Guide to VAX/VMS Performance Management

Order Number: AI–Y515B–TE

April 1986

This manual is a conceptual and tutorial guide for experienced users responsible for optimizing performance on VAX/VMS systems.

Revision/Update Information:	This revised manual supersedes the *Guide to VAX/VMS Performance Management* Version 4.0.
Software Version:	VAX/VMS Version 4.4

digital equipment corporation
maynard, massachusetts

April 1986

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital

ZK-2842

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

* Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

Contents

FIGURES

TABLES

Preface

This manual presents techniques for evaluating, analyzing, and optimizing performance on a VAX/VMS configuration. Discussions address such wide-ranging concerns as the following:

- Understanding the relationship between workload and system capacity
- Learning to use performance-analysis tools
- Responding to complaints about performance degradation
- Helping the site adopt those programming practices that result in the best system performance
- Using the system features that distribute the workload for better resource utilization
- Knowing when to apply software corrections to system behavior—"tuning" the system to allocate resources more effectively
- Evaluating the effectiveness of a tuning operation; knowing how to recognize success and when to stop

The manual includes detailed procedures to help you evaluate resource utilization on your system and to diagnose and overcome performance problems resulting from memory limitations, I/O limitations, CPU limitations, human error, or combinations of these. The procedures feature sequential tests that use VAX/VMS tools to generate performance data; the accompanying text explains how to evaluate it.

Whenever an investigation uncovers a situation that could benefit from adjusting system values, those adjustments are described in detail, and hints are provided to clarify the interrelationships of certain groups of values. When such adjustments are not the appropriate or available action, other options are defined and discussed.

Decision-tree diagrams summarize the step-by-step descriptions in the text. These diagrams should also serve as useful reference tools for subsequent investigations of system performance.

This manual does not describe methods for capacity planning, nor does it attempt to provide details about using VAX RMS features, since users should refer instead to the *Guide to VAX/VMS File Applications* for that information. Likewise, the manual does not discuss DECnet–VAX performance issues, since the *VAX/VMS Networking Manual* provides that information.

Intended Audience

This manual addresses system managers and other experienced users responsible for maintaining a consistently high level of system performance, for diagnosing problems on a routine basis, and for taking appropriate remedial action.

Structure of This Document

The manual is divided into five chapters, each covering a related group of performance management topics.

- Chapter 1, Introduction to Performance Management, provides a review of workload management concepts, guidelines for evaluating user complaints about system performance, and a discussion of performance investigation and tuning strategies.
- Chapter 2, Review of VAX/VMS Resource Management, describes VAX/VMS resource management mechanisms.
- Chapter 3, Managing System Resources, explains how to use the Monitor Utility and other VAX/VMS tools to collect and analyze data on your system's hardware and software resources. Included are suggestions for reallocating certain resources should analysis indicate the need.
- Chapter 4, Diagnosing Performance Problems, outlines procedures for investigating performance problems and isolating specific resource limitations.
- Chapter 5, Compensating for Resource Limitations, provides recommendations for improving performance with available resources.

Associated Documents

For additional information on the topics covered in this manual, you can refer to the following documents:

- *VAX/VMS System Manager's Reference Manual*
- *Guide to VAX/VMS File Applications*
- *VAX/VMS Run-Time Library Routines Reference Manual*
- *VAX/VMS System Services Reference Manual*
- *VAX/VMS Utility Routines Reference Manual*
- VAX/VMS Utility Reference Manuals

Conventions Used in This Document

Convention	Meaning
RET	A symbol with a one- to three-character abbreviation indicates that you press a key on the terminal, for example, RET .
$ SHOW TIME 05-MAY-1986 11:55:22	Command examples show all output lines or prompting characters that the system prints or displays in black letters. All user-entered commands are shown in red letters.
$ TYPE MYFILE.DAT . . .	Vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to the particular command is shown or that not all the data a user would enter is shown.
file-spec,...	Horizontal ellipsis indicates that additional parameters, values, or information can be entered.
[logical-name]	Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

1 Introduction to Performance Management

Performance management of a VAX/VMS system means optimizing your hardware and software resources for the current workload. This task entails several distinct but related activities:

- Acquiring a thorough familiarity with your workload and an understanding of how that workload exercises the system's resources. This knowledge, combined with an appreciation of the VAX/VMS resource management mechanisms described in Chapter 2, will enable you to establish realistic standards for system performance in areas such as the following:
  - Interactive and batch throughput
  - Interactive response time
  - Batch job turnaround time
- Monitoring system behavior on a routine basis in order to determine when and why a given resource is nearing capacity. Chapter 3 describes procedures you can use to track resource usage.
- Investigating reports of degraded performance from users.
- Planning for changes in the system workload or hardware configuration and being prepared to make any necessary adjustments to system values.
- Performing, after installation, certain optional system management operations.

To help you understand the scope and interrelationship of these activities, the following sections provide

- A review of workload management concepts
- Guidelines for evaluating user complaints about system performance
- A discussion of performance investigation and system tuning strategies
- A checklist of system management performance options

1.1 Knowing Your Workload

One of the most important assets that a system manager brings to any performance evaluation is an understanding of the normal workload and behavior of the system. While this manual can indicate how to measure or quantify the workload and how to draw certain conclusions about that workload, it is impossible to address the unique aspects of every workload. Therefore, each system manager must assume the responsibility for understanding the system's workload sufficiently to be able to recognize normal and abnormal behavior; to predict the effects of changes in applications, operations, or usage; and to recognize typical throughput rates. The qualified system manager can readily answer such questions as:

- What is the typical number of users on the system at each time of day?
- What is the typical response time for various tasks for this number of users, at each hour of operation?
- What are the peak hours of operation?
- Which jobs typically run at which time of day?
- Which commonly run jobs are intensive consumers of the CPU? of memory? of disk?
- Which applications involve the most image activations?
- Which parts of the system software, if any, have been modified or user-written, such as device drivers?
- Are there any known system bottlenecks? Are there any anticipated ones?

Clearly, these are questions this manual cannot answer; at best it will point out the tools used for obtaining the answers. Yet, each system manager must know the answers to these questions to be able to address any performance issues.

If you are a novice system manager, you should dedicate considerable time to observing system operation with the following tools:

- Monitor Utility
- Accounting Utility
- SHOW commands (available through DCL)

Chapter 3 provides detailed procedures for using the Monitor Utility and (to a lesser extent) other VAX/VMS tools to observe and evaluate system performance. Over time you will learn the typical page fault rate for your system, the typical CPU usage, the normal memory usage, typical modes of operation, and so forth. You will begin to see how certain activities affect system performance and how the number of users or the time of day affects some of the values. As you continue to monitor your system, you will come to know what range of values is acceptable, and you will be better prepared to use these same tools, together with your knowledge, to detect aberrations. *Routine evaluation of the system is critical for effective performance management. You cannot afford to wait for problems to develop before you learn how your system performs.*

You can become better educated about your system's operation if you use the Monitor and Accounting utilities on a regular basis to capture and analyze certain key data items (see Sections 3.2 and 3.3). By collecting this data, you will also be in a position to see usage trends and to predict when your system may reach its capacity. You should exercise care, however, in selecting the items you want to measure and the frequency with which you capture the data. If you are overzealous, the consumption of system resources to collect, store, and analyze the data can distort your picture of the system's workload and capacity. You will most certainly use excess storage capacity. The best policy is to have a specific plan before capturing data for how you will analyze and apply it.

With MONITOR, select an appropriate collection interval. As a guideline, avoid an interval value so short that you require unnecessary disk storage for the data, but do not select an interval so large that you miss significant events occurring in the interim.

You should also be particularly judicious in using image-level accounting on your system. (You enable image-level accounting with the DCL command SET ACCOUNTING/ENABLE=IMAGE. Use the /DISABLE=IMAGE qualifier to disable it.) As a rule, you should enable image-level accounting only when you plan to invoke ACCOUNTING to process the information provided in the file SYS$MANAGER:ACCOUNTNG.DAT. Once you have collected enough data for your purposes, disable image-level accounting. While image activation data can be very helpful in performance analysis, it is wasteful of processing time and disk storage if merely collected and never used.

1.1.1 Workload Management

System performance is directly proportional to the efficiency of workload management. Each installation must develop its own strategy for this key task. Before you attempt to adjust any system values, always ask yourself the following questions:

- Is there a time of day when the workload "peaks," that is, when it is noticeably heavier than at other times?
- Is there any way to balance the workload better? Perhaps some voluntary measures can be adopted by users, after appropriate discussion.
- Could any jobs be run better as batch jobs, preferably during nonpeak hours?
- Have primary and secondary hours of operation been employed with users? (See Section 1.1.2.) If not, could system performance benefit by adopting this practice? If the primary and secondary hours are in use, are the choices of hours the most appropriate for all users? (Plan to review this issue every time you either add or remove users or applications, to ensure that the desired balance is maintained.)
- Can future applications be designed to work around any known or expected system bottlenecks? Can present applications be redesigned somewhat, for the same purpose? (See the *Guide to VAX/VMS File Applications*.)
- Are you using to the utmost the code-sharing ability that the VAX/ VMS system offers you? If not, you will find that code sharing provides an excellent means to conserve memory, thereby improving performance over the life of the system. (See Section 2.2.3.)

Do not adjust any system values until you are satisfied that all these issues are resolved and that your workload management strategy is correct.

1.1.2 Workload Distribution

You should distribute the workload as evenly as possible over the time the computer is running. While scheduling interactive users evenly is made difficult by conventional working and sleeping hours, some of the following techniques should prove workable:

- Run large jobs as batch jobs—Establish a site policy that encourages the submission of large jobs on a batch basis. Regulate the number of batch streams so that batch usage is high when interactive usage is low. You might also want to use DCL command qualifiers to run batch jobs at lower priority, adjust the working set sizes, and/or control the number of

concurrent jobs. Chapter 9 in the *VAX/VMS System Manager's Reference Manual* discusses batch jobs.

- Restrict system use—Do not permit more users to log in at one time than the system can support with an adequate response time. You can restrict the number of interactive users with the DCL command SET LOGINS /INTERACTIVE (see the *VAX/VMS DCL Dictionary*). You can also control the number of concurrent processes with the MAXPROCESSCNT system parameter, and the number of remote terminals allowed to access the system at one time with the RJOBLIM system parameter (see the *VAX/VMS System Generation Utility Reference Manual*).

  You might also restrict use of the system by groups of users to certain days and hours of the day. You can use the Authorize Utility to define the permitted login hours for each user. In particular, refer to the AUTHORIZE qualifiers /PRIMEDAYS, /P_RESTRICT, /PFLAGS, /SFLAGS, and /S_RESTRICT. Remember you can use the DCL command SET DAY (see the *VAX/VMS DCL Dictionary*) to override the conventional day of the week associations for primary and secondary days. For example, you might need to specify a primary day of the week as a secondary day when it is a holiday.

- Design applications to reduce demand on binding resources—If you know where your system bottlenecks are or where they will likely occur in the near future, you can distribute the workload more evenly by planning usage that minimizes demand on the bottleneck point(s). (See the *Guide to VAX/VMS File Applications.*)

1.1.3 Code Sharing

To ensure optimum performance of your system, you must make certain that frequently used code is shared. This important system feature should not be overlooked as a cost-effective means of optimizing memory utilization. (See Section 2.2.3.)

The site-independent startup command procedure creates permanent global sections for system programs and subroutines by installing them as known images with the shared attribute. You should use the same type of approach for user programs and subroutines that have been designed for sharing and have reached a production status or are in general use. However, be sure to use the site-specific startup command procedure to install them as known images with the shared attribute. (See the *VAX/VMS System Manager's Reference Manual.*)

Programmers should be encouraged to write shareable code. Useful references include the *VAX/VMS Linker Reference Manual,* and the *VAX/VMS Run-Time Library Routines Reference Manual.*

1.2 Evaluating User Complaints

Typically, an investigation into system performance begins when you receive a complaint about slowdown of interactive response times, or about some other symptom of decreased throughput. You may or may not observe the behavior firsthand.

Before you decide that the current complaint reflects a true performance problem, however, you should be convinced that hardware resources are adequate. You should also know the workload reasonably well and be sure you have been managing it according to the guidelines in Section 1.1.

You then need certain basic facts:

- The number of users on the system at the time the problem occurred
- The response times
- Evidence of jobs hung and unable to complete

As you receive this information, you should match it to your knowledge of the normal workload and operation of your system, following the steps shown in Figure 1–1. In this decision-tree diagram, and all the subsequent ones like it, the procedure begins at a numbered node, proceeds through nodes that ask questions, and ends with one of the following:

- A numbered node that matches a continuation node on another diagram that follows
- A concluding node that has no number

At this point, you simply want to eliminate false alarms and inaccurate reports; you need to convince yourself that there is some evidence of a problem. The best proof occurs when you can observe or duplicate the problem.

1.2.1 Recognizing Hardware and MWAIT Problems

Hardware problems are a common source of performance complaints. For example, if devices are offline or malfunctioning, performance can degrade. Check the operator log and the error log for indications of problems with specific devices. You can issue the DCL commands SHOW ERROR and ANALYZE/ERROR to help determine of hardware is contributing to a performance problem.

It is a good idea to review the previous day's error log as part of your morning routine. The following DCL command (which requires the SYSPRV privilege) provides a count of errors logged since yesterday:

```
$ ANALYZE /ERROR /BRIEF /LOG /OUTPUT=NL: /SINCE=YESTERDAY
```

You can see at a glance whether a more detailed error analysis is in order. If a hardware problem does exist, correct it as quickly as possible, and then reevaluate the performance complaint.

For more information on error logging, see Chapter 10 in the *VAX/VMS System Manager's Reference Manual;* for information on using the Analyze /Error_Log Utility, refer to the *VAX/VMS Error Log Utility Reference Manual.*

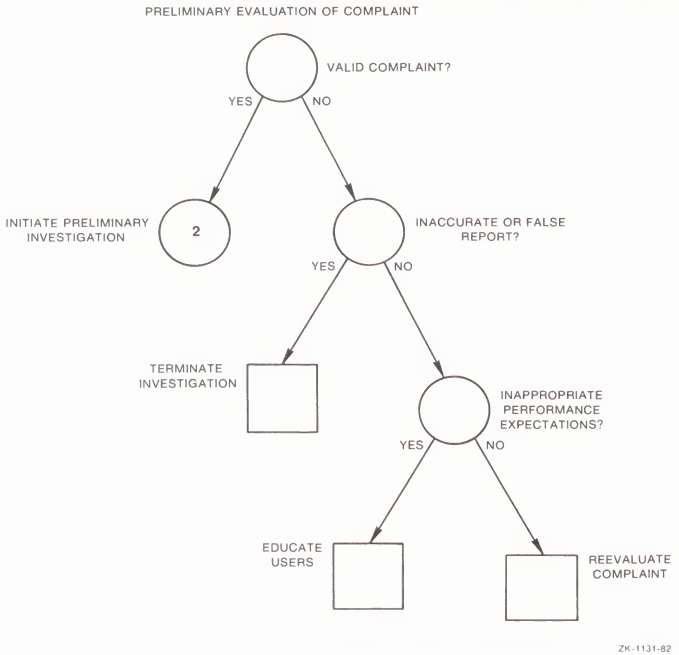
Another potential cause of performance degradation occurs when a process enters the miscellaneous resource wait (MWAIT) state because some resource, such as a page file or mailbox capacity, is unavailable. Issue the DCL command MONITOR STATES and look for processes in the MWAIT state. Processes in this state are blocked either by a miscellaneous resource wait or a mutual exclusion semaphore (MUTEX). Displays generated by the MONITOR PROCESSES and SHOW SYSTEM commands show the reason for the MWAIT state. (For an explanation of MWAIT codes, refer to the discussion of the MONITOR STATES command in the *VAX/VMS Monitor Utility Reference Manual.*)

Note

If the system fails to respond while you are investigating an MWAIT condition, check the system console for error messages.

Should the MWAIT persist after you increase the capacity of the appropriate resource, investigate the possibility of a programming design error.

Figure 1–1 Verifying the Validity of a Performance Complaint



1.2.2 Recognizing Unrealistic Expectations

Always bear in mind that what appears at first to be a performance problem can turn out to be really a case of unrealistic expectations. Perhaps users expect response times to remain constant, even as the system workload increases. Perhaps an unusual set of circumstances has caused exceptionally high demand on the system all at once. If you find any of these to be the cause of unsatisfactory performance, you may find that education of the users is the appropriate action. Adjusting system values will accomplish nothing in such circumstances.

Whenever you can anticipate a temporary workload change that will affect your users, you should try to notify them through broadcasts and/or text in the system notices.

1.3 Preparing for a Performance Investigation

Once you have established that there is a persistent performance problem, you normally become involved in an investigative process to determine the cause and possible corrective action. The process usually becomes easier as you gain experience with the system and its normal operation. A number of software tools exist (such as the Monitor and Accounting utilities) that can help in the evaluation of system performance. (See Chapter 3.) Again, as you become experienced in using these tools, more rapid diagnosis is possible. Generally, system performance problems turn out to be the result of poor operation, lack of understanding of the workload and its operational ramifications, lack of resources, poor application design, human error, or combinations of these. You will rarely need to make major adjustments to system parameters.

1.3.1 Understanding What Tuning Is

Tuning is the process of altering various system values to obtain the optimum *overall* performance possible from any given configuration and workload. However, the process does not include the acquisition and installation of additional memory or devices, although in many cases such additions (when made at the appropriate time) can vastly improve system operation and performance.

Always aim for best overall performance, that is, performance viewed over time. The workload is constantly changing on most systems. What guarantees optimal performance at one point in time may not produce optimal performance a short time later as the workload changes. You can only hope to establish values that, on average, produce the best overall performance.

Before you undertake any action, you must recognize that the following sources of performance problems cannot be cured by adjusting system values:

- Improper operation
- Unreasonable performance expectations
- Insufficient memory for the applications attempted
- Inadequate hardware configuration for the workload, such as too slow a processor, too few buses for the devices, too few disks, and so forth
- Improper device choices for the workload, such as using disks with insufficient speed or capacity, or installing DZ11s when DMF32s would be more beneficial

- Hardware malfunctions
- Human errors, such as poor application design or allowing one process to consume all available resources

When it becomes necessary to make adjustments, you normally select a very small number of values for change, based on a careful analysis of the behavior being observed. These values are usually either system parameters (as described in the *VAX/VMS System Generation Utility Reference Manual*), or entries in the User Authorization File (UAF) that affect particular users (see Chapters 5 and 6 in the *VAX/VMS System Manager's Reference Manual*). Normally, you modify system parameters using the AUTOGEN command procedure described in the *VAX/VMS System Manager's Reference Manual*. One of AUTOGEN's special features is that it makes automatic adjustments for you in associated parameters. To control the values in the UAF, you use the Authorize Utility, which is described in the *VAX/VMS Authorize Utility Reference Manual*.

Chapter 5 of this manual shows you how to pinpoint which parameters are likely to produce the optimum change. Included are some hints for selecting new values.

1.3.2 Predicting When Tuning Is Required

DIGITAL believes, for several reasons, that tuning is rarely required for VAX/VMS systems. First of all, the system includes the AUTOGEN command procedure, which establishes initial values for all the configuration-dependent system parameters so that they match your particular configuration.

Second, the system includes features that in a limited way permit it to adjust itself dynamically during operation. That is, the system detects the need for adjustment in certain areas, such as the nonpaged dynamic pool, working set size, and the number of pages on the free and modified page lists. The system makes rough adjustments in these areas automatically for you. As a result, these areas can grow dynamically, as appropriate, during normal operation. (More details about several of these VAX/VMS automatic adjustment features appear in Section 2.2.)

Finally, experience has shown that the most common cause of disappointment in system performance is insufficient hardware capacity. Once the demand on a system exceeds its capacity, adjusting system values will not result in any improvements, simply because such adjustments are a means of trading off or juggling existing resources.

Although tuning is rarely required, be aware that it is appropriate in response to two particular situations:

1. If you have adjusted your system for optimal performance with current resources and then acquire new capacity, you must plan to compensate for the new configuration. In this situation, the first and most important action is to execute the AUTOGEN command procedure.
2. If you anticipate a dramatic change in your workload, you should expect to compensate for the new workload.

1.4 Evaluating Tuning Success

Whenever you make adjustments to your system, you must plan to spend time monitoring its behavior afterward, to ensure that you obtain the desired results and no unexpected or undesired results. You can observe results with both the Monitor Utility (see the *VAX/VMS Monitor Utility Reference Manual*) and the various forms of the DCL command SHOW (see the *VAX/VMS DCL Dictionary*).

For example,you might consider running some programs whose results you believe are fixed and reproducible, at the same time that you run your normal workload. If you run the programs and measure their running times under nearly identical workload conditions both before and after your adjustments, you can obtain a basis for comparison.

However, when applying this technique, remember to take the measurements under very similar workload conditions. Also, remember that this test alone does not provide conclusive proof of success. There is always the possibility that your adjustments may have favored the performance of the image you are measuring—to the detriment of other images. Therefore, in all cases, continue to observe system behavior closely for a time after you make any changes.

1.5 Deciding When To Stop Tuning

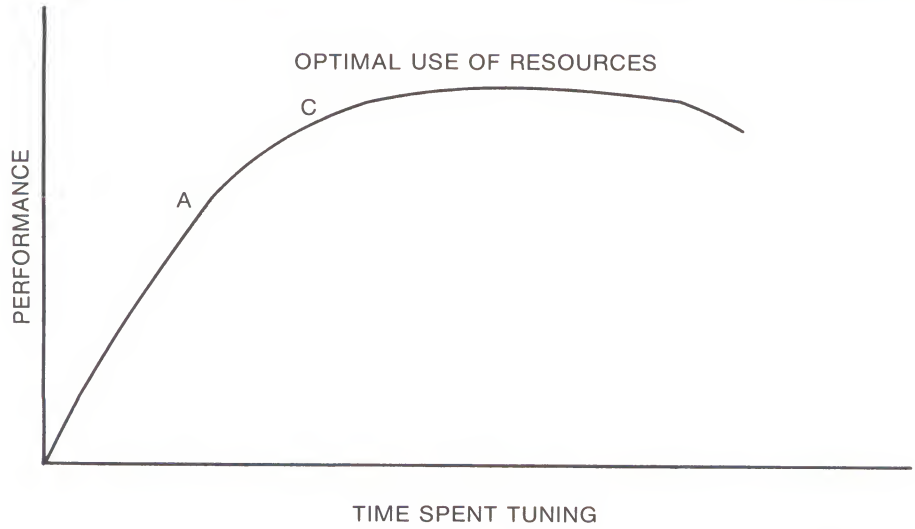
In every effort to improve system performance, there comes a point of diminishing returns. In other words, you will find that once you obtain a certain level of improvement, you can spend a great deal of time tuning the system beyond that point and achieve only marginal additional improvement. Figure 1–2 represents this pattern.

Recognizing this phenomenon, you will be in a better position to know when to stop. As a guideline, if you make some adjustments and see a marked improvement, then make more adjustments and see about half as much improvement, then fail to make more than a small improvement on your next attempt or two, you should stop and evaluate the situation. You can

probably assume you have done your best and you are close to point C on the graph. In most situations, this is the point at which to stop. If you are not satisfied with the final performance, you should carefully address the issue of increasing your capacity through the addition of hardware. Generally, memory is the single piece of hardware needed to solve the problem. However, some situations warrant obtaining additional disks or more CPU power. Very few situations warrant the expense of continued adjustments for such minimal potential improvement, once the improvement depicted at point C has been obtained.

When you use the AUTOGEN command procedure to set the system parameters, you should expect performance to be at a point between A and B in Figure 1–2. However, in a system that has been improperly adjusted, you would find performance to be near the origin point in the diagram. Because such systems usually exhibit blatant symptoms with fairly obvious and simple solutions, you will likely find that tuning—in the form of adjustments to certain critical system values—produces a high return for the time and effort invested, and that there is a much lower risk of error.

Figure 1–2 Time Spent Tuning Versus Performance Improvements



1.6 Performance Options for the System Manager

Following is a list of optional system management operations, normally performed after installation, that often result in improved overall performance. Note, however, that not all options are appropriate at every site.

- Decompress system libraries—Most of the libraries shipped with Version 4.0 and later versions of the VAX/VMS operating system are in a compressed format in order to conserve disk space; the CPU dynamically decompresses them whenever they are accessed, and the resulting performance slowdown is especially noticeable during link operations and when requesting online help. If you have sufficient disk space, decompressing the libraries will improve CPU and elapsed time performance. To do this, invoke the command procedure SYS$UPDATE:LIBDECOMP.COM. The decompressed object libraries take up about 25 percent more disk space than when compressed; the decompressed help libraries take up about 50 percent more disk space.

- Disable file system high-water marking—This security feature guarantees that users cannot read data they have not written. It is implemented by erasing the previous contents of the disk blocks allocated every time a file is created or extended. High-water marking is set by default whenever a volume is initialized.

  Disabling the feature will improve system performance by an amount that varies depending on how often new files are created, how often existing files are extended, and how fragmented the volume is. To disable high-water marking, you can specify the /NOHIGHWATER qualifier when initializing the volume, or you can issue the following DCL command at any time:

  ```
  $ SET VOLUME/NOHIGHWATER_MARKING device-spec[:]
  ```

  Then dismount and remount the volume. However, you should consider the security implications of disabling this feature.

- Set RMS file extend parameters—In Version 4.0, the VAX RMS file extend default has been reduced from 80 to 0 blocks. Because files extend in increments of twice the multiblock count (default 16), system defaults now provide file extension of only 32 blocks. Thus, when files are created or extended, increased I/O may slow performance. The problem can be overcome by specifying larger values for SYSGEN file extend parameters or by setting the SYSGEN parameter RMS_EXTEND_SIZE=80. (See the *VAX/VMS System Generation Utility Reference Manual.*)

- Relink images—Beginning with VAX/VMS Version 4.0, the Run-Time Library (VMSRTL) is separated into five smaller libraries. Running images linked under previous versions of VAX/VMS will therefore incur the image activation costs of mapping all five libraries, even if only one is needed. You may improve performance by relinking pre-Version 4.0 images that reference Run-Time Library routines, so that only the required libraries are mapped and activated.
- Install frequently used images—When an image is accessed concurrently by more than one process on a routine basis, install the image with the Install Utility, specifying the /OPEN, /SHARED, and /HEADER_RESIDENT qualifiers. You will thereby ensure that all processes use the same physical copy of the image, and that the image will be activated in the most efficient way.

  Generally, an image takes about two additional physical pages when installed /OPEN/HEADER_RESIDENT/SHARED. The utility's LIST/FULL command shows the highest number of concurrent accesses to an image installed with the /SHARED qualifier. This information can help you decide if installing an image is worth the space. For more information on the Install Utility, refer to the *VAX/VMS Install Utility Reference Manual*.
- Reduce system disk I/O—You can move frequently accessed files off the system disk and use logical names or, where necessary, other pointers to access them. For example:
  - SYSUAF.DAT (SYSUAF is the logical name)
  - RIGHTSLIST.DAT (RIGHTSLIST is the logical name)
  - VMSMAIL.DAT (VMSMAIL is the logical name)
  - NETUAF.DAT (NETUAF is the logical name)
  - JBCSYSQUE.DAT (File specification in the START/QUEUE /MANAGER command is the logical name)
  - ERRFMT log files (SYS$ERRORLOG is the logical name)
  - MONITOR log files (SYS$MONITOR is the logical name)
  - Default DECnet account (DECNET record in SYSUAF file)

You may also want to consider moving paging and swapping activity off the system disk by creating large secondary page and swap files on a less heavily utilized disk.

2 Review of VAX/VMS Resource Management

Once you have taken the necessary steps to manage your workload, you can approach reports of performance problems as if they imply the need for investigation. Before you proceed, however, it is imperative that you be well versed in the concepts of VAX/VMS resource management. If you lack such understanding, you are likely to encounter unnecessary problems in your tuning attempts. For this reason, what follows is a fairly intensive review of this key area.

In Section 2.1, an analogy is employed to explain many aspects of memory management. Concepts are introduced in terms of a workshop environment and then translated into VAX/VMS terms. In Section 2.2, automatic working set adjustment and swapper trimming are described. Section 2.2.4 discusses process scheduling.

Note

Many readers will not require the full review of VAX/VMS memory management concepts that Section 2.1 provides. For that reason, boldface is used to highlight the parts specific to VAX/VMS. Readers can check their familiarity with the VAX/VMS vocabulary and concepts by reading just the parts printed in boldface. If some of the terms are unfamiliar, it would be wise to read the entire section to obtain the benefit of the explanations offered through the analogy. However, if this material is well understood, readers can proceed directly to Section 2.2.

2.1 A VAX/VMS Memory Management Analogy

To better understand how VAX/VMS manages system resources for its users, consider the analogy of a large workshop. In some ways, a computer system and a workshop have many aspects in common.[1]

Imagine a large workshop where a number of workers come to build different kinds of objects. Workers tend to work on their own projects, with their own set of parts and construction schematics; that is, the projects are not usually done in teams. The workshop is managed by a supervisor with the help of a few assistants. It is this supervisor's responsibility to see that as many

[1] Several of the basic ideas for the workshop/warehouse example were borrowed from the work of Jeffrey Berryman at the University of British Columbia, Canada, and S. B. Schwartz at Digital Equipment Corporation, Maynard, MA.

projects as possible are finished as quickly as possible, by allocating the various resources of the workshop (workbench space, time in the workshop, and so forth).

Since it is often the case that the workers are building very intricate objects that require many parts—more parts than can fit all at once in the workshop—the workshop is supplemented by one or more warehouses that can store the parts when they are not being actively used by any of the workers. The warehouses are located a short distance down the road.

In a similar way, with the VAX/VMS operating system, a number of processes can run in available physical memory. (A process is a schedulable entity on the system.) There is one operating system, VAX/VMS, which consists of the executive (which is always resident in physical memory) and other components. Each process performs work, which involves, in simple terms, the manipulation (processing) of data. The system design tries to ensure that each process can complete its work as quickly as possible. In addition to main memory, the configuration supports several secondary storage devices (disks), where additional bytes are stored.

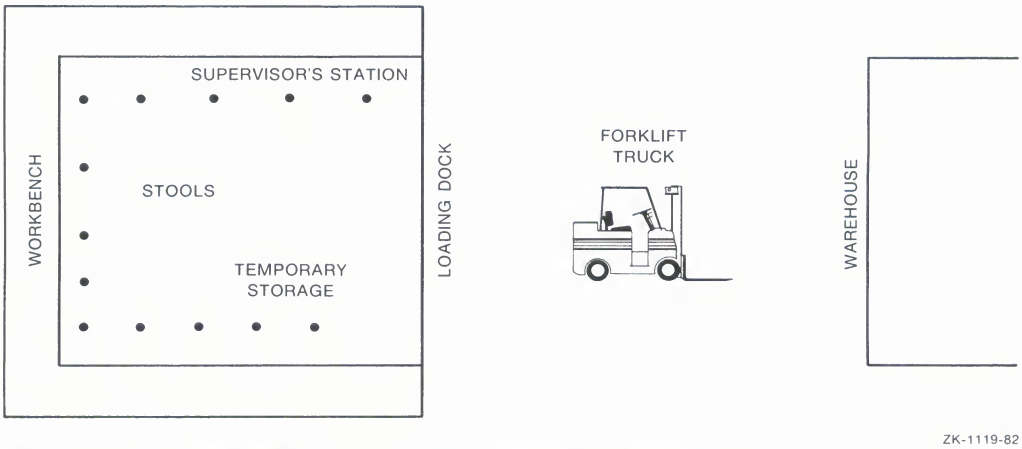
In the imaginary workshop there is one long workbench that occupies three of the four walls. The supervisor and each worker work at stations along the workbench. The fourth wall is taken up by a loading dock and some temporary storage space. The loading dock is used by the forklift trucks that move parts to and from the warehouses. Each warehouse has its own forklift. (See Figure 2–1.)

As workers report for work, the supervisor gives them a stool and assigns them to a portion of the workbench where they will be able to keep some or all of the parts they need to complete their projects. The supervisor decides how many workers to allow in the workshop at any given time by limiting the number of stools given out, or by allocating big areas of the workbench to each worker so that fewer workers will fit around the bench. (Workers who cannot get access to the workbench simply have to wait in the corner by the loading dock until a stool or some of the workbench is free.)

With VAX/VMS, physical memory can be thought of as divided into three major parts, according to their usage. There is the portion available for the processes to work, there is the portion reserved for the resident executive, and there is a portion for the page cache, where data is stored for movement from and to the disk(s). Each disk has only one access path available to transfer data from and to physical memory (that is, to perform the disk I/O).

There are enough balance slots reserved in physical memory for the maximum number of processes expected to run concurrently, including the operating system. If the number of balance slots is reduced, fewer processes can run concurrently. The operating system and each process have their individual working spaces in physical memory, known as their working sets. More precisely, working sets are process-specific lists. The working set includes all the valid pages in memory for any particular process. Pages in the working set usually represent a subset of the total number of pages in the process's page tables.

Figure 2–1 Illustration of the Workshop Layout



Now, there are a few unusual aspects to this workshop. The first is that all the parts the workers use are the same size and shape, although they are not all the same color. This means that the pieces are versatile, so that they can be used to make almost anything. This uniformity in same size and shape is used to advantage in the workshop. The parts can be organized in a variety of different containers that have the same capacity, and therefore they can be moved around very easily.

For example, all parts out in a warehouse are stored in crates of the same size. So it is very simple to store new parts in a warehouse, because any empty crate will be the right size. It is also easy to transport lots of crates around, since they stack so neatly.

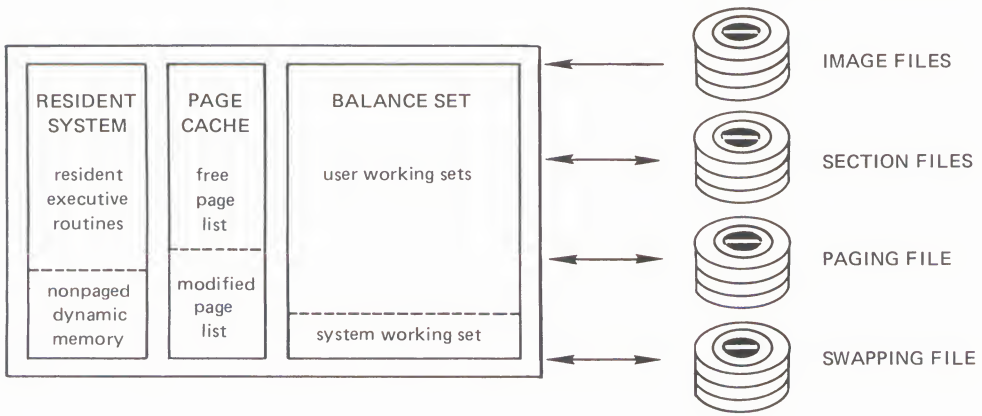
Furthermore, the workers in the workshop always keep their parts in trays that are just big enough to handle exactly one crateful of parts. This procedure has two advantages: it simplifies the process of crating and uncrating the parts when they arrive on the loading dock (the contents of a crate are just dumped onto a tray), and it makes very efficient use of the workbench, since parts are never left lying about.

In VAX/VMS, the basic addressable unit for data is the byte. Bytes are always stored in groups, called pages, with 512 bytes to a page. Pages are kept in the working sets or in sections of physical memory known as the page cache, as well as on the disks. The page is a convenient vehicle for moving uniform numbers of bytes into and out of memory. Note that, although bytes are always the same size, they do, of course, have varying contents.

Figure 2–2 illustrates the configuration of memory for VAX/VMS systems.

When the demand for workbench space and shop time is low, things run well. The problems occur when many workers all want to use the shop at the same time. If the supervisor is not careful, the shop gets too crowded; workers start getting in each other's way and the net output of the shop starts to drop.

Figure 2–2 VAX/VMS Memory Configuration



To prevent such undesirable circumstances, the supervisor sets up some general shop procedures. First, so as to be fair, workers are assigned to *fixed*-length shifts as they report to work—thus guaranteeing a worker a

certain amount of time in the shop, at the workbench. If, when a worker's shift expires, there is no one else waiting, the worker can keep on working as much overtime as desired. However, if someone *is* waiting, the worker gives up the stool, cleans the trays off that portion of the workbench, and lets the worker who is waiting take a shift.

With VAX/VMS, it is also important to maintain an even balance in the use of memory and the number of processes running at once. Each process has an available amount of time to perform its work—its quantum. (The quantum is a fixed slice of time; if no other process is waiting to exercise its quantum, the current process can keep renewing its quantum and retain control of the CPU.)

In the workshop, a typical project begins when a worker reports for work to the shop and goes to the supervisor to find out where and when to start working. The supervisor provides a stool, allocates a section of the long workbench for use, and records the starting time of the shift for later use. The worker then goes to the loading dock and, as soon as the appropriate forklift is free, drives out to the warehouse where the crates of parts needed for the new project are kept. The worker stacks several crates onto the forklift, and then drives them back to the loading dock.

Back at the shop, the worker has to get enough empty trays to hold all the parts in the crates just brought in from the warehouse. Once the trays are lined up, the worker uncrates the forklift load and arranges the trays on the assigned section of workbench. Then assembly work begins. When a part (say, of a particular color) is needed that is not currently available in the workshop, the worker goes back to the loading dock, gets on the appropriate forklift, and goes out to the warehouse to fetch the crate containing the needed part.

The supervisor's assistant uses an elaborate map to keep track of all the crates and trays in the workshop and the warehouses, and strives to ensure that each worker has a steady supply of trays for each project. When the worker needs another crate (which is in the open area of the workshop), a delay or interruption must occur so that the worker can retrieve it. However, first the worker checks whether there is room for it on the workbench. If not, the worker must remove a tray to the staging area, even if work never began on that tray.

During image activation, the groundwork is laid so that the process can bring in the first set of pages from the image file and use them in its own working set.

The job of scheduling physical memory falls to the swapper process. The swapper keeps track of the pages in both physical memory and on the disk paging and swapping files, so that it can ensure that each process has a steady supply of pages for each job.

When the process's demand for pages exceeds those available in the working set, some of the process's pages must be moved to the page cache to make room.

In VAX/VMS, there are two sections to the page cache in physical memory; those pages whose contents have been modified are stored on the modified page list, while those that have not been modified are kept on the free page list. When the page cache begins to fill up, the swapper transfers a cluster of pages from the modified page cache out to disk, to what is known as a paging file.

Sometimes the process needs additional pages that are stored in either an image file or a paging file. This, too, involves a page fault. If there is insufficient space in the working set, the process must move one or more of its pages to the page cache, as before. The process brings in groups of pages from the image file on disk, assuming that the process is likely to reference pages other than the one just referenced.

You can imagine the delays that sometimes occur in the workshop because there is only one forklift truck to each warehouse. If there were only one large warehouse with its one truck, the waits for the truck could be much worse.

With VAX/VMS, a similar type of bottleneck can occur if many processes begin page faulting at the same time, particularly if there is only one paging file for all processes, and the speed of retrieval is that of loading between memory and disk, which is slower than the memory accesses required to update the memory management database. Under such circumstances, installing additional paging files on separate disks or creating a larger cache can alleviate the bottleneck. (However, it may prove even more profitable to address the cause of the excessive page faulting in the first place.)

Table 2–1 summarizes the components of the analogy and their VAX/VMS equivalents.

Table 2–1 Corresponding Terms in the Workshop and VAX/VMS Analogy

Workshop Component	VAX/VMS Equivalent
Supervisor	Operating system
Workers	Processes
Workshop	Physical memory
Warehouse	Disk
Storage area in warehouse	Image, page or swap file; Section file

Table 2–1 Corresponding Terms in the Workshop and VAX/VMS Analogy (Cont.)

Workshop Component	VAX/VMS Equivalent
Workbench	Balance set
Workspace	Working set
Stools	Balance slots or resident process headers
Forklift truck	I/O channel
Crates	Disk blocks
Trays	Pages of memory
Parts	Bytes
Loading Dock	Page cache
Shift	Quantum
Interruption when crate is needed from loading dock or warehouse	Page fault—interruption when page is needed from cache or disk

2.2 Advanced Memory Management Mechanisms

VAX/VMS employs several sophisticated memory management mechanisms to improve performance on the system. This section describes four of these mechanisms: automatic working set adjustment, swapper trimming, memory sharing, and scheduling.

The VAX/VMS operating system, as provided by DIGITAL, enables these features by default. In the majority of situations, they produce highly desirable results in optimizing system performance. However, under special, rare circumstances, they might contribute to performance degradation by incurring their own overhead. The following sections describe these features and provide insight into how to adjust, or even turn them off, through tuning.

2.2.1 VAX/VMS Automatic Working Set Adjustment

The automatic working set adjustment (AWSA) feature refers to a system where processes can acquire additional working set space (physical memory) under control of VAX/VMS. VAX/VMS recognizes the amount of page faulting that is occurring for each process and factors this into the operation.

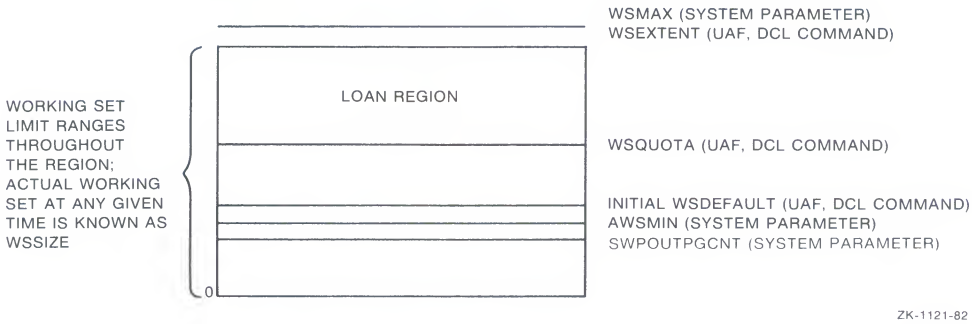
All processes have an initial default limit of pages of physical memory (a working set limit, referred to as WSDEFAULT). Any process that needs more space in memory is allowed to expand to the amount of a larger limit, known as the working set quota (referred to as WSQUOTA). Figure 2–3 illustrates these important regions.

Because page faulting is costly, VAX/VMS has another feature for extending working set space to needy processes, provided the system has free memory available. If the conditions are right, the process can borrow working set space up to a final limit known as its working set extent, or WSEXTENT.

In addition to the various limits placed on the working set size, system managers need to consider the actual number of pages the working set requires. In this chapter, the term *working set count* represents that number. The working set count consists of the process's pages plus any global pages that the process uses.

Whenever a process working set size increases, the growth occurs in increments, according to the value of the system parameter WSINC. VAX/VMS recognizes or reviews the need for growth only at the end of the next occurring quantum and after that minimum interval of time established by the system parameter, AWSTIME. You should think of the time from the start of the quantum right after an adjustment occurs, until the next quantum after the time specified by AWSTIME elapses, as the adjustment period. (The length of the quantum is defined by the system parameter, QUANTUM.) The system samples the page faulting rate of each process over that adjustment period defined by the system parameters AWSTIME and QUANTUM. For example, if the system quantum is 200 milliseconds and AWSTIME is 700 milliseconds, VAX/VMS would be reviewing the need to add or subtract pages from a process every time the process had consumed 800 milliseconds of CPU time, or every fourth quantum.

Figure 2–3 The Working Set Regions for a Process



By reviewing the need for each process to add some pages to its working set limit through the AWSA feature, VAX/VMS can better balance the working set space allocation among processes. Since the goal of this activity is to reduce the amount of page faulting, VAX/VMS decides whether to grant memory by comparing the current amount of page faulting that each process is undergoing, against a norm that has been established overall for all processes in the system. (The system parameters PFRATH and PFRATL define the upper and lower limits of acceptable page faulting, respectively.)

At the end of each process's adjustment period, if the page fault rate for the process is high (compared to PFRATH), VAX/VMS approves an increase in the working set size of that process in the amount of the system parameter WSINC, up to the value of its WSQUOTA, for the next adjustment period. If the increase puts the process above the value of WSQUOTA and thus requires a loan, VAX/VMS checks the availability of free memory against an established system norm, the value of the system parameter BORROWLIM. The AWSA feature only allows a process to grow above its WSQUOTA value if there are at least as many pages of free memory as specified by the parameter BORROWLIM. In this way, the AWSA feature ties loans to the available capacity.

Even though VAX/VMS may intend to let a process grow during its next adjustment period, if too many processes attempt to add pages at once, an additional mechanism is needed to let VAX/VMS stop the growth while it is occurring. (You could think of it as VAX/VMS needing to renege on its promise. However, VAX/VMS only stops the growth of processes that have already had the benefit of growing beyond their quota.) Therefore, whenever a process page faults after its working set count exceeds its quota, VAX/VMS

examines the value of the system parameter GROWLIM before it allows the process to use more of its WSINC loan. (Note that this activity is not tied into an adjustment period, but rather is an event-driven occurrence based on page faulting.) If there are at least as many pages on the free list as required by GROWLIM, VAX/VMS continues to allow the process to add pages to its working set. If the number of free pages does not equal or exceed GROWLIM, VAX/VMS will not allow the process to grow; the process must give up some of its pages before it reads in new pages.

While some heavily faulting processes are being allowed to extend their working set sizes, processes that are not page faulting heavily can be giving up some of their working set limit through voluntary decrementing. Processes whose page fault rate is lower than PFRATL (when PFRATL is nonzero) are subject to giving up pages. As with growth, this reduction occurs at the next quantum end after AWSTIME has elapsed. The amount of the reduction is defined by the system parameter WSDEC, but no process will be reduced below the minimum size defined by AWSMIN.

Figure 2–4 illustrates how the page fault rate and working set size are related for most processes. Under the influence of the page fault rate values PFR1 and PFR2, the working set size tends to fluctuate between points A and B, under the shaded portion of the graph.

Not all working sets for all images exhibit the same curve as depicted in Figure 2–4. For example, for other images the working sets might behave more like the curves in Figures 2–5 or 2–6. Yet each of these characteristic curves illustrates that, as you decrease the working set size, you should expect the page fault rate to increase. Note that if you establish a maximum acceptable page fault rate of PFR1, for each image there is a minimum required working set size, as shown at point A on each figure. If you determine that the minimum level of page faulting is defined by PFR2 for all images, then for each image there is a point (shown as point B) that is the maximum size the working set needs to reach.

Figure 2–4 Effect of Working Set Size on Page Fault Rate—Graph 1

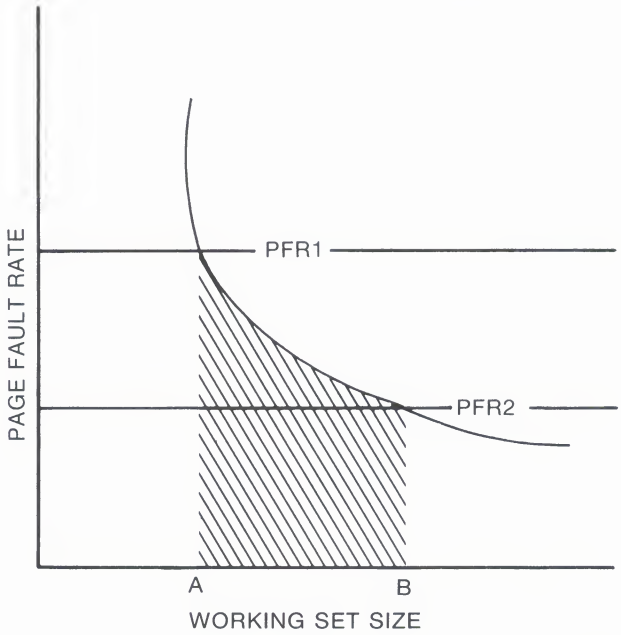


Figure 2–5 Effect of Working Set Size on Page Fault Rate—Graph 2

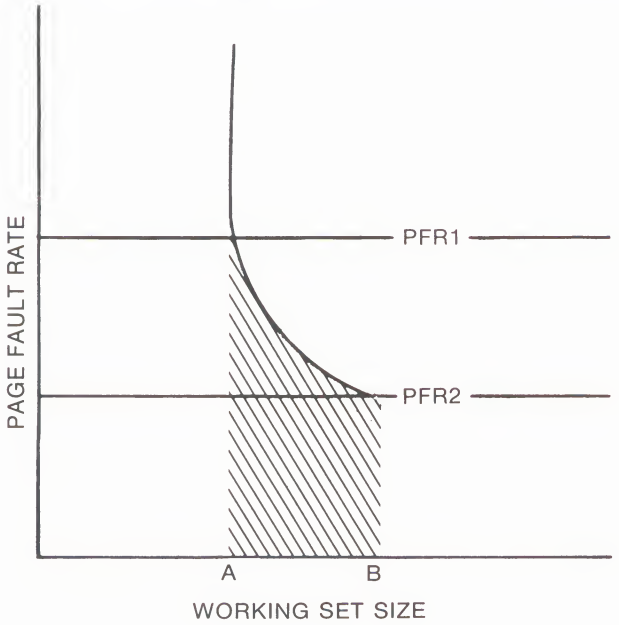


Figure 2–6 Effect of Working Set Size on Page Fault Rate—Graph 3

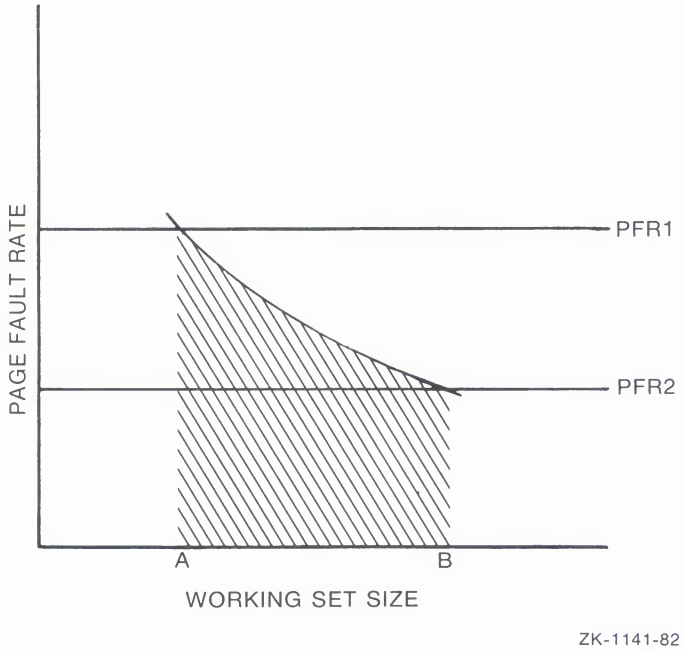
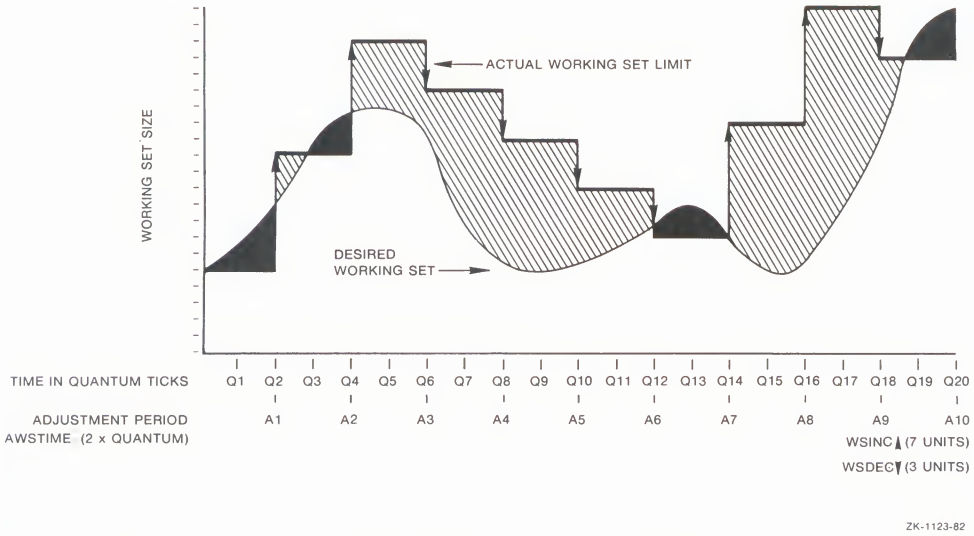


Figure 2–7 illustrates how automatic working set adjustment works over time, across the whole system, to minimize the amount of page faulting by adjusting working set sizes in balance with the amount of free memory available. The units used for AWSTIME, WSDEC, and WSINC in Figure 2–7 are simply for illustration; they are not recommendations.

In the figure, the shaded area identifies where paging occurs. The portion between the desired working set size and the actual working set limit (shown with cross-hatching) represents unnecessary memory overhead—an obvious case where it costs memory to minimize page faulting.

Figure 2–7 An Example of Automatic Working Set Adjustment at Work



One more point should be clarified. At the time an image exits, the process's working set limit drops automatically back to the value of WSDEFAULT.

2.2.1.1 Working Set Sizes

The whole memory management strategy depends initially on the values in effect for the working set quota (WSQUOTA) and working set extent limit (WSEXTENT).

For a process, these values are derived from the User Authorization File (UAF) as initially assigned by the system manager. When establishing values, the system manager makes a conscious decision about the appropriate values for each user, or else lets the system assign the default values defined in the DEFAULT record. (The AUTHORIZE command SHOW DEFAULT displays the default values.) Note that each value in the UAF represents an upper limit for the process. When an interactive process runs, the values in effect may have been lowered or raised either by the corresponding qualifiers on the last SET WORKING_SET command to affect them, or by the system service $ADJWSL.

Subprocesses and detached processes receive their working set characteristics when created by the $CREPRC system service or the DCL command RUN (process). If specific values are not provided at that time, the subprocess or detached process receives default working set characteristics from the corresponding system PQL parameters as follows:

Parameter	Characteristic
PQL_DWSDEFAULT	Default WSDEFAULT
PQL_DWSQUOTA	Default WSQUOTA
PQL_DWSEXTENT	Default WSEXTENT.

PQL parameters are described in the *VAX/VMS System Generation Utility Reference Manual.*

When a batch queue is created, the DCL command INITIALIZE/QUEUE establishes the default values for jobs with the /WSDEFAULT, /WSQUOTA, and /WSEXTENT qualifiers. These qualifiers may, however, be set to defer to the user's values in the UAF. When a batch job runs, the values in effect may have been lowered by the corresponding qualifiers on the DCL commands SUBMIT or SET QUEUE/ENTRY.

The time and forethought you (and your user community) use to establish and maintain appropriate values for WSDEFAULT, WSQUOTA, and WSEXTENT are always returned in better system performance. Remember that WSQUOTA should be large enough that a process can perform reasonably well without a loan, yet small enough that a single process is not guaranteed an inequitable share of memory when memory is tight.

The most cost-effective working set limit lies just above the point where performance drops sharply. A two-phase strategy is recommended for setting working sets at their optimal sizes:

1. Figure initial working set limits for different types of processing on a rule-of-thumb basis.
2. Adjust working set limits based on observed behavior.

2.2.1.2 *Initial* Working Set Limits and Characteristics

For processing that involves system components, the following working set limits are suggested:

- Small (60 to 300 pages)—For editing, and for compiling and linking small programs ("typical" interactive processing)
- Large (350 to 700 pages)—For compiling and linking large programs, and for sorting ("typical" batch processing)

Working set limits for user programs depend on the code-to-data ratio of the program and on the amount of data in the program. Programs that are mostly code—that include a small or moderate amount of data, or use VAX RMS to process data on a per-record basis—should require only a small working set. The amount of code should not matter as long as it is reasonably linear. Programs that may need to manipulate large amounts of data internally (such as sort procedures, linkers, compilers, assemblers, and librarians) typically require large working sets.

The following guidelines are suggested for initial working set characteristics:

- System parameters—Set WSMAX at the highest number of pages required by any program.
- UAF options—For each user, set WSQUOTA at the largest number of pages required by a program that the user will run interactively. Set WSDEFAULT at the median number of pages required by a program that the user will run interactively. Set WSEXTENT at the largest number of pages you anticipate the process will need. Be as realistic as possible in your estimate.
- Batch queues for user-submitted jobs—For each batch queue, set WSEXTENT (using the DCL commands INITIALIZE/QUEUE or START /QUEUE) to the largest number of pages required. Set WSQUOTA to the number of pages that will allow the jobs to complete within a reasonable amount of time. Set WSDEFAULT at the median number of pages required.

This arrangement effectively forces users to submit large jobs for batch processing, since the jobs will not run efficiently interactively. To further restrict the user who attempts to run a large job interactively, you can impose CPU time limits in the UAF.

2.2.1.3 Adjustments to Working Set Sizes

Once you establish initial values according to the guidelines above, you should let the system run and observe its performance. If you observe unsatisfactory performance, refer to Section 5.2.4.2.

2.2.1.4 Adjustments to Automatic Working Set Adjustment Parameters

As indicated in earlier discussions, the delicate mechanism of automatic working set adjustment also depends heavily on the values of the key system parameters of PFRATH, PFRATL, WSINC, WSDEC, QUANTUM, AWSTIME, AWSMIN, GROWLIM, and BORROWLIM. Normally, the default values that the system provides for these parameters correctly match the operational needs.

The parameters PFRATL and WSDEC, which control voluntary decrementing, are very sensitive to the application workload. For the PFRATH and PFRATL parameters,it is possible to define values that may appear to be reasonable page faulting limits, but that yield poor system performance. The problem is due to the VAX/VMS page replacement algorithm and the time spent maintaining the operation within the page faulting limits. For example, for some values of PFRATL, you may observe that a process continuously page faults as its working set size grows and shrinks while the process attempts to keep its page fault rate between the limits imposed by PFRATH and PFRATL. However, you might observe the same process running in approximately the same size working set, without page faulting once, with PFRATL turned off (set to zero). (To prevent sites from encountering an undesirable extreme of oscillation, VAX/VMS turns off voluntary decrementing by initially setting the parameter PFRATL equal to zero. You will only achieve voluntary decrementing if you deliberately turn it on.)

The possibility that automatic working set adjustment parameters are out of balance is so slight that you should not attempt to modify any of the key parameter values without a very thorough understanding of the entire mechanism. Furthermore, some of the system parameters values should be related to others. For example, BORROWLIM should be greater than FREELIM. Such interrelationships are discussed in the *VAX/VMS System Generation Utility Reference Manual*.

Again, it must be emphasized: if you plan to change any of these automatic working set adjustment parameters, review the documentation for all of them, at length, before proceeding. you should also be able to explain why you want to change the parameter(s) and what system behavior you predict will occur. In other words, never make whimsical or guesswork changes to the automatic working set adjustment parameters on a production system.

It is certainly possible for you to turn off borrowing for any process by setting the process WSEXTENT value equal to its WSQUOTA value. It is also possible to circumvent the automatic working set adjustment feature through the DCL command SET WORKING_SET/NOADJUST. Use caution in turning off automatic working set adjustment with the /NOADJUST qualifier, since conditions could arise that would force the swapper to trim the process back to the value of the SWPOUTPGCNT system parameter. (Swapper trimming is fully described in Section 2.2.2.) Once automatic working set adjustment is disabled for a process, the process can not increase its working set size after the swapper trims the process to the SWPOUTPGCNT value. If the value of SWPOUTPGCNT is too low, the process is restricted to that working set size and will fault badly.

You can also turn off automatic working set adjustment on a system-wide basis, simply by setting the value for WSINC to zero. However, you should first think out and properly justify these actions.

2.2.1.5 Performance Management Strategies and Automatic Working Set Adjustment

Earlier discussions in this section have shown the effects that the automatic working set adjustment parameters can have on system behavior. Your site needs to determine which behavior is more desirable under the conditions that prevail with your workload. By developing a strategy for performance management that considers the desired automatic working set adjustment, you will know when the parameters are out of adjustment and how to direct your tuning efforts.

Sites typically choose one of the following two general strategies for tuning automatic working set adjustment parameters:

1. Tune to provide a rapid response whenever the load demands greater working set sizes.
2. Tune for a less dynamic response that will stabilize and track moderate needs for working set growth.

The first strategy works best in the time-sharing environment, where there can be wild fluctuations in demand for memory from moment to moment. The second strategy works better in a production environment where the demand tends to be more predictable and far less volatile.

To implement the highly responsive strategy, you would set PFRATH low (possibly even to zero), set a low value for AWSTIME, and set a relatively large value for WSINC. You would start processes off with small values for their working set defaults, but you would provide for either large working set quotas or generous loans by setting BORROWLIM low and by defining large WSEXTENT values.

To implement the less dynamic strategy, you would establish moderate values for AWSTIME, WSINC, and PFRATH. (For example, you might set WSINC equal to approximately 10% of the typical value for WSDEFAULT on your system.) You might also provide somewhat more generous working set defaults, and you would not need to set BORROWLIM so low as to ensure that loans would always be granted.

2.2.2 VAX/VMS Swapper Trimming

Sometimes, if process requirements so dictate, the VAX/VMS operating system will "swap out" processes to a swapping file on disk so that the remaining processes can have the benefit of the use of memory without excessive page faulting. Swapping refers to writing a process out to a reserved disk file known as a swapping file.

To better balance the availability of memory resources among processes, the operating system normally reclaims memory through a somewhat more complicated sequence of actions than simple swapping. This method of reclaiming memory is known as "swapper trimming." The name indicates that both trimming and swapping of processes may be involved, and that the swapper performs the act.

Swapper trimming can be initiated by VAX/VMS at any time that it detects too few pages in the free page list. That is, whenever the number of free pages drops below the value of the system parameter FREELIM, VAX/VMS will take necessary action to obtain at least as many free pages as specified by the value of the system parameter FREEGOAL.

- First, VAX/VMS checks whether the minimum number of pages exists in the modified page list to make it worthwhile to write them out. This minimum value is dictated by the value of the system parameter MPW_THRESH.
- If the minimum exists, VAX/VMS invokes the modified page writer to write out the modified page list and free its pages for the free page list.

However, if not enough pages could be obtained from the modified page list to match FREEGOAL, VAX/VMS does not activate the modified page writer. Instead, VAX/VMS concludes that some of the processes should be "trimmed," that is, forced to relinquish some of their pages or else swapped out. When this is done, all remaining processes obtain a more equitable chance of getting free pages for their needs.

Trimming takes place on two levels, and it is attempted before the system resorts to swapping. On the process level, the swapper checks for processes that have loans out, that is, processes that have borrowed on their working set extent. Such processes can be trimmed, at the swapper's discretion, back to their working set quota.

Next, if this amount of trimming fails to produce a sufficient number of pages (perhaps because few if any loans were out at the time), the swapper can trim on the second level. Here, the swapper refers to its system-wide trimming value, the system parameter SWPOUTPGCNT. The swapper attempts to trim as many candidates as necessary back to the value of SWPOUTPGCNT, which is the minimum number of pages any process is allowed to retain in memory before it is swapped out. However, because the swapper does not want to trim pages needed by an active process, it selects the processes that are candidates for trimming based on their state. As soon as the needed pages are acquired, the swapper stops trimming on the second level.

If trimming on the second level fails to produce enough pages, the swapper resorts to swapping out processes from its list of likely candidates. Memory is always reclaimed from suspended processes before it is taken from any other processes. The actual algorithm used for the selection of processes in each of these cases is complex, but those processes that are in local event flag wait or hibernate wait states are among the next likeliest candidates.

In addition, as the swapper determines which processes to swap out, it compares the length of real time that a process has been waiting since entering the hibernate or local event flag wait states against the system parameter LONGWAIT. This test allows VAX/VMS to differentiate between those processes that have been idle for some time and are likely to remain idle, and those processes that have not been idle too long and might be likely to become computable sooner. From its candidates, VAX/VMS selects as the better candidates for outswapping those processes that have been idle for as long a time period as specified by LONGWAIT. By freeing up pages through outswapping, VAX/VMS should allow enough processes to satisfy their CPU requirements, so that those processes that were waiting can resume execution sooner.

When the system is lightly loaded, a large, low-priority, compute-bound process may increase its working set beyond the quota value. Prior to VAX/VMS Version 4.0, such processes, as long as they remained computable, were rarely outswapped, even when memory later became overcommitted under more normal system loading conditions, and when the large processes were making little progress (no page faults, no direct or buffered I/O, no CPU time accumulation over a significant time period). Thus, to reclaim memory sufficient to prevent excessive paging by interactive processes, it was sometimes necessary to suspend the large processes.

VAX/VMS Version 4.0 provides an alternative to this procedure by introducing, for purposes of outswap candidate selection, a new *dormant* process "pseudo class." Two criteria define a dormant process:

1 The process must be a non real-time process whose current priority is equal to or less than the SYSGEN parameter DEFPRI (default 4).

2 The process must be a computable process that has not had a significant event (page fault, direct or buffered I/O, CPU time allocation) within an elapsed time period defined by the SYSGEN parameter DORMANTWAIT (default 10 seconds).

After suspended (SUSP) processes, dormant processes are the most likely candidates for memory reclamation by the swapper.

If you understand this memory management mechanism, you can probably appreciate both how it works to reduce the amount of costly swapping that occurs in a system, and how it might, in the worst possible case, turn out to be costly in itself. The worst case occurs when the swapper trims out pages that processes truly need, to the point that they begin to fault heavily and the system-wide paging reaches the saturation point.

You should try to determine for your overall workload a minimum working set size that permits some work to be done reasonably efficiently, but that is below the peak efficiency value. By setting SWPOUTPGCNT to this value, you allow VAX/VMS to efficiently reclaim memory from processes without resorting to swapping processes out of the balance set.

It is possible to turn off the second level of swapper trimming on a system-wide basis, if necessary, by increasing SWPOUTPGCNT to such a large value that second-level trimming is never permitted. However, the swapper will still trim processes that are above their working set quota back to it, as appropriate. If you encounter a situation where any swapper trimming causes excessive paging, you may decide it would be preferable as a corrective action to eliminate second-level trimming and initiate swapping sooner. In this case, you tune the swapping with the SWPOUTPGCNT parameter.

For a process with the PSWAPM privilege, you can also turn off swapping and second-level trimming with the DCL command SET PROCESS /NOSWAPPING.

Be aware that swapper trimming is more beneficial on most systems than voluntary decrementing. The reason lies in the fact that swapper trimming occurs on an as-needed basis, whereas voluntary decrementing occurs on a continuous basis. Furthermore, as previously explained, there is potential for voluntary decrementing to reach a detrimental condition of oscillation. Thus, you will find that the AUTOGEN command procedure establishes parameter values when the system is first installed to provide for swapper trimming but to disable voluntary decrementing.

2.2.3 VAX/VMS Memory Sharing

In simplest terms, memory sharing allows multiple processes to map to (and thereby gain access to) the same page(s) of physical memory. The sharing of memory (either code or data) is accomplished under VAX/VMS through a system-wide global page table that is similar in function to the system page table.

Figure 2–8 Example Without Shared Code

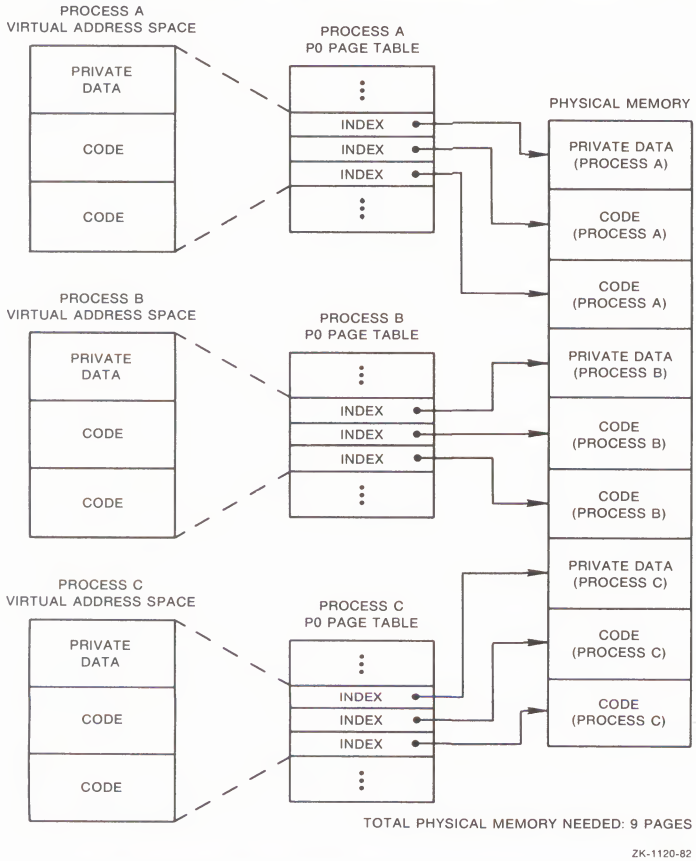
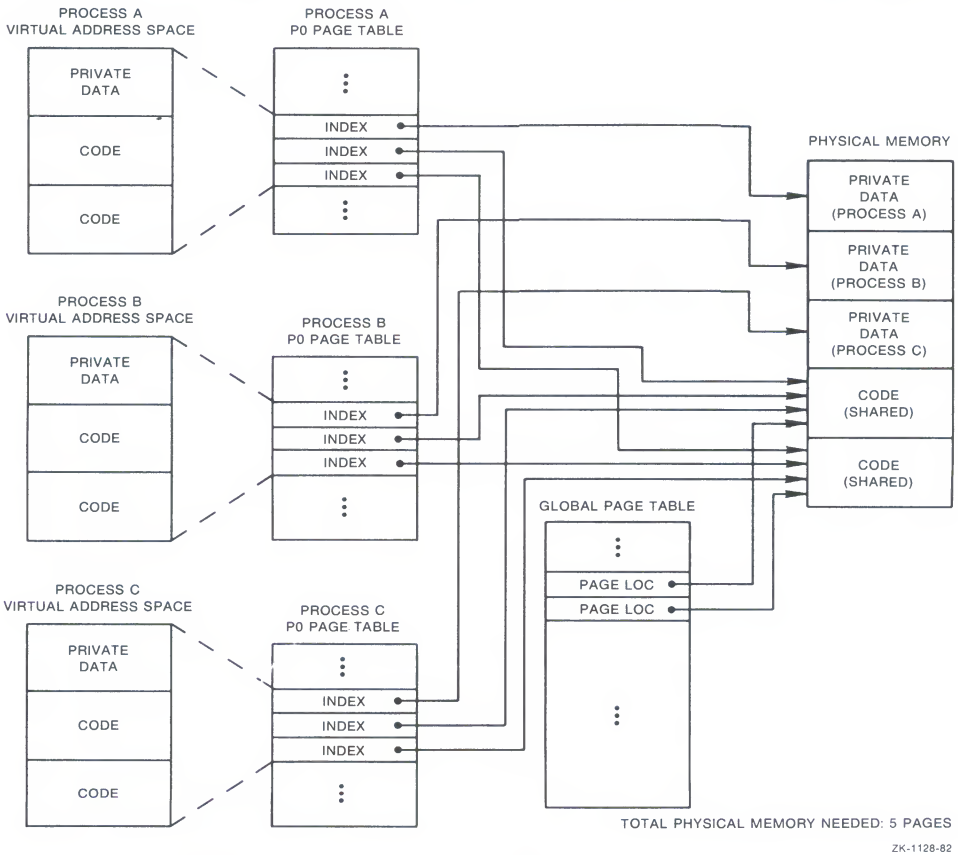


Figure 2–9 Example With Shared Code



Figures 2–8 and 2–9 illustrate how memory can be conserved through the use of global (shared) pages. The three processes (A, B, and C) run the same program, which consists of two pages of read-only code and one page of writeable data. Figure 2–8 shows the virtual-to-physical memory mapping required when each process runs a completely private copy of the program. Figure 2–9 illustrates the physical memory gains possible and the data-structure linkage required when the read-only portion of the program is shared by the three processes. Note that each process must still maintain a private data area to avoid corrupting the data used by the other processes.

The memory that is saved by sharing is the product that results when the number of pages of shared read-only code is multiplied by the number that is one less than the total number of sharing processes. In the example, Figure 2–8 shows that nine pages of memory are required when there is no sharing. However, Figure 2–9 shows that 4 pages of memory are saved by sharing (2 pages of shared code times 3 minus 1 processes). A more realistic example is even more impressive. If 30 users share 300 pages of code concurrently, the savings are 8700 pages.

A small amount of overhead is required to obtain these memory savings. The overhead consists of the data-structure space required for the global page table entries and global section table entries, both of which are needed to provide global mapping. Each global page requires one global page table entry (4 bytes each, allocated from the global page table, which is part of the system process header). Each global section requires a global section table entry (32 bytes in the global section table, which is also part of the system process header) and a global section descriptor (48 bytes, allocated from paged dynamic pool). (For more information on global sections, see the *VAX/VMS Linker Reference Manual*.)

Two system parameters determine the maximum sizes for the two data structures in the system process header. GBLPAGES defines the size of the global page table and GBLSECTIONS defines the size of the global section table. Since these two data structures are part of the system process header, the system working set size (determined by the system parameter SYSMWCNT) must be increased whenever you increase GBLPAGES. The AUTOGEN command procedure automatically increases the value of the system parameter SYSMWCNT by one for every 128 pages you add to GBLPAGES.

Once a shareable image has been created it can be installed as a permanently shared image. (See the *VAX/VMS Linker Reference Manual* and the *VAX/VMS Install Utility Reference Manual*). Memory will only be saved, however, when there is more than one process actually mapped to the image at a time.

Remember, also, to use AUTHORIZE to increase the user's working set characteristics (WSDEF, WSQUO, WSEXTENT), wherever appropriate, to correspond to the expected use of shared code. (Note, however, that this increase does not mean that the actual memory usage will increase. Sharing of code by many users actually decreases the memory requirement.)

If physical memory is especially limited, you may decide to investigate whether there is much concurrent image activation that results in savings. If you find there is not, there is no reason to employ code sharing. You can use the following procedure to determine if there is active sharing on image sections that have been installed as shareable.

1 Invoke the Install Utility and issue the LIST/FULL command. For example:

```
$ RUN SYS$SYSTEM:INSTALL
INSTALL> LIST/FULL LOGINOUT
```

INSTALL displays information in the following format:

```
DISK$VAXVMSRL4:<SYSO.SYSEXE>.EXE
    LOGINOUT;3        Open Hdr     Shar Priv
        Entry access count          = 44
        Current / Maximum shared    = 3 / 5
        Global section count        = 2
        Privileges = CMKRNL SYSNAM TMPMBX EXQUOTA SYSPRV
```

2 Observe the values shown for the Current/Maximum shared access counts:

- The Current value is the current count of concurrent accesses of the known image.
- The Maximum value is the highest count of concurrent accesses of the image since it became known (installed). This number appears only if the image is installed with the /SHARED qualifier.

The Maximum value should be at least 3 or 4. A lower value indicates that overhead for sharing is excessive.

In general, your intuition, based on knowledge of the workload is the best guide. Remember that the overhead required to share memory is counted in bytes of memory, while the savings are counted in pages of physical memory. Thus, even if you merely suspect there is occasional concurrent use of an image, the investment required to make it shareable is worthwhile.

2.2.4 VAX/VMS Scheduling

The VAX/VMS scheduler controls both when and how long a process executes. For this reason, it has impact on the demand on the CPU, an impact that can ultimately affect system performance. It is important to understand the role of the scheduler as you analyze system performance and consider ways to improve it.

The scheduler tries to obtain maximum performance from concurrently executing processes. To achieve this, the scheduler rotates the control of the CPU among the processes that are computable, so that all the computing processes receive frequent and equitable chances to complete their processing requirements. As part of the optimization, it allows operations to overlap. For example, if a process must wait for an I/O completion, another process is allowed to run.

The VAX/VMS scheduler uses a modified round-robin form of scheduling: processes receive a chance to execute on a rotating basis, according to process priority and state. At some point, each computable process receives a time slice for execution. (The time slice equals the system parameter QUANTUM, and rotating the time slices among the processes is called time-slicing.) Once its quantum starts, each process executes until

- A process of higher priority becomes computable
- The process is no longer computable because of a resource wait
- The process itself voluntarily enters a wait state
- The quantum ends

If there is no other computable process at the same priority ready to execute when the quantum ends, the current process receives another time slice.

A change in process state causes the scheduler to reexamine which process should be allowed to run. For example, a process that completes a terminal input operation changes from local event flag wait state to the computable state. When required to select the next process for scheduling, the scheduler examines the priorities assigned to all the processes that are computable and selects the process with the highest priority. Priorities are numbers from 0 through 31. Processes with priorities of 16 and above receive maximum access to the CPU resource (even over system processes) whenever they are computable. (These priorities, therefore, are used for real-time processes.)

Another important scheduler feature is priority boosting. For processes below priority 16, the scheduler can increase and decrease process priorities. While processes run, the scheduler recognizes events such as I/O completions, the completion of an interval of time, and so forth. As soon as one of the recognized events occurs and the associated process becomes computable, the priority of that process may be increased. The amount of the increase is related to the associated event. For example, if the event is the completion of terminal I/O input, a large increase is given, so that the process can run again sooner. Then the scheduler examines which computable process has the highest priority, and, if necessary, causes a *context switch* so that the highest priority process runs. As soon as a process is scheduled, its priority is reduced by one, to allow processes that have received a priority boost to begin to return to their base priority. However, the priority is never decreased below the base priority or into the real-time range.

For real-time processes (those with base priorities in the 16 to 31 range) there are some special distinctions. These processes never receive a priority boost, nor do they experience automatic working set adjustments or quantum-based time-slicing. Although quantum-based sharing of the processor works well for other processes, VAX/VMS permits real-time processes to run until either

they voluntarily enter a wait state or a higher priority real-time process becomes computable.

From a tuning standpoint, you have very few controls you can use to influence process scheduling. You can modify the base priorities of processes and you can modify the length of time for a quantum. All other aspects of process scheduling are fixed by the behavior of the scheduler and the characteristics of the workload.

A process receives a default base priority from the /PRIO qualifier in the user's UAF record, or from the DEFAULT record in the UAF. A process can also change its priority with the system service $SETPRI. With the DCL command SET PROCESS/PRIORITY, users can always reduce the priority of their own processes. However, users need the ALTPRI privilege to use the same command to increase the priority of their processes. A user requires privilege (GROUP or WORLD) to change the priority of other processes.

A detached process or subprocess receives its base priority when created by the $CREPRC system service or the DCL command RUN. If no priority is specified, the priority of the creator is used.

When a batch queue is created, the DCL command INITIALIZE/QUEUE /PRIORITY establishes the default priority for jobs. However, when the user submits a job with the DCL command SUBMIT or changes characteristics of that job with the DCL command SET QUEUE/ENTRY, the user can adjust the priority with the /PRIORITY qualifier. (With either command, increases are permitted only for submitters with the OPER privilege.)

3 Managing System Resources

For practical purposes, managing the performance of a VAX/VMS system is best approached as managing its resources. These include both hardware components (CPU, memory, peripherals), and software (application programs, optional products, and operating system facilities such as the Extended QIO Processor (XQP), and memory and I/O management mechanisms).

If you have properly configured your system and installed it using the AUTOGEN command procedure, and if you have followed the workload management recommendations in Chapter 1, you will rarely need to adjust system values. Good management practice dictates, however, that you evaluate your system both on a routine basis and in the following special circumstances:

- Whenever a significant alteration occurs in the system hardware or software environment, such as an update or an upgrade
- Whenever the workload changes in kind or amount
- Whenever you suspect a performance problem

This approach to performance management will allow you to collect and become familiar with the statistics that represent "normal" behavior for individual resources and for the system as a whole. You can then respond with confidence to reported problems, recognize changes in the system's behavior, and make decisions about possible hardware changes.

Bear in mind, however, that a statistic that represents normal behavior on one system may be unreasonable for a different one, because CPU speed, workload, and user expectations can vary widely from system to system.

This chapter describes procedures to help you evaluate the performance of the CPU, memory, and disk I/O subsystem resources using the Monitor Utility and (to a lesser extent) other standard VAX/VMS utilities. Discussions focus on the utilization of each hardware resource by major VAX/VMS software components and on the the measurement, analysis, and possible reallocation of the hardware resources. Suggestions for corrective actions are provided, in case your evaluation indicates that improvements are possible.

Section 3.2 describes the use of the Accounting Utility to obtain image-level accounting data and provides guidelines for interpreting the data. Section 3.3 explains how to maintain and interpret MONITOR summary reports. The three major VAX/VMS hardware resources are examined in the following sections:

- **The CPU Resource**—Section 3.5
- **The Memory Resource**—Section 3.6
- **The Disk I/O Resource**—Section 3.7

Table 3–2 in Section 3.8 summarizes important MONITOR statistics and includes rules of thumb for evaluating the performance of the resources characterized by those statistics.

In summary, the purpose of this chapter is to help you verify that your system is performing well and to provide information to aid you in maintaining performance at an acceptable level. Detailed problem analysis of a system that already performs well is not discussed. For information on detailed problem analysis, refer to Chapters 4 and 5.

3.1 Evaluation Ground Rules

As you conduct your evaluations, keep the following ground rules in mind:

- Complete the entire evaluation. It is important to examine all the resources in order to evaluate the system as a whole. A partial examination may lead you to attempt an improvement in an area where it may have minimal effect, because more serious problems exist elsewhere.
- Become as familiar as possible with the applications running on your system. Get to know what their resource requirements are. You can obtain a good deal of relevant information from the ACCOUNTING image report described in Section 3.2. User's guides associated with DIGITAL or third-party software can also be very helpful in identifying resource requirements.
- If you believe that a change in software parameters or hardware configuration may improve performance, execute such a change cautiously, being sure to make only one change at a time. Evaluate the effectiveness of the change before deciding to make it permanent.

Note

When specific values or ranges of values for MONITOR data items are recommended, they are intended only as general guidelines and will not be appropriate in all cases.

For purposes of the following discussion, it is assumed that the workload management techniques and installation guidelines described in Chapter 1 have been followed. It is further assumed that your system can be classified as a general timesharing system. Some information in this chapter may not apply to certain specialized types of systems or to applications such as workstations, database management, real-time operations, transaction processing, or any system in which a major software subsystem is in control of resources for other VAX/VMS processes.

Note that the procedures outlined in this chapter differ from those in Chapters 4 and 5 in the following ways:

- They are designed to help you conduct an evaluation of your system and its resources, rather than to execute an investigation of a specific problem. If you discover problems during an evaluation, you can refer to the decision-tree diagrams in Chapters 4 and 5 for further analysis.
- For simplicity, they are less exhaustive, relying on certain rules of thumb to evaluate the major hardware resources and to point out possible deficiencies, but stopping short of pinpointing exact causes.
- They are centered around the use of the Monitor Utility, particularly the summary reports, both standard and multifile.

3.2 Collecting and Interpreting Image-Level Accounting Data

Image-level accounting information can be quite useful in helping you to gain an understanding of resource utilization on a per-image basis. By knowing which images are heavy consumers of resources at your site, you can better direct your efforts toward controlling them and the resources they consume. Images used frequently are typically good candidates for code sharing, whereas images that consume large quantities of various resources may be forced to run in a batch queue, where the number of simultaneous processes can be controlled. Example 3–1 illustrates the type of accounting report that can provide the resource utilization information necessary to manage images.

Note

This example assumes that image-level accounting records have been collected previously. (You enable image-level record collection by issuing the DCL command SET ACCOUNTING /ENABLE=IMAGE.) Since the collection of image-level accounting data consumes CPU cycles, and since the collected records can consume a significant amount of disk space, remember to enable image-level accounting only for the period of time needed for the report. After you have collected the data you need, issue the DCL command SET ACCOUNTING/ DISABLE=IMAGE to disable image-level accounting.

A series of commands like the following will generate output similar to that shown in Example 3–1:

```
$ ACCOUNTING  /TYPE=IMAGE  /OUTPUT=BYNAM.LIS -
_$ /SUMMARY=IMAGE -
_$ /REPORT=(PROCESSOR,ELAPSED,DIRECT_IO,FAULTS,RECORDS)
$ SORT  BYNAM.LIS  BYNAM.ORD  /KEY=(POS=16,SIZ=13,DESCEND)
   .
   .
   .
(Edit BYNAM.ORD to relocate heading lines)
   .
   .
   .
$ TYPE BYNAM.ORD
```

Example 3–1 Sample Image-Level Accounting Report

```
From:  8-MAY-1986 11:09                                    To:  8-MAY-1986 17:31
Image name      Processor Time      Elapsed Time      Direct I/O   Page Faults   Total Records
--------------------------------------------------------------------------------------------
EDT             0 00:34:21.34   0 15:51:34.78          5030        132583          390
DTR32           0 00:19:30.94   0 03:17:37.48          7981         83916           12
PASCAL          0 00:15:19.42   0 01:04:19.57         38473        143107           75
MAIL            0 00:10:40.88   1 02:54:02.89         26139        106854          380
VAX11C          0 00:05:57.31   0 00:19:13.59           426         23180           26
LINK            0 00:05:44.41   0 00:23:54.54          7443         57092          111
RTPAD           0 00:04:58.40   0 20:49:19.24           668          8004           72
LOGINOUT        0 00:04:53.98   0 02:01:31.81          2809         67579          893
EMACS           0 00:04:30.40   0 05:25:01.37           420          8461            1
MACRO32         0 00:04:26.22   0 00:14:55.00          1014         34016           46
BLISS32         0 00:03:45.80   0 00:12:58.87            98         32797            8
DIRECTORY       0 00:03:26.20   0 01:22:34.47          1020         27329          275
FORTRAN         0 00:03:13.87   0 00:14:15.08          1157         28003           47
NOTES           0 00:01:39.90   0 02:06:01.95          8011          6272           32
DELETE          0 00:01:37.31   0 00:57:43.31           834         25516          332
TYPE            0 00:01:06.35   0 00:28:58.26           406         14457          173
COPY            0 00:00:57.08   0 00:11:11.40          2197          4943           42
SHOW            0 00:00:56.39   0 00:24:53.22            23         11505          166
ACC             0 00:00:54.43   0 00:03:41.46           132          2007            7
MONITOR         0 00:00:53.91   0 02:37:13.84           159          5649           40
CALENDAR        0 00:00:43.55   0 00:30:15.52          1023          3557           25
PHONE           0 00:00:40.56   0 00:54:59.39            24          1510           33
ERASE           0 00:00:37.88   0 00:03:51.04           105          9873          113
LIBRARIAN       0 00:00:35.58   0 00:03:37.98          1134         10297           62
FAL             0 00:00:34.27   0 00:20:56.63           110          4596          122
SDA             0 00:00:27.34   0 00:09:28.68            52          4797            3
SET             0 00:00:27.02   0 00:02:30.28           160          9447          206
NETSERVER       0 00:00:26.89   0 02:38:17.90           263         10164          407
CDU             0 00:00:24.32   0 00:01:57.67            13         21906           17
VMSHELP         0 00:00:12.83   0 00:05:40.96           121          1943           14
RENAME          0 00:00:09.56   0 00:00:57.44             6          3866           47
SDL             0 00:00:09.55   0 00:01:19.78            11          3158            4
SUBMIT          0 00:00:08.14   0 00:01:08.50             9          2991           28
NCP             0 00:00:07.30   0 00:02:26.20             7          1765           16
QUEMAN          0 00:00:06.44   0 00:01:38.75           201          1561           20
```

This example shows a report of system resource utilization for the indicated period, summarized by unique image name, in descending order of CPU utilization. Only the top 35 CPU consumers are shown. (The records could just as easily have been sorted differently.) The Total Records column is a count of image terminations, requested by specifying the RECORDS report key in the ACCOUNTING command that generated the report. The /SINCE and /BEFORE qualifiers may be specified to select any time period of interest.

The From: and To: timestamps show that the accounting period ran for 6 1/2 hours and included an entire afternoon. Assume that this report represents a typical workload for the installation, and examine the data in the various columns.

- Image Name—Most image names are those of programming languages and operating system utilities, indicating that the report was probably generated in a program development environment.
- Processor Time—Data in this column shows that no single image is by far the highest consumer of the CPU resource. It is therefore unlikely that the installation would benefit significantly by attempting to reduce CPU utilization by any one image.
- Direct I/O—In the figures for Direct I/O, you can see that there are two top images, PASCAL and MAIL. One way to compare them is by calculating I/O operations per second. The total elapsed time spent running PASCAL is roughly 3860 seconds, while the time spent running MAIL is a little under 96843 seconds (several people used MAIL all afternoon). Calculating on a time basis, then, MAIL caused roughly 1/3–1/4 of an I/O operation per second, whereas PASCAL caused about 10 operations per second.

  Note that by the same calculation, LINK caused about 5 I/O operations per second. It would appear that a sequence of PASCAL/LINK commands contributes somewhat to the overall I/O load. One possible approach from here would be to look at the VAX RMS buffer parameters set by the main PASCAL users. You can find out who used PASCAL and LINK by issuing a DCL command of the form:

```
$ ACCOUNTING /TYPE=IMAGE -
_$ /IMAGE=(PASCAL,LINK) -
_$ /SUMMARY=(IMAGE,USER) -
_$ /REPORT=(ELAPSED,DIRECT)
```

  This command selects image accounting records for the PASCAL and LINK images by image name and user name, and requests Elapsed Time and Direct I/O data. You can examine this data to determine whether the users are employing VAX RMS buffers of appropriate size. DIGITAL recommended that two fairly large buffers be used for sequential I/O, perhaps in the range of 12 to 32 blocks each.
- Page Faults—As with direct I/O, page faults are best analyzed on a time basis. One technique is to compute faults-per-10-seconds of processor time and compare the result with the value of the SYSGEN parameter PFRATH. A little arithmetic shows that on a time basis PASCAL is incurring more than 1555 faults per 10 seconds. Suppose that the value of PFRATH on this system is 120 (120 page faults per 10 seconds of processor time), which is considered typical in most environments. What can you conclude by comparing the two values?

Whenever a process's page fault rate exceeds the PFRATH value, VAX/VMS memory management attempts to increase the process working set, subject to system management quotas, until the fault rate declines below PFRATH. So if an image's fault rate is persistently greater than PFRATH, it is not obtaining all the memory it needs.

Clearly, the PASCAL image is causing many more faults per CPU second than would be considered normal for this system. You should therefore make an effort to examine the working set limits and working set adjustment policies for the PASCAL users. To lower the PASCAL fault rate, the process working sets must be increased—either by adjusting the appropriate UAF quotas directly, or perhaps by setting up a "PASCAL batch queue" with generous working set values.

- Total Records—These figures represent the count of activations for images run during the accounting period: in other words, they show each image's relative "popularity." You can use this information to ensure that the most popular images are installed (see Section 1.6). For customer applications, you might consider linking options such as /NOSYSSHR and reassigning PSECT attributes to speed up activations (see the *VAX/VMS Linker Reference Manual*).

  Note that the number of LOGINOUT activations far exceeds that of all other images. This situation could result from a variety of causes, including attempts to breach security, an open terminal line, a runaway batch job, or a large number of network operations. Further ACCOUNTING commands would be necessary to determine the exact cause. At this site, it turned out that most of the activations were caused by an open terminal line. The problem was detected by an astute system manager, who checked the count of LOGFAIL entries in the ACCOUNTING log file.

  You can also use information in this field to examine the characteristics of the "average" image activation. That knowledge would be useful if you wanted to determine whether it would be worthwhile to set up a special batch queue.

  For example, the "average" PASCAL image uses 51 seconds of elapsed time, and the "average" LINK uses 13 seconds. You can therefore infer that the "average" PASCAL and LINK sequence takes about a minute. This information could help you persuade users of those images to run PASCAL and LINK in batch mode. If, on the other hand, the average time were only 5 seconds, batch processing would probably not be worthwhile.

3.3 Maintaining and Interpreting MONITOR Summaries

As a foundation for the evaluation strategy discussed in this chapter, you must develop a database of performance information for your system by running MONITOR continuously as a background process. The SYS$EXAMPLES directory provides three command procedures you can use to establish the database. Instructions for installing and running the procedures are contained in the comments at the beginning of each one. Following is a brief summary of these procedures:

- SUBMON.COM—Starts MONITOR.COM as a detached process. You should invoke SUBMON.COM from the DCL procedure SYS$MANAGER:SYSTARTUP.COM.
- MONITOR.COM—Creates a summary file from the recording file of the previous boot, then begins recording for this boot. The recording interval is 10 minutes.
- MONSUM.COM—Generates two VAXclusterwide multifile summary reports; one for the previous 24 hours, and one for the previous day's prime-time period (9 A.M. to 6 P.M.). These are mailed to the system manager, and then the procedure resubmits itself to run each day at midnight.

While MONITOR data is recorded continuously, a summary report can cover any contiguous time segment. The command file MONSUM.COM, which is executed every midnight, generates and mails the two multifile summary reports described above. These reports are not saved as files, so if you wish to keep them, you must either extract them from your mail file or alter the MONSUM.COM command procedure to save them. Example 3–3 in Section 3.8 is a typical "prime-time" VAXcluster multifile summary.

The report you require for the evaluation procedure is one that covers a period you feel best represents the typical operation of your system. You may wish, for example, to evaluate your system only during hours of peak activity. To generate a summary of the appropriate time segment, edit the MONSUM.COM procedure and change the beginning and ending times on one of the two MONITOR commands that produce the summary reports.

Note

The summary reports produced by MONSUM.COM are in the multifile summary format—there is one column of averages for each node in a VAXcluster, as well as some overall "row statistics". For noncluster systems, the row statistics may be ignored.

If you prefer to use a report in the standard summary format (which includes current, minimum, and maximum statistics), execute a MONITOR playback summary command referencing the input data file of interest as the only file in the /INPUT list. Note that a new data file is created for each system whenever it reboots. Remember to use the /BEGINNING and /ENDING qualifiers to select the desired time period.

Once you have printed a copy of a MONITOR summary report, you are ready to begin the evaluation of your system. Keep in mind, however, that although the discussions in this chapter focus on summary reports, you are encouraged to observe current system activity regularly by running MONITOR in live mode. In live mode, always begin an analysis with the MONITOR CLUSTER and MONITOR SYSTEM classes to obtain an overview of system performance; then monitor other classes to examine components of particular interest. Note that all references to MONITOR items in this chapter are assumed to be for the *average* statistic, unless otherwise noted.

In multifile reports, a page or more is devoted to each MONITOR class. Each column represents one node and is headed by the node name and beginning and ending times of the segment requested. In most cases, time segments for all nodes will be roughly the same. Differences of a few minutes are typical, because data collection on the various nodes is not synchronized.

In some cases, one or more time segments will be shorter than others; in these cases, some of the requested data was not recorded (probably because the nodes were unavailable). Note that if data is unavailable for some period within the bounds of a request, that fact is not explicitly specified.

However, such a gap can occur only when the column of data uses more than one input file; and if multiple files contributed to the column, the number is shown in parentheses to the right of the node name. In cases where a time segment is missing, this number must be greater than 1. If no number appears, there is only one input data file for that column, and the column therefore includes no missing time segments.

To summarize: if all beginning and ending times are not roughly the same, or if a parenthesized number appears, some data may be unavailable, and you may want to base your evaluation on a different time segment that includes more complete data. Whenever the multifile report is based on incomplete data, the Row Average statistic may be weighted unfairly in favor of one or more nodes.

Note

While interpreting MONITOR statistics, keep in mind that the collection interval has no effect on the accuracy of MONITOR *rates*. It does, however, affect *levels*, because these represent sampled data. In other words, the smaller the collection interval, the more accurate MONITOR level statistics will be.

(For more information on MONITOR rates and levels, refer to the *VAX/VMS Monitor Utility Reference Manual.*)

Although the interval value supplied with MONITOR.COM is adequate for most purposes, it does represent a tradeoff between statistical accuracy and the consumption of disk space. Thus, before you base major decisions on MONITOR level statistics, be sure to verify them by running MONITOR for a time with a much smaller collection interval, while carefully observing disk space usage.

3.4 Understanding System Responsiveness

Overall responsiveness of a system depends largely on the responsiveness of its CPU, memory, and disk I/O resources. If each resource responds satisfactorily, then so will the entire system.

Not only must each resource operate efficiently by itself, but it must also interact with other resources. A resource that is not performing well can cause system degradation not only because of its own poor performance, but because other resources that depend on it will operate less efficiently as well. Thus, an important aspect of your evaluation is to distinguish between resources that may be performing poorly because they are overcommitted, and those that may be doing so because one or both of the following conditions has occured:

- They are blocked by the overcommitted resource.
- They are incurring additional overhead operations caused by the overcommitted resource.

An overcommitted resource that causes the others to be blocked or burdened with overhead operations is said to be a system's "binding resource" or "bottleneck". Proper identification of such a resource is critical to correction of a performance problem. Upgrading a nonbinding resource will do nothing to improve a bottlenecked system.

Detection of bottlenecks is particularly important for interactions of the CPU with each of the other resources. CPU blockage occurs when CPU capacity, though it appears sufficient to meet demand, cannot be used because the CPU must wait for disk I/O to complete or memory to be allocated. You should make every effort to maintain sufficient disk I/O and memory capacity so that the CPU is not unduly blocked.

Because of the potential for bottlenecks, it is especially important to maintain balance among the capacities of your system's resources. When upgrading to a faster CPU, it is wise to consider the effect the additional CPU power will have on the other primary resources. For example, since the faster CPU can

initiate more I/O requests per unit of time, you must ensure that the disk I/O subsystem has sufficient capacity to handle the increased traffic.

3.4.1 Evaluating and Improving Responsiveness of System Resources

The following sections describe procedures for evaluating the CPU, memory, and disk I/O subsystem resources, and for improving the responsiveness of those resources should your evaluation indicate that improvement is possible. For each resource, key MONITOR statistics are identified and discussed with a view to helping you answer such questions as the following:

- How well is the resource responding to requests for service?
- How well is the capacity of the resource meeting demand?
- Does the resource have any excess capacity, and if so, can that capacity be attributed to blockage by another, overcommitted resource?

Two prime measures of resource responsiveness are the size of the queue of requests for service (compute queue) and the amount of time it takes the system to respond to those requests (response time). For each resource, you can use MONITOR summaries to examine or estimate one or both of these quantities.

In addition, you can investigate four possible ways to improve responsiveness:

- **Equitable sharing**—Is the resource shared equitably among processes?
- **Reduction of resource consumption by the system**—Can the system's consumption of a resource be reduced, thereby making available more of that resource to users? The effective amount of a resource available to users is that remaining after the operating system has used its portion. Remember that although the operating system performs essential services for users, it consumes the same hardware resources users need to do their work.
- **Load balancing**—How well distributed is the demand for a resource? Can overall system responsiveness be improved, either by reconfiguring hardware or by better distributing the demand for it?
- **Offloading**—Can overall system responsiveness be improved by offloading some of the activity on a resource to other less heavily utilized resource types? For example, excess memory capacity is often used to reduce the demand on an overworked disk I/O subsystem by increasing the size of each I/O transfer, thereby reducing the total number of I/O operations. The CPU benefits as well, because it needs to do less work executing system service and device driver software. The primary means of offloading I/O to memory is the extensive use in the system of caches

(page caches, XQP caches, VAX RMS blocking) to reduce the number of I/O operations.

If the responsiveness of a resource that is performing poorly cannot be improved by these methods, you should consider augmenting its capacity with additional or upgraded hardware.

3.5 Understanding the CPU Resource

The CPU is the central resource in your system, and it is the most costly to augment. Good CPU performance is vital to that of the system as a whole, because the CPU performs the two most basic system functions: it allocates and initiates the demand for all the other resources, and it provides instruction execution service to user processes.

3.5.1 Evaluating CPU Responsiveness

You can assess CPU responsiveness by first observing the average size of the compute queue and then examining idle time and process scheduling wait states to estimate available CPU capacity.

3.5.1.1 The Compute Queue

The MONITOR statistics of interest in this section are as follows:

- STATES—Number of processes in Compute (COM) and Compute (Outswapped) (COMO) scheduling states

Since only one VAX/VMS process can execute on a CPU at any given time, the CPU resource must be shared in a sequential fashion: the system allocates it for a period of time known as a *quantum* to each process that is not waiting for other resources.

During its quantum, a process may execute until any of the following events occur:

- The process is preempted by a higher-priority process.
- The process voluntarily yields the CPU by requesting a wait state for some purpose (for example, to wait for the completion of a user I/O request).
- The process enters an involuntary wait state, such as when it triggers a hard page fault (one that must be satisfied by reading from disk).

Since several processes may be ready to use the CPU at any given time, and since the CPU can be allocated to only one process at a time, the system maintains a queue of processes waiting for the CPU. Such processes are in the Compute (COM) or Compute (Outswapped) (COMO) scheduling states. A good measure of CPU response is the average number of processes in these two states over time—that is, the average length of the compute queue.

If the number of processes in the compute queue is close to 0, unblocked processes will rarely need to wait for the CPU. In typical cases, the larger this number, the longer the processes must wait for service. An exception occurs when the processes in the compute queue are at different priorities. A given process must then wait only for other processes of equal or greater priority.

Several factors, including interrupt stack time, the computing requirements of the processes in the compute queue, CPU type, and scheduling priority, determine how long any given process must wait to be granted its quantum of CPU time. The effect of a large compute queue is worst when the COM processes are strictly compute bound, because they may retain the CPU for the entire quantum period. This situation represents the worst-case delays in acquiring the CPU.

Assuming no interrupt stack time, each of several compute-bound processes of the same priority (one in CUR state and the others in COM state) will acquire the CPU once every second (with the default QUANTUM value of 200 milliseconds). As the number of such processes increases, there is a proportional increase in the waiting time. But if the processes are not compute bound, they may relinquish the CPU before having consumed their entire quantum period, thus reducing waiting time for the CPU.

Because of MONITOR's sampling nature, the utility rarely detects processes that remain only briefly in the COM state. Thus, if MONITOR shows COM processes, you can assume they are the compute-bound type. (Note, however, that the NULL job is always in COM state.)

Try to keep in mind that perceived response time is a subjective issue. Notions of "fast" and "slow" are rarely consistent among any user population. The best way to determine a reasonable length for the compute queue at your site is to note its length during periods when all the system's resources are performing adequately, and users perceive response time to be satisfactory. Then watch for deviations from this value and try to develop a sense for acceptable ranges.

Be sure, however, that the other resources are performing adequately. Otherwise, they could block or induce overhead on the CPU, thereby affecting its responsiveness. If that situation occurs, you may find it difficult to correlate the length of the compute queue with overall system performance.

There are several ways to attempt to make more of the CPU available to processes by shortening the average length of the compute queue. These are discussed in Section 3.5.2. If you are unsuccessful in all of these attempts, the only way to improve responsiveness is to increase CPU capacity either through a hardware upgrade, or by adding one or more CPUs to your VAXcluster.

3.5.1.2 Estimating Available CPU Capacity

The MONITOR statistics of interest in this section are as follows:

- STATES—All items
- MODES—Idle time

To estimate available CPU capacity, observe the average amount of idle time and the average number of processes in the various scheduling wait states. While idle time is a measure of the percentage of unused CPU time, the wait states indicate the reasons that the CPU was idle, and may point to utilization problems with other resources.

Before using idle time to estimate growth potential or as an aid to balancing the CPU resource among processes in a VAXcluster, ensure that the other resources are not overcommitted, thereby causing the CPU to be underutilized. MONITOR data on the scheduling wait states provides clues about potential problems with the memory and disk I/O resources. You should, however, also conduct the complete evaluations described in Sections 3.6 and 3.7.

Whenever a process enters a scheduling wait state—a state other than CUR (process currently using the CPU) or COM (computable process)—it is said to be blocked from using the CPU. Most times, a process enters a wait state as part of the normal synchronization that takes place between the CPU and the other resources. But certain wait states can indicate problems with those other resources that could block viable processes from using the CPU. It is important to realize that performance problems caused by CPU blockage can be corrected only by improving the responsiveness of the resource causing the blockage.

Scheduling wait states may be categorized as *voluntary* or *involuntary*. Processes enter *voluntary* wait states directly; they are placed in *involuntary* wait states by the system. These two types of wait states are discussed in Sections 3.5.1.2.1 and 3.5.1.2.2. Outswapped states are associated with memory management and are discussed in Section 3.6.1.2.

3.5.1.2.1 Voluntary Wait States

The MONITOR statistics of interest in this section are as follows:

- STATES—Number of processes in Local Event Flag Wait (LEF), Common Event Flag Wait (CEF), Hibernate (HIB) and Suspended (SUSP) states
- LOCK—ENQs Forced To Wait Rate

Processes in the Local Event Flag Wait (LEF) state are said to be "voluntarily" blocked from using the CPU; that is, they are temporarily requesting to wait before continuing with CPU service. Since the LEF state can indicate conditions ranging from "normal" waiting for terminal command input to waiting for I/O completion or locks, you can obtain no useful information about potentially harmful blockage simply by observing the number of processes in that state. You can usually assume, though, that most of them are waiting for terminal command input (at the DCL prompt).

Some processes may enter the LEF state because they are awaiting I/O completion on a disk or other peripheral device. If the I/O subsystem is not overloaded, this type of waiting is temporary and inconsequential. If, on the other hand, the I/O resource, particularly disk I/O, is approaching capacity, it could very well be causing the CPU to be seriously underutilized. Long disk response times are the clue that certain processes are in the LEF state because they are experiencing long delays acquiring disk service. If your system exhibits unusually long disk response times, refer to Section 3.7.3 and try to correct that problem before attempting to improve CPU responsiveness.

Still other processes in the LEF state may be waiting for a lock to be granted. This situation may arise in environments where extensive file sharing is the norm—particularly in VAXclusters. Check the ENQs Forced to Wait Rate. (This is the rate of $ENQ lock requests forced to wait before the lock was granted.) Since the statistic gives no indication of the duration of lock waits, it does not provide direct information about lock waiting. But a value significantly larger than your system's normal value may be a clue that users will start to notice delays. If you suspect that the lock waiting is caused by file sharing (VAX RMS and the XQP use locks to synchronize record and file access), attempt to reduce the level of sharing, if possible. If you suspect that it results from user or third-party application locks, attempt to influence the redesign of such applications.

Processes may also enter the LEF state or the other voluntary wait states (Common Event Flag Wait (CEF), Hibernate (HIB) and Suspended (SUSP)) when system services are used to synchronize applications. Such processes have temporarily abdicated use of the CPU; they do not indicate problems with other resources.

3.5.1.2.2 Involuntary Wait States

The MONITOR statistics of interest in this section are as follows:

- STATES—Number of processes in miscellaneous resource wait (MWAIT) state
- PROCESSES—Types of resource waits (RWxxx)

Involuntary wait states are not requested by processes, but are invoked by the system to achieve process synchronization in certain circumstances. The Free Page Wait (FPG), Page Fault Wait (PFW) and Collided Page Wait (COLPG) states are associated with memory management and are discussed in Section 3.6.1.1.1. The current section is concerned with the miscellaneous resource wait (MWAIT) state.

The presence of processes in the MWAIT state indicates that there may be a shortage of a systemwide resource (usually page or swap file capacity), and that the shortage is blocking these processes from the CPU. If you see processes in this state, check the type of resource wait by examining the MONITOR PROCESSES data available in the collected recording files. Since a standard summary report contains only the very last PROCESSES display, and since the multifile summary report does not contain any PROCESSES data, you must check the resource wait states by playing back the data files and examining each PROCESSES display. Issue a MONITOR command like the following:

```
$ MONITOR /INPUT=SYS$MONITOR:file-spec /VIEWING_TIME=1 PROCESSES
```

This command will display all the PROCESSES data available in the input file. Look for RWxxx scheduling states, where xxx is a three-character code indicating the depleted resource for which the process is waiting. (The codes are listed in the *VAX/VMS Monitor Utility Reference Manual* under the description of the STATES class.) Mutex wait state (indicated by the state keyword MUTEX in the MONITOR PROCESSES display) is a temporary wait state, and is not of interest in this discussion.

The most common types of resource waits are those signifying depletion of the page and swap files. The RWSWP state indicates a swap file of deficient size, while RWMBP, RWMPE, and RWPGF may indicate a page file that is too small. (You can determine page and swap file sizes and the amount of available space they contain by issuing the SHOW MEMORY/FILES /FULL command.) The RWAST state indicates that the process is waiting for a resource, the availability of which will be signaled by delivery of an asynchronous system trap (AST). In most instances, either an I/O operation is outstanding (incomplete), or a process quota has been exhausted.

3.5.2 Improving CPU Responsiveness

It is always good practice to check the four methods for improving CPU responsiveness, to see whether there are ways to buy back more CPU power. This is particularly true if you have determined that a large compute queue is the cause of poor CPU performance. Before taking action, however, be sure to complete your evaluation of all the system's resources. You must resolve any pending memory or disk I/O responsiveness problems before attempting to improve CPU responsiveness.

3.5.2.1 Equitable CPU Sharing

If you have concluded that a large compute queue is affecting the responsiveness of your CPU, try to determine whether the resource is being shared on an equitable basis. Ask yourself the following questions:

- Have you assigned different base priorities to different classes of users?
- Is your system supporting one or more real-time processes?
- Are some users complaining about poor service while others have no problems?

The VAX/VMS operating system uses a round-robin scheduling technique for all non-real-time processes at the same scheduling priority. However, there are 31 priority levels, and as long as a higher-level process is ready to use the CPU, none of the lower-level processes will execute. A compute-bound process whose base priority is elevated above that of other processes can usurp the CPU. Conversely, the CPU will service processes with base priorities lower than the system default only when no other processes of default priority are ready for service.

Do not confuse inequitable sharing with the VAX/VMS priority-boosting scheme, which gives temporary priority boosts to processes encountering certain events, such as I/O completion. These boosts are temporary, and they cannot cause inequities.

You can detect inequitable sharing by looking at the CPU Time column of the MONITOR PROCESSES display in a standard summary report (not included in the multifile summary report). A process with a CPU time accumulation much higher than that of other processes may be suspect. A better means of detection is to use the MONITOR playback feature to obtain a display of the top CPU users during each collection interval. To view the display, issue a command of the form

```
$ MONITOR /INPUT=SYS$MONITOR:file-spec /VIEWING_TIME=1 PROCESSES /TOPCPU
```

You may want to select a specific time interval using the /BEGINNING and /ENDING qualifiers if you suspect a problem. Check whether the top process changes periodically.

It may sometimes be difficult to judge whether processes are receiving appropriate amounts of CPU allocation, because the allocation depends on their processing requirements. If you find that the MONITOR collection interval is too large to provide a sufficient level of detail, issue the command on the running system (live mode) during a representative period using the default three-second collection interval. If you discover an inequity, try to obtain more information about the process and the image being run by issuing the DCL command SHOW PROCESS /CONTINUOUS. Your analysis of that information my lead you to consider a reexamination of your priority assignment policies for both interactive and batch users.

3.5.2.2 Reduction of CPU Consumption by the System

This section will help you answer the question, "Can I reduce demand on the CPU by curtailing consumption by the system?"

The MONITOR statistics of interest in this section are as follows:

- MODES—All items

Depending on the amount of service required by your system, VAX/VMS functions may consume anywhere from almost no CPU cycles to a significant amount. Any reductions you can make in VAX/VMS services represent additional available CPU cycles. These can be used by processes in the Compute (COM) state, thereby lowering the average size of the compute queue and making the CPU more responsive.

The information in this section will help you identify the system components that are using the CPU. You can then decide whether it is reasonable to reduce the involvement of those components.

The principal body of information about system CPU activity is contained in the MONITOR MODES class. Its statistics represent rates of clock ticks (10-millisecond units) per second; but they can also be viewed as percentages of time spent by the CPU in each of the various modes: Interrupt Stack, Kernel, Executive, Supervisor, User, PDP–11 Compatibility, and Idle. Interrupt stack time is really kernel mode time that cannot be charged to a particular process. It is therefore sometimes convenient to consider these two together. PDP–11 compatibility mode is a special case of user mode time.

Following is a list of some of the activities that execute in each processor mode.

- Interrupt stack—CPU time spent handling interrupts from peripheral devices such as disks, tapes, printers, and terminals. The majority of system scheduling code executes on the interrupt stack, because for most of the time spent executing that code, there is no current process.

  In a VAXcluster, services performed on behalf of a remote node execute on the interrupt stack, because there is no local process to which the time can be charged. These include functions involving System Communication Services (SCS), such as remote lock requests and Mass Storage Control Protocol (MSCP) requests.

- Kernel—Most local system functions execute in kernel mode. These include local lock requests, file system (XQP) requests, memory management and most system services (including $QIO).

- Executive—The major VAX/VMS consumer of executive mode time is VAX RMS. Some optional products such as ACMS, DBMS, and Rdb also run in executive mode.

- Supervisor—The command language interpreters DCL and MCR execute in this mode.

- User—Most user-written code executes in this mode.

- Compatibility—PDP-11 utility and user code executes in this mode.

- Idle—Time consumed by the NULL process. When not executing, the NULL process is always in COM state at priority 0.

Although MONITOR provides no breakdown of modes into component parts, you can make inferences about how the time is distributed within a mode by examining some of the other MONITOR classes in your summary report and through your knowledge of the workload.

3.5.2.2.1 Interpreting MONITOR MODES Data

As a general rule, the combination of interrupt stack and kernel mode time should be less than 40 percent of the total CPU time used. Note that some functions (such as VAX RMS locking and file system requests) that ran in executive mode under VAX/VMS Version 3.0 have been moved to kernel mode in Version 4.0 in conjunction with VAXcluster development. You can therefore expect an increase in kernel mode time and a decrease in executive mode time in VAX/VMS Version 4.0 as compared to Version 3.0.

Interrupt Stack Time

The MONITOR statistics of interest in this section are as follows:

- IO—Buffered I/O Rate
- DLOCK—All items
- SCS—All items

High interrupt stack time (greater than 15%) can be caused by excessive interrupts from peripheral devices. For example, DZ11 terminal controllers need to interrupt the CPU for every character read or written. Applications that frequently perform large terminal I/O operations such as "screen painting" through this controller can cause high interrupt stack time. Line printers, such as the LP11 series, which interrupt the CPU every four characters, can also cause increased interrupt stack time. Both of these devices generate buffered I/O operations that you can observe with the Buffered I/O Rate item in the MONITOR IO class. The MONITOR PROCESSES/TOPBIO command will show which processes are generating the most buffered I/O operations.

In VAXcluster systems, interrupt stack time per node may be higher than in noncluster systems, because of the remote services performed. However, if this time appears excessive, you should investigate the remote services and look for deviations from typical values. Issue the following commands:

- MONITOR DLOCK—Observe the distributed lock manager activity. Activity labeled "incoming" and "outgoing" is executed on the interrupt stack.
- MONITOR SCS/ITEM=ALL—Observe internode traffic over the Computer Interconnect (CI).
- SHOW DEVICE /SERVED /ALL—Observe the MSCP Server activity.

Note

Even though VAXcluster systems can be expected to consume marginally more CPU resources than noncluster systems because of this remote activity, there is no measurable loss in CPU performance when a system becomes a member of a VAXcluster. VAXclusters achieve their sense of "clusterness" by making use of SCS, a very low overhead protocol. Furthermore, in a quiescent cluster with default SYSGEN parameter settings, each system needs to communicate with every other system only once every five seconds.

Kernel Mode Time

The MONITOR statistics of interest in this section are as follows:

- MODES—Kernel Mode
- IO—Page Fault Rate, Inswap Rate, Logical Name Translation Rate
- LOCK—New ENQ Rate, Converted ENQ Rate, DEQ Rate
- FCP—All items
- PAGE—Demand Zero Fault Rate, Global Valid Fault Rate, Page Read I/O Rate
- DECNET—Sum of packet rates

High kernel mode time (greater than 25%) may indicate several conditions warranting further investigation:

- A memory limitation. In this case, the MONITOR IO class should indicate a high page fault rate and/or a high inswap rate. Refer to Section 3.6 for information on the memory resource.
- Excessive local locking. Become familiar with the locking rates (New ENQ, Converted ENQ and DEQ) shown in the MONITOR LOCK class, and watch for deviations from the typical values. (In VAXcluster environments, use the DLOCK class instead; only the local portion of each of the locking rates is executed in kernel mode.)
- A high process creation rate. Process creation is a CPU-intensive operation. Process accounting can help determine if this activity is contributing to the high level of kernel mode time.
- Excessive file system activity. The file system, also known as the XQP, performs various operations on behalf of users and VAX RMS. These include file opens, closes, extends, deletes, and window turns (retrieval of mapping pointers). The CPU Tick Rate of the MONITOR FCP class can be viewed as a percentage of the CPU being consumed by the file system. It is highly dependent on application file handling, and can be kept to a minimum by encouraging efficient use of files, minimizing disk fragmentation by performing periodic backups, and so forth. The Erase Rate of the FCP class is the rate of erase operations performed to support the high-water marking security feature. If you do not require this feature at your installation, be sure to set your volumes to disable it (see Section 1.6).

- Excessive direct I/O rate. While direct I/O activity, particularly disk I/O, is important in an evaluation of the I/O resource, it is also important in an evaluation of the CPU resource, because it can be costly of CPU cycles. The direct I/O rate is included in the MONITOR IO class. The top users of direct I/O are indicated in the MONITOR PROCESSES /TOPDIO class.
- A high image activation rate. The image activation code itself does not use a significant amount of CPU time, but it can cause consumption of kernel mode time by activities like the following:
  - An excessive amount of logical name translation as file specifications are parsed.
  - Increased file system activity to locate and open the image and associated library files (which activity also generates buffered I/O operations).
  - A substantial number of page faults as the images and libraries are mapped into working sets.
  - A high demand zero fault rate (shown in the MONITOR PAGE class). This activity may be accompanied by a high global valid fault rate and/or a high page read I/O (hard fault) rate.

  Two possible causes of a high image activation rate are as follows:
  - Excessive use of DCL command procedures. You should expect to see high levels of supervisor mode activity if this is the case. Frequently invoked, stable command procedures are good candidates to be rewritten as images.
  - Migration of applications from PDP–11 to VAX processors. To fit large programs into the small address space of the PDP–11 architecture, the programs are often broken into smaller pieces and chained together. The large virtual address space of the VAX/VMS architecture makes this unnecessary. Each piece of the large program migrated directly from a PDP–11 environment becomes a separate image, with the result that references to the various pieces cause high activation rates.
- Excessive use of DECnet. Become familiar with the packet rates shown in the MONITOR DECNET class, and watch for deviations from the typical values.

Executive Mode Time

High levels of executive mode time may be an indication of excessive VAX RMS activity. File design decisions and access characteristics can have a direct impact on CPU performance. For example, consider how the design of indexed files may affect the consumption of executive mode time:

- Bucket size determines average time to search each bucket.
- Fill factor and record add rate determine rate of bucket splits.
- Index, key, and data compression saves disk space and can reduce bucket splits, but requires extra CPU time.
- Use of alternate keys provides increased retrieval flexibility, but requires additional disk space, and additional CPU time when adding new records.

Be sure to consult the *Guide to VAX/VMS File Applications* when designing a VAX RMS application. That manual contains descriptions of available alternatives along with their performance implications.

3.5.2.3 CPU Offloading

Following are some techniques you might use to reduce demand on the CPU:

- Decompress the system libraries (see Section 1.6).
- Force compute-intensive images to execute only in a batch queue, with a job limit. A good technique for enforcing such batch execution is to use the Access Control List facility as follows:

  ```
  $ SET FILE /ACL = (IDENTIFIER=INTERACTIVE+NETWORK, ACCESS=NONE) file-spec
  ```

  This command will force batch execution of the image file for which the command is issued.
- Implement off-shift timesharing or set up batch queues to spread the CPU load across the hours when the CPU would normally not be used.
- Submit larger batch jobs. The size of the queue file SYS$SYSTEM:JBCSYSQUE.DAT and the amount of work the job controller has to do to read it are minimized when multiple small print or batch submissions are combined in fewer larger entries.
- Disable code optimization. Compilers such as FORTRAN and BLISS do some code optimizing by default. However, code optimization is a CPU-and memory-intensive operation. It may be beneficial to disable

optimization in environments where frequent iterative compiles are done. Such activity is typical of an educational environment where students are learning a new language.

- Use a dedicated batch engine. It may be beneficial during prime time to set up in a VAXcluster one system dedicated to batch work, thereby isolating the compute-intensive, noninteractive work from the online users. You can accomplish this by making sure that the cluster-accessible generic batch queue points only to executor batch queues defined on the batch system. If a local area terminal server is used for terminal access to the cluster, you can limit interactive access to the batch system by making that system unknown to the server.

3.5.2.4 CPU Load Balancing in a VAXcluster

The MONITOR statistics of interest in this section are as follows:

- MODES—Time spent by processes in each mode

You can improve responsiveness on an individual CPU in a VAXcluster by shifting some of the workload to another, less utilized processor. You can do this by setting up generic batch queues, or by assigning terminal lines to such a processor. Some terminal server products perform automatic load balancing by assigning users to the least heavily utilized processor.

Note

Do not attempt to load balance among CPUs in a VAXcluster until you are sure that other resources are not blocking (and thus not inflating idle time artificially) on a processor that is responding poorly—and until you have already done all you can to improve responsiveness on each individual processor in the cluster.

Your principal tool in assessing the relative load on each CPU is the MODES class in the MONITOR multifile summary. Compare the Idle Time figures for all the processors. The processor with the most idle time may be a good candidate for offloading the one with the least idle time.

A hint on interpreting idle time is in order. On a VAXcluster member system where low-priority batch work is being executed, there may be little or no idle time. But such a system may still be a good candidate for receiving more of the VAXcluster workload. The interactive workload on that system may be very light, so that it would have the capacity to handle more default-priority work, at the expense of the low-priority work.

There are several ways to tell whether a seemingly 100% busy processor is executing mostly low-priority batch work. The first is to issue a MONITOR command like the following and observe the TOPCPU processes:

```
$ MONITOR /INPUT=SYS$MONITOR:file-spec /VIEWING_TIME=1 PROCESSES /TOPCPU
```

Second, you can examine your batch policies to see whether the system is favored for such work. A third way is to use the ACCOUNTING image report described in Section 3.2 (or a similarly generated process accounting report) to examine the kind of work being done on the system.

Following are some techniques for VAXcluster load balancing. Once you have determined the relative CPU capacities of individual member systems, you might do any of the following:

- Use a local area terminal server to distribute interactive users.
- Increase the job limit for batch queues on high-powered systems. The distributed job controller attempts to balance the number of currently executing batch jobs with the batch queue job limit, across all executor batch queues pointed to by a generic queue. You can increase the percentage of jobs that the job controller will assign to the higher-powered CPU by increasing the job limit of the executor batch queue on that system.
- Design batch workloads to execute in parallel across a VAXcluster. For example, a large system build procedure could be redesigned so that all nodes in the VAXcluster would participate in the compilation and link phases. Synchronization would be required between the two phases and could be accomplished with the DCL command SYNCHRONIZE.
- Reallocate lock directory activity. If your VAXcluster contains member systems of widely varying CPU power, for example, VAX–11/780s and VAX 8650s, you may want to allow the more powerful processors to handle a larger portion of the distributed lock manager directory activities. This may be done by increasing the SYSGEN parameter LOCKDIRWT above the default value of 1 on the more powerful machines. Note that this approach can be beneficial only in VAXclusters that support high levels of lock directory activity.

  Consider a VAXcluster containing three VAX–11/780s and one VAX 8650. Increasing LOCKDIRWT to 3 on the VAX 8650 will cause it over time to handle about half of the directory functions, while each of the VAX–11/780s will handle approximately one-sixth. In other words, the three-to-one ratio means that the VAX 8650 is three times more likely to be chosen as the directory node for a particular resource than any of the VAX–11/780s. The MONITOR DLOCK class shows the lock directory activity on each system.

3.6 Understanding the Memory Resource

The memory resource shares some similarities with the other resources, but it exhibits some notable differences. It is similar to CPU and disk in that it is a single resource pool that must must be shared, but different in the sense that it can be separated into pieces of varying size, all of which can be allocated to processes simultaneously. The process may retain its allocation of memory until memory is demanded by other processes (page faulting), at which time the sizes of the pieces are reconfigured. In some cases, certain processes must wait longer for their allocations (swapping).

The key to good performance of the memory subsystem is to maintain working sets of appropriate size for resident processes. As a rule, the total of all resident process working set quotas should be within the amount of free memory available on the system. When there is abundant free memory available, the borrowing mechanism of VAX/VMS memory management allows working sets to grow to the value specified in the user authorization file by WSEXTENT. However, you should set the WSQUOTA value such that user programs can have reasonable faulting behavior even if they can only grow to WSQUOTA. See Section 2.2.1 for guidelines on estimating appropriate WSQUOTA values.

Erratic code and data reference patterns by user programs can cause memory to be used inefficiently. The effectiveness of a virtual memory system is based upon good locality of reference. If an application has been designed with poor virtual address reference patterns, it can require an extremely large WSQUOTA value to perform satisfactorily. In addition, applications such as AI and CAD/CAM, which perform an inordinately large amount of dynamic memory allocation, often require large WSQUOTA values.

One way to obtain information about working set values on the running system is to use the procedure shown in Example 3–2. You may wish to execute it several times during some representative period of loading to gain an idea of the steady-state working set requirements for your system.

This procedure produces output like the following:

```
                        Working Set Information

                                    WS    WS    WS    WS  Pages  Page
Username    Processname    State  Extnt Quota Deflt Size in WS faults  Image
SYSTEM      ERRFMT         HIB     1024   512   100   60    60    165 ERRFMT
SYSTEM      CACHE_SERVER   HIB     1024   512   100  512    75     55 FILESERV
SYSTEM      CLUSTER_SERVER HIB     1024   512   100   60    60    218 CSP
SYSTEM      OPCOM          LEF     2048   512   100  210    59   5764 OPCOM
SYSTEM      JOB_CONTROL    HIB     1024   512   100  360   238   1459 JOBCTL
SYSTEM      CONFIGURE      HIB     1024   512   100  125   121    101 CONFIGURE
SYSTEM      SYMBIONT_0001  HIB     1024   512   100  668    57  67853 PRTSMB
DECNET      NETACP         HIB     1500   750   175 1200   812  10305 NETACP
DECNET      EVL            HIB     1024   350   175  210    33  84080 EVL
SYSTEM      REMACP         HIB     1024   350   175   60    47     74 REMACP
SYSTEM      VAXsim_Monitor HIB     1024   200   100  350   210   1583 VAXSIM
SYSTEM      DBMS_MONITOR   LEF     1000   512   150   62    62    488 DBMMON
SYSTEM      TINKERBELLE    LEF     1024   350   175  325   177   1627
SYSTEM      NULF           COM     1024   350   250  350   246   1007 FAC
HALL        CFAI           COM     2400  1024   512  662   358    567 CFAI
VTXUP       VTX_SERVER     LEF     2400  1024   512  962   696    624 VTXSRV
WEINSTEIN   Jane           LEF     2400  1024   512  662   432  13132 EDT
HURWITZ     HURWITZ        LEF     2400  1024   512  512   350   4605
CARMODY     CARMODY        LEF     2400  1024   512  812   546  16822 MAIL
CAPARILLIO  CAPARILLIO     CUR     2400  1024   512  512   282  10839
STRATFORD   Kathy          LEF     2400  1024   512  512   210   9852
FREY        _VTA270:       LEF     2400  1024   512  512   163   1021
CHRISTOPHER _VTA271:       LEF     2400  1024   512  512   252    379
STANLEY     STANLEY        LEF     2048  1024   512  512   295  10369
MINSKY      MINSKY         LEF     2400  1024   512  512   143  60316
TESTGEN     TESTGEN        LEF     4100  1024   512  234    84  75753
CLAYMORE    Cluster Buster LEF     2400  1024   512 1262   932   1919 CREATOR
DINEAUX     Sally          LEF     2400  1024   512  512   330  31803
DECNET      SERVER_0848    LEF     1024   350   175  325   183    647 NETSERVER
LUZ         Lars           LEF     2400  1024   512 1024   980  95420 TEX
DECNET      MAIL_222       LEF     1024   350   175  325   234    526 MAIL
STEVENS     STEVENS        LEF     2400  1024   512  512   221   7851
ZEN         _VTA259:       LEF     2400  1024   512 1024   319   4267 SHOW
ZEN         ZEN_2          LEF     2400  1024   512  512   171   3026
```

Note that WS Deflt refers to the default working set size, which is reestablished at each image activation. WS Size is the current size of the working set. When the number of pages actually allocated (Pages in WS) reaches this threshold, subsequent page faults will cause page replacement. The Pages in WS column includes both private and global pages. WS Extent and WS Quota represent threshold values to which WS Size may be adjusted. Finally, the Page faults column shows the total number of faults that have occurred since process creation.

Example 3–2 Sample Procedure to Obtain Working Set Information

```
$ !
$ ! WORKSET.COM - Command file to display working set information.
$ !               Requires 'WORLD' privilege to display other processes.
$ a             =       """
$ pid           =       ""
$ context       =       ""
$ IF p1.NES. "" THEN pid = p1
$ WRITE sys$output -
"                           Working Set Information"
$ WRITE sys$output ""
$ WRITE sys$output -
"                                       WS    WS    WS    WS  Pages  Page"
$ WRITE sys$output -
  "Username    Processname   State  Extnt Quota Deflt  Size in WS faults  Image"
$ WRITE sys$output ""
$ START:
$ IF p1.EQS."" THEN pid = F$PID(context)
$ IF pid.EQS."" THEN EXIT
$ pid           =       a+pid+a
$ username      =       F$GETJPI('pid,"USERNAME")
$ IF username.EQS."" THEN GOTO START
$ processname   =       F$GETJPI('pid,"PRCNAM")
$ imagename     =       F$GETJPI('pid,"IMAGNAME")
$ imagename     =       F$PARSE(imagename,,,"NAME")
$ state         =       F$GETJPI('pid,"STATE")
$ wsdefault     =       F$GETJPI('pid,"DFWSCNT")
$ wsquota       =       F$GETJPI('pid,"WSQUOTA")
$ wsextent      =       F$GETJPI('pid,"WSEXTENT")
$ wssize        =       F$GETJPI('pid,"WSSIZE")
$ globalpages   =       F$GETJPI('pid,"GPGCNT")
$ processpages  =       F$GETJPI('pid,"PPGCNT")
$ pagefaults    =       F$GETJPI('pid,"PAGEFLTS")
$ pages         =       globalpages + processpages
$ text          =       F$FAO("!AS!15AS!5AS!5(6SL)!7SL!AS",-
  username,processname,state,wsextent,wsquota,wsdefault,wssize,-
  pages,pagefaults," "+imagename)
$ WRITE sys$output text
$ IF p1.NES."" THEN EXIT
$ GOTO START
```

3.6.1 Evaluating Memory Responsiveness

The key measure of responsiveness for the memory management subsystem is the amount of time required for a process to be allocated its share of memory. Since allocation time is not measured directly, you should be concerned with the rates of the two memory management activities that extend the processing time experienced by processes in a virtual memory system—namely, *page faulting* and *swapping*. These activities not only incur overhead on the CPU and disk resources, but they also block the execution of processes during the time the system needs to allocate memory and the time the processes spend waiting for memory allocation. Thus, your goal in evaluating the memory resource is to ensure that faulting and swapping rates are kept within reasonable bounds.

3.6.1.1 Page Faulting

The MONITOR statistics of interest in this section are as follows:

- PAGE—All items

Whenever a process references a virtual page that is not in its working set, a page fault occurs. For process execution to continue, VAX/VMS memory management software is called to acquire and map a physical page into the working set.

The fault may be *hard* or *soft*. A hard fault (measured by the Page Read I/O Rate item in the MONITOR PAGE class) is one that requires a read operation from a page or image file on disk. A soft fault is one that is satisfied by mapping to a page already in memory; this may be a global page or a page in the *secondary page cache*. (The secondary page cache consists of the Free List and Modified List; the *primary page cache* is each process's working set.) The following categories of soft faults are measured and reported in the MONITOR PAGE class:

- Free List Fault Rate—the rate of page faults satisfied by reclaiming from the free list a page that was previously allocated to a process. An excessive rate of free list faults can occur when working set quotas are too small, causing excessive page replacement.
- Modified List Fault Rate—the rate of page faults satisfied by reclaiming a page from the modified page list. An excessive rate of modified list faults can occur when working set quotas are too small.
- Demand Zero Fault Rate—the rate of page faults satisfied by allocating a free page and initializing its contents to zero. This type of fault is typically seen during image activation and whenever the virtual address space is expanded.

- Global Valid Fault Rate—The rate of page faults satisfied by mapping a shared page that is already valid (one already in another process's working set). Swapping or image activation can cause an elevated global valid fault rate.
- Write in Progress Fault Rate—The rate of page faults satisfied by mapping to a page that is in the process of being written back to disk. The rate for this type of fault is typically very low.

The total Page Fault Rate is equal to the sum of the hard fault rate (Page Read I/O Rate) plus the soft fault rate, which is the sum of the five categories listed above.

System Fault Rate is the rate of faults for which the referenced virtual address is in system space (hex address 80000000 and above). It is not included in the overall Page Fault Rate, and is discussed separately in Section 3.6.2.2.

Your own judgment, based on familiarity with the data in your MONITOR summaries, is the best determinant of an acceptable Page Fault Rate for your system.

When any of the following thresholds are exceeded, you should attempt to improve memory responsiveness. (See Section 3.6.2.)

- Hard faults (Page Read I/O Rate) should be kept as low as possible, but to no more than 10% of the overall Page Fault Rate. When the hard fault rate exceeds this threshold, you can assume that the secondary page cache is not being used efficiently.
- Overall Page Fault Rate begins to become excessive when more than 5% of the CPU is devoted to soft faulting (faulting that involves no disk I/O). Since an average soft fault takes about half a millisecond to service on a VAX–11/780 processor under VAX/VMS Version 4, a good rule of thumb is to keep the total fault rate below about 100 per second on a VAX–11/780-class processor (MicroVAX II, VAX 8200, for example). On slower processors, such as VAX–11/730, MicroVAX I, and VAX–11/750, try to keep page faulting in the 30–60 per second range. Processors faster than the VAX–11/780 (VAX–11/785, VAX 8600, VAX 8650, for example) can handle proportionally higher page fault rates, up to several hundred for the VAX 8650, and still be within the 5% rule of thumb. While these rules do not represent absolute upper limits, rates that exceed the suggested limits are warning signs that the memory resource should either be improved by one of the four means listed in Section 3.6.2, or that a memory upgrade should perhaps be considered. Note, however, that more memory will not reduce the number of page faults caused by image activation.

3.6.1.1.1 The Secondary Page Cache

The MONITOR statistics of interest in this section are as follows:

- STATES—Number of processes in Free Page Wait (FPG), Collided Page Wait (COLPG), and Page Fault Wait (PFW) states

Paging problems typically occur when the secondary page cache (Free List and Modified List) is too small. This systemwide cache, which is sized by AUTOGEN, should be large enough to ensure that the overall fault rate in not excessive and that most faults are soft faults.

When evaluating paging activity on your system, you should check for processes in the Free Page Wait (FPG), Collided Page Wait (COLPG), and Page Fault Wait (PFW) states and note departures from normal figures. The presence of processes in Free Page Wait almost always indicates serious memory management problems, because it implies that the free list has been depleted.

Processes in Page Fault Wait and Collided Page Wait are waiting for hard faults (from disk) to be satisfied. Note, however, that while hard fault waiting is undesirable, it is not as serious as swapping.

An average free list size that is between the values of the FREELIM and FREEGOAL system parameters is usually an indicator of deficient memory and is often accompanied by a high page fault rate. If either condition exists, or if the hard fault rate exceeds the recommended percentage, you must consider enlarging the free and modified lists, if possible. Enlarging the secondary page cache could reduce hard faulting, provided such faulting is not the result of image activation.

To enlarge the cache, you must increase the free list either by acquiring more memory or by freeing up memory allocated for other purposes—that permanently assigned to the system, or that allocated to user processes (up to WSQUOTA). In most cases, you will want to determine whether working set values specified for users in the UAF are larger than necessary for the users to perform their work efficiently, before you attempt to decrease those values.

Note, however, that memory freed up by this type of reallocation can cause an increase in the overall fault rate unless the working set values initially specified in the UAF were overgenerous.

If you are able to increase the size of the free list, you can then allocate more memory to the modified list. Using AUTOGEN, you can increase the modified list by adjusting the appropriate MPW system parameters. (See the *VAX/VMS System Generation Utility Reference Manual* for a description of MPW parameters.)

3.6.1.2 Swapping and Swapper Trimming

The MONITOR statistics of interest in this section are as follows:

- STATES—Number of outswapped processes
- IO—Inswap Rate

Swapping is an expensive operation. It places a large data transfer load on the disk I/O subsystem, and may cause the CPU to be underutilized, because outswapped processes are blocked from using it. Try to minimize swapping activity and to keep the inswap rate near 0, and less than 1 per second.

Before a process can use the CPU, it must receive its required allotment of memory. If it is in one of the outswapped states, it is waiting for memory and cannot use the CPU.

As a general rule, swapping should be kept to a minimum. But if you have chosen to allow a low level of swapping on your system—following the common-sense principle that processes that do not need to execute for long periods should not occupy memory—you should expect to see some processes in outswapped states. Thus, while outswapped processes may sometimes indicate that harmful swapping is taking place, a more reliable measure of the condition is a high inswap rate—a rate greater than one process per second.

Before attempting to improve a system with a high inswap rate, first check for a condition known as "artificially induced swapping". This condition occurs when there are no available balance set slots.

Check the BALSETCNT system parameter. Swapping may have been artificially induced because BALSETCNT is set too low (see Section 5.2.12). You can obtain information on balance slots with the DCL command SHOW MEMORY. If BALSETCNT appears to be set correctly, and the inswap rate is high, refer to Section 3.6.2 for other methods of improving memory responsiveness.

Swapper trimming is a memory management function that provides an alternative to swapping. For most categories of processes, the working set will be trimmed down to a smaller value in preference to swapping. When memory is scarce, the system will trim processes down to their WSQUOTA values, a function called *first-level trimming*. If a critical memory shortage develops, *second-level timming* may occur—the system will trim processes down to the value of the SWPOUTPGCNT system parameter before swapping. The default value of this parameter is typically very low, which usually eliminates most swapping, but causes page faulting to increase. The presence of several processes with working sets equal to the value of SWPOUTPGCNT is a clue that second-level trimming has occurred.

It is an indicator that, at least for a time, memory was, and possibly still is, very scarce.

3.6.2 Improving Memory Responsiveness

It is always good practice to check the four methods for improving memory responsiveness to see whether there are ways to free up more memory, even if no problem seems to exist currently.

3.6.2.1 Equitable Memory Sharing

Always check memory for inequitable sharing if you believe page faulting is excessive. Since page fault behavior is so heavily dependent on the page referencing patterns of user programs, the WSQUOTA values you assign may be satisfactory for some programs but not for others. Use the ACCOUNTING image report described in Section 3.2 to identify the programs (images) that are the heaviest faulters on your system, and then compensate by encouraging users to run such images as batch jobs on queues you have set up with large WSQUOTA values.

You may be able to detect inequitable sharing by looking at the Faults column of the MONITOR PROCESSES display in a standard summary report (it is not contained in the multifile summary report). A process with a page fault accumulation much higher than that of other processes may be suspect, although it depends on how long the process has been active. A better means of detection is to use the MONITOR playback feature to view a display of the top page faulters during each collection interval:

```
$ MONITOR /INPUT=SYS$MONITOR:file-spec /VIEWING_TIME=1 PROCESSES /TOPFAULT
```

You may want to select a time interval using the /BEGINNING and /ENDING qualifiers when you suspect that a problem has occurred.

Check to see whether the top process changes periodically. If it appears that one or two processes are consistently the top faulters, you may want to obtain more information about which images they are running and consider upgrading their WSQUOTA values, using the guidelines in Section 2.2.1. Sometimes a small adjustment in a WSQUOTA value can make a drastic difference in the page faulting behavior if the original value was somewhere around the "knee" of the working-set/page-fault curve. (See Figures 2-4, 2-5, and 2-6.)

If you find that the MONITOR collection interval is too large to provide sufficient detail, try issuing the command on the running system (live mode) during a representative period using the default three-second collection interval. If you discover an inequity, try to obtain more information about

the process and the image being run by issuing the SHOW PROCESS /CONTINUOUS command.

Another way to check for inequitable sharing of memory is to use the WORKSET.COM command procedure described in Section 3.6. Examine the various working set values and ensure that the allocation of memory, even if not evenly distributed, is appropriate.

3.6.2.2 Reduction of Memory Consumption by the System

The MONITOR statistics of interest in this section are as follows:

- PAGE—System Fault Rate
- POOL—All items

The VAX/VMS system uses physical memory for storage of the code and data structures it requires to support user processes. You have control over the sizes of two of the memory areas reserved for the system: the system working set and the nonpaged pool area. Both of these areas are sized by AUTOGEN. The sizes it selects are normally adequate, but may not be optimal, because AUTOGEN cannot anticipate all operational requirements.

The system working set is an area of physical memory reserved to satisfy page faults of virtual addresses in system space. Such virtual addresses can be code or data (paged pool, for example). Since the same system working set is used for all processes on the system, there is very little locality associated with it. This means that, if allowed to fault even moderately, the working set may exhibit page thrashing, and can create a systemwide bottleneck. You must therefore try to limit system faulting to an absolute minimum—no more than 1 fault per second. The rate is easily controlled with the SYSMWCNT system parameter. Keep in mind, however, that pages allocated to the system working set by raising the value of SYSMWCNT are considered permanently allocated to the system and are therefore no longer available for process working sets.

The nonpaged pool area is a portion of physical memory permanently allocated to the system for the storage of data structures and device drivers. Its initial size is determined by AUTOGEN, but automatic expansion will occur if necessary. The system expands pool as required by permanently allocating a page of memory from the free list. Pages freed up in this fashion are not available for use by process working sets until the system is rebooted.

Refer to the In Use figures from the MONITOR POOL class when you are considering adjustments in the sizes of the three preallocated pools (SRP, IRP, LRP) and the general dynamic pool. Using the standard summary report, compare the maximum In Use figure with the size of your initial allocation (values of the SRPCOUNT, IRPCOUNT, LRPCOUNT,

and NPAGEDYN system parameters). If the In Use value exceeds the initial allocation, pool expansion has occurred. Pool may be expanded to values defined by the system parameters SRPCOUNTV, IRPCOUNTV, LRPCOUNTV, and NPAGEVIR. You must trade off the permanent allocation of memory for nonpaged pool against the small amount of CPU overhead required to do pool expansion. If physical memory on your system is limited, it may be reasonable to accept a low to moderate amount of expansion.

Note

All POOL items are averages of levels, or snapshots. Since they are not rates, their accuracy is dependent on the collection interval.

3.6.2.3 Memory Offloading

While the most common, and probably most cost-effective type of offloading is that performed by shifting the CPU and disk resources onto memory, it is possible to improve memory responsiveness by offloading it onto disk. This procedure is recommended only when sufficient disk resource is available and its use is more cost effective than purchasing additional memory.

Some of the CPU offloading techniques described in Section 3.5.2.3 apply also to memory. Additional techniques are as follows:

- Install images with the appropriate attributes. When an image is accessed concurrently by more than one process on a routine basis, it should be installed /SHARED so that all processes use the same physical copy of the image. The LIST/FULL command of the Install Utility shows the highest number of concurrent accesses to an image installed with the /SHARED qualifier. This information can help you decide whether installing an image is worth the space. Generally, an image takes about two additional physical pages when installed /OPEN/HEADER_RESIDENT/SHARED.

- Favor process swapping over working set trimming for process-intensive applications. There are cases in which an image creates several subprocesses that may not be used continuously during the run time. These idle processes take up a share of physical memory, so that it may be wise to swap them out. This typically occurs when users walk away from their terminals for long periods of time.

  The following two techniques, used concurrently, will make the system favor swapping out inactive processes over trimming the working sets of highly active processes:

  — On a per process basis—Increase the working set quotas of the active processes.

— On a systemwide basis—Increase the value of the SYSGEN parameter SWPOUTPGCNT; perhaps as high as a typical WSQUOTA. As a result, fewer pages will be trimmed, so it is more likely that swapping will occur.

After making adjustments, monitor the inswap rate closely. If it becomes excessive, lower the value of SWPOUTPGCNT.

3.6.2.4 Memory Load Balancing

The following MONITOR statistic is of interest in this section:

- PAGE—Free List Size

You can balance the memory load by using some of the techniques described in Section 3.5.2.4 to shift user demand.

To balance the load by reconfiguring memory hardware, first examine the multifile summary report and look at the Free List Size item of the PAGE class. This item gives a rough idea of the relative amounts of free memory available on each CPU. If a system seems to be deficient in memory and is experiencing memory management problems, perhaps the best solution is to reconfigure the VAXcluster by moving some memory from a memory-rich system to a memory-poor one—provided the memory type is compatible with both CPU types.

Note

The Free List Size item is an average of levels, or snapshots. Since it is not a rate, its accuracy is dependent on the collection interval.

3.7 Understanding the Disk I/O Resource

Since the major determinant of system performance is the efficient use of the CPU, and since a process typically cannot proceed with its use of the CPU until a disk operation is completed, the key performance issue for disk I/O performance is the amount of time it takes to complete an operation.

The principal measure of disk I/O responsiveness is the average amount of time required to execute an I/O request on a particular disk—that disk's *average response time*. It is important to keep average response times as low as possible to minimize CPU blockage. If your system exhibits unusually long disk response times, refer to Section 3.7.4 for suggestions on improving disk I/O performance.

To help you interpret response time as a measure of disk responsiveness, some background information about the components of a disk transfer and the notions of disk capacity and demand is provided in the following sections.

3.7.1 Components of a Disk Transfer

The following table shows a breakdown of a typical disk I/O request on a VAX–11/780 processor running the VAX/VMS operating system. You will find it helpful to understand the amount of time each system I/O component consumes in order to complete an I/O request.

Table 3–1 Components of a Typical Disk Transfer on a VAX–11/780 (Four to Eight Block Transfer Size)

Component	Percentage of Elapsed Time	Greatest Influencing Factors
I/O Preprocessing	4	Host CPU speed
Controller Delay	2	Time needed to complete controller optimizations
Seek Time	58	Optimization algorithms Disk actuator speed Relative seek range
Rotational Delay	20	Rotational speed of disk Optimization algorithms
Transfer Time	12	Controller design Data density and rotational speed
I/O Postprocessing	4	Host CPU speed

Note that the CPU time required to issue a request is only 8% of the elapsed time, and that the majority of the time (for 4-8 block transfers) is spent performing a seek and waiting for the desired blocks to rotate under the heads. Larger transfers will spend a larger percentage of time in the "transfer time" stage. It is easy to see why I/O-bound systems do not improve by adding CPU power.

3.7.2 Disk Capacity and Demand

As with any resource, the disk resource can be characterized by its capacity to do work and by the demand placed upon it by consumers.

In evaluating disk capacity in a performance context, the primary concern is not the total amount of disk space available, but the speed with which I/O operations can be completed. This speed is determined largely by the time it takes to access the desired data blocks (seek time and rotational delay), and by the data transfer capacity (bandwidth) of the disk drives and their controllers.

3.7.2.1 Seek Capacity

Overall seek capacity is determined by the number of drives (and hence, seek arms) available. Since most disk drives can be executing a seek operation simultaneously with those of other disk drives, the more drives available, the more parallelism you can obtain.

3.7.2.2 Data Transfer Capacity

A data transfer operation requires a data channel—the path from disk through controller, across buses to memory. In this context, a channel consists of all the disks on a MASSBUS, a UDA50, or a single K.sdi of a Hierarchical Storage Controller (HSC). Data transfer on one channel may occur concurrently with data transfer on other channels. For this reason, it is a good idea to attempt to locate on separate channels disks that have large data transfer operations. On a MASSBUS or UDA50, when the channel is occupied with data transfer, seek initiation for other devices on the channel must wait for the transfer to complete. On an HSC, seek operations may be initiated for other devices on a channel transferring data. Thus, for the MASSBUS and UDA50, it is generally a good idea to locate files that have large data transfer operations on different channels from seek-intensive files, if possible.

3.7.2.3 Demand

Demand placed on the disk resource is determined by the user workload and by the needs of the system itself. The demand on a seek arm is the number, size (distance), and arrival pattern of seek requests for that disk. Demand placed on a channel is the number, size, and arrival pattern of data transfer requests for all disks attached to that channel.

In a typical VAX/VMS timesharing environment, 90% of all I/O transfers are smaller than 16 blocks. Thus, for the vast majority of I/O operations, data transfer speed is not the key performance determinant; rather, it is the time required to access the data (seek and rotational latency of the disk unit). For this reason, the factor that typically limits performance of the disk subsystem is the number of I/O operations it can complete per unit of time, rather than the data throughput rate. One exception to this rule is swapping I/O, which uses very large transfers. Certain applications, of course, can also perform large data transfers; MONITOR does not provide information about transfer size, so it is important for you to gain as much information as possible about the I/O requirements of applications running on your system. Knowing whether elevated response times are the result of seek/rotational delays or data transfer delays provides a starting point for making improvements.

3.7.3 Evaluating Disk I/O Responsiveness

The MONITOR statistics of interest in this section are as follows:

- Response Time (calculated)
- DISK—I/O Operation Rate, I/O Request Queue Length

The principal measure of disk I/O responsiveness is the average response time of each disk. While not provided directly by the Monitor Utility, it can be estimated using the I/O Operation Rate and I/O Request Queue Length items from the DISK class.

Note

Since, for each disk, the total activity from all nodes in the VAXcluster is of primary interest, all references to disk statistics will be to the Row Sum column of the MONITOR multifile summary, instead of the Row Average.

Disk statistics are provided in the MONITOR DISK class for mounted disks only. I/O Operation Rate is the rate of I/O operations completed on each mounted disk. It includes system I/O (paging, swapping, XQP) and user I/O. While operation rates are influenced by the hardware components of each disk and channel, and depend upon transfer size, a general rule of thumb for operations of the size typically seen on timesharing systems can be stated: For RA-series and MASSBUS disks, an I/O rate less than 8 per

second represents a light load, 15 per second is moderate, and a disk with an operation rate of 25 or more is heavily loaded. These figures are independent of host CPU configuration.

The I/O Request Queue Length item is the average number of I/O requests outstanding at any time during the measurement period, including those being serviced and those waiting for service. So, for example, a queue length of 1.0 indicates that, on the average, there was one request in service throughout the measurement period.

Note

Although this item is an average of levels, or snapshots, its accuracy is NOT dependent on the MONITOR collection interval, since it is internally collected once per second.

As useful as these two measurements are in assessing disk performance, an even better measure is that of average response time. It can be estimated from these two items, for each disk, by using the following formula:

```
Average response time (in milliseconds) =
(Average queue length / average I/O operation rate) * 1000
```

Average disk response time is an important statistic, because it gives you a means of ranking the relative performance of your disks with respect to each other, and comparing their observed performance against a value in the range of 25 to 40 milliseconds. Although faster response times are possible, values in this range represent the best you can reasonably expect to achieve for RA-series and MASSBUS disks on a timesharing system with little or no contention. Situations that may increase response time include:

- Contention caused by multiple users accessing and transfering data on the same drive or channel
- Large transfer sizes

Since a certain amount of disk contention is expected in a timesharing environment, response times may be expected to be longer than the achievable values.

The response time measurement is especially useful because it indicates the perceived delay from the norm, independent of whether the delay was caused by seek-intensive or data-transfer-intensive operations. Disks with response time calculations significantly larger than achievable values are good candidates for improvements, as discussed below. However, it is worth checking their levels of activity before proceeding with any further analysis. The response time figure says nothing about how often the disk has been used during the measurement period. Improving disks that show a high response time, but are used very infrequently, may not noticeably improve overall system performance.

In most environments, a disk with a sustained queue length greater than 0.20 may be considered moderately busy, and worthy of further analysis. You should try to determine whether activity on disks that show excessive response times, and that are at least moderately busy, is primarily seek-intensive or data-transfer-intensive. Such disks exhibiting moderate to high operation rates are most likely seek-intensive, whereas those with low operation rates and large queue lengths (greater than 0.50) tend to be data-transfer-intensive. (An exception is a seek-intensive disk that is blocked by data transfer from another disk on the same channel; it may have a low operation rate and a large queue length, but is not itself data-transfer-intensive). If, after attempting to improve disk performance using the four means discussed in Section 3.7.4, a problem still exists, you should consider upgrading your hardware resources. An upgrade to address seek-intensive disk problems usually centers around the addition of one or more spindles (disk drives), whereas data transfer problems are usually addressed with the addition of one or more data channels.

Note

All the disk measurements discussed in this chapter are averages over a relatively long period of time, such as a "prime-time" work shift. Significant response time problems may exist in bursts, and may not be obvious when examining long-term averages. If you suspect performance problems during a particular time, obtain a MONITOR multifile summary for that period by playing back the data files you already have, using the /BEGINNING and /ENDING qualifiers to select the period of interest. If you are not sure whether significant peaks of disk activity are occurring, check the I/O Request Queue Length MAX columns of individual summaries of each node. To pinpoint the times when peaks occurred, play back the data file of interest, and watch the displays for a CUR value equal to the MAX value already observed. The period covered by that display is the peak period.

Disk I/O Statistics for MSCP-Served Disks

Special consideration must be given to MSCP-served disks in a VAXcluster. For these disks, the MONITOR figures mean different things, depending on whether you are examining the server node (the one to which the MSCP-served disk is physically attached) or one of the client nodes (the remaining VAX nodes in the cluster). For the client nodes, the I/O operation rate and queue length refer to the pseudo-operations initiated on those nodes. For the server node, they refer to the sum of the real operations initiated locally on that node and the real operations initiated on behalf of the client nodes. For this reason, it is best to compute a separate set of remote statistics and local statistics (I/O operation rate, queue length, and response time). The local statistics are those listed for the server node. The remote I/O operation rate and queue length figures must be recalculated by subtracting

the contributions of the server node from the Row Sums. You can now use these figures to calculate the average response time for remote requests.

3.7.4 Improving Disk I/O Responsiveness

It is always good practice to check the four methods for improving disk I/O responsiveness to see whether there are ways to use the available capacity more efficiently, even if no problem seems to exist currently.

3.7.4.1 Equitable Disk I/O Sharing

If you identify certain disks as good candidates for improvement, check for excessive use of the disk resource by one or more processes. The best way to do this is to use the MONITOR playback feature to obtain a display of the top direct I/O users during each collection interval. The direct I/O operations reported by MONITOR include all user disk I/O and any other direct I/O for other device types. In many cases, disk I/O represents the vast majority of direct I/O activity on VAX/VMS systems, so you can use this technique to obtain information on processes that may be supporting excessive disk I/O activity.

Issue a MONITOR command similar to the following:

```
$ MONITOR /INPUT=SYS$MONITOR:file-spec /VIEWING_TIME=1 PROCESSES /TOPDIO
```

You may want to specify the /BEGINNING and /ENDING qualifiers to select a time interval that covers the problem period.

If it appears that one or two processes are consistently the top direct I/O users, you may want to obtain more information about which images they are running and which files they are using. Since this information is not recorded by MONITOR, it must be obtained another way. One suggestion is to run MONITOR in live mode and issue DCL SHOW commands when the situation reoccurs. Another is to use the ACCOUNTING image report described in Section 3.2. Finally, you can simply survey heavy users of system resources.

To run MONITOR in live mode, choose a representative period, and use the default three-second collection interval. When you have identified a process that consistently issues a significant number of direct I/O requests, look for more information about the process and the image being run by using the SHOW PROCESS /CONTINUOUS command. In addition, you can use the SHOW DEVICE /FILES command to show all open files on particular disk volumes; it is important to know the file names of heavily used files to perform the offloading and load balancing operations described in see Sections 3.7.4.3 and 3.7.4.4.

3.7.4.2 Reduction of Disk I/O Consumption by the System

The VAX/VMS system uses the disk I/O subsystem for three activities: paging, swapping, and XQP operations. This kind of disk I/O is a good place to start when setting out to trim disk I/O load. All three types of system I/O can be reduced readily by offloading to memory. Swapping I/O is a particularly data-transfer-intensive operation, while the other types tend to be more seek-intensive.

Paging I/O Activity

The MONITOR statistics of interest in this section are as follows:

- PAGE—Page Fault Rate, Page Read Rate, Page Read I/O Rate, Page Write Rate, Page Write I/O Rate

Page Read I/O Rate, also known as the hard fault rate, is the rate of read I/O operations necessary to satisfy page faults. Since the system attempts to cluster several pages together whenever it performs a read, the number of pages actually read will be greater than the hard fault rate. The rate of pages read is given by the Page Read Rate. To compute the average transfer size (in blocks) of a page read I/O operation, divide the Page Read Rate by the Page Read I/O Rate.

Most page faults are *soft* faults. Such faults require no disk I/O operation, because they are satisfied by mapping to a global page or to a page in the secondary page cache (Free List and Modified List). For the cache to function effectively, the rate of *hard* faults—those requiring a disk I/O operation—should be less than 10% of the overall page fault rate, with the remaining 90% being soft faults. Even if the hard fault rate is less than 10%, you should try to reduce it further if it represents a significant fraction of the disk I/O load on any particular node or individual disk (see Section 3.6.1.1.1). Note that the number of hard faults resulting from image activation can be reduced only by curtailing the number of image activations, or by exercising LINKER options such as /NOSYSSHR (to reduce image activations) and reassignment of PSECT attributes (to increase the effectiveness of page fault clustering).

The Page Write I/O Rate represents the rate of disk I/O operations to write pages from the modified list to backing store (page and section files). As with page read operations, page write operations are clustered. The rate of pages written is given by the Page Write Rate. To compute the average transfer size (in blocks) of a page write I/O operation, divide the Page Write Rate by the Page Write I/O Rate. The frequency with which pages are written depends on the page modification behavior of the workload and on the size of the modified list. In general, a larger modified list must be written less often than a smaller one.

You can obtain information on each page file, including the disks on which they are located, with the DCL command SHOW MEMORY/FILES/FULL.

Swapping I/O Activity

The following MONITOR statistic is of interest in this section:

- IO—Inswap Rate

Swapping I/O should be kept as low as possible. The Inswap Rate item of the IO class lists the rate of inswap I/O operations. In typical cases, for each inswap, there may also be just as many outswap operations. Try to keep the inswap rate as low as possible—no greater than 1. This is not to say that swapping should always be eliminated. A very low level of swapping, implemented by adjusting the SWPOUTPGCNT system parameter, may be desirable to force inactive processes out of memory.

Swap I/O operations are very large data transfers; they can cause device and channel contention problems if they occur too frequently. Issue the DCL command SHOW MEMORY/FILES/FULL to list the swap files in use. If you have disk I/O problems on the channels servicing the swap files, attempt to reduce the swap rate (see Section 5.2.12).

File System (XQP) I/O Activity

The MONITOR statistics of interest in this section are as follows:

- FCP—Disk Read Rate, Disk Write Rate, Erase Rate
- FILE_SYSTEM_CACHE—All items

To determine the rate of I/O operations issued by the XQP on a nodewide basis, add the Disk Read Rate and Disk Write Rate items of the FCP class for each node and compare this number to the sum of the I/O Operation Rate figures for all disks on that same node. If this number represents a significant fraction of the disk I/O on that node, attempt to make improvements by addressing one or more of the following three sources of XQP disk I/O operations: cache misses, erase operations, and fragmentation.

First check the FILE_SYSTEM_CACHE class for the level of activity (Attempt Rate) and Hit Percentage for each of the seven caches maintained by the XQP. The categories represent types of data maintained by the XQP on all mounted disk volumes. When an attempt to retrieve an item from a cache "misses," the item must be retrieved by issuing of one or more disk I/O requests. It is therefore important to supply memory caches large enough to keep the hit percentages high and disk I/O operations low.

Cache sizes are controlled by the ACP/XQP system parameters. Data items in the FILE_SYSTEM_CACHE display correspond to ACP/XQP parameters as follows:

FILE_SYSTEM_CACHE Item	ACP/XQP Parameters
Dir FCB	ACP_SYSACC ACP_DINDXCACHE
Dir Data	ACP_DIRCACHE
File Hdr	ACP_HDRCACHE
File ID	ACP_FIDCACHE
Extent	ACP_EXTCACHE ACP_EXTLIMIT
Quota	ACP_QUOCACHE
Bitmap	ACP_MAPCACHE

The values determined by AUTOGEN should be adequate. However, if hit percentages are low (less than 75%), you should increase the appropriate cache sizes (using AUTOGEN), particularly when the attempt rates are high.

If you decide to change the ACP/XQP cache parameters, remember to reboot the system to make the changes effective. For more information on these parameters, refer to the *VAX/VMS System Generation Utility Reference Manual*.

If your system is running with the default HIGHWATER_MARKING attribute enabled on one or more disk volumes, check the Erase Rate item of the FCP class. This item represents the rate of erase I/O requests issued by the XQP to support the high-water marking feature. If you did not intend to enable this security feature, see Section 1.6 for instructions on how to disable it on a per-volume basis.

When a disk becomes seriously fragmented, it can cause additional XQP disk I/O operations, and consequent elevation of the disk read and disk write rates. You can restore contiguity for badly fragmented files by using the Backup and Convert utilities or the DCL command COPY /CONTIGUOUS. It is a good performance management practice to perform image backups of all disks periodically, using the output disk as the new copy. BACKUP consolidates allocated space on the new copy, eliminating fragmentation. You can test individual files for fragmentation by issuing the DCL command DUMP /HEADER to obtain the number of file extents. The fewer the extents, the lower the level of fragmentation will be. Pay particular attention to heavily used indexed files, especially those from which records are frequently deleted. You can use the Convert Utility to reorganize the index file structure.

3.7.4.3 Disk I/O Offloading

Following are some techniques for offloading disk I/O onto other resources, most notably memory.

- Increase the size of the secondary page cache and XQP caches.
- Install frequently used images to save memory and decrease the number of I/O operations required during image activation. (See Section 1.6.)
- Decompress library files (especially HELP files) to decrease the number of I/O operations and reduce the CPU time required for library operations. Users will experience faster response to DCL HELP commands. (See Section 1.6.)
- Use global data buffers (if your system has sufficient memory) for the following system files: VMSMAIL.DAT, SYSUAF.DAT, and RIGHTSLIST.DAT.
- Tune applications to reduce the number of I/O requests by improving their buffering strategies. However, you should make sure that you have adequate working sets and memory to support the increased buffering. This approach will decrease the number of accesses to the volume at the expense of additional memory requirements to run the application.

  Following are suggestions of particular interest to application programmers:

  — Read or write more data per I/O operation.

    - For sequential files, increase the multiblock count to move more data per I/O operation while maintaining proper process working set sizes.
    - Turn on deferred write for sequential access to indexed and relative files; an I/O operation then occurs only when a bucket is full, not on each $PUT. For example, without deferred write enabled, 10 $PUTs to a bucket that holds 10 records require 10 I/O operations. With deferred write enabled, the 10 $PUTs require only a single I/O operation.

  — Enable read ahead/write behind for sequential files. This provides for the effective use of the buffers by allowing overlap of I/O and buffer processing.

  — Given ample memory on your system, you may consider having a deeper index tree structure with smaller buckets, particularly with shared files. This approach sometimes reduces the amount of search time required for buckets and may also reduce contention for buckets in high-contention index file applications.

- For indexed files, you might try to cache the entire index structure in memory by manipulating the number and size of buckets.
- If it is not possible to cache the entire index structure, you may be able to reduce the index depth by increasing the bucket size. This will reduce the number of I/O operations required for index information at the expense of increased CPU time required to scan the larger buckets.

3.7.4.4 Disk I/O Load Balancing

The objective of disk load balancing is to minimize the amount of contention for use of the following:

- Disk heads available to perform seek operations
- Channels available to perform data transfer operations

You can accomplish that objective by moving files from one disk to another, or by reconfiguring the assignment of disks to specific channels.

Contention causes increased response time, and, ultimately, increased blocking of the CPU. In many systems, contention (and therefore response time) for some disks is relatively high, while for others, response time is near the achievable values for disks with no contention. By moving some of the activity on disks with high response times to those with low response times, you will probably achieve better overall response.

Use the guidelines in Section 3.7.3 to identify disks with excessively high response times that are at least moderately busy, and attempt to characterize them as mainly seek-intensive or data-transfer-intensive. Then use the following techniques to attempt to balance the load—by moving files from one disk to another or by moving an entire disk to a different channel.

- Separate data-transfer-intensive activity and seek-intensive activity onto separate disks (and separate channels for MASSBUS and UDA50 devices).
- Distribute seek-intensive activity evenly among the disks available for that purpose.
- Distribute data-transfer-intensive activity evenly among the disks available for that purpose (on separate channels where possible).

Note

On an HSC controller, it is important to know which disks are attached to which K.sdi data channels. This information may be obtained at the HSC console.

To move files from one disk to another, you must know, in general, what each disk is used for, and, in particular, which files are ones for which large transfers are issued. You can obtain a list of open files on a disk volume by issuing the DCL command SHOW DEVICE/FILES. However, since the system does not maintain transfer-size information, your knowledge of the applications running on your system must be your guide.

Following are some suggestions for load balancing system files:

- Use search lists to move read-only files, such as images, to different disks. This technique is not well suited for write operations to the target device because the write will take place to the first volume/directory for which you have write access.
- Define volume sets to distribute access to files requiring read and write access. This technique is particularly helpful for applications that perform many file create and delete operations, because the file system will allocate a new file on the volume with the greatest amount of free space.
- Move paging and swapping activity off the system disk by creating, on a less heavily utilized disk, secondary page and swap files that are significantly larger than the primary ones on the system disk. This technique is particularly important for a shared system disk in a VAXcluster, which tends to be very busy.
- Move frequently accessed files off the system disk. Use logical names or, where necessary, other pointers to access them. (See Section 1.6 for a list of frequently accessed system files.) This technique is particularly effective for a shared system disk in a VAXcluster.

3.8 Summary of Important MONITOR Data Items

Table 3–2 provides a quick reference to the MONITOR data items you will probably need to check most often in evaluating your resources.

Example 3–3, a typical VAXcluster "prime-time" multifile summary report, provides an extended context for the data items in the table.

Table 3–2 Summary of Important MONITOR Data Items

Item	Class	Comment[1]	Section
Compute Queue (COM + COMO)	STATES	Good measure of CPU responsiveness in most environments. Typically, the larger the compute queue, the longer response time will be.	3.5.1.1
Idle Time	MODES	Good measure of available CPU cycles, but only when processes are not unduly blocked because of insufficient memory or an overloaded disk I/O subsystem.	3.5.1.2
Inswap Rate	IO	Rate used to detect memory management problems. Should be as low as possible, no greater than 1 per second.	3.6.1.2
Interrupt Stack Time + Kernel Mode Time	MODES	Time representing service performed by the system. Should normally not exceed 40% in most environments.	3.5.2.2 3.5.2.2.1
Executive Mode Time	MODES	Time representing service performed by VAX RMS and some database products. Its value will depend on how much you use these facilities.	3.5.2.2
Page Fault Rate	PAGE	Overall page fault rate (excluding system faults). Paging may demand further attention when it exceeds the following values: • Between 30–60 per second for MicroVAX I, VAX–11/730, and VAX–11/750 processors • 100 per second for VAX–11/780-class processors • Proportionally higher for processors faster than VAX–11/780	3.6.1.1
Page Read I/O Rate	PAGE	The hard fault rate. Should be kept below 10% of overall rate for efficient use of secondary page cache.	3.6.1.1

[1]The values and ranges of values shown are *averages*. They are intended only as general guidelines and will not be appropriate in all cases.

Table 3–2 (Cont.) Summary of Important MONITOR Data Items

Item	Class	Comment[1]	Section
System Fault Rate	PAGE	Rate should be kept to minimum, no more than 1 fault per second.	3.6.2.2
Response Time (ms) (computed)	DISK	Expected value is 25–40 milliseconds for RA-series and MASSBUS disks with no contention and small transfers. Individual disks will exceed that value by an amount dependent on the level of contention and the average data transfer size.	3.7.3
I/O Operation Rate	DISK	Overall I/O operation rate. Following are normal load ranges for RA-series and MASSBUS disks in a typical timesharing environment, where the vast majority of data transfers are small: 1 to 8—lightly loaded 9 to 15—light to moderate 16 to 25—moderate to heavy More than 25—heavily loaded	3.7.3
Page Read I/O Rate + Page Write I/O Rate + Inswap Rate (times 2) + Disk Read Rate +Disk Write Rate	PAGE PAGE IO FCP FCP	System I/O operation rate. The sum of these items represents the portion of the overall rate initiated directly by the system.	3.7.4.2
Cache Hit Percentages	FILE_ SYSTEM_ CACHE	XQP cache hit percentages should be kept as high as possible, no lower than 75% for the active caches.	3.7.4.2

[1]The values and ranges of values shown are *averages*. They are intended only as general guidelines and will not be appropriate in all cases.

Example 3–3 Sample Prime Time VAXcluster Multifile Summary Report

```
+-----+          VAX/VMS Monitor Utility
| AVE |               PROCESS STATES
+-----+            MULTI-FILE SUMMARY
```

Node:	CURLEY (2)	LARRY	MOE	STOOGE (3)	Row	Row	Row	Row
From:	17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01	Sum	Average	Minimum	Maximum
To:	17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03				
Collided Page Wait	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Mutex & Misc Resource Wait	0.00	0.00	0.01	0.02	0.0	0.0	0.00	0.02
Common Event Flag Wait	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Page Fault Wait	0.03	0.09	0.35	0.13	0.6	0.1	0.03	0.35
Local Event Flag Wait	16.36	21.81	26.01	18.08	82.2	20.5	16.36	26.01
Local Evt Flg (Outswapped)	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Hibernate	16.36	17.50	20.33	14.44	68.6	17.1	14.44	20.33
Hibernate (Outswapped)	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Suspended	0.00	0.00	0.00	0.04	0.0	0.0	0.00	0.04
Suspended (Outswapped)	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Free Page Wait	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Compute	3.43	2.14	1.50	1.15	8.2	2.0	1.15	3.43
Compute (Outswapped)	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Current Process	0.96	1.00	1.00	0.95	3.9	0.9	0.95	1.00

(Continued on next page)

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
+------+          VAX/VMS Monitor Utility
| AVE |          TIME IN PROCESSOR MODES
+------+            MULTI-FILE SUMMARY
```

Node:	CURLEY (2)	LARRY	MOE	STOOGE (3)				
From:	17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01	Row Sum	Row Average	Row Minimum	Row Maximum
To:	17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03				
Interrupt Stack	5.21	10.56	5.40	2.02	23.2	5.8	2.02	10.56
Kernel Mode	11.52	16.20	12.22	4.92	44.8	11.2	4.92	16.20
Executive Mode	2.17	4.53	4.28	1.48	12.4	3.1	1.48	4.53
Supervisor Mode	1.06	0.97	4.60	0.70	7.3	1.8	0.70	4.60
User Mode	78.51	10.23	7.98	6.47	103.2	25.8	6.47	78.51
Compatibility Mode	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Idle Time	1.49	57.47	65.49	84.37	208.8	52.2	1.49	84.37

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
+-----+          VAX/VMS Monitor Utility
| AVE |       PAGE MANAGEMENT STATISTICS
+-----+          MULTI-FILE SUMMARY
```

Node:	CURLEY (2)	LARRY	MOE	STOOGE (3)				
From:	17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01	Row	Row	Row	Row
To:	17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03	Sum	Average	Minimum	Maximum
Page Fault Rate	20.93	32.17	34.56	40.32	128.0	32.0	20.93	40.32
Page Read Rate	7.36	16.47	25.02	26.16	75.0	18.7	7.36	26.16
Page Read I/O Rate	0.79	1.98	4.07	1.41	8.2	2.0	0.79	4.07
Page Write Rate	2.05	5.14	6.25	3.06	16.5	4.1	2.05	6.25
Page Write I/O Rate	0.03	0.09	0.21	0.07	0.4	0.1	0.03	0.21
Free List Fault Rate	5.03	7.89	6.55	8.80	28.2	7.0	5.03	8.80
Modified List Fault Rate	5.42	6.38	4.68	7.24	23.7	5.9	4.68	7.24
Demand Zero Fault Rate	4.84	8.00	9.96	14.99	37.8	9.4	4.84	14.99
Global Valid Fault Rate	4.76	7.77	9.08	7.70	29.3	7.3	4.76	9.08
Wrt In Progress Fault Rate	0.01	0.02	0.02	0.02	0.0	0.0	0.01	0.02
System Fault Rate	0.45	2.16	0.15	0.58	3.3	0.8	0.15	2.16
Free List Size	2915.60	4888.03	1459.72	106504.46	115767.8	28941.9	1459.72	106504.46
Modified List Size	178.60	241.53	166.81	345.26	932.2	233.0	166.81	345.26

(Continued on next page)

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
+-----+          VAX/VMS Monitor Utility
| AVE |          I/O SYSTEM STATISTICS
+-----+            MULTI-FILE SUMMARY
```

Node:	CURLEY (2)	LARRY	MOE	STOOGE (3)				
From:	17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01	Row Sum	Row Average	Row Minimum	Row Maximum
To:	17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03				
Direct I/O Rate	7.14	9.12	6.68	6.78	29.7	7.4	6.68	9.12
Buffered I/O Rate	6.90	11.74	15.98	16.74	51.3	12.8	6.90	16.74
Mailbox Write Rate	0.24	0.45	0.42	0.41	1.5	0.3	0.24	0.45
Window Turn Rate	0.30	0.44	0.63	0.56	1.9	0.4	0.30	0.63
Log Name Translation Rate	4.61	6.81	8.06	11.47	30.9	7.7	4.61	11.47
File Open Rate	0.68	0.47	0.47	1.20	2.8	0.7	0.47	1.20
Page Fault Rate	20.93	32.17	34.56	40.32	128.0	32.0	20.93	40.32
Page Read Rate	7.36	16.47	25.02	26.16	75.0	18.7	7.36	26.16
Page Read I/O Rate	0.79	1.98	4.07	1.41	8.2	2.0	0.79	4.07
Page Write Rate	2.05	5.14	6.25	3.06	16.5	4.1	2.05	6.25
Page Write I/O Rate	0.03	0.09	0.21	0.07	0.4	0.1	0.03	0.21
Inswap Rate	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Free List Size	2914.86	4887.46	1459.75	106504.46	115766.5	28941.6	1459.75	106504.46
Modified List Size	178.63	241.46	166.09	345.26	931.4	232.8	166.09	345.26

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
+-----+          VAX/VMS Monitor Utility
| AVE |          FILE PRIMITIVE STATISTICS
+-----+             MULTI-FILE SUMMARY
```

Node:	CURLEY (2)	LARRY	MOE	STOOGE (3)				
From:	17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01	Row	Row	Row	Row
To:	17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03	Sum	Average	Minimum	Maximum
FCP Call Rate	1.94	1.76	1.85	3.58	9.1	2.2	1.76	3.58
Allocation Rate	0.06	0.13	0.10	0.18	0.4	0.1	0.06	0.18
Create Rate	0.04	0.10	0.10	0.13	0.3	0.0	0.04	0.13
Disk Read Rate	0.82	1.65	1.30	1.22	5.0	1.2	0.82	1.65
Disk Write Rate	0.40	0.63	0.52	0.83	2.3	0.5	0.40	0.83
Volume Lock Wait Rate	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
CPU Tick Rate	2.65	2.42	2.43	1.10	8.6	2.1	1.10	2.65
File Sys Page Fault Rate	0.07	0.07	0.07	0.07	0.3	0.0	0.07	0.07
Window Turn Rate	0.30	0.44	0.63	0.56	1.9	0.4	0.30	0.63
File Lookup Rate	0.48	0.76	0.78	1.47	3.5	0.8	0.48	1.47
File Open Rate	0.68	0.47	0.47	1.20	2.8	0.7	0.47	1.20
Erase Rate	0.01	0.05	0.03	0.09	0.2	0.0	0.01	0.09

(Continued on next page)

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
          +------+          VAX/VMS Monitor Utility
          | AVE  |          NONPAGED POOL STATISTICS
          +------+             MULTI-FILE SUMMARY
```

Node:	CURLEY (2)	LARRY	MOE	STOOGE (3)				
From:	17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01	Row Sum	Row Average	Row Minimum	Row Maximum
To:	17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03				
SRPs Available	66.06	1288.09	1558.03	523.73	3435.9	858.9	66.06	1558.03
SRPs In Use	1234.43	2726.59	2094.98	2360.15	8416.1	2104.0	1234.43	2726.59
IRPs Available	42.76	156.51	117.66	970.75	1287.7	321.9	42.76	970.75
IRPs In Use	811.20	822.92	1045.85	904.93	3584.9	896.2	811.20	1045.85
LRPs Available	17.66	30.85	29.40	7.06	84.9	21.2	7.06	30.85
LRPs In Use	53.70	59.92	67.59	17.75	198.9	49.7	17.75	67.59
Dynamic Bytes Available	36945.60	142162.37	96619.55	39931.02	315658.5	78914.6	36945.60	142162.37
Dynamic Bytes In Use	381682.65	407762.06	442004.43	685618.50	1917067.7	479266.9	381682.65	685618.50
Holes In Pool	48.96	54.81	102.09	57.20	263.0	65.7	48.96	102.09
Largest Block	21238.40	123714.96	40064.00	25351.46	210368.8	52592.2	21238.40	123714.96
Smallest Block	15.46	16.00	16.00	15.28	62.7	15.6	15.28	16.00
Blocks Less or Eq 32 Bytes	13.56	11.92	22.85	15.51	63.8	15.9	11.92	22.85

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
+-----+          VAX/VMS Monitor Utility
| AVE |          LOCK MANAGEMENT STATISTICS
+-----+             MULTI-FILE SUMMARY
```

Node:	CURLEY (2)	LARRY	MOE	STOOGE (3)				
From:	17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01	Row Sum	Row Average	Row Minimum	Row Maximum
To:	17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03				
New ENQ Rate	6.01	18.04	11.26	13.55	48.8	12.2	6.01	18.04
Converted ENQ Rate	10.01	7.17	6.64	18.65	42.4	10.6	6.64	18.65
DEQ Rate	5.89	17.76	11.04	13.35	48.0	12.0	5.89	17.76
Blocking AST Rate	0.06	0.17	0.09	0.10	0.4	0.1	0.06	0.17
ENQs Forced To Wait Rate	0.08	0.25	0.11	0.13	0.5	0.1	0.08	0.25
ENQs Not Queued Rate	0.03	0.09	0.05	0.02	0.2	0.0	0.02	0.09
Deadlock Search Rate	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Deadlock Find Rate	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Total Locks	565.79	1604.27	1098.14	1251.62	4519.8	1129.9	565.79	1604.27
Total Resources	608.53	1015.50	922.68	1074.02	3620.7	905.1	608.53	1074.02

(Continued on next page)

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
+------+          VAX/VMS Monitor Utility
| AVE |             DECNET STATISTICS
+------+            MULTI-FILE SUMMARY
```

Node:	CURLEY (2)	LARRY	MOE	STOOGE (3)	Row Sum	Row Average	Row Minimum	Row Maximum
From:	17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01				
To:	17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03				
Arriving Local Packet Rate	1.06	1.82	1.80	1.88	6.5	1.6	1.06	1.88
Departng Local Packet Rate	1.43	1.71	1.66	1.79	6.6	1.6	1.43	1.79
Arriving Trans Packet Rate	0.00	0.33	0.00	0.00	0.3	0.0	0.00	0.33
Trans Congestion Loss Rate	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Receiver Buff Failure Rate	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
LRPs Available	17.66	30.87	29.46	7.06	85.0	21.2	7.06	30.87

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
                +------+          VAX/VMS Monitor Utility
                | AVE |       FILE SYSTEM CACHING STATISTICS
                +------+           MULTI-FILE SUMMARY

                  Node:        CURLEY (2)          LARRY              MOE              STOOGE (3)
                  From: 17-JAN-1986 12:44  17-JAN-1986 09:01  17-JAN-1986 09:00  17-JAN-1986 09:01
                  To:   17-JAN-1986 18:09  17-JAN-1986 18:02  17-JAN-1986 18:01  17-JAN-1986 18:03
```

						Row	Row	Row	Row
						Sum	Average	Minimum	Maximum
Dir FCB	(Hit %)	94.50	89.39	90.76	96.95	371.6	92.9	89.39	96.95
	(Attempt Rate)	0.49	0.80	0.83	1.53	3.6	0.9	0.49	1.53
Dir Data	(Hit %)	79.48	60.87	64.44	87.13	291.9	72.9	60.87	87.13
	(Attempt Rate)	1.36	3.12	2.37	3.46	10.3	2.5	1.36	3.46
File Hdr	(Hit %)	70.35	74.12	76.04	75.61	296.1	74.0	70.35	76.04
	(Attempt Rate)	1.85	1.69	1.88	2.82	8.2	2.0	1.69	2.82
File ID	(Hit %)	99.02	97.87	98.46	97.77	393.1	98.2	97.77	99.02
	(Attempt Rate)	0.04	0.11	0.09	0.12	0.3	0.0	0.04	0.12
Extent	(Hit %)	99.12	97.39	95.41	96.89	388.8	97.2	95.41	99.12
	(Attempt Rate)	0.15	0.50	0.39	0.59	1.6	0.4	0.15	0.59
Quota	(Hit %)	98.66	99.62	100.00	99.95	398.2	99.5	98.66	100.00
	(Attempt Rate)	0.03	0.06	0.03	0.16	0.2	0.0	0.03	0.16
Bitmap	(Hit %)	7.31	23.14	28.77	3.84	63.0	15.7	3.84	28.77
	(Attempt Rate)	0.00	0.02	0.04	0.13	0.2	0.0	0.00	0.13

(Continued on next page)

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
          +-----+          VAX/VMS Monitor Utility
          | AVE |            DISK I/O STATISTICS
          +-----+             MULTI-FILE SUMMARY
I/O Operation Rate
             Node:       CURLEY (2)          LARRY               MOE               STOOGE (3)
             From: 17-JAN-1986 12:44  17-JAN-1986 09:01  17-JAN-1986 09:00  17-JAN-1986 09:01    Row      Row      Row      Row
             To:   17-JAN-1986 18:09  17-JAN-1986 18:02  17-JAN-1986 18:01  17-JAN-1986 18:03    Sum  Average  Minimum  Maximum
$111$DUA2:     TSDPERF            1.54                0.87                0.94                1.15    4.5      1.1     0.87     1.54
$111$DUA3:     DUMPDISK           0.02                0.01                0.01                0.08    0.1      0.0     0.01     0.08
$111$DUA4:     PAGESWAPDISK       0.10                0.66                1.53                0.00    2.3      0.5     0.00     1.53
$111$DUA5:     BPMDISK            0.16                0.05                1.11                0.15    1.4      0.3     0.05     1.11
$111$DUA6:     QUALD              0.10                2.30                1.03                2.66    6.1      1.5     0.10     2.66
$111$DUA7:     SQMCLUSTERV4       2.12                3.38                4.90                2.87   13.2      3.3     2.12     4.90
$111$DUA11:    TIMEDISK           0.25                3.01                0.05                1.14    4.4      1.1     0.05     3.01
$111$DUA12:    QMISDB             0.11                0.32                0.40                0.13    0.9      0.2     0.11     0.40
$111$DUA13:    TSDPERF1           0.47                0.06                1.37                1.33    3.2      0.8     0.06     1.37
$111$DUA18:    TEAMSLIBRARY       0.00                0.08                0.00                0.00    0.0      0.0     0.00     0.08
$111$DJA8:     ORLEAN             0.02                0.00                0.05                0.00    0.0      0.0     0.00     0.05
MOE$DRA5:      USER01             0.00                0.00                0.03                0.00    0.0      0.0     0.00     0.03
MOE$DMA1:      UVMSQAR            0.00                0.00                0.00                0.00    0.0      0.0     0.00     0.00
$111$DJA1:     MPI$DATA           1.33                0.00                0.00                0.00    1.3      0.3     0.00     1.33
HSC007$DUA0:   SYSTEMDISK         0.00                0.00                0.00                0.03    0.0      0.0     0.00     0.03
```

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
          +------+          VAX/VMS Monitor Utility
          | AVE |             DISK I/O STATISTICS
          +------+            MULTI-FILE SUMMARY
I/O Request Queue Length
             Node:        CURLEY (2)          LARRY              MOE              STOOGE (3)
             From: 17-JAN-1986 12:44  17-JAN-1986 09:01  17-JAN-1986 09:00  17-JAN-1986 09:01      Row      Row      Row      Row
             To:   17-JAN-1986 18:09  17-JAN-1986 18:02  17-JAN-1986 18:01  17-JAN-1986 18:03      Sum  Average  Minimum  Maximum
$111$DUA2:     TSDPERF           0.06               0.03               0.03               0.05      0.1      0.0      0.03     0.06
$111$DUA3:     DUMPDISK          0.00               0.00               0.00               0.00      0.0      0.0      0.00     0.00
$111$DUA4:     PAGESWAPDISK      0.00               0.02               0.05               0.00      0.0      0.0      0.00     0.05
$111$DUA5:     BPMDISK           0.00               0.00               0.04               0.00      0.0      0.0      0.00     0.04
$111$DUA6:     QUALD             0.00               0.07               0.03               0.09      0.2      0.0      0.00     0.09
$111$DUA7:     SQMCLUSTERV4      0.10               0.15               0.24               0.13      0.6      0.1      0.10     0.24
$111$DUA11:    TIMEDISK          0.01               0.10               0.00               0.04      0.1      0.0      0.00     0.10
$111$DUA12:    QMISDB            0.00               0.01               0.01               0.00      0.0      0.0      0.00     0.01
$111$DUA13:    TSDPERF1          0.03               0.00               0.03               0.04      0.1      0.0      0.00     0.04
$111$DUA18:    TEAMSLIBRARY      0.00               0.00               0.00               0.00      0.0      0.0      0.00     0.00
$111$DJA8:     ORLEAN            0.00               0.00               0.00               0.00      0.0      0.0      0.00     0.00
MOE$DRA5:      USER01            0.00               0.00               0.00               0.00      0.0      0.0      0.00     0.00
MOE$DMA1:      UVMSQAR           0.00               0.00               0.00               0.00      0.0      0.0      0.00     0.00
$111$DJA1:     MPI$DATA          0.03               0.00               0.00               0.00      0.0      0.0      0.00     0.03
HSC007$DUA0:   SYSTEMDISK        0.00               0.00               0.00               0.00      0.0      0.0      0.00     0.00
```

(Continued on next page)

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
          +-----+          VAX/VMS Monitor Utility
          | AVE |   DISTRIBUTED LOCK MANAGEMENT STATISTICS
          +-----+              MULTI-FILE SUMMARY
```

Node:		CURLEY (2)	LARRY	MOE	STOOGE (3)	Row Sum	Row Average	Row Minimum	Row Maximum
From:		17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01				
To:		17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03				
New ENQ Rate	(Local)	2.78	8.03	5.55	5.71	22.0	5.5	2.78	8.03
	(Incoming)	0.33	8.23	2.17	2.02	12.7	3.1	0.33	8.23
	(Outgoing)	2.89	1.76	3.53	5.80	14.0	3.5	1.76	5.80
Converted ENQ Rate	(Local)	2.11	4.12	4.79	4.27	15.3	3.8	2.11	4.79
	(Incoming)	6.71	2.31	0.54	1.35	10.9	2.7	0.54	6.71
	(Outgoing)	1.17	0.72	1.31	13.03	16.2	4.0	0.72	13.03
DEQ Rate	(Local)	2.78	8.00	5.54	5.71	22.0	5.5	2.78	8.00
	(Incoming)	0.27	8.06	2.06	1.95	12.3	3.0	0.27	8.06
	(Outgoing)	2.82	1.69	3.43	5.68	13.6	3.4	1.69	5.68
Blocking AST Rate	(Local)	0.01	0.06	0.03	0.04	0.1	0.0	0.01	0.06
	(Incoming)	0.05	0.02	0.05	0.02	0.1	0.0	0.02	0.05
	(Outgoing)	0.00	0.09	0.01	0.03	0.1	0.0	0.00	0.09
Dir Functn Rate	(Incoming)	3.24	2.21	2.26	1.49	9.2	2.3	1.49	3.24
	(Outgoing)	1.52	2.01	1.98	3.87	9.4	2.3	1.52	3.87
Deadlock Message Rate		0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
          +-----+          VAX/VMS Monitor Utility
          | AVE |              SCS STATISTICS
          +-----+            MULTI-FILE SUMMARY
Kbytes Map Rate
            Node:       CURLEY (2)          LARRY              MOE              STOOGE (3)
            From: 17-JAN-1986 12:44  17-JAN-1986 09:01  17-JAN-1986 09:00  17-JAN-1986 09:01      Row      Row      Row      Row
            To:   17-JAN-1986 18:09  17-JAN-1986 18:02  17-JAN-1986 18:01  17-JAN-1986 18:03      Sum  Average  Minimum  Maximum
CURLEY                        0.00               0.00               0.15               0.00      0.1      0.0     0.00     0.15
VANITY                       29.50              29.00              28.15              35.89    122.5     30.6    28.15    35.89
MOE                           0.00               0.00               0.00               0.00      0.0      0.0     0.00     0.00
STOOGE                        0.00               0.00               0.08               0.00      0.0      0.0     0.00     0.08
LARRY                         0.00               0.00               0.00               0.00      0.0      0.0     0.00     0.00
DECEIT                       20.85               0.00               0.00               0.18     21.0      5.2     0.00    20.85
HSC003                        0.00               0.00               0.00               0.00      0.0      0.0     0.00     0.00
HSC007                        0.00               0.00               0.00               2.43      2.4      0.6     0.00     2.43
```

(Continued on next page)

Example 3–3 (Cont.) Sample Prime Time VAXcluster Multifile Summary Report

```
+-----+          VAX/VMS Monitor Utility
| AVE |            SYSTEM STATISTICS
+-----+            MULTI-FILE SUMMARY
```

Node:	CURLEY (2)	LARRY	MOE	STOOGE (3)				
From:	17-JAN-1986 12:44	17-JAN-1986 09:01	17-JAN-1986 09:00	17-JAN-1986 09:01	Row Sum	Row Average	Row Minimum	Row Maximum
To:	17-JAN-1986 18:09	17-JAN-1986 18:02	17-JAN-1986 18:01	17-JAN-1986 18:03				
Interrupt Stack	5.21	10.56	5.40	2.02	23.2	5.8	2.02	10.56
Kernel Mode	11.52	16.20	12.22	4.92	44.8	11.2	4.92	16.20
Executive Mode	2.17	4.53	4.28	1.48	12.4	3.1	1.48	4.53
Supervisor Mode	1.06	0.97	4.60	0.70	7.3	1.8	0.70	4.60
User Mode	78.51	10.23	7.98	6.47	103.2	25.8	6.47	78.51
Compatibility Mode	0.00	0.00	0.00	0.00	0.0	0.0	0.00	0.00
Idle Time	1.49	57.47	65.49	84.37	208.8	52.2	1.49	84.37
Process Count	37.13	42.42	49.20	34.84	163.6	40.9	34.84	49.20
Page Fault Rate	20.93	32.17	34.56	40.32	128.0	32.0	20.93	40.32
Page Read I/O Rate	0.79	1.98	4.07	1.41	8.2	2.0	0.79	4.07
Free List Size	2950.86	4908.18	1489.87	106503.10	115852.0	28963.0	1489.87	106503.10
Modified List Size	215.00	275.42	180.22	346.08	1016.7	254.1	180.22	346.08
Direct I/O Rate	7.14	9.12	6.68	6.78	29.7	7.4	6.68	9.12
Buffered I/O Rate	6.90	11.74	15.98	16.74	51.3	12.8	6.90	16.74

4 Diagnosing Resource Limitations

When you suspect that your system performance is suffering from a limited resource, you can begin to investigate which resource is most likely responsible. In a correctly behaving system that becomes fully loaded, one of the three resources, memory, I/O, or CPU becomes the limiting resource. Which resource assumes that role depends on the kind of load your system is supporting.

4.1 Diagnostic Strategy

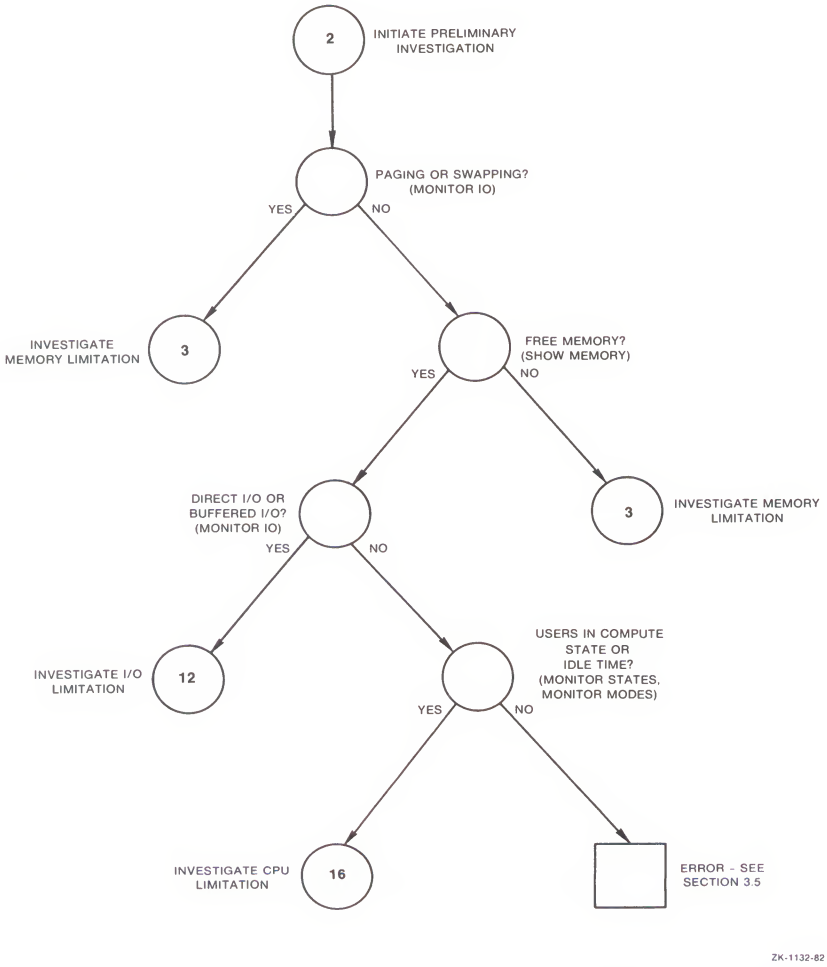
If you are uncertain where to begin, you may want to proceed by first checking the possibility of memory limitations. You can next check I/O limitations, followed by CPU limitations. It may be that your final investigations will lead you to conclude that the real source of the problem is human error, possibly misuse of the resources by one or more users.

This section describe how to investigate each of these possibilities. The investigative procedures are summarized in Figure 4–1. Note that the diagram includes command recommendations to help you obtain required information. The recommended commands appear in parentheses below the description of the information required.

The procedures use the process of elimination to determine the source of performance problems. That is, there are some fairly simple tests you can use to rule out certain classes of problems. However, you must be able to observe the undesirable behavior while you are running these tests. You can determine nothing with these methods unless your system is exhibiting the problem.

Be aware that it is possible to have overlapping limitations; that is, you could find that you have a memory limitation and an I/O limitation occurring simultaneously. With the methods outlined here, reiterating as necessary, you should be able to detect all major limitations for further resolution.

Figure 4–1 Steps in the Preliminary Investigation Process



Following are general descriptions of the resource limitations that most commonly lead to performance degradation.

- **Memory limitations**—Memory limitations are manifestations of such diverse problems as too little physical memory for the work attempted, inappropriate use of the memory management features, improper assignments of memory resources to users, and so forth.

  You can rule out memory limitations if you use the DCL commands MONITOR IO or MONITOR SYSTEM and observe a substantial amount of free memory (see the entries for Free List Size and Modified List Size) and then also find that little or no paging (see Page Fault Rate) and little or no swapping (see Inswap Rate) are occurring. However, when you observe swapping, little free memory, or significant paging, you should investigate memory limitations further. See Section 4.2.

- **I/O limitations**—This type of limitation occurs when the number or speed of devices is insufficient. You will also find an I/O limitation when application design errors either place inappropriate demand on particular devices, or do not employ sufficiently large blocking factors or numbers of buffers.

  To determine if you can rule out an I/O limitation, issue the DCL command MONITOR IO or MONITOR SYSTEM and observe the rates for direct I/O and buffered I/O. If your system is not performing any direct I/O, you do not have a disk I/O limitation. If you observe that there is no buffered I/O, you do not have a terminal I/O limitation. If, however, you see that either or both operations are occurring, you cannot rule out the possibility of an I/O limitation. In this case, you should proceed to Section 4.3.

- **CPU limitations**—The CPU may become the binding resource when the workload places extensive demand on it. Perhaps all the work becomes heavily computational, or there is some condition that gives unfair advantages to certain users.

  To determine if you may be suffering from a CPU limitation, use the DCL command MONITOR STATES. If many of your processes are in the computable state, you can definitely conclude you have a CPU limitation. (If you find that many processes are in the computable outswapped state, be sure to address the issue of a memory limitation first. See Section 4.4.)

  You might also use the DCL command MONITOR MODES to observe the amount of user mode time. If the user mode time is high, there is likely a limitation occurring around the CPU utilization. The MONITOR MODES display also reveals the amount of idle time, which is sometimes called the null time. If there is almost no idle time, it is fair to conclude that the CPU is being heavily utilized.

A third indicator of a CPU limitation that the MONITOR MODES display provides is the amount of kernel mode time. A high percentage of time in kernel mode may indicate excessive consumption of the CPU resource by the operating system. This problem is more likely the result of a memory limitation, but could indicate a CPU limitation as well. If you decide to investigate the CPU limitation further, proceed through the steps in Section 4.4.

When you have completed your preliminary investigation, you are ready to pinpoint the cause of the observed behavior and to conclude, in general terms, what remedies are available to you. The next sections included suggestions for observing and isolating the particular kinds of problems that can occur, by category of resource limitation. After isolating the problems, you can proceed to one or more of the specific corrective procedures outlined in this chapter or Chapter 5.

Once you take the appropriate remedial action, you must monitor the effectiveness of the changes, and, if you do not obtain sufficient improvement, try again. In some cases, you will need to repeat the same steps, but either increase or decrease the magnitude of the changes you made. In other cases, you will proceed further in the investigation and uncover some other underlying cause of the problem and corrective steps to take. The diagrams and text do not attempt to depict this looping. Rather, repetition is always implied, pending the outcome of the changes. Tuning is frequently an iterative process. The approach to tuning presented by this chapter and the following one assumes that multiple causes of performance problems are uncovered by repeating the steps shown until you achieve satisfactory performance.

Note that effective tuning requires that you be able to observe the undesirable performance behavior while you test.

You will find it especially helpful to keep a listing of the current values of all your system parameters nearby as you conduct the following investigations. One method for obtaining the list is the following, in which you specify a file name:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  SET/OUTPUT=filename
SYSGEN>  SHOW/ALL
SYSGEN>  SHOW/SPECIAL
SYSGEN>  EXIT
$ PRINT/DELETE filename
```

(See the *VAX/VMS System Generation Utility Reference Manual.*)

4.2 Isolating Memory Limitations

The key to successful performance management of a VAX/VMS system is to keep the memory management activity to a minimum. You will find that memory limitations cause paging and/or swapping, precisely the activities you want to minimize. It requires skillful balancing of the memory management mechanism to reduce one without incurring too much of the other.

Whenever you detect paging or swapping on a system with degraded performance, you should investigate a memory limitation. If you observe instead a lack of free memory, but no serious paging or swapping, the system may be just at the point where it will begin to experience excessive paging or swapping if demand grows any more. In this case, you have a bit of advance warning, and you may want to examine some preventive measures. Section 4.2.5 describes the situation of scarce free memory without excessive paging or swapping.

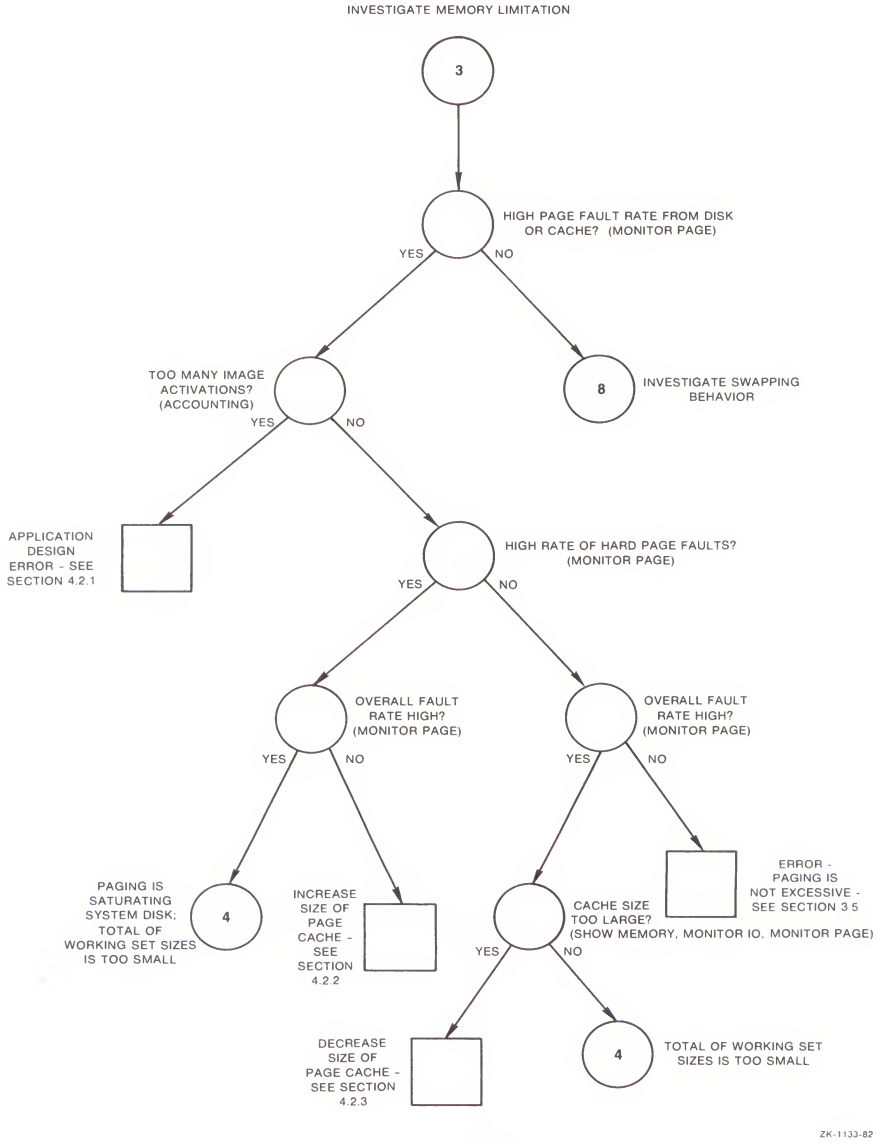
4.2.1 Analyzing the Excessive Page Faulting Symptom

There are no universally applicable scales that rank VAX/VMS page faulting rates from moderate to excessive. Although the only good page faulting rate is zero page faults per second, you need to think in terms of the maximum tolerable rate of page faulting for your system. Once you have defined this maximum value, you should view any higher page fault rate as excessive. Remember that, since paging always consumes system resources (CPU and I/O), its harmfulness depends entirely on the availability of the resources consumed.

In judging what page faulting rate is the maximum tolerable rate for your system, you must consider your configuration and the type of paging that is occurring. For example, on a system with slow disks, what might otherwise seem to be a low rate of paging to the disk could actually represent intolerable paging because of the response time through the slow disk. This is especially true if the percentage of page faults from the disk is high relative to the total number of faults. You can only judge page fault rates in the context of your own configuration. Furthermore, the numbers must be examined in the context of both the overall faulting and the apparent system performance. The system manager who knows the configuration can best evaluate the impact of page faulting.

Once you have determined that the rate of paging is excessive, you need to determine the cause. As Figure 4–2 shows, you can begin by looking at the number of image activations that have been occurring.

Figure 4–2 Investigating Excessive Paging—Phase I



4.2.1.1 Image Activations Are Excessive

If you happen to have had image-level accounting enabled as described in Section 1.1, you can use the Accounting Utility to examine the total number of images started. If, in your judgment, this number is in the low-to-normal range for typical operations at your site, you can assume that the problem lies elsewhere. However, if you suspect your system is suffering from excessive image activations, but you have not been collecting the information for ACCOUNTING to process, you can check the display produced by the MONITOR PAGE command for demand zero faults. Whenever you find that 50 percent or more of all faults are demand zero faults, you have evidence that corroborates the possibility that image activations are too frequent. You should enable image-level accounting at this time and collect enough data to confirm the conclusion.

You can determine how to reduce the number of image activations by reviewing the design of the applications according to the guidelines presented in Section 5.2. Note that the problem of paging induced by image activations is unlikely to respond to any attempt at system tuning. The appropriate action involves application design changes.

If, in spite of corrective action, your performance degradation persists, it is the result of multiple conditions. As is generally the case when you are not fully satisfied with the improvement obtained from any of the procedures, you should return to Section 4.1 to pursue further investigations.

4.2.1.2 Characterizing Hard Versus Soft Faults

Next you should characterize your page faulting. There are two kinds of paging, which you can think of as hard paging and soft paging. Paging from the disk is hard paging, and it is the less desirable of the two kinds of paging. Soft paging refers to paging from the page cache in main memory. Although soft paging is also undesirable when excessive, it is normally much less costly to overall system performance than disk paging, simply because it is faster.

All the system tuning solutions for excessive paging involve a reallocation of the memory resource, and nothing more. However, you should not reduce the size of the operating system's working set and offer that memory to the process working sets or the page cache, because it is much more costly to performance when the system incurs page faults than when other processes experience either hard or soft page faults. In fact, you should always strive to keep the system page fault rate below two faults per second. (You can observe the system fault rate with the MONITOR PAGE command.) Thus, rather than reducing the system's working set and risking the possibility of introducing system page faulting, you should consider purchasing more memory first.

Page Cache Is Too Small

In situations of excessive paging not due to image activations, you should determine what kinds of faults and faulting rates exist. Use the MONITOR PAGE command and your knowledge of your workload. If you are experiencing a high hard fault rate (represented by Page Read I/O Rate), evaluate the overall faulting rate (represented by Page Fault Rate). If the overall faulting rate is low while the hard fault rate is high, the page cache is ineffective, that is, the size of the free page list and/or the modified page list is too small. You need to increase the size of the cache. This relatively rare problem occurs when a system has been mistuned. (Perhaps AUTOGEN was bypassed.)

Before deciding to acquire more memory, you could try increasing the values of MPW_LOLIMIT, MPW_THRESH, FREEGOAL, and FREELIM. (See Section 5.2.2.) Optionally you might also try reducing the system parameter BALSETCNT (Section 5.2.13) or reducing the working set characteristics (Section 5.2.4). However, be forewarned. If these changes result immediately in the problems described below when the cache is too large and the working sets are too small (and lowering the cache parameter values a bit does not bring them into balance), you have no other tuning options. You must reduce demand or acquire more memory. (See Section 5.2.26.)

System Disk Is Saturated by Page Faulting

If you have the combination of a high hard fault rate with high faulting overall, it is quite possible the load is too high on your system, which means that the system disk activity is saturated and you must reduce the page faulting to disk.

However, first perform the checks described in Section 4.2.1.3 for small working set sizes. This action will rule out or correct the limited possibility that the combination of heavy overall faulting with heavy hard faulting is due to too large a page cache while too many processes attempt to work with small working sets. The solution will require you to reduce the cache size and increase the WSQUOTA values.

If this investigation fails to produce results, you can conclude that the system disk is saturated. You should consider

- Adding another page file on another disk
- Reducing demand
- Adding more memory

Since adding more memory (Section 5.2.26) is less costly than acquiring a disk, it is usually preferable, unless you have another disk drive available that is underutilized. See Section 5.2.25.

Page Cache Is Too Large

On the other hand, if you find that your faults are mostly of the soft variety, check to see if the overall faulting rate is high. If so, you may have the relatively rare problem of an unnecessarily large page cache. As a guideline, you should expect the size of your page cache to be one order of magnitude less than the total memory consumed by the balance set under load conditions.

The only way to create a page cache that is too large is by seriously mistuning a system. (Perhaps AUTOGEN was bypassed.) Section 5.2.3 describes how to reduce the size of the page cache through the MPW_LOLIMIT, MPW_THRESH, FREEGOAL, and FREELIM system parameters.

4.2.1.3 Total Working Set Size Is Too Small—Overview of the Problem

If your page cache size is appropriate, you need to investigate the likelihood that excessive paging is induced when a number of processes attempt to run with working set sizes that are too small for them. If the total memory for the balance set is too small, one of the following three possibilities (or a combination thereof) is at work:

1. The working set size may be inappropriate because
   - The working sets have been set too small with the WSDEFAULT and WSQUOTA characteristics in the UAF
   - The effective working set quota has been lowered by DCL commands or system services that were invoked as the process ran
   - The processes are not succeeding in borrowing working set space (in the loan region)
2. Perhaps the automatic working set adjustment feature (AWSA) has been turned off or is for some reason not as effective as it could be.
3. Swapper trimming may be reducing the working set sizes too vigorously.

Figures 4–3, 4–4, and 4–5 summarize the procedures that follow for isolating the cause of working set sizes that are too small.

Figure 4–3 Investigating Excessive Paging—Phase II

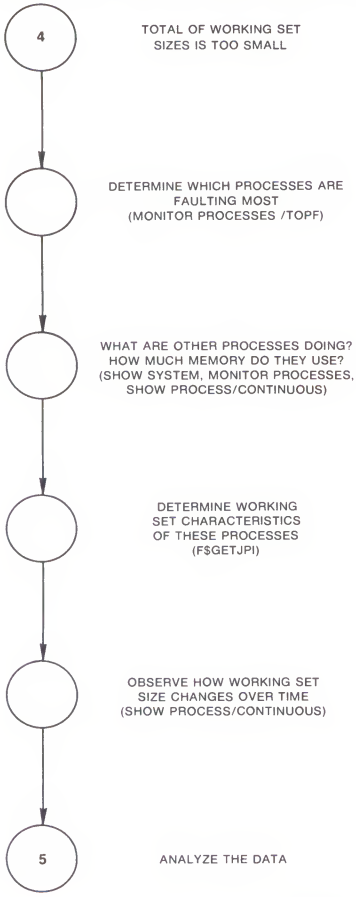
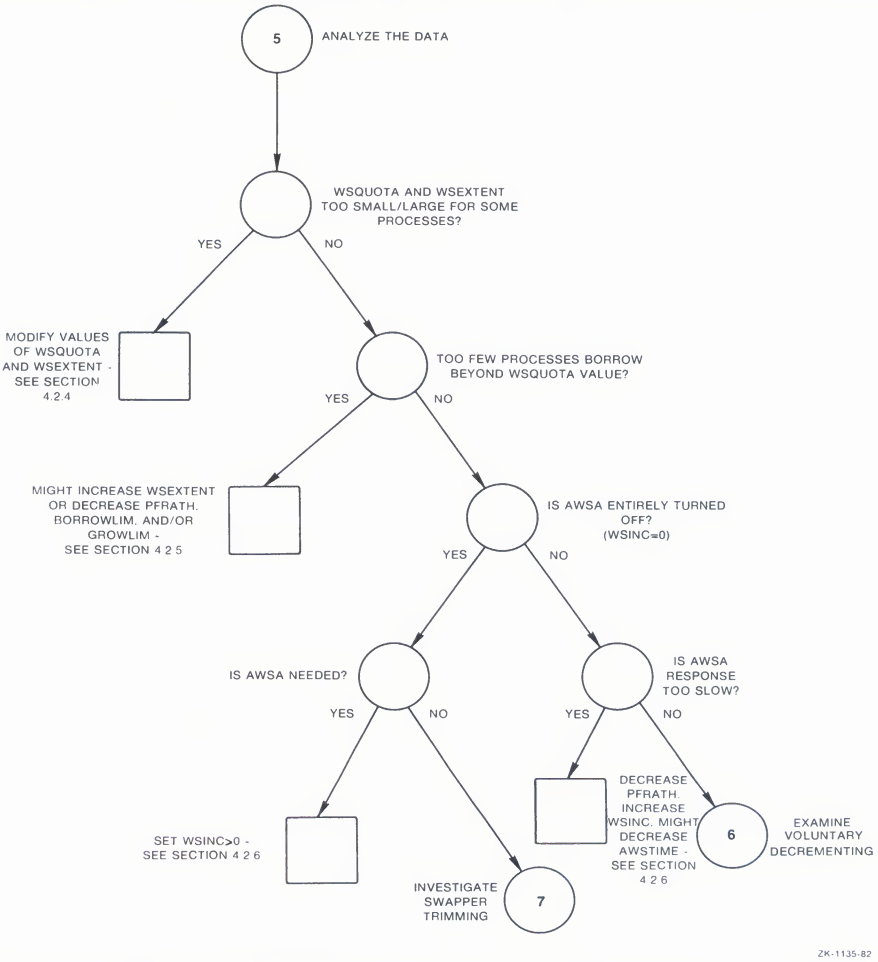


Figure 4–4 Investigating Excessive Paging—Phase III



4.2.1.4 WSDEFAULT, WSQUOTA, and WSEXTENT Values Are Inappropriate

Begin to narrow down the possible causes of too small a total working set size by looking first at your system's allocation of working set sizes. To gain some insight into the workload and which processes have too little memory,

1. Issue the MONITOR PROCESSES/TOPFAULT command to learn which processes are faulting because their working set sizes are too small
2. Use the SHOW PROCESS/CONTINUOUS command to learn what the top faulting processes are doing and how much memory they are using
3. Look at the memory consumed by the other larger processes with the SHOW SYSTEM and MONITOR PROCESSES commands.

Perhaps you can conclude that one large process (or perhaps several) does not need as much memory as it is using. If you reduced its WSQUOTA and/or WSEXTENT values, the other processes could use the memory the large process currently takes. (See Section 5.2.4.)

However, to form any firm conclusions at this point, you need to learn more about the process's behavior as its working set size grows and shrinks. Use the MONITOR PROCESSES command and the lexical function F$GETJPI for this purpose.

To look at the current values as the process executes, follow these steps:

1. Note the process identification number (PID) on the MONITOR PROCESSES display
2. Ensure that you have the WORLD privilege
3. For each heavily faulting process you want to investigate, request these items:

   Working set quota size
   Process page count
   Global page count
   Working set extent

   To request the items, enter in quotation marks (") the process identification number (pid), as shown in the following commands:

```
WSQUOTA = F$GETJPI("pid","WSQUOTA")
SHOW SYMBOL WSQUOTA
WSSIZE = F$GETJPI("pid","WSSIZE")
SHOW SYMBOL WSSIZE
PPGCNT = F$GETJPI("pid","PPGCNT")
SHOW SYMBOL PPGCNT
GPGCNT = F$GETJPI("pid","GPGCNT")
SHOW SYMBOL GPGCNT
WSEXTENT = F$GETJPI("pid","WSEXTENT")
SHOW SYMBOL WSEXTENT
```

(Suggestion: write a program or command procedure that requests the process identification number and then formats and displays the resulting data.)

The lexical function item PPGCNT represents the process page count, while GPGCNT represents the global page count. You need these values to determine how full the working set list is. The sum of PPGCNT plus GPGCNT is the actual amount of memory in use, and should always be less than or equal to the value WSSIZE. By sampling the actual amount of memory in use while processes execute, you can begin to evaluate just how appropriate the values of WSQUOTA and WSEXTENT are for each.

If the values of WSQUOTA and WSEXTENT areeither unnecessarily restricted or too large in a few obvious cases, they need to be adjusted; proceed next to Section 5.2.4.

4.2.1.5 Borrowing Is Ineffective

If you observe that few of the processes are able to take advantage of loans, then borrowing is ineffective. Section 5.2.5 discusses how to make the necessary adjustments so that borrowing is more effective.

4.2.1.6 AWSA May Be Disabled

Next, you need to investigate the status of automatic working set adjustment (AWSA). Check the value of the system parameter WSINC. If you find WSINC is greater than zero, you know that automatic working set adjustment is essentially turned on. (More precisely, the part of automatic working set adjustment that permits working set sizes to grow is turned on). However, at the same time, you should also check whether or not WSDEC and/or PFRATL are zero. While setting WSINC=0 turns the full automatic working set adjustment mechanism off, setting PFRATL=0 when WSINC is greater than zero will disable just that part of automatic working set adjustment that provides the voluntary decrements in the working set sizes. (For example, in Figure 4–7, if PFRATL and WSDEC equaled zero, the actual working set limit line would have leveled off at Q4 and would not have changed until Q20.)

If automatic working set adjustment is disabled, processes are unable to increase their working set sizes. You will observe that although processes have WSQUOTA values greater than their WSDEFAULT values, those processes that are currently active (doing some computing) do not show a working set size count above their WSDEFAULT value. At the same time your system is experiencing heavy page faulting. You should enable automatic working set adjustment, by setting WSINC greater than zero, so that working set growth is possible. See Section 5.2.10.

4.2.1.7 AWSA Is Ineffective—Overview

Now, if automatic working set adjustment is turned on, there are four ways that it may be performing less than optimally, and you must evaluate them.

1. AWSA may not be responding quickly enough to increased demand. That is, when page faulting increases significantly, working set sizes are not increased quickly enough to sufficiently large values.
2. AWSA with voluntary decrementing enabled may be causing the working set sizes to oscillate.
3. AWSA with voluntary decrementing enabled may be shrinking the working sets too quickly, thereby inducing unnecessary paging.
4. AWSA may not be decrementing the working set sizes where possible because voluntary decrementing is disabled.

AWSA Is Not Responsive to Increased Demand

If you use the SHOW PROCESS/CONTINUOUS command for those processes that MONITOR PROCESSES/TOPFAULT shows are the heaviest page faulters, and you find that the automatic working set adjustment is not increasing their working set sizes quickly enough in response to their faulting. If the default values of WSINC, PFRATH or AWSTIME have been changed, you should restore them to their original values, and consider adjusting the WSDEF and WSQUO values of the offending process.

AWSA with Voluntary Decrementing Enabled Causes Oscillations

It is possible for the voluntary decrementing feature of automatic working set adjustment to cause processes to go into a form of oscillation where the working set sizes never stabilize, but rather keep growing and shrinking while accompanied by page faulting. When you observe this situation, through the SHOW PROCESS/CONTINUOUS display, you should disable voluntary decrementing by setting PFRATL=0. See Section 5.2.7.

AWSA Shrinks Working Sets Too Quickly

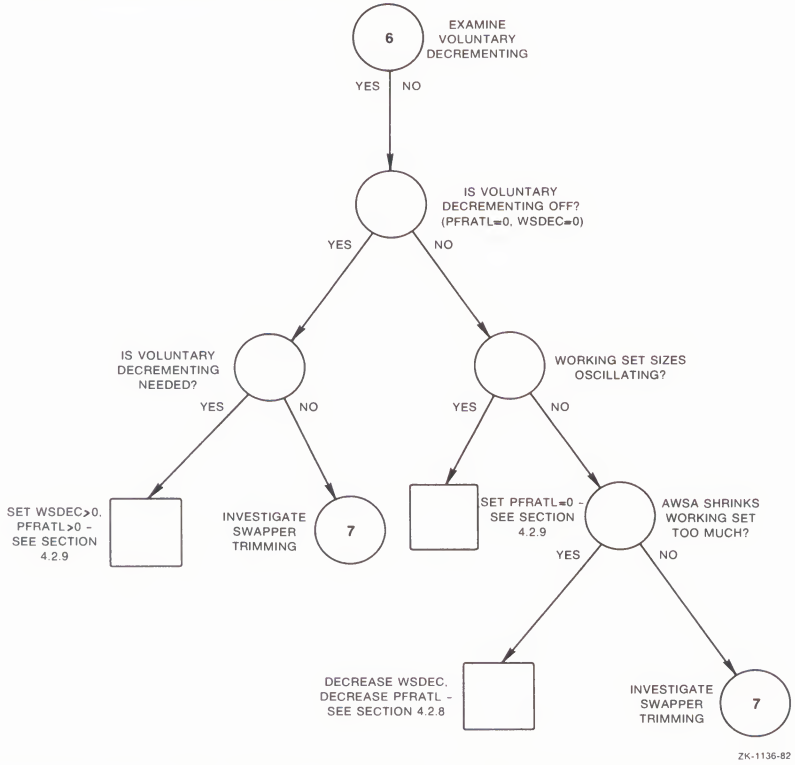
From the SHOW PROCESS/CONTINUOUS display you can also determine if the voluntary decrementing feature of automatic working set adjustment is shrinking the working sets too quickly. In that event, you should consider decreasing WSDEC and decreasing PFRATL. See Section 5.2.8.

AWSA Needs Voluntary Decrementing Enabled

You might observe the case of one or more processes that rapidly achieve a very large working set count and then maintain that size over some period of time. However, you know or suspect that those processes should not require that much memory continuously. Although those processes are not page faulting at all, other processes are. You should check whether voluntary decrementing is turned off (PFRATL=0 and optionally WSDEC=0). See Figure 4–5. It may be that for your workload, voluntary decrementing would bring about improvement, since it is time based, not load based. You could enable voluntary decrementing according to the suggestions in Section 5.2.9 to see if any improvement is forthcoming.

If you decide to take this step, keep in mind that it is the exception rather than the rule. You could make conditions worse rather than better. Be certain to monitor your system very carefully to ensure that you do not induce working set size oscillations in your overall workload, as described above. If no improvement is obtained, you should turn off voluntary decrementing. Probably your premise that the working set size could be reduced was incorrect. Also, if oscillations do result that do not seem to stabilize with a little time, you should turn voluntary decrementing off again. You must explore instead, ways to schedule those processes so that they least disrupt the workload.

Figure 4–5 Investigating Excessive Paging—Phase IV



4.2.1.8 Swapper Trimming Is Too Vigorous

Perhaps there are valid reasons why at your site WSINC has been set to zero to turn off automatic working set adjustment. For example, the applications may be well understood, and the memory requirements for each image may be so predictable that the values for WSDEFAULT and WSQUOTA can be accurately set. Furthermore, it is possible that if automatic working set adjustment is enabled at your site, you are satisfied that your system is using appropriate values for WSQUOTA, WSEXTENT, PFRATH, BORROWLIM, and GROWLIM. In these situations, perhaps swapper trimming is to blame for the excessive paging. In particular, perhaps trimming on the second level may be too severe.

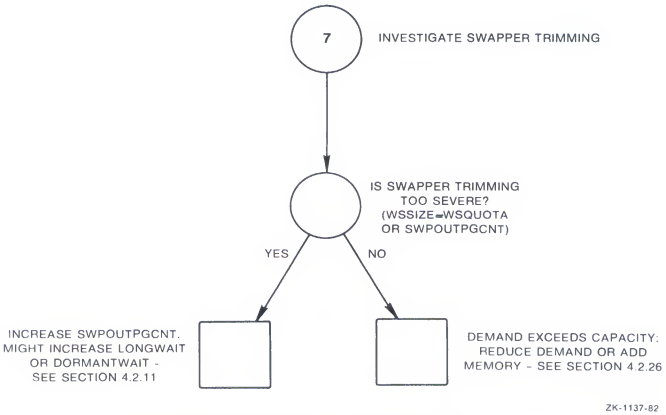
Figure 4–6 illustrates the investigation for paging problems induced by swapper trimming. Again, you must determine the top faulting processes and evaluate what is happening and how much memory is consumed by these processes. Use the MONITOR PROCESSES/TOPFAULT and MONITOR PROCESSES commands. By selecting the top faulting processes and scrutinizing their behavior with the SHOW PROCESS/CONTINUOUS command, you can determine if there are many active processes that seem to display working set sizes at

- Their WSQUOTA value
- The system-wide value set by the system parameter SWPOUTPGCNT

Either finding indicates that swapper trimming is too severe.

If such is the case, consider increasing the system parameter SWPOUTPGCNT, at the same time evaluating the need to increase the system parameter LONGWAIT. The swapper uses LONGWAIT to detect those processes that are truly idle. If LONGWAIT specifies too brief a time, the swapper may swap temporarily idle processes that would otherwise have become computable again soon. See Section 5.2.11. For computable processes, the same condition may occur if DORMANTWAIT is set too low.

Figure 4–6 Investigating Excessive Paging—Phase V



4.2.2 Analyzing the Swapping Symptom

Experience with VAX/VMS systems has shown that swapping of active processes is less desirable than modest paging, This is true because swapping involves disk accesses (true of only hard page faults). Swapping requires each process and its context to be written out to disk, an event that is normally slower than the average paging operation, since it involves more blocks. There is additional system overhead for swapping caused by stopping and starting processes. Finally, in using the disk resource heavily, the swapper may cause additional entries in the queue on its disk, thus delaying other processes that need access to that disk.

Not only is swapping costly in terms of performance, but its relative cost is higher for slower processors. In fact, the single-disk, slower-speed system pays the highest price of all for swapping, since all other access to the disk is delayed while the disk is used for swapping. For example, swapping is much more costly to system performance on the slower VAX–11/730 processors than it is on the VAX–11/780 processors.

If your processor speed is an issue, you may decide to reduce swapping and make yours a system that primarily pages. First, however, be certain that swapping is truly harming performance for your workload.

4.2.3 Detecting Harmful Swapping

Harmful swapping manifests itself in heavy consumption of the CPU resource and the disk, to the detriment of other processes. You should use the following tests to check for any symptoms that indicate swapping is harmful:

- Issue the DCL command MONITOR IO and examine the inswap rate. If the rate is zero, you have no swapping, and you need not pursue this series of tests any further.
- Check the MONITOR PROCESSES/TOPCPU display to see if the NULL process receives a significant amount of service from the CPU. If you find this condition in conjunction with swapping, the swapping is definitely harmful and needs to be remedied.
- Issue the DCL command MONITOR STATES. If you observe few processes in the COMO state, swapping is not affecting CPU operations.

If your swapping passes these three tests, you can conclude that swapping is not so harmful on your system that you should eliminate it.

However, indications of harmful swapper activity, such as heavy disk or CPU consumption, warrant some attention. (Figures 4–7, 4–8, and 4–9 summarize the investigation for swapping.) You may want to consider converting your system to one that only pages and rarely if ever swaps, particularly if your system is a small configuration. You accomplish this by

- Lowering the system parameter SWPOUTPGCNT
- Setting the system parameter BALSETCNT equal to a value that is two less than the value of the system parameter MAXPROCESSCNT
- Adding more memory

Optionally, you may decide to reduce the process working set quotas (in the UAF). See Section 5.2.12.

Even if you tune your system so that it rarely swaps, you still need a swap file on your system. However, the space requirement for the swapping file is reduced. If disk space is at a premium, you can adjust your swapping file space requirement to 75 percent of its previous value with the AUTOGEN command procedure. (See the *VAX/VMS System Manager's Reference Manual.*)

If you find that your system is showing symptoms of harmful swapping and that performance has degraded, there are two possible causes:

1 **No free balance slots**—If there are no free balance slots, use the DCL command SHOW MEMORY to check the number of free balance slots. If the number available is small, and you know there is still adequate free memory (which you can also check with SHOW MEMORY), then you should be able to alleviate the swapping by increasing the system parameter BALSETCNT. See Section 5.2.13.

2 **Insufficient free memory for all the working sets**—If there are free balance slots, but the total of the working set sizes exceeds available memory, you can safely conclude that there is not enough free memory to support all the working sets at once. This condition can result from one or more of the following factors:

   - Improper partitioning of memory due to a page cache that is too large
   - Situations where some users use unreasonably large amounts of memory
   - Demand that is simply too high for capacity

   Use the SHOW MEMORY display to determine the total usable memory (the total physical memory less the memory used by VAX/VMS). Next, determine how much memory is allocated to the page cache, by adding

the values for the two system parameters FREEGOAL and MPW_THRESH. If the page cache size is more than 15 percent of the total usable memory, the page cache may be too large.

Only when a system has been seriously mistuned should you find that the page cache is too large. (Perhaps AUTOGEN was bypassed.) Section 5.2.14 describes how to reduce the size of the page cache through the MPW_LOLIMIT, MPW_THRESH, FREEGOAL, and FREELIM system parameters.

If you determine that the page cache is not too large, or having reduced its size, you find that there is still insufficient free memory for all the working sets, you need to investigate other potential causes for the problem. These are described in the next sections.

4.2.4 Investigating Why Processes Consume Unreasonable Amounts of Memory

Swapping can be induced whenever one or a small number of processes devour memory at the expense of other processes. You can find out if a few users are using large amounts of memory by examining the display produced by the MONITOR PROCESSES command.

Figure 4–7 Investigating Swapping—Phase I

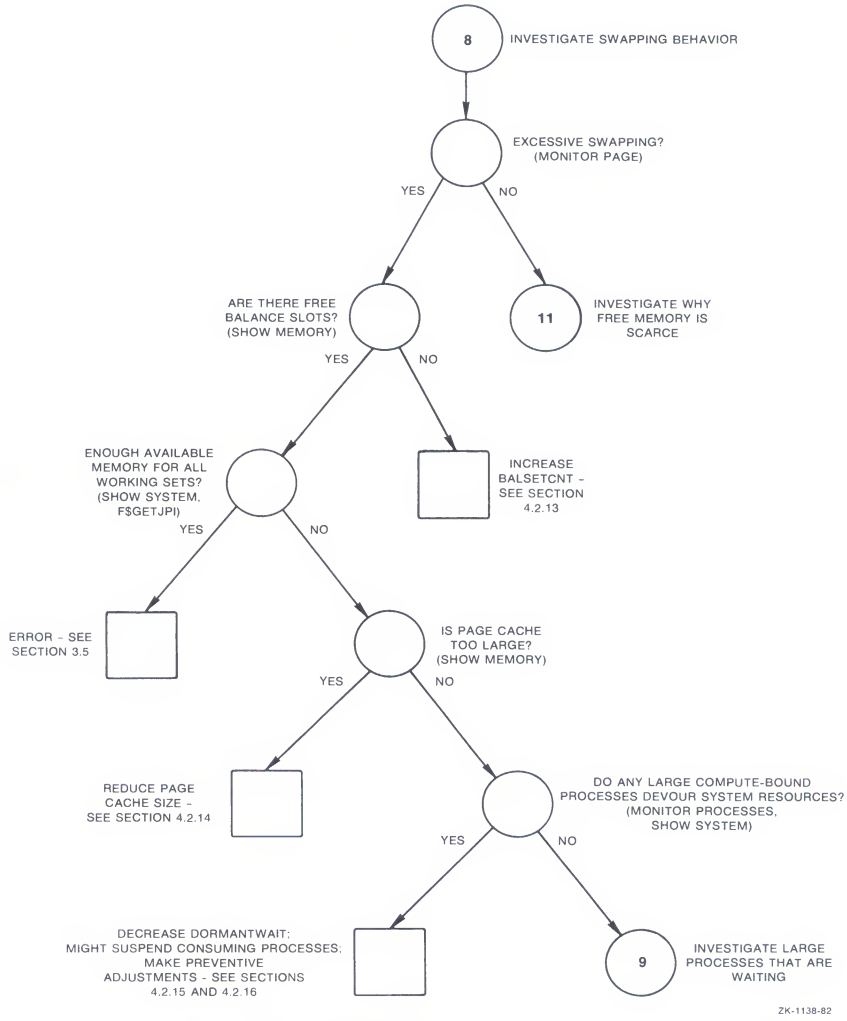


Figure 4–8 Investigating Swapping—Phase II

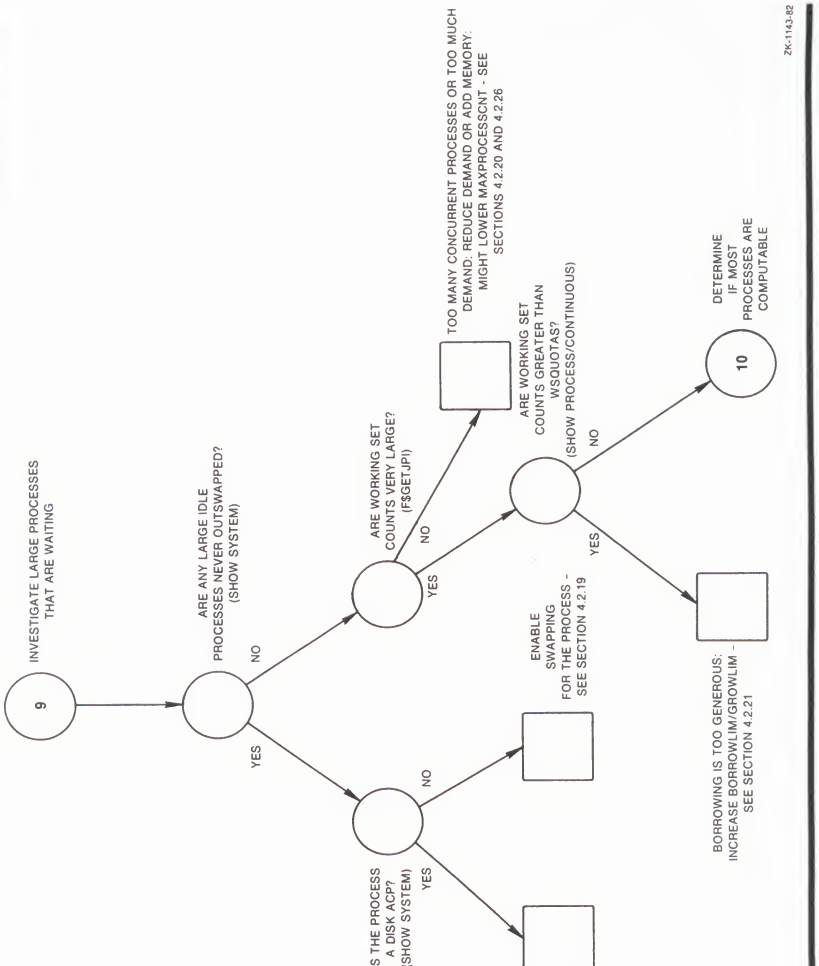
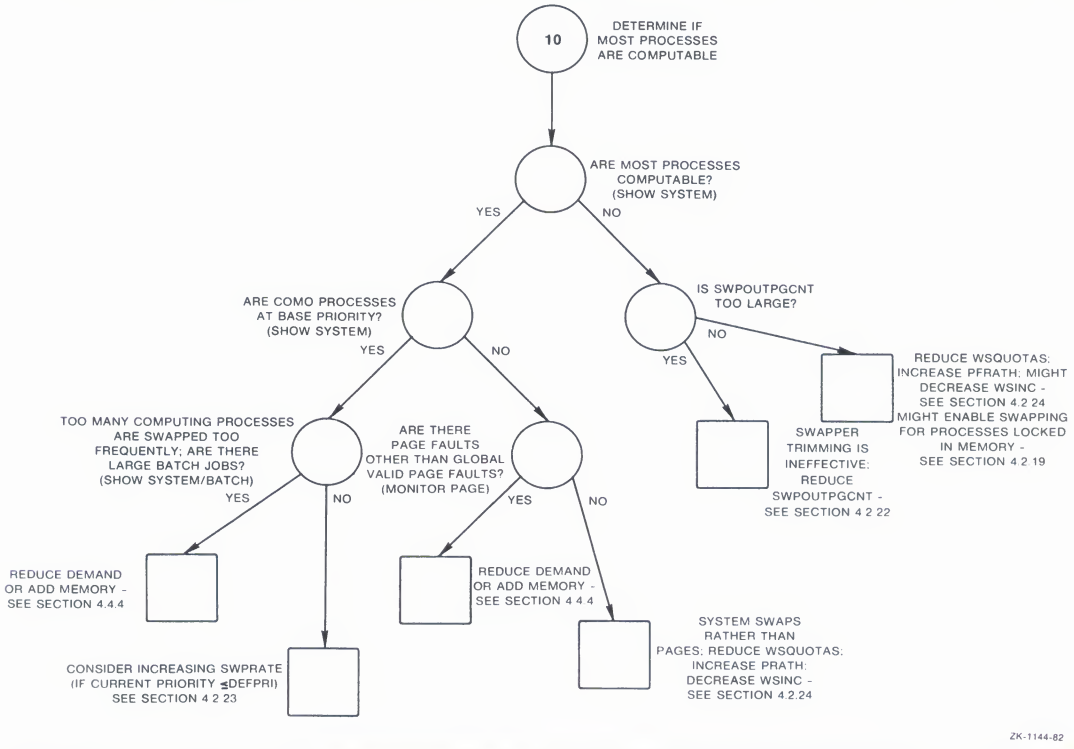


Figure 4–9 Investigating Swapping—Phase III



4.2.4.1 Large, Compute-Bound Process Gains Inordinate Control of Memory

At this point you should be particularly alert for the situation where one or more very large, compute-bound processes at low priority consume memory at the expense of a number of smaller processes. Typically the smaller processes may be trying to perform some terminal I/O, such as editing. When memory becomes tight, the large process that is compute-bound is less likely to be selected for outswapping than any process that is in the local event flag wait state. Consequently, in this situation, VAX/VMS will select processes running the editor for outswapping as soon as they start to wait for I/O. As a result, the editing processes will experience symptomatically poor response times due to frequent outswapping. The SHOW SYSTEM

command provides a valuable tool for checking the priority and state of the large process.

Note the process identification number from the MONITOR PROCESSES display and ensure that you have the WORLD privilege. Then for each large process you want to investigate, use the lexical function F$GETJPI, as described in Section 4.2.1.4, to request the working set quota, size, process page count, global page count, and working set extent.

If you that find any of the processes are above their working set quota, you may want to decrease DORMANTWAIT and monitor performance for a time. If decreasing DORMANTWAIT proves ineffective, you can issue the DCL command SET PROCESS/SUSPEND (as described in Section 5.2.16) to suspend the large, compute-bound process that is over WSQUOTA. This action offers a rapid means of restoring other process activities. (Once the process is suspended, the swapper can trim the process to its SWPOUTPGCNT value.) As soon as SHOW PROCESS/CONTINUOUS reveals that the process has been trimmed, you can safely resume it. If the AWSA is set correctly, the problem should not recur, since the process will be unable to grow beyond its quota while memory is scarce.

However, you must determine the underlying cause of the problem (for example, the working set quota may be too large for the process) and take corrective action. You could, for example, increase WSEXTENT. Borrowing will then be reclaimed by the swapper. If the large, compute-bound process is not above its working set quota, suspending the process might provide temporary relief, but as soon as you allow the process to resume, it can start to devour memory again. Thus, the most satisfactory corrective action is the permanent solution, as discussed in Section 5.2.4.2.

4.2.4.2 Large Waiting Process Is Never Outswapped

While using the SHOW SYSTEM command to look for large processes that are compute-bound, you may instead observe that one or more large processes are hibernating or are in some other wait state. Possibly swapping has been disabled for these processes. You could use the SHOW PROCESS /CONTINUOUS command for each process to determine if any inactive process escapes outswapping. As a next step, you could invoke the System Dump Analyzer (SDA) with the DCL command ANALYZE/SYSTEM, to see if the process status line produced by the SDA command SHOW PROCESS reveals the process status of PSWAPM.

If you find a process that is not allowed to swap, yet apparently consumes a large amount of memory when it is inactive, you may conclude that swapping should be enabled for it. Enabling swapping would give other processes a more equitable chance of using memory when memory is scarce and the large process is inactive. You should discuss your conclusions with the owner of the process to determine if there are valid reasons why the

process must not be swapped. (For example, most real-time processes should not be swapped.) If the owner of the process agrees to enable the process for swapping, use the DCL command SET PROCESS/SWAPPING (which requires the PSWAPM privilege). See Section 5.2.19.

If the offending process is a disk ACP (ODS–1 only), you need to set the system parameter ACP_SWAPFLGS appropriately and reboot the system. See Section 5.2.18.

4.2.4.3 Too Many Processes Compete for Available Memory

If the data you collected with the F$GETJPI lexical function reveals that the working set counts (the actual memory consumed by the processes) are not particularly large, you may simply have too many processes attempting to run concurrently for the memory available. At this point, you should ensure that ACP_XQP_RES is set to 1 (default) so that sharing of XQP pages is enabled. If it is, and the problem persists, you may find that performance improves if you reduce the system parameter MAXPROCESSCNT, which specifies the number of processes that can run concurrently. See Section 5.2.20.

However, if MAXPROCESSCNT already represents the number of users who must be guaranteed access to your system at once, reducing MAXPROCESSCNT is not a viable alternative. Instead, you must explore other ways to reduce demand (redesign your application, for example) or add memory. See Section 5.2.26.

4.2.4.4 Borrowing Is Too Generous

For the processes that seem to use the most memory, use the SHOW PROCESS/CONTINUOUS command to check if the processes are operating in the WSEXTENT region; that is, their working set sizes range between the values of WSQUOTA and WSEXTENT. if not, it might be beneficial to increase the values of BORROWLIM and/or GROWLIM. Increasing both BORROWLIM and GROWLIM discourages loans when memory is scarce. By judiciously increasing these values, you will curtail the rate of loans to processes with the largest working sets, particularly during the times when the workload peaks. See Section 5.2.21.

4.2.4.5 Swapper Trimming Is Ineffective

If memory is insufficient to support all the working set sizes of active processes, ineffective swapper trimming may be the cause.

In this case, the value of SWPOUTPGCNT may be too large. You should compare the value of SWPOUTPGCNT to the actual working set counts you observe. If you decide to reduce SWPOUTPGCNT, be aware that you will increase the amount of memory reclaimed every time second-level trimming is initiated. Still, this is the parameter that most effectively converts a system from a swapping system to a paging one and vice versa. As you lower the value of SWPOUTPGCNT, you run the risk of introducing excessive paging. If this situation occurs, and you cannot achieve a satisfactory balance between swapping and paging, you must reduce demand or add memory. See Section 5.2.26.

4.2.4.6 Many Working Sets Are Too Large

If you conclude that SWPOUTPGCNT is not too large, you have already determined that the working sets are fairly large but not above quota and that few processes are computable. You will probably discover that one or more of the following conditions exist:

- The working set quotas are too large in some cases.
- The parameter WSINC is too large or PFRATH is too low.
- Too many working sets are locked in memory and cannot be outswapped.

The first two conditions can be determined from information you have collected. However, if you suspect that too many users have used the DCL command SET PROCESS/NOSWAPPING to prevent their processes from being outswapped (even when not computable), you need to invoke the F$GETJPI lexical function for suspicious processes. (Suspicious processes are those that remain in the local event flag wait state for some time while the system is swapping heavily. You can observe that condition with the SHOW SYSTEM command.) If the flag PSWAPM in the status field (STS) is on, the process cannot be swapped. (The documentation for the system service $GETJPI specifies the status flags. See the *VAX/VMS System Services Reference Manual*).

As an alternative, you can use the ANALYZE/SYSTEM command to invoke SDA to issue its SHOW PROCESS command for the suspicious processes. Those that cannot be swapped will include the designation PSWAPM in the status line at the top of the display.

If you determine that one or more processes should be allowed to swap, you should seek agreement and cooperation from the users. (If agreement is reached, but users do not follow through, you could remove the users' PSWAPM and/or SETPRV privileges with the /PRIVILEGES qualifier of the Authorize Utility.) See Section 5.2.19.

4.2.4.7 Disk Thrashing Occurs

If you find that a large number of processes are computable at this point in your investigation, you should ensure that disk thrashing is not initiated by the outswapping of processes while they are computing. (Disk thrashing means excessive reading and writing to disk that accomplishes little; in this case, it is the outswapping of processes rapidly followed by the inswapping of the same processes.)

Processes in the computable outswapped (COMO) state on the MONITOR STATES display are normally those that have finished waiting for a local event flag and are ready to be inswapped. On a system without swapping, they are new processes. However, you may find computable outswapped processes that were swapped out while they were computable. Such undesirable swapping is harmful if it occurs too frequently.

A particular workload problem must exist to provoke this situation. Suppose a number of compute-bound processes attempt to run concurrently. The processes will not be changing states while they compute. Moreover, since they are computing they escape second-level swapper trimming to the SWPOUTPGCNT value. This condition can result in memory becoming scarce, which then could force the processes to begin swapping in and out among themselves. Whenever an outswapped process becomes computable, the scheduler is awakened to begin rescheduling. A process that is outswapped while it is computable also prompts immediate rescheduling. Thus, if the processes can not gain enough processing time from the CPU before being outswapped, and if they are outswapped while they are computable, thrashing occurs.

If you issue the SHOW SYSTEM command and note that many of the computable outswapped processes are at their base priority, you should check to be sure that the processes are not being swapped out while they are computable. (The fact that the processes are at their base priority implies they have been attempting to run for some time. Moreover, a number of COMO processes all at base priority strongly suggests that there is contention for memory among computable processes.)

You can issue the SHOW PROCESS/CONTINUOUS command for the COM processes and observe whether they fail to enter the LEF state before they enter the COMO state. Alternatively, you might observe whether their direct and buffered I/O rates remain low. Low I/O rates also imply that the processes have seldom gone into a local event flag wait state.

If you observe either indication that processes are being outswapped while computable, it is probable that too many highly computational processes are attempting to run concurrently, or that DORMANTWAIT is set too low. However, you should rule out the possible effects of too many batch jobs running at the same time, before you attempt to adjust the rate at which processes are inswapped.

Issue the DCL command SHOW SYSTEM/BATCH to determine the number of batch jobs running concurrently and the amount of memory they consume. If you conclude that the number of concurrent batch jobs could be affecting performance, you can reduce the demand they create by modifying the batch queues with the /JOB_LIMIT qualifier. Include this qualifier on the DCL command you use to establish the batch queue (INITIALIZE/QUEUE or START/QUEUE).

If you have ruled out any possible memory contention from large concurrent batch jobs, you can conclude that the solution involves correcting the frequency at which the system outswaps then inswaps the computable processes. Assuming the system parameter QUANTUM represents a suitable value for all other workloads on the system, you can draw the second conclusion. If you find the current priorities of the compute-bound processes are less than or equal to DEFPRI, you should consider increasing the special parameter SWPRATE, so that inswapping of compute-bound processes occurs less frequently. In that way, the computing processes will have a greater amount of time to run before they are outswapped to bring in the COMO processes. See Section 5.2.23.

4.2.4.8 System Swaps Rather Than Pages

If you have found a large number of computable processes that are not at their base priority, and if their working sets are fairly large yet not above their working set quotas, you should investigate whether any real paging is occurring. Even when there is no real paging, there may be paging induced by swapping activity. You can identify paging due to swapping whenever a high percentage of all the paging is due to global valid page faults. Use the display produced by the MONITOR PAGE command to evaluate the page faulting.

If you conclude that most of the paging is due to swapper activity, your system performance may improve if you induce some real paging by decreasing the working set sizes, an action which may reduce swapping. To induce paging, you might also reduce the automatic working set adjustment growth by lowering WSINC or increasing PFRATH. See Section 5.2.24.

4.2.4.9 Demand Exceeds Available Memory

If you reach this point in the investigation and still experience swapping in combination with degraded performance, you have ruled out all the appropriate ways for tuning the system to reduce swapping. The problem is that the available memory can not meet demand.

4.2.5 Analyzing the Limited Free Memory Symptom

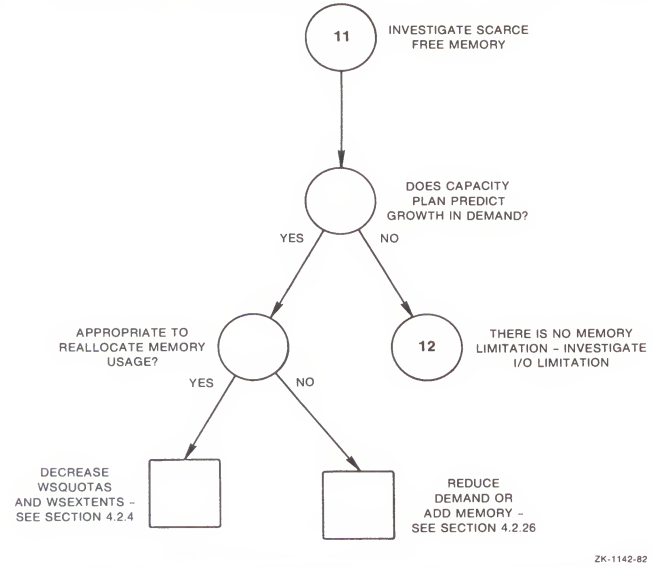
If you find that your system seems to run low on free memory at times, you are receiving advance warning that you are likely to encounter paging or swapping problems in the near future.

You should carefully investigate your capacity and anticipated demand. If you see little future growth in demand, then you are unlikely to experience a problem in the near future. However, if you see that your future demand will soon exceed your capacity, it is time to review all possible options. If you conclude that the only suitable option is to order memory, you may want to order it now, so that it can be installed before serious performance problems occur.

Before you decide to order more memory, you might want to look at how you have allocated memory. See Figure 4–10. Perhaps you could benefit by adjusting physical memory utilization so that the page cache is larger and there is less disk paging. To make this adjustment, you may have to relinquish some of the total working set space.

If working set space has been too generously configured in your system, you have found an important adjustment you can make before problems arise. Section 5.2.4.2 describes how to decrease working set quotas and working set extents.

Figure 4–10 Investigating Limited Free Memory



4.2.6 Special VAX–11/782 Tuning Considerations

If you have a VAX–11/782 attached processor system, you will find that excessive paging and/or swapping is a severe liability to performance. The handling of the page faults incurred on the secondary processor places on the primary processor scheduling overhead as well as memory management overhead. The most important single action you can take on a VAX–11/782 system to improve performance is to reduce paging and swapping, even if it means purchasing more memory.

4.3 Isolating I/O Limitations

At this point you have observed either a direct I/O rate or a buffered I/O rate, and need to determine if there could be an I/O limitation causing degraded system performance. Direct I/O (Section 4.3.1) is generated by *disks*, *tapes*, and some other types of high-speed communication links (such as the DR780). Buffered I/O (Section 4.3.4) can be produced by a number of devices, including terminals, line printers, the console disk drive, and communications devices (such as the DMR11).

4.3.1 Disk or Tape Operation Problems (Direct I/O)

Direct I/O problems for disks or tapes reveal themselves in long delay times for I/O completions. The easiest way to confirm a direct I/O problem is to detect a particular device with a queue of pending requests. A queue indicates contention for a device or controller. For disks, the MONITOR command MONITOR DISK/ITEM=QUEUE_LENGTH provides this information.

Since direct I/O refers to direct memory access (DMA) transfers that require relatively little CPU intervention, the performance degradation implies one or both of the following device-related conditions:

- The device is not fast enough.
- The aggregate demand on the device is so high that some requests are blocked while others are being serviced.

For a disk or tape I/O limitation that degrades performance, the only relatively low-cost solution available through tuning the software uses memory to increase the sizes of the caches and buffers used in processing the I/O operations, thereby decreasing the number of device accesses. The other possible solutions all involve purchasing additional hardware, which is much more costly.

4.3.2 Determining I/O Rates

When you issue the MONITOR IO command and observe evidence of direct I/O, you will probably be able to determine whether the rate is normal for your site. A direct I/O rate for the entire system that is either higher or lower than what you consider normal warrants investigation. See Figures 4–11 and 4–12.

You should proceed in this section only if you deem the operation rates of disk or tape devices to be significant among the possible sources of direct I/O on your system. If necessary, rule out any other possible devices as the primary source of the direct I/O with the lexical function F$GETDVI, as shown above.

Compare the I/O rates derived in this manner or observed on the display produced by the MONITOR DISK command, with the rated capacity of the device. (If you do not know the rated capacity, you should find it in literature published for the device, such as a peripherals handbook or a marketing specifications sheet.)

4.3.2.1 Device I/O Rate Is Below Capacity

Sometimes you may detect a lower direct I/O rate for a device than you would expect. This condition implies that either very large data transfers are not completing rapidly (probably in conjunction with a memory limitation centered around paging and swapping problems), or there are some other devices are blocking the disks or tapes.

If you have already investigated the memory limitation and taken all possible steps to alleviate it (which is the recommended step before investigating an I/O problem), then you should try to determine the source of the blockage.

A blockage in the I/O subsystem suggests that I/O requests are queueing up because of a bottleneck. For disks, you can determine that this condition is present with the MONITOR DISK/ITEM=QUEUE_LENGTH command.

When you find a queue on a particular device, you cannot necessarily conclude that that device is the bottleneck. At this point, simply note all devices with queues, for later reference. (You will need to determine which processes are issuing the I/O operations for the device with queues.)

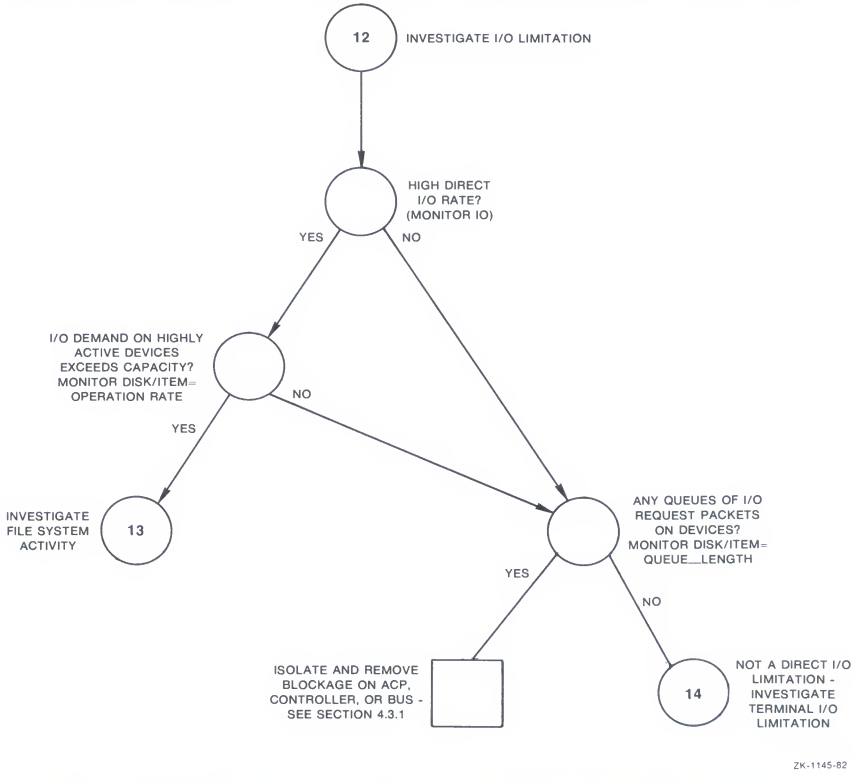
As the next step, you should rule out the possibility of an ancillary control process (ACP)-induced lockout situation. (Note that this condition arises only if you have ODS-1 disks.) If the system attempts to use a single ACP for both slow and fast devices, I/O blockages may occur when the ACP attempts to service a slow device. This situation can only occur if you have mounted a device with the /PROCESSOR qualifier.

4.3.2.2 Direct I/O Rate Is Abnormally High

An abnormally high direct I/O rate for any device, in conjunction with degraded system performance, suggests that I/O demand for that device exceeds its capacity. First, you need to find out where the I/O operations are occurring. Issue the MONITOR PROCESSES/TOPDIO command. From this display, you can determine which processes are heavy users of I/O, and, in particular, which processes are succeeding in completing their I/O operations—not which processes are waiting.

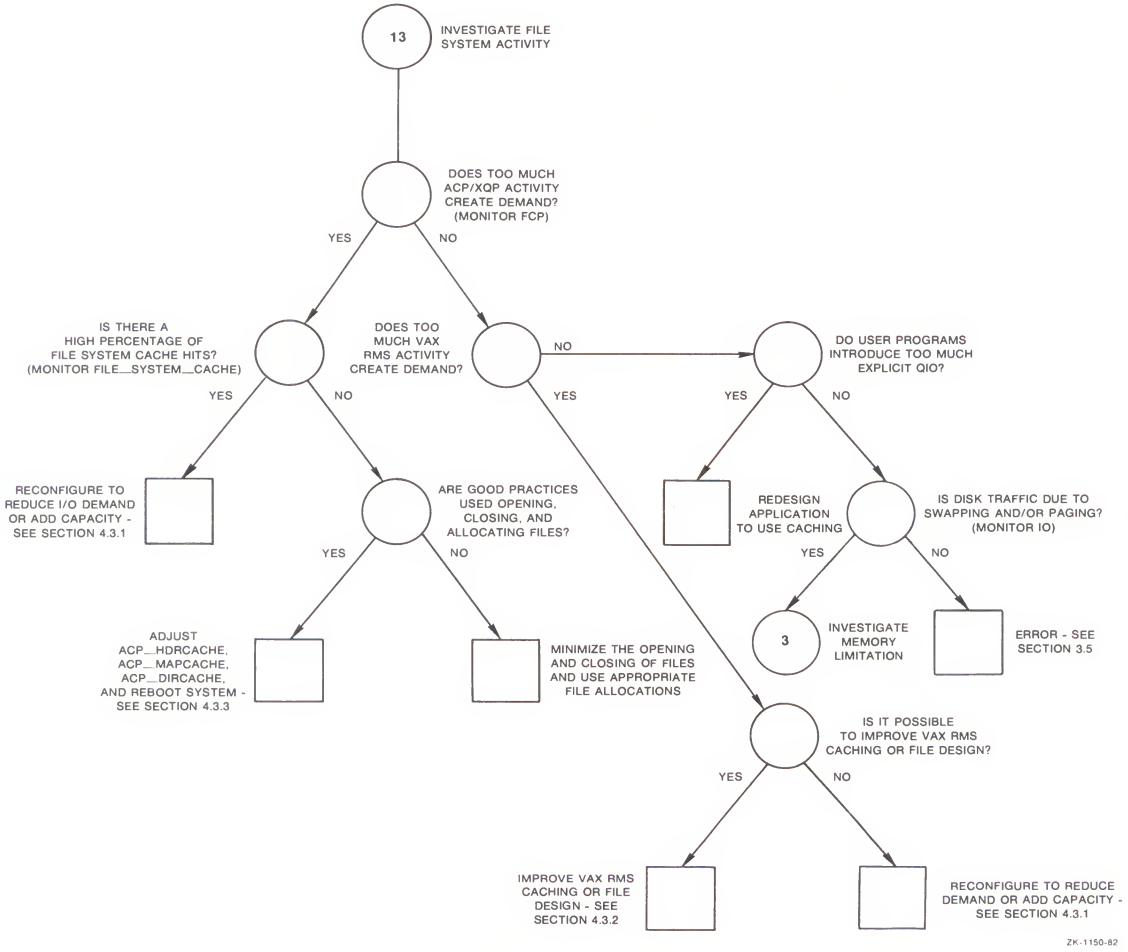
Next, you must determine which of the devices used by the processes that are the heaviest users of the direct I/O resource also have the highest operations counts, so that you can finally identify the bottleneck area. Here, you must know your workload sufficiently well to know the devices the various processes use. If you note that these devices are among the ones you found queued up, you have now found the bottleneck point(s).

Figure 4–11 Investigating Disk I/O Limitations—Phase I



Once you have identified the device that is saturated, you need to determine the types of I/O activities it experiences. Perhaps some of them are being mishandled and could be corrected or adjusted. Possibilities are file system caching, VAX RMS buffering, use of explicit QIOs in user programs, and paging or swapping. After you eliminate these possibilities, you may conclude that the device is simply unable to handle the load.

Figure 4–12 Investigating Disk I/O Limitations—Phase II



File System Caching Is Suboptimal

To evaluate the effectiveness of caching, observe the display produced by the MONITOR FILE_SYSTEM_CACHE command. If cache hits are 70 percent or greater, caching activity is normal. A lower percentage, combined with

large numbers of attempts, indicates that caching is less than optimally effective.

First, you should be certain that your applications are designed to minimize the opening and closing of files. You should also verify that the file allocation and extent sizes are appropriate. Use the DCL command DIRECTORY/SIZE=ALL to display the space used by the files and the space allocated to them. If the proportion of space used to space allocated seems close to 90 percent, no changes are necessary. However, significantly lower utilization should prompt you to set more accurate values, either explicitly or by changing the defaults, particularly on critical files. You use the RMS_EXTEND_SIZE system parameter to define the default file extents on a system-wide basis. The DCL command SET RMS_DEFAULT /EXTEND_QUANTITY permits you to define file extents on a per-process basis (or on a system-wide basis if you also specify the /SYSTEM qualifier). For more information, see the *Guide to VAX/VMS File Applications.*

If these are standard practices at your site, then you should see Section 5.3.3 for a discussion of how to adjust the following ACP system parameters: ACP_HDRCACHE, ACP_MAPCACHE, and ACP_DIRCACHE.

VAX RMS Errors Induce I/O Problem

Misuse of VAX RMS can cause direct I/O limitations. If users are blocked on the disks because of multiblock counts that are unnecessarily large, instruct the users to reduce the size of their disk transfers by lowering the multiblock count with the DCL command SET RMS_DEFAULT/BLOCK_COUNT. See Section 5.3.2.

If this course is partially effective, but the problem is widespread, you may decide to take action on a system-wide basis. You can alter one or more of the system parameter(s) in the RMS_DFMB group with AUTOGEN, or you can include the appropriate SET RMS_DEFAULT command in the system-wide login command procedure. See the *Guide to VAX/VMS File Applications.*

Explicit QIO Usage Is Too High

Next you need to determine whether any process using a device is executing a program that employs explicit specification of QIOs rather than VAX RMS. If you issue the MONITOR PROCESSES/TOPDIO command, you can identify the user processes worth investigating. It is possible that the user-written program is not designed properly. It may be necessary to redesign the program to use caching to improve the efficiency of the QIOs. Note that VAX/VMS does not provide any automatic caching for QIOs.

4.3.2.3 Disk Activity Is Due to Paging or Swapping

If you do not detect processes running programs with explicit user-written QIOs, you should suspect that the operating system is generating disk activity due to paging and/or swapping activity. The paging and/or swapping may be quite appropriate and not introducing any memory management problem. However, some aspect of the configuration is allowing this paging and/or swapping activity to block other I/O activity, introducing an I/O limitation. Issue the MONITOR IO command to inspect the Page Read I/O Rate and Page Write I/O Rate (for paging activity) and the Inswap Rate (for swapping activity). Note that since system I/O activity to the disk is not reflected in the direct I/O count that MONITOR provides, MONITOR IO is the correct tool to use here.

If you find indications of substantial paging and/or swapping at this point in the investigation, you need to consider whether the paging and/or swap files are located on the best choice of device, controller, or bus in the configuration. You should also consider whether introducing secondary files and separating the files would be beneficial. A later section discusses relocating the files to bring about performance improvements.

4.3.3 Reduce I/O Demand or Add Capacity

The only low-cost solutions that remain require reductions in demand. You could try to shift the workload so that less demand is placed simultaneously on the direct I/O devices. Or you might reconfigure the magnetic tapes and disks on separate buses to reduce demand on the bus. (If there are no other available buses configured on the system, you may want to acquire buses so that you can take this action.)

If none of the above solutions improved performance, you may need to add capacity. You probably need to acquire disks with higher transfer rates rather than simply add more disks. However, if you have been employing magnetic tapes extensively, you may want to investigate ways of shifting your applications to use disks more effectively. Section 5.3.1 provides a number of suggestions for reducing demand or adding capacity.

4.3.4 Terminal Operation Problems (Buffered I/O)

Terminal operation, when improperly handled, can present a serious drain on system resources. However, the resource that is consumed is the CPU, not I/O. Terminal operation is actually a case for CPU limitation investigation, but is included here because it may initially appear to be an I/O problem.

You will first suspect a terminal I/O problem when you detect a high buffered I/O rate on the display for the MONITOR IO command. See Figure 4–13. Next you should issue the MONITOR STATES command to check if processes are in the COM state. This condition, in combination with a high buffered I/O rate, suggests that the CPU is constricted by terminal I/O demands. If you do not observe processes in the computable state, you should conclude that while there is substantial buffered I/O occurring, the system is handling it well. In that case, the problem lies elsewhere. Proceed to Section 4.4 to investigate other forms of CPU limitation.

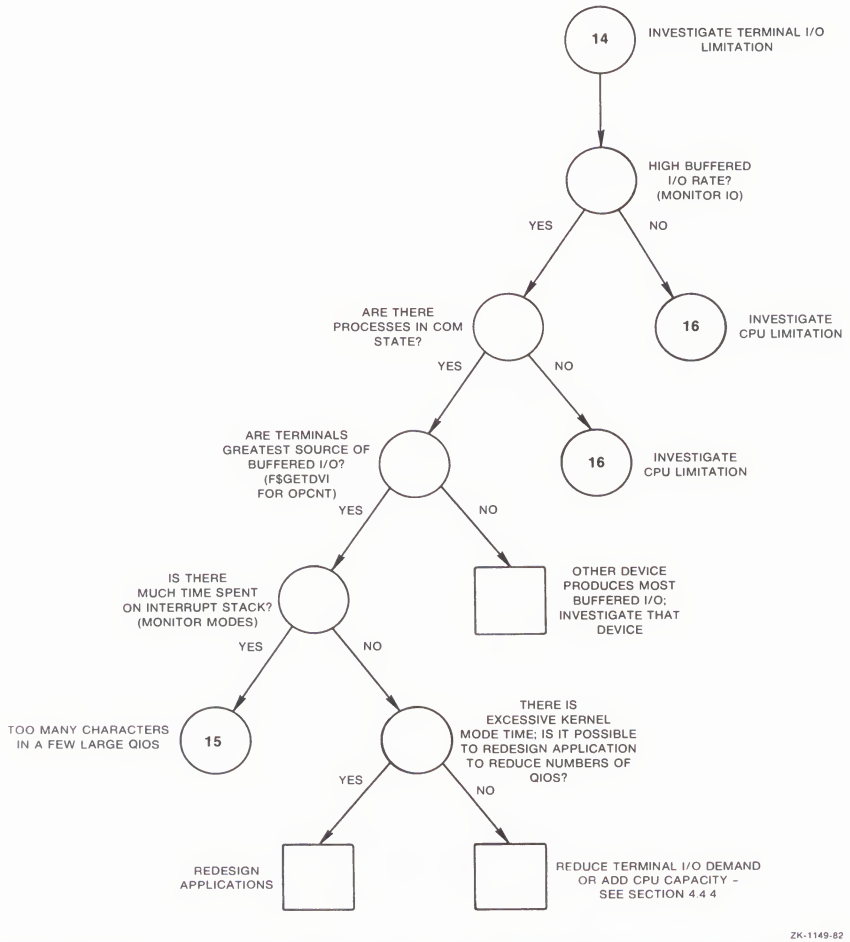
If you do observe processes in the COM state, you must verify that the high buffered I/O count is actually due to terminals, not to communications devices, line printers, graphics devices, non-DIGITAL devices or instrumentation, or devices that emulate terminals. You must examine the operations counts for all such devices with the lexical function F$GETDVI. (See Section 4.3.2.) A high operations count for any device other than a terminal device indicates that you should explore the possibility that the other device is consuming the CPU resource.

If you find that the operations count for terminals is a high percentage of the total buffered I/O count, you can conclude that terminal I/O is degrading system performance. To further investigate this problem, issue the MONITOR MODES command. From this display you should expect to find much time spent either on the interrupt stack or in kernel mode. Too much time on the interrupt stack suggests that too many characters are being transmitted in a few very large QIOs. Too much time in kernel mode may indicate that too many small QIOs are occurring.

4.3.4.1 Interrupt Stack Time Is Excessive

If interrupt stack time is excessive, if you know that most of the terminal I/O is for output, you might achieve improvement by replacing DZ11s or DZ32s with a device capable of burst output, such as the DMF32. Burst output means the output of a large number of characters at once for each interrupt. The DMF32 is an example of a device that offers two forms of burst output: silo mode transfers and direct memory access (DMA) transfers. A device that can handle burst output can reduce time on the interrupt stack for terminal I/O output, by transferring larger numbers of characters at once and interrupting only when the transfer is completed. Thus, you potentially eliminate a significant number of interrupts, plus the time spent in the

Figure 4–13 Investigating Terminal I/O Limitations—Phase I



interrupt service routine for each interrupt. Section 5.3.4 discusses burst output devices, including the applications where they provide the maximum benefit, in much greater detail.

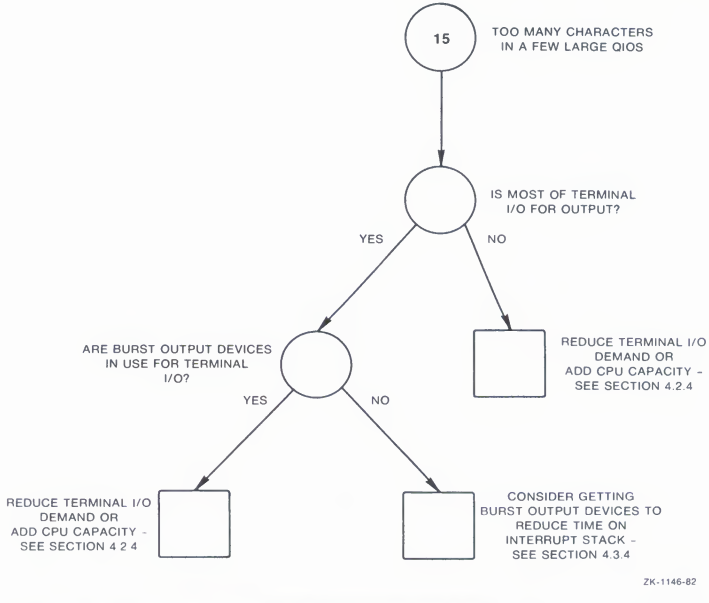
If you find that burst transfers are not likely to reduce the primary source of your terminal I/O limitation, you must either explore ways to reduce demand or add CPU capacity. See Section 5.3.4.4.

4.3.4.2 Kernel Mode Time Is Excessive

If the MONITOR MODES display shows much time spent in kernel mode, perhaps the sheer number of QIOs involved is burdening the CPU. See Figure 4–14. You should explore whether the application can be redesigned to group the large number of QIOs into smaller numbers of QIOs that transfer more characters at a time. Such a design change could alleviate the condition, particularly if burst output devices are in use. It is also possible that some adjustment in the workload is feasible that would balance the demand.

If neither of these approaches is possible, you need to reduce demand or increase the capacity of the CPU. See Section 5.3.4.4.

Figure 4–14 Investigating Terminal I/O Limitations—Phase II



4.4 Isolating CPU Limitations

The surest way to determine whether a CPU limitation could be degrading performance is to check for a state queue with the MONITOR STATES command. If any processes appear to be in the COM or COMO state, a CPU limitation may be at work. However, if no processes are in the COM or COMO states, you need not investigate the CPU limitation any further.

If processes are in the COM or COMO state, they are being denied access to the CPU. One or more of the following conditions is occurring:

- Processes are blocked by the execution of another process at higher priority.
- Processes are time-slicing with other processes at the same priority.
- Processes are blocked by excessive activity on the interrupt stack.
- Processes are blocked by some other resource. (Note that this last possibility means the limitation is not a CPU limitation but is instead a memory or I/O limitation.)

4.4.1 Processes Are Blocked by a Higher-Priority Process

If you suspect the system is performing suboptimally because processes are blocked by a process running at higher priority, you need access to an account that is already running. As a first step, ensure you have the ALTPRI privilege; then set your priority to 15 with the DCL command SET PROCESS /PRIORITY=15. Note that without this much access, it may not be possible to further investigate the problem, unless you crash the system and look at a crash dump. However, assuming you can gain access to the system, as Figure 4–15 shows, you can check for a high priority lockout by issuing the MONITOR PROCESSES/TOPCPU command. If you then examine (with the SHOW PROCESS/CONTINUOUS command) the current and base priorities of those processes that you found were the top users of the CPU resource, you can conclude whether any process is responsible for blocking lower-priority processes.

If you find that this condition exists, your option is to adjust the process priorities. See Section 5.4.1 for a discussion of how to change the process priorities assigned in the UAF, to define priorities in the login command procedure, or to change the priorities of processes while they execute.

Remember to restore the priority of the process you used for the investigation. Otherwise, you may find that process causes its own system performance problem.

4.4.2 Time-Slicing Occurs Between Processes

Once you rule out the possibility of preemption by higher-priority processes, you need to determine if there is a serious problem with time-slicing between processes at the same priority. Using the list of top CPU users, compare the priorities and assess how many processes are operating at the same one. If you conclude that the priorities are inappropriate, you can follow the steps outlined in Section 5.4.1 to change them.

However, if you decide that the priorities are correct and will not benefit from such adjustments, you are confronted with a situation that will not respond to any form of system tuning. Again, the only appropriate solution here is to adjust the workload to decrease the demand or add CPU capacity. See Section 5.4.4.

4.4.3 Interrupt Stack Activity Is Excessive

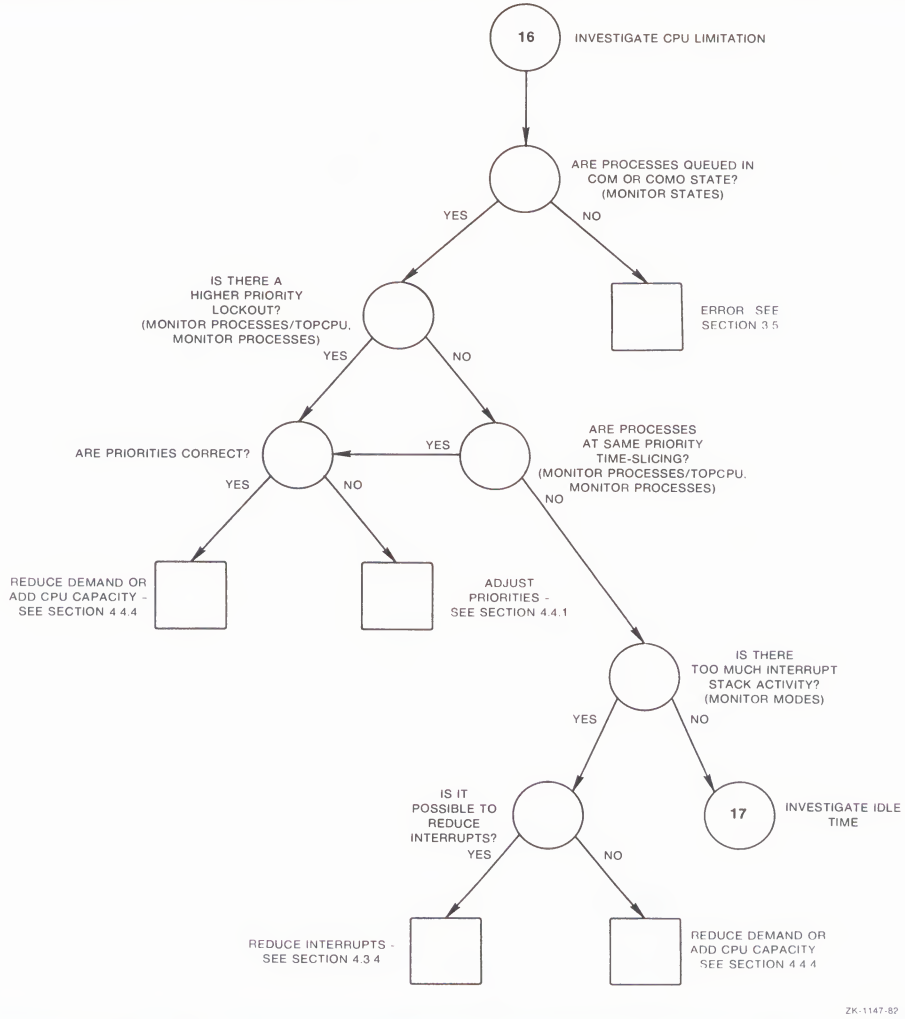
If you discover that blocking is not due to contention with other processes at the same or higher priorities, you need to find out if there is too much activity on the interrupt stack. In other words, is the rate of interrupts so excessive that it is preventing processes from using the CPU?

You can determine how much time is spent on the interrupt stack from the MONITOR MODES display. If the percentage of time on the interrupt stack is less than 10 percent, you could view this as moderate. However, should you observe percentages of 20 percent or more on processors other than VAX–11/780s, you should consider this time excessive. (The higher the percentage, the more effort you should dedicate to solving this resource drain.)

If the interrupt stack time is excessive, you need to explore which devices cause significant numbers of interrupts on your system and how you might reduce the interrupt rate. For example, if terminals provide the primary source of interrupts, perhaps the system performance would benefit by using DMF32s instead of DZ11s or DZ32s, so that terminal I/O is transferred in a burst. (See Sections 4.3.4 and 5.3.4 for a discussion of the use of burst output devices to reduce excessive time on the interrupt stack.)

The decisions you make will depend on the source of heavy interrupts. Perhaps they are due to communications devices or special hardware used in real-time applications. Whatever the source, you need to find ways to reduce the number of interrupts, so that the CPU can handle work from other processes. Otherwise, the solution may require you to adjust the workload or acquire CPU capacity. See Section 5.4.4.

Figure 4–15 Investigating Specific CPU Limitations—Phase I



4.4.3.1 Memory Limitation Is Disguised

Once you have either ruled out or resolved the above types of CPU limitation block, you need to determine which other resource limitation produces the

block. Your next check should be for the amount of idle time. See Figure 4–16. Use the MONITOR MODES command. If there is any idle time, another resource is the problem, and you may be able to tune for a solution. If you reexamine the MONITOR STATES display, you will likely observe a number of processes in the COMO state. You can conclude that this condition reflects a memory limitation, not a CPU limitation. Follow the procedures described in Section 4.2 to find the cause of the blockage, and then take the corrective action recommended in Chapter 5.

4.4.4 Operating System Incurs Overhead

If, however, the MONITOR MODES display indicates that there is no idle time, your CPU is 100 percent busy. You will find that processes are in the COM state on the MONITOR STATES display. You must answer one more question. Is the CPU being used for real work or for nonessential operating system functions? If you detect there is operating system overhead, you may be able to reduce it.

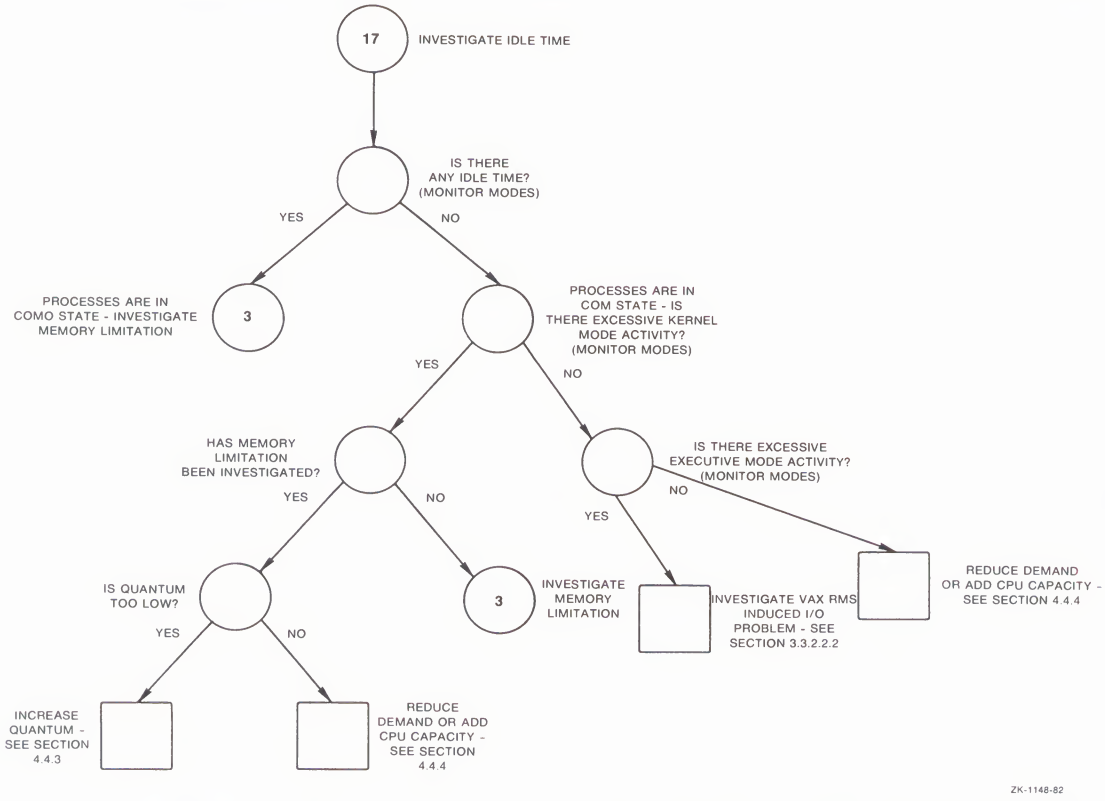
You must analyze the MONITOR MODES display carefully. If your system exhibits excessive kernel mode activity, it is possible that the operating system is incurring overhead in the areas of memory management, I/O handling, or scheduling. You should investigate the memory limitation and I/O limitation (Sections 4.2 and 4.3), if you have not already done so.

Once you rule out the possibility of improving memory management or I/O handling, the problem of excessive kernel mode activity may be due to scheduling overhead. However, you can do practically nothing to tune the scheduling function. There is only one case that might respond to tuning. The clock-based rescheduling that can occur at quantum-end is costlier than the typical rescheduling that is event driven by process state. Explore whether the value of the system parameter QUANTUM is too low and can be increased to bring about a performance improvement by reducing the frequency of this clock-based rescheduling (Section 5.4.3). If not, your only other recourse is to adjust the workload or acquire CPU capacity. See Section 5.3.4.4

4.4.5 VAX RMS Is Misused

If the MONITOR MODES display indicates that a great deal of time is spent in executive mode, it is possible that VAX RMS is being misused. If you suspect this problem, proceed to the steps described in Section 4.3.1 for VAX RMS-induced I/O limitations, making any changes that seem indicated. You should also consult the *Guide to VAX/VMS File Applications.*

Figure 4–16 Investigating Specific CPU Limitations—Phase II



4.4.6 CPU Is Operating at Full Capacity

If, however, at this point in your investigation the MONITOR MODES display indicates that most of the time is spent in supervisor mode, user mode, or compatibility mode, you are confronted with a situation where the CPU is performing real work and the demand exceeds the capacity. You must either make adjustments in the workload to reduce demand (by more efficient coding of applications, for example), or you must add CPU capacity. See Section 5.4.4.

4.5 Conclusion

At this point, you should know what particular resource is limited, and you should know which section of Chapter 5 suggests one or more possible remedies. If not, you may be making an error interpreting the output of one or more of the suggested tools. You should repeat the work you did for this chapter and then, if necessary, consider consulting your DIGITAL software specialist.

After you perform the recommended corrective actions in Chapter 5, you should repeat the steps in this chapter to observe the effects of the changes. As you repeat the steps, watch for the introduction of new problems introduced by the corrective actions or the discovery of previously undetected problems. Your goal should be to complete the steps in this chapter without uncovering a single serious symptom or problem.

5 Compensating For Resource Limitations

This chapter describes corrective procedures for each of the various categories of resource limitations described in Chapter 4.

Wherever the corrective procedure suggests changing the value of one or more system parameters, the description explains briefly whether the parameter should be increased, decreased, or given a very specific value. Relationships between parameters are identified and explained, if necessary. However, to avoid duplicating information available in the *VAX/VMS System Generation Utility Reference Manual*, complete explanations of parameters are not included. You should review descriptions of system parameters, as necessary, before changing the parameters.

5.1 Changing System Parameters

In most cases, you will want to use AUTOGEN to change system parameters, since AUTOGEN adjusts related parameters automatically. (For a discussion of AUTOGEN, refer to the *VAX/VMS System Manager's Reference Manual*.) In the few instances where it is appropriate to change a parameter in the special parameter group, more explanation of the parameter is given in this chapter, since special parameters are otherwise undocumented. Before you make any changes to your system parameters, however, make a copy of the existing version of the file that is in the SYSGEN work area, using a technique such as the following:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  WRITE SYS$SYSTEM:file-spec
SYSGEN>  EXIT
```

You may want to use a date as part of the file name you specify for file-spec, to readily identify the file later.

By creating a copy of the present values, you can always return to those values at some later time. Generally you use the following technique, specifying your parameter file for file-spec:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  USE SYS$SYSTEM:file-spec
SYSGEN>  WRITE ACTIVE
SYSGEN>  EXIT
```

However, if some of the parameters you changed were not dynamic, to restore them from the copied file you must instead use the SYSGEN command WRITE CURRENT, and then reboot the system.

If your tuning changes involve system parameters that are dynamic, plan to test the changes on a temporary basis first. This is the only instance where the use of SYSGEN is warranted for making tuning changes. (Once you are satisfied that the changes are working well, you should invoke AUTOGEN with the REBOOT parameter to make the changes permanent.)

If you are planning to change a system parameter and you are uncertain of an ultimate target value and also of the sensitivity of the specific parameter to changes, you should err on the conservative side in making initial changes. As a guideline, you might make a 10 percent change in the value first, so that you can observe its effects on the system. If you see little or no effect, you might try doubling or halving the original value of the parameter, depending on whether you are increasing or decreasing the parameter. If this magnitude of change has no effect, you should restore the parameter to its original value with the parameter file you saved before starting. If you cannot affect your system performance with changes of this magnitude, it is doubtful that you have selected the right parameter for change.

You should change only a few parameters at a time.

After you change system values or parameters, you must monitor the results, as described in Section 1.4. You have two purposes for monitoring. First, you must ensure that the changes are not inducing new problems. Second, you must evaluate the degree of success achieved.

You may want to return to the appropriate procedures of Chapter 4, as you evaluate your success after tuning and decide whether to pursue additional tuning efforts. However, always keep in mind that there is a point of diminishing returns in every tuning effort (see Section 1.5).

Whenever your changes are unsuccessful, make it a practice to restore the parameters to their previous values before you continue tuning. Otherwise, it can be difficult to determine which changes produce currently observed effects.

5.2 Compensating for Memory-Limited Behavior

The following sections describe procedures to remedy specific conditions that you may have detected as the result of the investigation described in Section 4.2.

5.2.1 Reduce Number of Image Activations

There are several ways to reduce the number of image activations. You and the programming staff should explore them all and apply those you deem feasible and likely to produce the greatest results.

Excessive image activations can result from running large command procedures frequently, since all DCL commands (except those performed within the command interpreter) require an image activation. If command procedures are introducing the problem, consider writing programs to replace them.

When code is actively shared, the cost of image startups decreases. Perhaps your installation has failed to design applications that share code. You should examine ways to employ code sharing wherever suitable. See Sections 1.1.3 and 2.2.3. (Note that you will not see the number of image activations drop when you begin to use code sharing, but you should see an improvement in performance. The effect of code sharing is to shift the type of faults at image activation from hard faults to soft faults, a shift that results in performance improvement.)

Yet another source of excessive image activations is migration of programs from other operating systems to a VAX/VMS system without any design changes. For example, programs that employ the chaining technique on another operating system will not use memory efficiently on a VAX/VMS system if you simply recompile them and ignore design differences. When converting applications to run on a VAX/VMS system, always consider the benefits of designing and coding each application for native mode operation.

5.2.2 Increase Page Cache Size

You can enlarge the page cache by simply increasing the four page cache parameters: FREEGOAL, FREELIM, MPW_THRESH, and MPW_LOLIMIT. It is not necessary to remove balance slots or to reduce the working set size of any of the processes.

You should first increase the number of pages on the free page list, by augmenting FREELIM and FREEGOAL. Aim to provide at least one page on the free page list for every process. FREEGOAL must always be greater than FREELIM. Generally a good target size for FREEGOAL is three times FREELIM. If you feel your workload warrants it, you can increase the modified page list size by increasing MPW_THRESH and MPW_LOLIMIT. Generally MPW_LOLIMIT should be less than 10 percent of physical memory.

You may optionally decide to reduce the number of balance slots, as described in Section 5.2.13.

5.2.3 Decrease Page Cache Size

You decrease the size of the page cache by reducing the values for the system parameters MPW_LOLIMIT, MPW_THRESH, FREEGOAL, and FREELIM, maintaining the ratios suggested in Section 5.2.2 above.

In general, acceptable performance can be obtained by a page cache size that is one order of magnitude less than the available space for it and the working sets.

5.2.4 Adjust Working Set Characteristics (for Quota and Extent)

You have concluded that the working set quota and/or working set extent characteristics are incorrect in some cases. The corrective action depends on how the values were established. You must know whether the values affect a process, subprocess, detached process, or batch job.

Furthermore, if you need to fix a situation that currently exists, you must evaluate the severity of the problem. In some cases, you may have to stop images or processes, or ask users to log off to permit your changes to become effective. You would only take such drastic action if the problem creates intolerable conditions that demand immediate action.

In addition to making specific changes in the working set quota and working set extent values, you should also address the need to modify the values of the system parameters BORROWLIM and GROWLIM. Section 5.2.5 includes a discussion of these changes.

Whenever you increase the values for working set extents, you should compare your planned values to the system parameter WSMAX, which specifies (on a system-wide basis) the maximum size that the working sets can achieve. It will do no good to specify any working set extent that exceeds WSMAX, since the working set can never actually achieve a count above the value of WSMAX. If you specify such a value, you should also increase WSMAX.

5.2.4.1 Establish Values for Ancillary Control Processes (ODS–1 Only)

This section will be of interest only if you are using ODS–1 disks.

Before studying the considerations for adjusting working set sizes for processes in general, consider the special case of the ancillary control process (ACP). (Note that you will be using an ACP for disks only if you have ODS-1 disks.) The default size of the working set (and, in this case, the working set quota, too) for all ACPs is determined by the system parameter ACP_WORKSET. If ACP_WORKSET is zero, the system calculates the working set size for you. If you want to provide a specific value for the working set default, you just specify the desired size in pages with AUTOGEN. (If your system uses multiple ACPs, remember that ACP_WORKSET is a system-wide parameter: any value you choose must apply equally well to all ACPs.)

If you decide to reduce ACP_WORKSET (with the intent of inducing modest paging in the ACP), use the SHOW SYSTEM command to determine how much physical memory the ACP currently uses. Then simply calculate the value that is 90 percent of the ACP's current usage. Set the system parameter ACP_WORKSET to the reduced value you calculate. However, to make the change effective for all ACPs on the system, not just the ones created after the change, you must reboot the system.

Once you reduce the size of ACP_WORKSET, observe the process with the SHOW SYSTEM command to verify that the paging you have induced in the ACP process is moderate. Your goal should be to keep the total number of page faults for the ACP below 20 percent of the direct I/O count for the ACP.

5.2.4.2 Establish Values for Other Processes

The following discussion applies to all processes other than ACPs. If the values were established for processes based on the defaults in the UAF, you should seek out the user, describe the intended change, and ask the user to issue the DCL command SET WORKING_SET/EXTENT and/or SET WORKING_SET/QUOTA, as appropriate.

If you observe satisfactory improvement from the new values, you must decide if the benefit would apply whenever the process runs or just during some specific activities. For specific cases, the user should issue the SET WORKING_SET command when needed. For a more consistent change, you would need to modify the UAF.

To modify values in the UAF, you invoke AUTHORIZE and use the SHOW and MODIFY commands to modify the values /WSQUOTA and /WSEXTENT for one or more users. If the SHOW command reveals that the values are the same as the defaults, probably the defaults have been applied. You should change all the assigned values in the existing records in the UAF,

as appropriate. Then you should also modify the DEFAULT record in the UAF so that new accounts will receive the desired values.

If the working set characteristic values were adjusted by the process through the DCL command SET WORKING_SET or by a system service, you must convince the owner of the process that the values were incorrect and should be revised.

If the values were adjusted with the SET WORKING_SET comand, the user can simply issue the command again, with revised values. However, if values were established by system services, and the process is currently running and causing excessive paging, either the user must stop the image with CTRL/Y or else you must stop the process with the DCL command STOP. (Changing values set by system services typically requires code changes in the programs before they are run again.)

5.2.4.3 Establish Values for Detached Processes or Subprocesses

If the problem is introduced by a detached process or subprocess, you must also determine how the values became effective. If the values were established by the RUN command, they can only be changed if the user stops the detached process or subprocess (if it is running) and thereafter always starts it with a revised RUN command. (The user can stop the detached process or subprocess with the DCL command STOP.)

If the values were introduced by a system service, it is also necessary to stop the running detached process or subprocess, but code changes will be necessary as well.

If, however, the values were established by default, you may want to revise the values of the system parameters PQL_DWSEXTENT and/or PQL_DWSQUOTA, particularly if the problem appears to be widespread. If the problem is not widespread, you can to request users to use specific values that are less than or equal to their UAF defaults. (Unprivileged users cannot request values that will exceed their authorized values in the UAF. If such an increase is warranted, change the UAF records.)

5.2.4.4 Establish Values for Batch Jobs

If the problem is introduced by a batch job, you must determine the source of the working set values.

If the values are those established for the queue when it was initialized, you cannot change them for this job while it is running. You must reinitialize the queue if you determine the changes would be beneficial for all future batch jobs. To reinitialize a batch queue, you must first stop it with the DCL command STOP/QUEUE, then restart it with the DCL command START /QUEUE. If the new working set values produce good results, you should ask the user to submit the job with the appropriate values in the future.

If the working set characteristics are obtained by default from the user's UAF, you might consider assigning values to the batch queues or creating additional batch queues. If you prefer to have values assigned from the UAF, but have discovered instances where the best values are not in effect, before you change the UAF records, you need to determine if the changes would be beneficial at all times, or only when the user submits certain jobs. It is generally better to ask the users to tailor each submission than to either change UAF values that affect all the user's activities or change batch queue characteristics that affect all batch jobs.

5.2.5 Tune to Make Borrowing More Effective

If you have found few processes are taking advantage of loans, you should consider making the following adjustments:

- Decrease PFRATH
- Decrease BORROWLIM and/or GROWLIM
- Increase the process limit WSEXTENT

In decreasing PFRATH, you will increase the rate at which processes increase their working sets with automatic working set adjustment. (See Section 2.2.1 for a complete description of automatic working set adjustment and its parameters. See Section 5.2.6 for guidelines for initial settings of the parameters.)

When you decrease BORROWLIM and/or GROWLIM, consider how much working set space you would like all processes to be able to obtain, according to the guidelines presented in Section 2.2.1. As a rough guideline, you could target for a BORROWLIM value between one-third to one-half of available memory, and a GROWLIM value between one-sixth to one-fourth of available memory.

Be generous in establishing values for the working set extents, since the memory is only used when needed. As a general practice, set the working set extent value to the largest value you expect will be needed. Section 5.2.4 describes the various ways you adjust the working set extent characteristic. (You might also need to increase WSMAX.)

5.2.6 Tune AWSA to Respond Quickly to Increased Demand

You may want to increase the response from automatic working set adjustment to paging so that AWSA rapidly establishes a working set size that keeps paging to a reasonable rate for your configuration and workload. To do so, you need to reduce PFRATH and/or increase WSINC.

Think of PFRATH as the target maximum paging rate for any process in the system. PFRATH should always be greater than PFRATL. As a rule, values of PFRATH larger than 500 (which specifies a desired maximum rate of 50 page faults per second of CPU time) or smaller than 10 are unreasonable.

The system parameter WSINC defines the number of pages by which the working set limit increases when AWSA determines that it needs to expand. The maximum practical value for this parameter is therefore the difference between WSMAX (which is the maximum size increase that any working set can experience) and MINWSCNT (which is the minimum working set size). In practical terms, however, to avoid wasting memory, it makes sense to set WSINC smaller than this difference. A fairly good rule of thumb is to set WSINC to match an approximation of a typical user's WSDEFAULT value. Such a value allows the processes to increase fairly rapidly, while limiting the potential maximum waste to the amount needed to minimally support one user.

If you are not fully satisfied with the results produced by tuning WSINC and PFRATH, you could decrease AWSTIME. However, do not decrease the value of AWSTIME below the value of QUANTUM. Your goal should be to achieve a value for AWSTIME that follows the overall trend in the total size of all the working sets. If you establish too small a value for AWSTIME, automatic working set adjustment may be responding to too many frequent drastic working set size changes and not to the overall trend the changes describe.

5.2.7 Disable Voluntary Decrementing

If you find that some of the working set sizes are oscillating continuously while the processes should be in a stable state of memory demand, it is possible that voluntary (time-based) decrementing is forcing paging. To avoid this, set PFRATL to zero. This will effectively turn off voluntary decrementing. As a result, your system will rely solely on load-based memory reclamation (swapper trimming or outswapping).

Optionally you may want to set WSDEC to zero. If you do, it will be more obvious to you or others at some future time that voluntary decrementing is turned off on the system. However, setting only WSDEC to zero does not disable the checking that automatic working set adjustment performs for voluntary decrementing.

5.2.8 Tune Voluntary Decrementing

It may be the case that some time-based working set trimming is desirable to reclaim memory that is not really needed (to avoid taking needed memory away from other processes, for example). However, the parameters are set so high that too much paging occurs. In this case, you should decrease WSDEC or PFRATL. Setting just PFRATL to zero or setting both WSDEC and PFRATL to zero turns off time-based decrementing. However, if you choose to maintain some voluntary decrementing, remember that to avoid fixed oscillation, WSDEC should be smaller than WSINC. In addition, WSINC and WSDEC should be relatively prime (that is, WSINC and WSDEC should have no common factors). A good starting value for WSDEC would be an order of magnitude smaller than a typical user's WSDEFAULT value.

5.2.9 Turn on Voluntary Decrementing

There is also the case where time-based shrinking is completely turned off when it should be turned on. To turn WSDEC and PFRATL on, set WSDEC and PFRATL both greater than zero and observe the guidelines in Section 5.2.8.

5.2.10 Enable AWSA

To turn on the part of automatic working set adjustment that permits processes to increase their working set sizes, you must set WSINC to a value greater than zero. The default parameter settings established by AUTOGEN at system installation are good starting values for most workloads and configurations.

5.2.11 Adjust Swapper Trimming

When you determine that a paging problem is caused by excessive swapper trimming, SWPOUTPGCNT is too small. There are two approaches you can use. The first is to increase SWPOUTPGCNT to a value that is large enough for a typical process on the system to use as its working set size. This approach effectively causes the swapper to swap the processes at this value rather than reduce them to a size that forces them to page heavily.

The second approach completely disables second-level swapper trimming by setting SWPOUTPGCNT to a value equal to the largest value for WSQUOTA for any process on the system. This has the effect of shifting the bulk of the memory management to outswapping, with no second-level swapper trimming.

In conjunction with swapper trimming, the system uses the system parameter LONGWAIT to control how much time must pass before a process is considered idle. The swapper considers idle processes to be better candidates for memory reclamation than active processes. The ideal value for LONGWAIT is the length of time that accurately distinguishes an idle or abandoned process from one that is momentarily inactive. Typically this value is in the range of 3 to 20 seconds. You would increase LONGWAIT to force the swapper to give processes a longer time to remain idle before they become eligible for swapping or trimming. This approach will prove most productive when the workload is mixed and includes interactive processes. If the workload is composed primarily of non real-time processes, you may consider increasing DORMANTWAIT.

5.2.12 Convert to a System That Rarely Swaps

To reduce the swapping activity on a system severely, you can set the system parameter BALSETCNT equal to a value that is two less than the value of the system parameter MAXPROCESSCNT, thus allowing the maximum number of processes to operate concurrently. At the same time you should set the system parameter SWPOUTPGCNT to a minimum value, such as 60 pages.

As a secondary action, you would reduce the working set quotas, following the recommendations included in Section 5.2.4.

These actions produce a system that primarily pages.

5.2.13 Adjust BALSETCNT

You may want to use the BALSETCNT system parameter as a tuning control for paging or swapping. Reducing BALSETCNT may reduce paging, while increasing BALSETCNT can be used to decrease swapping. BALSETCNT is a parameter that affects a number of other parameters, so you should be conservative in changing it.

5.2.13.1 Reduce BALSETCNT to Reduce Paging

If you reduce the number of balance set slots by decreasing the parameter BALSETCNT, you can potentially reduce the demand for memory by limiting the number of processes that compete for memory at a given time.

From the output provided by the DCL command SHOW MEMORY under a very heavy workload, you know the number of balance slots available and in use. If balance slots are available under heavy load, it is safe to reduce the value of BALSETCNT by that amount. However, if no balance slots are available and you reduce BALSETCNT, you are likely to force swapping to occur while the system is loaded.

5.2.13.2 Increase BALSETCNT to Decrease Swapping Problems

If active swapping is being caused by a lack of balance slots when there is available memory, the first step is to increase BALSETCNT. The easiest thing to do is to set BALSETCNT equal to a value that is two less than the system parameter MAXPROCESSCNT. This guarantees that a balance slot will be available for any process that can be created. Swapping will then be forced only when memory is insufficient to meet the demand.

Rather than immediately setting BALSETCNT equal to a value that is two less than MAXPROCESSCNT (which is the largest useful value), you might try a more gradual approach. Divide the remaining free memory (as displayed by the SHOW MEMORY command) by the size of a typical working set, and then increase BALSETCNT by this number.

5.2.14 Reduce Large Page Caches

If active swapping is caused by a lack of free memory, which in turn is caused by unnecessarily large page caches, as a first step reduce the size of the caches by lowering FREELIM and FREEGOAL, or MPW_LOLIMIT and MPW_THRESH. (Remember that MPW_HILIMIT relates to the maximum size of the modified page list, rather than the target minimum size.)

Good starting ratios for these parameters are given in Section 5.2.2. Keep in mind that the problem of overly large caches is caused by mistuning in the first place. The AUTOGEN command procedure will not generate page cache values that are excessively large.

5.2.15 Curtail Large, Compute-Bound Process

Before suspending a large, low-priority, compute-bound process, it is strongly recommended that you curtail its memory allocation. If the process has not had a significant event for 10 seconds or more (page fault, direct or buffered I/O, CPU time allocation), you can decrease DORMANTWAIT to make the process a more likely outswap candidate.

5.2.16 Suspend Large, Compute-Bound Process

When you decide to suspend a large, compute-bound process, be sure that it is not sharing files with other processes. Otherwise, the large, compute-bound process may have a shared file locked when you suspend it. If this should happen, you will soon observe that other processes become stalled. You must resume the large, compute-bound process as soon as possible with the DCL command SET PROCESS/RESUME. If you are unable to achieve the benefit suspending offers. In this case, refer to Section 5.2.4.2 for appropriate corrective action.

5.2.17 Control Growth of Large, Compute-Bound Processes

When it becomes clear that a large, compute-bound process gains control of more memory than is appropriate, you may find it helpful to lower the process's working set quota. You would take this action if you conclude that this process should be the one to suffer the penalty of page faulting, rather than forcing the other processes to be outswapped too frequently. Section 5.2.4.2 describes how to make adjustments to working set quotas.

5.2.18 Enable Swapping for Disk ACPs (ODS–1 Only)

If a disk ACP has been set up so that it will not be outswapped and you determine that the system would perform better if it were, you must use AUTOGEN to modify the system parameter ACP_SWAPFLGS and then reboot the system. The *VAX/VMS System Generation Utility Reference Manual* describes how to specify the flag value for ACP_SWAPFLGS that will permit swapping of the ACP.

5.2.19 Enable Swapping for Other Processes

If you determine that users have been disabling swapping for their processes and that the effect of locking one or more processes in memory has been damaging to overall performance, you must explore several options.

Should there be no valid reason to disable swapping for one or more of the processes, you must convince the users to stop the practice. If you are unable to win their cooperation, you can remove privileges so they can not disable swapping. (The PSWAPM privilege is required to issue the SET PROCESS/NOSWAPPING command.) You use the Authorize Utility to change privileges.

However, if the users have valid reasons for disabling swapping, you should carefully examine what jobs are running concurrently when the performance degrades. It is possible that rescheduling a few of the jobs will be sufficient to improve overall performance. See Section 5.2.25.

5.2.20 Reduce Number of Concurrent Processes

You can reduce the number of concurrent processes by lowering the value of MAXPROCESSCNT. A change in that value has implications for the largest number of system parameters. Therefore, you should change the value of MAXPROCESSCNT in conservative steps.

5.2.21 Discourage Working Set Loans When Memory Is Scarce

If working sets are too large because processes are using their loan regions (above WSQUOTA), you can curtail loaning by increasing GROWLIM and BORROWLIM. (To completely disable borrowing, just set GROWLIM and BORROWLIM equal to the special system parameter PHYSICALPAGES, which is the upper bound on the amount of physical memory that VAX/VMS will configure when the system is booted.)

You might also consider reducing the WSEXTENT size for some processes in the UAF file. If you go so far as to set the WSEXTENT values equal to the WSQUOTA values, you completely disable borrowing for those processes.

5.2.22 Increase Swapper Trimming Memory Reclamation

If you lower the value of SWPOUTPGCNT, you increase the amount of memory reclaimed every time second-level trimming is initiated. However, this is the parameter that most effectively converts a system from a swapping system to a paging one and vice versa. As you lower the value of SWPOUTPGCNT, you run the risk of introducing severe paging.

5.2.23 Reduce Rate Of Inswapping

If you increase the special system parameter SWPRATE, you will reduce the frequency at which outswapped processes are inswapped. SWPRATE is the minimum real time between inswaps of compute-bound processes. For this calculation, any process whose current priority is less than or equal to the system parameter DEFPRI is considered to be compute-bound.

5.2.24 Induce Paging To Reduce Swapping

To induce paging on a system that swaps excessively, you need to lower the working set quotas, as described in Section 5.2.4.2. In addition, you should increase the value of PFRATH and you might also reduce the value of WSINC. With these modifications you will slow down the responsiveness of automatic working set adjustment to paging. The processes will not acquire additional working set space as readily.

It might be worthwhile to check the number of concurrent jobs in the batch queues. Use the DCL command SHOW SYSTEM/BATCH to examine the number and size of the batch jobs. If you observe many concurrent batch jobs, you might decide to issue the DCL commands STOP/QUEUE and START/QUEUE/JOB_LIMIT to impose a restriction on the number.

5.2.25 Add Page Files

In the case where the system disk is saturated by paging, as described in Section 4.2.1, you may want to consider adding one or more page files, on separate disks, to share the activity. This option is more attractive when you have space available on a disk that is currently underutilized. Use the SYSGEN commands CREATE and INSTALL to add page files on other disks. (See the *VAX/VMS System Generation Utility Reference Manual*.)

The discussion of AUTOGEN in the *VAX/VMS System Manager's Reference Manual* includes additional considerations and requirements for modifying the size and location of the page file.

5.2.26 Reduce Demand or Add Memory

At this point, when all the tuning options have been exhausted, there are only two options: reduce the demand for memory by modifying the workload, or add memory to the system.

5.2.26.1 Reduce Demand

Section 1.1.2 describes a number of options you can explore to shift the demand on your system so that it is reduced at peak times.

5.2.26.2 Add Memory

If you conclude you need to add memory, your next concern is to determine how much memory. You should add as much memory as you can afford. If you need to establish the amount more scientifically, you could try this empirical technique.

- Determine or estimate a paging rate you believe would represent a tolerable level of paging on the system. (You should make allowances for global valid faults if many applications share memory, by deducting the global valid fault rate from the total page fault rate.)
- Turn off swapper trimming (set SWPOUTPGCNT to the maximum value found for WSQUOTA).
- Give the processes large enough working set quotas so that you achieve the tolerable level of paging on the system while it is under load.

The amount of memory required by the processes that are outswapped represents an approximation of the amount of memory your system would need to obtain the desired performance under load conditions.

Once you add memory to your system, be sure to invoke AUTOGEN so that new parameter values can be assigned on the basis of the increased physical memory size.

5.3 Compensating for I/O-Limited Behavior

This section describes procedures to remedy specific conditions that you may have detected as the result of the investigation described by Section 4.3.

All the tuning solutions for performance problems based on I/O limitations involve using memory to relieve the I/O subsystem. The two most accessible mechanisms are the ACP caches and VAX RMS buffering.

5.3.1 Remove Blockage Due to ACP

Of the four sources of bottlenecks, the ACP lockout problem is the easiest to detect and solve. Moreover, it responds to software tuning.

The solution for an ACP lockout caused by a slow disk sharing an ACP with one or more fast disks requires that you dismount the slow device with the DCL command DISMOUNT, then issue the DCL command MOUNT /PROCESSOR=UNIQUE, to assign a private ACP to the slow device. Note that you will be using an ACP for disks only if you have ODS-1 disks. However, be aware that each ACP has its own working set and caches. Thus, creating multiple ACPs requires the use of additional memory.

Also, there are situations that may share some of the symptoms of an ACP lockout that will not respond to adding an ACP. For example, when there is substantial I/O activity all directed to the the same device, so that the activity is in effect saturates the device, adding an ACP for another device without taking steps to redirect and/or redistribute some of the I/O activity to the other device yields no improvement.

5.3.1.1 Blockage Due to a Device, Controller, or Bus (ODS–1 Only)

When you are confronted with the situation where users are blocked by a bottleneck on a device, a controller, or a bus, your first step should be to evaluate whether you can take any action that would make less demand on the bottleneck point.

Reduce Demand on the Device That Is the Bottleneck

If the bottleneck is a particular device, you might try any of the following suggestions, as appropriate. The suggestions begin with areas that are of interest from a tuning standpoint, but progress to application design areas.

One of the first things you should determine is whether the problem device is used for paging or swap files and if this activity is contributing to the I/O limitation. If so, you need to consider ways to shift the I/O demand. Possibilities include moving either the swapping or page file (or both, if appropriate) to another disk. However, if the bottleneck device is the system disk, you cannot move the entire page file to another disk; a minimum page file is required on the system disk. See the discussion of AUTOGEN in the *VAX/VMS System Manager's Reference Manual* for additional information and suggestions.

Another way to reduce demand on a disk device is to redistribute the directories over one or more additional disks, if possible. As described earlier in this section, you may decide to allocate memory to multiple ACPs (ODS–1 only) to permit redistributing some of the disk activity to other disks. Section 5.3.2 discusses some of the implications of using VAX RMS to

alleviate the I/O on the device. Also consider that if the disks have been in use for some time, the files may be fragmented. You should run the Backup Utility to eliminate the fragmentation. (See the *VAX/VMS Backup Utility Reference Manual*.) If this approach is highly successful, institute a more regular policy for running backups of the disks.

As a next step, you should try to schedule work that heavily accesses the device over a wider span of time or with a different mix of jobs, so that the demand on the device is substantially reduced at peak times. Moving files to other existing devices to achieve a more even distribution of the demand on all the devices is one possible method. Modifications to the applications might also help distribute demand over several devices. Greater changes may be necessary if the file organization is not optimal for the application. For example, perhaps the application employs a sequential disk file organization, when an indexed sequential organization would be preferable.

Reduce Demand on the Controller That Is the Bottleneck

When a controller or MASSBUS is the bottleneck, examine the activity at the slowest device on the controller and its relationship to the other devices. For example, when tapes and disks operate on the same MASSBUS, the tapes can overburden the MASSBUS and result in poor disk performance. You may find it helpful to group the slower devices together on the same controller or MASSBUS (when you have more than one available).

Reduce Demand on the Bus That Is the Bottleneck

If the controllers are located on a UNIBUS, another suggestion is to place controllers on separate buses. Again, you want to segregate the slower speed units from the faster units. For example, you will find the greatest improvement if you can put tape controllers on one UNIBUS and disk controllers on a separate UNIBUS.

When a UNIBUS becomes the bottleneck, the only solution is to acquire another bus so that some of the load can be redistributed over both buses.

5.3.1.2 Enlarge Hardware Capacity

If there seem to be few appropriate or productive ways to shift the demand away from the bottleneck point using available hardware, you may have to acquire additional hardware. Adding capacity can refer to either supplementing the hardware with another similar piece, or replacing the item with one that is larger, faster, or both.

Try to avoid a few of the more common mistakes. It is easy to conclude that more disks of the same type will permit better load distribution, when the truth is that providing another controller for the disks you already have may bring much better results. Likewise, rather than acquiring more disks of the same type, the real solution may be replacing one or more existing disks with a disk that has a faster transfer rate. Another mistake to avoid is acquiring disks that immediately overburden the controller or bus you place them on.

To make the correct choice, you must know whether your problem is due to limitations in space and placement or to speed limitations. If you need speed improvement, be sure you know whether it is needed at the device or the controller. You must invest the effort to understand the I/O subsystem and the distribution of the I/O workload across it, before you can expect to make the correct choices and configure them optimally. You should try to understand at all times just how close to capacity each part of your I/O subsystem is.

5.3.2 Improve VAX RMS Caching

The *Guide to VAX/VMS File Applications* is your primary reference for information on tuning VAX RMS files and applications. VAX RMS reduces the load on the I/O subsystems through buffering. Both the size of the buffers and the number of buffers are important in this reduction. In trying to determine reasonable values for buffer sizes and buffer counts, you should look for the optimal balance between minimal VAX RMS I/O (using sufficiently large buffers) and minimal memory management I/O. Note that if you define VAX RMS buffers that are too large, you can more than fill the process's entire working set with these buffers, ultimately inducing more process paging.

5.3.3 Adjust File System Caches

The considerations for tuning disk file system caches are similar to those for tuning VAX RMS buffers. Again the issue is the minimizing of I/O. A disk file system maintains caches of various file system data structures such as file headers and directories. These caches are allocated from paged pool when the volume is mounted for ODS-2 volumes (default). (For an ODS-1 ACP, they are part of the ACP working set.) File system operations that only read data from the volume (as opposed to those that write) can be satisfied without performing a disk read if the desired data items are in the file system caches. It is important to seek an appropriate balance point that matches the workload.

To evaluate file system caching activity, issue the MONITOR FILE_SYSTEM_CACHE command and examine the data items displayed. (For detailed descriptions of these items, refer to the *VAX/VMS Monitor Utility Reference Manual*.) You can then invoke SYSGEN and modify, if necessary, appropriate ACP system parameters. Data items in the FILE_SYSTEM_CACHE display correspond to ACP parameters as follows:

FILE_SYSTEM_CACHE Item	ACP/XQP Parameters
Dir FCB	ACP_SYSACC
	ACP_DINDXCACHE
Dir Data	ACP_DIRCACHE
File Hdr	ACP_HDRCACHE
File ID	ACP_FIDCACHE
Extent	ACP_EXTCACHE
	ACP_EXTLIMIT
Quota	ACP_QUOCACHE
Bitmap	ACP_MAPCACHE

When you change the ACP cache parameters, remember to reboot the system to make the changes effective.

5.3.4 Reduce Interrupts for Terminal I/O

If your earlier investigation has led you to consider the use of burst transfers to reduce the interrupt stack time for your terminal I/O, there are several additional considerations you should investigate prior to making a final choice. The discussion in the next section pertains to the DMF32. Similar considerations might apply to other devices.

5.3.4.1 Choose the Appropriate Environment For DMF32s

It turns out that when an application writes 200 or more characters at a time in the mode known as NOFORMAT, the DMA feature of the DMF32 is most beneficial. (Note that DMA transfers are only permitted when the output buffer size is at least as large as the value established by the system parameter TTY_DMASIZE. Also, the standard formatted writes require the terminal driver to perform certain operations, such as wraparound, that preclude the use of DMA transfers.)

When an application writes less than 200 characters at a time but more than 10 characters, the *silo transfer* feature of the DMF32 is still 30 to 60 percent more efficient than a DZ11 or DZ32 transfer. When applications write out less than 10 characters at a time, there is no significant performance improvement of the DMF32 over the DZ11 or DZ32.

To summarize, if you do not observe time spent on the interrupt stack in your investigation, acquiring DMF32s probably will not relieve the terminal I/O problem. Furthermore, DMF32s improve terminal I/O operations only for output, and only if the size of the output buffers is at least 10 characters.

Since most record-oriented terminal writes transfer fewer characters than the value of TTY_DMASIZE, applications must be specially designed to take advantage of the DMA feature offered by the DMF32. As always, you must know your workload well to make an appropriate selection of equipment.

5.3.4.2 Consider Application Design

Among additional methods of reducing interrupt activity for terminal I/O are redesigning the applications and lowering the baud rate.

Programmers can reduce the number of interrupts for terminal I/O if they design applications that collect the QIOs into large operations that write as many characters as possible. In fact, the programmers may want to use QIOs that write as many characters as allowed by the system parameter MAXBUF, which identifies the maximum number of characters permitted for buffered I/O transfers (see the *VAX/VMS System Generation Utility Reference Manual*). If you can identify several offending programs that are used frequently that do not collect the QIOs, you may find the benefits warrant the expenditure of time for redesigning and recoding the programs.

Wherever possible, programmers should design applications for video terminals that update the affected portions of the screen, rather than designing applications that rewrite the whole screen.

5.3.4.3 Lower the Baud Rate if Terminals Smooth Scroll

If many of the applications at your site write characters to terminals in the smooth scrolling mode using the DZ11 or DZ32 interface, you might consider reducing the baud rate to lower the frequency with which the DZ11 interrupts for another character. In this particular environment (smooth scroll writing with DZ11s or DZ32s), lowering the baud rate effectively decreases the CPU time spent processing interrupts. For example, you could operate two terminals at 4800 baud and incur the same number of interrupts that one terminal would generate at 9600 baud, while the decrease in the baud rate would be barely perceptible to the users.

5.3.4.4 Reduce Demand or Increase CPU Capacity

You have detected the need to reduce demand on the CPU or increase the CPU capacity as the result of terminal I/O activity. At this point you have no tuning solutions left, and in fact, if you have followed all previous suggestions that applied, you seem to need capacity more than anything else. Section 5.4.4 describes a few additional ways you might reduce the demand on your CPU and then discusses the topic of increasing CPU capacity.

5.4 Compensating for CPU-Limited Behavior

This section describes procedures to remedy specific conditions that you may have detected as the result of the investigation described by Section 4.4.

There are only two ways to apply software tuning controls to alleviate performance problems related to CPU limitations: specify explicit priorities (for jobs or processes) and modify the system parameter QUANTUM. Your other two options are really not tuning solutions.

5.4.1 Adjust Priorities

When a given process or class of processes receives inadequate CPU service, the surest technique for improving the situation is to raise the priority of the associated processes. To avoid undesirable side-effects that can result when a process's base priority is raised permanently, it is often better to simply change the application code to raise the priority only temporarily. You shoult adopt this practice for critical pieces of work.

Priorities are established for processes through the UAF value. Users with appropriate privileges (ALTPRI, GROUP, or WORLD) can modify their own priority or those of other processes with the DCL command SET PROCESS /PRIORITY. Process priorities can also be set and modified during execution with the system service $SETPRI. See Section 2.2.4.

Priorities are assigned to subprocesses and detached processes with the DCL command RUN/PRIORITY or with the $CREPRC system service at process creation. The appropriately privileged subprocess or detached process can modify its priority while running with the $SETPRI system service.

Batch queues are assigned priorities when they are initialized (INITIALIZE /QUEUE/PRIORITY) or started (START/QUEUE/PRIORITY). While you can adjust the priorities on a batch queue by stopping the queue and restarting it (STOP/QUEUE and START/QUEUE/PRIORITY), the only way to adjust the priority on a process while it runs is through the system service $SETPRI.

5.4.2 Reduce Interrupts

Section 5.3.4 discusses numerous ways to reduce terminal interrupts.

5.4.3 Increase QUANTUM

By reducing QUANTUM, you can reduce the maximum delay a process will ever experience waiting for the CPU. The tradeoff here is that, as QUANTUM is decreased, the rate of time-based context switching will increase, and therefore the percentage of the CPU used to support CPU scheduling will also increase. When this overhead becomes excessive, performance will suffer. In general, do not adjust QUANTUM unless you know exactly what you expect to accomplish, and you are aware of all the ramifications of your decision.

5.4.4 Reduce Demand or Add CPU Capacity

Very few tuning controls that reduce demand on the CPU; you need to explore ways to schedule the workload so that there are fewer compute-bound processes running concurrently. Section 1.1.2 includes a number of suggestions for accomplishing this goal.

You may find it possible to redesign some applications with improved algorithms to perform the same work with less processing. When the programs selected for redesign are those that run frequently, the reduction in CPU demand can be significant.

You also want to control the concurrent demand for terminal I/O. Section 5.3.4 addresses the effects of terminal I/O on the CPU and good terminal I/O management techniques.

If you find that none of the previous suggestions or workload management techniques satisfactorily resolve the CPU limitation, you need to add capacity. It is most important to determine which type of CPU capacity you need, since there are two different types that apply to very different needs.

Workloads that consist of independent jobs and data structures lend themselves to operation on multiple CPUs. If your workload has such characteristics, you could add a processor to gain CPU capacity. The processor you choose might be of the same speed or faster, but it could also be slower. It takes over some portion of the work of the first processor. (Separating the parts of the workload in optimal fashion is not necessarily a trivial task.)

Other workloads must run in a single-stream environment since many pieces of work depend heavily on the completion of some previous piece of work. These workloads demand that CPU capacity be increased by increasing the CPU speed with a faster model of processor. Typically the faster processor performs the work of the old processor, which is replaced rather than supplemented.

To make the correct choice, you must analyze the interrelationships of the jobs and the data structures.

Index

A

B

C

D

E

F

H

I

K

L

M

O

P

Q

R

S

T

V

W

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line