

OpenVMS System Performance Management

Instructor Guide

Training



COMPAQ

OpenVMS System Performance Management

Instructor Guide

Training



COMPAQ

Notice

© 1999 Compaq Computer Corporation

COMPAQ, the Compaq logo, and the Digital logo are registered in the U.S. Patent and Trademark Office. Other product names mentioned herein may be the trademarks of their respective companies.

Confidential computer software. Valid license from Compaq or authorized sublicensor required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Compaq Computer Corporation shall not be liable for technical or editorial errors or omissions contained herein. The information in this document is subject to change without notice.

Aero, ALPHA, ALPHA AXP, AlphaServer, AlphaStation, Armada, BackPaq, COMPAQ, Compaq Insight Manager, CompaqCare logo, Counselor, DECterm, Deskpro, DIGITAL, DIGITAL logo, DIGITAL Alpha Systems, Digital Equipment Corporation, DIGITAL UNIX, DirectPlus, FASTART, Himalaya, InfoPaq, Integrity, LicensePaq, Ministation, NetFlex, NonStop, OpenVMS, PaqFax, Presario, ProLiant, ProLinea, ProSignia, QuickBack, QuickFind, Qvision, RDF, RemotePaq, RomPaq, ServerNet, SERVICenter, SmartQ, SmartStart, SmartStation, SolutionPaq, SpeedPaq, StorageWorks, Systempro/LT, Tandem, TechPaq, TruCluster, and Tru64 UNIX are registered in United States Patent and Trademark Office.

Atalla, C-Series, Expand, FOX, Guardian, iTP, Measure, Netelligent, and PointView are trademarks of Compaq Computer Corporation.

Microsoft, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

Intel, Pentium, and Intel Inside are registered trademarks and Xeon is a trademark of Intel Corporation.

UNIX is a trademark in the US and other countries, licensed exclusively through X-Open Company Ltd.

OpenVMS System Performance Management

Instructor Guide

December 1999

Module 1 — General Topics

Course Description	1-1
Course Length	1-1
Intended Audience	1-1
Course Content and Objectives	1-1
Prerequisites	1-1
Resources	1-2
Course Organization	1-3
Course Map	1-8
Course Goals	1-9
Course Nongoes	1-9
Course Conventions	1-10
Preparing to Teach	1-11
Preparing the System	1-12

Module 2 — General Performance Topics

General Performance Topics	2-1
Instructor Notes	2-1
Key Points	2-1
Time Required to Teach	2-1
Layout of Virtual Address Space	2-2
Instructor Notes	2-2
Key Points	2-3
Time Required to Teach	2-3
Image Activation and Paging	2-4
Instructor Notes	2-4
Key Points	2-5
Time Required to Teach	2-5
Interpreting Paging Metrics	2-6
Instructor Notes	2-6
Key Points	2-6
Time Required to Teach	2-6
Working Sets and Automatic Working Set Adjustment	2-7
Instructor Notes	2-7
Key Points	2-8
Time Required to Teach	2-8
Locality and Paging Performance	2-9
Instructor Notes	2-9
Key Points	2-9
Time Required to Teach	2-9
Labs	2-9

Shared Image Costs and Benefits	2-10
Instructor Notes	2-10
Key Points	2-10
Time Required to Teach.....	2-10
Modified Page List and Modified Page Writing	2-11
Instructor Notes	2-11
Key Points	2-11
Time Required to Teach.....	2-11
The Swapper and Memory Reclamation.....	2-12
Instructor Notes	2-12
Key Points	2-13
Time Required to Teach.....	2-13
System Faults	2-14
Instructor Notes	2-14
Key Points	2-14
Time Required to Teach.....	2-14
General I/O Flow	2-15
Instructor Notes	2-15
Key Points	2-15
Time Required to Teach.....	2-15
Analyzing the Effects of Fragmentation	2-16
Instructor Notes	2-16
Key Points	2-16
Time Required to Teach.....	2-16
Controller Optimizations.....	2-17
Instructor Notes	2-17
Key Points	2-17
Time Required to Teach.....	2-17
Labs.....	2-17
Files-11 ODS-2 Concepts	2-18
Instructor Notes	2-18
Key Points	2-18
Time Required to Teach.....	2-18
Contiguity and the File System	2-19
Instructor Notes	2-19
Key Points	2-19
Time Required to Teach.....	2-19
File System Caches.....	2-20
Instructor Notes	2-20
Key Points	2-20
Time Required to Teach.....	2-20
RMS Structures and Design Considerations	2-21
Instructor Notes	2-21
Key Points	2-22
Time Required to Teach.....	2-22

RMS Utilities.....	2-23
Instructor Notes	2-23
Key Points	2-23
Time Required to Teach.....	2-23
Labs.....	2-23
RMS I/O Buffering Considerations	2-24
Instructor Notes	2-24
Key Points	2-24
Time Required to Teach.....	2-24
MONITOR RMS	2-25
Instructor Notes	2-25
Key Points	2-25
Time Required to Teach.....	2-25
Understanding CPU Performance	2-26
Instructor Notes	2-26
Key Points	2-27
Time Required to Teach.....	2-27
Understanding Time Spent in Modes	2-28
Instructor Notes	2-28
Key Points	2-28
Time Required to Teach.....	2-28
OpenVMS Scheduling and Priorities.....	2-29
Instructor Notes	2-29
Key Points	2-29
Time Required to Teach.....	2-29
PIXSCAN and DORMANTWAIT	2-30
Instructor Notes	2-30
Key Points	2-30
Time Required to Teach.....	2-30
Labs.....	2-30
Locking Concepts	2-31
Instructor Notes	2-31
Key Points	2-31
Time Required to Teach.....	2-31
Distributed Locking Concepts.....	2-32
Instructor Notes	2-32
Key Points	2-32
Time Required to Teach.....	2-32
MSCP Server	2-33
Instructor Notes	2-33
Key Points	2-33
Time Required to Teach.....	2-33

Appendices

- Appendix A — Tailoring OpenVMS
- Appendix B — Virtual Balance Slots
- Appendix C — Fast I/O

Course Description

Course Length

4.5 days

Intended Audience

OpenVMS system managers, applications programmers, and system programmers.

Course Content and Objectives

This course intends to present OpenVMS performance under three subsystems: Memory Management, I/O, and CPU. The students will learn to isolate performance bottlenecks to one of the three subsystems. Students will learn the effects of SYSGEN parameters on each of the subsystems, when to change these parameters, and when other management modifications or the purchase of new hardware will be required. Students will learn sufficiency (keeping the system running), as well as tuning considerations. The relative merits/drawbacks of using AUTOGEN, will also be described.

Prerequisites

Students should have a fundamental understanding of OpenVMS System Management, including:

- How to set up accounts.
- How to use SYSGEN/SYSMAN to change a system parameter.
- How to run AUTOGEN.
- How to use the INSTALL utility.
- How to INITIALIZE and MOUNT a disk.

Students should have a fundamental understanding of programming concepts, including:

- Understanding the purpose of a compiler.
- Understanding the purpose of a linker.
- Generally understanding how a computer executes a program.

Resources

Students should receive:

- An OpenVMS Performance Student Guide.
- An OpenVMS Performance Labs and Solutions Handout. These may be separate.
- Students should have access (or possibly receive) to:
 - The OpenVMS Performance Guide.
 - A lab system, running OpenVMS.

Course Organization

- General Performance Topics
 - Performance Issues
 - Performance Bottlenecks
 - OpenVMS Performance Tools
- Layout of Virtual Address Space
 - Layout of Virtual Address Space
 - P0 Space
 - P1 Space
 - S0/S1/S2 Space
- Image Activation and Paging
 - Page Faults
 - Producing an Image File
 - Image Activation and Paging
 - Valid Pages and the Free and Modified Page Lists
 - The Free List Fault Rate
 - The Modified List Fault Rate
 - Secondary Hard Faults
- Interpreting Paging Metrics
 - Sample Display from \$MONITOR Page
 - Interpreting the \$MONITOR Page Display
 - Interpreting Paging Metrics
- Working Sets and Automatic Working Set Adjustment
 - The Working Set List
 - Working Set Size vs. Working Set List Size
 - Parameters Affecting the Working Set List Size
 - Working Set List Parameter Relationships
 - Automatic Working Set Adjustment
 - AWSA Mechanics
 - Working Set Performance Considerations
 - AWSA Performance Considerations

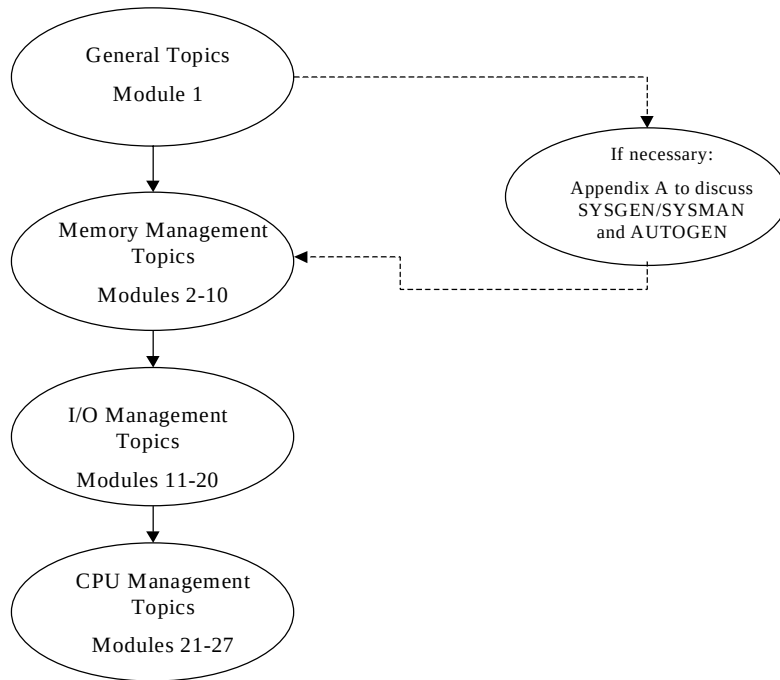
- Locality and Paging Performance
 - Locality
 - Locality Issues
 - Effects of Poor Locality
 - Effect of Improving Locality
- Shared Image Costs and Benefits
 - Shared Image Costs and Benefits
 - Shared and Other Install Benefits
 - Image Activation and Paging
 - Shared Image Benefits
 - Resident Images
 - Shared Address Data
- Modified Page List and Modified Page Writing
 - Modified Page List
 - Modified Page List Parameters and Metrics
 - Modified Page List Performance Considerations
 - MPL Performance and Analysis
- The Swapper and Memory Reclamation
- System Faults
- General I/O Flow
 - General I/O Flow
 - Geometry of Disks
 - Time-Based Components of a Disk Transfer
 - Optimization of Disk Transfers
 - Virtual I/O Cache
 - RMS File Concepts
- Analyzing the Effects of Fragmentation
 - Fragmentation
 - Spindle Contention

- Controller Optimizations
 - Controller Optimizations
 - Locating Bottlenecks
 - Other Optimizations Through Layered Products
 - Volume Shadowing
 - Disk Striping
 - RAID
- Files-11 ODS-2 Concepts
 - Files-11 Terminology
 - Directory Concepts
 - Files-11 Reserved Files
 - File Open Operation
- Contiguity and the File System
 - Window Turns
 - Split Transfers
- File System Caches
- RMS Structures and Design Considerations
 - Structure for Record Organizations
 - File Organizations
 - Sequential and Relative File Structure
 - Indexed File Organization
 - Indexed File Structure Descriptions
 - Structures Supporting Alternate Keys
 - RMS Interfaces for Affecting File Structure
 - Factors Affecting Bucket Splits
 - Duplicate Keys for Data Records
 - Compression
 - Alignment Considerations

- RMS Utilities
 - File Definition Language
 - Techniques for Creating Data Files Using an FDL File
 - Creating a Data File from an FDL File
 - Optimizing RMS Indexed Files
- RMS I/O Buffering Considerations
 - Buffers and Buckets
 - Read Ahead
 - Write Behind
 - Deferred Write
 - Setting Up Multiple Buffers
 - Determining the Number of Local Buffers
- MONITOR RMS
- Understanding CPU Performance
 - Understanding CPU Ratings
 - CPU Performance and Symmetric Multiprocessing
- Understanding Time Spent in Modes
 - Kernel Mode
 - Executive Mode
 - Supervisor Mode
 - User Mode
 - Interrupt Stack/State Time
 - MP Synchronization Time
 - Idle Time
 - Compatibility Mode Time
 - Analyzing Time Spent in Modes
- OpenVMS Scheduling and Priorities
 - Quantum and CPU Time
 - Priorities
 - States of Concern

- PIXSCAN and DORMANTWAIT
 - PIXSCAN
 - DORMANTWAIT
- Locking Concepts
 - Locking Concepts
 - Other Characteristics of Locks
 - Application Design and Locking Performance
 - Interpreting the \$MONITOR LOCK Display
- Distributed Locking Concepts
 - Distributed Locking Concepts
 - The LOCKDIRWT SYSGEN Parameter
 - Performance Issues with Distributed Locking
- MSCP Server
 - MSCP Server
 - Interpreting \$MONITOR MSCP

Course Map



Course Goals

- Students will be able to interpret performance metrics supplied by:
 - The MONITOR utility
 - DCL Commands including:
 - ◆ \$SHOW SYSTEM
 - ◆ \$SHOW STATUS
 - ◆ \$SHOW MEMORY
 - The INSTALL utility
- Students will be able to identify bottlenecks in the following subsystems:
 - Memory Management
 - I/O
 - CPU
- Students will be able to size working set parameters.
- Students will be able to monitor page file space.
- Students will be able to describe the advantages of balancing disk I/O.
- Students will be able to improve the performance characteristics of indexed files using the CONVERT utility.
- Students will be able to identify factors contributing to time spent executing in processor modes.
- Students will be able to describe file locking design considerations to attain good performance.
- Students will be able to monitor distributed lock traffic and impact its performance.
- Students will be able to identify the impact of changing key system parameters.

Course Nongoals

- Network performance
- Finding a magic system parameter that will make your system run faster.

Course Conventions

In general text:

- Arial font identifies descriptive text.
- **Courier New Bold** identifies commands.

In system displays/examples/screen dumps:

- Courier New identifies system display.
- **Courier New Bold** identifies user input.

Preparing to Teach

This course covers an awesome amount of material. The student guide is fairly large, but consolidates a lot of information that has been gathered over 20 years. Bear in mind, many approaches to this material are filled with almost mystical interpretations of performance metrics. Your goal in preparing for delivery of this course should be to develop a fundamental understanding of how OpenVMS works and apply common sense to interpreting performance data.

You should first read the *OpenVMS Performance Management* manual. The manual takes, for the most part, a surface view of OpenVMS principles of operation. It tends to try to generalize the internals. In the process, its technical content is sometimes inaccurate. Take the information in the manual with a grain of salt.

This is not an internals course, and it should not be taught as one. However, to really understand the performance metrics, an understanding of internals is invaluable. I would strongly recommend that you at least skim through the memory management discussion in Ruth Goldenberg's *Internals and Data Structures* manuals. Read both the VAX and Alpha discussions, because the two systems handle memory management a little differently. The discussion in the *OpenVMS AXP Internals and Data Structures* manual is a little dated (V1.5). The full 64-bit capabilities (P2, PT, and S2 space) did not come out until V7.0.

The most recent discussion of 64-bit address space will be found in the *OpenVMS Alpha Guide to 64-bit Addressing and VLM Features* manual.

It might be a good idea to take an internals course that covers memory management.

It would be useful to examine the *VMS File System Internals* book and the *Guide to File Applications* manual for I/O related information.

Walk through the student guide and try the examples. As you see how the numbers change, you will develop a firmer foundation for what they are telling you. Compare and contrast the material in the student guide against the Performance Management manual. They differ on some points.

Analyze the examples. A picture is worth a thousand words. An example that shows 20%, 50%, or 200% improvement in performance may be worth a million.

When you teach the course try, to get the students into lab by 2:00 to 2:30 every day. When they play with the system, they should get a better feel for the impact of the numbers.

Preparing the System

It is best if the students have a workstation each. Otherwise, the lab results of one student may affect those of another.

The system should be running the most current version of OpenVMS and should have a C compiler installed and licensed.

If you reproduce the files listed at the end of the Labs Module(/handout?) using the same names listed, the BUILD.COM command procedure will build the programs and create the PERF\$ logical name.

Student accounts should be fully privileged.

General Performance Topics

Module 2

General Performance Topics

Instructor Notes

This module should act as a foundation for later interpretation of performance metrics. This is your chance to discuss the potential areas of contention for system resource. Make the point that the student should not be looking for a "magic" SYSGEN parameter to improve performance. Managing system resources, process parameters (like working sets), and applications will get the biggest bang for the buck. The students should understand that there may be a point where the only effective method for improving is BUY NEW HARDWARE. However, they should understand what hardware to buy.

Arrival rate numbers tell the number of work requests you are performing per unit of time. An example of this is "I/Os per second". One problem with the arrival rate numbers is that they do not give you the whole picture. It takes a lot longer to process a 100 block disk I/O than a 1 block I/O. Service time numbers tell you how long it took to process a request. OpenVMS is not always great at telling you about service time. The "Queue depth" numbers from MONITOR DISK tell you about service time.

Feel out the students. If they have no experience with SYSGEN/SYSMAN and AUTOGEN you may need to take a short detour, or point them, to Appendix A.

Key Points

- The students should understand that performance includes:
 - Sufficiency: making sure the system runs.
 - Efficiency: not wasting resource.
 - Effectiveness: responsiveness to users.
- The students should have an appreciation for scale. CPU related activities are normally hundreds, to thousands, of times faster than going to disk.

Time Required to Teach

This module should take about 15-20 minutes to cover.

Layout of Virtual Address Space

Instructor Notes

This module serves two purposes:

- It provides an overview of the layout of OpenVMS virtual address space. This will be important for future paging discussions.
- It addresses sufficiency-related parameters.

P0, P1, and S0/S1 address spaces are common between VAX and Alpha systems. Alpha, typically, uses 32-bit, sign-extended pointers to access these address spaces. It is important that students understand P0 space comes and goes with most DCL commands and will have the biggest impact on paging performance.

P2 space also comes and goes with the programs and will probably have more impact on paging in the future. P2 space contains only data for programs and normally, must be requested. You use P2 space in a C program when you compile with long (64-bit) pointers.

This module focuses on one key, sufficiency related aspect of the system: non-paged pool. It is important that students appreciate that non-paged pool is a subset of system space. Students sometimes get the idea that all of OpenVMS is in non-paged pool.

Certain situations can cause pool to become so fragmented, through the lookaside lists, that it will eventually crash. An example of this behavior occurs with Pathworks. A PC can throw a large number of messages at the OpenVMS host that chew up the 64 byte lookaside list. When a chunk of pool is requested, the variable pool is exhausted. Aggressive reclamation then kicks in. If there are thousands of 64 byte packets, it may take a long time to deallocate the entire list and repopulate it. This time may cause spinwait timeouts in an SMP environment. If this is a problem, you might consider setting NPAG_AGGRESSIVE to 100 to allow the pool allocation routines to simply expand pool. This tends to be a little better behavior than crashing the system.

You may want to skim through the actual contents of the address spaces. The student guide contains a fairly detailed description of each of the key areas. The students can read this information on their own.

Processes need process headers to run. BALSETCNT identifies how many process headers can be memory resident simultaneously. If it is too small, you may experience artificial swapping. This swapping is artificial, because you have enough memory, but have run into an artificial restriction. There are not enough balance set slots to hold these process headers.

MAXPROCESSCNT defines the number of process entry slots on the system, effectively the maximum number of processes that can be running concurrently. Two process entry slots do not use balance set slots, the idle code (NULL pseudo-process) and the SWAPPER. If BALSETCNT is set equal to MAXPROCESSCNT minus 2 (the normal AUTOGEN default value), artificial swapping cannot happen.

Key Points

- Programs are mapped into P0 space most times that a DCL command is entered.
- Program execution causes most of your page faulting.
- Each process has its own P0 space (and P1, P2, and PT).
- Some of system space is pageable, like Paged Pool and RMS.
- Non-page pool expansion has occurred if \$SHOW MEMORY/POOL/FULL shows that the "current size" of pool is greater than the "initial size". If this is the case, you should consider raising the minimum value for the SYSGEN parameter NPAGEDYN.
- BALSETCNT defines the number of process headers that may be memory resident simultaneously.
- If BALSETCNT is too small, artificial swapping will occur.
- CTLPAGES is the size of the process allocation region.
- If CTLPAGES is too small, you may be blocked from creating process logical names or running programs linked against lots of shareable images.
- If PAGEDYN is too small, you will be blocked from creating system/group/shared logical names, among other things.
- If CHANNELCNT is too small, programs may not be able to open large numbers of files.
- NPAG_AGGRESSIVE set to 100 will cause pool to expand if it is fragmented.

Time Required to Teach

This module should take 30 to 45 minutes to present.

Image Activation and Paging

Instructor Notes

This is a pivotal module for understanding paging metrics. It is important for the students to get an appreciation that page faults are required. You cannot run programs without them. It is very likely that students may not really understand what function virtual addresses serve. A review may be in order.

FYI: programs are represented in machine code. On the Alpha, all instructions are 4 bytes in size. When the program is activated, the Program Counter (PC) is filled in with an address, via a Jump to Subroutine (JSR) instruction. The PC is updated to contain the address of the next instruction. That instruction will be 4 bytes away, unless a jump or branch instruction is issued.

For example:

Address	Instruction
20000	Macro instruction 1
20004	Macro instruction 2
20008	Macro instruction 3

Address translation happens at least once, possibly twice, every time you execute an instruction.

The address translation is first done by the CPU, using a Translation Buffer (or address translation cache). If the address is not in the translation buffer, the PAL code walks through the conceptual translation steps in the first diagram in the module. If the page is valid, the PAL code completes the translation. If not, the PAL code generates an exception (page fault). The VAX does address translation in similar fashion, except that microcode does the translation.

The details of address translation are very different between the VAX and Alpha processor (see the Internals manual for details). The concept is the same. Consider that a page table is an array of entries that identify where the page is currently. The Virtual Page Number (VPN) is used as an index into the array. The Virtual Page Number is computed, using high bits in the virtual address.

The second diagram is admittedly very ugly. However, it defines the essence of how memory management works. Do not worry too much about the details of the diagram, at this point. The rest of the module describes how we get to this point. The important points, when you first hit the diagram:

Each process has its own page table.

The page table entries can describe pages that:

- We can get at directly (they are valid).
- Require a soft fault.
- Require a hard fault.

The image file discussion is there to lay foundations for discussing the SYSGEN parameter PROCSECTCNT and to describe how page table entries are built at image activation. Important in the image activation process is that the program is NOT directly loaded into memory at image activation. It is loaded by the pager, on demand.

There are 4 fundamental phases in the life of a page:

Phase	Type of Fault	Quality
Image Activation	Faults may be hard (Read I/Os) or soft (Demand Zero, Global Valid, or Free List)	Required
The page is valid	Not a fault	N/A
The page has been displaced from the working set list	Faults are soft (Free or Modified List)	Indicate the working set list may be too small
The page has been taken by another process	Faults are hard (Read I/Os)	Indicate you are running out of memory

Page 3-18 references pages "written back to image file"

The reference here is to the possibility of writable global sections. Although, these may be rare they are a possibility I did not want to dismiss. Although you are probably acquainted with these concepts, please see the *OpenVMS Programming Concepts Manual* Section 10.3.4 on sharing data, the *OpenVMS Internals and Data Structures* manual, or the discussion of the SYS\$CRMPSC system service in the *OpenVMS System Services Reference Manual* for more details.

Key Points

- Page faults are required.
- Hard page faults occurring with a high Demand Zero fault rate are probably due to image activation.
- Hard page faults occurring with a small Free Page List Size are due to memory exhaustion.
- High Free and Modified List Fault Rates may indicate that working set lists are too small.
- The quality of the page fault is more important than the quantity.

Time Required to Teach

This module will probably take 60 to 90 minutes to cover.

Interpreting Paging Metrics

Instructor Notes

This module is your chance to apply and review the material that you covered in the previous module. I would recommend that you walk through the fields in MONITOR PAGE and try to get the students to identify what they mean, helping them along. Let them read the descriptions on their own.

The next 2 examples are interesting, because they show that the page fault rate is going down, but performance is degrading. The degradation is visible through the Read I/Os rate increasing.

Note on examples on 4-10 to 4-12:

The intent is to move the students away from striving toward meaningless

“Rules of Thumb” and toward a more pragmatic and empirical view of performance metrics. The common mistake of students is attempting to determine a “High” page fault rate. While 31055 (page 4-11 top) sounds high, especially to an old “VAXmaster” that goes by the rule of thumb of 100 faults per second is high, what does this really mean? Well, it translates to kernel mode CPU time, relative to the speed of the CPU. Note, however, the displays that follow have a continually dropping number of total faults (~28600 to ~22800 to ~140). The number of “hard” faults, at the same time is increasing (19 to ~23 to 35). As the hard faults increase, the disk becomes the bottleneck. This is a classic example of “thrashing”.

Again, the point here is that although the numbers at the bottom of 4-12 may appear “better”, due to their lower values, users are experiencing response time degradation.

Key Points

- Read I/Os are hard faults.
- Soft faults would be Fault Rate minus the Read I/O rate.
- Free and Modified List Faults are soft, but not necessarily good.
- The Fault Rate is almost meaningless without looking at the quality of the faults.

Time Required to Teach

This module should take about 15 to 20 minutes to teach.

Working Sets and Automatic Working Set Adjustment

Instructor Notes

The discussion in this module diverges somewhat from the recommendations in the Performance Management Manual. The working set list is a limit. It is intended to block processes from grabbing inordinate amounts of memory and causing the system to thrash about doing more work than is necessary to get its work done. The recommendation for working set sizes is more conservative than that in the performance guide. The philosophy here is simple. Spread the wealth equally amongst the processes. By setting the QUOTA to the amount of available memory divided by the maximum number of processes you should not run out of memory. Setting a higher EXTENT, 2 to 4 times the QUOTA allows for growth under varying workloads.

Note, that it is very common for the PQL_M parameters to be set higher than their corresponding values in the User Authorization File. This tends to void the work done on the UAF. You can get around this by lowering the PQL_M values.

AUTOGEN tends to set BORROWLIM very low. This is not a problem unless you are in a memory tight environment. In which case, you will find that you are giving out loans and taking them back quickly. It is probably not a bad idea to recommend raising GROWLIM and BORROWLIM. Setting BORROWLIM to twice your EXTENT sizes means that a single process will not block others from obtaining loans. Setting GROWLIM to half BORROWLIM will allow processes that have received loans to use them. We will be discussing the tickler and troller later on, but you should probably set FREEGOAL to a value less than GROWLIM.

The tuning session is a good example of methods of receiving optimal gain. After all the work of tuning we saw response time drop from 43 to 38 seconds. This is about a 15% improvement in response time. (Response time is an important measure because of the hard faults incurred here.) Increasing memory from 32MB to 128MB yielded a response time of about 4 seconds.

The TBSKIPWSL scheme takes place on any processor that supports the TBCHK internal processor register. This should include most current processors.

Key Points

- The WSQUOTA should be set based on available memory (supply) and maximum number of concurrent processes (demand).
- Use a higher WSEXTENT to handle varying workloads.
- The WSDEFAULT should be set to the WSQUOTA.
- Consider raising BORROWLIM.
- Use the memory you do have. If you are incurring lots of Free and Modified Page list faults and the Free Page List size is high, increase your working set list sizes.
- If you are incurring image activation related faults, larger working set list sizes will not help.
- If you are exhausting memory, you want to lower, NOT raise, working set list sizes.

Time Required to Teach

This module should take about 60 to 90 minutes to cover.

Locality and Paging Performance

Instructor Notes

This module presents the classic problem that a little mistake in a program can haunt the system manager. Note, the memory demand required by the poorly written program is 30 times higher than the properly written program!

The C compiler is, now, smart enough to optimize away code that does not do anything with the data it processes. The example in the labs shows how the optimizations may be defeated.

Key Points

- Know your languages when you use multi-dimensional arrays.
- The largest gains in performance are, commonly, obtained through tuning applications.

Time Required to Teach

This module should take about 15 to 20 minutes to cover.

Labs

You should be ready for labs, either just before, or after this module.

Shared Image Costs and Benefits

Instructor Notes

This is a place where students can obtain real gains in image activation performance.

The costs for unused GBLPAGES and GBLSECTIONS are fairly insignificant. The gains are there even if the images are not run concurrently. Any image that is run frequently should at least be installed with the /SHARE qualifier. Note, you will actually get more page faults with images installed /SHARE. On a soft page fault, only one page gets made valid. On a hard fault, up to 4 Alpha pages and 64 VAX pages may be made valid.

Installing images /RESIDENT will actually eliminate page faults. The cost here is much greater, as unused pages will not be available on the free list. Resident images are only available on Alpha platforms.

The discussion on shareable images is included for programmers. If you do not have any programmers in the class, you can mention the pages and skip over them.

Shared address data is new to V7.2 and is only available on Alpha systems. It allows pages that contain addresses that need to be fixed up, but otherwise would be read-only, to be shared. This eliminates the CPU time for fixup operations and turns the normal hard fault into a soft fault.

See the Internals manual for a discussion of fixup operations.

Key Points

- The fastest image activation will be obtained by installing images with the /RESIDENT/SHARED=DATA/HEADER_RESIDENT qualifiers.
- Using the /SHARED=DATA/HEADER_RESIDENT can be close in performance at a much smaller cost.
- Do not tweak down GBLPAGES. You do not get much gain and may break applications that use the SYS\$CRMPSC system service.
- For images to be installed /RESIDENT, they must be linked with the /SECTION_BINDING and /NOTRACEBACK qualifiers.
- Pages reserved with the GH_RSRVPGCNT SYSGEN parameter are not available for paging.
- Installing images has no impact on page faults associated with accessing writable data.

Time Required to Teach

This module should take about 60 to 90 minutes to cover, dependent on whether you include a discussion of shareable images.

Modified Page List and Modified Page Writing

Instructor Notes

This is a quick tour of modified page writing. AUTOGEN normally sets the modified page list parameters just fine. Larger working sets are in order if the Modified List Fault rate is high.

Running out of page file space can hang processes and effectively hang the system.

Key Points

- If you notice temporary hangs on the system, check page file usage.

Time Required to Teach

This module should take about 15 to 20 minutes to cover.

The Swapper and Memory Reclamation

Instructor Notes

The SWAPPER wakes up for three memory reclamation reasons:

- The free page list has dropped below FREELIM pages in size.
 - Trim processes back to WSQUOTA/SWPOUTPGCNT and possibly SWAP.
- The free page list has fallen below FREEGOAL*2 pages.
 - Tickle. Trim infrequently running process' working sets (but not list sizes) to SWPOUTPGCNT.
- The free page list has fallen below FREEGOAL pages in size.
 - Troll. Swap out idle processes.
- If the first case happens, system performance has severely degraded. Ticking and Trolling are much more common reasons for swapping to occur.

The diagram on relative relationships of free page list parameters on page 9-9 may appear to contradict the settings normally made by AUTOGEN with relation to these parameters.

Please consider:

BORROWLIM defines the minimum boundary of pages that must be on the Free page list, before loans will be granted to a process' working set list size.

GROWLIM defines the minimum boundary of pages that can be on the Free page list, before a process' working set can use available working set list entries, when the list size is above its quota. In effect, this limits the process' ability to use loans when memory has become tight.

The “troller” kicks in when the Free page list falls below 2*FREEGOAL.

The “ticker” kicks in when the Free page list falls below FREEGOAL.

By using the recommendations that AUTOGEN makes, you are forcing a thrashing behavior when memory becomes tight, i.e. you are giving out loans at the same time you are reclaiming memory. This appears to be the method that they used to run “Savings and Loans” in Ohio, but no way to “run a bank.

It is actually of little issue if memory is not tight. Having run on low memory systems most of my technical life, the rational approach seems, to me (and nobody has come up with a convincing argument to the contrary, including OpenVMS engineering) to be following the guidelines on page 9-9.

Perhaps you are incredulous that AUTOGEN could be wrong here. All I can say is that I have illustrated this point on numerous occasions to students in lab. The difference in response time is noticeable. The difference in throughput is illustrated in the examples on pages 5-19 (AUTOGEN's settings and presumably your recommendation) and the sample run on page 5-26. Please note the parameter changes shown on page 5-28. These were conservative changes and FREEGOAL should probably have been dropped more to between 450 and 500, based on a BORROWLIM setting of 1000.

I never "slam" AUTOGEN and believe that it gets 95+% of its settings adequate, but it seems that here they missed the boat.

Of course, if memory is not tight this is not an issue, assuming these parameters are not set overly large.

Key Points

- To minimize ticking, raise BORROWLIM to more than twice FREEGOAL.
- To minimize trolling, raise BORROWLIM to greater than FREEGOAL.
- Paging is more desirable than swapping.
- Outswapped processes are a good indication that you need more memory.

Time Required to Teach

This module should take 10 to 15 minutes to teach.

System Faults

Instructor Notes

System faults occur when you touch pageable system space that is not valid. They are not inherently bad. The reason they are differentiated, from other faults, is the system's working set is sized based on a separate SYSGEN parameter, SYSMWCNT.

To determine how many total and hard system faults since the last boot use:

```
$ ANA/SYS
OpenVMS (TM) Alpha system analyzer

SDA> read sysdef
%SDA-I-READSYM, 9194 symbols read from SYS$COMMON:[SYS$LDR]SYSDEF.STB;1
SDA> ex @mmg$gl_sysphd+phd$l_pageflts !Total faults
FFFFFFFF.80D00214: 0000012A  "*..."
SDA> ex @mmg$gl_sysphd+phd$l_pgfltio !Hard faults
FFFFFFFF.80D00290: 00000000.000000A8  "....."
SDA> ev 12a
Hex = 00000000.0000012A  Decimal = 298
SDA> ev a8
Hex = 00000000.000000A8  Decimal = 168          ADP$PS_SHUTDOWN
                                                    BUG$_BADSWPPAG
                                                    CHF$FH_MCH_SAVF0
                                                    CPU$Q_SYS_ASN
                                                    FCB$L_GRP_PROT
                                                    |
                                                    (remaining symbols suppressed by
default
)
SDA>
```

Key Points

If the system fault rate is more than 1 or 2 a second, consider increasing SYSMWCNT.

Time Required to Teach

This module should take 5 to 10 minutes to cover.

General I/O Flow

Instructor Notes

This module should identify the components of I/O. It acts as an overview. The focus is on disk I/O, because you tend to have the most control over it. Note that the CPU time associated with I/O is fairly insignificant. Seeking and rotational are the most costly components of I/O.

The Virtual I/O cache may skew your I/O metrics. The Virtual I/O cache will not cache files open for write access in a VMSccluster.

Key Points

- Seeks are the most expensive component of a disk transfer.
- Seeks can happen in parallel on separate disks.
- Rotational latency is the next most expensive component.
- You can minimize the impact of these costs by spreading your I/Os across multiple spindles (disks).

Time Required to Teach

This module should take about 15 to 20 minutes to teach.

Analyzing the Effects of Fragmentation

Instructor Notes

This module is nice because it shows students the impact of seeking and rotational latency in a straight forward fashion. Improvement is seen by spreading disk I/Os across disks.

Note, that the direct I/Os in the first example do not match the I/O operations. The discrepancy is due to the split transfers, which are not counted as direct I/Os.

Note, that, by the time we are done with the example, throughput has more than doubled.

Key Points

- Fragmentation costs extra seek and rotational latency time.
- Spreading I/Os across spindles allows for simultaneous seeking and better response time.
- These improvements do not usually require application changes.

Time Required to Teach

This module should take about 20 to 30 minutes to cover.

Controller Optimizations

Instructor Notes

While controller optimizations help throughput, for response time they may be too late. The analogy has been made that the HSx optimizations are similar to rearranging the deck chairs on the Titanic. By the time the queue fills up response time has degraded.

It is not always possible, or easy, to identify "hot" files (files with a lot of I/O activity) and move them to "cold" disks (disks with minimal activity). Striping makes for a nice alternative.

Key Points

- Controllers and drivers attempt to minimize seek overhead on very active disks.
- Striping allows I/Os to be, transparently, spread across disks.

Time Required to Teach

This module should take about 10 to 15 minutes to cover.

Labs

This is probably a good point to break for the day 2 labs.

Files-11 ODS-2 Concepts

Instructor Notes

This module introduces the student to Files-11 concepts. The intent is to give the student a better understanding of the file system caches.

For more information on Files-11, check the *VMS File System Internals* book.

Key Points

- File opens are somewhat expensive.
- When a file is open, it should remain open until no longer needed.
- File system activity can be tracked through \$MONITOR FCP.

Time Required to Teach

This module should take about 20 minutes to cover.

Contiguity and the File System

Instructor Notes

This module revisits the effect of fragmentation.

It points out that Window turns are more likely on randomly accessed files, like indexed files. Split transfers are more likely on sequentially accessed files.

Window turns may be minimized by larger Window Control Blocks (WCBs), but this ignored the real problem associated with fragmented files, seeks and rotational latency.

Key Points

- Window turns tend to occur on randomly accessed files.
- Split transfers tend to occur on sequentially accessed files.

Time Required to Teach

This module should take about 15 to 20 minutes to cover.

File System Caches

Instructor Notes

Typically, the file system caches are sized properly.

Note, these caches are used by the file system (XQP) only.

A historical note, the ACP prefix comes from the Ancillary Control Process that was replaced, in Version 4.0, by the eXtended Qio Procedures (XQP).

One problem can occur if the cache sizes are made too large. If there is insufficient contiguous paged pool (sized by PAGEDYN), the shared caches will be shrunk. These caches include ACP_DINDXCACHE, ACP_DIRCACHE, ACP_HDRCACHE, and ACP_MAPCACHE. Shrunk caches can be very disastrous to performance. If the caches are shrunk, they will be set very small. The maximum buffers in FCP cache will normally be 14.

The Bitmap cache is not touched frequently, because blocks are allocated from the Extent cache.

Key Points

- The file system caches may be monitored, using \$ MONITOR FILE_SYSTEM_CACHES.
- A goal, over time, should be a 90%, or better, cache hit rate.
- If you raise the size of the shared caches (ACP_MAPCACHE, ACP_DINDXCACHE, ACP_DIRCACHE, and ACP_HDRCACHE), raise PAGEDYN accordingly.

Time Required to Teach

This module should take about 20 minutes to cover.

RMS Structures and Design Considerations

Instructor Notes

You will want to review the *Guide to File Applications* prior to teaching this module.

The indexed file structure is pretty complicated. Note, not all keys are stored in the index buckets. Only the key for the record, with the highest key value in the data bucket to which it is pointing, is in the index bucket.

Bucket splits are caused by attempting to add records to full buckets. Buckets are costly for sequential access, because the split bucket is normally allocated from the end of the file. This means that you will access a block, then have to seek toward the end of the file, and back again. Bucket splits are also costly for secondary key access, because the Secondary Index Data Record is not updated with the new bucket location of the split records. This causes extra I/O.

Many bucket splits can cause more index records to be created. This, in turn, causes more index buckets and eventually more levels in the index file structure. More levels translates to more I/Os.

The best way to populate an indexed file is in ascending order of primary key.

Many additions over time, spread across key values, will cause bucket splits.

The Fill Factor is only used at creation time. Once a bucket is filled to a percentage greater than the fill factor, the next record will be placed into a new bucket. The smaller the Fill Factor, the more reserved space. This may minimize bucket splits. However, it may cause the number of levels to increase. This increase will occur because there will, likely, be more data buckets, hence more index records. A smaller fill factor should only be used if you can guarantee that additions will be spread evenly over key values.

Turning off index compression, allows binary searches on the key lookups. Unless you have very large keys, you should turn this feature off.

Key Points

- Buckets are allocation units.
- Buffers hold records.
- Bucket splits are caused by many record additions or poor population techniques.
- Bucket splits degrade the performance of sequential and secondary key access.
- Bucket splits may cause more levels.
- Use the CONVERT utility to populate indexed files.
- Use the CONVERT utility to reorganize indexed files that have many additions.

Time Required to Teach

This module should take 45 to 60 minutes to cover.

RMS Utilities

Instructor Notes

Students should be encouraged to use FDL files, in conjunction with the convert utility, to create indexed files.

The convert utility will improve performance characteristics of indexed files, even without the optimized FDL file, because the file will be repopulated naturally.

A lower data fill percentage in a file that has been created with a Fill factor of 100 is a good indication that you have bucket splits in a file.

Key Points

- \$CONVERT/RECLAIM does not normally improve the performance of indexed files. It does allow you to reuse empty buckets, caused by deletions.
- \$CONVERT by itself will improve the performance of indexed files. This is due to proper population techniques.
- \$CONVERT/FDL with an optimized FDL file will yield the best performance for indexed files.

Time Required to Teach

This module should take 30 to 45 minutes to teach.

Labs

This is probably a good point to break off for day 3 labs.

RMS I/O Buffering Considerations

Instructor Notes

One way to use extra memory is to increase the read buffer sizes for RMS files. This is useful for sequential files and easy to implement. Just increase the multi-block count.

Reading more data per I/O, means that you will experience less overhead for rotational latency and seeking.

Multiple buffers are of most advantage in indexed files. The more buffers, the more data you cache, particularly index records. More cached data translates to less I/Os. The default number of buffers for an indexed file is 2!

Global buffers are potentially very powerful. They may incur additional overhead in a VMScluster environment, due to locking. If you use global buffers, you may need to increase RMS_GBLBUFQUO and GBLPAGFIL.

Key Points

- Larger buffers will increase sequential file throughput.
- More buffers will improve indexed file access, by eliminating some I/Os associated with accessing index records.

Time Required to Teach

This module should take about 30 to 45 minutes to teach.

MONITOR RMS

Instructor Notes

There is not a large overhead for enabling RMS statistics.

The CACHING option is probably the most useful display.

Key Points

- Use \$MONITOR RMS/ITEM=CACHING to track improvements in buffering performance for files.
- The OPERATIONS item will give you a good indication of whether you are performing reads or writes. This information may be useful in determining whether shadowing will improve or degrade performance.

Time Required to Teach

This module should take about 15 to 20 minutes to teach.

Understanding CPU Performance

Instructor Notes

The first key point to CPU performance is that everything is not as simple as it seems. Timing of instructions are very dependent on:

- The state of the caches.
- Alignment issues.
- Your ability to properly use the pipelines.

The GEM backend compilers that Compaq provides give the best optimizations available. All standard compilers use these optimizations on Alpha platforms.

Alignment traps should not occur unless the compiler cannot detect their possibility. Poorly aligned data that is detected by the compiler will be cleaned up, through padding. If you force `/NOMEMBER_ALIGNMENT`, extra instructions will be generated. This mechanism is significantly less costly than dealing with alignment traps.

Symmetric Multiprocessing (SMP) scaling is dependent on:

- The ability for multiple processes to use the CPU.
- Possible use of Kernel threads.
- How much Interrupt state/stack time is occurring.

Note, with the exception of `FAST_PATH` CI interrupts, all interrupts are fielded by the primary processor. This behavior may minimize some of the potential benefits of SMP.

SMP performance is also affected by the amount of MP synchronization time that you experience. Waiting on Spinlocks wastes CPU cycles. Currently, you hit a limit at 6 to 10 CPUs, where you will normally not experience any performance improvement by adding CPUs. You may also experience degradation of CPU performance as you add CPUs beyond these limits.

OpenVMS Galaxy environments give several potential benefits for a CPU bound environment, including:

- The ability to spread interrupt servicing across multiple primary processors in a single box.
- The ability to dynamically move CPUs between instances, dependent on business needs.
- Shared memory.

If you are unfamiliar with Galaxy, peruse *The OpenVMS Galaxy Guide*.

One very crude way to view a Galaxy is as multiple cluster nodes cooperating within the same box. An "instance" is a separate instantiation of the OpenVMS operating system, running on a segmented hardware partition.

Key Points

- Nonaligned data that is detected, by the compiler, will cost you about 3 times the performance of using aligned data.
- Nonaligned data that is not detected, by the compiler, will cost you about 10 times the performance of using aligned data.
- Do not expect CPU performance to grow in a linear fashion as CPUs are added to an SMP configuration.
- FAST_PATH CI interrupt processing spreads interrupt processing across at least 2 CPUs.
- OpenVMS Galaxy may be a good way to segment interrupt processing in a single box.
- OpenVMS Galaxy allows the system manager to dynamically reconfigure CPUs, dependent on the current demands of the business.

Time Required to Teach

This module should take about 45 minutes to cover.

Understanding Time Spent in Modes

Instructor Notes

The important thing in this module is learning what you have to live with and what is overhead that can be eliminated. For instance, if you are caching files effectively and minimizing I/Os, you will be processing records faster. If you are processing records faster, you will be enqueueing and dequeuing more locks. Hence, you will experience a higher percentage of kernel mode time and less idle time (waiting for I/Os) to complete.

However, if you have a process waking up every 5 minutes scanning an entire disk looking for a file, when you could direct the search to a single directory, you have kernel mode time that could be eliminated. This time will be visible through \$MONITOR FCP.

Key Points

- Attempt to identify the activities that may be contributing to kernel mode time.
- Optimize those activities that can be optimized.
- Some idle time is good for response time.
- Idle time is bad for throughput.

Time Required to Teach

This module should take 30 to 45 minutes to cover.

OpenVMS Scheduling and Priorities

Instructor Notes

The first important point in this module is that QUANTUM can be reduced to improve response time. This is especially true for systems that are supporting 100s to 1000s of users. A QUANTUM setting of 20 allows 5 processes/threads to use the CPU per second. A QUANTUM setting of 4 allows 25 processes to use the CPU per second. Throughput will degrade, but users will probably be happier.

Scheduling states to watch for are:

- COM
- PFW
- FPG
- RWMPB/RWSCS/RWPFF

Key Points

- Consider lower priorities for batch queues.
- Consider a lower QUANTUM for response time oriented systems.
- Watch for unusual states.

Time Required to Teach

This module should take about 30 to 45 minutes to cover.

PIXSCAN and DORMANTWAIT

Instructor Notes

These parameters work just fine. Whenever I have had to change them on a system, it is because someone previously messed them up.

Key Points

- Leave PIXSCAN and DORMANTWAIT at their default settings!

Time Required to Teach

This module should take about 15 minutes to teach.

Labs

This is probably a good point to break off for day 4 labs.

Locking Concepts

Instructor Notes

Most locking problems are caused by poor application design. One application is holding onto a lock too long and blocking another. Unfortunately, it is very difficult to detect this behavior through use of the MONITOR utility. If you suspect that this is happening, use SDA. The SDA command SHOW RESOURCE/CONTENTION will show held locks that are blocking others. You will have to do a little research to determine which processes are blocking which. Processes that are being blocked will normally show up in LEF wait states.

Much has been made of sizing RESHASHTBL and LOCKIDTBL. Note, any attempts to tune these values beyond AUTOGEN settings will yield only marginal improvement in performance.

Key Points

- If an application is running slow, but all other performance metrics are normal, look for locking problems.
- Deadlock finds indicate a severe application or database design problem.

Time Required to Teach

This module should take 30 to 45 minutes to cover.

Distributed Locking Concepts

Instructor Notes

It is important to understand the distributed nature of the lock manager. The lock data is interest based. Nodes, which have no interest in a resource, maintain no information about the resource, with the possible exception of directory information.

Since locking is interest based, the boot order of nodes may affect the mastership of resources. The SHOW\$DEBUG logical may be of interest, if this appears to be a problem. Setting the logical to a non-null value will give you the information on mastership.

If you notice excessive Interrupt state time, look for tree movement in SDA. The SYSGEN parameter PE1 can be used to block or minimize tree movement.

One potential problem area with distributed locking is the Local Area VMScluster. It is common to set LOCKDIRWT to 1 on the boot members and 0 on the satellites. This strategy has the advantage of minimizing cluster state transition time. It has the disadvantage of increasing interrupt state time on the boot members for distributed locking at the same time that they are MSCP serving. Note, if the boot members are primarily serving disks, they are probably not very interested in locks associated with resources. It may be better from a performance standpoint to set the values opposite to the default settings, i.e. boot members set to 0 and satellites set to 1.

Key Points

- Distributed locking is becoming a problem if you consistently detect RWSCS wait states.
- Use PE1 to block or minimize tree movement.
- Consider LOCKDIRWT settings of 0 for MSCP servers, that are primarily servers and do not support significant user activity.

Time Required to Teach

This module should take about 45 minutes to cover.

MSCP Server

Instructor Notes

MSCP server time is spent on the interrupt stack. Its activity will block other cluster-related activities like distributed locking. Consider methods for isolating the two activities.

Larger buffers can improve throughput.

Key Points

- If the Buffer Wait Rate exceeds 2, consider increasing MSCP_BUFFER.

Time Required to Teach

This module should take about 15 to 20 minutes to cover.

Appendix A Through E

The appendices are supplied for the following purposes:

- Appendix A
 - Use this appendix if students come into the class with little or no background with SYSGEN/SYSMAN and/or AUTOGEN.
- Appendix B
 - This may be of interest to VAX users.
- Appendix C
 - Fast I/O is available. It does not significantly speed up I/O. It will reduce some of the CPU overhead and spinwait time associated with I/O.
- Appendix D
 - Lab Exercises
- Appendix E
 - Lab Solutions