

The package control script is executed to bring up (or bring down) the packaged application in a Serviceguard environment.

7-2. SLIDE: Sample Package Configuration



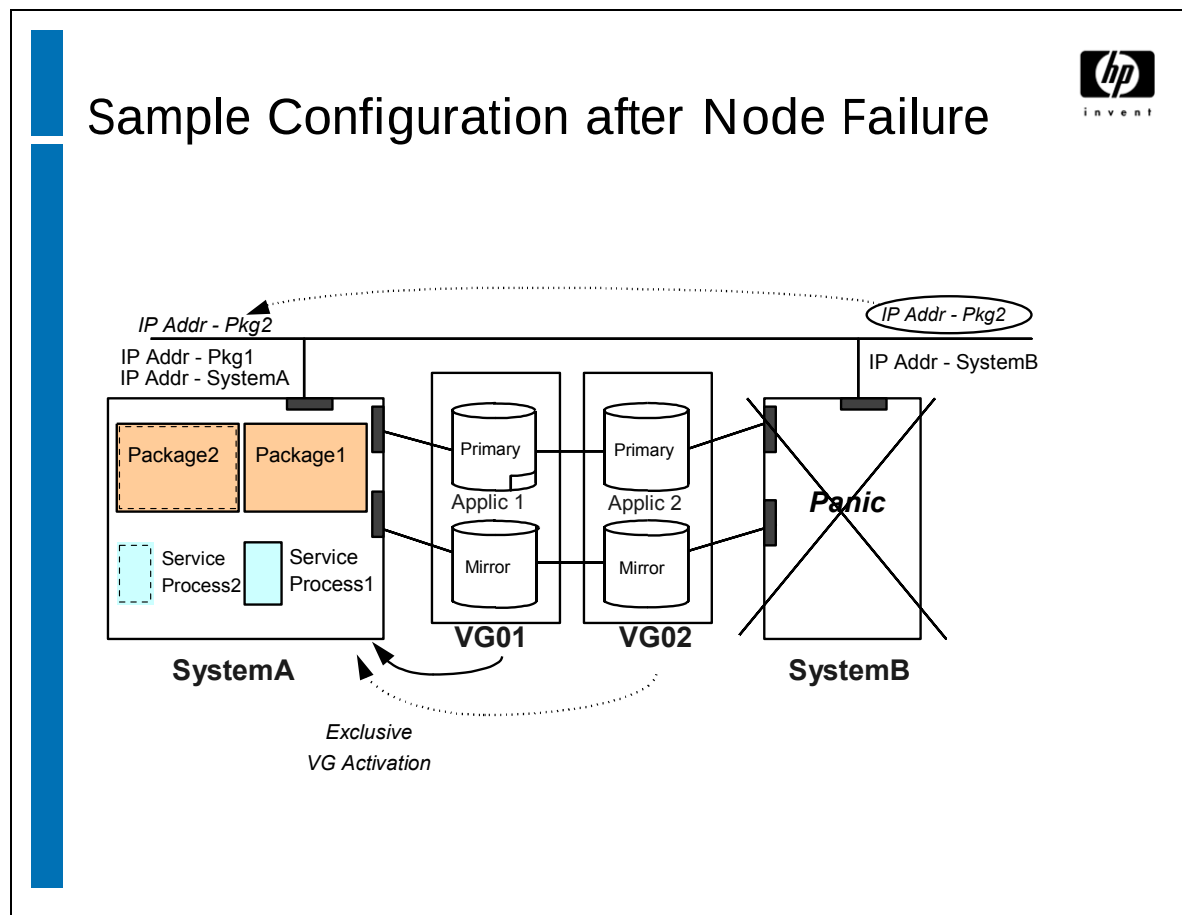
Student Notes

The slide shows a sample Serviceguard cluster configured with two packages.

Package1 was started on SystemA using the pkg.conf and the pkg.cntl files. Three things should be noted related to the execution of Package1 on SystemA:

- SystemA has volume group vg01 exclusively activated. This is because vg01 contains the application specific data for Package1.
- The LAN card on SystemA contains two IP addresses: one for the host SystemA and one for Package1.
- The service process for Package1 is currently running on SystemA. Should the service process terminate, Serviceguard will interpret that as an application failure and will try to restart the service process or move the application over to SystemB.

7-3. SLIDE: Sample Configuration after Node Failure



Student Notes

The above diagram illustrates what happens when SystemB fails (assume a CPU or memory board failure).

At `NODE_TIMEOUT` seconds, SystemA will detect that SystemB is no longer sending heartbeat messages. SystemA will respond by attempting to reform the cluster without SystemB. Since SystemA makes up exactly 50 percent of the cluster, it will need to get the cluster lock disk. This will not be a problem because SystemB is down and will not be competing for the cluster lock disk.

Once the cluster reforms, SystemA detects Package2 is not running in the cluster. If Package2 is configured to run on SystemA, SystemA runs the package script for Package2, which exclusively activates vg02 onto SystemA, assigns the Package2's IP address to the LAN card on SystemA, and starts the service process for Package2 on SystemA. SystemA is now running both packages.

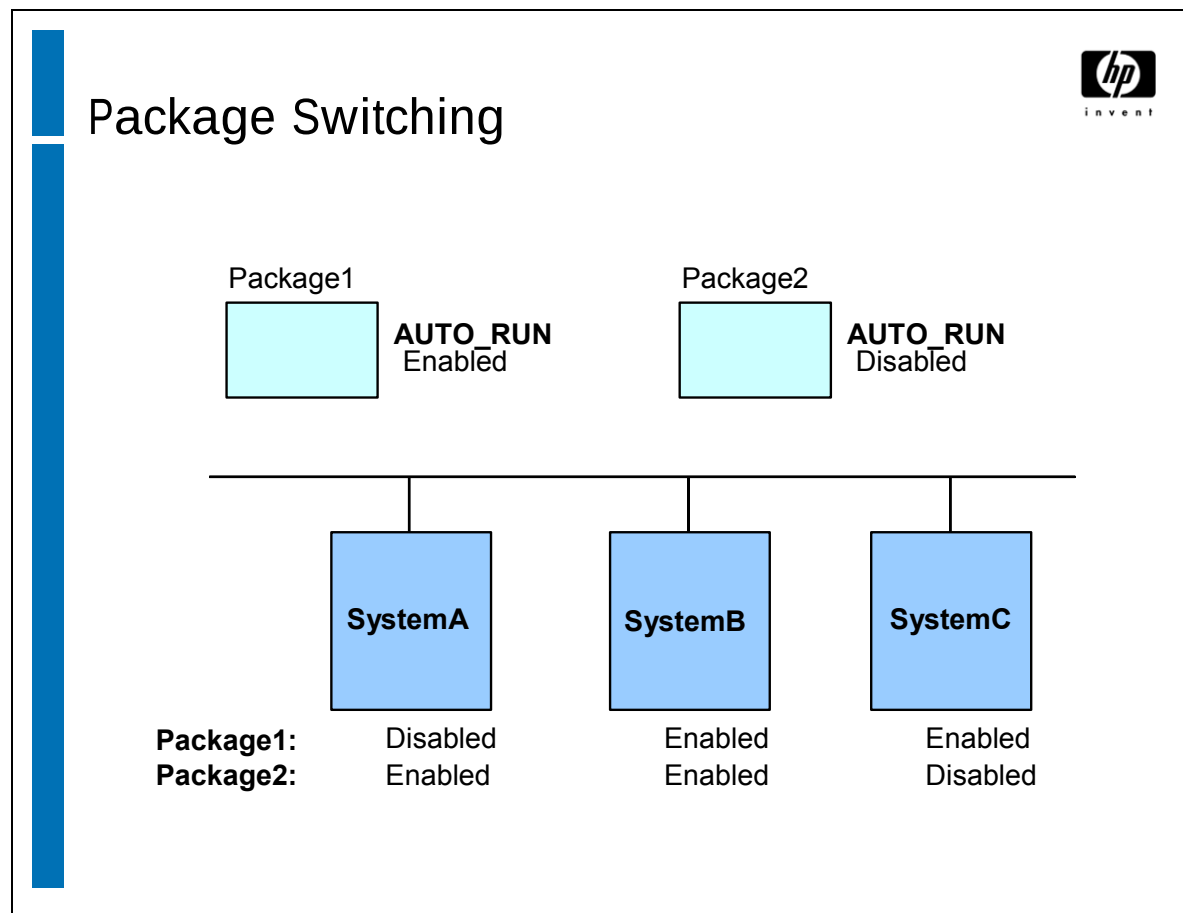
Once SystemB is fixed, it can be rebooted and the cluster will reform to be a 2-node cluster again with SystemB. However, Package2 will *not* automatically move back to SystemB during reformation. Moving the package back to SystemB would require the application be brought down (hence an application outage). Unless Package2 is configured to move back to SystemB

automatically, Serviceguard leaves it on SystemA. Package behavior is configurable on a per-package basis.

Package2 could be manually moved back by the system administrator by executing the following commands:

```
# cmhaltpkg -v Package2  
    and then  
# cmrunpkg -v -n SystemB Package2
```

7-4. SLIDE: Package Switching



Student Notes

Two parameters need to be enabled for a package to switch between nodes in a cluster: **AUTO_RUN** and **NODE_SWITCHING**.

Auto Run

Every package has an attribute called **AUTO_RUN**, which determines if a package can be run automatically among nodes in the cluster. If **AUTO_RUN** is disabled, the package will not move to other nodes, even if the node was enabled to receive the package.

Node Switching

Every node has an attribute per package called **SWITCHING**, which determines if that package can be received (switched to) by that node. If **SWITCHING** is disabled on a node for a particular package, that package will not move to that particular node, even though the package is enabled for package switching.

Example

There are two packages shown on the slide: Package1 and Package2. Package1 is capable of switching to other nodes within the cluster, but Package2 is not capable of switching.

The three nodes shown on the slide are selectively enabled / disabled for the various packages. Because Package2 is disabled, the fact that SystemB and SystemC can receive the package is irrelevant. Also, note that Package1 will not move to SystemA due to SystemA being disabled for Package1.

The only package movement that can take place is Package1 moving to SystemB or SystemC.

7-5. SLIDE: Viewing Package Status



Viewing Package Status

```
# cmviewcl -v
```

CLUSTER	STATUS			
cluster1	up			

NODE	STATUS	STATE		
SystemA	up	running		

Network_Parameters:

INTERFACE	STATUS	PATH	NAME
PRIMARY	up	0/1/2/0	lan1
STANDBY	up	0/2/1/0	lan2

PACKAGE	STATUS	STATE	AUTO_RUN	NODE
Package1	up	running	enabled	SystemA

Script_Parameters:

ITEM	STATUS	NAME	MAX_RESTARTS	RESTARTS
Service	up	service1.1	2	0
Subnet	up	10.10	0	0

Node_Switching_Parameters:

NODE_TYPE	STATUS	SWITCHING	NAME
Primary	up	enabled	SystemA
Alternate	up	disabled	SystemB

Student Notes

The status of packages and services can be seen with the `cmviewcl` command. Package related fields to monitor in this output include the following:

- **AUTO_RUN** specifies whether a package can switch to other nodes should the package fail.
- **SWITCHING** specifies which nodes on a per package basis are able to receive the package.
- **Script parameters** are resources in which the package is dependent upon in order to be considered healthy. These resources are configured in the package's run script.
- The above screen shot shows a cluster configured with LVM disks providing cluster lock services. Had this cluster been configured using a Quorum Server, then the Quorum Server would also be seen if the -v (verbose) option was used with the `cmviewcl` command.

7-6. SLIDE: Modifying Package Status



Modifying Package Status

Enable Package for Switching To Any Node:

```
cmmodpkg -e pkg_name
```

Enable a Specific Node to Receive a Package:

```
cmmodpkg -e -n nodename pkg_name
```

Student Notes

The `cmmodpkg` command performs two operations. It enables / disables a package's ability to switch to another node, and it enables / disables a particular node from running specific packages.

The slide shows the syntax for enabling (`-e` option) a package or node for switching. The Serviceguard Administrator also would also commonly use the `-d` option to disable a package or node for switching.

Therefore, the complete syntax for the `cmmodpkg` command actually takes two forms:

- **To enable or disable a package for switching:**

```
# cmmodpkg [ -v ] { -e | -d } pkg_name
```

- **To enable or disable a node to / from receiving a package :**

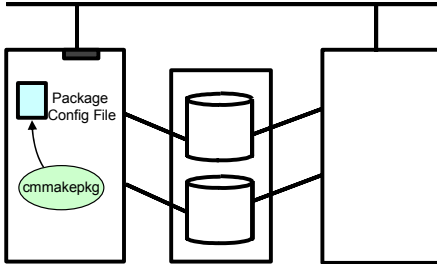
```
# cmmodpkg [ -v ] { -e | -d } -n node_name pkg_name
```

7-7. SLIDE: Step 1: Build Package Configuration File



Step 1: Build Package Configuration File

```
PACKAGE_NAME  
FAILOVER_POLICY  
FAILBACK_POLICY  
NODE_NAME          original_node  
NODE_NAME          adoptive_node  
AUTO_RUN  
NODE_FAIL_FAST_ENABLED  
RUN_SCRIPT  
RUN_SCRIPT_TIMEOUT  
HALT_SCRIPT  
HALT_SCRIPT_TIMEOUT  
SERVICE_NAME  
SERVICE_FAIL_FAST_ENABLED  
SERVICE_HALT_TIMEOUT  
SUBNET  
RESOURCE_NAME  
RESOURCE_POLLING_INTERVAL  
RESOURCE_START  
RESOURCE_UP_VALUE
```



Step-by-Step Procedure to Package an Application

1. Build the Package Configuration File
2. Build the Package Run/Halt Script
3. Compile and Distribute the Binary File

Student Notes

There are three main steps for packaging an application for a Serviceguard environment. The three main steps are:

1. Build the package configuration file.
2. Build the package run/halt script.
3. Compile and distribute the binary file.

Step 1: Build the Package Configuration File

The first step in building the package configuration file is to run the `cmmakepkg` command to create the package configuration template:

```
# cmmakepkg -v -p package.conf
```

The next step is to edit / customize the file with the `vi` editor:

```
# vi package.conf
```

The following are some of the more important fields in the package configuration file

<i>PACKAGE_NAME</i>	The name of the package. The package name must be unique to the cluster. PACKAGE_NAME can be as long as 39 characters.
<i>PACKAGE_TYPE</i>	This indicates whether this package is to run as a FAILOVER or SYSTEM_MULTI_NODE package. FAILOVER Package runs on one node at a time and if a failure occurs it can switch to an alternate node. SYSTEM_MULTI_NODE Package runs on multiple nodes at the same time. It cannot be started and halted on individual nodes. The only package that runs with this attribute is Cluster Volume Manager (CVM). Further, both NODE_FAIL_FAST_ENABLED and AUTO_RUN must be set to YES for this type of package. All SERVICES must have SERVICE_FAIL_FAST_ENABLED set to YES.
<i>FAILOVER_POLICY</i>	The failover policy determines how Serviceguard selects a failover node when the package fails.
<i>FAILBACK_POLICY</i>	The failback policy determines whether Serviceguard returns the package back to its original node when the node is available again.
<i>NODE_NAME</i>	The names of the primary node and adoptive nodes for the package. The order is important. The first listed will be the primary node. The second listed will be the first adoptive. This would be followed by additional adoptive node names in the order of failover.
<i>AUTO_RUN</i>	The AUTO_RUN parameter determines whether the package will automatically be started when the cluster is started. AUTO_RUN replaces PKG_SWITCHING_ENABLED, which was obsoleted in MC/ServiceGuard, version 11.12.
<i>LOCAL_LAN_FAILOVER_ALLOWED</i>	The LOCAL_LAN_FAILOVER_ALLOWED parameter determines if the cluster software will be allowed to switch LANs locally (i.e. transfer to a standby LAN card) in the event of a primary LAN card failure.
<i>NODE_FAIL_FAST_ENABLED</i>	When set to YES (NO is the default), this parameter will cause the node to TOC panic if the package halt script fails or if the node loses contact with its subnet.

<i>SCRIPT_LOG_FILE</i>	The default filename and location for the package logfile is /etc/cmc/cluster/pkgname/pkg_control_script.log. If you wish to rename this file and/or change the location, you may uncomment this line and specify an absolute path and filename to the alternate location.
<i>RUN_SCRIPT</i>	The absolute path name of the package run script. It is recommended this be the same file as the <i>HALT_SCRIPT</i> .
<i>RUN_SCRIPT_TIMEOUT</i>	The amount of time to allow the <i>RUN_SCRIPT</i> to complete. If the script does not complete within the time out value, then the package is halted and disabled.
<i>HALT_SCRIPT</i>	The full path name of the package halt script. It is recommended this be the same file as the <i>RUN_SCRIPT</i> .
<i>HALT_SCRIPT_TIMEOUT</i>	The amount of time, in seconds, to allow the <i>HALT_SCRIPT</i> to complete. If the script does not complete within the time out value, then the package is halted and disabled.
<i>STORAGE_GROUP</i>	Reserved for use with CVM.
<i>SERVICE_NAME</i>	The Serviceguard name of a service. A <i>SERVICE_NAME</i> should be unique, single words, without quotes. <i>SERVICE_NAME</i> is the link between the package configuration file and the package control (run / halt) script, and must be the same in both files.
<i>SERVICE_FAIL_FAST_ENABLED</i>	This parameter indicates whether the failure of the service should result in a TOC panic. If set to YES, a TOC panic will occur if this service fails. The default is NO.
<i>SERVICE_HALT_TIMEOUT</i>	The amount of time, in seconds, to allow the service process to be terminated with the SIGTERM signal (-15). If the service does not terminate within the time out value, the SIGKILL signal (-9) will be sent to the service process.
<i>SUBNET</i>	The health of the package depends on this subnet address. If Serviceguard can't access the subnet, the package will fail. With <i>NODE_FAILFAST_ENABLED</i> set to YES, the node will TOC panic. Regardless of whether or not the package will be assigned an IP address, the <i>SUBNET</i> line must have an entry.
<i>RESOURCE_NAME</i>	This line is the fully qualified resource path name.
<i>RESOURCE_POLLING_INTERVAL</i>	This is an optional field. If missing, the default value is 60, and indicates how often, in seconds, the resource is to be monitored.

<i>RESOURCE_START_VALUE</i>	This option can be either <code>AUTOMATIC</code> or <code>DEFERRED</code> . The default is <code>AUTOMATIC</code> , and means that Serviceguard will start resource monitoring automatically when the node starts up. If <code>DEFERRED</code> is selected, Serviceguard will not attempt to start up resource monitoring when the node starts up. Rather, the user is expected to specify all the <code>DEFERRED</code> resources in the package run script. In this way, the <code>DEFERRED</code> resources will be started up from the package run script during package run time.
<i>RESOURCE_UP_VALUE</i>	We must always have one or more of these <code>RESOURCE_UP_VALUE</code> lines. Specifics options are documented in the package configuration file itself.
Access Control Policy Parameters	By default, Serviceguard configuration can only be performed by the system administrator. Access control policies allow the root user to delegate certain tasks to non-root users. The package configuration file has entries for <code>USER_NAME</code> , <code>USER_HOST</code> , and <code>USER_ROLE</code> . <code>USER_NAME</code> is the user to whom certain access will be granted. <code>USER_HOST</code> is where <code>USER_NAME</code> can issue Serviceguard commands (not necessarily a node in the cluster). <code>USER_ROLE</code> defines <code>USER_NAME</code> 's privileges and must be <code>PACKAGE_ADMIN</code> .

7-8. TEXT PAGE: Sample Package Configuration File

HIGH AVAILABILITY PACKAGE CONFIGURATION FILE (template)

This is an example of a configuration file.

NOTE: This file **MUST** be edited before it can be used. For complete details about package parameters and how to set them, consult the Serviceguard, Serviceguard OPS edition, or Serviceguard SGeRAC man pages or manuals.

Special Note

The following package configuration file is from version 11.17 of Serviceguard.

```
# *****  
# ***** HIGH AVAILABILITY PACKAGE CONFIGURATION FILE (template) *****  
# *****  
# ***** Note: This file MUST be edited before it can be used. *****  
# * For complete details about package parameters and how to set them, *  
# * consult the Serviceguard Extension for RAC manuals.  
# *****
```

```
# Enter a name for this package. This name will be used to identify the  
# package when viewing or manipulating it. It must be different from  
# the other configured package names.
```

PACKAGE_NAME

```
# Enter the package type for this package. PACKAGE_TYPE indicates  
# whether this package is to run as a FAILOVER, MULTI_NODE, or  
# SYSTEM_MULTI_NODE package.  
#  
#     FAILOVER      package runs on one node at a time and if a failure  
#                   occurs it can switch to an alternate node.  
#  
#     MULTI_NODE    package runs on multiple nodes at the same time and  
#                   can be independently started and halted on  
#                   individual nodes. Failures of package components such  
#                   as services, EMS resources or subnets, will cause  
#                   the package to be halted only on the node on which the  
#                   failure occurred. Relocatable IP addresses cannot be  
#                   assigned to MULTI_NODE packages.  
#  
#     SYSTEM_MULTI_NODE  
#                   package runs on all cluster nodes at the same time.  
#                   It can not be started and halted on individual nodes.  
#                   Both NODE_FAIL_FAST_ENABLED and AUTO_RUN must be set  
#                   to YES for this type of package. All SERVICES must  
#                   have SERVICE_FAIL_FAST_ENABLED set to YES.  
#  
# NOTE: Packages which have a PACKAGE_TYPE of MULTI_NODE and  
#       SYSTEM_MULTI_NODE are not failover packages and are only
```

```
# supported for use by applications provided by Hewlett-Packard.
#
# Since MULTI_NODE and SYSTEM_MULTI_NODE packages can run on more
# than one node at a time and do not failover in the event of a
# package failure, the following parameters cannot be
# specified when configuring packages of these types:
#
#     FAILOVER_POLICY
#     FAILBACK_POLICY
#
# Since an IP address can not be assigned to more than node at a
# time, relocatable IP addresses can not be assigned in the
# package control script for MULTI_NODE packages. If volume
# groups are used in a MULTI_NODE package, they must be
# activated in a shared mode and data integrity is left to the
# application. Shared access requires a shared volume manager.
#
# Examples : PACKAGE_TYPE    FAILOVER (default)
#             PACKAGE_TYPE    MULTI_NODE
#             PACKAGE_TYPE    SYSTEM_MULTI_NODE
#
#
```

PACKAGE_TYPE	FAILOVER
--------------	----------

```
# Enter the failover policy for this package. This policy will be used
# to select an adoptive node whenever the package needs to be started
# The default policy unless otherwise specified is CONFIGURED_NODE.
# This policy will select nodes in priority order from the list of
# NODE_NAME entries specified below.
#
# The alternative policy is MIN_PACKAGE_NODE. This policy will select
# the node, from the list of NODE_NAME entries below, which is
# running the least number of packages at the time this package needs
# to start.
```

FAILOVER_POLICY	CONFIGURED_NODE
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

```
# Enter the fallback policy for this package. This policy will be used
# to determine what action to take when a package is not running on
# its primary node and its primary node is capable of running the
# package. The default policy unless otherwise specified is MANUAL.
# The MANUAL policy means no attempt will be made to move the package
# back to its primary node when it is running on an adoptive node.
#
# The alternative policy is AUTOMATIC. This policy will attempt to
# move the package back to its primary node whenever the primary node
# is capable of running the package.
```

FAILBACK_POLICY **MANUAL**

```
# Enter the names of the nodes configured for this package.  Repeat
# this line as necessary for additional adoptive nodes.
#
# NOTE:   The order is relevant.
#         Put the second Adoptive Node after the first one.
#
# Example : NODE_NAME  original_node
#           NODE_NAME  adoptive_node
#
```

Module 7

Packages and Services

```
# If all nodes in the cluster are to be specified and order is not
# important, "NODE_NAME *" may be specified.
#
# Example : NODE_NAME *
```

NODE_NAME

```
# Enter the value for AUTO_RUN. Possible values are YES and NO.
# The default for AUTO_RUN is YES. When the cluster is started the
# package will be automatically started. In the event of a failure the
# package will be started on an adoptive node. Adjust as necessary.
#
# AUTO_RUN replaces obsolete PKG_SWITCHING_ENABLED.
```

AUTO_RUN **YES**

```
# Enter the value for LOCAL_LAN_FAILOVER_ALLOWED.
# Possible values are YES and NO.
# The default for LOCAL_LAN_FAILOVER_ALLOWED is YES. In the event of a
# failure, this permits the cluster software to switch LANs locally
# (transfer to a standby LAN card). Adjust as necessary.
#
# LOCAL_LAN_FAILOVER_ALLOWED replaces obsolete NET_SWITCHING_ENABLED.
```

LOCAL_LAN_FAILOVER_ALLOWED **YES**

```
# Enter the value for NODE_FAIL_FAST_ENABLED.
# Possible values are YES and NO.
# The default for NODE_FAIL_FAST_ENABLED is NO. If set to YES,
# in the event of a failure, the cluster software will halt the node
# on which the package is running. All SYSTEM_MULTI_NODE packages must have
# NODE_FAIL_FAST_ENABLED set to YES. Adjust as necessary.
```

NODE_FAIL_FAST_ENABLED **NO**

```
# Enter the complete path for the run and halt scripts. In most cases
# the run script and halt script specified here will be the same script,
# the package control script generated by the cmmakepkg command. This
# control script handles the run(ning) and halt(ing) of the package.
#
# Enter the timeout, specified in seconds, for the run and halt scripts.
# If the script has not completed by the specified timeout value,
# it will be terminated. The default for each script timeout is
# NO_TIMEOUT. Adjust the timeouts as necessary to permit full
# execution of each script.
#
# Note: The HALT_SCRIPT_TIMEOUT should be greater than the sum of
# all SERVICE_HALT_TIMEOUT values specified for all services.
#
# The file where the output of the scripts is logged can be specified
# via the SCRIPT_LOG_FILE parameter. If not set, script output is sent
# to a file named by appending '.log' to the script path.
#
#SCRIPT_LOG_FILE
```

RUN_SCRIPT
RUN_SCRIPT_TIMEOUT **NO_TIMEOUT**
HALT_SCRIPT
HALT_SCRIPT_TIMEOUT **NO_TIMEOUT**

```
# Enter the names of the storage groups configured for this package.
# Repeat this line as necessary for additional storage groups.
#
# Storage groups are only used with CVM disk groups. Neither
# VxVM disk groups or LVM volume groups should be listed here.
# By specifying a CVM disk group with the STORAGE_GROUP keyword
# this package will not run until the CVM system multi node package is
# running and thus the CVM shared disk groups are ready for
# activation.
#
# NOTE: Should only be used by applications provided by
#       Hewlett-Packard.
#
# Example : STORAGE_GROUP  dg01
#           STORAGE_GROUP  dg02
#           STORAGE_GROUP  dg03
#           STORAGE_GROUP  dg04
#

# Enter the names of the dependency condition for this package.
# Dependencies are used to describe the relationship between packages
# To define a dependency, all three attributes are required.
#
# DEPENDENCY_NAME must have a unique identifier for the dependency.
#
# DEPENDENCY_CONDITION
#       This is an expression describing what must be true for
#       the dependency to be satisfied.
#
#       The syntax is: <package name> = UP , where <package name>
#       is the name of a multi-node or system multi-node package.
#
# DEPENDENCY_LOCATION
#       This describes where the condition must be satisfied.
#       The only possible value for this attribute is SAME_NODE
#
# NOTE:
# Dependencies can only be defined for packages within a CFS cluster. These are
# automatically setup in the SYSTEM-MULTI-NODE and MULTI-NODE packages created for
# disk groups and mount points. Customers configure dependencies for FAILOVER type
# packages only; and the dependency would be on a MULTI-NODE mount point (MP) package.
#
# Example :
#           DEPENDENCY_NAME      SG-CFS-MP-1
#           DEPENDENCY_CONDITION SG-CFS-MP-1=UP
#           DEPENDENCY_LOCATION  SAME_NODE
#
#DEPENDENCY_NAME
#DEPENDENCY_CONDITION
#DEPENDENCY_LOCATION SAME_NODE
#

# Enter the SERVICE_NAME, the SERVICE_FAIL_FAST_ENABLED and the
# SERVICE_HALT_TIMEOUT values for this package. Repeat these
# three lines as necessary for additional service names. All
# service names MUST correspond to the SERVICE_NAME[] entries in
# the package control script.
#
# The value for SERVICE_FAIL_FAST_ENABLED can be either YES or
# NO. If set to YES, in the event of a service failure, the
```

Module 7

Packages and Services

```
# cluster software will halt the node on which the service is
# running.  If SERVICE_FAIL_FAST_ENABLED is not specified, the
# default will be NO.
#
# SERVICE_HALT_TIMEOUT is represented as a number of seconds.
# This timeout is used to determine the length of time (in
# seconds) the cluster software will wait for the service to
# halt before a SIGKILL signal is sent to force the termination
# of the service.  In the event of a service halt, the cluster
# software will first send a SIGTERM signal to terminate the
# service.  If the service does not halt, after waiting for the
# specified SERVICE_HALT_TIMEOUT, the cluster software will send
# out the SIGKILL signal to the service to force its termination.
# This timeout value should be large enough to allow all cleanup
# processes associated with the service to complete.  If the
# SERVICE_HALT_TIMEOUT is not specified, a zero timeout will be
# assumed, meaning the cluster software will not wait at all
# before sending the SIGKILL signal to halt the service.
#
# Example: SERVICE_NAME          DB_SERVICE
#          SERVICE_FAIL_FAST_ENABLED    NO
#          SERVICE_HALT_TIMEOUT        300
#
# To configure a service, uncomment the following lines and
# fill in the values for all of the keywords.
#
#SERVICE_NAME          <service name>
#SERVICE_FAIL_FAST_ENABLED    <YES/NO>
#SERVICE_HALT_TIMEOUT    <number of seconds>

# Enter the network subnet name that is to be monitored for this package.
# Repeat this line as necessary for additional subnet names.  If any of
# the subnets defined goes down, the package will be switched to another
# node that is configured for this package and has all the defined subnets
# available.
# The subnet names could be IPv4 or IPv6. The network subnet
# names that are to be monitored for this package could be a mix
# of IPv4 or IPv6 subnet names

#SUBNET

# The keywords RESOURCE_NAME, RESOURCE_POLLING_INTERVAL,
# RESOURCE_START, and RESOURCE_UP_VALUE are used to specify Package
# Resource Dependencies.  To define a package Resource Dependency, a
# RESOURCE_NAME line with a fully qualified resource path name, and
# one or more RESOURCE_UP_VALUE lines are required.  The
# RESOURCE_POLLING_INTERVAL and the RESOURCE_START are optional.
#
# The RESOURCE_POLLING_INTERVAL indicates how often, in seconds, the
# resource is to be monitored.  It will be defaulted to 60 seconds if
# RESOURCE_POLLING_INTERVAL is not specified.
#
# The RESOURCE_START option can be set to either AUTOMATIC or DEFERRED.
# The default setting for RESOURCE_START is AUTOMATIC.  If AUTOMATIC
# is specified, Serviceguard will start up resource monitoring for
# these AUTOMATIC resources automatically when the node starts up.
# If DEFERRED is selected, Serviceguard will not attempt to start
# resource monitoring for these resources during node start up.  User
# should specify all the DEFERRED resources in the package run script
# so that these DEFERRED resources will be started up from the package
# run script during package run time.
```

```
#
# RESOURCE_UP_VALUE requires an operator and a value. This defines
# the resource 'UP' condition. The operators are =, !=, >, <, >=,
# and <=, depending on the type of value. Values can be string or
# numeric. If the type is string, then only = and != are valid
# operators. If the string contains whitespace, it must be enclosed
# in quotes. String values are case sensitive. For example,
#
#                                     Resource is up when its value is
#                                     -----
#
#      RESOURCE_UP_VALUE      = UP              "UP"
#      RESOURCE_UP_VALUE      != DOWN           Any value except "DOWN"
#      RESOURCE_UP_VALUE      = "On Course"     "On Course"
#
# If the type is numeric, then it can specify a threshold, or a range to
# define a resource up condition. If it is a threshold, then any operator
# may be used. If a range is to be specified, then only > or >= may be used
# for the first operator, and only < or <= may be used for the second operator.
# For example,
#
#                                     Resource is up when its value is
#                                     -----
#
#      RESOURCE_UP_VALUE      = 5                5                (threshold)
#      RESOURCE_UP_VALUE      > 5.1              greater than 5.1  (threshold)
#      RESOURCE_UP_VALUE      > -5 and < 10       between -5 and 10 (range)
#
# Note that "and" is required between the lower limit and upper limit
# when specifying a range. The upper limit must be greater than the lower
# limit. If RESOURCE_UP_VALUE is repeated within a RESOURCE_NAME block, then
# they are inclusively OR'd together. Package Resource Dependencies may be
# defined by repeating the entire RESOURCE_NAME block.
#
# Example : RESOURCE_NAME          /net/interfaces/lan/status/lan0
#           RESOURCE_POLLING_INTERVAL 120
#           RESOURCE_START            AUTOMATIC
#           RESOURCE_UP_VALUE         = RUNNING
#           RESOURCE_UP_VALUE         = ONLINE
#
#           Means that the value of resource /net/interfaces/lan/status/lan0
#           will be checked every 120 seconds, and is considered to
#           be 'up' when its value is "RUNNING" or "ONLINE".
#
# Uncomment the following lines to specify Package Resource Dependencies.
#
#RESOURCE_NAME          <Full_path_name>
#RESOURCE_POLLING_INTERVAL <numeric_seconds>
#RESOURCE_START         <AUTOMATIC/DEFERRED>
#RESOURCE_UP_VALUE      <op> <string_or_numeric> [and <op> <numeric>]
#
# Access Control Policy Parameters.
#
# Three entries set the access control policy for the package:
# First line must be USER_NAME, second USER_HOST, and third USER_ROLE.
# Enter a value after each.
#
# 1. USER_NAME can either be ANY_USER, or a maximum of
#    8 login names from the /etc/passwd file on user host.
# 2. USER_HOST is where the user can issue Serviceguard commands.
#    If using Serviceguard Manager, it is the COM server.
#    Choose one of these three values: ANY_SERVICEGUARD_NODE, or
#    (any) CLUSTER_MEMBER_NODE, or a specific node. For node,
#    use the official hostname from domain name server, and not
#    an IP addresses or fully qualified name.
# 3. USER_ROLE must be PACKAGE_ADMIN. This role grants permission
```

Module 7

Packages and Services

```
#    to MONITOR, plus for administrative commands for the package.
#
# These policies do not effect root users. Access Policies here
# should not conflict with policies defined in the cluster configuration file.
#
# Example: to configure a role for user john from node noir to
# administer the package, enter:
# USER_NAME  john
# USER_HOST  noir
# USER_ROLE  PACKAGE_ADMIN
```


7-9. SLIDE: Step 2: Build Package Run/Halt Script



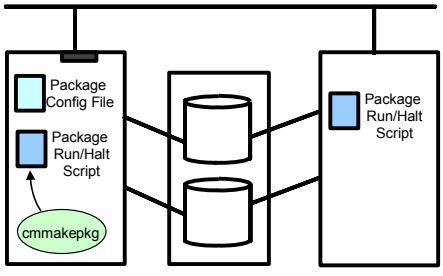
Step 2: Build Package Control Script

```
PATH=
VG[0]=
LV[0]=" "; FS[0]=" "; FS_MOUNT_OPT[0]="

IP[0]="
SUBNET[0]="

SERVICE_NAME[0]=
SERVICE_CMD[0]=" "
SERVICE_RESTART[0]=" "

function customer_defined_run_cmds
{
}
function customer_defined_halt_cmds
{
}
```



Step-by-Step Procedure to Package an Application

1. Build the Package Configuration File
2. **Build the Package Run/Halt Script**
3. Compile and Distribute the Binary File

Student Notes

Once the package configuration file is created, the package control script (also known as the (run / halt script)) needs to be created.

Step 2: Build the Package Run/Halt Script

The procedure for building the package control script is to create a template with the command:

```
# cmmakepkg -v -s package_script_name
```

The next step is to edit/customize the script with the vi editor:

```
# vi package_script_name
```

The following are some of the major fields in the package control script:

<i>PATH</i>	The value of the PATH variable used when executing the control script.
<i>VGCHANGE</i>	The name of the volume group activated exclusively by the control script.
<i>CVM_ACTIVATION_CMD</i>	This is reserved for future use with CVM.
<i>VG</i>	The name of the volume group activated exclusively by the control script.
<i>CVM_DG</i>	Cluster Volume Manager Disk Groups.
<i>VXVM_DG</i>	The name of the VxVM Disk Group used by the package.
<i>LV</i>	The name of the logical volume mounted by the control script.
<i>FS</i>	The name of the file system mount point used when mounting the logical volume.
<i>FS_MOUNT_OPT</i>	The list of mount options to be used when mounting the logical volume on the file system directory (mount point).
<i>FS_UMOUNT_COUNT</i>	The number of unmount attempts for each file system.
<i>FS_MOUNT_RETRY_COUNT</i>	The number of mount retries for each file system.
<i>IP</i>	The IP address of the package. The IP and SUBNET lines are used in pairs. If the package will not be assigned an IP, there is no need to specify the SUBNET
<i>SUBNET</i>	The subnet of the package. This determines which subnet and interface the package will use to transfer data. (See the description for IP for further information on this parameter)
<i>SERVICE_NAME</i>	The name of the service process that is started. This is the link between the package configuration and control scripts. It must be exactly the same in both files. The only exception is that quotes are okay in the control script but not in the configuration script.
<i>SERVICE_CMD</i>	The full pathname of the command used to start the service process.
<i>SERVICE_RESTART</i>	The number of times to restart the service process if it fails. Valid configurations are:

- no restarts

`SERVICE_RESTART[0]=` or

`SERVICE_RESTART[0]="-r 0"`

- restart n-times on the same node

`SERVICE_RESTART[0]="-r n"` (n = number ≥ 0)

- always restart on the same node

`SERVICE_RESTART[0]="-R"`

customer_defined_run_cmds

Application-specific commands executed when starting the package. These commands run **before** services are started.

customer_defined_halt_cmds

Application-specific commands executed when halting the package. These commands are run **after** services are halted.

7-10. TEXT PAGE: Sample Control Script File

Special Note

**The following package control script is from version 11.17 of Serviceguard.
To conserve space in this course book, most of the functions in the second
half of the script have had the code within the function removed.**

```
# @(#) A.11.17.00 Date: 09/23/05  $
# *****
# *
# *          HIGH AVAILABILITY PACKAGE CONTROL SCRIPT (template)
# *
# *          Note: This file MUST be edited before it can be used.
# *
# *****

# The environment variables PACKAGE, NODE, SG_PACKAGE,
# SG_NODE and SG_SCRIPT_LOG_FILE are set by Serviceguard
# at the time the control script is executed.
# Do not set these environment variables yourself!
# The package may fail to start or halt if the values for
# these environment variables are altered.

# NOTE: Starting from 11.17, all environment variables set by
# Serviceguard implicitly at the time the control script is
# executed will contain the prefix "SG_". Do not set any variable
# with the defined prefix, or the control script may not
# function as it should.

. ${SGCONFFILE:=/etc/cmcluster.conf}

# UNCOMMENT the variables as you set them.

# Set PATH to reference the appropriate directories.
PATH=$SGSBIN:/usr/bin:/usr/sbin:/etc:/bin

# VOLUME GROUP ACTIVATION:
# Specify the method of activation for volume groups.
# Leave the default ("VGCHANGE="vgchange -a e") if you want volume
# groups activated in exclusive mode. This assumes the volume groups have
# been initialized with 'vgchange -c y' at the time of creation.
#
# Uncomment the first line (VGCHANGE="vgchange -a e -q n"), and comment
# out the default, if your disks are mirrored on separate physical paths,
#
# Uncomment the second line (VGCHANGE="vgchange -a e -q n -s"), and comment
# out the default, if your disks are mirrored on separate physical paths,
# and you want the mirror resynchronization to occur in parallel with
# the package startup.
#
# Uncomment the third line (VGCHANGE="vgchange -a y") if you wish to
# use non-exclusive activation mode. Single node cluster configurations
```

```
# must use non-exclusive activation.
#
# VGCHANGE="vgchange -a e -q n"
# VGCHANGE="vgchange -a e -q n -s"
# VGCHANGE="vgchange -a y"
VGCHANGE="vgchange -a e"                                # Default

# CVM DISK GROUP ACTIVATION:
# Specify the method of activation for CVM disk groups.
# Leave the default
# (CVM_ACTIVATION_CMD="vxdg -g \${DiskGroup} set activation=exclusivewrite")
# if you want disk groups activated in the exclusive write mode.
#
# Uncomment the first line
# (CVM_ACTIVATION_CMD="vxdg -g \${DiskGroup} set activation=readonly"),
# and comment out the default, if you want disk groups activated in
# the readonly mode.
#
# Uncomment the second line
# (CVM_ACTIVATION_CMD="vxdg -g \${DiskGroup} set activation=sharedread"),
# and comment out the default, if you want disk groups activated in the
# shared read mode.
#
# Uncomment the third line
# (CVM_ACTIVATION_CMD="vxdg -g \${DiskGroup} set activation=sharedwrite"),
# and comment out the default, if you want disk groups activated in the
# shared write mode.
#
# CVM_ACTIVATION_CMD="vxdg -g \${DiskGroup} set activation=readonly"
# CVM_ACTIVATION_CMD="vxdg -g \${DiskGroup} set activation=sharedread"
# CVM_ACTIVATION_CMD="vxdg -g \${DiskGroup} set activation=sharedwrite"
CVM_ACTIVATION_CMD="vxdg -g \${DiskGroup} set activation=exclusivewrite"

# VOLUME GROUPS
# Specify which volume groups are used by this package. Uncomment VG[0]="
# and fill in the name of your first volume group. You must begin with
# VG[0], and increment the list in sequence.
#
# For example, if this package uses your volume groups vg01 and vg02, enter:
#     VG[0]=vg01
#     VG[1]=vg02
#
# The volume group activation method is defined above. The filesystems
# associated with these volume groups are specified below.
#
#VG[0]="

# CVM DISK GROUPS
# Specify which cvm disk groups are used by this package. Uncomment
# CVM_DG[0]=" and fill in the name of your first disk group. You must
# begin with CVM_DG[0], and increment the list in sequence.
#
# For example, if this package uses your disk groups dg01 and dg02, enter:
#     CVM_DG[0]=dg01
#     CVM_DG[1]=dg02
#
# The cvm disk group activation method is defined above. The filesystems
```

```

# associated with these volume groups are specified below in the CVM_*
# variables.
#
#CVM_DG[0]=""/>
# NOTE: Do not use CVM and VxVM disk group parameters to reference
# devices used by CFS (cluster file system). CFS resources are
# controlled by the Disk Group and Mount Multi-node packages.
#
# VxVM DISK GROUPS
# Specify which VxVM disk groups are used by this package. Uncomment
# VXVM_DG[0]=""/> and fill in the name of your first disk group. You must
# begin with VXVM_DG[0], and increment the list in sequence.
#
# For example, if this package uses your disk groups dg01 and dg02, enter:
#     VXVM_DG[0]=dg01
#     VXVM_DG[1]=dg02
#
# The cvm disk group activation method is defined above.
#
#VXVM_DG[0]=""/>
#
# NOTE: A package could have LVM volume groups, CVM disk groups and VxVM
#       disk groups.
#
# NOTE: When VxVM is initialized it will store the hostname of the
#       local node in its volboot file in a variable called 'hostid'.
#       The Serviceguard package control scripts use both the values of
#       the hostname(1m) command and the VxVM hostid. As a result
#       the VxVM hostid should always match the value of the
#       hostname(1m) command.
#
#       If you modify the local host name after VxVM has been
#       initialized and such that hostname(1m) does not equal uname -n,
#       you need to use the vxdctl(1m) command to set the VxVM hostid
#       field to the value of hostname(1m). Failure to do so will
#       result in the package failing to start.
#
# VOLUME GROUP AND DISK GROUP DEACTIVATION RETRY COUNT
# Specify the number of deactivation retries for each disk group and volume
# group at package shutdown. The default is 0.
DEACTIVATION_RETRY_COUNT=0
#
# RAW DEVICES
# If you are using raw devices for your application, this parameter allows
# you to specify if you want to kill the processes that are accessing the
# raw devices at package halt time. If raw devices are still being accessed
# at package halt time, volume group or disk group deactivation can fail,
# causing the package halt to also fail. This problem usually happens when
# the application does not shut down properly.
# Note that if you are using Oracle's Cluster Ready Service, killing this
# service could cause the node to reboot.
# The legal values are "YES" and "NO". The default value is "NO".
# The value that is set for this parameter affects all raw devices associated
# with the LVM volume groups and CVM disk groups defined in the package.

```

```
KILL_PROCESSES_ACCESSING_RAW_DEVICES="NO"

# FILESYSTEMS
# Filesystems are defined as entries specifying the logical volume, the
# mount point, the mount, umount and fsck options and type of the file system.
# Each filesystem will be fsck'd prior to being mounted. The filesystems
# will be mounted in the order specified during package startup and will
# be unmounted in reverse order during package shutdown. Ensure that
# volume groups referenced by the logical volume definitions below are
# included in volume group definitions above.
#
# Specify the filesystems which are used by this package. Uncomment
# LV[0]=""; FS[0]=""; FS_MOUNT_OPT[0]=""; FS_UMOUNT_OPT[0]="";
# FS_FSCK_OPT[0]="";
# FS_TYPE[0]=" and fill in the name of your first logical volume,
# filesystem, mount, umount and fsck options and filesystem type
# for the file system. You must begin with LV[0], FS[0],
# FS_MOUNT_OPT[0], FS_UMOUNT_OPT[0], FS_FSCK_OPT[0], FS_TYPE[0]
# and increment the list in sequence.
#
# Note: The FS_TYPE parameter lets you specify the type of filesystem to be
# mounted. Specifying a particular FS_TYPE will improve package failover time.
# The FSCK_OPT and FS_UMOUNT_OPT parameters can be used to include the
# -s option with the fsck and umount commands to improve performance for
# environments that use a large number of filesystems. (An example of a
# large environment is given below following the description of the
# CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS parameter.)
#
# Example: If a package uses two JFS filesystems, pkg01a and pkg01b,
# which are mounted on LVM logical volumes lvol1 and lvol2 for read and
# write operation, you would enter the following:
#     LV[0]=/dev/vg01/lvol1; FS[0]=/pkg01a; FS_MOUNT_OPT[0]="-o rw";
#     FS_UMOUNT_OPT[0]=""; FS_FSCK_OPT[0]=""; FS_TYPE[0]="vxfs"
#
#     LV[1]=/dev/vg01/lvol2; FS[1]=/pkg01b; FS_MOUNT_OPT[1]="-o rw"
#     FS_UMOUNT_OPT[1]=""; FS_FSCK_OPT[1]=""; FS_TYPE[1]="vxfs"
#
#LV[0]=""; FS[0]=""; FS_MOUNT_OPT[0]=""; FS_UMOUNT_OPT[0]="";
#FS_FSCK_OPT[0]="";
#FS_TYPE[0]="";
#
# VOLUME RECOVERY
#
# When mirrored VxVM volumes are started during the package control
# bring up, if recovery is required the default behavior is for
# the package control script to wait until recovery has been
# completed.
#
# To allow mirror resynchronization to occur in parallel with
# the package startup, uncomment the line
# VxVOL="vxvol -g \${DiskGroup} -o bg startall" and comment out the default.
#
# VxVOL="vxvol -g \${DiskGroup} -o bg startall"
VxVOL="vxvol -g \${DiskGroup} startall"           # Default

# FILESYSTEM UNMOUNT COUNT
# Specify the number of unmount attempts for each filesystem during package
```

Module 7

Packages and Services

```
# shutdown. The default is set to 1.
FS_UMOUNT_COUNT=1

# FILESYSTEM MOUNT RETRY COUNT.
# Specify the number of mount retrys for each filesystem.
# The default is 0. During startup, if a mount point is busy
# and FS_MOUNT_RETRY_COUNT is 0, package startup will fail and
# the script will exit with 1. If a mount point is busy and
# FS_MOUNT_RETRY_COUNT is greater than 0, the script will attempt
# to kill the user responsible for the busy mount point
# and then mount the file system. It will attempt to kill user and
# retry mount, for the number of times specified in FS_MOUNT_RETRY_COUNT.
# If the mount still fails after this number of attempts, the script
# will exit with 1.
# NOTE: If the FS_MOUNT_RETRY_COUNT > 0, the script will execute
# "fuser -ku" to freeup busy mount point.
FS_MOUNT_RETRY_COUNT=0

#
# Configuring the concurrent operations below can be used to improve the
# performance for starting up or halting a package. The maximum value for
# each concurrent operation parameter is 1024. Set these values carefully.
# The performance could actually decrease if the values are set too high
# for the system resources available on your cluster nodes. Some examples
# of system resources that can affect the optimum number of concurrent
# operations are: number of CPUs, amount of available memory, the kernel
# configuration for nfile and nproc. In some cases, if you set the number
# of concurrent operations too high, the package may not be able to start
# or to halt. For example, if you set CONCURRENT_VGCHANGE_OPERATIONS=5
# and the node where the package is started has only one processor, then
# running concurrent volume group activations will not be beneficial.
# It is suggested that the number of concurrent operations be tuned
# carefully, increasing the values a little at a time and observing the
# effect on the performance, and the values should never be set to a value
# where the performance levels off or declines. Additionally, the values
# used should take into account the node with the least resources in the
# cluster, and how many other packages may be running on the node.
# For instance, if you tune the concurrent operations for a package so
# that it provides optimum performance for the package on a node while
# no other packages are running on that node, the package performance
# may be significantly reduced, or may even fail when other packages are
# already running on that node.
#
# CONCURRENT VGCHANGE OPERATIONS
# Specify the number of concurrent volume group activations or
# deactivations to allow during package startup or shutdown.
# Setting this value to an appropriate number may improve the performance
# while activating or deactivating a large number of volume groups in the
# package. If the specified value is less than 1, the script defaults it
# to 1 and proceeds with a warning message in the package control script
# logfile.
CONCURRENT_VGCHANGE_OPERATIONS=1

# CONCURRENT FSCK OPERATIONS
# Specify the number of concurrent fsck to allow during package startup.
# Setting this value to an appropriate number may improve the performance
# while checking a large number of file systems in the package. If the
```



```
# specified value is less than 1, the script defaults it to 1 and proceeds
# with a warning message in the package control script logfile.
CONCURRENT_FSCK_OPERATIONS=1

# CONCURRENT MOUNT AND UMOUNT OPERATIONS
# Specify the number of concurrent mounts and umounts to allow during
# package startup or shutdown.
# Setting this value to an appropriate number may improve the performance
# while mounting or un-mounting a large number of file systems in the package.
# If the specified value is less than 1, the script defaults it to 1 and
# proceeds with a warning message in the package control script logfile.
CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS=1

# Example: If a package uses 50 JFS filesystems, pkg01aa through pkg01bx,
# which are mounted on the 50 logical volumes lv01..lv050 for read and write
# operation, you may enter the following:
#
#     CONCURRENT_FSCK_OPERATIONS=50
#     CONCURRENT_MOUNT_AND_UMOUNT_OPERATIONS=50
#
#     LV[0]=/dev/vg01/lv01; FS[0]=/pkg01aa; FS_MOUNT_OPT[0]="-o rw";
#     FS_UMOUNT_OPT[0]="-s"; FS_FSCK_OPT[0]="-s"; FS_TYPE[0]="vxfs"
#
#     LV[1]=/dev/vg01/lv02; FS[1]=/pkg01ab; FS_MOUNT_OPT[1]="-o rw"
#     FS_UMOUNT_OPT[1]="-s"; FS_FSCK_OPT[1]="-s"; FS_TYPE[0]="vxfs"
#
#     :           :           :
#     :           :           :
#     :           :           :
#     LV[49]=/dev/vg01/lv050; FS[49]=/pkg01bx; FS_MOUNT_OPT[49]="-o rw"
#     FS_UMOUNT_OPT[49]="-s"; FS_FSCK_OPT[49]="-s"; FS_TYPE[0]="vxfs"
#
# IP ADDRESSES
# Specify the IP and Subnet address pairs which are used by this package.
# You could specify IPv4 or IPv6 IP and subnet address pairs.
# Uncomment IP[0]=" " and SUBNET[0]=" " and fill in the name of your first
# IP and subnet address. You must begin with IP[0] and SUBNET[0] and
# increment the list in sequence.
#
# For example, if this package uses an IP of 192.10.25.12 and a subnet of
# 192.10.25.0 enter:
#     IP[0]=192.10.25.12
#     SUBNET[0]=192.10.25.0
#     (netmask=255.255.255.0)
#
# Hint: Run "netstat -i" to see the available subnets in the Network field.
#
# For example, if this package uses an IPv6 IP of 2001::1/64
# The address prefix identifies the subnet as 2001::/64 which is an available
# subnet.
# enter:
#     IP[0]=2001::1
#     SUBNET[0]=2001::/64
#     (netmask=ffff:ffff:ffff:ffff::)
# Alternatively the IPv6 IP/Subnet pair can be specified without the prefix
# for the IPv6 subnet.
#     IP[0]=2001::1
#     SUBNET[0]=2001::
```

```

#           (netmask=ffff:ffff:ffff:ffff::)
#
# Hint: Run "netstat -i" to see the available IPv6 subnets by looking
# at the address prefixes
# IP/Subnet address pairs for each IP address you want to add to a subnet
# interface card. Must be set in pairs, even for IP addresses on the same
# subnet.
#
#IP[0]="
#SUBNET[0]="

# SERVICE NAMES AND COMMANDS.
# Specify the service name, command, and restart parameters which are
# used by this package. Uncomment SERVICE_NAME[0]="", SERVICE_CMD[0]="",
# SERVICE_RESTART[0]=" and fill in the name of the first service, command,
# and restart parameters. You must begin with SERVICE_NAME[0], SERVICE_CMD[0],
# and SERVICE_RESTART[0] and increment the list in sequence.
#
# For example:
#     SERVICE_NAME[0]=pkg1a
#     SERVICE_CMD[0]="/usr/bin/X11/xclock -display 192.10.25.54:0"
#     SERVICE_RESTART[0]=" # Will not restart the service.
#
#     SERVICE_NAME[1]=pkg1b
#     SERVICE_CMD[1]="/usr/bin/X11/xload -display 192.10.25.54:0"
#     SERVICE_RESTART[1]="-r 2" # Will restart the service twice.
#
#     SERVICE_NAME[2]=pkg1c
#     SERVICE_CMD[2]="/usr/sbin/ping"
#     SERVICE_RESTART[2]="-R" # Will restart the service an infinite
#                               number of times.
#
# Note: No environmental variables will be passed to the command, this
# includes the PATH variable. Absolute path names are required for the
# service command definition. Default shell is /usr/bin/sh.
#
#SERVICE_NAME[0]="
#SERVICE_CMD[0]="
#SERVICE_RESTART[0]="

# DEFERRED_RESOURCE NAME
# Specify the full path name of the 'DEFERRED' resources configured for
# this package. Uncomment DEFERRED_RESOURCE_NAME[0]=" and fill in the
# full path name of the resource.
#
#DEFERRED_RESOURCE_NAME[0]="

# DTC manager information for each DTC.
# Example: DTC[0]=dttc_20
#DTC_NAME[0]=

# HA_NFS_SCRIPT_EXTENSION
# If the package uses HA NFS, this variable can be used to alter the
# name of the HA NFS script. If not set, the name of this script is
# assumed to be "ha_nfs.sh". If set, the "sh" portion of the default
# script name is replaced by the value of this variable. So if
# HA_NFS_SCRIPT_EXTENSION is set to "package1.sh", for example, the name

```

```
# of the HA NFS script becomes "ha_nfs.package1.sh". In any case,
# the HA NFS script must be placed in the same directory as the package
# control script. This allows multiple packages to be run out of the
# same directory, as needed by SGeSAP.
#HA_NFS_SCRIPT_EXTENSION=""

# START OF CUSTOMER DEFINED FUNCTIONS

# This function is a place holder for customer define functions.
# You should define all actions you want to happen here, before the service is
# started. You can create as many functions as you need.

function customer_defined_run_cmds
{
# ADD customer defined run commands.
: # do nothing instruction, because a function must contain some command.

    test_return 51
}

# This function is a place holder for customer define functions.
# You should define all actions you want to happen here, after the service is
# halted.

function customer_defined_halt_cmds
{
# ADD customer defined halt commands.
: # do nothing instruction, because a function must contain some command.
    test_return 52
}

# END OF CUSTOMER DEFINED FUNCTIONS

# START OF RUN FUNCTIONS

function verify_physical_data_replication
{
}

function verify_ha_nfs
{
}

function ha_nfs_file_locks
{
}

function activate_volume_group
{
}

function activate_disk_group
```

Module 7
Packages and Services

```
{  
}  
  
function check_dg  
{  
}  
  
function freeup_busy_mountpoint_and_mount_fs  
{  
}  
  
function check_vxvm_vol_available  
{  
}  
  
function check_and_mount  
{  
}  
  
function retry_print  
{  
}  
  
function add_ip_address  
{  
}  
  
function get_ownership_dtc  
{  
}  
  
function start_services  
{  
}  
  
# END OF RUN FUNCTIONS.  
  
# START OF HALT FUNCTIONS  
  
function stop_resources  
{  
}
```

```
function halt_services
{
}

function disown_dtc
{
}

function remove_ip_address
{
}

function umount_fs
{
}

function deactivate_volume_group
{
}

function deactivate_disk_group
{
}

# END OF HALT FUNCTIONS.

# FUNCTIONS COMMON TO BOTH RUN AND HALT.

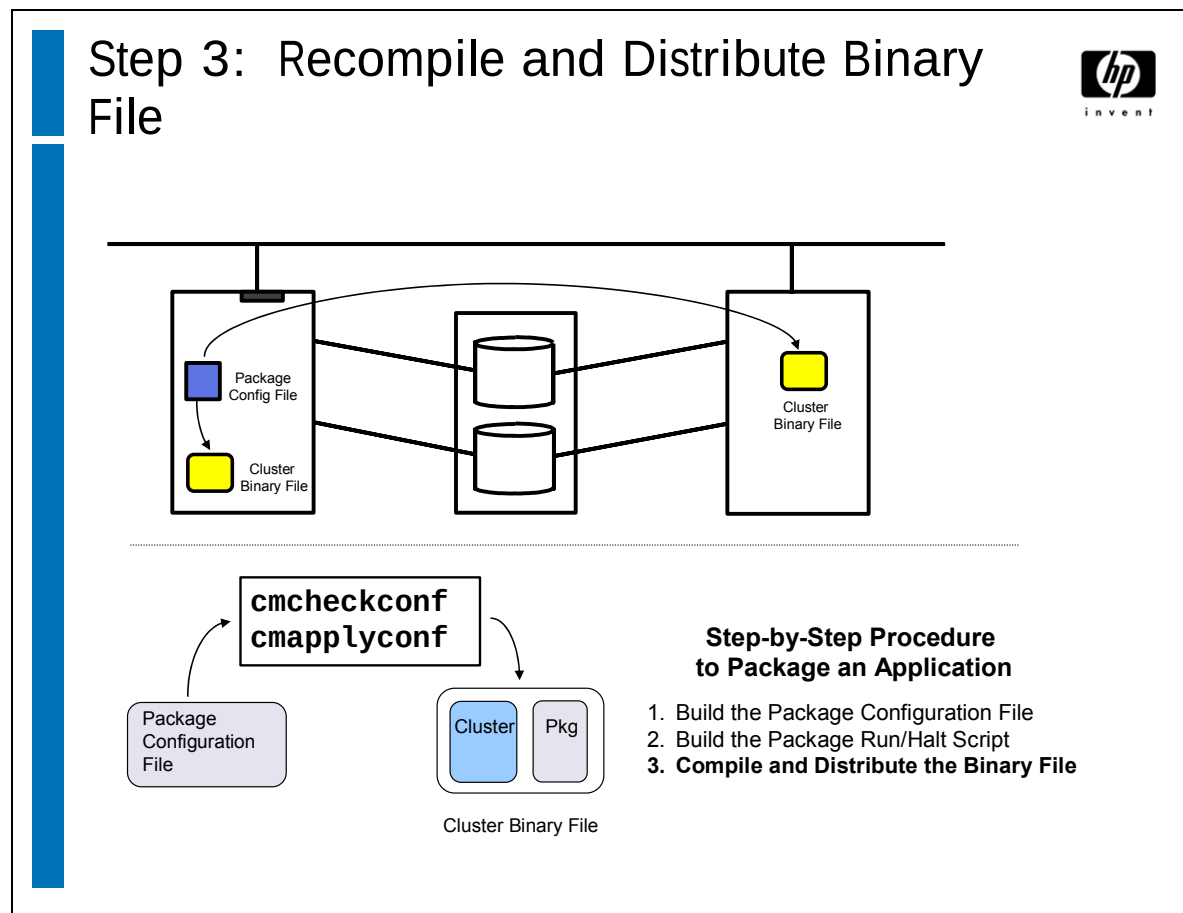
function test_return
{
}

# END OF FUNCTIONS COMMON TO BOTH RUN AND HALT

#-----MAINLINE Control Script Code Starts Here-----

#-----MAINLINE Control Script Code Ends Here-----
```

7-11. SLIDE: Step 3: Recompile and Distribute Binary File



Student Notes

The final step in the package creation process is to rebuild the cluster binary file to include the package information.

Step 3: Recompile and Distribute the Binary File

The commands to verify the package files and integrate the package information into the cluster binary file are:

```
# cmcheckconf -v -P package_configuration_file
```

```
# cmapplyconf -v -P package_configuration_file
```

Note that the package information is *added* to the cluster binary file. The cluster binary file no longer needs to be rebuilt from scratch (starting with MC/ServiceGuard release 10.10).

7-12. SLIDE: Package Scripts



Package Scripts

Package Run Script

- Script called by Serviceguard to start the application
- Core functionality provided by HP in the form of a template
- Customer edits template to be specific for their application
- One run script per package; script must be located on each node in the cluster

Package Halt Script

- Script called by Serviceguard to halt the application
- This script is normally the same script as the run script

package_script start (starts the application – never executed manually)

package_script stop (stops the application – never executed manually)

Student Notes

The package run / halt script contains all the information to bring up and bring down the application completely. It is highly recommended that these two scripts be the same file (referred to as a package “control script”), and the behavior of the script be determined by the first passed parameter.

The package control script template is created with the `cmmakepkg` command:

```
cmmakepkg -v -s pkg.cntl
```

Once the control script template is created, it must be customized for the specific package.

One of the parameters specified in the package control script is `SERVICE_CMD`. This command (which will be a service process) will be spawned and monitored by Serviceguard in the following way:

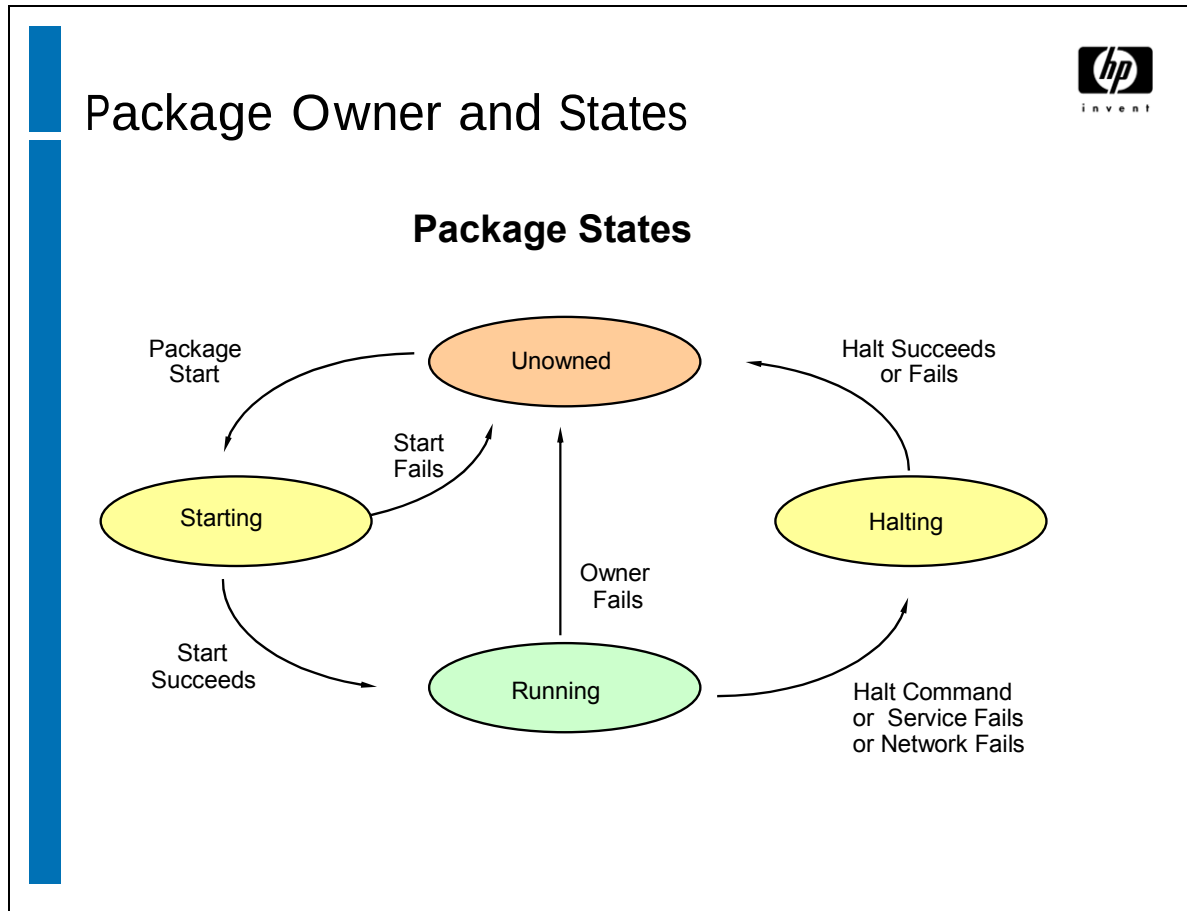
- The command will be spawned by `cmrunserv`. Once spawned, the resulting PID will be monitored every 10 seconds.

Packages and Services

- If the PID exits or terminates, Serviceguard responds with a service failure and takes appropriate actions.
- If SERVICE_CMD is a shell script that starts other processes (like database processes) and then exits, Serviceguard will see this normal exit as a failure and Serviceguard will respond accordingly.

NOTE: Serviceguard's mission is to keep persistent processes persisting.
Therefore, SERVICE_CMD must be a persistent process.

7-13. SLIDE: Package Owner and State



Student Notes

Package Owner

The owner of a package is the node that is currently executing the package.

- The coordinator determines which node will be the owner for all the packages. The current package switching state and node switching states are used to determine the owner. If no owner can be assigned, then the owner is set to `NO_OWNER`.
- If a package owner fails, the coordinator sets the owner to `NO_OWNER` and immediately begins to look for a new owner.

Package State

The package state is used to indicate the health and status of a package. There are four possible package states:

Unowned This state indicates the package is NOT running on any node and no attempt is being made to start up the package. A package would be in this state if package

	switching is disabled, or node switching has been disabled for all nodes related to this package.
Starting	This state indicates the package run script is currently being executed in an attempt to bring up the package. Packages do not stay in this state for very long since this is a transition state between Unowned and Running.
Running	This state indicates the package is up and running. This will be the state for most packages within a healthy cluster.
Halting	This state indicates the halt script is being executed to bring down the package on the current machine. Once the halt script completes, the package will transition into an Unowned state and the coordinator will try to find a new owner (if one exists depending on the package and node switching states). If the package has been halted manually with the <code>cmhaltpkg</code> command, the package will not be started on another node, even if package switching is enabled and a node is enabled to accept it.

Package Manager

Package management is performed by the package manager instance on each node in the following types of clusters: (1) a Serviceguard cluster, (2) a ServiceGuard OPS edition cluster, or (3) a Serviceguard with the Serviceguard Extension for RAC cluster. Each package has an *owner*, a *state*, and *attributes* associated with it. It is the owning node's responsibility to keep this package information and report it as needed to the coordinator. The package manager instance, which owns a particular package, is referred to as the *package owner*.

Package Coordinator

Package management is coordinated among nodes by the package manager instance, which resides on the cluster coordinator node. This package manager instance is referred to as the *package coordinator*.

The package coordinator –

- is on the same node as the *Cluster Coordinator*.
- directs package manipulation for all packages. The package manager instance, which owns a package, performs the actual work.
- keeps the information on the owner, state, and attributes and uses it to perform package management.
- must pass the information on to all nodes in the cluster. This allows global package information to persist across reboots.

NOTE: All nodes must have the same information to compensate for failures of the owning node or cluster coordinator.

7-14. SLIDE: Package Configuration Procedure



Package Configuration Procedure

1. `mkdir /etc/cmcluster/package_name` (Create directory for package specific files)
2. `cd /etc/cmcluster/package_name` (Change to the directory)
3. `cmmakepkg -v -p pkg.conf` (Create package configuration template)
4. `vi pkg.conf` (Modify package parameters)
5. `cmmakepkg -v -s pkg.cntl` (Create package control script template)
6. `vi pkg.cntl` (Modify script variables)
7. `ftp pkg.cntl` to other systems in the cluster (Distribute the control script to all nodes)
8. `cmcheckconf -v -P /etc/cmcluster/package_name/pkg.conf`
(Verify the package configuration file)
9. `cmapplyconf -v -P /etc/cmcluster/package_name/pkg.conf`
(Compile package into cluster binary file)

Student Notes

The slide shows the basic steps for configuring a package to run in a Serviceguard environment.

(Steps 1-4) Create the Package Configuration File

The files related to a particular package should be kept together in a subdirectory under `/etc/cmcluster`. The first step in configuring a package is to create a subdirectory for the package, and then create the package configuration file in this subdirectory.

Use `cmmakepkg -v -p pkg_config_file` to create the package configuration file.

(Steps 5 and 6) Create the Package Control Script

Once the package configuration file is created, the package control script needs to be created using the same approach. First use `cmmakepkg -v -s pkg_control_file` to create a template script, then modify the template to be specific for the application.

(Step 7) Copy Package Control Script and Other Files to Other Nodes

Serviceguard does not distribute the packaging related ASCII files to the other nodes in the cluster. This task must be done by the system administrator. The system administrator

should copy the package control script and any other package-related files (like an application monitor script, if being used) to the other nodes within the cluster. The packaging files should be kept in subdirectories by exactly the same pathnames using the same names.

(Steps 8 and 9) Create and Distribute the Binary File

Once the package files have been customized and distributed, the files should be checked (in conjunction with the cluster configuration file) for syntax and logic errors. The `cmcheckconf` command performs these checks. `cmcheckconf` is monolithic in that it doesn't check syntax among packages. This means `cmcheckconf` won't flag multiple packages that are configured with the same resources (volume groups are a particular problem, for example).

Specifically, `cmcheckconf` checks the following with respect to packages:

- Valid package name (and that at least one `NODE_NAME` entry is included).
- There are no duplicate parameter entries.
- Values for parameters are within permitted ranges.
- Run and halt scripts exist on all nodes in the cluster and are executable
- Configured resources are available on all cluster nodes

Once the files have been verified to contain no errors, the `cmapplyconf` command is used to create and distribute a new binary file.

7-15. SLIDE: The Package Script Log File



The Package Script Log File

- The default name of the Package Script log file is:
`/etc/cmcluster/package_name/pkg.cntl.log`
- The name is typically the control script name with a “.log” suffix.
- There is one log file per package.
- The log file contains high level descriptions of major actions performed by the control script.
- More helpful than `syslog` file for package troubleshooting

Student Notes

For each package control script, a corresponding package control script log file will be automatically created the first time the script is executed.

This log file contains information related to the starting and halting of the package. The log file provides detailed information on activities performed by the script (such as VG activations and IP address assignments) for each invocation of the package control script.

By default, the name of the package control script log file is
`/etc/cmcluster/pkg_name/pkg_name.cntl.log`.

Beginning with A.11.17, it is possible to configure a package log file with a different location and/or a different name.

See the `#SCRIPT_LOG_FILE` line in the `package.conf` file for instructions to change the default package log file name and location.

7-16. SLIDE: Review of Commands for Controlling a Cluster

Review of Commands for Controlling a Cluster



Package Management Commands

- cmrunpkg
- cmhaltpkg
- cmmodpkg

Cluster Management Commands

- cmruncl
- cmhaltcl
- cmviewcl

Node Management Commands

- cmrunnode
- cmhaltnode

Student Notes

Package management, cluster management, and node management commands are used to manage the cluster once the binary file has been created and distributed.

Package Management Commands

cmrunpkg [-v] [-n NodeName] PackageName

[-v] Verbose output

[-n] Run package on a specific node

cmhaltpkg [-v] [-n NodeName] PackageName

[-v] Verbose output

[-n] Halt package on a specific node

(if package is not running on "NodeName", it will not be halted)

cmmodpkg [-v] {-e|-d} [-n NodeName] . . . PackageName

[-v] Verbose output

-e Enable

-d Disable

[-n] Modify package for a specific node - else global package attributes are modified

Cluster Management Commands

```
cmruncl [-v] [-f] [-n NodeName]
[ -v ]   Verbose output
[ -f ]   Force cluster to start up without warning messages
[ -n ]   Run cmcld on a specific node

cmhaltcl [-v] [-f]
[ -v ]   Verbose output
[ -f ]   Force cluster to shutdown even if packages are running. This option will cause
         the halt script to be executed for running packages.

cmviewcl [-v] [-n NodeName] [-p PackageName] [-l {pkg|cluster|node}]
[ -v ]   Verbose output
[ -n ]   View information only about the specific node
[ -p ]   View information only about the specific package
[ -l ]   pkg | cluster | node      Display only package, cluster, or node
                                         specific information
```

WARNING: Serviceguard cannot guarantee data integrity if you try to start a cluster with the `cmruncl -f -n` command. These options should be used only in troubleshooting situations; they should *not* be used in a production environment.

Node Management Commands

```
cmrunnode [-v] [NodeName]
[ -v ]   Verbose output

cmhaltnode [-v] [-f] [NodeName]
[ -v ]   Verbose output
[ -f ]   Force node to shutdown even if packages are running on the node. This option
         will cause the halt script to execute for running packages.
```

7-17. REVIEW: Check Your Understanding

Directions

Answer the following questions.

1. Where should the application executables reside? A single copy on the shared disks along with the application database and other dynamic data files; or two copies, one on each server in a non-shared (private) volume group?
2. Should the shared file systems have entries in `/etc/fstab`? Why not?
3. Should the shared volume groups be activated when the system boots (which happens by default)? How can this be prevented?
4. How can you view the contents of the shared data with the cluster down? With the package down? In other words, how do you gain access to the database to do database maintenance, etc, and insure no users are accessing the application (either by taking down the package or the cluster)?
5. If we kill an application on `node1` it automatically restarts on `node2`, but when we kill it on `node2` it does not automatically restart on `node1`. Why not? What must we do so it can switch back?

6. How can we manually move a single package from node1 to node2?
7. How can we take one node out of service (and move applications/packages to adoptive nodes)? Return it to service? Will packages automatically return to primary node?
8. How can we stop one application, for example, for maintenance? How do we restart it?
9. How does Serviceguard detect node failures, network failures, and application failures?
10. Can we perform database maintenance with the cluster down?
11. Why does Serviceguard require all nodes to be present when the cluster is first formed? How can we override this? Would we want to?

12. Under what circumstances is a floating IP address beneficial/necessary? Would this IP address be added to your DNS server? What host name should be used? Can an application/package have multiple IP addresses? Would this be of value?
13. How would you disable lower priority applications during a failover period? Notify administrators?
14. If application, server, disk, router, SCSI card, network card, etc fails, it should be transparent to the user. What if it is also transparent to the administrator? How can we insure against this?

7-18. LAB: Package Configuration #1: logprog Package

Directions

In this lab, we will configure a package called logprog to run in our cluster.

NOTE: We will add additional features to this package in later modules.

Part 1: Compile our application.

On Node1

1. The source code (logprog.c) for your application can be found in the /labs directory. Compile and save it as /usr/sbin/logprog. This will be the application around which we will build our package.

When this program runs, it will echo a status message to /tmp/logprog.out once every 3 seconds.

Compile your program as follows:

```
# cd /labs; cc logprog.c -o /usr/sbin/logprog
```

2. In one window, execute your logprog program. Open another window and verify the output with tail -f /tmp/logprog.out

```
Window1# /usr/sbin/logprog
```

```
Window2# tail -f /tmp/logprog.out
```

3. After verifying that logprog operates as expected, press <ctrl>C to stop it.

Part 2: Create and Edit the Package Configuration File

On Node1

1. Make a directory for the logprog package files.
2. Change directory to the logprog package directory that was just created. Once there, create the package configuration file template.
3. Next, edit the package configuration file template.

```
# vi logprog.conf
```

and modify the following parameters:

```
PACKAGE_NAME      logprog
NODE_NAME          <node1>
NODE_NAME          <node2>
RUN_SCRIPT         /etc/cmcluster/logprog/logprog.cntl
RUN_SCRIPT_TIMEOUT NO_TIMEOUT
HALT_SCRIPT        /etc/cmcluster/logprog/logprog.cntl
HALT_SCRIPT_TIMEOUT NO_TIMEOUT
SERVICE_NAME      logprog_service
SUBNET             X.X.X.X    ← Use netstat -in to obtain the cluster 10-net
                               subnet (Verify with your instructor)
```

4. Save the changes and exit the file.

Part 3: Create and Edit the Package Control Script, and Distribute this Script to All Nodes

On Node1

1. First, we create the package control script template.
2. Secondly, edit the package control script template. Remember that this file is a script and that all variable assignments cannot contain any spaces.

```
# vi logprog.cntl
```

and modify the following parameters:

```
SERVICE_NAME[0]="logprog_service"  
SERVICE_CMD[0]="/usr/sbin/logprog"  
SERVICE_RESTART[0]="-r 2"
```

3. Save changes and exit the file.
4. Next, on Node2, create a subdirectory for your logprog package.
5. Now, back on Node1, make the logprog control script executable, and copy it to Node2 within the cluster.
6. Copy your new /usr/sbin/logprog program to the same directory on Node2.

Part 4: Create and Distribute the Binary File, and Start the Package

On Node1

1. Check the package configuration file for errors and fix as needed.
2. Since the configuration file is okay, apply the new configuration, thus compiling and distributing a new cluster binary file.
3. View the package status.
The package should be halted, but enabled to run on either node.
4. Start the package.
5. We can test the `logprog` package behavior by doing the following:

```
# cmhaltpkg -v logprog  
# cmviewcl -vp logprog  
  
# cmrunpkg -v -n <adoptive_node> logprog  
# cmmodpkg -v -e logprog  
  
# cmviewcl -vp logprog
```

6. We can further test our package by killing the logprog PID. We will kill the logprog process PID 3 times; the first two times because we specified 2 retries, and the third time to force a failover.

In the previous step, we moved the package to the <adoptive_node>. Execute the following steps on <adoptive_node> to force a failover back to <primary_node>.

```
# ps -ef | grep logprog      ← Kill the logprog process for the first time.
# kill <pid>                  ← The process should restart on the same node.

# ps -ef | grep logprog      ← Kill the logprog process for the second time.
# kill <pid>                  ← The process should restart on the same node.

# ps -ef | grep logprog      ← Kill the logprog process for the third time.
# kill <pid>                  ← Process should fail over to <primary node>.

# cmviewcl -v                ←The logprog package restarts on <primary node>.

# cmmodpkg -v -e -n <adoptive_node> logprog

# cmviewcl -v                ← logprog is now running on <primary node>.
```

7. The package should now be running on <primary_node>. Execute the following steps on <primary_node> to force a failover back to <adoptive_node>.

```
# ps -ef | grep logprog      ← Kill the logprog process for the first time.
# kill <pid>                  ← The process should restart on the same node.
# ps -ef | grep logprog      ← Kill the logprog process for the second time.
# kill <pid>                  ← The process should restart on the same node.

# ps -ef | grep logprog      ← Kill the logprog process for the third time.
# kill <pid>                  ← Process should fail over to <adoptive node>.
# cmviewcl -v                ← The logprog package restarts on
                             <adoptive_node>.

# cmmodpkg -v -e -n <primary_node> logprog

# cmviewcl -v                ← logprog now running on <adoptive node>.
```

8. Congratulations on configuring your first Serviceguard package!

7-19. SOLUTION: Check Your Understanding

Directions

Answer the following questions.

1. Where should the application executables reside? A single copy on the shared disks along with the application database and other dynamic data files; or two copies, one on each server in a non-shared (private) volume group?

Answer:

Two copies: one on each server.

2. Should the shared file systems have entries in /etc/fstab? Why or why not?

Answer:

No. We want Serviceguard to mount and unmount these file systems as packages are started and stopped on each node.

3. Should the shared volume groups be activated when the system boots (which happens by default)? How can this be prevented?

Answer:

No. We want Serviceguard to activate volume groups as needed. Edit /etc/lvmrc and set AUTO_VG_ACTIVATE=0.

4. How can you view the contents of the shared data with the cluster down? With the package down? In other words, how do you gain access to the database to do database maintenance, etc., and insure no users are accessing the application (either by taking down the package or the cluster)?

Answer:

```
# vgchange -c n vg01
# vgchange -a y vg01
# mount ...
```

5. If we kill an application on **node1** it automatically restarts on **node2**, but when we kill it on **node2** it does not automatically restart on **node1**. Why not? What must we do so it can switch back?

Answer:

Packages are not allowed to fail back until the administrator can correct the problem on **node1** that caused the original failover.

To switch the package back:

```
# cmmmodpkg -e -n node1 pkg
```


6. How can we manually move a single package from **node1** to **node2**?

Answer:

```
<Node1> # cmhaltpkg -v pkg_name
```

```
<Node1> # cmrunpkg -v -n <node2> pkg_name
```

7. How can we take one node out of service (and move applications/packages to adoptive nodes)? Return it to service? Will packages automatically return to primary node?

Answer:

```
<Node1> # cmhaltnode -v -f <node2>          ← to halt  
<Node1> # cmrunnode -v <node2>              ← to restart the node
```

No, packages will not automatically return to their node.

8. How can we stop one application, for example, for maintenance? How do we restart it?

Answer:

```
# cmhaltpkg -v pkg_name  
# cmrunpkg -v pkg_name
```

9. How does Serviceguard detect node failures, network failures, and application failures?

Answer:

Node failure — no heartbeat. We can miss heartbeats for many reasons other than node failure ... hung daemons, bad network, network traffic, real time user processes, etc.

Application failure — by definition, when the service monitor exits, the application has failed. An operator killing a process by mistake, a bug in the monitor script, etc. can mislead Serviceguard.

Network failure - all the network cards on the same subnet (primary and failover cards) attempt to communicate using their hardware addresses. (By default, this communication occurs every two seconds).

10. Can we perform database maintenance with the cluster down?

Answer:

Yes. See question 4 above.

11. Why does Serviceguard require all nodes to be present when the cluster is first formed? How can we override this? Would we want to?

Answer:

If we are forming a cluster and one or more nodes is unreachable, there is always a possibility that the applications are already running on those unreachable nodes. In order to insure having the same application running on more than one node at a time does not corrupt our application database, the cluster does not form without all nodes present and accounted for. If we had an extended power outage, and for whatever reason one of the nodes is not booting (power supply destroyed by power surge), we would want to override this with the **-n** option of **cmrunc l**.

12. Under what circumstances is a floating IP address beneficial/necessary? Would this IP address be added to your DNS server? What host name should be used? Can an application/package have multiple IP addresses? Would this be of value?

Answer:

In order for clients to be able to transparently reconnect to their server processes after a failover, the application must have a floating IP address. Multiple IP addresses could be used to balance client traffic (and clients) over multiple subnets. The floating IP address would be assigned/maintained just like any other IP address. It should be documented on the DNS server. The hostname should be unique and reflect the application/package name.

13. How would you disable lower priority applications during a failover period? Notify administrators?

Answer:

The **start** script can check to see if the package is running on the adoptive system, and if so, can page the administrator; kill non-essential applications, and issue **cmhalt pkg** on non-essential packages, etc.

14. If application, server, disk, router, SCSI card, network card, etc fails, it should be transparent to the user. What if it is also transparent to the administrator? How can we insure against this?

Answer:

Eventually a redundant component will also fail and the package or cluster will be down. Ongoing monitoring is an essential component of high availability.

7-20. LAB Solution: Package Configuration #1: logprog Package

Directions

In this lab, we will configure a package called logprog to run in our cluster.

NOTE: We will add additional features to this package in later modules.

Part 1: Compile our application.

On Node1

1. The source code (logprog.c) for your application can be found in the /labs directory. Compile and save it as /usr/sbin/logprog. This will be the application around which we will build our package.

When this program runs, it will echo a status message to /tmp/logprog.out once every 3 seconds.

Compile your program as follows:

```
# cd /labs; cc logprog.c -o /usr/sbin/logprog
```

2. In one window, execute your logprog program. Open another window and verify the output with tail -f /tmp/logprog.out

```
Window1# /usr/sbin/logprog
```

```
Window2# tail -f /tmp/logprog.out
```

3. After verifying that logprog operates as expected, press <ctrl>C to stop it.

Part 2: Create and Edit the Package Configuration File

On Node1

1. Make a directory for the logprog package files.

Answer:

```
# cd /etc/cmcluster
# mkdir logprog
```

2. Change directory to the logprog package directory that was just created. Once there, create the package configuration file template.

Answer:

```
# cd logprog
# cmmakepkg -v -p logprog.conf
```

3. Next, edit the package configuration file template.

```
# vi logprog.conf
```

and modify the following parameters:

PACKAGE_NAME	logprog
NODE_NAME	<node1>
NODE_NAME	<node2>
RUN_SCRIPT	/etc/cmcluster/logprog/logprog.cntl
RUN_SCRIPT_TIMEOUT	NO_TIMEOUT
HALT_SCRIPT	/etc/cmcluster/logprog/logprog.cntl
HALT_SCRIPT_TIMEOUT	NO_TIMEOUT
SERVICE_NAME	logprog_service
SUBNET	X.X.X.X ← Use netstat -in to obtain the 10-net subnet

4. Save the changes and exit the file.

Part 3: Create and Edit the Package Control Script, and Distribute this Script to All Nodes

On Node1

1. First, we create the package control script template.

Answer:

```
# cd /etc/cmcluster/logprog
# cmmakepkg -v -s logprog.cntl
```

2. Secondly, edit the package control script template. Remember that this file is a script and that all variable assignments cannot contain any spaces.

```
# vi logprog.cntl
```

and modify the following parameters:

```
SERVICE_NAME[0]="logprog_service"
SERVICE_CMD[0]="/usr/sbin/logprog"
SERVICE_RESTART[0]="-r 2"
```

3. Save changes and exit the file.
4. Next, on Node2, create a subdirectory for your logprog package.

Answer:

```
# mkdir /etc/cmcluster/logprog
```

5. Now, back on Node1, make the logprog control script executable, and copy it to Node2 within the cluster. Also, copy your new /usr/sbin/logprog program to the same directory on Node2.

Answer:

```
# chmod 755 logprog.cntl
# rcp logprog.cntl <Node2>:/etc/cmcluster/logprog/.
```

6. Copy your new /usr/sbin/logprog program to the same directory on Node2.

Answer:

```
# rcp /usr/sbin/logprog <Node2>:/usr/sbin/.
```

Part 4: Create and Distribute the Binary File, and Start the Package

On Node1

1. Check the package configuration file for errors and fix as needed.

Answer:

```
# cd /etc/cmcluster/logprog
# cmcheckconf -v -P logprog.conf
```

Fix any errors.

2. Since the configuration files are okay, apply the new configuration, thus compiling and distributing a new cluster binary file.

Answer:

```
# cd /etc/cmcluster/logprog
# cmapplyconf -v -P logprog.conf
```

3. View the package status.

Answer:

```
# cmviewcl -v
# cmviewcl -vp logprog
```

The package should be halted, but enabled to run on either node.

4. Start the package.

Answer:

```
# cmmodpkg -e logprog
```

5. We can test the logprog package behavior by doing the following:

```
# cmhaltpkg -v logprog
# cmviewcl -vp logprog

# cmrunpkg -v -n <adoptive_node> logprog
# cmmodpkg -v -e logprog

# cmviewcl -vp logprog
```

6. We can further test our package by killing the logprog PID. We will kill the logprog process PID 3 times; the first two times because we specified 2 retries, and the third time to force a failover.

In the previous step, we moved the package to the <adoptive_node>. Execute the following steps on <adoptive_node> to force a failover back to <primary_node>.

```
# ps -ef | grep logprog      ← Kill the logprog process for the first time.
# kill <pid>                  ← The process should restart on the same node.

# ps -ef | grep logprog      ← Kill the logprog process for the second time.
# kill <pid>                  ← The process should restart on the same node.

# ps -ef | grep logprog      ← Kill the logprog process for the third time.
# kill <pid>                  ← Process should fail over to <primary node>.

# cmviewcl -v                ←The logprog package restarts on <primary node>.

# cmmodpkg -v -e -n <adoptive_node> logprog

# cmviewcl -v                ← logprog is now running on <primary node>.
```

7. The package should now be running on <primary_node>. Execute the following steps on <primary_node> to force a failover back to <adoptive_node>.

```
# ps -ef | grep logprog      ← Kill the logprog process for the first time.
# kill <pid>                  ← The process should restart on the same node.
# ps -ef | grep logprog      ← Kill the logprog process for the second time.
# kill <pid>                  ← The process should restart on the same node.

# ps -ef | grep logprog      ← Kill the logprog process for the third time.
# kill <pid>                  ← Process should fail over to <adoptive node>.
# cmviewcl -v                ← The logprog package restarts on
                             <adoptive_node>.

# cmmodpkg -v -e -n <primary_node> logprog

# cmviewcl -v                ← logprog now running on <adoptive node>.
```

8. Congratulations on configuring your first Serviceguard package!

Module 8 — Package Policies

Objectives

Upon completion of this module, you will be able to do the following:

- Use `FAILOVER_POLICY` to determine which node a package starts up on after the package fails.
- Use the `FAILBACK_POLICY` to determine which node a package starts up on after a node fails.
- Determine proper Access Control Policies for a package
- Determine when `NODE_FAIL_FAST` or `SERVICE_FAIL_FAST` should be enabled.

8-1. SLIDE: Package Policies



Package Policies

- Failover policy
- Failback policy
- Access Control Policy
- Node fail fast
- Service fail fast

Student Notes

Failover Policy

The failover policy determines to which system a package fails during a failover situation. The two choices are to fail to a pre-configured system, or to fail to a system with the least number of packages.

Failback Policy

The failback policy determines when a package returns to its original node. The two choices are to automatically return the package as soon as the original node is available, or to require the system administrator to manually return the package to the original node.

Node Fail Fast

The `NODE_FAIL_FAST` parameter determines if the node is to fail upon a package failure. If multiple packages exist on one node and one of the packages fails, by default, just the failed package will move to another node. The node containing the failed package does not "fail" or TOC, and the remaining packages are allowed to continue on the node. This behavior is referred to as **node fail fast disabled**.

When a package is configured with "NODE_FAIL_FAST_ENABLED YES", then the node will TOC if the package fails for any reason.

Service Fail Fast

Along the same line of failing (TOCing) the node when a package fails, it is also possible to fail the node when a particular service process fails. This is generally considered drastic behavior; however, if the service process affects the health of the whole system, then such action may be appropriate. The default for every service process is **service fail fast disabled**.

8-2. SLIDE: Failover Policies

Failover Policies



- Failover Policy is a per package policy.
- Used to select an adoptive node whenever the package is started.
- CONFIGURED_NODE. Select the nodes in priority order. This option is the default.
- MIN_PACKAGE_NODE. Select the node which is running the least number of packages at this time.

Student Notes

Configuration of a `FAILOVER_POLICY` is done through the ASCII package configuration file. An excerpt of the description from the package file is shown below:

```
# Enter the failover policy for this package. This policy will be used
# to select an adoptive node whenever the package needs to be started.
# The default policy unless otherwise specified is CONFIGURED_NODE.
# This policy will select nodes in priority order from the list of
# NODE_NAME entries specified below.
#
# The alternative policy is MIN_PACKAGE_NODE. This policy will select
# the node, from the list of NODE_NAME entries below, which is
# running the least number of packages at the time this package needs
# to start.
```

`FAILOVER_POLICY`

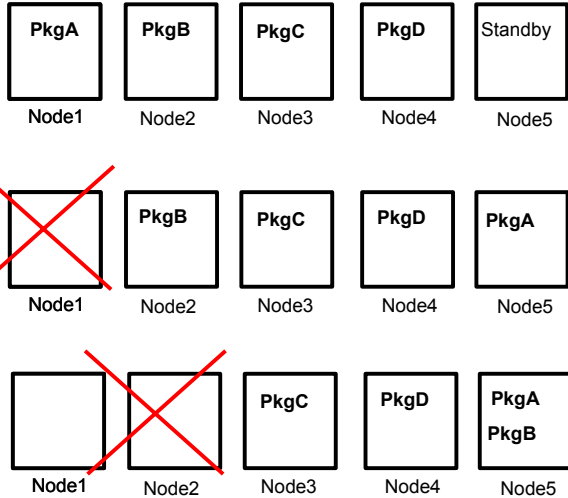
`CONFIGURED_NODE`

8-3. SLIDE: FAILOVER_POLICY: CONFIGURED_NODE



FAILOVER_POLICY: CONFIGURED_NODE

- Each package has a prioritized list of nodes.
- At failover time, package moves to next node in the list.
This is the default behavior.
- PkgA:
Node1, Node5, Node2, Node3, Node4
- PkgB:
Node2, Node5, Node1, Node3, Node4
- PkgC:
Node3, Node5, Node2, Node4, Node1
- PkgD:
Node4, Node5, Node3, Node1, Node2



The diagram illustrates the failover process for four packages (PkgA, PkgB, PkgC, PkgD) across five nodes (Node1 to Node5). Each package has a prioritized list of nodes. When a node fails, the package moves to the next node in its list. In this example, Node1 fails, so PkgA moves to Node5. Then Node2 fails, so PkgB moves to Node5. Then Node3 fails, so PkgC moves to Node5. Finally, Node4 fails, so PkgD moves to Node5. The result is that all four packages are running on Node5, and Node1 is left empty.

Student Notes

Packages begin their execution on their primary nodes (the first node listed in the package configuration file under the `NODE_NAME` parameter).

In the example on the slide, PkgA starts on Node1. After Node1 fails, PkgA moves to the next `NODE_NAME` in the prioritized list. For PkgA, this is Node5. Later, Node1 becomes healthy again and rejoins the cluster, but PkgA does not get returned to Node1.

Node2 then fails. PkgB moves to the next `NODE_NAME` in its prioritized list which is also Node5! In this situation, we would have two packages on Node5 and none on Node1.

A solution discussing how this situation could have been avoided is presented on the next slide.

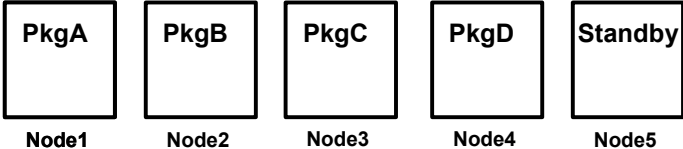
The `CONFIGURED_NODE` configuration is useful when you want to control specifically what the standby will be for a particular package or you simply want to connect the shared disks between only 2 nodes.

8-4. SLIDE: FAILOVER_POLICY: MIN_PACKAGE_NODE

FAILOVER_POLICY: MIN_PACKAGE_NODE



- At package start time, package will be started on the node with the fewest packages.
- Package will start only on nodes listed in the NODE_LIST.



Node	Package
Node1	PkgA
Node2	PkgB
Node3	PkgC
Node4	PkgD
Node5	Standby

Student Notes

The failover algorithm for MIN_PACKAGE_NODE is:

- At the time of the failure event, the failing package determines the list of possible adoptive nodes by querying the current state of the nodes in its configured list.
- The adoptive node is selected by selecting the node that is running fewer packages.
- If multiple nodes have the same number of packages, then the node listed first in the NODE_NAME list is selected.

8-5. SLIDE: Rotating Standby



Rotating Standby

Failover Policy can be used to perform the rotating standby operation.

Example:

	PkgA	PkgB	PkgC	PkgD	Standby
1	Node1	Node2	Node3	Node4	Node5
2	~~Node1~~	Node2	Node3	Node4	Node5
3	Node1	Node2	Node3	Node4	Node5
4	Node1	~~Node2~~	Node3	Node4	Node5

The diagram illustrates the rotating standby operation across four states. In state 1, PkgA is on Node1, PkgB on Node2, PkgC on Node3, PkgD on Node4, and the Standby role is on Node5. In state 2, Node1 fails (indicated by a red X), and PkgA moves to Node5. In state 3, Node1 rejoins the cluster and becomes the logical standby. In state 4, Node2 fails (indicated by a red X), and PkgB moves to Node1.

Student Notes

The above slide walks through an example of a "rotating standby" which can easily be implemented with the `FAILOVER_POLICY` of `MIN_PACKAGE_NODE`.

1. In the example, all packages can run on all nodes, and the data for each package is connected to all nodes in the cluster.
2. When node1 fails, PkgA moves to node5 because it has the fewest number of packages.
3. When node1 rejoins the cluster, node1 becomes the **logical standby**, because if a node were to fail, node1 would have the fewest number of packages running on it.

As PkgA continues on node5, another failure occurs, this time on node2. PkgB moves to node1, because it has the least number of packages. After node2 is repaired and rejoins the cluster, node2 will become the logical standby.

8-6. SLIDE: Failback Policies

Failback Policies



- FAILBACK_POLICY is a per package policy.
 - Used to allow the movement of a package back to its primary node.
 - MANUAL. No failback is attempted. Packages must manually be returned to the primary node when desired, or will be moved back if a failure on the current node occurs. This is the default.
 - AUTOMATIC. Packages will be moved from a non-primary node back to the primary node whenever the primary node has the ability to run the package.

Student Notes

Configuration of FAILBACK_POLICY is done through the ASCII package configuration file. An excerpt of the description from the package configuration file is shown below:

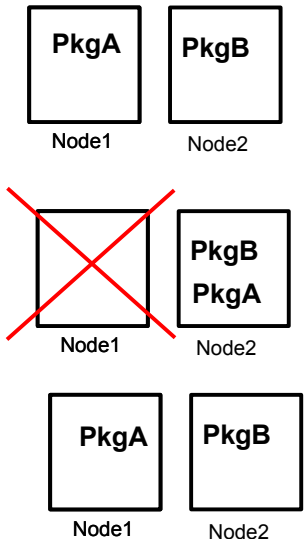
```
# Enter the failback policy for this package. This policy will be
# used to determine what action to take when a package is not
# running on its primary node and its primary node is capable of
# running the package. The default policy unless otherwise specified
# is MANUAL. The MANUAL policy means no attempt will be made to
# move the package back to its primary node when it is running on an
# adoptive node.
#
# The alternative policy is AUTOMATIC. This policy will attempt to
# move the package back to its primary node whenever the primary
# node is capable of running the package.
```


8-7. SLIDE: Example of Automatic Failback



Example of Automatic Failback

- PkgA and PkgB configured for node1 and node2.
- Node1 panics. PkgA moves to Node2
- Node1 reboots and PkgA is automatically halted and started on node1.



Student Notes

In the above slide, assume PkgA and PkgB have their `FAILBACK_POLICY` set to `AUTOMATIC`.

When `node1` goes down, the normal failover activities take place, moving the package to `node2`. When `node1` reboots and rejoins the cluster, the `FAILBACK_POLICY` is reference to determine whether to return PkgA to its original node. Since the policy is set to `AUTOMATIC`, when `node1` rejoins the cluster, PkgA is halted automatically on `node2`, and restarted on `node1`.

8-8. SLIDE: Access Control Policies



Access Control Policies

- Role-Based Access Control

By default, only the root user can access/control a package
Other users (optionally from other hosts) can be allowed certain privileges
- Access Control Policy Parameters

USER_NAME	ANY_USER –or- a list of specific users
USER_HOST	Where the user can execute SG commands
USER_ROLE	PACKAGE_ADMIN
- Limitations

Only the root user can configure a cluster or packages

Student Notes

Beginning with Serviceguard A.11.16, and on a per-package basis, there are options in the package configuration file that allow non-root users to manage/monitor a package. These options follow:

USER_NAME
USER_HOST
USER_ROLE

The first line must be USER_NAME, second USER_HOST, and third USER_ROLE.

1. USER_NAME can either be ANY_USER, or a maximum of 8 login names from the /etc/passwd file on user host.
2. USER_HOST is where the user can issue package-specific Serviceguard commands. If using Serviceguard Manager, it is the COM (Cluster Object Manager) server. Choose one of these three values: ANY_SERVICEGUARD_NODE, or (any) CLUSTER_MEMBER_NODE, or a specific node. For node, use the official hostname from domain name server, and not an IP address or fully qualified name.

3. `USER_ROLE` may only be `PACKAGE_ADMIN`, which includes `MONITOR`, plus administrative commands for packages in the cluster

Access control policy does not set a role for configuration capability. To configure a cluster or a package, a user must log on to one of the cluster's nodes as root (UID=0). Access control policy cannot limit root users' access.

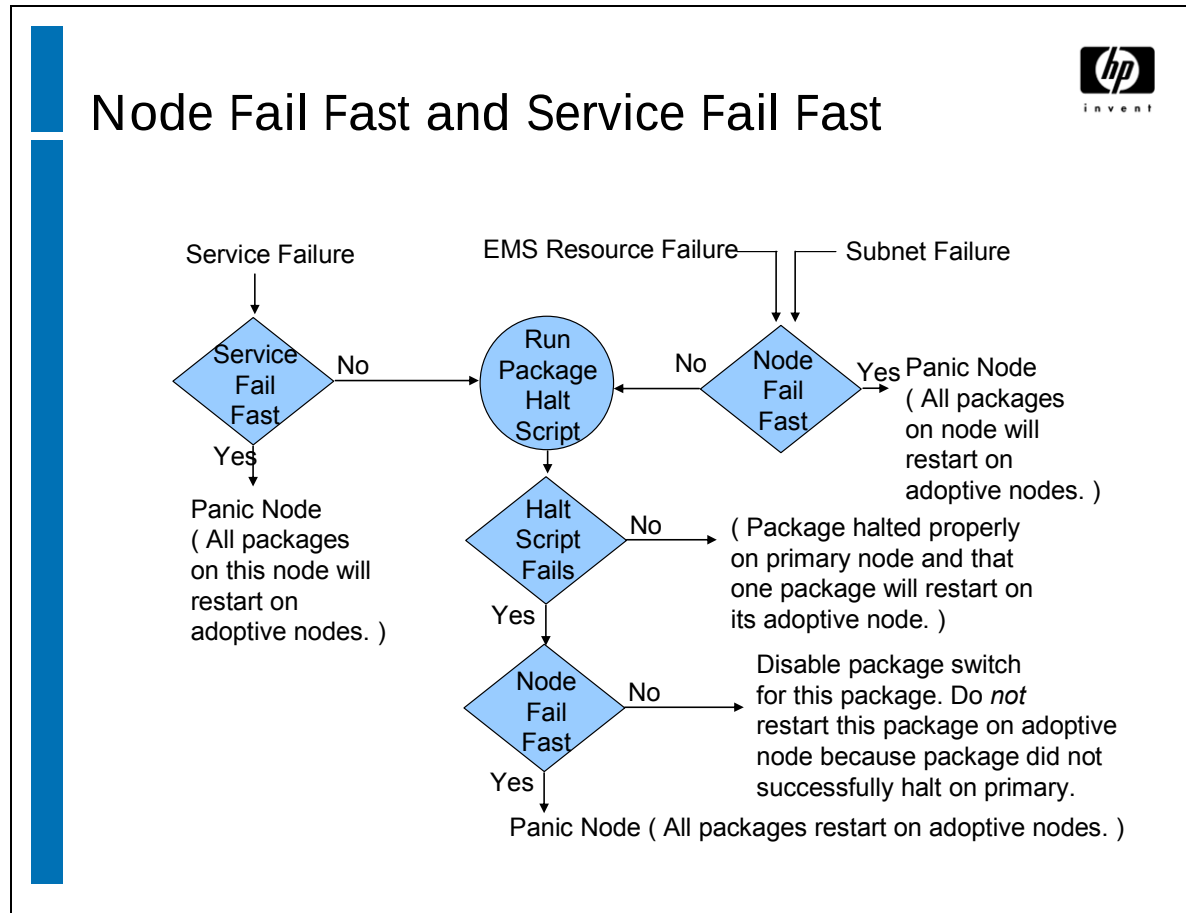
`MONITOR` and `FULL_ADMIN` can only be set in the cluster configuration file, and they apply to the entire cluster. `PACKAGE_ADMIN` can be set in the cluster or a package configuration file. If set in the cluster configuration file, `PACKAGE_ADMIN` applies to all configured packages. If set in a package configuration file, `PACKAGE_ADMIN` applies to that package only.

Conflicting or redundant policies will cause an error while applying the configuration, and will stop the process. If a role is configured in a cluster, do not duplicate it in a package.

Example: to configure a role for user `oracle` from node `rx26u891` to administer the `oracle` package, enter the following in the `oracle` package config file:

```
USER_NAME  oracle
USER_HOST  rx26u891
USER_ROLE  FULL_ADMIN
```

8-9. SLIDE: Node Fail Fast and Service Fail Fast



Student Notes

Node Fail Fast

If multiple packages exist on one node and one of the packages fails, by default, only the failed package will move to another node. The node containing the failed package does not panic, and the remaining packages are allowed to continue running on the node. This behavior is the default behavior and is configured with "NODE_FAIL_FAST_ENABLED NO" in the package configuration file.

When a package is configured with "NODE_FAIL_FAST_ENABLED YES", the node will fail, or TOC if:

- The node loses contact with the subnet configured by its SUBNET parameter,
- The package halt script fails, or
- A Resource failure occurs (from EMS: see Module 14).

Failing a node does not give the software a chance to gracefully bring down other packages or non-Serviceguard software, so this parameter should only be set when a package contains a service that is so mission-critical that loss of that service makes the node unusable.

This behavior is referred to as **node fail fast enabled**.

Service Fail Fast

Similarly to Node Fail Fast Enabled, it is also possible to fail the node when a particular service process fails. This is generally considered drastic behavior. However, if the service process affects the health of the whole system, then such action may be appropriate.

The default for every service process is to disable this attribute. This attribute can be selectively enabled on a per-service process basis. The variable `SERVICE_FAIL_FAST_ENABLED` is set to YES or NO to achieve the desired result.

The slide shows the flow of what happens when a service fails or a package fails depending on the two “fail fast” settings.

Service Failures

When a service fails, the default behavior is to restart the service if any restarts exist. If all the restarts have been used (or none were configured), then the package halt script is executed to bring down the package. If, however, service fail fast is enabled, the node panics immediately and the halt script is not executed.

Package Failures

Three situations can cause a package to fail on a healthy node:

- Subnet failure. If the package is configured to be dependent on a subnet and that subnet is determined to be inaccessible by the local system, then a package failure will occur.
- Halt script failure. If an error is encountered or the halt script times out during the execution of the halt script, the package has failed.
- If EMS is being used (to be discussed later in Module 16), and if there is an EMS resource failure, the package will be brought down and potentially failed over to an adoptive node.

Example

A database package is configured with “`SERVICE_FAIL_FAST_ENABLED NO`” and “`NODE_FAIL_FAST_ENABLED YES`”.

If a critical service process prematurely terminates, the halt script for the database package is executed. If the halt script succeeds in gracefully bringing down the database package, then the node is disabled for the package and the package restarts on an adoptive node in the cluster. A node failure does not occur.

If, however, the halt script fails (assume there were users logged into the database and it could not be shutdown), the halt script error would cause a node failure (panic), since “`NODE_FAIL_FAST_ENABLED YES`” was set. The panic of the node causes the `cm1vmd` process to disappear, and the package IP address is no longer configured on the local LAN card.

Package Policies

If "SERVICE_FAIL_FAST_ENABLED NO", the halt script would have failed and the node would have been left up. This would have left the cmlvmd process running and, depending on where and how the halt script failed, would have left the package IP address configured. The package would not be able to start on another node due to the volume group being exclusively accessed by the first node and the IP address still assigned.

8-10. LAB: FAILBACK POLICY

Directions

1. Modify your logprog package to fail back when the primary node comes back to the cluster, and test it out.

Note: It is not necessary to halt your cluster prior to performing the the package configuration changes below, but we are doing so to ensure the logprog package starts up on the primary node when we restart the cluster. In a production environment, this would not be necessary.

Answer:

In the "**logprog.conf**" file, change the line that now reads ...

FAILBACK_POLICY	MANUAL
-----------------	--------

to read ...

FAILBACK_POLICY	AUTOMATIC
-----------------	-----------

and then reconfigure the package in the normal way.

```
# cmhaltcl -f
# cd /etc/cmcluster/logprog
# cmcheckconf -v -P logprog.conf
# cmapplyconf -P logprog.conf
# cmruncl
# cmviewcl -vp logprog
# cmhaltnode -f (on primary)
# cmviewcl -vp logprog (and verify logprog is on adoptive node)
# cmrunnode (on primary)
# cmviewcl -vp logprog (and verify logprog comes back to primary node
again)
```

2. Did the logprog package come back to the primary node?
3. While there might be compelling reasons to use the AUTOMATIC failback policy, we will be performing other operations with the logprog package that require the MANUAL failback policy. Therefore, modify your logprog package configuration file again to reflect a MANUAL failback. Then check and apply.

8-11. LAB: Access Control Policy

Directions

In this lab, we will give user50 PACKAGE_ADMIN status to the logprog package, and we'll give this user this capability only on nodes within your cluster. While we could have given PACKAGE_ADMIN capability to individual users in the cluster ASCII file, there is no way to prevent those users from executing commands on packages for which they would not ordinarily be responsible.

In the package configuration file, assigning individual users PACKAGE_ADMIN capability will allow us to more tightly control who manages which packages.

1. We don't currently have a user50 on our systems, so let's add one. Assign this user a password of "hp". Be sure to do this on both nodes in your cluster.
2. Edit the logprog package configuration file and give user50 the full access to this package from either node in your cluster.
3. Check and apply the logprog package configuration file.
4. Login as user50 and halt the logprog package. Since the /usr/sbin directory (where most of the Serviceguard commands reside) is not in user50's PATH, you will need to specify the absolute path to the commands below. Start the package again on the adoptive node. Then bring it back to its primary node.

5. Login as root again for the remainder of the lab.

8-12. LAB: SERVICE_FAIL_FAST_ENABLED (Optional)

Directions

Special Note

This lab and the following lab provide an opportunity to use the SERVICE_FAIL_FAST_ENABLED and NODE_FAIL_FAST_ENABLED options on packages. These options are often not well understood.

These options generally *should not* be used in production environments.

Both of these options will cause a server to panic when an application-related fault occurs. Specifically, with SERVICE_FAIL_FAST_ENABLED, a service (application) failure will cause that node to panic. Generally, application failure can be handled with less severe action than a node panic. Also, it is entirely possible that when the package restarts on the adoptive node it will also fail there, causing both nodes to panic. For these reasons, SERVICE_FAIL_FAST_ENABLED is virtually never implemented in production environments.

The NODE_FAIL_FAST_ENABLED option causes a node to panic if the package attempts to halt, and the halt script fails. Again, in a production environment, this would generally be avoided. Less severe methods than a panic can be used to overcome an application-related failure (in this case a halt script failure). Also, the error could be encountered on the adoptive node, causing both nodes to panic.

1. First, halt the cluster. Then, restart the cluster. Finally, make sure your package is running.
2. Before continuing, make a backup copy of the `logprog.conf` and the `logprog.cntl` files.

- Now, modify the ASCII configuration file for the `logprog` package so that `SERVICE_FAIL_FAST_ENABLED` is enabled.
- Check the modified `logprog` configuration file. Make any necessary corrections and recheck if necessary. Apply the modified configuration. Will Serviceguard allow you to make these changes while the package is active?
- Start the `logprog` package. Kill it until a failover occurs. What happened? After a minute or so, the node on which `logprog` is running will panic.

8-13. LAB: NODE_FAIL_FAST_ENABLED (Optional)

Directions

In this part of the lab, we will configure and test the package again: this time by enabling the NODE_FAIL_FAST_ENABLED policy.

1. Wait for the failed system to reboot and rejoin the cluster.
2. Modify the ASCII configuration file for the logprog package so that SERVICE_FAIL_FAST_ENABLED is disabled, and NODE_FAIL_FAST_ENABLED is enabled.
3. Halt the logprog package. Check the modified configuration. If necessary, correct any errors and recheck the configuration. Apply the modified configuration. Verify that SERVICE_FAIL_FAST_ENABLED is disabled, and that NODE_FAIL_FAST_ENABLED is enabled.
4. Start the logprog package. Ensure that package switching is enabled.
5. Kill the logprog until a failover occurs. Did the node fail? Why or why not?

6. Re-enable the package switching for the node that just failed.
7. On the node where logprog is currently running, make a backup copy of the control script for logprog, if you have not already done so.

8. On the node where logprog is currently running, force an error in the halt script for logprog. You can do this by editing the logprog.cntl file. Add the command exit 99 to the customer_defined_halt_cmds function.

```
# vi logprog.cntl
```

```
function customer_defined_halt_commands
{
    exit 99
}
```

9. View the node_switching parameters for the logprog package. Make sure that logprog is enabled on all nodes. If not, enable it.
10. Kill the logprog enough times to force a failover. Did the node panic this time? Why?

11. Once the failed system reboots, restore the backup copy of the `logprog` control script.
12. Halt the `logprog` package if it is running. Modify the ASCII configuration file for `logprog` to disable both `NODE_FAIL_FAST_ENABLED` and `SERVICE_FAIL_FAST_ENABLED`. Check and apply the modified configuration.
13. Finally, start the `logprog` package.

8-14. LAB Solution: FAILBACK POLICY

Directions

1. Modify your logprog package to fail back when the primary node comes back to the cluster, and test it out.

Note: It is not necessary to halt your cluster prior to performing the the package configuration changes below, but we are doing so to ensure the logprog package starts up on the primary node when we restart the cluster. In a production environment, this would not be necessary.

Answer:

In the "**logprog.conf**" file, change the line that now reads ...

FAILBACK_POLICY	MANUAL
-----------------	--------

to read ...

FAILBACK_POLICY	AUTOMATIC
-----------------	-----------

and then reconfigure the package in the normal way.

```
# cmhaltcl -f
# cd /etc/cmcluster/logprog
# cmcheckconf -v -P logprog.conf
# cmapplyconf -P logprog.conf
# cmruncl
# cmviewcl -vp logprog
# cmhaltnode -f (on primary)
# cmviewcl -vp logprog (and verify logprog is on adoptive node)
# cmrunnode      (on primary)
# cmviewcl -vp logprog (and verify logprog comes back to primary node
                        again)
```

2. Did the logprog package come back to the primary node?

Answer:

Yes.

3. While there might be compelling reasons to use the AUTOMATIC failback policy, we will be performing other operations with the logprog package that require the MANUAL failback policy. Therefore, modify your logprog package configuration file again to reflect a MANUAL failback.

Answer:

```
# cmhaltpkg logprog
# vi /etc/cmcluster/logprog/logprog.conf
```

Change FAILBACK from AUTOMATIC to MANUAL

```
# cmcheckconf -P /etc/cmcluster/logprog/logprog.conf
# cmapplyconf -P /etc/cmcluster/logprog/logprog.conf
```


8–15. LAB Solution: Access Control Policy

Directions

In this lab, we will give user50 PACKAGE_ADMIN status to the logprog package, and we'll give this user this capability only on nodes within your cluster. While we could have given PACKAGE_ADMIN capability to individual users in the cluster ASCII file, there is no way to prevent those users from executing commands on packages for which they would not ordinarily be responsible.

In the package configuration file, assigning individual users PACKAGE_ADMIN capability will allow us to more tightly control who manages which packages.

1. We don't currently have a user50 on our systems, so let's add one. Assign this user a password of "hp". Be sure to do this on both nodes in your cluster.

Answer:

```
<node1> # useradd -m user50
<node1> # passwd user50
<node2> # useradd -m user50
<node2> # passwd user50
```

2. Edit the logprog package configuration file and give user50 the full access to this package from either node in your cluster.

Answer:

```
# vi /etc/cmcluster/logprog/logprog.conf
```

USER_NAME	user50
USER_HOST	CLUSTER_MEMBER_NODE
USER_ROLE	PACKAGE_ADMIN

3. Check and apply the logprog package configuration file.

Answer:

```
# cmcheckconf -P /etc/cmcluster/logprog/logprog.conf
# cmapplyconf -P /etc/cmcluster/logprog/logprog.conf
```

Module 8
Package Policies

4. Login as user50 and halt the logprog package. Since the /usr/sbin directory (where most of the Serviceguard commands reside) is not in user50's PATH, you will need to specify the absolute path to the commands below. Start the package again on the adoptive node. Then bring it back to its primary node.

Answer:

```
# login user50
$ /usr/sbin/cmhaltpkg logprog
$ /usr/sbin/cmrunpkg -n <node2> logprog
$ /usr/sbin/cmviewcl -vp logprog ← and verify that it's running on <node2>
$ /usr/sbin/cmhaltpkg logprog
$ /usr/sbin/cmmodpkg -e logprog
$ /usr/sbin/cmviewcl -vp logprog ← and verify that it's running on <node1>
```

5. Login as root again for the remainder of the lab.

Answer:

```
# login root
```

8-16. LAB Solution: SERVICE_FAIL_FAST_ENABLED (Optional)

Directions

Special Note

This lab and the following lab provide an opportunity to use the SERVICE_FAIL_FAST_ENABLED and NODE_FAIL_FAST_ENABLED options on packages. These options are often not well understood.

These options generally *should not* be used in production environments.

Both of these options will cause a server to panic when an application-related fault occurs. Specifically, with SERVICE_FAIL_FAST_ENABLED, a service (application) failure will cause that node to panic. Generally, application failure can be handled with less severe action than a node panic. Also, it is entirely possible that when the package restarts on the adoptive node it will also fail there, causing both nodes to panic. For these reasons, SERVICE_FAIL_FAST_ENABLED is virtually never implemented in production environments.

The NODE_FAIL_FAST_ENABLED option causes a node to panic if the package attempts to halt, and the halt script fails. Again, in a production environment, this would generally be avoided. Less severe methods than a panic can be used to overcome an application-related failure (in this case a halt script failure). Also, the error could be encountered on the adoptive node, causing both nodes to panic.

1. First, halt the cluster. Then, restart the cluster. Finally, make sure your package is running.

Answer:

```
# cmhaltcl -v -f
# cmruncl -v
```

2. Before continuing, make a backup copy of the logprog.conf and the logprog.cntl files.

Answer:

```
# cd /etc/cmcluster/logprog
# cp logprog.conf logprog.conf.save
# cp logprog.cntl logprog.cntl.save
```

Module 8
Package Policies

3. Now, modify the ASCII configuration file for the `logprog` package so that `SERVICE_FAIL_FAST_ENABLED` is enabled.

Answer:

```
# vi logprog.conf

SERVICE_FAIL_FAST_ENABLED      YES
```

4. Check the modified `logprog` configuration file. Make any necessary corrections and recheck if necessary. Apply the modified configuration. Will Serviceguard allow you to make these changes while the package is active?

Answer:

```
# cmcheckconf -v -P logprog.conf
# cmhaltpkg -v logprog
# cmapplyconf -v -P logprog.conf
```

5. Start the `logprog` package. Kill it until a failover occurs. What happened?

Answer:

```
# cmmodpkg -v -e logprog
# ps -ef | grep logprog
# kill <pid>
```

After a minute or so, the node on which `logprog` is running will panic.

8-17. LAB Solution: NODE_FAIL_FAST_ENABLED (Optional)

Directions

In this part of the lab, we will configure and test the package again: this time by enabling the NODE_FAIL_FAST_ENABLED policy.

1. Wait for the failed system to reboot and rejoin the cluster.
2. Modify the ASCII configuration file for the logprog package so that SERVICE_FAIL_FAST_ENABLED is disabled, and NODE_FAIL_FAST_ENABLED is enabled.

Answer:

```
# vi logprog.conf

SERVICE_FAIL_FAST_ENABLED      NO
NODE_FAIL_FAST_ENABLED          YES
```

3. Halt the logprog package. Check the modified configuration. If necessary, correct any errors and recheck the configuration. Apply the modified configuration. Verify that SERVICE_FAIL_FAST_ENABLED is disabled, and that NODE_FAIL_FAST_ENABLED is enabled.

Answer:

```
# cmhaltpkg -v logprog
# cmcheckconf -v -P logprog.conf
# cmapplyconf -v -P logprog.conf
```

4. Start the logprog package. Ensure that package switching is enabled.

Answer:

```
# cmmodpkg -v -e logprog
# cmviewcl -v -p logprog
```

5. Kill the logprog until a failover occurs. Did the node fail? Why or why not?

Answer:

```
# ps -ef | grep logprog
# kill <pid>
```

The node does not fail because the halt script was successful. This is the difference between `NODE_FAIL_FAST` and `SERVICE_FAIL_FAST`.

6. Re-enable the package switching for the node that just failed.

Answer:

```
# cmmopkg -v -e -n node_name logprog
```

7. On the node where logprog is currently running, make a backup copy of the control script for logprog, if you have not already done so.

Answer:

```
# cp logprog.cntl logprog.cntl.save
```

8. On the node where logprog is currently running, force an error in the `halt` script for logprog. You can do this by editing the `logprog.cntl` file. Add the command `exit 99` to the `customer_defined_halt_cmds` function.

```
# vi logprog.cntl
```

```
function customer_defined_halt_commands
{
    exit 99
}
```

9. View the `node_switching` parameters for the logprog package. Make sure that logprog is enabled on all nodes. If not, enable it.

Answer:

```
# cmviewcl -vp logprog
# cmmopkg -v -e -n node_name logprog
```

10. Kill the `logprog` enough times to force a failover. Did the node panic this time? Why?

Answer:

```
# ps -ef | grep logprog
# kill <pid>

<etc>
```

The node fails this time because the halt script fails (with return code 99).

11. Once the failed system reboots, restore the backup copy of the `logprog` control script.

Answer:

```
# cd /etc/cmcluster/logprog
# mv logprog.cntl.save logprog.cntl
```

12. Halt the `logprog` package if it is running. Modify the ASCII configuration file for `logprog` to disable both `NODE_FAIL_FAST_ENABLED` and `SERVICE_FAIL_FAST_ENABLED`. Check and apply the modified configuration.

Answer:

```
# cmhaltpkg -v logprog
# mv logprog.conf.save logprog.conf
# cmcheckconf -v -P logprog.conf
# cmapplyconf -v -P logprog.conf
# cmviewcl -v -p logprog
```

13. Finally, start the `logprog` package.

Answer:

```
# cmmodpkg -e logprog
```

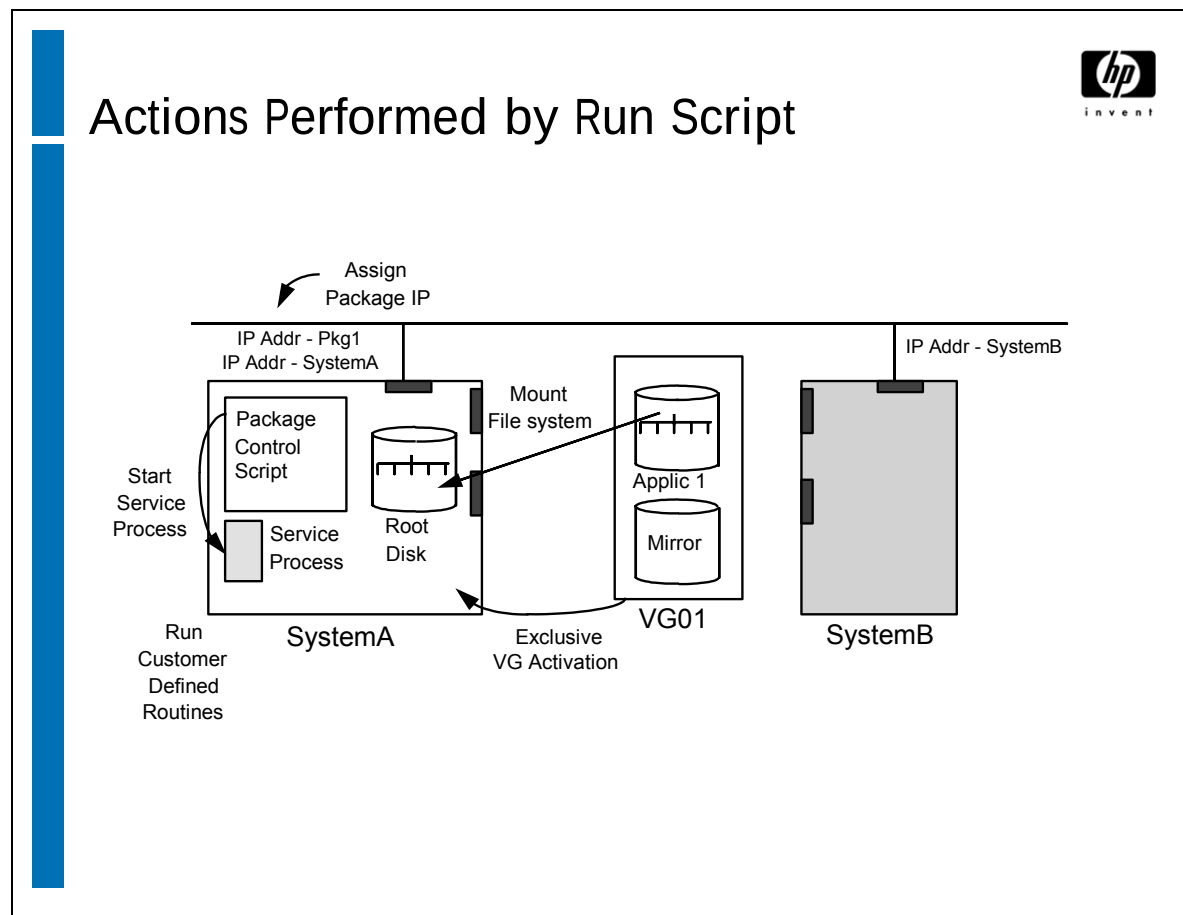
Module 9 — Application Monitoring Scripts

Objectives

Upon completion of this module, you will be able to do the following:

- Configure and implement an application monitor.
- List actions performed by the package control script.
- List three rules for Serviceguard service processes.

9-1. SLIDE: Actions Performed by Run Script



Student Notes

The package run script (also called the package control script) is responsible for starting the application on the nodes within the cluster.

The following actions are **always** performed by the package run script:

- Activating the volume group, if being used.
- Checking and mounting the file system, if being used.
- Assigning the package IP address, if being used.
- Executing any customer-defined run commands, if being used.
- Starting the service processes.

9-2. SLIDE: Control Script



Control Script

```
if [[ $1 = "start" ]]
then
    activate_volume_group
    check_and_mount
    add_ip_address
    customer_defined_run_cmds
    start_services
elif [[ $1 = "stop" ]]
then
    halt_services
    customer_defined_halt_cmds
    remove_ip_address
    umount_fs
    deactivate_volume_group
fi
```

Excerpt from
Package
Control Script

Student Notes

The slide shows an excerpt from the package control script. The excerpt is an abbreviated version of the main program contained within the package control script.

The major activities performed by the control script are grouped into subroutines. The order in which the subroutines are called corresponds to the order in which the major activities are performed.

NOTE: The sequence of activities used to start the package is in reverse order to the sequence of activities used to stop the package.

9-3. SLIDE: Rules for Service Processes



Rules for Service Processes

For Serviceguard Version 11.17:

General Rules:

- A package may have zero to a maximum of 30 service processes.
- Each service process can have zero to an unlimited number of restarts.
- If a single service process cannot be restarted, the package is halted.

A Service Process MUST:

- Be defined in the package configuration file.
- Be invoked from within the control script.
- Always be present in order for the package to be considered healthy.

A Service Process must NOT:

- Be a process which is spawned and respawned as requests come in.
- Be a process which only the application can invoke (i.e. Oracle daemons).

Student Notes

Care must be taken when defining service processes.

How Are Service Processes Started and Monitored by Serviceguard?

Service processes are defined in both the package configuration file and the package control script. Within the package control script, the parameter SERVICE_CMD contains the full path name of the program to run as the service process.

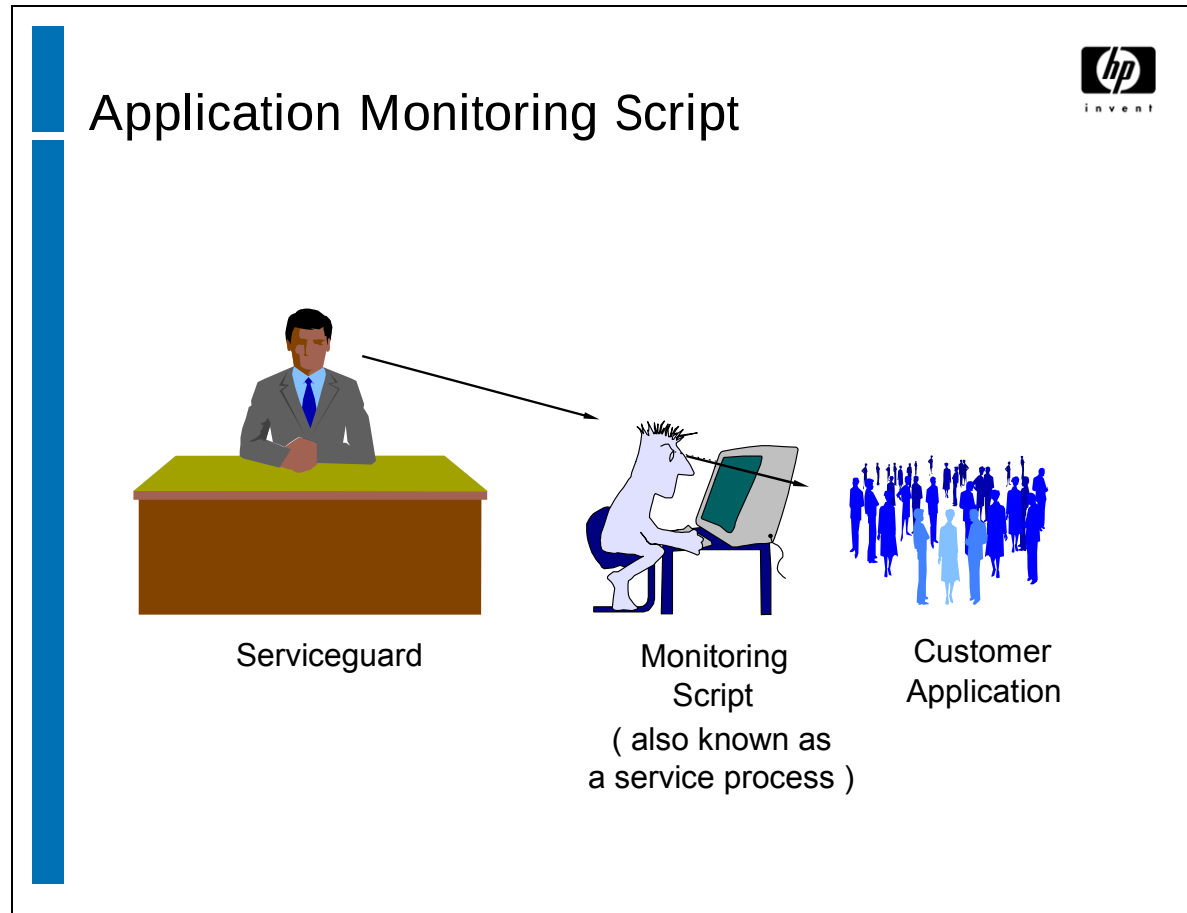
The service process specified in SERVICE_CMD is spawned by the `cmrunserv` command. The resulting process ID is then monitored by Serviceguard to determine the health of the application.

Based upon the above description, the following do not make good service processes:

- **Launch program.** A launch program is typically a script which spawns (or launches) a number of other programs and background processes related to an application and then exits. An example of a launch program is `dbstart`, a database program which starts the Oracle background daemon processes and then terminates.

- **Connection-oriented processes.** Connection-oriented processes are those which are spawned when a user request comes in. These processes service the user request and then exit when the request is completed. Examples of connection-oriented processes are those spawned by `inetd`, such as `telnetd`, `ftpd`, and `logind`.
- **Application-spawned processes.** Application-spawned processes are those which are started by some other application process or program and cannot be easily spawned from the command line or through the `cmrunserv` command. Examples of application-spawned processes are the Oracle daemons, such as `dbwr` or `lgwr`. Another example would be daemons started by the operating system at boot time, such as `nfsd`. Because Serviceguard does not start these processes directly, it does not know which PID to monitor since there could be multiple occurrences of these processes.

9-4. SLIDE: Application Monitoring Script



Student Notes

An application monitoring script is a persistent process that checks periodically to see if the application is still running. If the application is found to still be healthy, the script sleeps for a predefined number of seconds before checking again.

If, during one of its checks, it finds the application is not healthy, it may try to resolve the problem or simply exit.

Serviceguard uses these application monitoring programs as service processes started by the `cmrunserv` command. By using these monitoring scripts, Serviceguard is only concerned with whether the monitoring program is running. It leaves it up to the monitoring script to ensure the health of the application.

A major advantage of a monitoring script is the amount of flexibility that can be used to monitor the health of the application. In addition to checking whether critical application processes are running, the monitoring script can also check items like file systems being full, errors in log files, and deadlock conditions in database tables.

A second advantage of the monitoring scripts is the ability to build logic into the scripts to fix problems related to the application when they occur. Rather than exiting when an application process dies, the monitoring script could first try to respawn the application process before exiting. Rather than terminating when the file system becomes full, the monitoring script could try to free up space (maybe by looking for core files and removing them) before terminating.

The end result is that Serviceguard watches the monitoring script, and the monitoring script watches all aspects of the application.

The creation of monitoring scripts does not always have to be done from scratch. Many applications have sample scripts and instructions on how to "package" the application to run in a Serviceguard environment.

9-5. SLIDE: Example: The logprog Program



Example: The logprog Program

- Activate volume group
- File system check and mount
- Add IP address
- Customer defined run commands
Start the logprog program here.
 logprog
- Start logprog Monitoring Process (aka a service process)
Logprog.mon checks to see if the logprog program is running.
 If daemons are OK, sleep for 10 seconds, and check again.
If daemons are NOT OK, try to fix. If cannot be fixed, exit.

Student Notes

The slide shows the five major activities performed by the package control script when starting the logprog program with a monitoring script.

1. Activate the volume group if one exists. Please recall, though, that for the logprog package, there is no volume group currently being used.
2. Check and mount any file systems related to the package. Once again, however, for the logprog package, there are no file systems.
3. Add the IP address for the logprog package. Here also, we omit this step since the logprog package does not require IP addresses.
4. Execute any commands in the customer_defined_run_cmds function in the package control script. This function is where the command to start logprog is added.
5. Start the logprog monitoring script to monitor the logprog program started in step 4.

9–6. LAB: logprog Package Monitoring Script

Directions

1. Instead of using `/usr/sbin/logprog` as the service, create a separate service monitoring script called `/etc/cmcluster/logprog/logprog.mon` as follows:

```
# cd /etc/cmcluster/logprog
```

```
# vi logprog.mon
```

```
#!/usr/bin/sh

LOG="/etc/cmcluster/logprog/logprog.cntl.log"

echo "Now entering the logprog package monitor \c"      >> $LOG
echo "script on $(hostname) at $(date)."                >> $LOG
while true
do
    if ps -ef | grep -v grep | grep -q "/usr/sbin/logprog"
    then
        echo "Package logprog apparently ok \c"
        echo "on $(hostname) at $(date)."
        sleep 10
    else
        echo "Package logprog failed at $(date) \c"
        echo "from node $(hostname)."
        exit
    fi
done >> $LOG
```

The extra echo statements in the above script are sometimes referred to as "debug lines". Once the package and its scripts are working properly, these debug lines may be safely commented out. Good programming practice would suggest that these "debug lines" would be commented out (but not removed) from the script since they may be needed again in the future.

2. Make the monitor script executable.
3. Copy the monitor script to other system in your cluster.

Application Monitoring Scripts

4. In the logprog package control script `/etc/cmcluster/logprog/logprog.cntl`, change the following two lines. To accomplish this, remove the `SERVICE_CMD[0]="/usr/bin/logprog"` line and replace it with the following. Also, please remember to change the number of restarts to 0.

```
SERVICE_CMD[0]="/etc/cmcluster/logprog/logprog.mon"
SERVICE_RESTART[0]="-r 0"
```

In other words, Serviceguard will now be watching the `logprog.mon` process, and we will use zero restarts. In turn, the `logprog.mon` process will be watching the health of the `logprog` package itself.

5. In the `customer_defined_run_cmds` function of the `logprog` package control script, `/etc/cmcluster/logprog/logprog.cntl`, remove the line beginning with the colon (`:`). Then, in its place, enter the command to start our `logprog`. Do not remove the `"test_return 51"` line.

```
/usr/sbin/logprog &
```

Important: Do not forget to put your `logprog` process in the background!

6. In the `customer_defined_halt_cmds` function, remove the line beginning with the colon (`:`), and also remove any blank lines. Do not remove the `"test_return 52"` line. Then, replace it (them) with the following few lines. This will stop our `logprog` if it is running:

```
function customer_defined_halt_cmds
{
LOG="/etc/cmcluster/logprog/logprog.cntl.log"

echo "Now entering the customer_defined_halt_cmds \c"           >> $LOG
echo "on $(hostname) at $(date)."

```

Once again, the extra `echo` statements in the above code are sometimes referred to as "debug lines". Once the package and its scripts are working properly, these debug lines may be safely commented out but not removed (as discussed above in step 1).

NOTE: If Serviceguard believes the halt script did not properly stop the application, the halt script will fail, and Serviceguard will not start the package on the adoptive node to avoid risking application data corruption.

7. Copy the revised package control script to node2.
8. Verify that all the pieces work. Open a terminal window and execute `tail -f /etc/cmcluster/logprog/logprog.cntl.log` to view the package activities.
9. Halt the cluster and then verify that the logprog program has also exited.
10. Now, restart the cluster, which automatically restarts the logprog package.
11. Kill the logprog process (one time) and verify it restarts on the adoptive node.
12. Verify that the echo message was written to the package log file on the failing node.
13. Return the logprog package to the primary node by halting, then immediately restarting the cluster.
14. Congratulations on building and testing your first Serviceguard package using a monitor script!

9-7. LAB Solution: logprog Package Monitoring Script

Directions

1. Instead of using `/usr/sbin/logprog` as the service, create a separate service monitoring script called `/etc/cmcluster/logprog/logprog.mon` as follows:

```
# cd /etc/cmcluster/logprog
```

```
# vi logprog.mon
```

```
#!/usr/bin/sh

LOG="/etc/cmcluster/logprog/logprog.cntl.log"

echo "Now entering the logprog package monitor \c"      >> $LOG
echo "script on $(hostname) at $(date)."                >> $LOG
while true
do
    if ps -ef | grep -v grep | grep -q "/usr/sbin/logprog"
    then
        echo "Package logprog apparently ok \c"
        echo "on $(hostname) at $(date)."
        sleep 10
    else
        echo "Package logprog failed at $(date) \c"
        echo "from node $(hostname)."
        exit
    fi
done >> $LOG
```

The extra echo statements in the above script are sometimes referred to as "debug lines". Once the package and its scripts are working properly, these debug lines may be safely commented out. Good programming practice would suggest that these "debug lines" would be commented out (but not removed) from the script since they may be needed again in the future.

2. Make the monitor script executable.

Answer:

```
# chmod 755 /etc/cmcluster/logprog/logprog.mon
```

3. Copy the monitor script to other system in your cluster.

Answer:

```
# rcp logprog.mon <node2>:/etc/cmcluster/logprog/.
```

4. In the `logprog` package control script `/etc/cmcluster/logprog/logprog.cntl`, change the following two lines. To accomplish this, remove the `SERVICE_CMD[0]="/usr/bin/logprog"` line and replace it with the following. Also, please remember to change the number of restarts to 0.

```
SERVICE_CMD[0]="/etc/cmcluster/logprog/logprog.mon"
SERVICE_RESTART[0]="-r 0"
```

In other words, Serviceguard will now be watching the `logprog.mon` process, and we will use zero restarts. In turn, the `logprog.mon` process will be watching the health of the `logprog` package itself.

5. In the `customer_defined_run_cmds` function of the `logprog` package control script, `/etc/cmcluster/logprog/logprog.cntl`, remove the line beginning with the colon (`:`). Then, in its place, enter the command to start `logprog`. Do not remove the `"test_return 51"` line.

```
/usr/sbin/logprog &
```

Important: Do not forget to put your `logprog` process in the background!

6. In the `customer_defined_halt_cmds` function, remove the line beginning with the colon (`:`), and also remove any blank lines. Do not remove the `"test_return 52"` line. Then, replace it (them) with the following few lines. This will stop our `logprog` if it is running:

```
function customer_defined_halt_cmds
{
LOG="/etc/cmcluster/logprog/logprog.cntl.log"

echo "Now entering the customer_defined_halt_cmds \c"          >> $LOG
echo "on $(hostname) at $(date)."
```

Once again, the extra `echo` statements in the above code are sometimes referred to as "debug lines". Once the package and its scripts are working properly, these debug lines may be safely commented out but not removed (as discussed above in step 1).

NOTE: If Serviceguard believes the halt script did not properly stop the application, the halt script will fail, and Serviceguard will not start the package on the adoptive node to avoid risking application data corruption.

7. Copy the revised package control script to node2.

Answer:

```
# rcp /etc/cmcluster/logprog/logprog.cntl  
<node2>:/etc/cmcluster/logprog
```

8. Verify that all the pieces work. Open a terminal window and execute `tail -f /etc/cmcluster/logprog/logprog.cntl.log` to view the package activities.

Answer:

```
# tail -f /etc/cmcluster/logprog/logprog.cntl.log
```

9. Halt the cluster and then verify that the logprog program has also exited.

Answer:

```
# cmhaltcl -v -f  
# ps -ef | grep logprog    ( if necessary, kill the PID for the logprog process )
```

10. Now, restart the cluster, which automatically restarts the logprog package.

Answer:

```
# cmrunccl -v
```

11. Kill the logprog process (one time) and verify it restarts on the adoptive node.

Answer:

```
# ps -ef | grep logprog
# kill <pid>

# cmviewcl -v -p logprog
```

12. Verify that the echo message was written to the package log file on the failing node.

Answer:

```
# vi logprog.cntl.log
```

13. Return the logprog package to the primary node by halting, then immediately restarting the cluster.

Answer:

```
# cmhaltcl -v -f
# cmruncl -v
```

14. Congratulations on building and testing your first Serviceguard package using a monitor script!

Module 10 — Best Practices

Objectives

Upon completion of this module, you will be able to do the following:

- View and interpret current activities within the cluster.
- View cluster-related activities over time.
- View activities performed by the `run` and `halt` scripts.
- Use contributed Serviceguard commands to perform troubleshooting.
- Understand the Safety Time Register
- Replace a failed LVM lock disk

10-1. SLIDE: Best Practices



Best Practices

- Important log file information
- Resolve cluster and package configuration issues
- Test cluster operations
- Contributed commands
- Other considerations

Student Notes

Using "best practices" involves all the items on the above slide, including knowing which log files to monitor, knowing how to interpret and resolve configuration problems, and knowing what commands (including the contributed commands) to use to further diagnose problems.

This module covers the above topics.

10-2. SLIDE: Monitoring the syslog File



Monitoring the syslog File

```
# tail -f /var/adm/syslog/syslog.log

May 2 15:30:51 sysA cmcld[824]: Clvmd initialized successfully
May 2 15:30:53 sysA cmcld[824]: Executing '/etc/cmcluster/pkg.cntl'
May 2 15:30:54 sysA LVM[3460]: vgchange -a e /dev/vg01
May 2 15:30:57 sysA CM-pkg[956]: cmrunserv servl.l>>pkg.cntl.log2>&1
May 2 15:30:57 sysA cmcld[824]: Started pkg on node sysA
May 2 15:48:44 sysA cmcld[824]: Executing '/etc/cmcluster/pkg.cntl stop'
for package pkg
May 2 15:48:49 sysA CM-pkg[824]: cmmmodnet -r -i 15.19.81.75 15.19.80.0
May 2 15:48:50 sysA LVM[3460]: vgchange -a n /dev/vg01
May 2 15:48:51 sysA cmcld[824]: Halted pkg on node sysA
May 2 15:48:52 sysA cmcld[824]: Disabled pkg on node sysA
May 2 15:48:52 sysA cmcld[824]: Package script exited with NO_RESTART
May 2 15:48:52 sysA cmcld[824]: Examine file /etc/cmcluster/pkg.cntl.log
May 2 15:48:52 sysA cmcld[824]: Switching disabled on pkg
```

Student Notes

Use the `tail /var/adm/syslog/syslog.log` command to view events and activities that happened within the cluster in the recent past. The Serviceguard daemon `cmcld` writes status information and activities related to the cluster to the `syslog` file. Examples of entries found in the `syslog` file include the following:

- Initialization and shutdown of the cluster.
- Activation and deactivation of cluster volume groups.
- Package start and halt attempts.
- LAN card failures and recoveries.
- Cluster reformation due to kernel delays

A recommended "best practice" is to monitor the `syslog` file for occurrences of the text, **"NO_RESTART"**. This text string is written to the `syslog` file when a failure condition is encountered AND there is no enabled node for the package to be restarted on. This means the package will be in a DOWN state. The **"NO_RESTART"** is written in all caps to indicate the urgency of this message. It is the only string which Serviceguard writes to the `syslog` file in all caps.

Kernel delays will show up as Warnings. Examples of these include the following:

```
May 2 15:50:34 sysA cmcld: Warning: cmcld process was unable to run for  
the last 8 seconds,  
May 2 15:50:34 sysA cmcld: which is longer than the node timeout  
(5 seconds )
```

After this you normally see a cluster reformation. This indicates a kernel delay preventing cmcld from running.

10-3. SLIDE: Package Script Troubleshooting

Package Script Troubleshooting	
	
<u>Package Script Activity</u>	<u>Status</u>
Activate Volume Group	(OK)
File System Check and Mount	(OK)
Add IP Address	(OK)
Customer Defined Commands	(OK)
Start Services	(OK)
Activate Volume Group	(OK)
File System Check and Mount	(OK)
Add IP Address	(FAILED)

Unmount File System	
Deactivate Volume Group	

Student Notes

When a package starts, the following five major activities are performed by the package control script:

1. Activate volume group.
2. Check and mount file systems.
3. Add package IP address.
4. Execute customer-defined run commands (start application).
5. Start the service processes.

If a failure occurs in step 1 – 3, then the script needs to undo (or back out of) all previous completed steps to allow the package to restart on other nodes.

For example, if the package control script failed while adding the package IP address (as shown in the slide), then it is critical that the file system be unmounted and the volume group be deactivated such that the package has a chance of starting successfully on another node in the cluster. In this case, this is exactly what the package would do.

10-4. SLIDE: Package Script Log — Example 1



Package Script Log — Example 1

```
# tail -f /etc/cmcluster/pkg/pkg.cntl.log

##### Node "sysA": Starting pkg at Sun Apr 6 21:40:33 PST
Apr  6 21:40:33 - "sysA": Activating volume group /dev/vg01 with exclus
Volume group /dev/vg01 has been successfully changed.
Apr  6 21:40:36 - "sysA": Checking filesystems:
/dev/vg01/lvol1
Apr  6 21:40:37 - "sysA": Mounting /dev/vg01/lvol1 at /mcsg_dir
/mcsg_dir: No such file or directory
ERROR: Function check_and_mount
ERROR: Failed to mount /dev/vg01/lvol1 to /mcsg_dir
Apr  6 21:40:38 - "sysA": Deactivating volume group /dev/vg01
Volume group /dev/vg01 has been successfully changed.
-----
##### Node "sysA": Starting pkg at Sun Apr 6 21:46:09 PST
Apr  6 21:46:09 - "sysA": Activating volume group /dev/vg01 with exclus
Volume group /dev/vg01 has been successfully changed.
Apr  6 21:46:13 - "sysA": Checking filesystems:
/dev/vg01/lvol1
Apr  6 21:46:13 - "sysA": Mounting /dev/vg01/lvol1 at /mcsg_dir
Apr  6 21:46:14 - "sysA": Adding IP address 15.19.81.75 to subnet
```

Key on date field

Student Notes

All major activities performed by the package control script are written to the package control script log file. Each package will have its own control script log file called `/etc/cmcluster/pkg/pkg.cntl.log`. Each entry in the log file corresponds to one of the five activities performed by the package control script. Each entry is indexed with the date and time the specific activity was started.

Example

The example on the slide shows a package control script that succeeds in activating the volume group and checking the file system, but fails during the mount of the file system. Through the log file, we see the script backing out of the failed attempt by deactivating the volume group.

In this example, the error occurred because the mount point directory did not exist. Once the mount point directory is created, the script is re-executed with the package successfully being started (as shown on the bottom half of the slide).

This approach of backing out of a failed package start up attempt is only used if the error occurs during steps 2 or 3. If an error occurs during steps 4 or 5, then a different approach for backing out of the failed start up attempt is used.

10-5. SLIDE: Package Script Log — Example 2



Package Script Log — Example 2

```
# tail -f /etc/cmcluster/pkg/pkg.cntl.log

##### Node "sysA": Starting pkg at Sun Apr 6 21:46:09 PST
Apr  6 21:46:09 - "sysA": Activating volume group /dev/vg01 with exclus
Volume group /dev/vg01 has been successfully changed.
Apr  6 21:46:13 - "sysA": Checking filesystems:
Apr  6 21:46:13 - "sysA": Mounting /dev/vg01/lvol1 at /mcsg_dir
Apr  6 21:46:14 - "sysA": Adding IP address 15.19.81.75 to subnet
SQLDBA> Connected.
SQLDBA> ORACLE instance started.
Database mounted.
ORA-01157: cannot identify data file 1 - file not found
Attempting to dismount database.....Database dismounted.
Attempting to shutdown instance.....ORACLE instance shutdown.
Apr  6 21:46:48 - "sysA": Starting service oracle_monitor
-----

##### Node "sysA": Halting pkg at Sun Apr 6 21:40:55 PST
Apr  6 21:46:55 - "sysA": Halting service oracle_monitor
ERROR: No such process with PID
SQLDBA> Connected.
SQLDBA> DBA-00342: unable to complete internal login
```

Key on date field

Student Notes

The example on the slide illustrates how the script handles errors that occur during steps 4 and 5.

If the first three major steps of the package control script succeed (volume group activation, file system checking and mounting, and adding IP address), then the script will execute step 4 and 5 regardless of whether they are successful, and the script will no longer try to back out or undo previously successful steps.

If an error occurs during step 4 (the start up of the customer application) or step 5 (the start up of the service process), it is the responsibility of the halt script to undo all of the previous successful activities.

Example

In the example on the slide, steps 1–3 succeed. During step 4 (in this case, starting an Oracle application), an error occurs. Despite the error, the script continues to step 5 and starts up the service process.

When the service process starts (in this case the Oracle monitoring script), it quickly realizes the application is not healthy and terminates. The termination causes the cmcld daemon to launch the halt script for the package. The halt script then backs out (undoes) all five of the major activities.

10-6. SLIDE: Common Cluster Configuration Issues

Common Cluster Configuration Issues



Cluster-wide configuration issues:

- The `/etc/cmcluster/cmclnodelist` file does not contain names of all nodes in the cluster.
- Missing or incorrect entries in `/etc/services` or `/etc/inetd.conf`.
- Standby LAN card configured with an IP address.
- Multiple LAN cards on one node configured for the same subnet.
- LVM volume groups have non-matching minor numbers.
- Cluster lock disk not cabled to all nodes in cluster.
- An IP address is used for a nodename in `cmclconfig.ascii` file.
- Volume Group containing cluster lock disk must be activated (or already marked for clustering) before running the `cmapplyconf` command.

Student Notes

The following are some common problems that may occur when forming a cluster:

- The username `root` must be given as the second field for each entry in `/etc/cmcluster/cmclnodelist`. The following is a sample of a properly written `/etc/cmcluster/cmclnodelist` file for a 4-node cluster:

r213c2	root
r213c3	root
r213c4	root
r213c5	root

- Entries are missing from `/etc/services` or `/etc/inetd.conf`. This is especially common when the nodes in the cluster are NIS clients. In this case, the NIS master `/etc/services` file needs updating to include the high availability service entries.
- The standby LAN card is configured. In order for Serviceguard to configure a LAN card as standby, the LAN card must not be configured with an IP address.

- If the standby LAN card has been configured with an IP address, remove all references to it in `/etc/rc.config.d/netconf` and reboot.
- The heartbeat IP is configured for the same subnet as the data IP. If a separate LAN card is used for a dedicated heartbeat, that IP address must be on a separate subnet from the primary LAN card. This is a requirement of routing protocols that become confused when two LAN cards within the same machine are on the same subnet.
- There are volume groups containing inconsistent minor numbers between two systems. The configuration of shared LVM volume groups must be the same for all systems in the cluster.
- The cluster lock is cabled to only some nodes in the cluster. The cluster lock disk must be cabled to every node in the cluster. Remember, the cluster lock is only required for two nodes, but is recommended for three or four nodes.
- An IP address (instead of a host name) is used to specify a `NODE_NAME` in cluster configuration ASCII file. All node names specified in the cluster configuration file must be host names; they cannot be the IP addresses of the nodes.
- The cluster lock disk could not be initialized. The cluster lock disk is initialized when the `cmapplyconf` command is executed. In order to initialize the cluster lock disk, the volume group must be activated. If the volume group is not activated, an attempt to activate it in exclusive mode will be made. If the attempt fails, an error is returned indicating the cluster lock disk could not be initialized. To avoid this error, either ensure the volume group is already activated prior to running the `cmapplyconf` command or make sure that the cluster lock volume group is marked for clustering.

10-7. SLIDE: Common Package Configuration Issues



Common Package Configuration Issues

Package Configuration issues:

- Service name in `pkg.conf` does not match service name in `pkg.cntl`.
- Variable definition fields (like `$SERVICE_CMD`) are not quoted.
- Corrected package scripts are not propagated to other nodes in cluster.
- Mount point directory does not exist on all nodes in cluster.
- Monitor script errors (for example, run script does not have permission to execute the monitor script).

Student Notes

The following are some common problems that may occur when configuring a package:

- Inconsistent service names are used in the package configuration file and package control script. The name specified in the `SERVICE_NAME` variables must be same across both the package configuration file and control script.
- String variables with spaces are not quoted. Variables within the package control script must be quoted if they contain a space as part of the string.
- Corrections to script files are not propagated to other nodes in the cluster. It is easy to forget to propagate changes to control scripts to other nodes within the cluster. Forgetting to propagate these changes will lead to errors when the package fails to other nodes.
- Mount point directories exist only on one node. Mount point directories need to exist on every node that may potentially run the package.

- There are application monitoring script errors. Application monitoring scripts contain the same issues as package control scripts: forgetting to propagate changes, string variables not being quoted, and execute permission not existing on the file.

10-8. SLIDE: Testing Cluster Operations



Testing Cluster Operations

Components of the cmcld daemon which need to be tested:

- Test the Package Manager
- Test the Cluster Manager
- Test the Network Manager

Student Notes

Once the Serviceguard system is configured, the above components should be tested to ensure that they behave correctly. The following procedures below test that the cluster responds properly in the event of a package failure, a node failure, or a LAN failure.

Testing the Package Manager

To test that the package manager is operating correctly, perform the following procedure for each package on the cluster.

1. Obtain a PID of a service process:

```
# ps -ef | grep <service_process>
```

2. Kill the PID of the service process:

```
# kill <PID_service_process>
```

3. View the package status:

```
# cmviewcl -v -p <package_name>
```

4. Once you have verified the package fails over successfully, move the package back to the primary node:

```
# cmmmodpkg -v -e -n <primary_node> <package_name>
# cmhaltpkg -v <package_name>
# cmrunpkg -v <package_name>
# cmmmodpkg -v -e <package_name>
```

Testing the Cluster Manager

To test that the cluster manager is operating correctly, perform the following procedure.

1. Power off one of the nodes in the cluster:
2. Monitor the status of the cluster; ensure that the cluster reforms:

```
# cmviewcl -v
```

Ensure that the powered off node is in a halted state, and that its packages have been restarted.

3. Restore power to the halted node.
4. Monitor the status of the cluster; ensure that the node rejoins the cluster:

```
# cmviewcl -v
```

The node should be recognized by the cluster, but the packages should not be running on the node.

5. Move the package(s) back to the node:

```
# cmhaltpkg -v <package_name>
# cmrunpkg -v <package_name>
# cmmmodpkg -v -e <package_name>
```

Testing the Network Manager

To test that the network manager is operating correctly, perform the following procedure for each node in the cluster:

1. Identify the primary and standby LAN cards on the node:

```
# lanscan
# cmviewcl -v
```

2. Disconnect the LAN from the primary LAN card.
3. View the status of the LAN cards; the local switch should have occurred so that the primary LAN card is down, and the standby LAN card is up.

```
# cmviewcl -v
```

Module 10
Best Practices

4. Reconnect the LAN to the original primary LAN card. Verify the primary LAN card is back up.

```
# cmviewcl -v
```


10-9. SLIDE: Useful Troubleshooting Commands



Useful Troubleshooting Commands

<code>cmquerycl</code>	Verify and troubleshoot cluster and packages
<code>cmcheckconf</code>	Verify and troubleshoot cluster and packages
<code>cmscancel</code>	Creates a report regarding resources in the cluster
<code>cmviewconf</code>	Extracts data from the binary file, even when the cluster is not running
<code>cmsetlog</code>	Set logging levels for the <code>cmcl</code> cluster daemon

`cmsetlog` is circled with a dotted line. An arrow points from a box below to it.

This is a contributed command (no support).

Student Notes

The above slide shows a number of commands which can be used to collect information about the cluster.

The `cmquerycl` and `cmcheckconf` Commands

In addition to being able to verify cluster configuration information, the `cmquerycl` and `cmcheckconf` commands can also be used to troubleshoot the cluster.

These two commands check:

- The network addresses and network connections
- The cluster lock disk connectivity
- The validity of configuration parameters in the cluster and package configuration files

The `cmset log` Command

The `cmset log` command is used to modify the amount of information logged by the `cmclld` cluster daemon. The amount of information logged may be set by functionality area or for certain categories of errors.

The syntax for the `cmset log` command is

```
cmsetlog [-M <module>] [-C <category>] [log level] [[-r] | [-s]] [-f file]
```

in which

- | | |
|---------------|---|
| -r | Resets logging to defaults. |
| -s | Stops logging to file and logs just to <code>syslog.log</code> |
| -f file | Specifies a log file. |
| -M <module> | Specifies a module in the cluster daemon to monitor. If no modules are specified, all modules are selected. The following modules can be specified: |
| CLM | cluster management |
| PKG | package management |
| NET | network interface |
| SRV | service monitoring |
| LOC | local communications |
| REM | remote communications |
| SDB | status database |
| RES | resources |
| SYN | synchronization |
| CSV | command server |
| COM | communications server |
| DLM | distributed lock manager |
| GMS | ops group membership service |
| LVM | shared LVM |
| UTL | general support |
| CDB | configuration database |
| STA | status database API |
| DEV | storage devices |
| QSM | quorum devices |
| SEC | security |
| UTD | utility daemons |
| CMP | config/status proxy |
| -C <category> | Specifies the category of messages to modify. If no categories are specified, all categories are selected. The following categories can be specified: |
| ERR | Internal errors. Logs internal error events detected. |
| XER | External errors. Logs external error events detected. |
| DTH | Fatal errors. Logs events which cause <code>cmclld</code> to terminate. |

INT	Internal events. Logs general Serviceguard events.
EXT	External events. Logs general nonServiceguard events.
PER	Periodic events. Logs heartbeats and other monitoring events.
<level>	Specifies the level of logging to use when writing to the <code>syslog.log</code> file. The lower levels provide less information, the higher levels provide more information.
0	Minimal logging (default)
1	Logs additional information on event handling
2	Logs basic logic flow (state changes) with minimal detail
3	Logs enhanced logic flow with moderate detail
4	Logs complete logic flow with maximum detail
5	Logs function entry and exit
6	Logs intrafunction execution flow (records a lot of data)

The `cmscancl` Command

The `cmscancl` command displays information about all the nodes in a cluster in a structural report that allows you to compare such items as IP addresses or subnets, physical volume names for disks, and other node-specific items for all nodes in the cluster. The `cmscancl` command actually executes several different HP-UX commands on all nodes and gathers the output into a report on the node where you run the command.

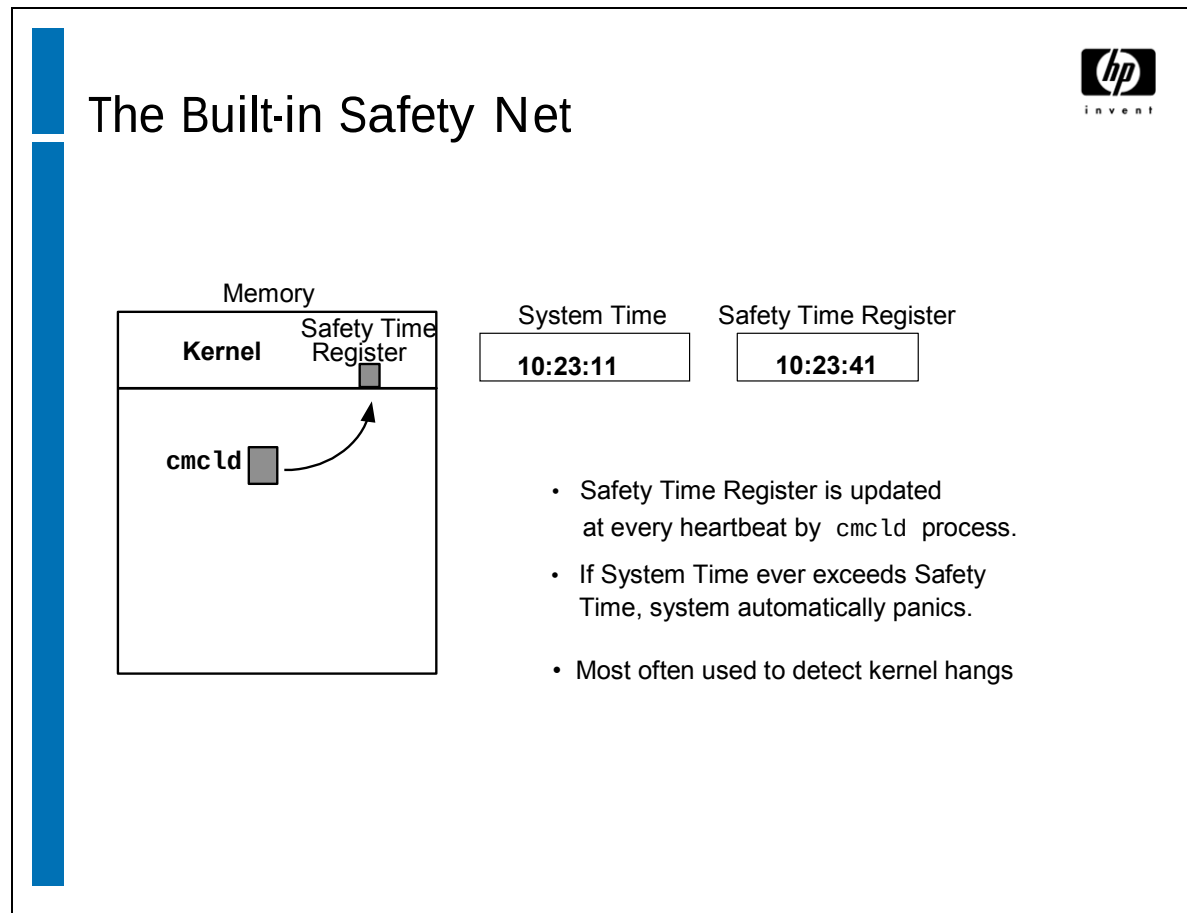
The data that is displayed by `cmscancl` follows:

Description	Source of Data
LAN device configuration and status	<code>lanscan</code> command
network status and interfaces	<code>netstat</code> command
file systems	<code>mount</code> command
LVM configuration	<code>/etc/lvmtab</code> file
LVM physical volume group data	<code>/etc/lvmpvg</code> file
link level connectivity for all links	<code>linkloop</code> command
binary configuration file	<code>cmviewconf</code> command

The `cmviewconf` Command

The `cmviewconf` command allows for the examination of the binary configuration file, even when the cluster is not running. The command displays the content of the binary file, `/etc/cmcluster/cmclconfig`, on the node where the command is executed.

10-10. SLIDE: The Built-In Safety Net



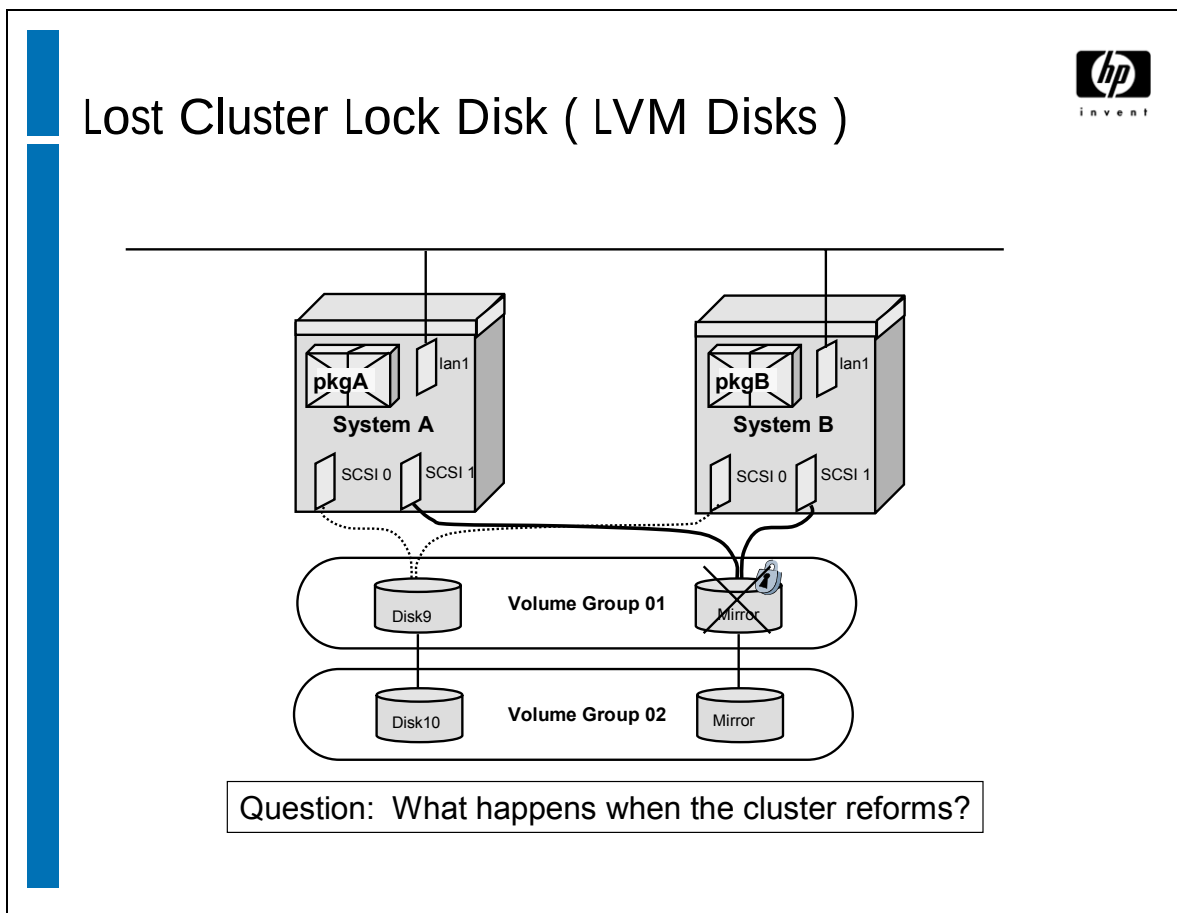
Student Notes

Safety time is a facility provided by the kernel to ensure a specified time period is never reached. If the predefined safety time is reached, the kernel panics the system rather than continuing to process past the safety time marker.

Serviceguard uses the safety time register to ensure the cmcld daemon is always running. If this daemon ever terminates, it is preferable for the system to TOC rather than jeopardize the integrity of the system. Every **HEARTBEAT_INTERVAL**, the cmcld daemon updates the safety time register to be at least 30 seconds (based on the cluster reformation time) ahead of the current system uptime. If the cmcld daemon terminates, the current system time will exceed the time in the safety register and the system will crash.

Serviceguard also uses the safety time register to detect kernel hangs.

10-11. SLIDE: Lost Cluster Lock Disk (LVM Disks)



Student Notes

We have already discussed the need for a lock disk. This slide focuses on using LVM disks to provide cluster lock services. If you are using a Quorum Server to provide cluster lock services, then the same discussion applies, except that we would be discussing the loss of the Quorum Server.

If the cluster splits exactly in half the lock disk is used as a tie-breaker to determine which half continues to run the packages and which half TOCs. If the cluster splits in half and the lock disk is not available then both halves will TOC and all packages will be down.

If the lock information is housed on a RAID 1 or RAID 5 LUN in a disk array, the loss of a single disk does not cause the lock information to be unavailable because the lock information is protected. Simply replace the broken disk and the array will automatically rebuild the data as usual.

Note: The user data on the lock disk was still available via a mirror copy of the data on another disk. On LVM mirrored disks as we are using in the classroom, the cluster lock resides in the BBRA of the lock disk. This is an LVM reserved area that is not mirrored.

If the lock information is housed on a LVM disk (and not in a RAID 1 or RAID 5 LUN) and the lock disk fails we have extra manual steps that must be taken to reinitialize the lock information.

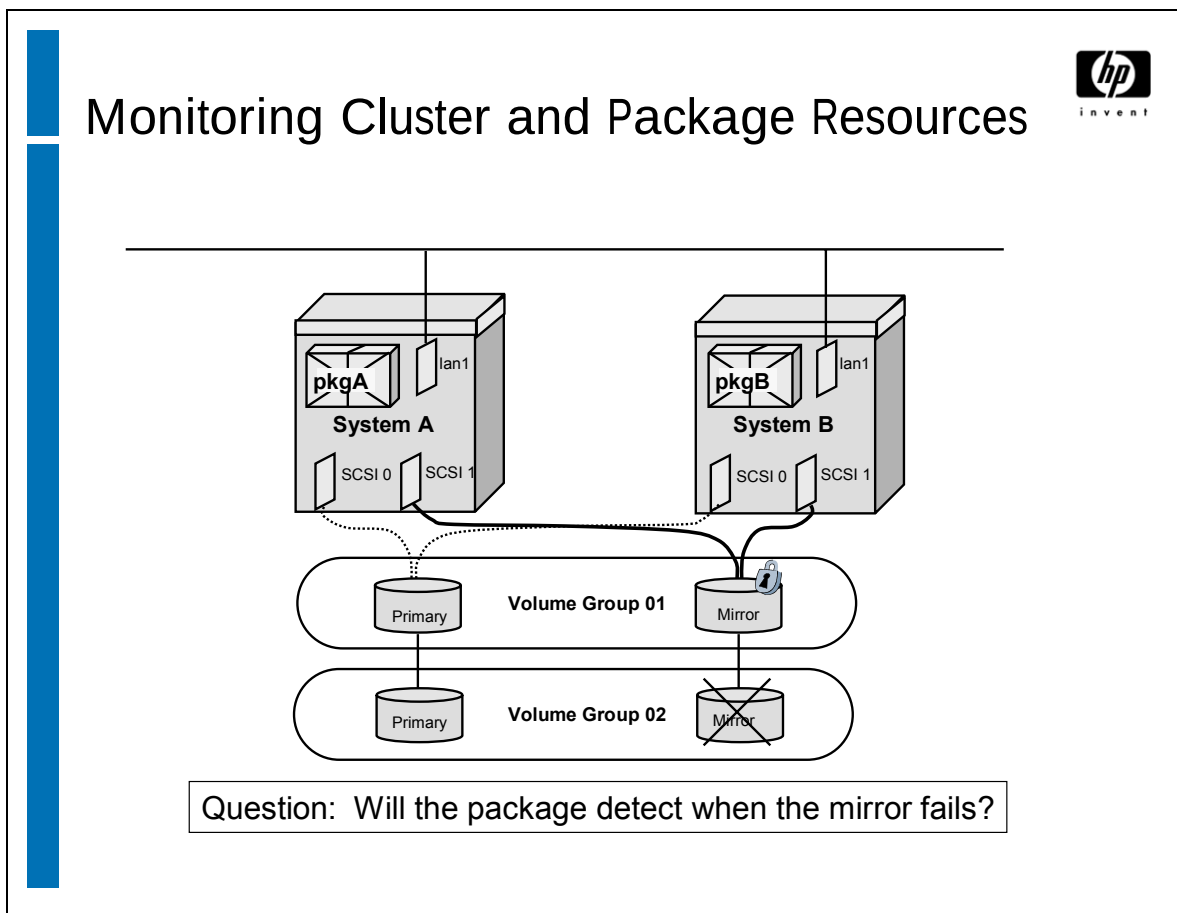
Steps to Recover a Lost Cluster Lock Disk

- Step 1: Replace the defective disk.
- Step 2: Execute `vgcfgrestore` to rebuild the LVM reserved areas on this disk.
Note: even though the lock information is stored in the LVM reserved area, the `vgcfgrestore` command does not reinitialize the lock information. The lock information is stored in the bad block relocation table. There wouldn't be any reason for `vgcfgrestore` to replace the BBRA, since it's likely we replaced the disk anyhow.
- Step 3: If the disk also has logical volumes, follow the standard procedures to either rebuild them or sync them with the mirror.
- Step 4: Serviceguard checks the lock disk on an hourly basis. After the `vgcfgrestore` command, review the syslog file of an active cluster node for not more than one hour and look for a message that the lock disk is healthy again.

NOTE: If the `vgcfgbackup` command was not done on the lock disk volume group after the cluster was formed, the following steps must take place to restore the cluster lock:

1. Stop the cluster.
2. Execute the `cmapplyconf` command on the cluster config file.
3. Restart the cluster.
4. Back up the volume group that contains the cluster lock using `vgcfgbackup`.

10-12. SLIDE: Monitoring Cluster and Package Resources



Student Notes

In general, in a high availability environment, virtually any single component can fail, and all applications should still be accessible when they do. In some cases, applications may need to restart on their adoptive nodes, in other cases they may continue on the primary.

For example, if a single disk fails, the data housed on that disk should still be available either on the LVM mirror copy or on the RAID 1 or RAID 5 LUN.

Not only is it necessary to build redundancy into the environment, but it is also necessary to know when a failure has occurred. If we do not detect a failure, and continue on the redundant device (eg. a mirror disk or standby LAN card), we expose ourselves to dual-points of failure. We must also constantly monitor for any kind of failure.

If a primary disk failure goes undetected by administrators, eventually the mirror copy will also fail, and the users will experience extended unplanned downtime. Presumably this is unacceptable, so all failures of any kind must be constantly monitored and corrected before the redundant component also fails.

Module 10
Best Practices

Items that should be monitored include:

- Disk
- Node
- Package
- LAN controllers
- Routers (and other network components)
- UPS
- Free disk space in file systems
- Performance
- Any component or resource the users need

To monitor some or all of the above involves one or more of the following:

- writing monitoring scripts,
- implementing OpenView Network Node Manager,
- implementing ITO,
- using SGMgr
- implementing EMS.

10-13. SLIDE: Notification for Package Failure



Notification for Package Failure

Objective: When package halts on primary or starts on adoptive node, notify the operations staff.

START

```
if [ $(hostname) = "adoptive" ]
then
    notify admin ( page, email, etc )
fi
```

HALT

```
if [ $(hostname) = "primary" ]
then
    notify admin ( page, email, etc )
fi
```

Student Notes

In a highly available environment it is essential that administrators are quickly notified of any failure or abnormality. A recommended practice would be to build admin notification into the start/halt scripts. If the package halts on the primary or starts on the adoptive node the administrator should be notified. The script can test for which node it's on and then page or e-mail the administrator.

Special Note

In the above slide, it is understood that we would not type in the word "adoptive", or the word "primary". Rather, we would type in the actual hostname of the machine which is the adoptive node or the actual hostname of the machine which is the primary node.

10-14. LAB: Package Monitoring with Email Notification

Directions

1. Add the code to the logprog package run / halt script so that when the package stops on the primary it mails the message:

`"logprog package has stopped on primary node"`

to root on the primary system. For maintenance purposes, it is recommended that the run / halt script be identical on both systems.

2. Add the code to the logprog package run / halt script so that when the package starts on the adoptive node it mails the message:

`"logprog package has started on adoptive node"`

to root on the adoptive system. Again, for maintenance purposes, it is recommended that the run / halt script be identical on both systems.

3. Copy the logprog.cntl and error.msg1 and error.msg2 files to node2.

4. Failover the logprog package and verify the mail messages are properly sent on both nodes.

5. Return the logprog package to the primary node.

10–15. LAB: Using the `cmset log` Command

Directions

Complete the following exercises.

1. Set the logging level of the `cmcl` daemon to a 1 for the cluster manager module. Only use a level of 1 for periodic events generated by the cluster manager. Answer "y" to the questions.
2. Continuously tail the end of the `/var/adm/syslog/syslog.log` file and note if any additional information is being added.
3. Reset the logging level to its default settings (so that the **syslog** file does not get so big that it fills up the file system).

10-16. LAB Solution: Package Monitoring with Email Notification

Directions

1. Add the code to the logprog package run / halt script so that when the package stops on the primary it mails the message:

"logprog package has stopped on primary node"

to root on the primary system. For maintenance purposes, it is recommended that the run / halt script be identical on both systems.

Answer:

- a. In the `customer_defined_halt_cmds` function, add the following code:

```
if [ $(hostname) = "primary_node_name" ]
then
    mail root < /etc/cmcluster/logprog/error.msg1
fi
```

- b. Create the file `error.msg1` and replace the `"primary_node_name"` above with your primary node name.

```
# vi error.msg1
```

logprog package has stopped on primary node

2. Add the code to the logprog package run / halt script so that when the package starts on the adoptive node it mails the message:

"logprog package has started on adoptive node"

to root on the adoptive system. Again, for maintenance purposes, it is recommended that the run / halt script be identical on both systems.

Answer:

- a. In the `customer_defined_run_cmds` function, add the following code:

```
if [ $(hostname) = "adoptive_node_name" ]
then
    mail root < /etc/cmcluster/logprog/error.msg2
fi
```

- b. Similar to the above, an `error.msg2` file needs to be created. Then replace the `"adoptive_node_name"` above with the name of your adoptive system.

```
# vi error.msg2
```

```
logprog package has started on adoptive node
```

3. Copy the `logprog.cntl` and `error.msg1` and `error.msg2` files to node2.

Answer:

```
# rcp logprog.cntl <node2>:/etc/cmcluster/logprog/.  
# rcp error.msg1 <node2>:/etc/cmcluster/logprog/.  
# rcp error.msg2 <node2>:/etc/cmcluster/logprog/.
```

4. Failover the logprog package and verify the mail messages are properly sent on both nodes.

Answer:

```
# ps -ef | grep logprog  
# kill <pid>  
# elm (on both nodes)
```

5. Return the logprog package to the primary node.

Answer:

```
# cmhaltpkg -v logprog  
# cmmodpkg -v -e -n <primary_node> logprog  
# cmmodpkg -v -e -n <adoptive_node> logprog  
# cmmodpkg -v -e logprog
```

10-17. LAB Solution: Using the `cmsetlog` command

Directions

Complete the following exercises.

1. Set the logging level of the `cmclld` daemon to a 1 for the cluster manager module. Only use a level of 1 for periodic events generated by the cluster manager. Answer "y" to the questions.

Answer:

```
# cmsetlog -M CLM -C PER 1
```

2. Continuously tail the end of the `/var/adm/syslog/syslog.log` file and note if any additional information is being added.

Answer:

```
# tail -f /var/adm/syslog/syslog.log
```

3. Reset the logging level to its default settings (so that the **syslog** file does not get so big that it fills up the file system).

Answer:

```
# cmsetlog -r
```

