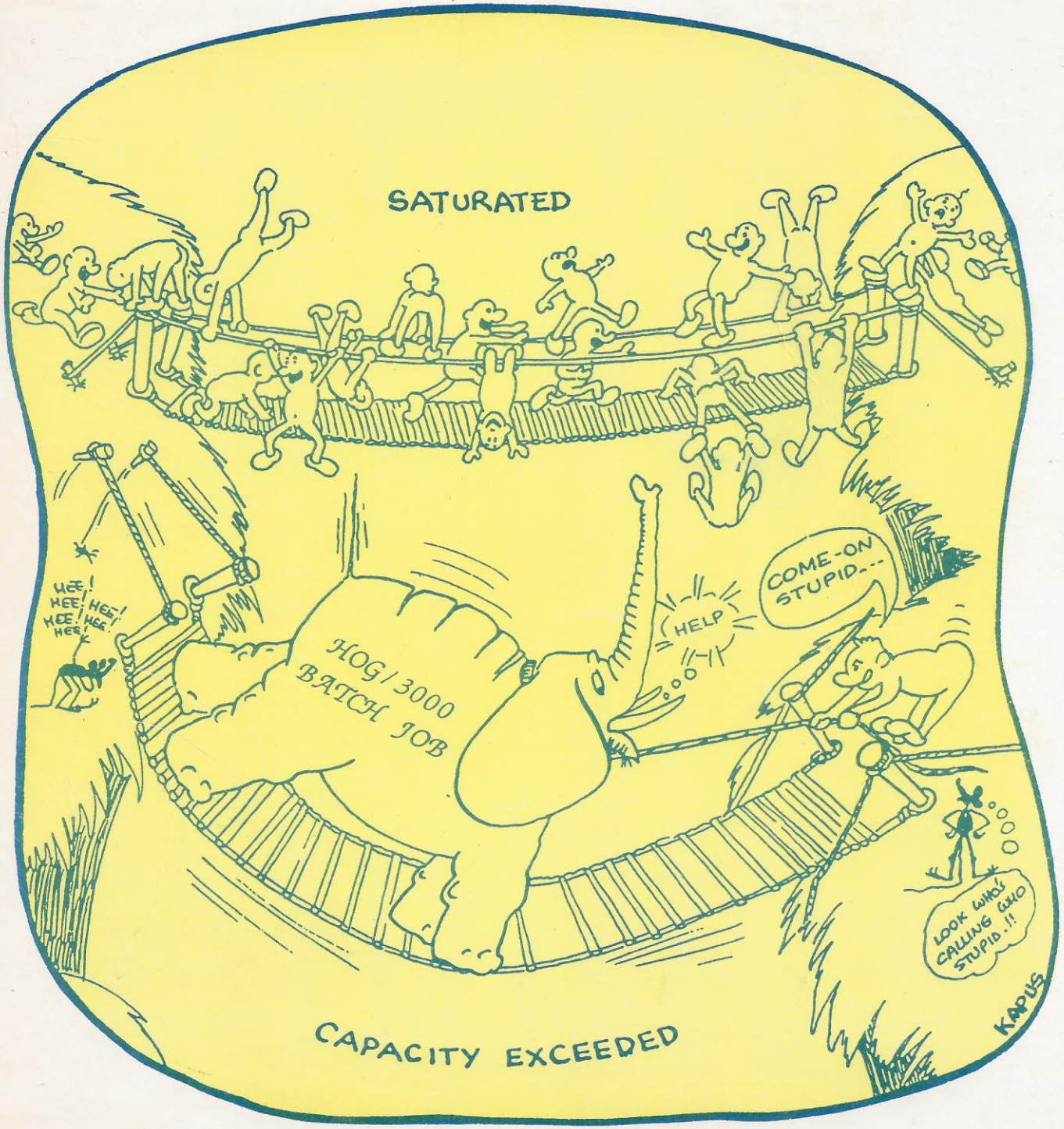
TAMING THE HP 3000

over 101 ways to monitor, manage, and maximize system performance on the Hewlett-Packard HP 3000

Volume I



Robert A. Lund

TAMING THE HP 3000

Volume I

TAMING THE HP 3000

Over 101 Ways to Monitor, Manage, and
Maximize System Performance on the
Hewlett-Packard HP 3000

Volume I

Robert A. Lund

PERFORMANCE PRESS PUBLISHING
ALBANY, OREGON

TAMING THE HP 3000 - Over 101 Ways to Monitor, Manage, and Maximize System Performance on the Hewlett-Packard HP 3000 - Volume I.

Manufactured in the United States of America

First edition 1987

Manuscript preparation by Performance Press, Albany, Oregon. Printing by Whirlwind Press, Albany, Oregon.

Acknowledgment is made to the following for permission to reprint copyrighted mateal: from American Federation of Information Processing Societies, the cover cartoon, "Copyright 1978 by American Federation of Information Processing Societies, Inc., used by permission."

ISBN 0-945325-00-2 (cloth)
0-945325-01-0 (paper)

Library of Congress card catalog number: 87-092025

Second Printing 3/88

For ordering information, call or write the publisher at:

Performance Press
P.O. Box 151
Albany, Oregon 97321
(503) 327-3800

To My Lord Jesus Christ

Who

Probably Doesn't Care Much

About System Performance

But Who Has

Relentlessly Demonstrated

His Mercy, Provision, and Love

For Me,

This Little Volume is

Gratefully Dedicated

Contents

Foreword

Robert Lund's book on tuning the HP 3000 for optimal performance is one of the very few books that is both helpful and readable at the same time. He explains many very complex topics in a manner that even those of us without a PhD in mathematics can readily understand. Even more important, the recommendations he suggests can be implemented by a system manager who has three weeks or three years of experience on the HP 3000.

Encapsulated in this book are numerous concepts and thoughts that will provide you, the system manager, with an unbelievable number of methods to improve your system's overall performance. A unique feature of this book is that it assumes you do not want to or cannot make changes to your programs. Therefore, this text is valuable to the majority of us that have purchased third party software.

Although some of the methods for performance optimization may have been previously presented elsewhere, this is the first book that ties all of the relevant concepts together. Furthermore, these concepts are presented in a comprehensible manner.

After reading even a few pages of this book, Robert's many years of experience with the HP 3000 will become quite apparent. What adds to the enjoyment of studying this book is his style. Having worked with Robert, it is very apparent that the sense of humor which permeates his personality has become an integral part of this book. System performance optimization has traditionally had a tendancy to be presented in a boring, abstract (and thus un-implementable) manner. You will not discover any complex statistical models in this book. Instead, you will find some very practical suggestions for tuning your HP 3000 to perform exceptionally well.

Maury Nichols
Performance Specialist
Hewlett-Packard

ACKNOWLEDGMENTS

Thucydides, writing of the Peloponnesian war, is reputed to have said, "If anybody shall pronounce what I have written to be useful, then I shall be satisfied." I could hope for no more concerning my own work. But if anyone does deem *Taming The HP 3000* useful, I cannot take all the credit. No man stands alone in his endeavors. He must rely on those who have gone before him and those who have stood with him. My efforts are no different.

Anyone who writes knows that an author's family always bears the greatest burden of sacrifice in the publication of a book. My wife, Colleen, bore patiently my desire to pen this little volume. During the final months of writing, she took over nearly all my responsibilities at home and literally was mother and father to our two daughters Jessica and Jamie. I thank her.

Much appreciation goes to Tony Phillips, Dave Jones, and Doug Strackbein whom were my managers (boot camp sargeants?) at Hewlett-Packard in Fullerton, California. They lived the HP Way and taught it to me. These guys are among the best. Thanks to all the SEs who invested in me (especially the Rambo District!). Thanks also goes to Dan Yates and the gang at Summit for all the support.

Technical accuracy in this hotly debated field is difficult to attain. Thanks to Jim Squires(HP), Maury Nichols (HP - especially brutal!), Larry Kemp (HP), Jim Knoop (HP), Vladimir and Eugene Volokh (VESOFT), and David Merit (BRADMARK) for reviewing the technical contents. Their comments and encouragement were vital.

Thanks also to all who helped with editing and proofing: Bill Lancaster, Margie Worden, Mike Knudson (final galley proofing and paste up), Stephanie Gallagher, Paul Blondin, and John Rist.

The "dungeon" in Tom and Nancy Birchard's house was invaluable for preparing much of the manuscript. Without the encouragement and prodding of my friend Tom it is questionable whether this book would have come into being. Thanks Tom.

Last but not least, thanks to my father and mother who always encouraged me to think, learn, and explore. Most of all they have ALWAYS loved me and believed in me.

Introduction

This book was written as part of an on-going quest to assist Hewlett-Packard HP 3000 users, many of whom the author supports, in the struggle to maintain an acceptable level of system performance.

The primary goal of this book is to help the reader improve system performance as well as think of new ways to squeeze out a few more transactions in a cost effective way. The implementation of just one idea could save the reader many times the cost of the book.

Some of the hints for performance improvement in this book are quite technical and thus will not be useful to you if you are not versed in HP 3000 internals. Others border on being very obvious. An attempt was made to include every conceivable idea for monitoring, managing, and especially improving system performance. It is the author's intent that this book provide the reader with many different ways to manage system performance on the HP 3000. It is also the author's wish that the ideas presented serve as a catalyst for more system performance research. You are encouraged to write Performance Press (P.O. Box 151, Albany, Oregon 97321) and submit your critique of this book as well as your performance war stories and improvement ideas.

HOW TO USE THIS BOOK

It is assumed that you have had at least a reasonable amount of experience on the HP 3000 as a programmer, system manager, MIS director, technical support person or some combination thereof. This book is broken up into sections which pertain to common performance bottleneck areas. It may be used as a reference guide, or even as a primer. See Appendix A for a crash course in performance basics.

For the most benefit, it is recommended that you simply scan quickly through the entire book noting the layout and perhaps briefly reading some of the ideas relevant to your area of expertise. If you are a novice in this area, taking time to study the primer (Appendix A) will provide some foundation for the rest of the book. Then you may wish to focus on a section which pertains to perhaps a current performance bottleneck that you are experiencing.

This book is intended to be a compendium of ideas, a sort of performance improvement reference guide, and not a text book on system performance. Though sufficient information is presented to help you understand each idea, you will probably obtain maximum benefit by having some basics mastered first. You could obtain these basics in one of two ways: experience or mastering the primer in Appendix A.

The subject at hand can be quite subjective. It seems that those who are skilled in this arena rely on intuition at times to help them arrive at a proper problem diagnosis/resolution. It is also understood that this strange science breeds hot debate between specialists. Good folks do differ in many of the issues involved.

You will note that although this author has consulted with numerous specialists inside and outside of HP, views expressed belong to the author. Debate will no doubt arise concerning the possible benefits of implementing the ideas for improvement. Please keep in mind that each of these ideas will provide some benefit under certain circumstances. Though perhaps providing a benefit in isolation, a single idea combined with any number of other ideas may prove to detract from system performance.

The areas of expected savings could be in: shortened batch run times, increased user productivity due to decreased response times, deferred hardware purchases, and the "hidden" benefit of increased user and management satisfaction. It must also be said that some of the information within these pages is time-dated. That is, each new version of MPE may bring a modification in the rules by which MPE plays its games. Some of the ideas set forth might be invalid two MPE versions from now.

Most successes are achieved by diligently adhering to the fundamentals within a given discipline. The same is true for optimizing system performance. Though most ideas presented here might not cause dramatic improvements (though some have), they will, in most cases, help somewhat. The net effect of the cumulative total, however, may produce substantial performance improvements. Developing good habits by implementing the basic ideas presented herein will provide you with a more efficient computer at worst and possibly revolutionize transaction throughput at best.

A WORD ABOUT FORMAT

Each idea is tagged with a number and a prefix corresponding to the chapter it pertains to. For example, #OM24 refers to Idea #24 in the Operational and Management Considerations chapter. This scheme was adopted to hopefully provide easy idea referencing.

FOR THE FANATIC

A true performance fanatic would try to implement every one of the ideas listed. Attempting to do so would probably never make sense in the typical production environment. However, it would be a good idea for those involved in the design and development stages to at least be aware of the factors which contribute to optimal system performance. These people should prioritize relevant ideas for

implementation early in the application design stage. It is therefore not the author's intent that the average reader will have the need, much less the resources, to implement all of the ideas presented. For the one who has an obsession with obtaining the utmost possible system performance, this little volume may be a feast.

ANY FREE LUNCHES?

Performance management is in reality a tradeoff of resources (time, money, personnel, etc.). As you may have guessed, there is no such thing as a free lunch. This was concisely put by one HP 3000 user,

> "Performance enhancement consists of trading one resource for another to achieve a desired balance for a particular circumstance."[1]

Realize therefore, that many of these ideas will not be practical for implementation. For example, if programming is not performed in the reader's shop, ideas which may be of benefit at the programming stage simply will be irrelevant.

It is suggested that a purchased copy of this book, or at least some of the performance optimization techniques described herein be passed on to those who engineered your software. Having been a programmer for some time has made this writer aware of a simple fact: When "under the gun," optimal resource utilization is probably the last thing on a programmer's mind. It also seems that deadlines and commitments rarely allow the luxury of modifying code after proper functionality has been obtained. It is therefore suggested that time be allotted for application performance tuning during the design cycle.

Even if some program development occurs in-house, modifying working programs for the sake of improved performance can be quite resource consumptive. An exception to this would be program resegmentation which may prove to be cost effective; being easy to do, it does not take much time and can help performance substantially in some cases. This author therefore favors "non-intrusive" performance enhancements when at all possible. It may be more cost effective to mitigate the effects of minor performance problems than to attempt a major application overhaul.

Purchasing additional hardware may be a better solution. The only problem with having a mentality of adding more hardware to solve a problem is the subtle effect it has on application designers and programmers. Since the price/performance curves of hardware have been falling in recent years, there seems to have been an associated rise in "HOG/3000" programming techniques. There is a balance between adding more hardware and spending more human resource to provide system efficiency. That balance is best determined by each individual HP 3000 shop.

---

1 James S. Overman, *Improving Application Performance Without Changing Your Programs,* (An unpublished paper given to the author, 1986), pg. 1

THE TRUE COST OF OPTIMUM PERFORMANCE

HP 3000 computer systems are so numerous, in part, due to the superb price/performance ratio they provide. This writer has observed that most new users on the system do not realize how much continual management it takes to insure optimal, consistent system performance. For example, over time TurboIMAGE/3000 detail datasets become quite inefficient, and frequently need reorganization. The typical shop is faced with either spending personnel time for database unloads or purchasing a tool that will perform the operation with greater system availability. This is merely one of many areas which are sometimes found out by surprise at a date long after the system has been in production.

Monitoring performance can also be costly; good tools and training as well as time for monitoring contribute to this expense. As the team at Robelle Consulting said in a recent article that addressed how to squeeze more performance out of current systems,

> "What we found was not some "secret" formula, but rather a mundane, continuous attention to the details of system performance."[2]

A WARNING AND A DISCLAIMER

Computing machines and living machines (organisms) have a few things in common. They are both complex, and they are both sensitive to their living (operating) environment. Stresses affect both with fairly predictable results. Properly diagnosing and correcting functional disorders requires a person to be trained in the theory associated with each **and** to gain experience in real life situations.

It is illegal for an unlicensed individual to diagnose and/or prescribe remedies for human ailments. Similarly, though not illegal, it is potentially as dangerous for an untrained person to tackle the intricacies of correcting computing machinery dysfunction. Some of the rectifying methods listed herein are potentially harmful to system performance if not understood and implemented correctly. It is for this reason that the author and the publisher must issue a disclaimer: You alone are liable for any problems that might arise from implementing any of the performance ideas in this book. By all means, contact a performance specialist if you have questions regarding your specific performance problems.

SEEKING OUTSIDE HELP

The author is an advocate of experimenting with various performance improvement ideas but must pose the following concern. This book is not intended to be a substitute for performance consulting. Unless you are experienced in this field with a few years of dedicated study in system performance, YOU WILL PROBABLY NEED HELP AT SOME POINT.

---

2 Shumko, Green, Greer, *What if...You Didn't Wait For Spectrum?,* (HP Professional, July, 1987), p. 82.

MAKE THE CONSULTANT WORK FOR YOU

Don't hesitate to contact performance support individuals, as an hour of their time may save you days of fumbling and perhaps embarrassment. Though fumbling time is often very educational, most of us do not have the luxury of having an HP 3000 to ourselves for the purpose of experimentation; we have jobs to perform.

When you do work with a performance consultant, make the most of the time. Don't be content with a stack of nebulous charts and graphs; make the person explain to you, in plain English, what the issues at hand are. Although some might not be agreeable to this, you might even have that person agree to letting you call for free consultation for a short period after the analysis is done, so that all of your questions are satisfied.

By combining your own study (bookwork and empirical observation) with tutorial sessions done by a specialist in conjunction with a performance analysis, you will better equip yourself to deal pro-actively with performance optimization on your HP 3000 in the future.

Another source of great help is third party software. By purchasing software that helps you manage your system, you usually obtain support from the supplier. For your convenience, partial vendor references are included when a particular software tool is mentioned. Please note the full address and phone number for each one in the Vendor Reference section.

THE BOTTOM LINE

It is the author's desire that you carefully consider the possible application of each idea presented within your environment. Some of the concepts in this book are perhaps obvious or even border on being absurd, but the majority of them have been tested in real life and have provided performance improvement under varying conditions and circumstances.

Chapter 1

Load Balancing Techniques

Developing a strategy to manage the various loads placed on the system may well prove to be the single best way to maximize system performance. Insuring an even usage of primary system resources is the objective in load management. This chapter provides a number of ways to help reach this goal.

Most systems which the author has been acquainted with have not usually been overloaded 100% of the time (unless there was a drastic under-configuration of hardware). System usage has generally fallen into "peak" and "nonpeak" periods where resource utilization was at a summit or a valley. When the CPU, for example, is being utilized at above 75% of its capacity in a purely busy state (not waiting on events), a "peak" time is likely to be experienced by system users. Figure 1.1 illustrates this concept of load variation throughout the day. Raising the periods of low utilization may be accomplished by simply shifting jobs and user input activity from the busy periods.

Sustaining heavy peak loads for any length of time may eventually have negative implications on transaction throughput. As Stan Freeman stated in a recent article on capacity planning,

> "...the rule most applicable to the HP 3000 states that average device utilization should not exceed 70-75 percent. Response time will rapidly degrade otherwise."[1]

It is important to keep in mind that Stan was referring to interactive workloads with the 75 percent utilization since even one batch job will usually send CPU utilization to 100 percent.

By spreading out loads placed on the system we are able to achieve more nearly what we would like to attain: To be able to run as much as possible on the system without producing resource saturation at any one point in time. That's the ideal. In reality, the ideal is often difficult to attain due to the dynamic nature and variety of loads placed on the system. But having observed characteristics of certain programs and habits of the user population with your unique system hardware/software configuration, you will be able to intelligently make decisions commensurate with this ideal.

1 Stan Freeman, *Capacity Planning on the HP 3000,* (INTERACT, June, 1986), p. 56

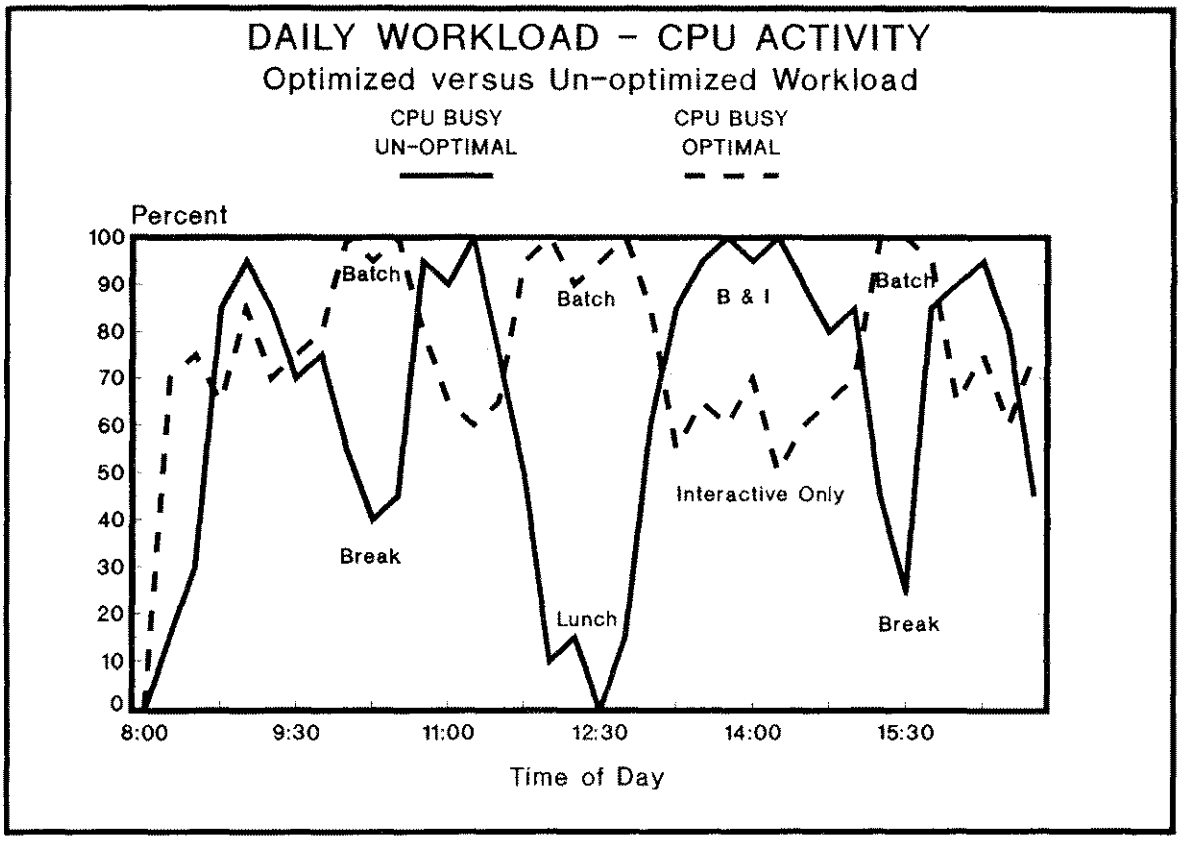


Figure 1.1 - Daily Workload Smoothing

Begin to implement a habit of consistent resource monitoring early in your system performance management game plan. Choose a tool that will produce periodic report(s) of resource utilization; see Chapter Six for suggestions. Graphical representation is generally more helpful in displaying trends in resource usage. It is very sobering to see a graph of CPU utilization for one day broken down into segments. The seasons of spikes will coincide with response time complaints, while the valleys will reveal how much resource was not being utilized at those particular times.

Interactive users have really only one objective definition of response time; that is consistency. All other definitions are subjective. Controlling the workload makes the response times consistent. Providing you with ways to smooth the spikes and raise the valleys due to load imbalances is the goal of this chapter. Controlling the workload is the best way to insure user satisfaction due to consistent response times.

#LB1 Schedule the Quantity of Jobs and Sessions Intelligently.

The number and type of processes running on the system defines the load as felt by the CPU. Without regard to even what the jobs are doing, the number of jobs

and sessions running concurrently can have a great impact on the total system performance. Figure 1.2 illustrates this point.

But how does one determine an optimal mix of sessions and jobs? Unfortunately there are only a few suggestions, and really only one way to know for sure. As for sessions, talk to your technical support person, software supplier, or other shops that are using similar software. This will provide you with an approximation. I stress the word approximation.

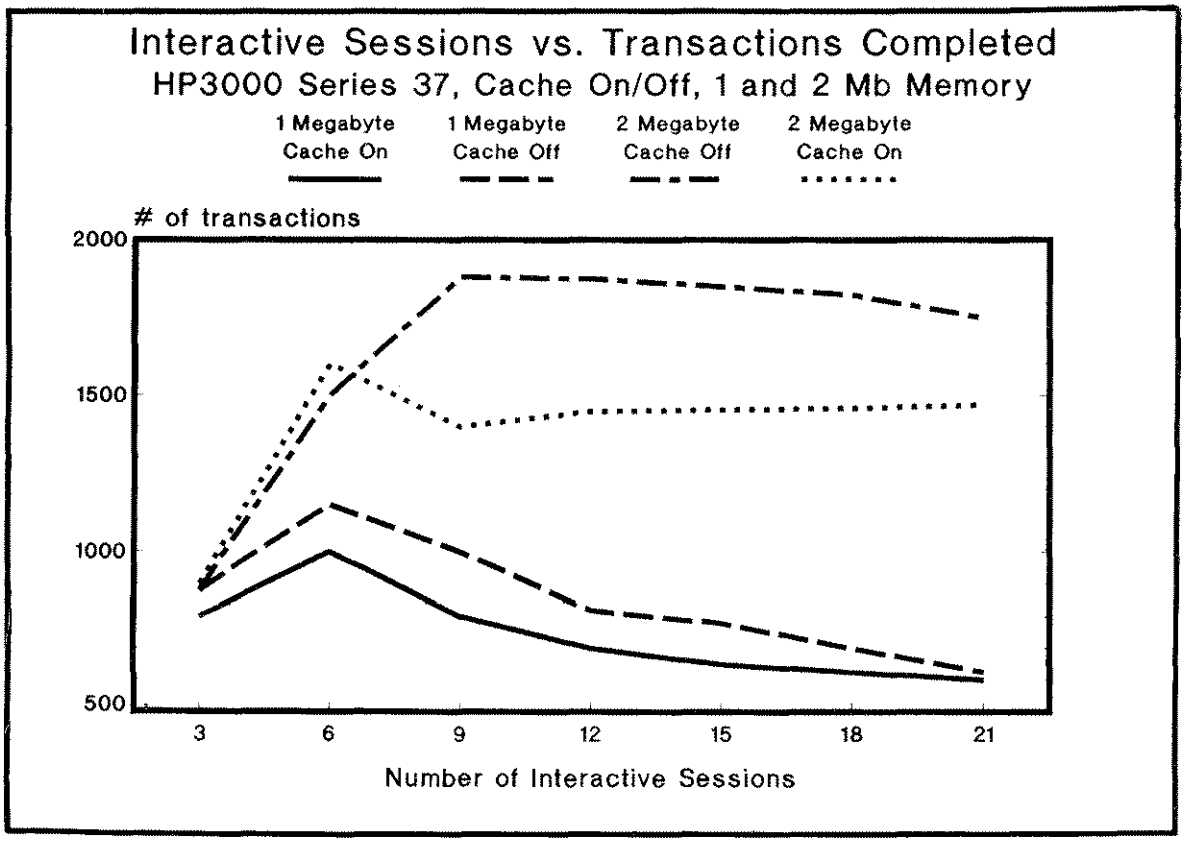


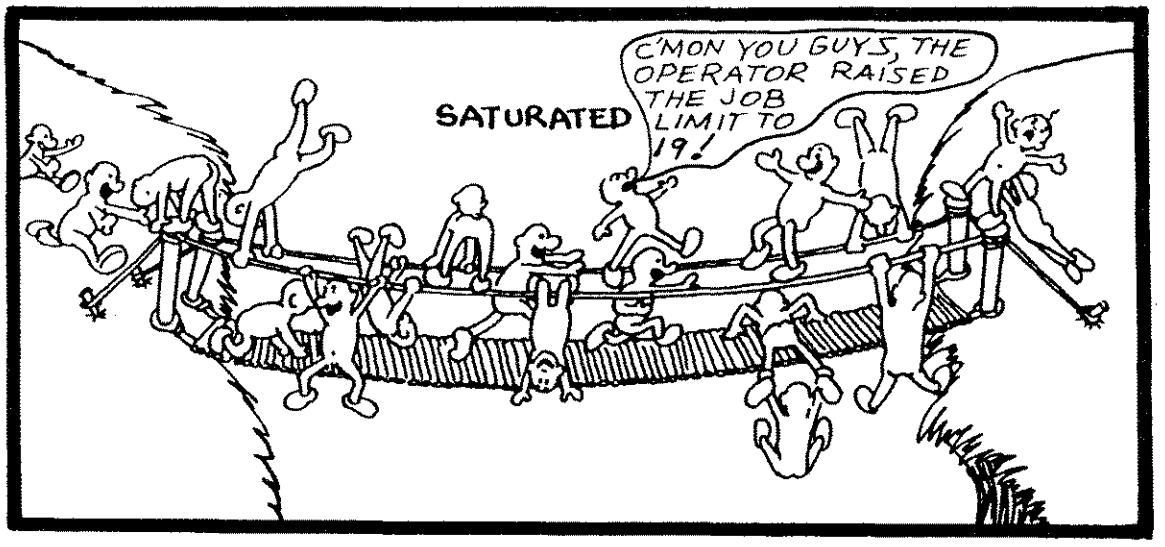
Figure 1.2 - Increasing Load vs. Transactions

Increasing the number of sessions and/or jobs does a couple of things. We begin to experience the law of diminishing returns. First of all, MPE overhead increases rapidly; then total transaction throughput begins to fall off. Second, system resources (CPU, memory, and disc) approach saturation, and online response time is negatively impacted. Realize however, that unless the same TurboIMAGE database is being accessed by both the jobs and the sessions, adding more jobs will have a negligible impact on the response time of the terminals (under the MPE V dispatcher).

Controlling the number of jobs and sessions will dramatically affect system performance. One way to help control the amount of load on the system is to allow use of the LIMIT command only to qualified personnel. Limiting its use will keep

folks from slipping that one "short" job under the fence. Restricting access to the console, and not allowing global use of the LIMIT command will help.

The only way to know an optimal number is simply to take your system's "pulse" under varying quantities of user sessions. Do this by using one of the monitoring tools listed in Chapter Six. But more than this, utilize the real test of system performance: How do the users perceive it? What are typical response times? Unfortunately, there are not any hard and fast rules by which to base the number of jobs/sessions which are mixed on your system. You may decide, as many shops have, not to run any batch jobs during daytime hours in order to favor session response time.



Here is a list of CPU types and a suggested beginning point for the number of jobs to run with online users. Keep in mind that the type and duration of a job could preempt this recommendation. That is to say, if you have a job that is doing FREE5s, expect skyrocketing response times. What this chart is really trying to point out is that it would not be advisable, as an exaggerated example, to run six jobs with interactive sessions on a Series 37.

As previously stated, unless jobs access similar databases as sessions, interactive users will not be negatively impacted much at all. Adding more batch jobs will merely increase resource contention among those jobs. In fact, as the number of batch jobs increase, resource contention will escalate to such an extent that at some point, the law of diminishing returns will begin to take effect. In many environments, batch jobs should not be run at all during the day. The following chart provides conservative maximums. Depending on the type of jobs, these may not be appropriate for your system at all:

Series CPU	Max jobs
III, 30, 33, 37	1
39, 4X, 5X, Micro	2
6X, 70	3

#LB2 Schedule the Type of Batch Jobs Intelligently

Not only does the quantity of process loads affect performance, but the nature of the loads does also. It would not be advisable to run, say, a batch job that updates the same database as users are updating or reading from at the same time. This is due to the increased competition for the IMAGE database that the processes would experience. Also, some of the policies of the MPE scheduler will allow some low priority processes to be raised in priority if they are currently holding a resource needed by a higher priority one. This would temporarily raise the batch job's priority, and perhaps negatively affect the overall performance of the system. Sessions and jobs which compete for the same database are a good example of this.

It is also helpful to know which resources programs need most. Running a number of programs concurrently which perform numerous CPU-intensive tasks will attempt to saturate the CPU resource (to the exclusion of say memory and disc I/O). Classifying programs by the type of resource needed most by them will allow you to intelligently schedule a mix which utilizes the three major resources on the system as evenly as possible.

Remember, each resource has a deliverable number of that resource's packets. By packets we mean a certain quantity of a particular device's commodity (CPU cycles, I/Os, etc). For example, the disc I/O subsystem may only be able to deliver a maximum of 40 I/Os per second. Thus we speak of packets of I/O resource. By consuming a large quantity of one resource's packets per a given time, a premature slowdown is experienced. But by mixing jobs and sessions which utilize disc, CPU, and memory resource in somewhat of a balanced fashion, system slowdown is forestalled.

It should be said that achieving a "balance" of resource utilization is easy to talk about, but often times difficult to realize in actuality. One way to begin would be to implement a performance monitoring gameplan, and to simply observe resource utilization traits of disc, CPU, and memory during particular load mixes. You may decide that running your dividend calculation along with accounts receivable aging is simply consuming too much CPU. Perhaps producing invoices along with online data entry to that same database is producing too much disc I/O and IMAGE contention for acceptable response time. Taking time to talk with software suppliers, and experimenting with various load mixes will help improve the overall throughput on the system.

#LB3 Defer All Batch Jobs to After Hours

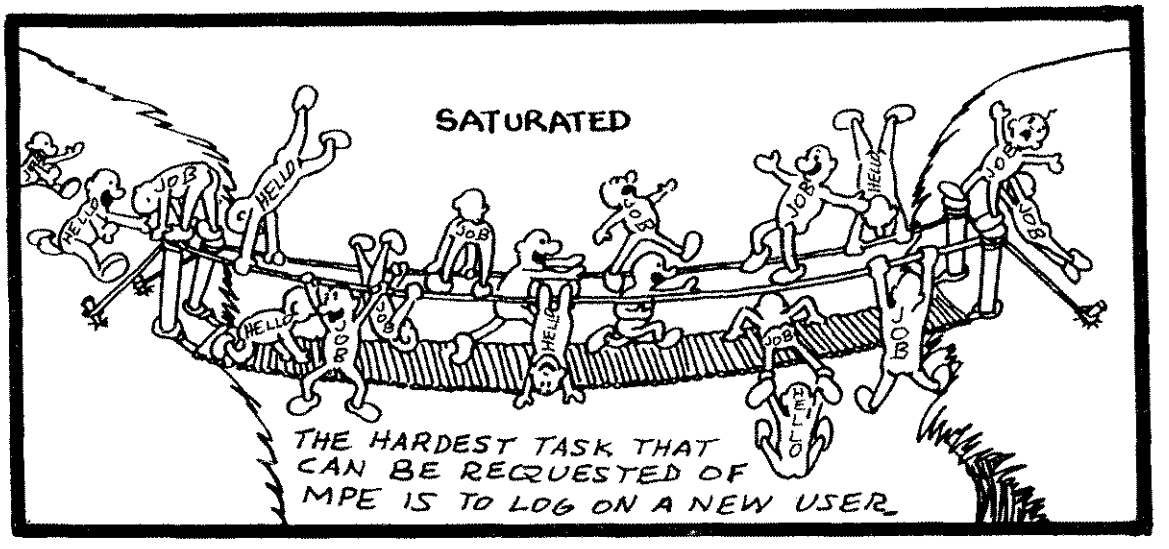
As a rule, session response times will be minimized by not running batch jobs concurrently. This is especially true if the jobs and sessions compete for similar resources. Though it is certainly possible to do both on your system, many HP 3000 sites prefer to defer batch processes to nonpeak periods. This is one way to obtain an immediate boost in overall system performance in many cases. In environments with heavy interactive production, user breaks and lunch times as well as after hours are good choices to perform some background processing (see Figure 1.1).

#LB4 Avoid/Reduce Session Logons and Logoffs

Brian Duncombe puts into words what most users know by experience when he says:

> "The hardest task that can be requested of MPE is to logon a new user."[2]

This idea is fairly obvious. However, many system managers do not realize how much overhead is incurred when just one user logs on. One way to minimize the impact of logons is to implement a process handling scheme. A master job could perform CREATE PROCESSes for terminal users in the morning thereby restricting user access to the HELLO command. Utilizing the STARTSESSION command is one other alternative. Anything you are able to do on your system that reduces or eliminates the need for session logons will help improve system performance.



Sometimes a poorly designed accounting structure and application mix is the cause for excessive logons. If users have to log on to a different account to run a particular application frequently, thought should be given how to modify the accounting structure security to perhaps rectify this situation. The author has observed more than a few occasions in which users were forced to log on to a separate account to simply run a program, and then log back on to the previous place. Developing a plan wherein users log on once at the beginning of a day (or better yet, are logged on by a master job or operator) will lighten the burden associated with session initiation.

There is a tremendous amount of work that MPE must perform in order to provide the user with the MPE colon prompt. The bulk of the overhead is due to disc I/O activity (wouldn't you have known it!). A partial list of this activity is I/O for UDC initialization, directory security checking, system table entry initialization, allocating virtual memory space for the user's stack, etc.

---

2 Brian Duncombe, *Performance Self-Analysis,* (Proceedings of the Interex HP 3000 IUG Amsterdam Conference, 1985), p. 819

There has been some discussion in the recent past that suggests logging off inactive terminals after a reasonable period of time. This is well and good if security is more of a concern than performance. An inactive logged-on session is a security risk. It must be understood, however, that a session logoff implies another logon at some point in the future (perhaps soon), and more overhead is incurred to reuse that terminal. It would be far better from a system load standpoint to leave sessions logged on rather than to keep incurring this type of overhead.

#LB5 Stagger Programmer/User Work Hours

Since the quantity and type of processes on the system defines the corresponding load that MPE experiences, one way to even this load out is to simply provide different shifts for data entry and programming. Try using early morning hours for personal productivity and good system performance; many programmers do come in early or leave late in order to obtain maximum resource availability for programming. Simply reserving computer time for data entry in the afternoon, and having development personnel commence their working day an hour or two earlier may provide the system with enough breathing room to render good response time for users during the afternoon peak periods.

#LB6 Move Online Programs Which Produce Paper to Batch

Interactive applications which produce hard copy output to a local printer (or even the system printer) may be best moved to a batch job. This will help improve system performance since interactive sessions are generally operating in the CS sub-queue and therefore are favored with ample CPU attention over batch jobs running in the DS or ES sub-queues.

By executing lengthy report generation via a session, one is not only tying up the terminal (and the user), but actually may be infringing on system resources in an unfair manner. The CS Sub-queue is generally intended to provide prompt system response time for users at terminals which are performing relatively short transactions. Lower sub-queues are intended for longer batch transactions. By performing longer transactions in the CS queue, other interactive users may be suffering.

Even though CPU intensive users are lowered in priority, in reality there are occasions where MPE's scheduling policies did not seem adequate to handle certain combinations of interactive loads with the default queue settings. See ideas #OM10 and #OM25 for more information on this. Running paper producing programs in a batch queue is one way to free up terminals and to insure proper adherence to MPE's scheduling scheme without modifying the queuing structure.

#LB7 Use a Batch Job Scheduling System to Schedule Jobs

If load balancing is beneficial in smoothing out resource utilization peaks and valleys (see #LB1 and #LB2 above), then intelligent batch job scheduling software might be an easily justifiable investment. Software packages exist which not only help in scheduling the logical run order for proper application execution, but which

also have a side benefit of allowing the operator to insure a proper quantity and mix of programs. For example, a scheme might be implemented which will not allow more than one CPU intensive job to be run at one time. This would be accomplished by utilizing the batch scheduler's scheme of forbidding certain jobs from running concurrently.

Listed below are some of the batch scheduling packages available at the time of this book's printing. You are encouraged to obtain demonstration packages from the vendors listed as they are often inexpensive, if not free. They are: JMS/3000 (Design/3000), MAESTRO (UNISON), and OCS/EXPRESS (Operation Control Systems).

#LB8 Use the Simple Scheduling Facility of the MPE STREAM Command to Schedule Jobs

The STREAM command as of T-mit release of MPE provides the ability to schedule batch jobs to be launched at a certain time or in a given period of time from now. This is another way to spread out job runs with minimal operator intervention. However, one disadvantage of scheduling jobs in this fashion is if a system interrupt were to occur, these scheduled jobs are not preserved.

#LB9 Avoid/Reduce Running Programs

Just as session logons produce excessive overhead for MPE, running programs does also. This is not referring to the actual execution of a program's code (though it is a load). What is being referenced are the actual steps which MPE must go through in order to just get the program ready to execute its instructions. All processes to be run must go through the system loader; this is a serial process - this means only one program at a time.

Quite a bit of I/O takes place during this load phase, as well as FOPENs, FCLOSEs, externals resolved, and many other actions on the part of MPE. Matters are not helped much by the fact that program files are unblocked, that is, a physical I/O must take place for each logical program record read from disc. Avoiding program executions will help performance; see Ideas #OM1 and #OM2 for more about this.

#LB10 Move Program Development Onto a Separate CPU

Program development, by its very nature, is very dynamic. That is, the type and consistency of load presented to the system is generally erratic. Data entry or reporting can be a bit more predictable in terms of load. Online compiles, QUERY/3000 usage, and source editing may tend to hurt response times and job completion times of the daily production. If the goal is to maximize resource availability for online users, procuring a second system for the sole purpose of program development may be a wise decision.

Used HP 3000s are becoming more plentiful with the advent of newer system announcements. Moving development staff onto another system will insulate the

production machine from the often drastic and erratic CPU and I/O needs inherent to program development. Performing online program compiles ought to be outlawed on production machines. See Idea #OM25 for a way to help prevent online compiles.

#LB11 Move Graphics and Word Processing Applications to a Separate CPU or PCs

Hewlett-Packard has been hinting for some time now that this is their recommended strategy. These two applications tend to be extremely CPU-intensive in nature. By having a dedicated computer for the specific purpose of one of these, we are able to help maximize the efficiency of our data processing environment. Note particularly the overhead incurred by HPWORD, HPDRAW, and DSG/3000.

Due to the large amounts of CPU needed especially by graphics products' for complex raster-to-vector conversions and calculations, and the excessive amounts of I/O, these products quickly consume a great percentage of the system's resources. To make things worse, they are designed to be run online. In the interactive sub-queue (CS queue) they especially exclude batch jobs from obtaining adequate CPU. With PCs being as common and inexpensive as they are, they are good candidates to provide the substantial CPU needs inherent to graphics and word processing applications. The HP Graphics Gallery Collection is one example of quality, PC based graphics.

Isolating as few as possible applications per CPU is good practice. This allows the operating system to spend a minimal amount of time shuffling between the often times radically varying needs of different application programs. Since program code is shared on the HP 3000, having a few programs run by many users will tend to use less system resources than an environment which has many programs with just a few users per program.

Another reason for distributing processing needs over multiple CPUs is one that might come as a mild shock to some folks. A very busy series 70, for example, may not be able to provide the same response time as a stand-alone PC. Given the rise of personal computers distributed in data processing environments, there are probably a lot of spare "MIPS" available much of the time. There is an HP 3000 shop in Europe that downloads data to be crunched by numerous PCs during the night, and then uploads the results back to the HP 3000 host. Most stand-alone PCs are capable of doing a tremendous amount of CPU-intensive work.

#LB12 Avoid Simultaneous Runs of Programs which Access the same Data Files

As is discussed elsewhere in this chapter, knowing your applications is indispensable to insuring optimal system performance. This is especially true in the area of disc I/O management. By running programs that compete for similar disc data, there is risk of impeding I/O activity through file lockouts, SIR competition, and especially I/O bottlenecking at the disc device or database level. Though it cannot be avoided at times, running batch programs which compete for the same data files (IMAGE, KSAM, or MPE files) as sessions may severely impede overall

system performance. Knowing which data is primarily accessed by which programs will allow you to further insure optimal performance by balancing I/O loading. It would not be too difficult to produce a chart that shows which files are primarily accessed by which programs. A sample chart might look like:

```
PROGRAM NAME        PRIMARILY ACCESSED FILES
-------------------------------------------------
AR002               ARBAS.DATA.PROD, CONTROL.DATA.PROD
                    MASTR.UTIL.PROD
GLCALC              GLBAS.DATA.PROD
STMTGEN1            HISTRY.AGRP.PROD, ARBAS.DATA.PROD
```

Such a list would be valuable in creating an overall system performance game plan. In this example, running AR002 at the same time as STMTGEN1 may produce some undesirable I/O contention due to simultaneous access to ARBAS.

Running various jobs together when online users are active and again when not active, and then observing the resultant I/O rates and disc queue lengths may provide you with data to help you avoid or favor certain job/session mixes.

#LM13 Force PIG/3000 Programs into Lower Sub-queues by Writing Programs which Intercept Associated RUN Commands.

At times it is difficult to limit the use of CPU-intensive programs from being run interactively. These CPU consumers tend to make other session users suffer. If PIG.PGMS.ACCT is one such program, then writing a short program that is named PIG and renaming the original PIG program to OLDPIG may provide a good solution to this performance problem. The new PIG program will do the following:

1) Perform a CREATEPROCESS on OLDPIG

2) Force OLDPIG into a lower sub-queue (DS or ES)

The beauty of this is that the user never knows what happened (unless they happen to do a SHOWQ and see their process at a low priority!). Users run PIG.PGMS.ACCT just like normal, but instead of being dispatched by MPE in the CS sub-queue (default for sessions), it is really executed at a similar priority as batch jobs.

Another way to accomplish this is to create the process in the CS sub-queue, and then after a specified interval of time, suspend the process. The new program then awakens the process to run again and then suspends it. This continues until the program runs to completion.

Using the first example for compilers might prove to be valuable and may help encourage programmers to perform compilations via batch in order to free up their terminal for other work during program compilation. See Idea #OM25 for another way to accomplish this.

Chapter 2

Operational and Management Considerations

Presented in this chapter are ways in which system managers and operations personnel may help insure optimal system performance. Most of these ideas fall under the umbrella of good HP 3000 housekeeping. Since housekeeping usually involves access to key resources and tools, certain capabilities (SM, OP, etc.) may be required. For example, dealing with system-wide UDCs will necessitate having SM capability.

Good management habits (with respect to system performance) developed early in an HP 3000 shop will provide a tremendous return in terms of resources not being wasted or abused. While no one idea in this chapter will revolutionize system performance (though perhaps in extreme cases some may), the sum of many of them integrated into your day-to-day operations will add up to the following benefits: optimal resource utilization, smoothing of system load, maximum disc space availability, delaying hardware upgrades, and more.

These ideas will provide the maximum benefit when, one by one, they are implemented into existing procedures. However, it is possible to be overwhelmed by the sheer number of the ideas presented. You are encouraged to choose and implement ones that you understand and feel comfortable with. By all means, experiment with the different ideas presented, and keep good notes.

PROGRAM ALLOCATION

#OM1 Allocate Frequently Used Programs

Allocating program files is nearly equivalent to running them except for actually beginning code execution. There are, however, a couple more steps which MPE performs when the RUN command is actually issued. It is good practice to issue the ALLOCATE command in favor of programs which are run more than once or twice per day. A common way of implementing program allocation is to include the allocations within the SYSSTART file, logon UDC for OPERATOR.SYS, or in a job executed at system startup time. Programs need to be re-allocated with each system startup, unless AUTOALLOCATE is utilized (see Idea #OM2).

Be aware that certain system tables are utilized when programs are allocated. These tables are: Code Segment table (CST), Code Segment Extension table (CSTX), CSTX Block table (number of concurrent running programs), the Loader Segment table (LST), and the Data Segment table (DST).

*** * * * * TRY THIS * * * * ***

Disable AUTOALLOCATE on your system if it is enabled. When the system is fairly busy, time how long it takes from the time that you type RUN <PROGNAME> to the time you actually receive your first prompt (EDITOR is a good choice to try). Be sure this program was not previously ALLOCATED. Then after writing down this time, issue the ALLOCATE command and then RUN the program once again noting the time. If the system is busy, the difference may even be perceivable without using a stopwatch.

Every program to be run must be fed through the MPE loader process. As was mentioned earlier, this process is single threaded. That is, only one process at a time may be truly loaded to prepare for the actual execution of that process. If two programs are ready to run, one will most certainly have to wait in line for the loader process to become available. If a fair number of programs are run throughout the day, this wait time could have a moderate impact on the overall performance of the system. Attending to small performance leaks may prevent floods from becoming a problem. The syntax for allocating a program is quite simple. It is as follows:

ALLOCATE <program.group.acct>

#OM2 Enable AUTO-ALLOCATE

As of the UA-mit and UB-mit versions of MPE, a new feature was introduced. The AUTOALLOCATE command, enabled by default, provides for permanent allocation of a program file whenever it is run. Only when certain system tables are needed by other processes are programs de-allocated. In early versions of MPE which supported AUTOALLOCATE, there were some fairly serious bugs associated with it. It is likely that most problems are resolved at the time of this book's printing. Other than any known bugs, HP has said that there are not any good reasons to disable the auto-allocate feature.

A new command associated with AUTOALLOCATE is the SHOWALLOCATE command. It provides a listing of all programs which have been allocated as a result of the AUTOALLOCATE and ALLOCATE commands. Also, remember to DEALLOCATE program files prior to purging, renaming, etc.

USER DEFINED COMMAND (UDC) MANAGEMENT

User defined commands (UDCs) are a mixed blessing: while sometimes eliminating manual effort, they always incur additional system resource utilization. Note especially that although a UDC may be used by a user only once every other day (eliminating some keystrokes), each time that user logs on, the UDCs enabled for that user will incur system overhead. Of course, the overhead incurred by UDCs is going to be more pronounced with users who log on frequently as opposed to those who do so once or twice a day. Simply put, UDCs always burglarize CPU and I/O, but only sometimes relieve some manual labor.

It is therefore up to the system manager to look at the overall UDC strategy as a resource exchange. If one or more system resources are at a premium, then user convenience may have to be sacrificed. If there is ample system resource then one might get quite carried away with UDCs without much penalty.

This section first praises the uses of UDCs, and then provides caveats. Idea #IO9 in Chapter Three discusses the proper blocking of UDC files for optimal performance.

#OM3 Use UDCs to Avoid Unnecessary Keystrokes

A broad definition of system performance must include obtaining the most from the human/machine partnership as possible. One way to maximize human/machine productivity is to provide operators, developers, and end users short cuts in performing their associated tasks. The proper design and implementation of User Defined Commands (UDCs) will make human tasks easier and more efficient and will not unduly burden the computer system. UDCs help cut down on the number of keystrokes required to execute certain commands.

Though adding to human efficiency, UDCs may incur undesirable overhead due to their sheer number and complexity. Increased time to log on to the system for users with numerous and cumbersome UDCs is also a possible side effect.

#OM4 Limit the Number of UDCs Enabled per Level

The number of records in a UDC file is directly proportional to the number of I/Os and, of course, processor time required by each logon. This is illustrated in figure 2.1. Note how the amount of I/O activity (as evidenced by cache requests in the SHOWCACHE display) escalates with the increase of individual user defined commands. There can be a very evident decrease in logon time by simply trimming infrequently used UDCs from UDC files (especially system-wide ones). Though the I/Os varied from test to test, figure 2.1 is illustrative of the fact that logon overhead increases as the number and complexity of UDCs increases. The number of I/Os depends on the number of records in the UDC file.

Logon with:	I/Os Incurred (cache requests)
1 UDC in file	86
2 UDCs in file	88
20 UDCs in file	100
35 UDCs in file	193
1 parm UDC	90

Figure 2.1 - UDCs and I/O Overhead

Whether a UDC is set at the user, account, or system level, overhead is incurred at each logon (interactive or batch) under that level. The file COMMAND.PUB.SYS is the master index for all UDCs on the system. MPE must read this file for each user that has UDCs set at any of the three levels. A UDC directory is then set up for that user's session (or job). Pointers are then set up to all UDC files. Then each line of every UDC is read in. The overhead incurred is obvious. The following exercise illustrates this.

*** * * * * * TRY THIS * * * * * ***

When you have exclusive use of the system (you are the only one logged on and no jobs), do the following: 1) Turn off all UDCs for your user logon via the proper SETCATALOG command. 2) Issue STOPCACHE and STARTCACHE commands on all drives (this flushes all cache domains and resets the number of cache request counters in the SHOWCACHE display. 3) Perform a SHOWCACHE noting the number of total cache requests. 4) Log on again under the same user and perform another SHOWCACHE comparing total cache requests with the one observed in step 3. 5) Enable one or more UDC files via the SETCATALOG command. 6) Repeat steps 2 through 4 again. Note how much more disc I/O overhead (and thus CPU consumption) is incurred with a logon that has UDCs versus one that does not. Try this with various quantities and complexities of UDCs.

#OM5 Keep Appropriate UDC Information at an Appropriate Level

As seen so far under this UDC section, merely having UDCs available incurs overhead. Whether a UDC is ever used, simply logging on to the system incurs a performance cost. A good strategy in UDC management, and therefore performance management, is to only keep those UDCs at the system and account levels which are necessary. If a UDC only needed by programmers is set at the system level, then each logon by every user incurs unnecessary resource consumption. As one system manager noted,

> "From a system basis, there shouldn't be much more in the system UDCs than contractions for the RUN and LISTF command. I make exceptions for MPEX and other commonly accessed tools. UDCs peculiar to software development, operations or system management are best defined at lower levels (account or user).[1]

#OM6 Prohibit Complex UDCs at the System Level

It is important to remember that system wide UDCs are scanned by anyone who issues a HELLO or JOB command. The complexity of a UDC increases the overhead incurred when users log on to the system. By complex we primarily mean the number of lines in the file. By using a simple I/O monitoring method (described in the TRY THIS box under Idea #OM4) this overhead is amply illustrated. Utilize this test for various kinds of UDCs to get a feel for the cost incurred by each. By forcing complex UDCs down towards the account and user levels, a fair amount of overhead is avoided that would be induced when any user logs on.

#OM7 Attempt to Limit one UDC File per Level

Each separate UDC file must be opened and scanned at the time an applicable user logs on. File opens are expensive. Consolidation of UDC files at any one level should be performed to help avoid unnecessary file opens. Ultimately this limits the number of file opens performed which are probably the single most costly component.

#OM8 Limit the Use of File Equates Within UDCs

This idea may seem to be fairly hair splitting, but the author has seen file equates in system-wide or account-wide UDCs be abused. CPU time and memory space are impacted by the over-use of file equations. Since each file equation must be looked at for just about every file open, CPU time is therefore wasted. If a lot of equates are set, say at the system level, it may be imagined as to the number of scans which must be performed for each file open. Although the time required to scan these equates is very small, it may add up if abused. It is recommended that only the most commonly used file statements be set system wide, and the less needed ones at either the account or user level.

---

1 Guy Smith, *Chase Busy Files to Spread HP 3000 Load,* (The Chronicle, December, 1986), p. 45

TRADITIONALLY-TABOO TUNING COMMANDS

The TUNE command is one of the few mechanisms available on the system that actually changes some of the rules by which MPE makes decisions. It must first be said that the TUNE command is capable of creating havoc on your system. Try the examples listed cautiously and check the parameter modifications with the SHOWQ command. In certain situations modifying scheduling parameters with the TUNE command will produce dramatic performance improvements. However, while the best case might be great, the worst case could be devastating. Just randomly altering the TUNE parameters will no doubt cause bizarre system behavior; Caveat Emptor.

#OM9 Provide Ailing Batch Jobs More CPU Attention by Overlapping the CS and DS Sub-queues.

Many have experienced the never-ending batch job. This may be due to the nature of the job, or the following scenario. If activity in the higher priority scheduling queues is excessive, it is possible for batch jobs to be virtually ignored. It is also possible for them to receive so little attention that they would take an untenable amount of time to complete.

A dramatic example of one instance occurred when the author was creating an operating system installation tape via a batch job. This was attempted while the system was extremely busy. The tape was barely moving every couple of minutes, when under normal conditions it would have been spinning quite consistently. After issuing the following command, it immediately received CPU attention enough to cause it to spin rapidly. The command used to accomplish this is as follows:

```
TUNE;DQ=198
```

The MPE scheduling algorithm utilizes a numbering system to determine which processes are to be favored with CPU attention more often than others. Since session processes typically execute in a medium priority range from 152 to 200, and DS queue batch jobs execute in the range of 202 to 238, overlapping the base of the DS queue (202 - default) into the CS queue by a small amount (in this case 198) will cause *ALL* batch jobs to compete for the CPU with some session processes.

CAUTION! Be extremely careful when using the TUNE command. Modification of just one parameter may cause unpredictable and erratic system performance. If you try some of the examples in this book, be diligent in returning the parameters back to the condition they were in prior to modifying them. Use the SHOWQ command to check your results.

Even though batch jobs may be favored, there is a cost associated with this benefit. Remember the axiom of "no free lunch." Users running in the CS sub-queue may suffer somewhat while batch jobs receive increased CPU attention. Be sure to return the TUNE parameter back to its original value by executing the following command:

```
TUNE;DQ=202
```

The SHOWQ command may be used to check the current values of these tuning parameters. Figure 2.2 illustrates what the bottom portion of a typical SHOWQ listing looks like.

```
D  U85    #J17
D  U86    #J17
D  U87    #J17
D  U88    #J17
D  U89    #J17
D  M330   #J61
D  U335   #J61

CQ MINQUANTUM=0, MAXQUANTUM=300, BASEPRI=152, LIMITPRI=200
DQ MINQUANTUM=1000, MAXQUANTUM=1000, BASEPRI=202, LIMITPRI=238
EQ MINQUANTUM=1000, MAXQUANTUM=1000, BASEPRI=240, LIMITPRI=253
MINIMUM CLOCK CYCLE=1000
```

Figure 2.2 The SHOWQ Summary Display

#OM10 Use the TUNE Command to Penalize Heavy Online CPU Hogs

Forcing the limit of the CS sub-queue to 236, or greater, will punish interactive sessions which are consuming an excessive amount of CPU. A good example of this would be online program compilation. These gluttonous processes will now be pushed over the edge of the CS sub-queue into the DS realm to compete with batch jobs. There they will vie with batch jobs for CPU attention. While those processes are lowered in priority, users whose transactions are short in nature will be rewarded with ample CPU attention. This is a good way to give DQ jobs temporary priority over long CQ transactions. As a result, some may initiate quicker, and short jobs may complete quickly. This would be accomplished by issuing:

TUNE;CQ=,236

#OM11 Use the TUNE Command to Favor Short Transactions

The general scheme on most HP 3000s is to favor interactive, short transactions and to allow longer batch-type ones to be given less resource consideration. Thus MPE compares the amount of time a process took to complete a transaction with the average amount of time it took for all processes to do so (ASTT - Average Short Transaction Time). If a process exceeds this ASTT value before completing a transaction, its priority is decreased. The MAX parameter of the SHOWQ display specifies the maximum number of milliseconds that a process may use the CPU before its priority is reduced. If we lower this from 300 (default) to 200, then processes which incur longer transactions will be rescheduled more often and thus penalized. The command used to favor short transactions is:

TUNE;CQ=,,0,200

DISC SPACE RESOURCE MANAGEMENT

Disc space is one component in the overall performance management game plan. In this section we are not speaking of disc I/O primarily (though it is

impacted), but mostly of the quality and quantity of available space. What is meant by quantity of disc space may be obvious; the meaning of quality may not be. In this author's definition, quality disc space is equivalent to having few large pieces of available space as opposed to many small ones.

The second law of thermodynamics simply stated says, "things are gonna get worse if left to themselves." This is especially true with disc space. Fragmentation, or the de-evolution of large free spaces on disc will affect the operating system's ability to locate needed space in a timely manner. Minimizing disc fragmentation helps insure maximum disc resource availability. Fragmentation is aggravated by a number of things:

1) The number of spoolfiles created, especially $STDLIST

2) The default spoolfile extent size (keep as large as possible - within reason)

3) The number and size of files which applications create

This section stresses the importance of developing and implementing a disc space management strategy, and then provides individual ideas to buttress and include within that plan.

#OM12 Set Up a Game Plan for Disc Free Space Management

An adequate number of large pieces of available disc space will help insure optimal disc I/O performance. Lack thereof will cause MPE to perform an unnecessary amount of searching from disc to disc when space is needed. Not only is search time avoided when large file space requests are made, but an inadequate amount of and/or fragmented disc space may contribute to spooler problems and job launching difficulties. Disc space problems often show up in these two areas since spooling and job launching represent areas of disc file utilization which are very dynamic, yet widely ignored.

A shop which monitors free space at least weekly and has a pro-active free space and fragmentation management plan will rarely, if ever, suffer the ills which are associated with a scarcity of this resource. As Jim Dowling has said,

> "Lack of disc space...isn't handled very well at all by MPE. Print spooling stops, programs fail, system backups fail and it even may become impossible to start the machine after it is shut down or fails."[2]

Appendix B is basically a primer for managing disc space. An excellent article on the subject was written by Jim Dowling and appeared in the July 1987 issue of the HP Professional magazine.

2 Jim Dowling, *HP 3000: Data Center Management,* (HP Professional, July, 1987), P. 64

#OM13 Use the VINIT Utility to Compact Disc Drives

The program VINIT may be used to compact small, fragmented pieces of unused disc space into larger, more useful areas. This program may be used at a terminal, or run in a batch job via the program PVINIT.PUB.SYS. The author favors the use of either of these programs (under appropriate conditions) to pro-actively manage disc fragmentation. The guidelines and explanation set forth in Appendix B are provided to help you not only understand the best way to tackle disc fragmentation, but also to assist you in setting up a manageable way to deal with this problem.

#OM14 Perform a Full System RELOAD to Compact Disc Drives

A full system RELOAD is the most thorough way to completely compact discs and recover lost disc space. However, it is probably the most costly. It also is somewhat dangerous due to the fact that most sites only perform one full backup before doing a RELOAD.

The minute a RELOAD is commenced, the files on all system domain volumes are effectively wiped clean. At that point your data is most likely only in one place...on your current full backup tape set (which was validated...right?).

For this reason it is also recommended that two, full, validated backups be performed prior to the RELOAD. HP's System Operation and Resource Management Reference manual explains the various RELOAD options. Appendix B also provides information regarding reloads and disc space management. Be aware that unless you selectively move or restore files after a RELOAD your file placement strategy will be adversely affected.

#OM15 Perform a RECOVER LOST DISC SPACE to Maximize Disc Freespace

Lost disc space recovery is best performed on a periodic basis particularly after a system failure has occurred. Doing so will reclaim "vanished" disc space due to temporary files and spoolfiles being open during a system interruption. The space occupied by these files is not recovered automatically by the system upon restart (except spoolfiles on a WARMSTART).

It must be said that, to the best of the author's knowledge, space is not lost on a routine basis. It is only an interrupt, when the above mentioned files are open, that incurs this loss. Again, Appendix B explains space recovery more thoroughly. Space recovery is optional on COOLSTARTs, UPDATEs, and COLDSTARTs, not available on a WARMSTART, and implicitly occurs during any of the five RELOAD options.

#OM16 Build Large Dummy Files to "Hide" Disc Space

One trick used many times by the author in potential disc space shortage situations is to create various sized files with or without any data in them. Then, as

space is needed for critical operations, selective ones are purged one-by-one. Placing pseudo data in the files as well as naming them something other than DUMMY, may help deter disc space thieves from purging them. When issuing the BUILD command, be sure to allocate just one large extent as follows:

```
BUILD DUMMY60;DISC=60000,1,1;DEV=1
```

This command reserves 60,000 sectors of disc space (on LDEV 1 for system recovery) which consists of one contiguous piece of disc space. It may be then purged at a later point when you need it most.

#OM17 Inflate Virtual Memory to Provide for Future Disc Space Needs

Another way to hide disc space, and to insure that no one captures it is to increase virtual memory beyond what is deemed necessary for normal operation. This space is really hidden, apart from the fact that users may notice certain drives contain large numbers for virtual memory. Then when space needs exceed that which is currently available, de-configuring some on drives other than LDEV 1, which requires a RELOAD, via a COOLSTART (or, better, a COLDSTART) will provide more. This method has the advantage of making this space relatively difficult to get at, and therefore encourages thriftiness with this resource; you will only use it when absolutely necessary. However, this does have a drawback of increasing head movement between user files and the freespace table.

#OM18 Locate and Shrink Files Which Contain Wasted Space

Often times MPE files are created with more available record space than is necessary even for distant future use. Re-setting a file's LIMIT back to or near the actual EOF mark will free up more space. Regularly auditing files which have a wide variation between the LIMIT and EOF will help insure that a few sectors scattered about do not add up to large quantities of wasted disc space resource. See Idea #IO8 for ways to accomplish file "scrunching." Locating these files may be performed by scanning with LISTF, monitoring backup listings, or using MPEX (VESOFT).

#OM19 Facilitate Mass File Purging with MPEX

Whether Murphy thought of it first or not, disc drives tend to fill up quickly, especially in a development environment. Three of the most expensive commands in terms of resource consumption are HELLO, RUN, and BUILD (e.g. KEEP in EDITOR). In every shop the author has worked or supported, disc space shortage has eventually become an issue. As a result, system managers send out pleas via WELCOME messages and memos. Being on the receiving end of these messages has made this writer aware of one thing: users will not heed those warnings (threats?) to eliminate unnecessary files unless it is either very easy to do so or it becomes very expensive not to do so (unable to build files, threat of job loss, etc).

Various methods have been developed to facilitate "searching and destroying" offending files. Most of these require varying degrees of manual intervention. One way would be to utilize the FLUSH feature of TDP. This is accomplished by issuing:

RUN TDP.PUB.SYS;INFO="SET PERMYES;FLUSH <fileset>

The fileset may include any set of MPE supported wildcards.

However, one of the best tools for accomplishing this task is MPEX (VESOFT). Though it is not free, this tool has very powerful commands for identifying various offending files by many criteria. Once identified, files may then be purged en masse. For example, to locate all files which have over 2000 sectors of used space, are not databases (PRIV file code), and are not in the PROD account, one would issue the following MPEX commands to locate and then purge them:

%!LISTF @.@.@ - @.@.PROD (CODE< >PRIV & DISCSPACE > 2000)

%!PURGE @.@.@ - @.@.PROD (CODE< >PRIV & DISCSPACE > 2000)

Needless to say, accomplishing the above tasks manually would require a fair amount of time. Whether MPEX or other means are used, freeing up as much disc space as possible is one way to insure maximum availability of this resource. Providing an easy way for your personnel to clean up file space will help accomplish this goal.

MISCELLANEOUS

#OM20 Disallow Indiscriminate use of FREE5

Illustrating the overhead incurred by FREE5, if you have never seen it, is easy to do. Simply run FREE5 during production hours and then sit back and wait for a few "are we down" phone calls. FREE5 performs a very large number of reads from the free space bit map, thus incurring heavy, temporary overhead. For this reason, place a lockword on the FREE5 program file. It is also a good idea to only allow access to it by responsible personnel.

#OM21 Restrict the Use of PURGEGROUP and PURGEACCT

Like the discussion for FREE5 above, the PURGEGROUP and PURGEACCT commands have a devastating effect on processes needing access to the system directory; directory accesses are fairly frequent in the typical HP 3000 environment. These commands should be executed only during non-peak hours. SM capability is required for PURGEACCT and AM capability for PURGEGROUP, so access is already somewhat limited by virtue of MPE's security rules. If necessary to perform either when users are active, it is good practice to give fair warning to all with the TELL command. This may help preempt any "system down?" phone call queries.

As an alternative to these commands, use MPEX's (VESOFT) %PURGE @.GROUP or %PURGE @.@.ACCOUNT. These commands do not lock the directory and thus incur far less overhead.

#OM22 Use Faster Copying Utilities than FCOPY

Use of DSCOPY or MPEX's FCOPY,FAST will dramatically reduce the time it takes to copy files. This is because FCOPY does not use multi-record NOBUF (which actually means unbuffered, multiple blocks) in performing its copying. The author has seen improvement up to ten times as fast using MPEX's FCOPY,FAST over MPE's FCOPY. FCOPY has other functionality that must not be ignored, but for simple file transfers it ought not to be utilized if there are faster alternatives.

#OM23 Use KLA/EXPRESS to Improve System Performance

This program from KLA & Associates acts as an extension to the MPE dispatcher/scheduler. It has been described by users of products which are notorious for being CPU hogs as indispensable. It will actually favor light CPU users, but penalize CPU hogs by sending them down into a lower sub-queue priority. The program has a range of customizing features which allow special situations to be favored. For example, batch jobs which normally get little attention may be favored in such a way as to not impact online activity excessively.

The author has heard of various HP 3000 sites in which response times have dropped from minutes to seconds. At the time of printing, the price for this product seems to be quite attractive in terms of price/performance. A demo of this product could be well worth your time. KLA/EXPRESS makes use of all available sub-queues within MPE's scheduling domain and therefore helps to maximize system performance.

#OM24 Implement a Faster Backup Scheme than MPE's Store

Maximum system availability and load satisfaction is the ultimate goal in performance management. This being the case, the time required for system backup reduces system availability. A number of utilities to perform backups faster and utilizing less tape have recently appeared on the market. Most of these products use data compression, special buffering, and even use different tape formats to accomplish faster backups. Time savings has been reported up to 80% from one vendor, but as a conservative high (based on customer feedback), they estimate a high of 55%. Due to the compression schemes, as much as 50% less tape is necessary in some cases. Listed below are products and vendors which the author is aware of at printing time:

PRODUCT	VENDOR
BACKPACK	TYMLABS
COPYCAT	Hewlett-Packard
TurboStore	Hewlett-Packard
BACKUP	Orbit
HIBACK	HI-COMP

#OM25 Remove IA Capability from CPU Hog Programs

Some CPU intensive programs ought not to be run interactively in many environments. If done so, CPU time is stolen from other online users. Typical offenders are compilers and report writers. These are usually best run in batch.

One way to prevent certain programs from being utilized online (CS queue) is to remove IA (interactive) capability from them. This might be considered by most programmers a "below-the-belt" shot, but nevertheless could be the best thing for insuring optimal interactive response times. This method will force these programs to be run in the batch queues (DS & ES). Either recompiling the program (leaving off IA capability), using MPEX %ALTFILE(CAP=-IA), or utilizing the contributed library program MAXCAP will accomplish this. Also, see Idea #OM10 for a way to drive heavy interactive CPU users down into the DS queue.

*** * * * Notes * * * ***

Chapter 3

Disc I/O Optimization & Reduction

This chapter deals with what has been generally understood to be the most common performance bottleneck on the HP 3000: Disc I/Os. There are four areas which contribute to I/O impedance:

1) Disc drives are slow. Their speed is limited to the electro-mechanical mechanisms employed to exchange data from disc to the computer.

2) Application software often times does not take advantage of efficient I/O transfer schemes.

3) Often times the CPU is not driven hard enough to keep up with the amount of throughput the discs are capable of handling.

4) Too much I/O demand by virtue of database inefficiencies, etc. is also another area in which I/Os can become excessive. Unreasonable demands are then placed on I/O devices.

It must be said up front that MPE Disc Cache has reduced many of the harmful effects in these problem areas. However, any one area may become severe enough to keep caching from maintaining acceptable throughput.

The physical limitations of disc drives is perhaps the greatest reason why I/Os are slow. Electro-mechanical accesses operate at a "snail's pace" in comparison to purely electrical ones. There are a number of new features and products which have emerged to help overcome the obstacles associated with disc I/O access. Disc controller caching, disc RPS, MPE Disc Caching, ram discs, and copying and sorting products which utilize more efficient algorithms are some of the common ways in which HP and non-HP vendors have tried to slay the I/O dragon.

Most of the products just mentioned will provide some measure of performance improvement. However, the typical disclaimer given with every performance enhancement idea or product is still true: system performance is application dependant. It is unreasonable to expect to see the same improvement on one machine with its own unique application mix/load as on another machine utilizing the same improvement feature(s). It really depends on where the bottleneck is and to what extent it is contributing to performance degradation.

In reality, it is often times poor software architecture which contributes heavily to I/O bottlenecks. Software design should provide for large file blocking factors to maximize the number of logical records transferred with each physical access; some regard this as a cardinal rule in application design. For example, SUPRTOOL (Robelle) is much faster than TurboIMAGE in data acquisition due, in part, to the fact that it utilizes enormous blocking factors. Attention should be given in the application design stage to optimize I/O transfers by making each physical disc access count.

There is also a speed disparity between CPU and I/O system at times. What is meant here is in many instances not enough multiprogramming is accomplished on the system to take advantage of the total capability of the I/O system. The CPU is a serial device; only one user can be serviced at a time, while there are usually many disc drives which are capable of dealing with many users at once. Simply put, the CPU is not pushed hard enough to keep up with all the disc drives. This is, of course, more pronounced on systems with many disc devices. Though we do not want to swamp the CPU, it is important to push it up to just before saturation in order to fully utilize the I/O system. Disc Caching has mitigated this problem (and others) by utilizing more CPU to facilitate faster I/O access.

In this chapter we will suggest some ideas which will help you either minimize disc I/Os, balance I/O requests more efficiently between drives, or just assist the system in processing these ever-so-precious I/Os as efficiently as possible. It must be noted that all ideas pertaining to maximizing I/O efficiency are not presented in this chapter. Some are presented under more appropriate sections like TurboIMAGE (chapter 4) and Hardware and Configuration Considerations (chapter 5).

DISC CACHING: Explained

John Busch and Alan Kondoff wrote in a paper describing Disc Caching,

> "MPE Disc Caching is an optional MPE subsystem which manages retrieval and replacement of disc "domains" or regions in excess main memory. It locates, moves, and replaces disc domains in main memory so that a significant portion of the references to disc storage can be resolved without incurring physical disc access delays."[1]

MPE Disc Caching has provided a substantial amount of relief in the area of I/O bottleneck for many systems. Systems which have excess CPU and main memory resource available usually experience from good to out-of-this-world improvements in system response times after the addition of Disc Caching. Essentially, Disc Caching takes advantage of a surplus of CPU and main memory to overcome the weaknesses associated with disc request bottlenecks. There are conditions, however, where caching will not provide a benefit; it may even incur performance degradation.

The effectiveness of Disc Caching is based on a simple principle: by bringing more data into memory than what was actually requested, the system "hopes" that this pre-fetched data will be read from or written to sometime in the near future. If

1 John Busch, Alan Kondoff, *MPE Disc Cache: In Perspective,* Publication specifics unknown

need for this data does occur, a disc request is eliminated; the I/O was satisfied by data in memory. Since an average disc I/O can take, on average, 600 milliseconds to complete, and a memory data transfer can be less than 4 milliseconds, the I/O request is satisfied many times faster. Considering the amount of I/Os performed on the typical HP 3000, this can prove to be a substantial time savings in a day.

NO FREE LUNCH

Of course we have gained nothing for free. Some CPU horsepower and main memory space has been spent. So one trade-off is the CPU time consumed to manage the caching mechanism. As a result of this management, there is less CPU available for user processing. If a system does not have a fair amount of CPU available for cache overhead (20 to 30 percent is not an uncommon high range for CPU busy on cache overhead), the benefits of caching may not be realized. It is possible that overall system transaction throughput (including terminal response times) may be negatively impacted.

The other trade-off is main memory availability. Disc Cache domains do not receive preferential treatment (after U-MIT), and will add to memory consumption. This is why a Disc Caching upgrade includes more main memory. The assumption is that before caching is installed your system has adequate memory. This is not always a good assumption. Many shops tend to push a particular resource's usage to the limit. Memory is no exception. Here is the bottom line regarding memory availability: MORE IS BETTER.

If memory is available, the Disc Caching manager will allocate portions of memory to store data recently requested from disc. The new areas created in memory to accommodate this data are called "cache domains." The greater number of domains, the more CPU will be required to manage them.

With the advent of Disc Caching, the designers provided four ways for system managers to manage the caching facility. The Disc Caching Control Panel is essentially "two dials and a switch..."[2]

DISC CACHING MANAGEMENT MECHANISMS

The four ways in which the Disc Caching facility is managed are:

1) Enable/disable caching on a per-drive basis by using the STARTCACHE <ldev> and STOPCACHE <ldev> commands respectively.

2) Enable/disable BLOCKONWRITE for all cached drives by using CACHECONTROL BLOCKONWRITE=YES/NO respectively.

---

2 Steve Cooper, *Variations on a Tune - Another Look at the Never-Ending Struggle Towards Optimal Performance,* (Proceedings of the Interex HP 3000 Madrid Conference, March, 1986), p. 415

3) Alter the size of cache domains created as the result of a sequential disc request with CACHECONTROL SEQUENTIAL=<1-96>.

4) Alter the size of cache domains created as the result of random disc requests with CACHECONTROL RANDOM=<1-96>.

These management mechanisms will be discussed in the following ideas.

#IO1 Use MPE Disc Caching

It is assumed that the majority of HP 3000 systems are utilizing Disc Caching. Caching is not supported on Series II, III, 30, or 33 HP 3000s. However, if you are on a series that does support caching, you will probably benefit from using it. As Steve Cooper aptly put it,

> "Caching is a wonderful thing if you've got the extra memory and are I/O bound."[3]

The benefit of Disc Caching will vary from system to system depending on a few criteria. If you are able to answer "yes" to the following four questions, then you will probably benefit from it:

1) Does the system have, on average, a surplus of CPU available (30% to 40% for most systems)?

2) Is there ample main memory available? It is best to insure that memory need is adequate for your current processing needs before adding Disc Caching.

3) Is there good locality to your disc reads? The author knows of no easy way to measure this other than asking HP to run a program called IODCP (threaten to buy something and they might do it for free!). Good locality is when the majority of disc accesses are clustered in groups physically on disc. Essentially locality is a measurement of how closely related your requests are in terms of disc addresses accessed.

4) Does the system have evidence of being I/O bound? A consistent average CPU pause for disc I/O in excess of 20 to 30 percent may point to I/O binding (assuming caching is disabled). Use of OPT, Surveyor, Probe, or Sysview will help here.

If all four of these are at least somewhat true on your system, you will probably be surprised by the benefit realized with the addition of Disc Caching. Once again, trying is knowing. Perhaps your HP sales representative will accommodate a "try and buy" on Disc Caching! Nearly every system the author has seen has benefited from caching to some extent.

One thing to watch out for when deciding on whether or not to use Disc Caching is for applications which perform backwards chained or serial record accesses. Caching obtains data from disc by beginning at a sector and retrieving data **forward**

3 Ibid., p. 416

from that position. Applications which perform reversed reads will therefore guarantee an exceedingly low number of future hits to cached domains. Temporarily disabling Disc Caching while such applications are running will eliminate unnecessarily spent CPU overhead.

#IO2 Experiment with the BLOCKONWRITE Parameter of the CACHECONTROL Command

This parameter is the one "switch" available on the Caching control panel. In essence, this controls whether a process continues executing when it performs a write or whether it is blocked (made to wait) until the request is safely posted to disc. The issues at stake are data integrity and performance.

By choosing BLOCKONWRITE=YES, you have better data integrity in the event of a system failure, but this comes at the expense of system performance. Setting BLOCKONWRITE=NO will, in most cases, produce better performance. But a system interruption might leave a database in a compromised state unless you are using IMAGE logging and/or Intrinsic Level Recovery (ILR). The following discussion will explain why.

If BLOCKONWRITE=YES and a write request is satisfied with a hit to a cache domain, the process will wait until the cache domain is flushed to disc, thus insuring the disc copy is updated. The process has to wait; performance is degraded. By specifying BLOCKONWRITE=NO, the cache manager requests an I/O to flush the domain to disc and control immediately returns to the program even if the physical post has not occurred. If that same cache domain is written to before the I/O is actually posted to disc, the calling program waits. This is why some consider the WRITE HITS display of the SHOWCACHE listing to be bad. But as Steve Cooper has rightly stated,

> "We want to minimize write hits to domains that are being written out at that moment. Write hits to "clean" domains are a good thing. Unfortunately, the cache statistics do not separate the good hits from the bad."[4]

In most cases, integrity issues aside, if you are using Disc Caching, you probably ought to set BLOCKONWRITE=NO. Utilizing IMAGE logging/ILR will allow you to set BLOCKONWRITE=NO and thus obtain a performance benefit without worrying about compromising integrity. As always, the performance resulting from BLOCKONWRITE=NO is environment specific; it does, however, warrant experimentation. With this parameter as well as the RANDOM and SEQUENTIAL fetch quantum values (discussed below), the best way to optimize is to find combinations of the three which first maximize PERCENT OF USER I/Os ELIMINATED, and secondly minimize DATA OVERHEAD as noted on the SHOWCACHE display.

---

4 Ibid., p. 418

In instances where I/O intensive jobs are competing with online activity and thus "hogging" the system, you might try setting BLOCKONWRITE=YES which will make processes performing write activity give up the CPU more often to allow read intensive processes to run.

*** * * * TRY THIS! * * * ***

Take a single or multiple batch job(s), run them once with BLOCKONWRITE=YES noting completion times. Repeat them again with BLOCKONWRITE=NO and compare the times. Try this also with batch and your interactive users. Gather statistics on user response times during typically busy days that include batch activity. Compare these timings with BLOCKONWRITE=NO and YES.

#IO3 Alter the Sequential Fetch Quantum of the CACHECONTROL Command

The Sequential Fetch Quantum value is the parameter that instructs the cache manager how much data to bring into a cache domain when a sequential type operation is performed. If an application is performing primarily sequential reads, then setting this value to a large number (96 is the maximum) will help insure that more future reads will be satisfied in memory rather than from disc. The expense involved is memory. By specifying 96 for the Sequential Fetch Quantum, 96 sectors of data is brought into memory each time a serial file operation occurs. Note that the default is 96.

The benefit provided by Disc Caching should not be impeded by not having enough main memory. Since a very undesirable thing is being eliminated (physical disc I/Os), it is better to insure a surplus of memory so that the Sequential value may be set high without causing memory stress. "Pay me now or pay me later!".

Experimentation is the best way to know how altering this parameter is helping or hurting. It is recommended that a set of batch jobs be run and various combinations of CACHECONTROL values be tried to see how the NUMBER OF USER I/Os ELIMINATED changes. Please note that no matter what kind of operation is being performed on an IMAGE database, random type reads and writes are utilized. Therefore, serially read MPE flat files are good candidates for using this "dial" on.

#IO4 Alter the Random Fetch Quantum of the CACHECONTROL Command

You may be wondering why the Disc Caching designers set the RANDOM default value at 16 sectors. If there is plenty of memory, why not bring in as much as possible? The reason for the value of 16 is that HP7933/35 disc drives have a buffer size of 16 sectors. If you have multiple HP7933/35 drives on the same GIC, using Disc RPS, and another drive on that same GIC is in-use, then the value of 16 seems

to be best. Otherwise the author has heard of instances where raising this value above 16 has produced fairly impressive performance gains (particularly with IMAGE applications).

Apart from more memory being consumed, setting the RANDOM value to above 16 will help alleviate a phenomenon documented by Jim Dowling. This problem is what he refers to as an "MPE Cache Domain Glut." It is simply described where the system

> "...takes longer to search a long chain of cache domains only to find that a physical I/O is necessary than it would have taken to do the physical I/O in the first place."[5]

Setting the RANDOM value to a small one encourages the proliferation of cache domains. The more the domains, the costlier it becomes for MPE to search through them every time an I/O is requested. Jim observed,

> "Systems with very large cache domain counts (>1200 on a busy drive) will often demonstrate this phenomenon.
>
> The faster CPU configurations do not search the cache domains any faster, but they also do not demonstrate the phenomenon as often.
>
> Caching environments where a drive has a large population of cache domains and has a low caching efficiency demonstrate this effect more prominently."[6]

By raising the RANDOM value, less domains are created, the chances of I/Os being eliminated improves, and we don't contribute to a possible domain glut.

#IO5 Selectively Enable/Disable Caching on Specific Disc Drives

One thing the author would like to see on the SHOWCACHE display would be the number of I/Os eliminated on a per drive basis (The cache activity screen in SYSVIEW (CAROLIAN) breaks this out nicely). Since the Number of User I/Os eliminated is the primary measuring stick of Disc Caching's effectiveness, seeing this number (without having to multiply the READ% by READ HIT%) would help one make the decision to enable or disable caching on a drive-by-drive basis. By disabling caching on one drive that has, say, a low percentage of I/Os eliminated, two things occur. First, some CPU overhead is returned to the system. Second, some memory is given back to the system.

---

5 James Dowling, *An Investigation Into the Effects of Memory Availability on Performance for HP 3000 Systems,* (The System Performance Handbook, Massachusetts, Volz Associates, 1987), p. 7
6 Ibid. p. 7

When is it advantageous to disable Disc Caching on a given drive? First multiply the READ% and READ HIT% on a per drive basis. The ones which have values under 40 to 50 percent or so might be good choices. In an article on Disc Balancing, Andy Tauber cites the following case for disabling caching,

> "In one observed case, when interactive daytime processing was taking place, caching was using 25 to 35 percent of the CPU for cache management and 5 to 6 percent paused for disc. When caching was turned off, the pause for disc went up by 1 or 2 percent and 25 to 35 percent of the CPU used for cache management was then freed for use by interactive processes. This reduced the overall response time on the machine because more CPU was available for user processing."[7]

This is one of the schemes used by a contributed library program called DCO (Disc Caching optimizer), and a third party tool named WATCHDOG (Atlantis). Both of these will dynamically enable/disable caching on a drive-by-drive basis in order to make best use of the CPU and memory resources, and yet not penalize the cache facility needlessly. Again, experimentation will only tell for your environment.

*** * * * * TRY THIS * * * * ***

This is an exercise for multi-drive systems. Perform a SHOWCACHE. Locate a drive that has a low I/O elimination reading (under 45% perhaps). Do this by multiplying the READ% by the READ HIT% and dividing the result by 100. Calculate this I/O elimination percentage for each drive. Begin to disable caching on the lowest numbered drives as you monitor the CPU busy on cache and CPU pause for Disc I/O. Issuing the STOPCACHE command on a drive will return some CPU back for use on user processing, while CPU pause for disc will increase some. The obvious goal is to get more CPU back with little penalty in the pause for I/O department.

#IO6 Use the Program DCO (Disc Cache Optimizer)

This contributed program (authored by an SE in The Netherlands) reportedly runs all the time and has as its aim to optimize all the Disc Cache parameters previously discussed in this chapter. Although the author has not had opportunity to utilize the program, he has heard of varying reports of its usefulness; ranging from fair to good. It is but mere speculation, but the program's author probably developed some algorithms based on certain statistics as seen in the SHOWCACHE command. The program alters the dials and switch based on these data points.

The author has been repeatedly told that it would be available from any HP SE...to no avail thus far. This author would appreciate any information regarding the program, and especially its usefulness. You are encouraged to write Performance Press if you have had experience using this.

---

7 Andy Tauber, *Disc Balancing,* (INTERACT, January, 1986), pp. 60-69

FILE HOUSEKEEPING

Once again, by giving attention to ways which make the file system do its job easier and faster, the system as a whole will benefit. This section encompasses ways to improve file access times through maintenance operations. Tools to assist in these operations will be mentioned when applicable to avoid strictly manual manipulation.

#IO7 Use an Appropriate Blocking Factor for Each File to Reduce Physical I/O

If it isn't clear by now, it should be said again: physical I/Os are not good for performance; they are slow; minimizing them is one goal in performance optimization.

If the blocking factor of a file is low then the number of physical accesses to the disc drive where that file resides will increase. This is because a physical I/O consists of retrieving a number of logical records as specified by the blocking factor. Thus a file with a blocking factor of one will incur one physical access to disc each time a file operation is requested (note that we are ignoring the effects of Disc Caching for this discussion). A blocking factor of twenty will allow the disc drive to retrieve twenty records at a time. If purely random reads and writes are being performed, at worst case a little memory and transfer time is being wasted by having a large blocking factor. If serial operations are being done, this may provide a large performance improvement by minimizing physical I/Os. In our example above, to read twenty records from the file blocked one record per block would incur twenty disc I/Os, while a read to the second file (blocked twenty records per block) would cause only one disc I/O to take place unless, of course, Disc Caching were enabled.

So the formula for determining how many physical I/Os would need to be performed on a file is as follows:

PHYSICAL I/Os = (# OF LOGICAL RECORDS / BLOCKING FACTOR)

Disc I/O tends to be a very common bottleneck on the system. It is seen from this formula that a significant improvement in application performance may be obtained by virtue of physical I/O reduction. Since the block size affects disc transfer times, regularity of disc access, and Disc Caching efficiency, paying attention to file blocking may provide you with a bit more "edge" in system performance.

There are a number of ways to perform file blocking. One way is to issue the MPE command:

BUILD TESTFILE;REC=-80,x,F,ASCII

Where x is the number of logical records per block. Minimizing disc space wasted and producing the maximum records per block can be difficult due to having to manually calculate the blocking factor. Manually, you could create a new file with

the desired blocking factor and then copy the contents of the old, poorly blocked file into it. A better way would be to utilize a contributed library program like BLOCKIT or BLOCKOPT. Another way is to use the BLOCKFACT=BEST feature of MPEX (VESOFT).

A side benefit to choosing an optimal blocking factor is a reduction in disc space wasted. If a file with a record length of 80 bytes is blocked at one record per block then the wasted space is:

WASTE (bytes)=256 (bytes per sector)-80 (record length)

In this case WASTE is 176 bytes per record. For a file of 2000 records this represents a loss of 1,375 sectors!

As far as performance is concerned, setting blocking factors high is extremely important. It is even possible, in some instances, to obtain better throughput by disabling Disc Caching and building data files with very large blocking factors! Any block less than 8k bytes is probably too small. You might start with a blocking factor of 100 and see if you get an error (exceeded maximum block size 24kb). It has been argued that such large blocks waste memory. This isn't the case, however. Since it is primarily batch jobs which are doing such large, sequential file accesses, and since normally only one or two such jobs are running, the buffer area for their data is probably quite small in comparison to the amount of memory on most systems. This is especially true with the deflation of memory prices in recent times.

Bottom line, it is better to set the block factors high for sequentially accessed files and ask questions later. Keep in mind also, that generally speaking, the larger the blocking factor the less disc space is wasted .

#IO8 Minimize File Allocation Time and Wasted Disc Space by Resetting the EOF on Files Which have Empty Space in Them

If a file is structurally built to hold 3500 records, but it only contains 175, there are 3,325 records storing "air". This results not only in wasted disc space, but if the file consists of a single extent, the time to allocate the file is longer than necessary. Condensing such files will at a minimum save on disc space, and if it is a single extent file, the time to allocate (part of the open) it will be reduced.

Accomplish file condensing by either using FCOPY or the SQUEEZE feature in MPEX. Using FCOPY on file FWASTE to produce a file called FCONSERV simply issue the following command:

FCOPY FROM=FWASTE;TO=FCONSERV;NEW;SUBSET

The SUBSET option has the effect of shrinking the file's limit back down to the end of file marker. A resulting LISTF might look like:

```
FILENAME  CODE --------LOGICAL RECORD----  ------SPACE----
                SIZE  TYP   EOF   LIMIT  R/B  SECTORS  #X  MX

FWASTE           80B  FA     20    1023    1      128    1   8
FCONSERV         80B  FA     20      20    1       21    7   7
```

A more efficient way would be to user the following MPEX command:

%ALTFILE FWASTE;SQUEEZE

This way a second step is avoided in having to rename the FCONSERV file back to the desired name.

#IO9 Block UDC Files with a Large Blocking Factor

Logon times vary dramatically in proportion to the number of UDC commands enabled for the user as well as the blocking factor of the UDC command files themselves. However, if you have Disc Caching this impedance is glossed over somewhat. Nevertheless, by reblocking the files in such a way that a minimum number of extents are used and a large-as-possible blocking factor utilized, I/O overhead is minimized. To build a file with a single extent and with a large blocking factor, issue the following MPE command:

BUILD UDCFILE;REC=-80,X,F,ASCII;DISC=100,Y,Z

where X is a large number (space waste is not much of a concern since UDC files are usually not real large); 100 is not a bad one to try. Y is the number of extents initially allocated and Z is the maximum number of extents ever allowed. Thus Y and Z should be 1. Once you have built this file, copy your UDC file into it.

#IO10 Use a Sorting Tool that is More Efficient than SORT/3000

A number of sort utilities have appeared on the market recently which aim to reduce the number I/Os needed to perform sorting. The author speculates that these utilities are faster due to a more efficient sorting algorithm, larger blocking factors, and Multi-rec/nobuf access.

Here are a few of the third party sorting utilities that the author is aware of: QSORT (OPT) and SORTMATE (Running Mate).

#IO11 Utilize DISC RPS when Appropriate

One of the components involved in disc I/O bottle-necking is called "channel contention." When multiple disc drives share a single GIC (channel), and disc requests are sent for two or more drives on that GIC, only one drive at a time can "hold" the channel and send data to the CPU. One or more drives may have to wait for that channel to become available before it (they) can complete a transaction. The amount of time a disc retains a monopoly on a channel figures into total I/O performance and, accordingly, overall system performance.

Before Disc RPS (Rotational Position Sensing), a disc that received a transaction request would hold the channel from the time it located the target track until the data transfer was complete. A seemingly small component of this time is from the time the track is located up to the time it took for the rotation of the drive (one half rotation on average) to position the target sector for transmission. This hold-up is referred to as rotational delay. If during this delay a second drive was ready to transmit, it would be prevented from doing so by the first drive's monopoly of the channel. Thus the second drive would be forced to wait as its data rotated around a second time or more. This delay contributes to overall I/O impedance.

RPS reduces the channel contention by the following method:

> "Rather than requesting the channel as soon as the target track is found, the drive requests the channel when it is a fixed time from the target sector. This frees the channel for a longer window of time. During this window of free time, other drives can utilize the channel."[8]

It may be obvious, but still worth mentioning, that a single drive on a GIC gains nothing from RPS. Only on multiple drives per GIC will RPS provide benefit. Also note that disc I/O writes of 4 kbytes and less do not benefit from RPS. A feature on 793x drives called Buffer Prefill is used for these writes.

It seems that a fairly high I/O rate is necessary to perceive a substantial benefit from RPS. The exceptions to this seem to be disc drives which have controller caching enabled (793zXP, 7936/37). It is also important to realize that if a non-RPS device shares a GIC with an RPS device, or a RPS device is utilized on a series 37 (unable to maintain a transfer rate as fast as the disc) then RPS may even provide a negative effect. Thus 792x drives should not share the same channel as 793x ones. Your disc drives should have firmware revision 5.1 or later, and you should be on MPE version T-Delta-3 or later (earlier "T" versions need a patch).[9]

#IO12 Utilize Disc Controller Caching When Appropriate

Recall that there are three types of data caching available for the HP 3000: CPU register cache, MPE software Disc Caching, and disc drive controller cache. Each method employs one common goal - to utilize memory to hold pre-fetched data in anticipation of future need for that data.

---

8 Hewlett-Packard Company, *Disc Performance Reference Guide for HP 3000 Systems,* (Idaho, 1987), p. 5-4

9 Ibid., p. 5-5

Disc controller caching (arrived with the 7933/35 drives) utilizes a minimum of one megabyte of memory to store data that is retrieved with each disc read. The goal of controller caching is to lower the amount of disc drive activity necessary to retrieve data. Remember disc drives are slow...hardware memory transfers are fast. Controller caching will scan its pages of data each time a request for disc data is made that is less than 4096 bytes. If the data is available in a cache page (a page consists of 4096 bytes) then the request is satisfied much faster than if a physical read to disc had to take place.

If requested data is not found in a cache page, then a physical read of disc must occur anyway. A slight penalty is incurred due to the fact that some time was lost by having to scan pages in the controller's cache memory.

Controller caching is an option on the 7933/35 family of disc drives. It is also available on third party devices (EMC Corp., most notably). It consists primarily of added memory and the enabling of a firmware caching handling mechanism. According to Hewlett-Packard, controller caching will provide some benefit for most sites that do not have MPE Disc Caching and whom especially have a lack of CPU horsepower. Note the following:

> "Controller cache provides more than just higher levels of disc performance over the standard drives. By moving the caching function out of the CPU and into the disc, valuable CPU cycles and memory can be freed. These resources can then be devoted to other activities. This, combined with the faster disc transactions, provides greater overall system efficiency in the form of system throughput and response time in the appropriate application."[10]

The 7933/35 drives contain one megabyte of RAM memory for read cache and the 7936/37s have two megabytes. The latter drives have provision for write caching in that the write requesting process does not wait until the request is physically posted to disc before continuing. Whereas the 7936/37 drives do have a write caching mechanism (under certain conditions) that allows the calling process to continue immediately after the target page is updated within the cache; the process therefore does not have to wait until the write request physically posts to disc. Thus the 7936/37 drives are said to allow "immediate response" for processes executing disc writes.

But when is disc controller caching appropriate? How will it impact MPE Disc Caching? Is it better to use both if available? These questions are best answered in a specific performance audit on your system, but the following points will at least give you some things to keep in mind in pondering the above questions.

- Using controller caching will allow a greater level of multiprogramming to take place due to freed up CPU. Thus, more jobs can be run if the burden of Disc Caching is removed from the CPU. Also concurrent controller cache searches will be able to take place.

- Your read-hit percentages should be in excess of 70% (as noted on the SHOWCACHE display) for controller caching to be best utilized.

---

10 Cf. *Disc Performance Reference Guide for HP 3000 Systems, op. cit., p. 4-26*

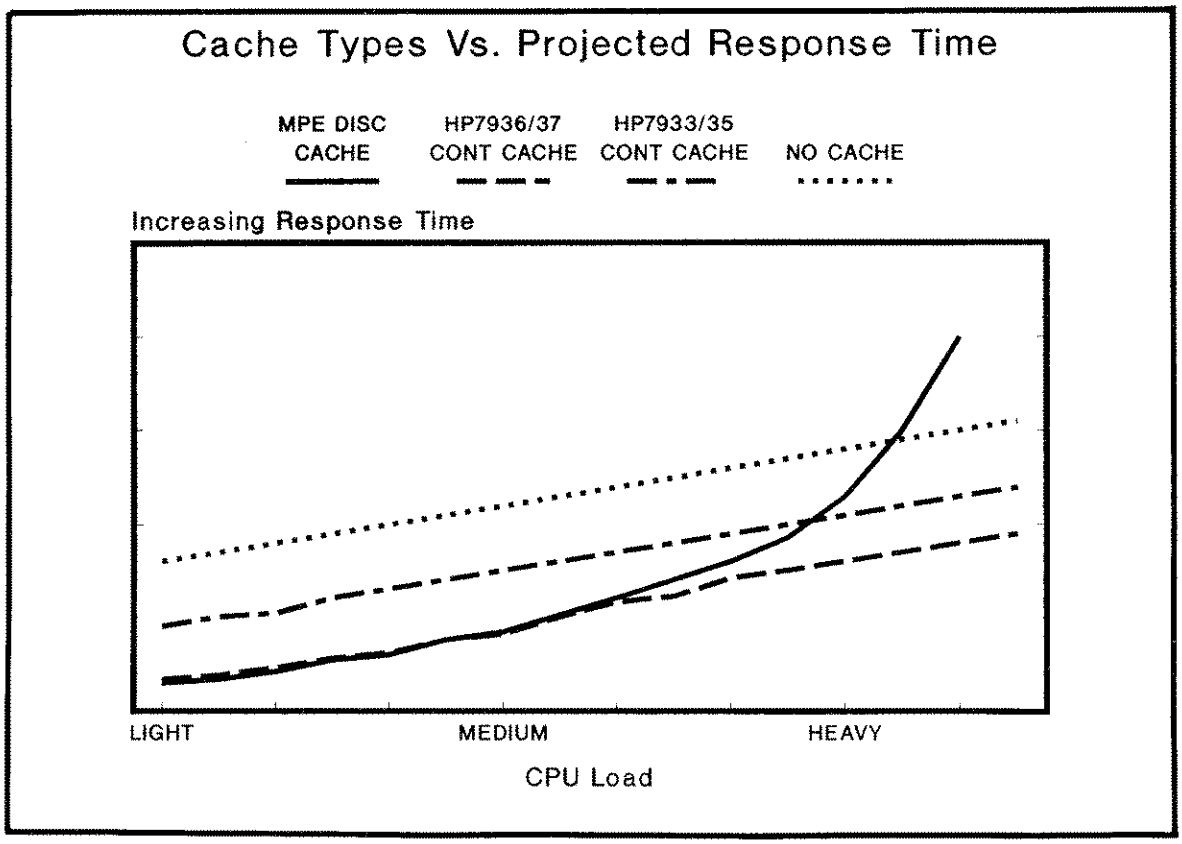


Figure 3.1 - Response Time vs. Caching Mechanisms

- If CPU utilization is greater than 90%, disabling MPE caching and utilizing controller caching is usually advised for 7933/35s; the 7936/37 is not as sensitive to this threshold as the 7933/35. The 7936/37 will provide benefit in some instances even when CPU busy is less than 90%; they really shine when the CPU utilization is over 90%.

- Generally speaking, the following performance is about what you would expect from controller cache and MPE Disc Cache:

  1) Worst - No caching of any kind

  2) Better - HP 7933/35 controller caching

  3) Best - Either MPE Disc Caching OR HP 7936/37 controller caching. While in some batch situations both might be beneficial, generally both together will not be good. Controller caching will out-perform MPE Disc Caching on a very CPU-bound system. Note Figure 3.1 which illustrates these points in terms of response time.

- There are probably only rare instances when both types of caching together would be useful. Note the following:

> "Testing shows that there are actually very few typical HP 3000 environments that will see a significant performance benefit from dual-cache. In most instances, the benefit from disabling MPE Disc Cache and freeing up valuable CPU cycles and memory is greater than that seen from enabling both caching schemes."[11]

So, while controller caching may be a good fit for certain environments, it most certainly is best prescribed on a site-by-site basis. This is one instance where it would be really advisable to seek outside performance consulting to see if the necessary capital outlay would be price/performance effective.

BALANCING DISC I/O

Utilizing every resource on the system as much as possible without approaching saturation is one goal of system performance maximization. This is especially true for disc I/O throughput. The author has seen many systems in which I/O throughput was diminished due to poor I/O balancing policies. Simply by utilizing some of the ideas in this section, one may see markedly improved system performance.

#IO13 Spread Files Across Discs

This idea involves first knowing something about the I/O activity on the system (very, very application dependant), and then taking action by placing files so as to utilize each disc drive as fully and as evenly as possible.

The best way to become familiar with the I/O activity on a per file basis is to run a program that reads system log file statistics and then sorts the data in various ways to provide the user with a list of the most active files. Once this data is known then corrective action may be taken.

It seems that the 80/20 rule shows up in many parts of life, and it is no less true in disc I/Os. Basically it says that 20 percent of the files will receive 80 percent of the disc I/O activity. Of course these numbers do vary, but it is surprising how often they come close to reality. Studies have shown that in a typical HP 3000 shop, most of the I/O activity focuses on a small subset of the files on the system. Identifying these 20 percent (or even less on large systems) and focusing efforts on them will provide you with the most benefit for time/money invested. As Andy Tauber has noted,

> "Disc balancing means spreading highly accessed files as evenly as practical across all disc drives on the system."[12]

---

11 Melody Armstrong, *Disc Controller Cache,* (INTERACT, September, 1987), p. 114
12 Cf. *Disc Balancing,* op. cit., p. 69

The following exercise (though time consuming) could pay real dividends by optimizing disc I/O accesses.

*** * * * * TRY THIS * * * * ***

Enable system logging event #5 (file closes) via SYSDUMP. Log files will grow rapidly, so be forewarned. Obtain a week or so of normal system operation data in the log files (LOG###.PUB.SYS). Then run the program FILERPT or DISCIO (both in the contributed library), following their instructions. Once you have created a report, identify the top 10 to 20 percent heaviest hit files. At a convenient time, perform a reload and specifically place those top-hitting files on alternating disc drives. You might use the DEV=X parameter in the RESTORE command. This is time consuming, but probably worth the effort. MPEX (VESOFT) would work nicely here also. Now you have all the extents of each of those files on the same drive. Let the system collect file usage statistics for another time period, and then run the statistics program once again. Pay close attention to the files that you manually placed. You may want to use a performance monitoring tool that provides disc I/O activity on a per-drive basis to see how balanced the I/Os are. Note also batch job completion times as well as user response times.

#IO14 Place Complementing Files on Separate Drives

Some applications access multiple files in single operations. Perhaps the most common example of this is the IMAGE master and detail relationship during chained retrieval; KSAM key and data files are also an example of this relationship. By having both sets on the same drive, the disc drive head must be repositioned almost constantly. Again the mechanical movement associated with this operation is many times slower than simply transferring data electronically without movement. Placing teamed filesets (IMAGE or otherwise) on alternating disc drives will help reduce the costly head movement. Best performance is achieved if files are located on disc drives which are on separate channels and IMBs.

One major vendor's application depends, in part, on specific disc drive placement for optimum system performance. If, after a reload, the files are skewed apart from the placement scheme prescribed, response times and batch completion times suffer quite noticeably.

#IO15 Insure I/O Locality by Placing Heavily Accessed Files Close Together on Disc Towards the Front/Center of the Drive

Encouraging concentration of I/O activity in as few geographic regions as possible will also contribute to efficiency in the I/O system. This means locating files that are accessed most often and not only spreading them across discs but also placing them in as close proximity to one another as possible. By doing this, costly disc head movements will be minimized. This has been referred to as head locality since it deals with how far the drive arm mechanism must move for each data

retrieval. A substantial amount of movement could impact I/O throughput. Tony Engberg buttresses this point and also explains the cost associated with doing so:

> "I/O should be localized by placing these heavily accessed files in close proximity on your drives, preferably centered on the discs. This is a somewhat painful activity to undertake, since it requires successive partial RESTOREs to specific devices. It is, nonetheless, one means of reducing...time spent being serviced by the I/O system."[13]

As alluded to by Tony, implementing this step involves first identifying these files and then during an ACCOUNTS RELOAD, selectively restoring these files to the drives first. One way to make this job a bit easier would be to create several device classes (e.g. MASTER, DETAIL, ARCHIVE) and then use MPEX to move the files to those specific device classes, or in the case of TurboIMAGE datasets, use the DBUTIL MOVE command.

GENERAL I/O MANAGEMENT

#IO16 Insure Optimal Disc Speed by Maintaining a Cool Temperature in the Computer Room

As funny as this may sound, it is nevertheless true that the speed of a disc drive's head actuator decreases with a rise in operating temperature! Although the author has not seen this formally documented anywhere, he has been repeatedly told that the head seek time decreases as temperature rises above a certain threshold.

#IO17 Keep VPLUS Forms in a Single File

By placing all forms necessary for an application in a single forms file, much file open/access activity is avoided. Also consider procuring terminals which support forms caching as this feature has an astounding effect in the speed by which VPLUS forms are switched and displayed.

13 Tony Engberg, *Response Time: Defining it and Speeding it up,* (INTERACT, April, 1986), p. 48

Chapter 4

Image/3000 Performance

Apart from the fabulous support of Hewlett-Packard and the renowned hardware reliability of the HP 3000 family, the database system known as IMAGE/3000 has contributed greatly to the popularity of this computer system. Many companies have purchased an HP 3000 just to have IMAGE. It is rugged, time-tested, simple, and yet powerful enough to handle incredibly complicated database applications. Although many would argue that it is old and not a relational database, few can argue with the success and reliability experienced by the tens-of-thousands whom have entrusted their data to IMAGE. Where IMAGE does lack, many third parties have arisen to provide utilities to fill these gaps. As famous as IMAGE may be, however, the infamous side of IMAGE is also well known and bemoaned.

From The U-MIT release of MPE and on, IMAGE is known as TurboIMAGE. For our discussion, we will utilize "IMAGE" unless there is a specific need to reference TurboIMAGE.

IMAGE AND THE SECOND LAW OF THERMODYNAMICS

By not abiding by some simple guidelines, a poorly designed IMAGE database will quickly become a performance bottleneck on the system. Worse yet is the raw fact that a well designed database can also become a performance bottleneck. This is due to the inherent entropy which occurs during the course of normal database operations at most HP 3000 sites. Entropy, as you will recall, is the tendency of an energy system to run down. Though not an energy system, IMAGE does exhibit some traits of the classic Second Law of Thermodynamics. It basically states that the entropy (disorder) of an energy system (database) will increase as time goes on. IMAGE has also fallen victim to this law.

As data is input into an IMAGE detail set and deleted again and again, its performance will usually proceed from good to worse if not managed. Certain housekeeping MUST be performed on some regular basis in most data-dynamic environments in order to insure optimal IMAGE performance. If not attended to with at least some understanding of how IMAGE works, its efficiency will indeed proceed from order to disorder.

This chapter discusses ways to design, manage, and monitor IMAGE databases with optimum performance in mind. Once again, design and programming ideas will be briefly presented since one charter of this book is to present as many non-intrusive performance enhancement ideas as possible. A fine, complete reference guide to IMAGE is found in the *IMAGE/3000 Handbook* (Wordware). If you are an IMAGE addict, you no doubt own a copy. This book is virtually nonoptional if you want to really understand IMAGE. Specific calculations and tuning parameters which have delicate criteria will not be discussed in this section. You will be referred to the *IMAGE/3000 Handbook* since the writers of that book (some of whom assisted in the development of IMAGE) discuss just about every conceivable optimizing idea in detail.

IMAGE OPTIMIZATION IDEAS

#IM1 Re-Pack Detail Sets After Mass Deletions

Repacking detail sets is one of the most beneficial yet bothersome ways to improve system performance. It has a most noticeable effect on systems which utilize IMAGE in their primary applications. That is to say, re-packing a detail set will provide a benefit ONLY if data has been deleted (and new records placed in the vacancies) or if data was never originally placed in the set in an efficient manner. The following is a brief explanation of why repacking is usually so necessary to maintaining good IMAGE performance.

When IMAGE places records in detail sets, under most common circumstances, they are placed in chained fashion. That is, an invoice line item record will have an invoice number as a common link with other line items for that particular invoice. Each record in that link "knows" the location of one preceding and one subsequent record (using backward and forward pointers). All of the line items for a particular invoice comprise what is referred to as a chain. Since IMAGE databases are merely special MPE files, they are subject to the same fateful laws of physical disc access as other files. This is where the performance problem occurs.

Every time the invoice referred to above is to be printed, selected line items must be retrieved from disc. If all the records are located physically next to each other, and happen to be in the same block on disc, then only one disc I/O is required to retrieve them for processing. If each chain member lies within a separate block then one disc I/O will be necessary per line item record. Since disc I/Os are slow in relation to memory data transfers, this is the heart of the bottleneck.

Performing detail set reorganization under the latter condition in the above paragraph will place all of the line items next to each other for optimum disc retrieval. The benefit that you obtain from packing these detail chains depends on your application. It especially depends whether or not the primary path (first search item in schema or overridden with a "!") is indeed the most accessed one since sets are repacked based on their primary path.

When do sets need reorganization? Most probably after massive deletions. But the best way to tell is to monitor them on a regular basis. Use either the contributed library program DBLOADNG (quite slow; untenable for very large databases), Robelle's HOWMESSY (fast - ten to twenty times faster than DBLOADNG), or other third party utilities to track a detail set's sloppiness over time . See the IMAGE monitoring section for use of these tools.

The folks at Robelle Consulting performed a study on the different tools available to reorganize detail sets.[1] They used HOWMESSY to see how each one performed. They began with a somewhat messy detail set. The number used as a measuring stick is the elongation factor. This value is a measure of dataset inefficiency with regard to the number of I/Os required to retrieve an average length chain.

Listed below are the products they used as well as the inefficiency numbers resulting. The test dataset began with an elongation of 4.41. This basically means that the number of I/Os required to retrieve all of the records in the average sized block is 4.41 more than if the records were removed, sorted by the primary key value, and then reloaded into adjacent entries on disc. Note the following results:

TOOL/VENDOR	ELONGATION
SUPRTOOL (Robelle)	1.17
DBMGR (DISC)	3.24
DBGENERAL (Bradmark)	1.10
ADAGER (Adager)	1.10
ADAGER - Optimized	1.00

Please note that the high value for DBMGR was reportedly due to a bug that has since been corrected. Although not intended to be a product recommendation list, this study does point out the efficiencies of some tools available. How much will doing this help? Again, IT DEPENDS! Suffice it to say that you really should try packing a detail set on your primary application database to see.

You may want to check with your programmer, database designer, or software supplier for advice on which sets would be best to try. The author has seen a varying number of different results from ok to fabulous. The author commonly receives reports of batch job run times being decreased considerably after a set reorganization.

But how to reorganize? There are a couple of ways. Of course each of them are a time/money juggling act. Detail reorganizations are an expensive venture. You may use the above mentioned tools as well as the good ole' HP supported way of performing a DBUNLOAD (to tape...aargh!), and a DBLOAD. Refer to

1 David Greer, *Detail Dataset Packing.* (What's Up Doc... Special Edition, December, 1986), pp.61-64

HP's IMAGE/3000 manual for details on how this is done. Keep in mind that this reorganization is performed on the primary path; see Idea #IM15 for more information on this.

One good thing about investing in one of the third party products is the fact that some of them are able to do this reorganization to disc (assuming space permits). The need for a human operator or robot to hang tapes is therefore eliminated. Automating this process makes reorganization much more feasible.

#IM2 Choose an Optimal Capacity for Master Sets

It must be first said that this idea has been the fuel for much debate. Not whether an optimal capacity should be found, but rather what an optimal capacity is and isn't. The general past consensus has been to make the capacity a prime number. This is because the very nature of primes is conducive to shattering patterns of numbers and thus encouraging uniqueness. Recent studies have shown that this may not be the case at all. There are times when a prime is not the best capacity. As David Merit pointed out:

> "...a prime capacity most often results in a poor distribution, if not the worst distribution, when compared with other capacities in its vicinity."[2]

This is a fairly radical departure from previous dogma on the subject; perhaps heretical to some. David's statement, however, is not a flippant, theoretical one. It was born out of empirical observation (and lots of it). The details of this study are explained in Bradmark's newsletter *Imagery* (July/August 1986).

But why does the capacity matter at all? It is because the hashing algorithm (the formula IMAGE uses to try to uniquely locate records in the set) uses the set capacity as a factor in its equation. A poorly picked master capacity may produce a phenomena called "clustering." This is where many of the entries compete for the same record locations; collisions occur, synonym chains are produced, and data retrieval becomes progressively worse due to the extra disc I/Os incurred. Merit concludes,

> "...a prime number rarely will result in a poor distribution of entries. But there normally are non-prime capacities that will result in better distributions to most prime capacities. In other words, if you're plucking a capacity out of the air, a prime number is best. But if you sample various capacities, the best one probably won't be prime." [3]

There are only a few of options that the author is aware of when it comes to choosing an optimum master set capacity. One way is to simply try different capacities and analyze the resulting distribution of the entries with HOWMESSY, DBGENERAL, or DBLOADNG. This could be done on real databases after hours or on sample data out of a production database. Various capacities could be tried and the resulting synonym patterns recorded. This method is very labor intensive, and unless you have a lot of time, not very

2 David Merit, *Image Performance Myths,* (HP Professional, July, 1987), p. 60
3 Ibid., p. 60

tenable. Or you could simply pick a prime and live with the consequences. Most folks do. However, you probably ought to focus your limited resources on the master datasets with low blocking factors that seem to have the most impact on your application's performance, and just leave the others alone.

The most intriguing way to accomplish a near optimum master capacity is to use the capacity sampling feature in DBGENERAL. This module creates a small sample database with live data from the master set of your choosing. A user-specified amount of data is drawn in to the test dataset. Various capacities are tried with the number of synonyms produced reported out at each level. When tested over a broad range of capacities, this will provide you with a tremendous plot of the best and worst capacities.

#IM3 Select an Optimum Block Size

All input and output to IMAGE sets is done to blocks of records and not individual records themselves. By having an inefficient blocking layout, just as with plain MPE files, disc I/O as well as disc space are wasted. Though IMAGE sets up defaults for block sizes and the number of buffers, there are instances when it may be advantageous to modify those. Consider the following:

> "The IMAGE database system was designed and coded at a time when disc space was much more limited and expensive than it is now. Accordingly, IMAGE optimizes the block size for each dataset to conserve *disc space*, not to maximize throughput and response time."[4]

Though IMAGE will choose a block size that tends to conserve disc space, it is kind enough to allow this value to be overridden. More memory is used with larger blocking factors, but this does not seem to be as much of an issue as it was during the early days of IMAGE. Most people will opt to procure memory to accommodate the performance-conducive features of IMAGE and other MPE functions. If memory and disc space are major issues for you, it may be best to go with IMAGE's defaults.

TurboIMAGE now allows a maximum blocksize of up to 2560 words. Though the Schema processor tends to favor disc space optimization, we can override this tendency. Assuming that you are willing to expend some extra disc space and memory, it may be worth trying.

In pre-TurboIMAGE, there is a limitation as to the size and number of buffers since the buffer area is shared with dataset information, security, etc. In TurboIMAGE, the buffers are contained in a data segment all by themselves. In pre-Turbo, the defaults are generally good enough. In fact, because of the buffer space limitations (pre-Turbo), performance could be hindered for interactive users by increasing the block size due to the then reduced number of buffers available for online users. See *Increasing Blockmax*, page 67, in the *Image/3000 Handbook* for detailed elaboration on this subject.

4 Robert M. Green, *Block That Buffer,* (The IMAGE/3000 Handbook, Wordware, 1987), p. 63

For TurboIMAGE users, it may help performance by increasing the BLOCKMAX and/or the blocking factors of each individual set. It is best to have the blocks be about the same size to optimize the buffer area usage. By having each buffer be similar in size, the total number of buffers in the buffer area is maximized. More buffers are highly desirable in order to maximize physical I/O efficiency. The theory needed to understand and perform the calculations necessary to choose a proper BLOCKMAX and blocking factors is found in the *Image/3000 Handbook*. As an alternative, you could simply utilize the REBLOCK feature of ADAGER or DBGENERAL.

#IM4 Recalculate Blocking Factors When Upgrading From IMAGE to TurboIMAGE

In order to accommodate the extra word necessary for the increase in path lengths, the conversion utility DBCONV, in some instances, is very negligent with resource optimization. If a master set needs to be increased, it is lengthened by one or more sectors. In most cases, this is an over-kill; many times only a few words are necessary. An adjustment of the blocking factors would reduce the disc space wasted by DBCONV's liberal use of this resource.

Memory space for the Data Base Buffers control block is also impacted. This is also due to the fact that DBCONV is unable to perform blocking factor changes. The wasted space in the master records ultimately means wasted memory space in the buffer areas.

Another impact is the fact that since wasted space is occurring in each block of records, we are unable to make the most efficient use of every physical disc I/O. Perhaps one more record in the block could be retrieved if the space had not been wasted.

The answer to the above problems is to select an optimum block size. This is the BLOCKMAX parameter; it is for the entire database. Then individual datasets are set to that block size by specifying your own blocking factors. This is done for every dataset that is blocked less than the BLOCKMAX. ADAGER's REBLOCK function may be used, or you may manually calculate and replace the values via a DBUNLOAD/DBLOAD. Refer to the IMAGE Handbook for detailed specifics on blocking factors.

A very thorough explanation and remedy of this problem appears in an article by David Merit in which he says,

> "It is a sad truth that the databases which end up being grossly inefficient under TurboIMAGE are the ones which were the most efficient under IMAGE because the most efficiently-blocked datasets are those which have little or no free space in their blocks and therefore no room for expansion."[5]

5 David Merit, *Convert to TurboIMAGE in Hours, Not Minutes*, (The Chronicle, February, 1987), p. 34

And also, he sums up by saying in that same article,

> "...in order to optimize disc and memory utilization after conversion, while at the same time correcting long-standing under-blocking by DBSCHEMA, you should spend a little of your own time...tuning things up."[5]

#IM5 Choose an Optimal Number of Image Buffers

IMAGE has one buffer pool that is in common for all users of that database. These buffers are the bridge between all dataset activity and the outside world of disc I/Os. The number of buffers which IMAGE allocates is based on the number of database accessors at a particular time. Thus, the buffer specification list that IMAGE provides by default is:

BUFFSPECS = 8(1/2), 9(3/4), 10(5/6) ... 17(19/120)

For three to four database accessors nine buffers are allocated. If more buffers are allocated, more memory is used. If you have a large number of databases on your system, the default might be ok. However, for a few large databases you will most certainly want to increase the buffers as explained below. If you are in a pre-Turbo environment, you will need to be careful as an IMAGE error 62 may result due to the buffer area filling up (DBCB).

There are a few problems associated with the default number of buffers. As James Overman has pointed out, these concerns are:

> "1) The complexity of larger databases quickly requires more buffers to properly maintain chain pointers;
>
> 2) Competition among users for buffers rapidly disposes of a user's current buffers;
>
> 3) The occasional expansion of the buffer area (when a new user opens the database) stops all users and invokes the memory manager to get additional space which may require swapping to and from the disc."[6]

To help overcome these shortcomings of the default buffer specifications, it is generally recommended to increase the number of buffers (via DBUTIL) to something like:

SET BUFFSPECS = 24 (1/120)

---

6 James S. Overman, *Improving Application Performance Without Changing Your Programs,* (Unpublished paper given to the author, 1986), p. 12

This will allocate twenty four buffers for any number of database accessors from 1 to 120. It may be advisable in some instances to increase the number of buffers to greater than twenty-four. Though the main expense is memory, the benefit could be substantial. Consider the following:

> "This single enhancement of TurboIMAGE (buffer expansion area), while not especially emphasized to date, is probably the *most important* performance enhancement (and possibly the only) that large and heavy IMAGE processing shops will enjoy."[7]

This is one way to gain a margin of improvement in IMAGE performance.

#IM6 Upgrade (downgrade?) to TurboIMAGE

TurboIMAGE (IMAGE-C) introduced a few fairly radical improvements over its predecessor, IMAGE-B. It allows for much larger chain lengths, buffer sizes, etc. It is considered to be multi-threaded under some circumstances; that is, multiple users may have access to the database at one time. Refer to the TurboIMAGE supplement in the *IMAGE/3000 Handbook* for a great overview of the structural enhancements and performance improvements.

Note what the TurboIMAGE Supplement to the *IMAGE/3000 Handbook* says in regard to the issue of single and multi-threading of certain intrinsic calls:

> "DBPUT and DBDELETE, and DBOPEN and DBCLOSE when adjusting the number of buffers, are still single-threaded in the sense that no other intrinsic can acquire buffers while any of these is executing. But, the intrinsics which do not use buffers are not impeded at all by DBPUTs and DBDELETEs. Even most modes of DBGET and DBUPDATE are semi-concurrent with each other: they lock the DBB (database buffer area) only to acquire a buffer, not while doing I/O - a significant increase in concurrency with TurboIMAGE."[8]

What about the performance improvements of TurboIMAGE? It is the author's experience that most sites which upgrade to TurboIMAGE view it as a downgrade in terms of performance. Few exceptions have been noted. Of course if the capacity changes were needed, then it may be viewed as an upgrade. One of the reasons for the perceived degradation (or at least a break even) is due to the drastic increase in memory requirements. If you're not careful when converting from IMAGE-B, disc and memory waste will occur due to the inefficiencies of the DBCONV program supplied by HP (the author has heard rumor that the DBCONV supplied in V-MIT does not commit these atrocities). It is this author's opinion that David Merit's article should be required reading prior to upgrading to TurboIMAGE. In the article, he states,

> "It is more than likely that an additional megabyte of memory will be required to properly support TurboIMAGE, unless few users are accessing the database."[9]

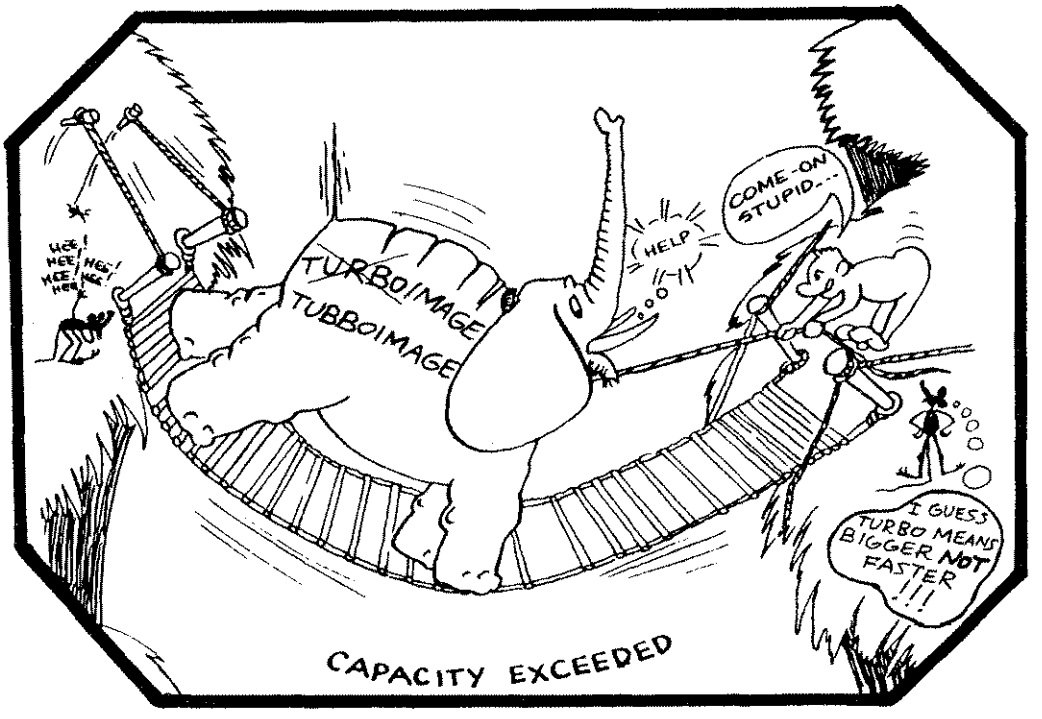
---

7 R.E. Van Valkenburgh, *Improving Your Performance,* (INTERACT, September, 1987), p. 42

8 Dennis Heidner and Marguerite Russell, *TurboIMAGE Supplement,* (The IMAGE/3000 Handbook, 1987), p. 16

9 cf. *Convert to TurboIMAGE in Hours, not Minutes,* op. cit., p. 34

The author has experienced an increase in memory needed for many sites which have installed TurboIMAGE. The new requirement would be masked on systems which were memory fat. In many cases, this "feature" of TurboIMAGE has pushed many individuals beyond the "knee" in the performance curve. The author has heard reports of an increase in TurboIMAGE's CPU requirements by as much as 5 to 10 percent. Any gain which might have been realized by the buffer area expansion, partial multi-threading, AUTODEFER, etc. may be shrouded by a memory shortage and a greater CPU utilization.



In summary, upgrading to TurboIMAGE ought to be approached cautiously. It is advisable to review its features as well as potential shortcomings prior to upgrading and to have a plan in place to optimize as much as possible. However, the author was recently told by an HP SE that upgrading to TurboIMAGE is necessary in order to remain on HP's software support. The TurboIMAGE Supplement in the *IMAGE/3000 Handbook* will provide you with good information regarding most aspects of TurboIMAGE.

#IM7 Utilize AUTODEFER Mode When Appropriate

Normally, IMAGE writes modified (dirty) buffers to disc immediately for PUTs and DELETEs. Files on disc are always kept current with the user modifications as represented by the changed buffers. This is non-output deferred mode.

When output deferred mode is utilized, IMAGE only writes dirty buffers to disc when absolutely necessary. The *IMAGE/3000 Handbook* describes this mode as "taking the brakes off." Brian Duncombe reported that the amount of disc accesses required for performing certain tasks could be reduced by a factor of two in some cases.[10]

Pre-Turbo users are able to utilize this feature with one accessor at a time (DBOBEN mode 3). TurboIMAGE opened it up for multiple users. It is referred to as AUTODEFER in TurboIMAGE and enabled with DBUTIL as follows:

ENABLE <database name> FOR AUTODEFER

The downside of using this feature is that database consistency and integrity are sure to be compromised in the event of a system interrupt. You **WILL** have to recover and rebuild the database if the program your running or system fails during use of output deferred. As you might have guessed, it is not a sound choice for online activity. If used for interactive users, it would be practically non-optional to have user logging enabled. It is therefore, best utilized for batch jobs executing after a full backup. Roll-forward recovery would be utilized in the event of a breakdown for programs without checkpoint recovery.

#IM8 Split Large Databases into Multiple Smaller Ones

Often times one large database can be an impediment to overall system performance even though primary resource utilizations may be moderate. This is because IMAGE is generally considered to be a serial resource. This is more true for pre-TurboIMAGE systems since these systems are considered to be completely single-threaded. That is, it is simply another area in which a queue of users will occur; each user will be waiting to get at the single-threaded resource. The concurrency feature of TurboIMAGE for serial reads and non-structural updates may not provide the breakthrough needed in your environment (though some sites have reported a moderate performance improvement - see #IM7 above).

A recent ad illustrates the serial nature of IMAGE quite nicely by portraying a merry-go-round with only one horse and dozens of children waiting in line. Such is a single IMAGE database.

By creating smaller, multiple bases around some logical breaking points, the single resource will now be many. One might create divisions around alphabetic breaks such as all customers with names from A to H will be in BASEA database. Obviously some program changes will be required; this being the one major drawback to implementing the multiple database idea.

---

10 Brian Duncombe, *Performance Self-Analysis,* (Proceedings of the HP 3000 IUG Amsterdam Conference, 1985), p. 829

It should be noted that though providing a performance gain due to breaking the single database monopoly, disc space and memory overhead will increase. Also, since TurboIMAGE now provides one DBU control block per user/per database, memory requirements can sky-rocket. So goes the never-ending resource juggling act! Just don't get too carried away with splitting databases.

#IM9 Perform Some Maintenance Activities After Hours

Because of the tremendous overhead involved in some IMAGE processing, certain activities should be performed after peak (especially online) activity periods. Three especially notorious examples would be DBPUTs and DBDELETEs to sorted chains (lots of shuffling involved), plain DBDELETEs, as well as QUERY reporting activity. I/O overhead can be astronomical, not to mention the resource monopoly contention (remember, IMAGE is a serial resource for DBDELETEs and DBPUTs) involved.

#IM10 Spread IMAGE Sets Across Discs to Minimize Disc Drive Contention

Refer to Idea #IO13 and #IO14 in Chapter Three of this book for some explanation as to why spreading master and detail sets across drives will help performance. Since much of the time master and detail sets interact back-and-forth, the author has found that placing related sets on alternate drives provides, in many cases, a performance benefit. This is due to reduced head contention.

If both related master and detail sets reside on the same drive, then the armature mechanism of the disc must be constantly repositioned during chained database activity. Spreading sets minimizes this. This is especially true for TurboIMAGE due to actual concurrent access which was not the case with IMAGE. Your database designer or technical support representative should be able to help you make a decision on which sets would be good candidates for implementation of this idea.

Sets may be moved to specific devices with the "RESTORE;DEV=<ldev #>" command, DBUTIL MOVE command, MPEX (VESOFT), ADAGER (Adager), DBGENERAL (Bradmark), or DBTUNE (HiComp).

#IM11 Disable QUERY's Locking Mechanism

When QUERY is run along with interactive users, it can quickly get access to the database and lock it, effectively stalling other users. Issue the SET LOCKOPTION=OFF within QUERY when it is only being run for reporting purposes.

IMAGE DESIGN TIPS

The following design tips will be presented in "bullet" fashion. That is, only minimal discussion (if any) will be present since the aim of this book is to help you, non-intrusively, solve your performance problems. End users are many times at the mercy of software houses and have little control over some of the ideas listed for performance improvement from the database design standpoint.

If you are involved in database design or restructuring you will want to explore each of these ideas further. Chapter 18 in the *IMAGE/3000 Handbook* includes thirty-nine valuable tips for database engineering.

#IM12 Forbid Serial Searches of Any Kind for Online Activity

Interactive user applications should perform chained reads only (except for very small serial scans). Chained access is fast and is intended for quick response time situations.

#IM13 Create Paths for Frequently Accessed Items

#IM14 Minimize the Number of Paths in the Most Actively Accessed Sets

#IM15 Put Much Thought Into Which Path is the Primary One

If you don't specify the primary path, IMAGE will, by default designate the first unsorted path, as seen in the schema, as the primary one. This is important since detail set reorganizations are performed along the primary path. It is generally recommended, therefore, to make the most actively accessed path the primary one.

#IM16 Consider Utilizing DBUPDATE in Lieu of DBPUT and DBDELETE

Robert Green noted that,

> "DBPUT and DBDELETE take 5 to 20 times longer per call than DBUPDATE, because DBUPDATE is not allowed to change "critical" fields (search items or sort items). That is, it cannot make "structural" changes to the data."[11]

Taking time at the design stage to figure ways to implement DBUPDATE has obvious benefits. One benefit is that with TurboIMAGE DBUPDATES are multi-threaded; DBPUTs and DBDELETES are not.

---

11 Robert Green, *HP 3000: Optimizing Batch Jobs,* (SCRUG Letter, May, 1981), P. 24

#IM17 Utilize Integer Keys when Appropriate

Many database designers have avoided use of integer keys (data types I, J, K, and R) due to their volatile nature if not implemented properly. The author knows of only one sure way to implement integer keys with no down-side; the upside is potentially great. Consider the following,

> "As long as the value of a J2 key does not exceed the capacity of the master set, the record number of an entry will equal its key value. Several wonderful things happen as a result:
>
> 1) The master set will contain no synonyms
>
> 2) No pad space is required in the set (no disc space is wasted)
>
> 3) Master entries with J2 keys can be batch-loaded many times faster than hashed masters
>
> 4) Since the physical position of an entry is its key value you can physically order the entries in any sequence desired.[12]

The risk in using integer keys occurs when the set's capacity falls below the value of the highest key value. As long as this one caveat is observed, integer keys can help insure optimal database performance.

#IM18 Use Byte Keys Unless Integer Ones are Appropriate

IMAGE uses different algorithms for integer and byte keys. It is generally recommended to utilize byte keys (data types X, U, and Z) since they produce the best hashing results. Note #IM17 for an exception to this.

#IM19 Minimize the Use of Sort Chains

Sorted paths have received bad press over the years and rightly so because of the rampant abuses that have prevailed by unknowledgeable users. They have their place with specific cautions kept in mind, but generally speaking it is advisable to avoid them if there are other alternatives. The section entitled *Can't I Ever Use Sorted Paths?* in the *IMAGE/3000 Handbook* ought to be thoroughly studied prior to implementing or shunning the use of this special feature of IMAGE.

---

12 Mark Trasko, *The Future of Database Technology,* (SuperGroup Association Magazine, July, 1986), p.32

IMAGE PROGRAMMING CONSIDERATIONS

As stated in the IMAGE design section above, discussion of programming ideas will be limited. This is partially due to the fact that virtually every idea has been discussed in other papers, and that the scope of this book does not emphasize "reinvention." Rather it aims to focus on solving performance problems without re-design and reprogramming.

#IM20 Use DBGET Mode 1 to Initialize Field Lists

This call will fail. However, The set list will be initialized properly for future DBGET calls.

#IM21 Minimize DBOPENs

A DBOPEN consumes a tremendous amount of resources to execute. This is true primarily for the first accessor of the database; especially if ILR (Intrinsic Level Recovery) is enabled. So much so that David Greer says,

> "...DBOPEN should get four stars as one of the all-time great consumers of HP3000 resources."[13]

For this reason, it has been suggested that a dummy program perform DBOPENs to all application databases in the morning to minimize the initial shock of the first DBOPEN. However, this problem is not as bad with TurboIMAGE especially if multiple buffer levels are not specified.

#IM22 Use the "*" in IMAGE Lists Whenever Possible

Using the "*" after an initial DBGET (whole list - worst, partial list - good, "@" - better, mode 1 DBGET - best) will avoid list processing and security checking which are fairly costly operations.

IMAGE PERFORMANCE MONITORING

There are only a few good tools available to provide you with IMAGE efficiency data. However, since IMAGE is part of the whole system and involves the CPU, memory, and disc I/O, other monitoring means mentioned in Chapter Six will provide you with indirect performance information also.

This section primarily involves brief explanations and use of some of the common IMAGE observation tools. Again, you are encouraged to procure one or more of these tools and experiment yourself before you implement any one (or more) of them.

---

13 David Greer, *How Slow is DBOPEN?,* (What's Up, Doc?, Special edition, 1986), p. 66

#IM23 Use DBLOADNG to Monitor Your Databases

DBLOADNG is a contributed library utility that produces a report on a per-database level. A sample of its output is provided and a brief discussion of major report points follows.

```
 Dataset            Capac-          Load  Secon- Max  Block
 #/Name      Type    ity  Entries    Factr daries Blks Factr
 4 EMPMAST    M      487    420      86.2  32.4   44    2
17 COSTING    D     4872   2709      55.6                7


Search    Max     Avg     Std  Expctd   Avg     Elon-  %Ineff
Item      Chain   Chain   Dev  Blocks   Blocks  gation ptrs
EMP-NR     16     9.85   3.68   1.86     2.83    1.52  20.6
EMP-NR      5     1.48    .74   1.10     1.90    1.73  67.6
```

Figure 4.1 - Sample DBLOADNG and HOWMESSY Report

The explanation of each item that follows is primarily derived from the scant DBLOADNG documentation that is available. Better discussions appear in Robelle's HOWMESSY manual and in the *IMAGE/3000 Handbook*. Only the non-obvious items will be briefly discussed.

1) Load Factor - Percentage of the capacity currently in use. The performance of master sets tends to deteriorate as the load factor approaches 100%. Details don't care.

2) Secondaries - Master sets only. Percentage of secondary entries. These are entries which cannot live in their hashed location due to that spot being taken. The lower the number here, the better. In some cases a high number will not hurt. There can, however, be problems without a single secondary.

3) Max Blks - Master sets only. This is the worst case number of contiguous full blocks without a single free entry. Should be low (1-10). If high, it means that IMAGE would have to search through this many blocks in order to find a free place to locate a secondary entry. In figure 4.1, the value of 44 is very bad.

4) Blk Fact - The number of records per physical block. A low blocking factor sometimes explains bad results in the Ineff ptrs and Max Blks columns. With TurboIMAGE, you might want to increase this by raising the BLOCKMAX above the 512 words to say 1024, and by specifying your own blocking factor in the schema.

5) Max Chain - For masters, this is the length of the longest synonym chain (the shorter, the better). Look out for Max Chain = 1 and Load Factor > 50%; this means that no hashing collisions have occurred thus far, indicating non-random hashing and potential problems later. For details, this is the maximum number of entries with the same value for this search item. Big chains can be bothersome.

6) Avg Chain - Average length of a chain. This is calculated by dividing the total number of entries in the set by the number of chains. For masters, this relates to secondaries, it should be between 1.01 to 1.50. Shorter is better. For a detail set, this should be relatively short. If it is too large, you might be better off not having the chain at all.

7) Std Dev - This is the standard deviation of the chain lengths. It shows the variation from the average that is required to encompass 90% of the cases. If small, then most of the chains are close to the Avg Chain length. If large, then many chains are longer or shorter than the average.

8) Expctd Blocks - The average expected blocks per chain. This column shows the average number of blocks each chain would occupy if entries with the same key value were adjacent. This number is used with Avg Blocks to compute Elongation.

9) Avg Blocks - This is the actual average blocks per chain. Since the number of blocks per chain determines the number of disc reads required to traverse the chain, the performance implications are conspicuous.

10) Ineff Ptrs - This is the sum of the pointers to different blocks divided by the total number of forward record pointers. If a pointer points to a different block for the next record in the chain (synonym for masters, chain for details) I/O overhead is incurred. Small is good here.

11) Elongation - This is the actual number of blocks divided by the optimal (expected) number of blocks. A value of 1.0 is perfect. In figure 4.1, the number of I/Os that IMAGE would have to issue is 1.52 times the number if all the detail entries were adjacent to each other on disc.

IM#24 Utilize HOWMESSY to monitor database efficiency

The discussion for DBLOADNG in the above idea is 100 percent applicable for the program HOWMESSY with one exception. HOWMESSY is 10 to 20 times faster than DBLOADNG!

IM#25 Use DBGENERAL's Diagnostics to Track Database Performance

This program (Bradmark) provides quite a bit of diagnostic and performance related information. Below are some of the areas which it monitors.

1) Master set analysis - Graphically shows master set data distribution and primary/secondary analysis

2) Detail set analysis - Chain statistics for each path with a summary similar to DBLOADNG and HOWMESSY but also provides the delete chain length

3) Master dataset capacity sampler - Allows dynamic search for optimum master set capacity values based on percentages of synonyms produced per each capacity tested

4) DBGENERAL's analysis is very useful because it breaks the statistics into different categories by chain length. Since a couple of inefficient long chains mixed in with very efficient short chains will throw a composite statistic. This scenario occurs quite often.

#IM26 Use the IMAGE Profiler

The IMAGE Profiler program provides IMAGE CPU and I/O statistics. It gives a live picture of a database's access. A trace feature will report data regarding each database intrinsic call. This feature incurs a noticeable performance degradation, however. Essentially, this program allows you to:

> "...determine the optimal number of buffers, which datasets and paths are used most frequently, and whether there are locking problems, among other things."[14]

Note that this tool generates a tremendous amount of data very rapidly, thus impacting disc space.

#IM27 Implement OMNIDEX for Rapid Database Searching

This product provides one more structure to an existing IMAGE database. This structure is a searching method that utilizes a binary tree. It boasts of generic retrieval and sorted sequential access, multiple keys in masters, keyword retrieval on text data, and record selection across multiple fields.

Since OMNIDEX indexes every word, a combination of partial values and full values may be searched on and the resultant qualifying records will be retrieved very fast. Reports of its benefit have been no less than very impressive. So much so that HP has implemented OMNIDEX at their Response Centers to index problem reports. This product is available from Dynamic Information Systems (DISC, Denver, Co.). If you utilize large databases with online inquiry, this product may provide you with a tremendous performance improvement.

---

14 cf. The TurboIMAGE Supplement, op. cit., p.35

Chapter 5

Hardware and Configuration Considerations

One of the more obvious ways to gain a greater level of system performance is to obtain faster, more efficient, or a larger quantity of hardware components. Upgrading the CPU, obtaining more memory, or adding another disc drive will in most cases provide some measure of system performance improvement. Though gain may be immediate, the cost is generally great. However, it goes without saying, without proper analysis the wrong hardware component may be upgraded.

It must be kept in mind that in many cases upgrading to a faster CPU may be compared to putting a more powerful engine in an automobile that has been neglected. Due to neglect, the old engine has become rusty in critical parts of the power train. Loose or worn mechanical parts in the transmission require more horsepower to obtain the previous level of acceptable performance. Abusive habits like driving up steep hills more often, keeping the parking brake partially engaged, or attempting to transport heavier payloads in few trips versus more frequent lighter loads, will all place a greater burden on the engine. Since there are many system performance components on the HP 3000 which may be analogous to our inefficient auto, providing more horsepower will provide relief to the immediate situation but will not cure the inefficiencies. If the root problem is not addressed (bad habits), the same dilemma will be faced at a later date with the larger CPU. Pay day will come.

It is generally recommended that hardware procurement be considered only after inefficiencies in the various system performance components are given due attention. The aim of most of the ideas in this book is to rectify those inefficiencies. There will generally come a time in a growing DP environment, however, when hardware upgrades are inevitable. This chapter includes some ways to improve system performance which involve upgrading and properly configuring the various system performance hardware components.

The role that proper system configuration plays in maximizing system performance may not be as obvious as is simply purchasing more/different hardware. But an improper hardware configuration will contribute to system performance degradation. For example, configuring more than four disc drives on one GIC (General I/O Channel) may contribute to an I/O bottleneck (depending

on the intensity of I/O activity). This is loosely analogous to providing too small an exhaust pipe on an engine. Since the goal of an efficient engine is to process (and therefore eliminate) as much fuel (I/Os in our analogy) as the user is demanding at the moment, narrow exhaust manifolds will cause a greater backpressure (disc requests queuing), and will simply degradate the engine's horsepower throughput. While this analogy is limited, it does help illustrate the idea of an improper configuration.

Ideas presented in this chapter will reflect configuration parameters which are generally accepted as optimal. Your experience may be different. Many areas in this chapter are debatable, so it is probably best to experiment and see what works best for your system.

HARDWARE UPGRADE

#HC1 Upgrade the Central Processing Unit (CPU)

Procuring a faster CPU will nearly always provide a noticeable performance improvement unless, of course, the system is I/O or memory bottlenecked. Since a faster CPU, by definition, is able to process more transactions per unit time, an immediate benefit is perceived from the upgrade. Figure 5.1 illustrates the "raw" CPU horsepower of various HP 3000 models as expressed in approximate MIPS (millions of instructions per second, or meaningless indicator of performance!):

Model CPU	Relative Speed
S30, S33	.15
S37	.20
SII, SIII	.25
Micro/3000	.38
S39, S40, S44	.40
S42, S48	.40
S42XP, S52, S58	.52
S64, S68	1.30
S70	1.60

Figure 5.1 - CPU Speed Comparisons (MIPS)

The term "raw" CPU is used because a broader view of each of the models can be taken. It is not often realized, for example, that a series 40 has the same physical CPU as a Series 48. The other differences in the models are the amount of memory and I/O devices which may be configured and whether Disc Caching is included. One must not forget the addition of the fine simulated wood-grain desk that is included with the series 44, 48, and 58! Adding more CPU cache memory (very expensive and fast memory used by the CPU to preload subsequent instructions) and some software changes to a series 42, 48, and 68 produces a series 52, 58, and 70 respectively.

Though perhaps obvious by now, it must be said that upgrading a series 40 to a series 44 will **NOT** provide greater CPU processing power. In reality, increasing the number of ports available for device connection on the series 44 may facilitate performance degradation due to the addition of more terminals and therefore more load.

For a good discussion of the series 58 performance, see the paper by Jim Kramer cited below. Jim performed quite a few benchmarks on the S58 in comparison with other "cousin" machines. His discussion of the S58 (which also applies to the S52 and S42XP) concludes with the following comment:

> "The Series 58 and Series 42XP provide help where it is needed in this era of Disc Caching: CPU and memory capacity. The processor performance increase, coupled with whatever help memory and caching provide, should give a significant boost to the current mid-range machines"[1]

The Micro/3000 and series 70 have provided users with a fairly dramatic improvement in system performance over the series 37 and series 68 respectively. Upgrading to either will usually enhance performance significantly. Note the following comment by Sid Smith explaining the most significant difference between the S68 and S70: the expansion of CPU memory cache (not to be confused with Disc Caching) from eight kilobytes to 128 kilobytes:

> "Expanded cache: this cache is not the same as Disc Caching which is a software product that makes use of main memory. The hardware CPU cache is a higher speed, more expensive memory that is used to store program instructions. The system will anticipate the next set of program instructions and load them into the CPU cache. The CPU will first search its cache for the next instruction before taking the long, arduous journey to main memory for the instruction. Here is an analogy: you are painting a mural, and most of your paint is in the next room. If you were a series 48, you would walk to the next room to get the paint when you need it. If you were a series 68, you would hire an 8-year-old to anticipate your needs and get the paint for you. Of course, the 8-year-old has a short memory, so can only get the right paint 95% of the time. For the other 5%, you must walk to the other room yourself to get the paint. If you were a series 70, you would hire a more expensive 16-year-old that could anticipate your needs an average of 98% of the time. Now, the difference between 95% and 98% may not seem like a lot until you consider that you could be requesting a new paint several hundred thousand times a second! One could easily exhaust one's resources in a hurry."[2]

#HC2 Distribute Major Applications Among Separate CPUs

This idea is a key thrust of distributed data processing. If a particular application (could be a department like programming, accounting, etc.) is requiring a majority of the CPU's attention, off-loading that application to a separate system will provide the remaining users relief.

---

1 Jim Kramer, *Series 58 Performance,* Proceedings of the Interex HP 3000 Madrid Conference, 1986), p. 557

2 Sid Smith, *The Series 70 - What is it? What can you expect from it?* (An unpublished paper given to the author, 1986)

One benefit is the ability to tune a particular machine to the idiosyncrasies of singular applications. These tuning issues might be file placement, CPU versus I/O intensity, batch versus interactive, etc. Some processing needs are mutually exclusive with others, and it makes good sense under some circumstances to split off applications for customized CPU attention.

However, the other issues involved must be noted when considering multiple CPUs. These include:

1) Replication of hardware and software maintenance costs (two copies of MPE)

2) Double the overhead for MPE to operate (given the fact that there are now two systems)

3) Multiple system backups

4) Redundancy of hardware (tape drives might be shared...check support issues first!)

In an article by the staff of Robelle Consulting, this issue is addressed as "Use One CPU Per Problem". They say,

> "Don't try to crowd everything onto one computer. Instead, use a separate CPU for each major application, or give each department its own machine. That way you make each application independent of the problems in other applications. If the payroll application is a hog, there's no reason for the accounting users to suffer!"[3]

#HC3 Add More Main Memory

This recommendation would only be valid, of course, if it was determined that there was a memory shortage. There have been many cases when a dramatic performance improvement resulted by adding memory to systems which exhibited sometimes even a minor need for memory.

Figure 5.2 illustrates the resultant transaction throughput for various memory configurations on a series 70 HP 3000. A severe memory load was induced on the system by 25 simulated interactive sessions and two batch jobs. Each process requested large amounts of stack and extra data segment space. Starting out with 2mb of memory (absurdly low for a series 70, but quite illustrative of memory trauma!) the applications were rerun each time with the addition of 1mb of memory, up to 9mb total. Notice the fairly dramatic increase in transactions completed as the "knee in the curve" is approached.

3 Shumko, Green, Greer, *What if...You Didn't Wait For Spectrum?* (HP Professional, July, 1987), p. 82

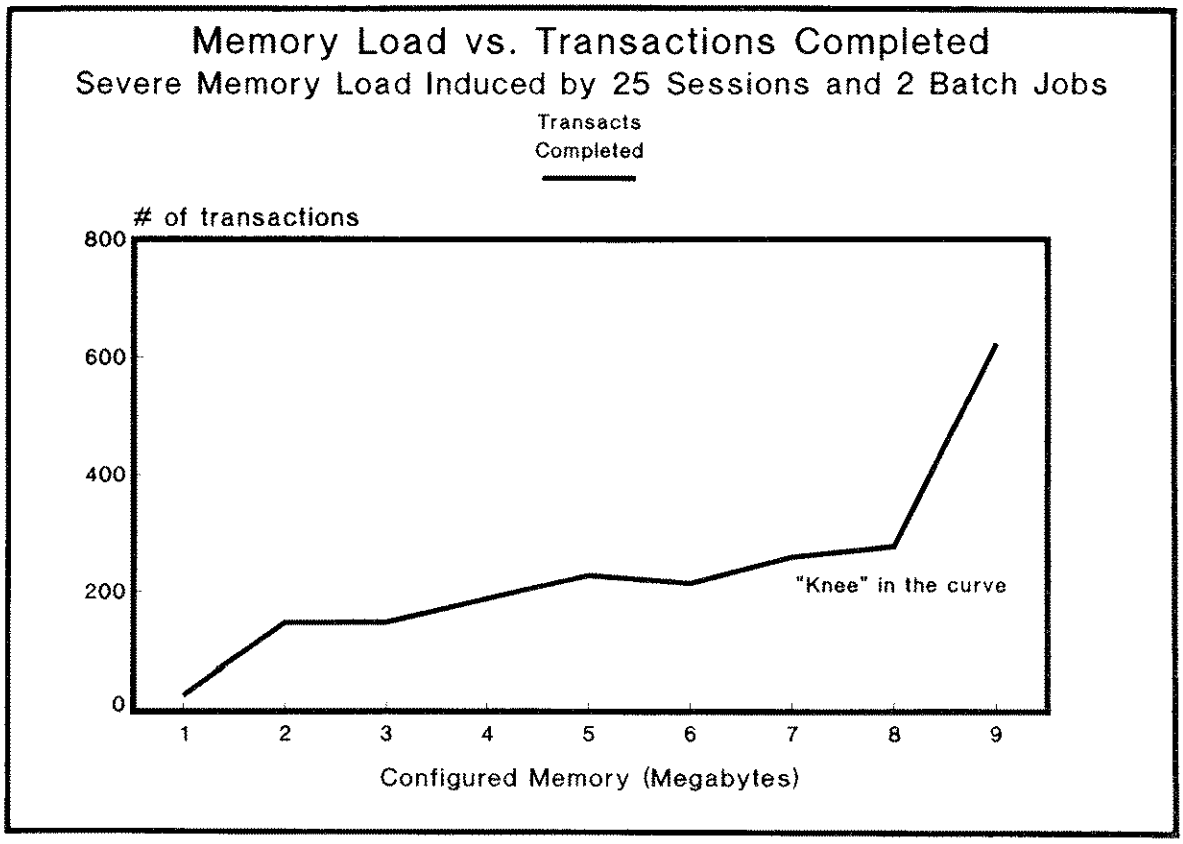


Figure 5.2 - Series 70 Memory vs. Transactions

Memory shortage is relatively easy to diagnose in many cases. Listed below are some indicators of memory pressure. Monitoring tools which provide this information are specified next to each indicator (see Chapter 6 for a description of each tool):

1) Low percentage memory freespace found (OPT, Sysview, Probe)

2) High Memory Access Manager (MAM) I/O rates (OPT, Sysview, Probe) - Be careful of Disc Caching I/Os skewing these numbers (Pre U-MIT); more than 10% of total I/Os due to MAM I/Os is one shortage indicator

3) Low percentage of recoverable overlay candidates (OPT, Sysview, Probe)

4) Consistent Global Garbage collection (OPT, Sysview, Surveyor, Probe); remember to disable Disc Caching in order to see Garbage Collection if you have more than two megabytes of memory

5) Fast memory clock cycle rate (OPT, Sysview, Surveyor, Probe)

6) Large percentage of CPU pause for memory management I/Os (OPT, Sysview, Probe, Surveyor)

7) If you upgraded to TurboIMAGE and have many DBOPENs issued, your memory resource has been impacted. The space for DBUs and XDSs has skyrocketed. Ask your technical support representative about a patch for this problem.

Once diagnosed as a memory problem, the addition of memory will usually reward you with a very noticeable improvement in system performance. The author highly recommends "over-buying" on memory as opposed to "under-buying." The reasons for this are so that Disc Caching will be able to have plenty of usable space and to provide a margin of growth in this resource. See Idea #HC14 for another reason to be liberal in memory acquisition. Also memory requirements for stack space have edged steadily upwards with each new MPE release. Having excess memory on hand will allow you to be prepared for each new MPE version. As Jim Dowling has said,

> "At best, inadequate memory will cancel the positive effects of a software improvement. At worst, it can cause a performance degradation."[4]

#HC4 Obtain Another Disc Drive

Since disc drive accesses involve electro-mechanical activity, I/Os are a common resource found in short supply. Adding disc drives onto systems which generate a fair amount of I/O will usually provide some gain in performance. This gain will be most noticeable on systems which have a single drive to begin with (single drive systems should not be sold!). Adding a second disc will contribute at least the following benefits:

1) System directory accesses are frequent and will have more dedicated attention

2) Spreading virtual memory accesses

3) Balancing the I/O accesses to user files in general

4) Allowing for the placement of complementing files (IMAGE master and detail sets on separate drives, for example)

5) Allowing spoolfiles to be spread

Though increased file storage space is usually the motive in purchasing another disc drive, the benefit of being able to process more disc I/Os should also be considered.

4 Jim Dowling, *Tuning Tips,* (NEWS, EMC^2 Corp. Newsletter, April, 1987)

#HC5 Upgrade to Faster Disc Drives

While adding any disc drive will provide an increase in I/O throughput (assuming a proper GIC load), upgrading existing drives to faster ones is also a way to improve system performance. Better technology has decreased many of the times associated with disc activity.

There are a number of different components that affect how a disc drive performs. These factors, in turn, have implications on overall system performance. The following figure 5.3[5] provides these time components for some common disc drives. All times are in milliseconds.

Model	O/H	Seek	Latency	Xfer	Contr. Cache	Avg. Tran. Time	I/Os Per Sec.
7914	4.0	28.1	8.3	1.2		41.6	24.0
7920	1.1	25.0	8.3	1.0		35.4	28.3
7925	1.1	25.0	11.1	1.0		38.2	26.2
7933	3.5	24.0	11.1	1.0		39.6	25.3
7933XP	4.5	24.0	11.1	1.0	-16.6*	24.0	41.7
7936	1.0	20.5	8.3	1.0		30.8	32.5
7936XP	1.2	20.5	8.3	1.0	-15.9*	15.1	66.2
7945	10.1	30.0	8.3	2.0		50.4	19.8
795X	3.0	29.0	8.3	1.2		41.5	24.1

Figure 5.3 - Disc Drive Statistics

NOTE: References to the 7933 include the 7935; 7936 includes 7937. * - These times represent the lower number of seeks due to the inclusion of disc controller caching based on a read hit rate of 70% and a read percentage of 70%.

After looking at the various time components associated with disc I/O transactions (most of which are hardware dependant) and then comparing the number of I/Os per second possible by each drive, you will see how much (or little) different ones might affect disc I/O performance. Keep in mind that the amount a faster drive will contribute to overall system performance is very site specific. It really depends on many things which are discovered best in a performance evaluation. You will also want to factor in hardware support costs.

5 Hewlett-Packard, *Disc Performance Reference Guide for HP 3000 Systems,* July 1987, p. 2-2

#HC6 Upgrade Slave Disc Drives to Masters

Some of the earlier disc drives (most notably the 7920 and 7925) had the option of being a master drive (its own controller) or a slave drive (utilizes the controller of the associated master). The implication of having slave drives is that only one I/O request can be serviced at a time for all the slaves and their master. Requests for files residing on separate slave drives with a common master would be queued up. Only one I/O could "come down the pipeline" at a given time while the target drive held the I/O channel (series IIIs with overlapped seeks are an exception to this). Providing individual controllers for each drive, and thus making them masters, is actually providing that many new pipelines for disc requests to be serviced.

If you are not sure which drives are masters and slaves, contact your hardware support representative and have them explain to you which ones are which. If you have any slave drives at all, upgrading them will, in all probability, surprise you as to the performance improvement that results. It is worth considering trading in these drives for newer technology disc drives.

#HC7 Add General I/O Channels (GICs) if You are GIC Limited

A general I/O channel (GIC) provides a pathway for I/O requests between the central bus (Inter-Module Bus - IMB) and peripheral devices which utilize HP Interface Bus (HPIB) communication. Consequently, a performance bottleneck will result with I/O requests if there are not enough GICS to support the number of desired HPIB devices and the desired I/O rates. There are a few different rules to keep in mind when configuring devices onto GICs. For example, no more than two high speed device types are supported on one GIC. The "speed" is not so much how fast the data travels, but rather how fast the device responds to a poll. Check with HP to determine which devices are slow or high speed. Almost all new peripherals are high speed devices.

There are a couple performance-critical issues when considering GICS. First, disc devices which are capable of Rotational Position Sensing (RPS) should not share the same GIC as non-RPS devices (non-RPS discs, tapes, printers, etc.) to obtain the maximum value of RPS (see Idea #IO11 for an explanation of RPS). Second, no more than four disc drives should share a single GIC. I/O throughput is slightly diminished at three per drive, a little bit more at four, but most certainly at five drives per GIC an I/O traffic jam is experienced if the I/O request rate is high enough. Consider the following statement from HP:

> "The importance of spreading peripherals is evident in situations where disc drives share a channel with tape drives or printers and all are utilized simultaneously. The tape drives and printers have a tendency to transfer larger blocks of data which require longer periods of channel time, thus making it increasingly difficult for the disc to connect to the channel when necessary. Even with the disc performance enhancement tools available today, this bottleneck could still occur because the non-disc devices monopolize the channel. By placing tape drives and printers on separate channels from disc drives, an environment is created that is more conducive to performance."[6]

6 Hewlett-Packard, *Disc Performance Reference Guide for HP 3000 Systems,* July, 1987, p. 6-8

Of course, you would not add a channel unless it was determined that your system was GIC limited. Nevertheless, adding GICs for the purpose of utilizing Disc RPS most efficiently or simply to provide an efficient path for I/Os is another way to help improve system performance.

#HC8 Add an Inter-Module Bus (IMB)

Since a single IMB has the capability to pass a finite amount of data through it, the addition of a second one (or even a third one in very large configurations) may serve to alleviate a "global" I/O bottleneck. Second and third IMBs are only available for series 6X and 70 machines. Of course the other reason for adding an IMB is to accommodate additional high speed devices. Since an IMB is limited to a bandwidth of 2.77 megabits/second, and since a GIC is capable of passing approximately 1 megabit/second, HP will not officially support three high speed GICS on an IMB (3 * 1 mb/sec > 2.77 mb/sec).

But when is an additional IMB justified in terms of performance (they are not cheap)? In talking with performance specialists and making some empirical observations, it seems that after about the fifth or sixth disc drive I/O rates begin to fall off, and that another IMB could be justified. This assumes a fair amount of activity, however. It really does boil down to how many I/Os are needed per second. The following figure 5.4[7] will illustrate the varying matrix of I/Os possible versus configured hardware:

DISC DRIVES	GICs	IMBs	I/Os PER SEC
1	1	1	22
2	1	1	44
2	2	1	44
3	1	1	53
3	2	1	66
4	1	1	60
4	2	1	88*
4	2	2	88*
4	4	2	88*
5	2	1	94*
5	3	2	110*
6	2	1	104*
6	3	2	132*

Figure 5.4 - I/Os Possible Under Varying Configurations

7 Volz Associates, *Performance Maximization Seminar Notebook* (Reprinted by permission)

Note the rates marked with an "*". In reality, these rates are rarely attained due to an eventual CPU speed limitation. The best way to determine whether an IMB will help throughput is to try it! You might ask HP to allow a "try-before-buy." Other ways would include either having a performance specialist do a study, or you decide how many I/Os per second your application would optimally require (not quick nor easy), and then compare with the preceding I/O rate chart.

#HC9 Upgrade from ADCCs to ATPs

HP 3000 models which appeared directly after the Series III began to use Asynchronous Data Communication Controllers (ADCCs) to handle serial communications. Though an improvement over its predecessor (ATCs), there was still a fair amount of CPU overhead associated with RS-232 communication. When a character was sent from a terminal, the ADCC passed it through to the CPU which in turn sent it back through the ADCC to the terminal. Thus much of the burden of terminal datacomm was still placed on the CPU. This is especially true for serial printers which send a CPU interrupt for every character printed.

Newer systems (from S64, S37 on) come with Advanced Terminal Processors (ATPs). Each group of ATP ports is controlled by its own microprocessor (CPU), and is essentially a computer within the computer. As a result, the ATP absorbs a tremendous amount of the datacomm burden which would have previously been handled by the main CPU. There are now ATP upgrade kits available for systems which came standard with ADCCs. The author has seen instances where upgrading from ADCCs to ATPs has reduced the amount of time the CPU had spent on the ICS (interrupt management - sometimes seen as overhead) by a fair amount, and thereby freeing up the CPU for user tasks. If at all possible, keep serial printers off ADCCs.

HARDWARE CONFIGURATION OPTIMIZATION

#HC10 Distribute I/O Devices Over as Independent Path as Possible

To minimize I/O contention between the various I/O devices on the system (this is referencing HPIB devices more than terminal devices), a good strategy would be to provide as independent as possible path for each one. In order to do this, a few rules must be kept in mind. Before actually presenting an I/O distribution plan, we'll look at some rules to keep in mind when configuring systems. Special attention will be given to disc drive I/O since they are the ones which usually provide the majority of our resource bottleneck grief. First, CS80 devices (793x, 791x discs) should be on GICs with each other and not be mixed with non-CS80 (792x discs) ones. This is in order to take advantage of the disc RPS feature (see Idea #IO11). Second, disc drives should be split onto a separate GIC after the addition of a 4th drive.

Now for an I/O distribution plan. Steven Cooper speaks out on hardware cabling and I/O distribution and says,

> "How things are cabled can have a greater impact on performance than just about any other factor. Do not assume that HP cabled you up optimally! Your discs should be spread over all of the IMBs and GICs as they can."[8]

I/O performance will be most optimal if the path that I/Os take are fairly independent. Steve goes on to recommend the following best-to-worst cases regarding device cabling. Given the situation where the system is accessing two separate files the resultant performance will be:

1) Best - if the target files are on two separate IMBs

2) Better - if they are on two separate GICs on the same IMB

3) Good - if on two separate master drives on the same GIC

4) Fair - if the files are spread between a slave drive and its master

5) Worst - if they reside on the same disc

By comparing your system's configuration with the above guidelines and keeping in mind the preceding rules, you will insure an optimal, independent path for I/O.

#HC11 Spread Discs Over GICs Using a Weighting Method

This idea, though not tested personally by the author, looks good on paper. In principle it agrees with the widely accepted rule of balancing I/O activity over GICS. Guy Smith, writing in the Chronicle[9], recommends assigning a loading factor to each HPIB device. Then, keeping other rules in mind (e.g. CS80 devices together, to take advantage of disc RPS), the devices are configured on GICs in such a way that the sum of the loading factors on each GIC are as close as possible.

Guy suggests assigning a load factor to disc drives which is equivalent to their capacity in mega-bytes. He also assigns a factor of 120 to line printers (loading factors for tape drives were not presented). Then, if GICs must be shared by discs and other devices, a sum of all the loads for a GIC should be as nearly the same as those on other GICs. No empirical numbers were given in his study; this idea does warrant further experimentation.

---

8 Steve Cooper, *Another Look at the Struggle Towards Optimal Performance,* (Proceedings of the Interex HP 3000 Madrid Conference 1986), p. 423

9 Guy Smith, *Database Housekeeping and Configuring Tips, (The Chronicle, Nov. 1986), p. 40*

#HC12 Specify an Appropriate Number of "DISC" Classes per Disc Drive

MPE places files (actually, file extents) on the available disc drives with the device class "DISC" in a "round-robin" fashion unless over-ridden with a specific device or class request. If a disc drive has three device class DISCs on it, it will receive three file extents rather than just one.

One goal in performance management is to take advantage of each resource as evenly as possible. By allocating a number of file extents in proportion to a disc drive's capacity, we have a pretty good chance of insuring that all disc drives are utilized evenly. What confuses this, of course, is the fact that though files may be spread evenly, they most certainly will have different chances of being accessed.

If, for example, a majority of active files were residing on one drive in a four drive system, we would probably see a greater amount of I/O activity on that drive while the other three experience much idle time.

Even though we may take pains to specify a proportionate number of device classes, fate may cause very large or very active files to end up on a single drive. This is why the author recommends to follow up this configuration idea with a study of the most active files on the system and then perhaps adjusting the layout once again.

Nevertheless, it is still good practice to specify disc classes according to drive capacity. If you had a 7920(64mb), 2 - 7925s (120 mb each), 2 - 7933s (404 mb each.), and a 7937 (571mb), a good disc class placement might look like this:

```
   LDEV       DEVICE       DEVICE CLASSES
-------------------------------------------------------
     1        HP7920       DISC
     2        HP7925       DISC,DISC
     3        HP7925       DISC,DISC
     4        HP7933       DISC,DISC,DISC,DISC,DISC,DISC
     5        HP7933       DISC,DISC,DISC,DISC,DISC,DISC
     6        HP7937       DISC,DISC,DISC,DISC,DISC,DISC,
                           DISC,DISC,DISC
```

This scheme will cause MPE to place three times the number of file extents on LDEV 5 than on LDEVs 2 and 3. Six times as many files will be placed on LDEVs 4 and 5 as on LDEV 1, and nine times as many on the 7937 when compared to LDEV 1. As was said earlier, MPE counts file extents and not the size of each extent when placing them. Therefore, providing a good disc class layout is foundational, but not the "end all" in file load balancing. Observing disc drive activity (Idea #MO2) and file activity statistics (Idea #MO16) will provide the data we need to fine tune the system's I/O load.

#HC13 On Certain Systems, Keep Device Class DISC Off of the System Disc (LDEV 1)

Within a broad definition of optimizing system performance we might include being able to provide maximum resource availability to the user population with acceptable levels of terminal response times and transaction throughput. Given this goal, many medium to large HP 3000 environments may benefit in more ways than one by implementing this idea. Keeping device class DISC off of the system disc (LDEV1) will be most relevant for systems that have a small disc drive as LDEV1. Since the system directory resides on LDEV1, and since busy systems will tend to have a good deal of directory activity, it makes sense to minimize user I/O to this drive. One suggestion is to utilize the system disc as follows:

1) Place a small to moderate amount of virtual memory on it (4k sectors is the minimum)

2) Delete device class DISC from it

3) Keep one device class SPOOL on it

4) Utilize it as an "archive" drive by placing program files (which act as their own virtual memory) on it; these might include unused or seldom used source code files

5) Put a class of SYSDISC or DISC1 on it to be able to direct user files to it by class name

Though doing the above might improve system performance by insuring maximum directory I/O throughput, the greater reason for utilizing this strategy follows.

The only things worse than a system failure are head crashes, fires, and running out of disc space on LDEV1! This idea will not solve the former two problems but will help minimize the probability of the latter occurring. By removing device class DISC, maximum control is given to you, so that new files (of possibly magnanimous size) will not show up by surprise through the generosity of MPE. The author has suffered through two instances in which attempting to perform a system COLDSTART resulted in a message, "ERROR 326 - OUT OF DISC SPACE ON LDEV 1". The only way out was to perform a RELOAD! This cost many hours of system down time. Needless to say, delivered system performance that day was close to nil. If adequate space is available on the other discs, removing device class DISC from LDEV1 may prove to be the insurance policy you need to insure unimpeded system disc accesses and maximum space availability on that drive.

#HC14 "Hide" Memory as a Reserve for Peak Periods

Tony Engberg has said,

> "At its root, response time is a perception."[10]

Part of the job of system manager is to ensure a fairly even level of system performance. That is, a wide variation in response times will produce unrest in the user population. Since peak periods, where response times rise, are typical at many sites, a resulting dissatisfaction rises for users. One way to ease these tensions is to simply not spoil everyone with fabulous performance during nonpeak periods!

Keeping a reserve of a critical resource like disc space or memory (probably the easiest to stockpile and hide) for peak usage is one way to "magically" improve performance. Of course the system would have to be reconfigured, say during lunch, in the case of increasing memory. This probably would not be practical but saving some of this resource for month end processing might be! Negatively biasing system performance during nonpeak periods will help set user expectations at a certain level. Thus you are able to implement a relatively controlled response time perception.

****** Notes ******

10 Tony Engberg, *Response Time: Defining it and Speeding it Up,* (INTERACT Magazine, April, 1986), p. 48

Chapter 6

Taking Your System's "Pulse"

Imagine for a moment that you are a health care professional. You would have received training not only in prescribing ways to cure illness and disease but also in accurately reading symptoms. Consequently when examining a patient, you read temperatures, check blood pressure, listen to the heart and respiratory tract, ask questions, and take pulse readings.

All of these data points begin to come together in your mind and provide you with an initial "gut feel" for what some of the possible causes of the patient's troubles might be. You take accurate measurements, and they are valuable. For example, if a temperature reading was 105° F, regardless of the actual cause of the fever, you would probably pursue an immediate course to lower it. But if none of the symptoms were excessive, you might rely a bit more on the combination of the overall symptoms together. This can be subjective. Nonetheless you would know that acquiring this feel for your patient has much value in helping you expedite a cure. Monitoring system performance is similar to this illustration in many ways. A mixture of subjective "feel" and objective data. Quite frankly, it doesn't matter how great your measurement tools say the system is performing if the user population is complaining about response time. The data is important, but it isn't the whole picture. Taking your system's pulse demands that you have accurate symptomatic data, but you must also spend time acquiring a subjective "feel" for how your system is operating.

In this section, ideas are presented to assist you in acquiring resource utilization information as well as ones that will help you gain the ability to take your system's "pulse." Some of the ideas might even seem to be silly, but the author (as well as others that the author has worked with) has utilized nearly every one of the ideas presented to assist in diagnosing and rectifying system performance problems.

There are many ways to monitor performance on the HP 3000. Some of the methods presented are free; some are not. It is often true that you get what you pay for. But in the case of performance monitoring, you might be able to get by with some of the public domain or contributed library software to take the measurements you need. The purchase of one good tool would probably be well worth the money since you have the advantage of ongoing support, many features, enhancements, etc. Contributed programs are updated at times, but you are still at the mercy of the author if you get locked in to using that particular tool and then it goes away.

The many ideas for monitoring performance which follow in this chapter will allow you to choose ways which are practical for your environment, budget, skill level, and time. You are encouraged to experiment with especially the "free" ways to monitor various resources. This is perhaps the best way to gain an education in this arena. It is recommended that you obtain demos of various tools to see how they will work in your shop. Nevertheless picking a select group of monitoring methods and integrating them into your routine will provide you with the data you need to make performance improvement decisions. Remember,

> "Performance optimization must be an ongoing effort-not just something you do when users complain about slow computer response. Continuous system monitoring is needed to forecast potential bottlenecks and to produce records on overall system use."[1]

So, in order to equip you to monitor your system on an on-going basis, ideas have been included which range from simple (inexpensive) to comprehensive (expensive). Simple ideas, discussed first, will include MPE commands, and a few sensory observation methods. Next will be contributed library and public domain programs. Products available for purchase will be discussed last. Also note that IMAGE "measuring sticks" are discussed in Chapter Four.

SIMPLE AND INEXPENSIVE MONITORING METHODS

#MO1 Watch the CPU's Activity Lights

CPU activity or register lights are included on all models of the "Classic" (pre-900 series) HP 3000 except for the series 37 and MICRO/3000. The latter ones have alphanumeric display which have little value in determining system activity. On other systems the red or yellow activity lights are useful for gaining a very shallow feel for what is going on in terms of CPU processing. Simply put, if the lights are flashing sporadically, the system is not too busy. If they are on solid (with no let up), then one of three cases prevails: either the system is hung, one process is looping, or the system is extremely busy. In some instances, observing these lights will provide you with one more data point to either confirm or deny a pending performance diagnosis. Watch these lights during interactive and batch times to get a feel for the differing CPU loads they present.

#MO2 Observe Disc Drive Activity Lights

Like #MO1 above, this method is sensory in nature. Simply standing back and looking at all of the disc drive activity lights will provide you with a feel for I/O activity. First, is the activity centered on only one or two drives? You might not have heavily hit files spread evenly across all drives. Second, how does the activity change throughout a typical day? Does the activity shift back and forth from sets of drives to others with varying applications? Some of this is going to occur. What you are looking for here are gross I/O imbalances between disc drives. It is ideal to spread the activity among as many drives as possible.

---

1. Isaac Blake, *System Performance and Optimization,* (Interact, April, 1984), p.47

Note especially how the patterns of drive access change after a system reload. If you do not have a file placement plan then you are at the mercy of MPE in regards to where files go after they are restored; the author has seen many situations where performance changed (better or worse) after a reload. It is therefore advisable that you stick to a methodology in regard to system reloads to preempt this potential problem from occurring. See the section entitled *BALANCING DISC I/O* in Chapter Three for ways to help insure equitable disc I/O activity.

If you have virtual memory configured on just a couple discs, and those drives seem to have an excessive amount of activity (compared to the nonvirtual memory ones), then you may have a memory shortage. Look at other memory problem symptoms to either confirm or deny this.

Try turning Disc Caching on and off for selective drives observing their activity lights. If activity increases dramatically when caching is off, then maybe you will have a new appreciation for this mysterious product!

#MO3 Utilize Jim Dowling's Disc Drive Stethoscope.

In one of Volz Associates' Performance Maximization seminars Jim Dowling spoke of a stethoscope that he used to monitor disc activity. Being somewhat gullible, the author was disappointed to learn that the instrument was merely his hand! He explained that watching activity lights is useful, but one thing that is not obvious from merely blinking lights is missing. The often times radical arm movement inside a disc drive could indicate poor head locality. That is, file placement on each platter might not be optimal. See the discussion in Chapter Three under *BALANCING DISC I/O* for more information here.

#MO4 Use the SHOWJOB and SHOWQ Commands to Check for CPU Overload

One simple way to quickly see whether or not a current slowdown is due to the CPU being swamped is to issue a SHOWJOB or SHOWQ command. These commands incur no memory manager or I/O system overhead. They utilize only CPU resource to accomplish their work. SHOWJOB simply reads and formats the Job Master system table (JMAT). SHOWQ scans the Process Control Block system table (PCB) for its data. If either of the resulting displays "stutter" when listing out their respective information, this may indicate that the CPU is experiencing overload at that time. Don't try to deduce too much from this; suffice to say, at least for that instance, the processor is really working hard. This is useful at times when you don't want to hassle firing up a monitoring tool which would, of course, incur some overhead and take time to load.

#MO5 Utilize the SHOWME Command to Check for Simple Memory and/or I/O Bottlenecks

If you issue a SHOWME command and it stutters or hesitates, this may indicate a memory shortage. The reason for this lies in the fact that the information peculiar to your session will be swapped out of memory into virtual memory (on disc) if the

memory manager is under pressure to make space for more urgent needs. If the SHOWME display does not immediately appear, this will indicate at the very least that your process data was swapped and may also indicate that the I/O system is experiencing a slowdown. Once again, be careful in jumping to conclusions with this information as you should follow up with more intense monitoring methods. This idea is simply one more data point that you may utilize for quick bottleneck diagnosis.

#MO6 Compare Batch Completion Times on STDLIST Output

Two numbers are available at the end of batch job listings. These are CPU seconds and wall time. CPU seconds represent the amount of CPU time that the job needed to complete, while wall time is how long the job took to complete if you timed it with a stopwatch. Keep in mind that on a multiprogramming system time slicing is occurring. That is, many processes share the CPU resource by being allotted a finite amount of its attention. A job that is run with the same input data will have nearly identical CPU time for each run regardless of the wall time. The wall time, however, may change drastically due to the fact that other processes impede that job's progress by being favored with ample CPU time. The job, therefore, will take longer to complete since it is made to wait. These two numbers will help you get a feel for how much wall time (system activity) and CPU time are required for various jobs. Note that CPU seconds and wall time numbers are available for sessions also. They are displayed with the SHOWME command and at the session's conclusion with the BYE command.

The author has found value in keeping a log of batch wall time completion times. This is especially true after you perform a database reorganization. Comparing batch run times is one way to measure the effectiveness of the reorganization.

THE SHOWCACHE COMMAND

The SHOWCACHE command provides a wealth of information regarding system performance particularly in the I/O arena. It is one of the few ways to peek at some internally kept statistics without utilizing a performance tool. It goes without saying that Disc Caching is a purchasable product, and not every system has it. Issuing the STARTCACHE command at an MPE prompt will let you know if caching is enabled on your system. Refer to figure 6.1 for a sample SHOWCACHE display.

#MO7 Use the Percentage of I/Os Eliminated as a Measurement of Disc Caching's Overall Effectiveness

This is one of the author's first data points used in analyzing a system performance problem. The percentage of user I/Os eliminated is really a bottom line number. It tells us whether the CPU and memory being consumed to reduce CPU pause for I/O is really being spent wisely. If the percentage is high then a good deal of physical disc requests (slow) were avoided and thus I/Os needed were satisfied by a hit to a cache domain in memory (fast).

```
DISC    CACHE     READ WRITE          PROCESS                    % OF    CACHE
LDEV  REQUESTS    HIT%  HIT%  READ%   STOPS      K-BYTES       MEMORY  DOMAINS
-------------------------------------------------------------------------------

1      3469543     78    86    74     854562       137           1       25
2      1747890     70    77    64     546903       223           2       15
3      1736494     75    74    66     474936      1022          12       53
4      3383370     87    76    82     567605       454           5       25
5      1577739     74    71    66     440735       151           1       19
18     2202843     80    78    69     495407       152           1        9

Total 14117879     79    78    72    3380148      2139          26      146

57% of user I/Os eliminated.
Data overhead is 59K bytes.
Sequential fetch quantum is 96 sectors.
Random fetch quantum is 96 sectors.
Block on Write = YES.
```

Figure 6.1 - SHOWCACHE Command Display

The author has been repeatedly told by individuals inside and outside of HP that a series 70 should be eliminating from 60 to 70 percent of all physical I/Os on average. Other machines generally experience less. The author uses a figure of approximately 50 to 55 percent as a low threshold. If the total percentage falls below this range then it is questionable whether caching should be enabled. Remember that caching's effectiveness is due to a few things. These criteria are spelled out in Chapter Three under Idea #IO1.

What if this percentage is very low? Then Disc Caching ought to be disabled. Very low might be down in the thirties. The lowest ever observed by the author was twenty-four percent. Needless to say, the performance on that system was in poor shape. But before you disable caching on all drives, take note of the next idea (#MO8).

#MO8 Monitor Disc Caching's Effectiveness on a Drive-By-Drive Basis

Whether the total USER I/Os ELIMINATED is high or not, you will probably want to calculate these numbers for each individual drive. The author wishes that the SHOWCACHE command would summarize the number of user I/Os eliminated for each drive. It is not difficult, however, to calculate it yourself. Referring to figure 6.1, we will figure how effective Disc Caching is for a couple of drives. The formula for doing so is:

% I/Os eliminated = (READ HIT% * READ%)/100

Take drive two for example. The calculation is as follows:

44.8% = (70 * 64)/100

So 44.8 percent of I/O requests which would have been performed to disc were satisfied in memory. This might sound like a fair amount until you look at, for

example, drive four. Its efficiency is 71.3 percent. If you are not short on memory or CPU, you may want to leave caching enabled. If more memory and CPU could help your system, then disabling caching on LDEV two would return some of these two resources back to the system. What we are trying to do is prevent CPU and memory from being unnecessarily spent by not having a reasonable amount of physical I/Os eliminated.

It is recommended that you periodically monitor the efficiency of caching on each drive. SYSVIEW (Carolian) does break out these efficiency numbers for each disc drive. The author has written a very simple program using the COMMAND intrinsic to issue SHOWCACHE commands every five or ten minutes. The listing is looked at each day, and then a determination is made as to how effective caching is performing globally and on a per drive basis.

As you gain a feel for which applications produce certain caching impacts, you will then know which drives to disable it on.

#MO9 Monitor the Quantity of Cache Domains Outstanding

Cache domains proliferate based on a couple of criteria. The first is demand. Disc requests that are not satisfied in memory cause a physical disc request to occur. That physical request retrieves not only the actual data requested but also brings in more data depending on the values of the sequential and random fetch quanta (configured via the CACHECONTROL command - see #IO3 and #IO4 in Chapter Three). This "chunk" of disc data then becomes a cache domain in memory. It may then be utilized to read from and write to in the future.

The number of the domains has a lot to do with how much CPU overhead is incurred. As the number of domains grows, each request for disc data must first be sought for in memory cache domains. If the data is not found, a physical request must be posted anyway. There seems to be a point when it would be better to simply issue a physical request than incur overhead due to searching large numbers of cache domains. Jim Dowling refers to a sluggishness phenomenon that is observed to occur on large systems which have over 1200 or so domains per disc drive outstanding. See the discussion under #IO4 (Chapter Three) for more detail.

Suffice to say that observing the number of cache domains and then correlating this information with other subjective and objective monitoring methods will help you determine how useful caching is at a point in time. For example, you might decide that drive three in figure 6.1 is using more memory (12%) and CPU time to search all the domains on it than is warranted based on its overall efficiency rating (49%). Disabling caching on this LDEV will return 12 percent of your memory back to you and some CPU. It may be worth trying. By locating drives which have the most cache domains and which have the lowest number of I/Os eliminated (remember, READ% * READ HIT%), you can then experiment with disabling this facility on a per-drive basis.

One more thing that affects the number of cache domains is the size of the fetch quanta. The sequential and random quanta specify how much data is brought from disc each time a physical read is performed. The higher the quantum number, the fewer the domains that result. The author has, on a few occasions, altered the size

of a fetch quantum and recorded the resulting percentage of I/Os eliminated. Refer to figure 6.2 for the rest of this discussion.

RANDOM FETCH QUANTUM SIZE	4	16	32	48	64	96
NO. OF CACHE DOMAINS	1993	1042	736	498	375	280
% OF I/Os ELIMINATED	37	*54*	50	49	48	46

Figure 6.2 Disc Caching Domains vs. Effectiveness

In this particular case, the random value was varied and the two discussed values were recorded. Notice how the most efficient value turned out to be the default random fetch value set at system startup time (16). Either increasing or decreasing the size/number of domains from this value produced a poorer overall caching effectiveness. Changing the Random value from 4 to 16 helped improve performance on this large series 70 partially because the number of domains which the CPU had to search decreased.

#MO10 Check the Amount of Memory Consumed by Disc Caching

How much memory should be available for Disc Caching? In the author's opinion, too much would be fine (though diminishing returns will occur at some point). All too often it is this resource that is in short supply, and it is fairly easy to fill up. Since disc I/Os are expensive (you cannot buy more after a certain point) and memory is cheap, why not give caching all that it is able to use? The author routinely sees caching use around 15 to 30 percent of available main memory. If the % OF MEMORY used by Disc Caching is quite low (say under 5 to 10 percent consistently for all drives), you either do not have much demand (rare) or you simply could use more memory. Of course don't base your procurement decision on this reason alone. Monitor this percentage over time and see if it gets higher than what you saw at first; check for other symptoms - see Idea #HC3 in Chapter Five. Perhaps borrowing a mega-byte of memory will give you a clue as to whether or not caching would provide more benefit if more were available.

#MO11 Correlate User Response Times With Performance Data

One way to help identify the cause of a response time problem is to associate various performance statistics with times of poor terminal response. By correlating performance data with data from users whom you have asked to fill in a response time log form, you will begin to associate poor response times with various bottleneck indicators. Figure 6.3 is an example of one such chart. Permission is hereby granted for you to copy this form as needed.

INTERACTIVE USER RESPONSE TIME LOG

Day _____ Date _____ Name _____ Dept _____

Time	Resp	Comments	Mem Clk	CPU Busy B	CPU Busy C	CPU Busy GC	CPU Busy O/H	CPU Pause M	CPU Pause U	CPU Pause M&U

Figure 6.3 User response time log

The following values needed for the form can be taken off just about any performance reporting tool:

Time - Exact time of response time problem entry

Resp - Response time in seconds as noted by user

Comments - Anything unusual; "slow", "burst", etc.

Mem Clk - Memory clock cycle rate

B - CPU busy

C - CPU busy on Disc Caching

GC - CPU busy on global memory garbage collection

O/H - CPU overhead

M - CPU pause for MAM I/O

U - CPU pause for User I/O

M&U - CPU pause for MAM and User I/O requests

Users simply fill in the time of day and approximate response time with comments for the kind of transactions they perform. You will have been running a performance data collection program in the background for that day (SURVEYOR, for example). Later you collect the forms from the users, fill in the performance numbers, and cross match the times on the forms and collection reports. Things like excessive CPU busy, pause for MAM activity, CPU pause for disc, and consistent GLOBAL GARBAGE collection are key indicators of response time hindrances. Watch particularly for CPU overhead and a fast memory clock cycle rate. In SURVEYOR this overhead figure is derived from adding the entire top line of its display together and subtracting from 100. Jim Dowling noticed that when this number exceeded eight percent on one system, user response times deteriorated rapidly.

Consider implementing a full-time monitoring scheme. Continually run a monitoring program that is able to log performance data to a disc file. Backup this log file on a daily or weekly basis. Use this data to fill out the form in figure 6.3. Use it also to explain to management and users why the system was acting the way it was on particular days.

This monitoring method is an excellent way for you to begin to acquire that subjective "feel" as has been discussed. For response time data be sure to explain to users that you are interested in the time it takes from when they press ENTER or RETURN to the time they are able to begin entering data once again.

#MO12 Use the REPORT Command to Monitor Resource Usage

File space, CPU seconds, and physical connect times for all groups and accounts are all kept track of by the system. The REPORT command is one convenient way to get at this information. A quick glance through its output once a week will provide you with a brief summary of who is consuming resources excessively and who is not. Some sites will actually put limits on various resources so that the particular department or user(s) will have to justify (beg for?) a greater allotment.

Since large file space is sometimes proportional to activity, you may want to keep a close watch on accounts that keep an enormous number of files on the system; they may be the ones whom have also consumed a majority of CPU, memory, and I/O resource. Note that there are a few contributed utilities that are a superset of the REPORT command. Most notably are REPORTR and REPORTER.

#MO13 Monitor System Log File Activity

There is a wealth of resource usage and system activity information that virtually goes untouched in many environments. Various events are kept track of depending on which ones you choose. Things like file closes, job terminations, spooling information, etc. are logged to a disc file (do a LISTF LOG#@.PUB.SYS to see them). You can write programs to read and report on them or simply use FCOPY to print them out for a quick, unformatted look. Contributed utilities which analyze these files abound, and there is always LISTLOG5.PUB.SYS provided by HP.

#MO14 Use FREE5 to Monitor Disc Freespace

A fairly extensive discussion on the subject of managing disc space is found in Chapter Two and Appendix B. Suffice to say that the listing produced by running FREE5 will display the various levels of disc fragmentation. Be forewarned that this program will incur a slowdown on most systems due to the fact that it incurs a tremendous amount of I/O activity to the freespace bit map for a period of time. It then calculates the various figures. You may also send the output to a printer by issuing a FILE FREE5OUT;DEV=LP prior to running it. If an excessive amount of fragmentation exists in the lower portion of the display, you may want to schedule some time to compact those discs. Please refer to the two above-mentioned sections for further reading.

CONTRIBUTED/PUBLIC DOMAIN MONITORING TOOLS

#MO15 Monitor Disc Space Usage with LOSTDISC

This little program provides a wealth of information on disc space utilization that is not easily attainable elsewhere. Refer to figure 6.4 for the output of the LOSTDISC program. It is found in the Boeing Tech account in the contributed software library.

```
        Virtual-Memory      Defective          Disc-sectors
LDEV     Size   Used PCNT     Tracks            Total     Free %Use
-------------------------------------------------------------------
  4     30720  12200  39%      0    0          2232204   917155  58%
  3     20480  11000  53%      0    0          1579916   214932  86%
  1     30720  17276  56%      0    0          1579916   981241  37%
  2     30720  14404  46%      0    0          1579916   613602  61%
-------------------------------------------------------------------
     112640  54880  48%      0    0          6971952  2726930  60%

 4063739   58.2% Perm files        8173   0.1% Misc. MPE tables
    7551    0.1% Temp files      112640   1.6% Virtual memory
      55    0.0% Passed files         0   0.0% Deleted tracks
   37240    0.5% Spool files     120813   1.7% Total MPE o/h

      16    0.2% Total lost disc space (in Kilosectors)
```

Figure 6.4 LOSTDISC display

#MO16 Use FILERPT to Identify Heavily Accessed Files

This program, that is also available in the contributed library, utilizes data collected in the system log files to identify files which have had the most records,

blocks, and closings occur. It is able to sort on the basis of any one of these categories, and then display a given percentage of them from the most active down to the least active. A sample listing of its output is shown in figure 6.5. Keep in mind that system logging event number five (file closes) must be enabled before data that FILERPT needs will be available. Another contributed program, DISCIO, performs essentially similar functions as FILERPT. FILERPT will be used for our discussion.

```
BLOCKS PROCESSED   Report v2.0
Mon, AUG 3, 1987, 4:35 PM

FILE NAME                  LDEV    REC COUNT     BLK COUNT   FCLOSE
                                                               COUNT
PARCIDXK.CUTA85.ACCT1         3   23,033,144.   23,033,144.     612
PROCMGMT.MAILDB.HPOFFICE      3    5,863,902.    3,829,021.     496
SORTSCR .SOURCE.TASSESS       1*   1,987,117.    1,987,117       94
RECHIST .PUB    .AUD          4    1,918,988.    1,918,988      355
ARM1DB11.PUB    .TASSESS      3    1,821,807.    1,821,807      183
QTPSCR  .TEST   .TASSESS      2*   3,846,578.      961,456       29
```

Figure 6.5 Sample FILERPT listing

In this particular case the report was sorted by the number of I/O blocks transferred. Keep in mind that the record count (logical I/Os) multiplied by the file's blocking factor equals the block size. So that is why the sixth file listed (QTPSCR) shows a BLK COUNT smaller than the REC COUNT; remember a physical block transfer consists of one or more logical records. Therefore the block count will always be equal to or less than the record count.

One thing to be aware of when running FILERPT on your system is the fact that all the extents for a file may not reside on the same LDEV. Notice the "*" next to two of the files under the LDEV column. This simply means that logical devices changed before the file closed. Perhaps only a small portion of the whole file inhabits the LDEV number shown. This may skew your strategy somewhat. Refer to the strategy listed under *Balancing Disc I/O* in Chapter Three for one way to deal with this problem. Also make sure all databases are closed for the particular log file you are using as input to FILERPT as the I/O statistics are kept in the final FCLOSE of the database.

The primary benefit of this report is to help you identify the most active files. Use either the REC COUNT or BLK COUNT as your measure of who the busy files are as opposed to FCLOSE COUNT. This will serve as your guide for spreading files across your available disc drives. However, one disadvantage is spelled out in the documentation that comes with the program. The problem is the fact that,

> "the number of blocks processed does not equal the number of physical I/Os if one of two cases is true: A rewind on a variable record file sets the number to zero, or files opened with

multi-rec which also have blocks which end on a sector boundary may access multiple blocks in a single I/O, the number is higher than the actual number of physical I/Os."[2]

Keep in mind that for IMAGE files the number of records processed will be equal to the number of blocks processed. This is because we are dealing with MPE file system records and not IMAGE records.

The number of times a file is opened and closed does have a dramatic impact on system performance. Identifying files which are opened and closed excessively will also benefit your overall performance optimization scheme. If, for example, you sort by FCLOSE COUNT and notice that a particular application file is being opened more than would be expected, you might then investigate further and see if perhaps there is a problem with that application.

#MO17 Monitor Your Process Activity with SOO5E

Son Of Overlord, or System Online Observer as some have called it, has been one of the most depended upon tools for monitoring system performance activity. There are many versions and variations of this program, all of which are offsprings of an older program called Overlord (its mistress is called MOO - Mistress of Overlord; no kidding!).

A sample listing of SOO5E is found in figure 6.6. The most common use of the program is that of identifying processes which are hogging the CPU (CPU%). It also provides other helpful process information such as the user name, priority queue (Q), job or session number (J/S#), stack space (STACK), and extra data segment size (XDS).

```
 ** SON OF OVERLORD ** Version 5E **   MPE V/E  G.02.03.  (  60 Sec. delay )
File name                  User name          CPU    CPU %   J/S#   Stack
@.@.@                      @.@                time     .10 Q @      size   Pin
-------------------------- ------------------ ------ ----- - ------ ----- ----
***** 313    31761000
<<   0 Displayed                                                        0    >>
<<  66 Not displayed                                                747300    >>
<<  66 Total active PCBs                                            747300    >>

***** 313    31827446
DSSERVER.NET.SYS           KEN      .REL15        4  1.19 C S23    23996   29
MAILMAN.UTIL.SYS           MAILMAN  .UTIL       165  1.97 D J1      7312   44
SUPRVISR.HPMAIL.SYS        MGR      .HPOFFICE    19   .44 D J11    11936   78
HPWORD.PUB.SYS             CRAIG    .HPWORD       5   .25 C S28    20940  218
HPWORD.PUB.SYS             JANE     .HPWORD       2   .25 C S25    17740  241
DSSERVER.NET.SYS           MGR      .INSTALL      2   .81 C S30    23996  242
VTRXCNTL.PGMS.TEST         MGR      .TEST         1   .87 D J21     9576  254
SOO5E.PUB.SYS              OPERATOR .SYS          1  1.35 D J22    29848  264
<<   8 Displayed                                                    145344    >>
<<  58 Not displayed                                                602596    >>
<<  66 Total active PCBs                                            747940    >>

 Time used:    4.596 CPU sec.;  62.019 Elapsed sec.    7.411% Utilization.
 Changes:      1 Added      1 Deleted                  (  8:50 AM )

 CI{                       } MANAGER .SYS          DELETED.
 CI{LISTF SUR@.@.@,1       } MANAGER .SYS          ADDED.
```

Figure 6.6 - SOO5E Display

2. *FILERPT Documentation,* (Contributed Library Documentation)

Use SOO5E to monitor activity on your system when you desire a quick peek at which programs are CPU intensive, to see why they are stuck (waiting for an event - only available in some versions), and to compare various stack sizes. There are other features available in some versions. Since the documentation is limited, you will have to simply ask someone who has used present or past versions.

#MO18 Monitor System and Process Activity with SURVEYOR

This program is one of the author's favorite performance tools. It contains a wealth of information, is fairly "bullet-proof" in terms of not causing system problems, displays both global, process, memory, and disc I/O information on one screen, and even provides some summary information. One of the best things about SURVEYOR is that it is free...public domain; not contributed library. The author of SURVEYOR wanted it to be public domain, that is, no cost whatsoever. Cheers to the author JAK (whoever that is)! It used to be in the TELESUP account, but due to a bug (fixed in version B.00.05), was pulled from there. The author is not sure if it will be seen in TELESUP again. Write Performance Press for information on how to obtain a copy of this program if it is not available to you.

The Appendix entitled *A System Performance Primer* is actually an introduction to system performance analysis using SURVEYOR. Remember this book is not intended to be a course in system performance but rather to provide you with a plethora of ways to observe it and improve it. Unfortunately the complexity of the subject requires one to know something about it in order to interpret what is going on. You either bring in a consultant or read and study and read and study... That is why Appendix A was included. It is a *very brief* introduction to managing your system's performance and to help you interpret some of the numbers presented in some of the performance monitoring tools available. Some pitfalls common to beginners are also discussed. Keep in mind that it is just an introduction.

#MO19 Utilize TUNER to Monitor Table and Resource Usage

Tuner is a contributed utility that usually resides in the PRV group of the TELESUP account. It is also available on some of the contributed library swap tapes. It is extremely useful for monitoring system table usage, virtual memory, spooling, and approximate memory usage (be careful here).

It is highly recommended that you check table usages from time to time with TUNER or a third party product. The following tables will cause system interrupts if overflowed: SBUFS, SWAP, ICS, and TRL (the PRIMARY and SECONDARY MESSAGE tables will also, though the version of TUNER used by the author does not report them); set them as high as possible (taking into account your configuration and workload). This next set of tables ought to be set high since processing may be slowed and/or processes may abort if enough free entries are not available: CST, XCST, DST, PCB, IOQ, and DSKIO. A good philosophy regarding table settings is to set performance and interrupt sensitive tables high and monitor them.

Table usage should be observed over various types and volumes of processing days. To be safe, you might want to configure the tables such that usage does not

exceed 60 or 70 percent. Configure tables fat unless you are memory skinny. Unless you purchase a tool that checks this data (OPT, SYSVIEW, PROBE), you need TUNER. Figure 6.7 is a typical display of a TUNER report. Check the limited documentation found in DOC.TELESUP for more features of TUNER.

```
TABLE    CONFG. VALUE    MAX IN USE    CUR. IN USE    MAX %    CUR %    SIZE/WDS
DST          4096            252            247        6.2      6.0         4
CST          512             275            275       53.7     53.7         4
XCST         2048            186            185        9.1      9.0         4
PCB          1026            39             39         3.8      3.8        21
IOQ          512 (506)       10             4          2.0       .8        12
DSKIO        900 (885)       17             5          1.9       .6        17
ATPTBUF      642 (642)       8              2          1.2       .3        69
SBUF         15  (13)        0              0         00.0     00.0       129
TRL          374             186            186       49.7     49.7         4
SPREQ        1391(363)       8              4          .6       .3         6
ICS          4096            426                      10.4                  1
CSTBK        308             32             32        10.4     10.4         1
SWAPT        5004            202            190        4.0      3.8         6
JPCNT        1000            6              6          .6       .6         1
VMEM         260096          16220          15092      6.2      5.8        SECTORS
SPOOL        1024000         10596          10596      1.0      1.0        SECTORS
MEMORY       4608            1772           1772      38.5     38.5        K WORDS
IN MEMORY: SYSCODE=469756  USERCODE=370500  SYSDATA=264088  USERDATA=710780
CURRENT # SESSIONS=3   # JOBS=3     MAX # OF SESSIONS=3   # JOBS=3
TIME (IN SECONDS) SINCE START=2      START: 08:10:07  CURRENT: 08:10:10
```

Figure 6.7 - TUNER Display

PURCHASABLE PERFORMANCE MONITORING TOOLS

#MO20 Use OPT/3000 to Monitor System Performance

OPT (Hewlett-Packard) stands for Online Performance Tool. It originally evolved from being an internal laboratory tool to a customer product. The scope of OPT is gargantuan. It has so many screens and features that this writer even gets lost sometimes using it! It has fabulous graphic displays and extensively covers process, memory, CPU, and disc I/O resource utilization. It provides much information, most of which is useful for a person skilled in system performance. One probably never has enough information in diagnosing a performance problem, but many screens in OPT go untouched due to lack of expertise of many users. The Performance class for OPT offered by HP is therefore desireable. The author favors the #CC (CPU usage breakdown) and #CM (CPU/memory statistics) screens for initial diagnosis.

Some of OPT's features are as follows: batch reporting (limited), log file capability, process stack marker tracing (very helpful at times), memory bank contents display (very intriguing), system table usage (graphic), I/O activity, and more.

Figures 6.8 through 6.11 are some of the screens available within OPT. OPT will definitely tell you much of what you need to know but is a bit unwieldy to use if your not familiar with it. A few of OPT's screens are shown below.

```
HP32238A.00.28  OPT/3000 GLOBAL RESOURCE USAGE REPORT                    9:26 AM

ACTIVITY IN CURRENT INTERVAL                                        4.1 SECONDS

                      10   20   30   40   50   60   70   80   90  100%
      MEMORY USAGE MC                CS   SD   DX                  X

         CPU STATE B      BPI                                      IX

                      10   20   30   40   50   60   70   80   90  100 per second
 DISC I/O ACTIVITY SX                                                (0/1/1)
-------------------------------------------------------------------------------
ACTIVITY OVER ALL INTERVALS                                        13.4 MINUTES

                      10   20   30   40   50   60   70   80   90  100%
         CPU STATE B   BPI                                       IXXO

                      10   20   30   40   50   60   70   80   90  100 per second
 DISC I/O ACTIVITY SX                                                (0/1/2)
-------------------------------------------------------------------------------
MEMORY USAGE LEGEND:          | CPU STATE LEGEND:           | DISC ACTIVITY LEGEND:
M Resident MPE C Code segments| B CPU Busy P Pause for I/O  | X Cached I/O
S Stacks       D Data segments| I Idle     G Garbage coll   | S Segment Manager I/O
X Cached Disc Domains         | X Cach Man O Overhead       | U User I/O
```

Figure 6.8 - OPT's Global Resource Usage Display

```
USER SUMMARY REPORT  Filters:   A (all processes)  C (CPU users only)    9:36 AM
                                B (paging)         D (user account)
 PIN    J/S#   USER.ACCOUNT      PROGRAM NAME (command)          CPU  %  Q PRI
-------------------------------------------------------------------------------
  28     S18   MEL.ES             USER PROGRAM FILE               424  0  C 152
  49      J7   MANAGER.SYS        USER PROGRAM FILE             21316  0  D 202
  50      J6   HALEY.SYS          USER PROGRAM FILE               565  0  L 100
  54      J9   MANAGER.SYS        USER PROGRAM FILE             40016  0  D 202
  60     J11   MGR.RSPOOL         USER PROGRAM FILE             56849  0  C 152
  61     J12   MGR.RSPOOL         USER PROGRAM FILE             54975  0  C 152
  62     J11   MGR.RSPOOL         USER PROGRAM FILE             90416  0  C 152
  63     J12   MGR.RSPOOL         USER PROGRAM FILE            110514  1  C 152
 124     S36   SCOTT.SM           USER PROGRAM FILE              8780  0  C 152
 171     S44   TOM.CSC            USER PROGRAM FILE              1794  0  C 152
 192     S24   MGR.PERFDEV        TDP.PUB.SYS                   12796  0  C 152
 202D    S41   RAT.PERFDEV        OPT78.LACH.PERFDEV             1619 10  C 156
```

Figure 6.9 - OPT's User Summary Report

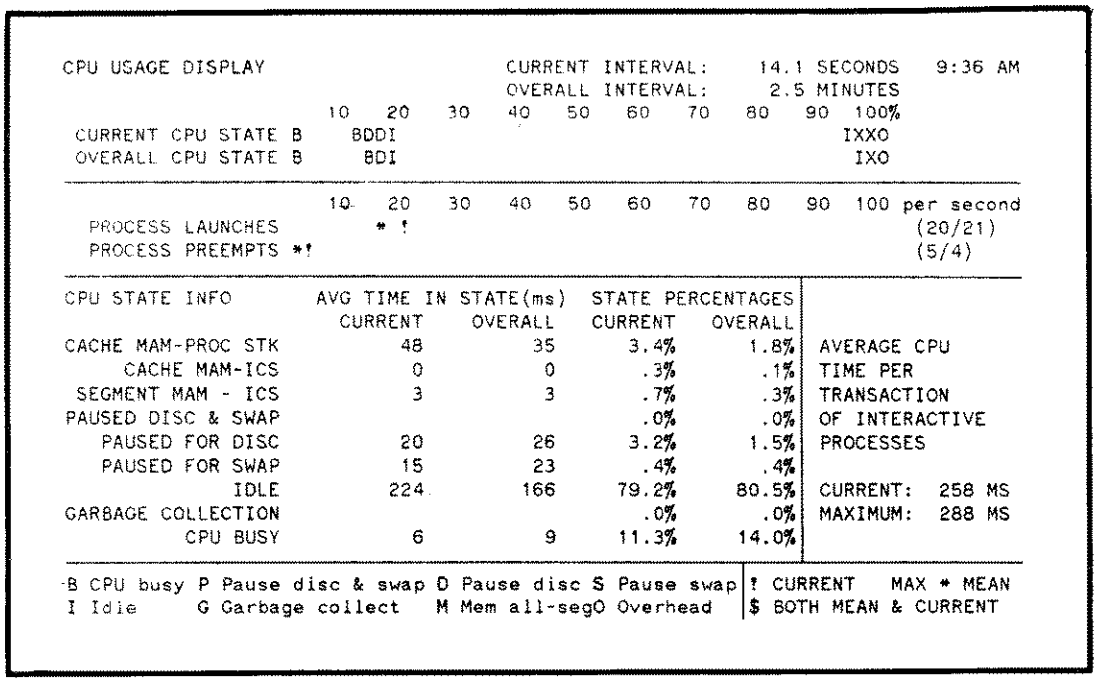


Figure 6.10 - OPT's CPU Usage Display

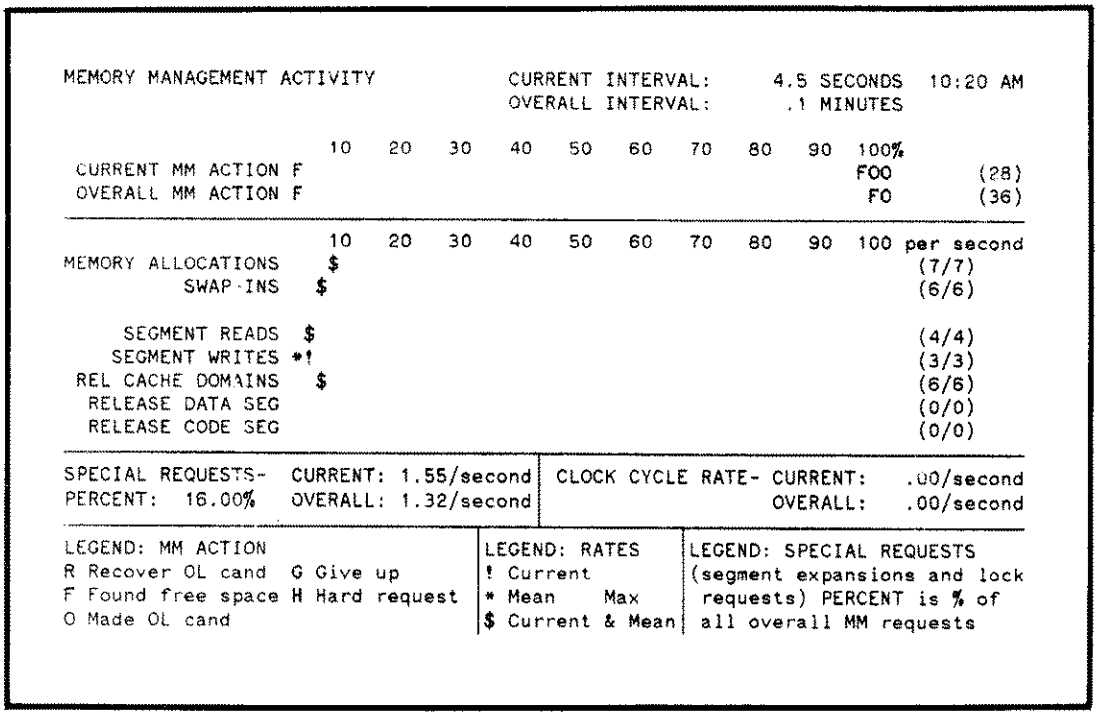


Figure 6.11 - OPT's Memory Management Activity Display

#MO20 Use SYSVIEW to Monitor System Performance

SYSVIEW (Carolian), has designed display screens around end user needs (as opposed to those of a lab scientist), and added some very user friendly benefits. Ease of use ranks high with SYSVIEW.

SYSVIEW was designed with the end user's performance monitoring needs in mind. Consequently it boasts of a number of features that make it worth obtaining a demonstration copy. This program ranks high on the author's list of monitoring methods. Some of SYSVIEW's outstanding features are listed below:

1) Clear, easy to understand screens that include relevant data.

2) An ASYST function that reports when a particular resource statistic exceeds the value that you have specified as acceptable (very useful).

3) All commands are available in batch mode.

4) Both process and global information are logged to a logfile if desired; a formatting program is included to review this data at a later date.

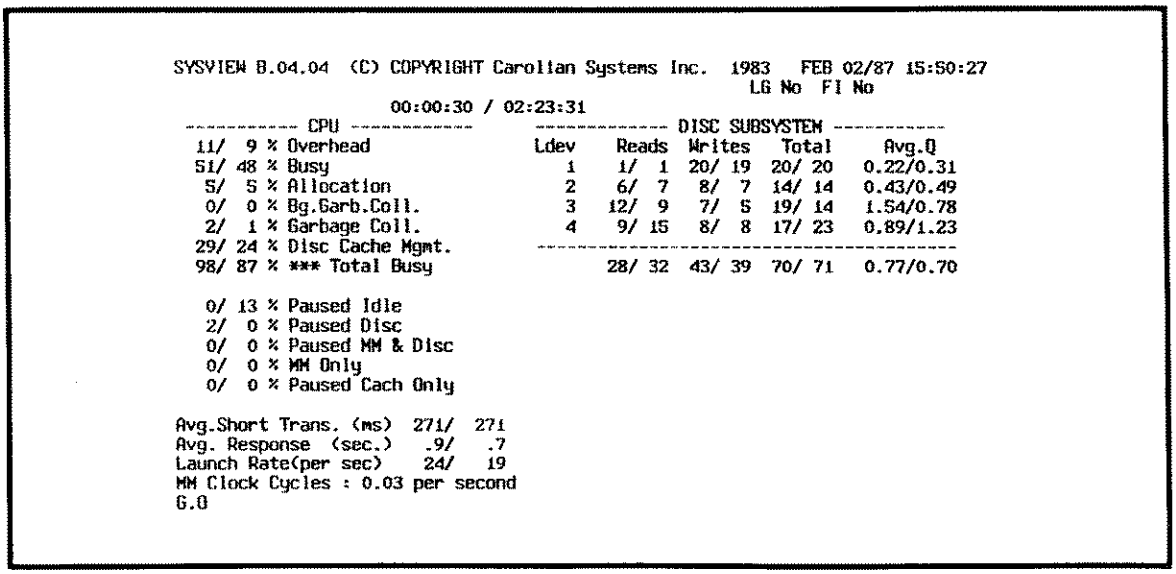
5) Groups of users/programs may be filtered to allow monitoring their resource utilization alone.

6) Good documentation with interpretation guidelines are included in the manual.

7) Disc Caching data is available with very relevant disc I/O information all on one screen.

8) Response times for average, total system, and individual processes are available.

A few of SYSVIEW's screens are included. Figures 6.12 through 6.15 represent a subset of the number of ones available in SYSVIEW. This program has more relevant information per screen than OPT, but lacks any graphical data display.

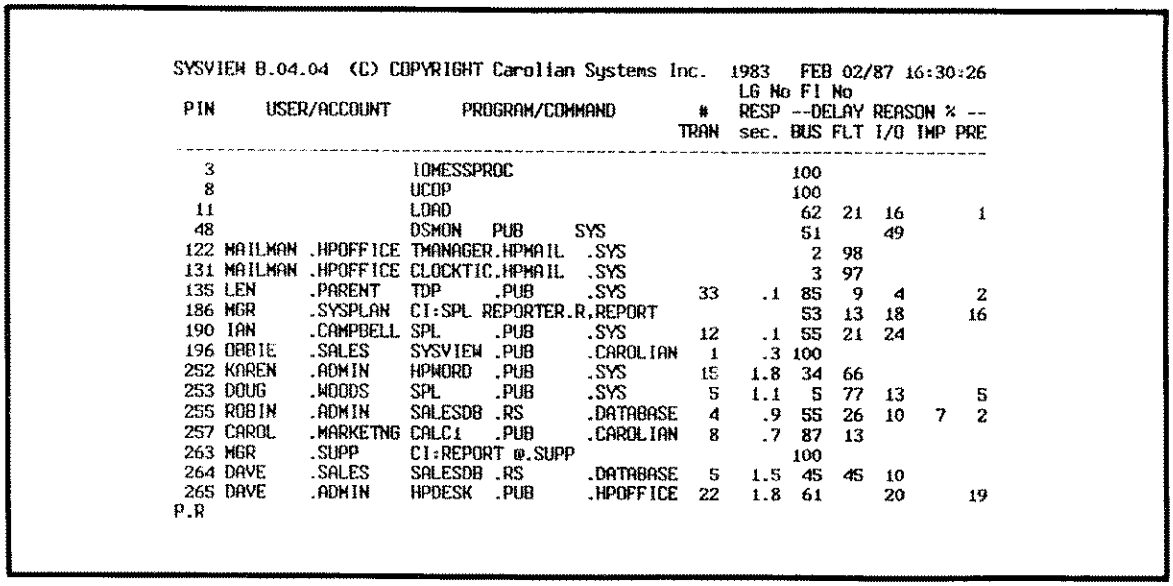


```
SYSVIEW B.04.04  (C) COPYRIGHT Carolian Systems Inc.  1983   FEB 02/87 15:50:27
                                                          LG No  FI No
                        00:00:30 / 02:23:31
 ---------- CPU ------------      ------------ DISC SUBSYSTEM -----------
  11/  9 % Overhead               Ldev    Reads  Writes   Total     Avg.Q
  51/ 48 % Busy                      1    1/  1  20/ 19  20/ 20   0.22/0.31
   5/  5 % Allocation                2    6/  7   8/  7  14/ 14   0.43/0.49
   0/  0 % Bg.Garb.Coll.             3   12/  9   7/  5  19/ 14   1.54/0.78
   2/  1 % Garbage Coll.             4    9/ 15   8/  8  17/ 23   0.89/1.23
  29/ 24 % Disc Cache Mgmt.         ----------------------------------------
  98/ 87 % *** Total Busy                28/ 32  43/ 39  70/ 71   0.77/0.70

   0/ 13 % Paused Idle
   2/  0 % Paused Disc
   0/  0 % Paused MM & Disc
   0/  0 % MM Only
   0/  0 % Paused Cach Only

Avg.Short Trans. (ms)  271/  271
Avg. Response  (sec.)   .9/   .7
Launch Rate(per sec)    24/   19
MM Clock Cycles : 0.03 per second
G.0
```

Figure 6.12 - SYSVIEW's Global Display



```
SYSVIEW B.04.04  (C) COPYRIGHT Carolian Systems Inc.  1983   FEB 02/87 16:30:26
                                                          LG No FI No
 PIN     USER/ACCOUNT       PROGRAM/COMMAND          #    RESP --DELAY REASON % --
                                                    TRAN  sec. BUS FLT I/O IMP PRE
-------------------------------------------------------------------------------
   3                        IOMESSPROC                        100
   8                        UCOP                              100
  11                        LOAD                               62  21  16       1
  48                        DSMON   PUB     SYS                51      49
 122 MAILMAN .HPOFFICE      MANAGER.HPMAIL  .SYS                2  98
 131 MAILMAN .HPOFFICE      CLOCKTIC.HPMAIL .SYS                3  97
 135 LEN     .PARENT        TDP     .PUB    .SYS       33    .1  85   9   4       2
 186 MGR     .SYSPLAN       CI:SPL REPORTER.R,REPORT           53  13  18      16
 190 IAN     .CAMPBELL      SPL     .PUB    .SYS       12    .1  55  21  24
 196 ORBIE   .SALES         SYSVIEW .PUB    .CAROLIAN   1    .3 100
 252 KAREN   .ADMIN         HPWORD  .PUB    .SYS       15   1.8  34  66
 253 DOUG    .WOODS         SPL     .PUB    .SYS        5   1.1   5  77  13       5
 255 ROBIN   .ADMIN         SALESDB .RS     .DATABASE   4    .9  55  26  10   7   2
 257 CAROL   .MARKETNG      CALC1   .PUB    .CAROLIAN   8    .7  87  13
 263 MGR     .SUPP          CI:REPORT @.SUPP                   100
 264 DAVE    .SALES         SALESDB .RS     .DATABASE   5   1.5  45  45  10
 265 DAVE    .ADMIN         HPDESK  .PUB    .HPOFFICE  22   1.8  61      20      19
P.R
```

Figure 6.13 - SYSVIEW's Process Display

```
A.FEB 02/87 16:15:31  CPU is over-utilized ( 81%)
                      CPU waiting for disc too much ( 11%)
                      Disc drive too busy (Ldev#3   23/sec)
                      Disc queue is too long (Ldev#3     1.83)

A.FEB 02/87 16:25:40  CPU is over-utilized ( 99%)
                      Disc queue is too long (Ldev#2     2.11)

A.FEB 02/87 17:06:41  Poor response for : PAT      .SALES    S#34    Ldev#54
                                          SALESDB .RS       .DATABASE
                                          #Trans = 2        Response =   8.3 sec

A.FEB 02/87 17:17:42  Poor response for : PAT      .SALES    S#34    Ldev#54
                                          SALESDB .RS       .DATABASE
                                          #Trans = 1        Response =  12.4 sec

A.
```

Figure 6.14 - SYSVIEW's ASYST Screen

```
SYSVIEW B.04.04  (C) COPYRIGHT Carolian Systems Inc.  1983   FEB 02/87 16:31:55
                                                        LG No FI No
Table Description              SYSDUMP Internal Current      Maximum       Size
-------------------------------------------------------------------------------
Code Segment Table                 384      576     243  42%     243  97%     4 Wds
Extended Code Segment Table        512      512     294  57%     294  59%     4 Wds
Data Segment Table                 750      752     421  56%     421  56%     4 Wds
Process Control Block Table        420      420      66  16%      79  19%    21 Wds
I/O Queue Table *                   96       96      26  27%      37  39%    12 Wds
Disc Request Queue *               192      192       7   4%      49  26%    17 Wds
System Buffers *                     8        8       0           1  13%   129 Wds
Swap Table *                      2047     2047     319  16%     399  19%     6 Wds
CST Block Table                     56       56      41  73%      41  73%     1 Wds
Special Request Table               20      583       0           0           6 Wds
Interrupt Control Stack           1024     1024                 593  58%     1 Wds
UCOP Request Queue                  64       64       0           0           2 Wds
Timer Request List                  64       64      14  22%      14  22%     4 Wds
Breakpoint Table                    24      404       0           0           Var
RIN Table                           64       64      20  31%      20  31%     3 Wds
Job Master Table                   150      150      24  16%      24  16%    38 Wds
Virtual Memory                      32       32      20  63%      25  78%    Ksec
Spooler Space                      128      128       9   7%       9   7%    Ksec
Note: * indicates maximum since last COOLSTART.
C.T
```

Figure 6.15 - SYSVIEW's System Tables Display

#MO21 Utilize PROBE/3000 to Track System Performance

PROBE/3000 (Strategic Systems) is the newest "kid on the block". The author was very impressed with the demo program. One immediate feature that has received comments by all whom the author has shown the program is on the first screen. The top five most active processes are displayed in descending order. This has great benefit since many times after glancing at the global CPU statistics it is desirable to isolate one or more processes which have monopolized the CPU.

The following features are but a few of the many which PROBE/3000 boasts of:

1) Aesthetically designed, relevant screens. PROBE is function key driven and very easy to use.

2) The very first screen has a plethora of useful information; it is designed in such a way to provide data for all major resources: memory clock cycle rate, top five CPU users, total disc I/O rates, CPU global statistics, number of jobs and sessions, and more.

3) It has two screens for databases. The first tells which databases are opened and then allows you to go to another screen based on a particular database to see which users are accessing it.

4) It allows MPE commands to be entered and then executed at a later time in the same fashion as that in MPE/XL. A command stack is kept of MPE commands entered, and then you can perform them simply by issuing a DO <# of target command in stack>. This number is obtained by entering LISTREDO; very nice feature!

5) It has numerous process screens with response times given on many of them.

6) A PRINT SCREEN function key allows you to obtain a hard copy snapshot of any screen.

7) An excruciatingly detailed Disc Caching I/O screen provides a wealth of current and cumulative Disc Caching statistics.

8) Screens are provided for locked SIRs, ALLOCATED programs, system tables (figure 6.19 - graphic and numeric display!), global process waits, and many more.

Figures 6.16 through 6.19 represent four of the many screens available.

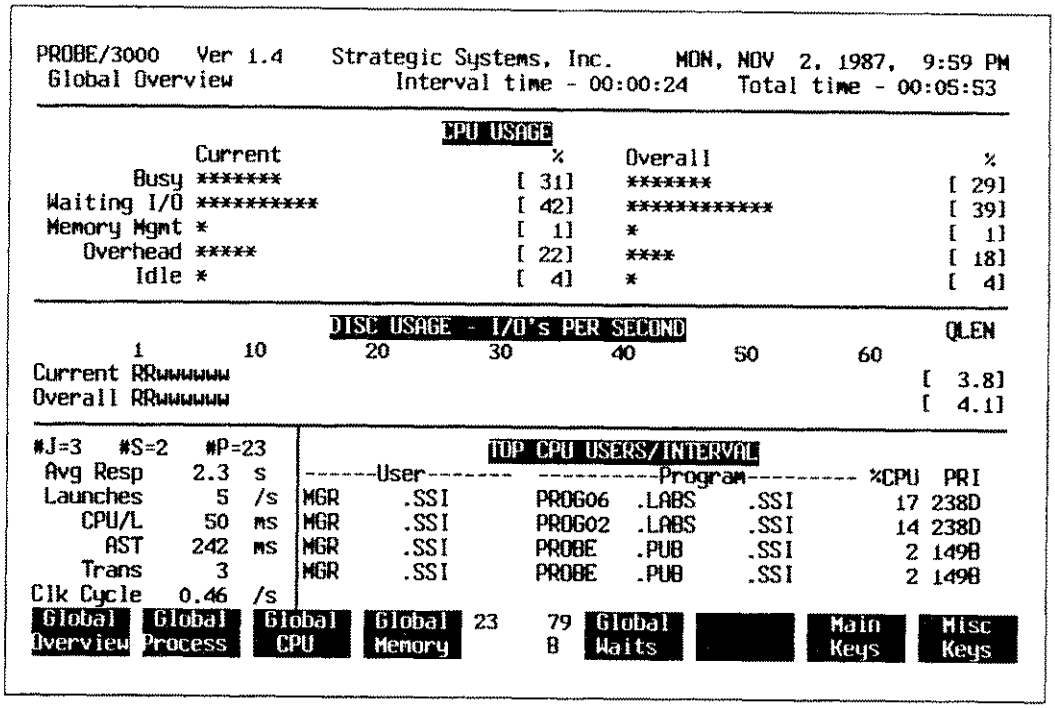


Figure 6.16 - PROBE/3000 Global Overview Display

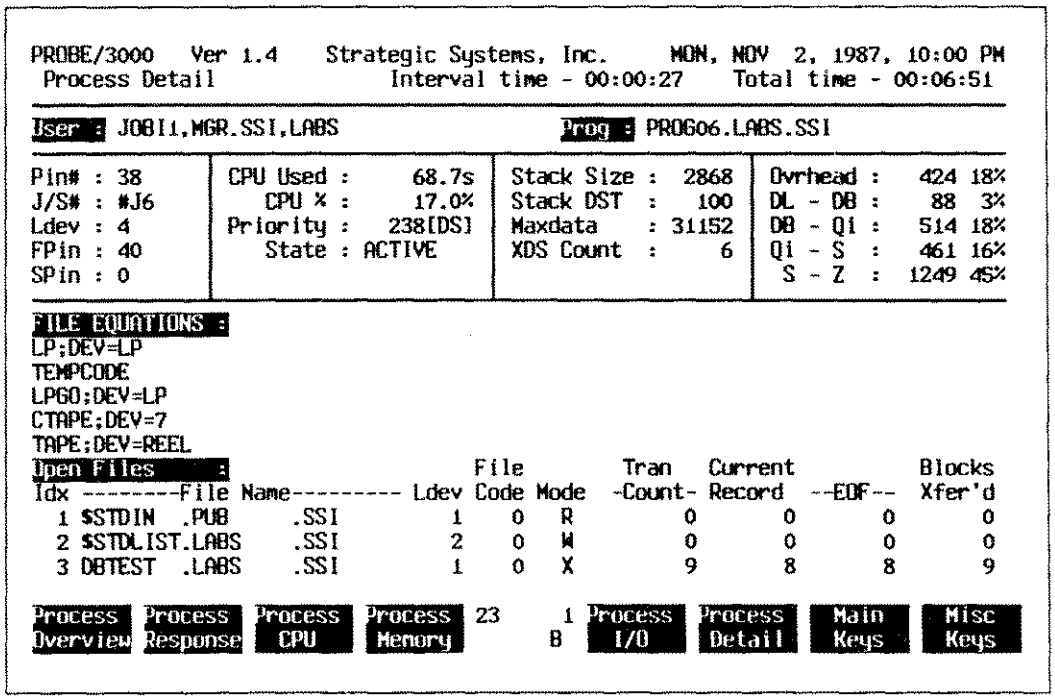


Figure 6.17 - PROBE/3000 Process Detail Display

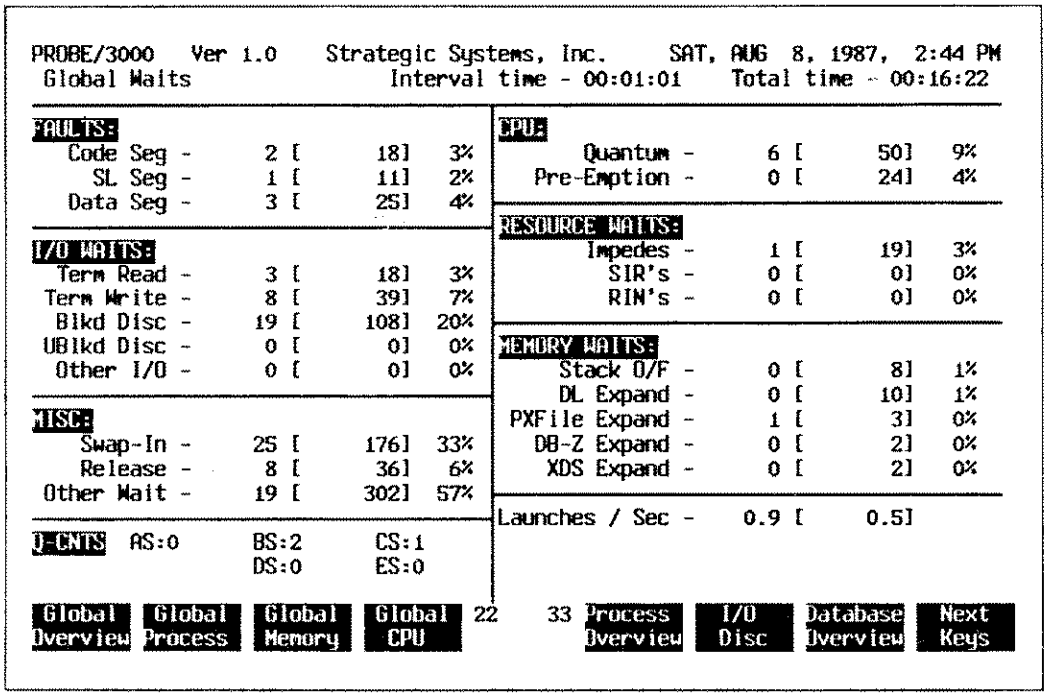


Figure 6.18 - PROBE/3000 Global Waits Display

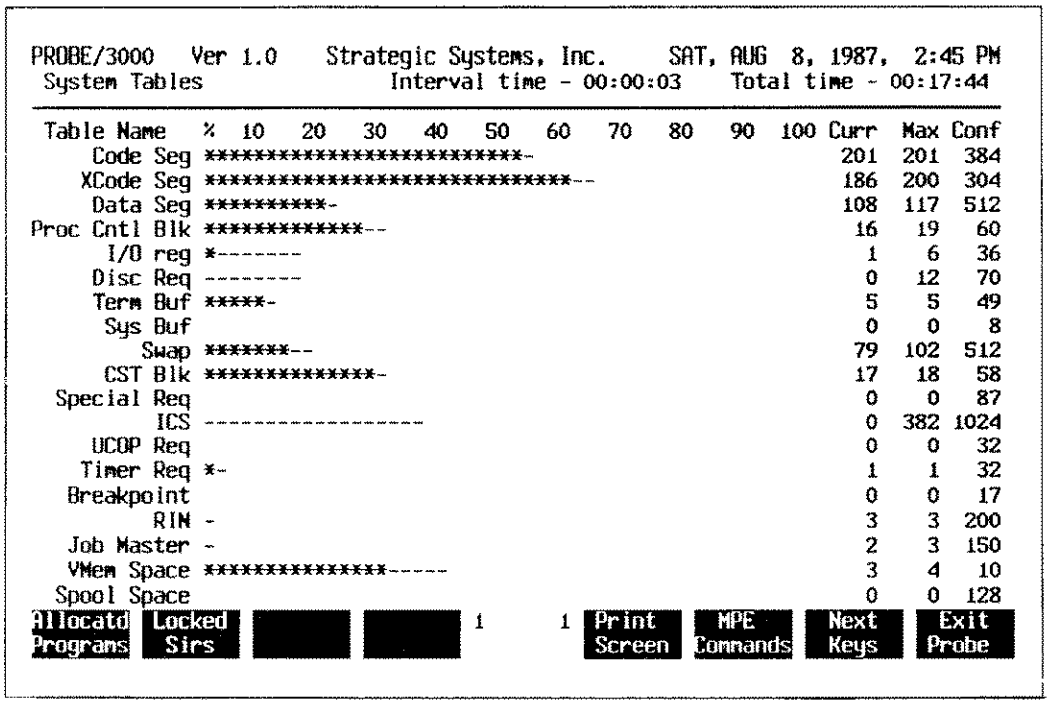


Figure 6.19 - PROBE/3000 - System Tables Display

#MO22 Use HPTREND to Look at Past Resource Utilization

This product is available from HP as a monthly or quarterly subscription service. It provides high level, graphical representation of key resource usage. It includes some of the following graphs of system resources:

1) Daily CPU, disc I/O, and terminal activity. These show percentages for CPU usage, rates for disc I/Os and the average number of terminal reads per minute. This same data is also averaged for a typical day.

2) A disc free space stacked bar chart provides free space and fragmentation history.

4) CPU usage by execution priority, batch jobs, and online terminals is given.

5) HPTREND also provides graphs for pages of printer output, I/Os by magnetic tape, connect hours by account, CPU hours by account, and more.

HPTREND will provide you with a reasonable picture of the overall performance of your system for the month or quarter that the report covers. This type of long-term data is indispensable in helping you develop a feel for your system's unique performance idiosyncrasies. Patterns such as month-end processing, holidays, and peak usage periods are all quite obvious on its charts.

#MO23 Utilize SYSPLAN to Track and Project System Performance

SYSPLAN (Carolian) is a performance data collection and reporting package which allows you to make accurate forecasting regarding system load/growth. It collects detailed data on the use of the system's resources. The data can then be graphed in any combination of users/applications/ports. SYSPLAN not only reports past trend information, but allows you to project growth into the future. You can even project the growth of any logical group (users, applications, ports).

With SYSPLAN you are not locked into a rigid set of resource utilization graphs. You have control over the reports and graphs. They can be as specific or broad as you desire them to be. SYSPLAN will report on the usage of the following resources: CPU busy, terminal transactions, Disc I/O rates, memory allocation, response time, disc freespace, printer usage, and tape usage. Forecasting is probably the most valuable feature since getting this information can be quite difficult and/or costly otherwise.

#MO24 Purchase an HP SNAPSHOT

An HP SNAPSHOT is essentially a program(s) that HP runs on your system for a set period of time (usually less than four hours) that collects very extensive performance statistics. That data is then massaged and then graphically and numerically displayed. An HP field performance specialist interprets the data and provides an extensive report on resource utilization during the collection window. If

you do not have a monitoring tool of your own and you experience a recurring performance problem, you will have to call for an HP SNAPSHOT (you could almost pay for a tool of your own for the price of one SNAPSHOT!). The scope of its coverage is so extensive that we will not even discuss its features. A data sheet is available from HP describing its gargantuan feature set.

#MO25 Purchase an HP CAPLAN

The HP CAPLAN product is essentially a capacity planning tool that utilizes a queuing network forecasting model to project system growth requirements. It uses data from a SNAPSHOT run as its input. Future system resource needs are then calculated based on "what-iffing". If you expect an increase in a certain number of users, transactions, etc., this data is then massaged and worked over with approximation algorithms. A final result for specific resource utilization is then presented. This whole process is surprisingly very accurate! If you are interested in this product, you will find information at your local HP field office. This service is not cheap (about $5,000 at printing time!).

#MO26 Use APS/3000 to Analyze Application Programs

Application Program Sampler (APS/3000 - HP) is a not-so-well-known performance tool that is used to interactively fine-tune programs. As an application program is running, APS/3000 monitors the amount of time spent executing each of the various program statements. After this data is collected, a fairly elaborate set of histograms are produced. By analyzing these histograms you are able to locate inefficient sections of code and then rectify them. This program is helpful not only during the development phase of applications but also after implementation. The author has used it on a number of occasions to isolate performance hindrances which were thought to be due to specific programs. Some of the features of APS/3000 are:

1) Data may be collected and put into a log file by APS/3000 on one HP 3000 and then may be transported to another system for examination.

2) Improper segmentation may be discovered by looking at the frequency of transfer between program segments.

3) Time spent waiting for various resources by certain program statements may be pinpointed.

4) CPU utilization may be determined for the entire program, specific program segments, procedures, and at the address level. When a good deal of time is being consumed by a portion of a program, that section is considered to be a "hot spot." The address range for that problem area is located in the target histogram, and then the spot is isolated in the program by using the VERBS map and the compiler listing.

Application tuning can be a very important component of an overall system performance gameplan. APS/3000 is one of the best tools that may be used to isolate application bottlenecks.

Appendix A

A System Performance Primer

This Appendix appeared in the September 1987 issue of INTERACT magazine under the title Taking Charge of Performance: the basics, by the author. You will notice that it is written in a rather casual style, and has as its primary intent to help a novice in this field gain some understanding of concepts and terminology as well as a simple game-plan to begin taking charge of system performance management.

It seems no matter what your role is in the company, your life is affected by the performance of a computer system.

If you are a user and the system is operating at its engineered optimum, things are great; if not, your productivity falls off. If you are a manager and the system is providing adequate performance, your life is running smoothly; if not, you know how "unsmooth" your life can be. In reality, an optimally running system is an asset that protects human productivity and perhaps even your job!

This article is aimed at those who have not had much (if any) training in system performance, and who desire to learn how to monitor and maximize the HP 3000's performance. The goal is to provide a limited amount of theory, and a means to learn about system performance (in particular, the reader's own system with its own unique workload) through observation - and lots of it.

Before diving into the actual study of your system's performance, I must discuss some seemingly philosophical issues regarding this queer science - insights I wish I had known in my early HP 3000 days.

How to become good at performance

What makes a person good at something is a blend of a well-rounded knowledge/theory base and experience in real life. This mixture is essential since experience often resets theory (I have in mind things like Columbus and the flat earth mentality) and vice-versa.

System performance, the study of the components that hinder or facilitate maximum transaction throughput and minimum terminal response time, has been referred to as more of an art than a science. Though I feel it is not as much of an art as programming, it still leans away from the side of the scale that includes

predictable science. That's not to say that transaction turnaround times (terminal response times) cannot be predicted. Since there are so many factors that affect performance, all too often the "experts" are predicting on the basis of "gut feel" and even educated guessing.

We would all like to be able to drop a few data points into a formula and pop out an exact answer. It just is not that way...quite yet. Those I would class as experts all seem to have one thing in common: they have observed numerous "ill" and "healthy" systems with respect to system performance. Most have learned an awful lot in that old academy of "hard-knocks." Not that they have shunned the theory, since I noticed that most of them were consumers of new information, but that they simply gained confidence by experience. And that is what I am advocating: gaining confidence by observing your system(s) over the long haul; blending an inductive and deductive approach to enable you to put your finger on the pulse of your system so you can make intelligent decisions about what maximizes throughput.

Warning number one - hasty conclusions

Before getting overly excited at the prospect of becoming a performance specialist, I need to toss a bit of water on your fire. It is essential that you resist the temptation to draw hasty conclusions about system performance. If you are taking the time to read this article, you most likely are not a specialist, since this is a very basic introduction. Therefore, you probably have limited exposure to the subject and limited experience in intelligently monitoring your system and it's particular application load/mix. Basing decisions on limited knowledge and experience can be financially and otherwise disastrous for you; I have seen both.

This is a discipline in which even the experts are often surprised. I'm saying this now because it will be especially tempting for you in the beginning to look for "allness" statements like: "You always need a megabyte of memory when your CPU is spending over x percent of its time doing global garbage collections"; or "Disc Caching should never be turned on when..." You get my drift. Don't formulate pat answers to be rigidly adhered to under every circumstance. Life, and especially the subject of system performance, is usually too complex to lend itself to simple answers.

Warning number two - when to call in an expert

Though you may intend to take charge of monitoring your system's activity and to implement "performance self care," you will probably need to call in a specialist at some point. If you really want to do a great monitoring job, you will need to spend money for a tool better than the one we will be using in this article. Here's why: My opinion is based on a finite amount of knowledge and experience. Unless your job is technical/performance support, your knowledge, experience, and especially time are probably limited. An expert is an expert because he or she has spent time becoming one. There is a time to call one. By beginning to learn some

basics in the subject, you can provide a great environment for the expert to come into: an environment of accountability. Don't be too proud of your newly learned basics to bring in the specialist, especially if there is a sizable capacity planning event that involves considerable consequences.

Warning number three - limited observation period

Now that I have said what I have in Warning Number Two, I would like to add that I have seen lots of memory thrown at systems that simply needed database reorganization, or Disc Caching parameters tuned. If you are basing major decisions on a specialist's results, just as in the medical world, there is nothing wrong with seeking a second opinion.

Remember, even the expert is limited. I have maintained that I tend to trust those who stand to lose something or gain nothing at all by giving me certain advice. This is another way of saying beware of expert biases. If a person is telling you that you need a faster CPU, and he sells "Belch-Fire-3" CPU upgrades, be careful. An unbiased person (is there such a thing?) is in a good position to advise you. The reason I say this is not to develop undue suspicion, but simply to ask three questions about a specialist: 1) Can he or she be trusted? 2) Does he or she care about my situation? 3) Does the "expert" know what he or she is talking about?

Warning number four - limited observation period

I know you are growing tired of warnings, but believe me, I wish I had known about these things in my early HP 3000 days. These things are a non-optional foundation to build your performance self care structure on. This one involves basing conclusions on a limited data collection period. I am not a statistician, but one thing I do know: The result of a sampling study is no better than the sample it is based on. You knew that! To be worth much, a conclusion based on sampling must use a representative workload. That is, when you are taking the system's pulse, be sure the applications at the time represent the symptom that you are trying to diagnose. I have seen performance collection tools run either over just a short period of time or at the wrong time. Either oversight will skew the results accordingly.

I have had the greatest success by collecting performance data over time (I don't mean 30 minutes!). If I wanted, I could make statistics collected during a specific short run period illustrate the need for a CPU upgrade, when in reality, the result is not representative of a steady state, but rather a "spike" in the system's activity. I am particularly glad to see products like HPTREND (HP) and SYSPLAN (Carolian Software) that enable you to see the big picture. Tools like these will help us avoid basing our performance analysis on a narrow statistical sampling.

Philosophy of performance management

As in most decisions in managing a computer system, you are simply juggling resources. Usually it's manpower, time, and money. This is also true for managing system performance. Managing system performance is much like managing one's health. Either manage pro-actively, or deal with a major problem that is quite out of hand at a later date. It's also true that bad performance symptoms can be glossed over by simply upgrading your CPU, getting faster disc drives, or adding more main memory. These options might be taken instead of locating and fixing "HOG/3000" application programs, balancing the system's most frequently accessed files, maintaining efficient databases, etc. If you have money to burn, or if you don't want the hassle of managing the system as I am advocating you do in this article, you can opt to throw more and/or faster hardware at the system. Just realize, in my opinion, you are not dealing with the root of the problem but merely with undesirable side effects of inefficiency.

Don't get me wrong, however. As your system grows in features and functionality, you will inevitably have to upgrade hardware. Suffice it to say that a better approach is to have a plan to monitor, maintain, and maximize performance on a before-the-fire basis. A newer generation of health care professionals is beginning to lobby for preventative health care, which involves what I call "systemic" management versus "symptomatic" management. The former emphasizes pro-active, and the latter post-active management. If you favor solving the problem before it occurs (and thereby avoiding those nasty "are we down?" phone calls) then read on; otherwise stop now as you will be bored and frustrated.

Some symbols to hang performance concepts on (a survival glossary)

Like any specialized field, the study of system performance has it's own set of esoteric terminology. I will give just enough to get you started in wading through the rest of this article and to help you get through other performance reading material.

Bottleneck - A needed resource that is in short supply and heavily used, causing the system as a whole not to perform optimally. Memory, CPU, disc, and certain application inefficiencies tend to be major bottleneck areas.

Disc Caching - A feature of MPE that makes any available memory act like an electronic disc drive. Since disc I/Os are very slow (they involve electro-mechanical movement) in contrast to main memory transfers (purely electrical), under certain conditions, Disc Caching can make a tremendous difference in improved system performance. More data is brought in to main memory than is needed in anticipation of future need. When a request is made for data, certain areas in main memory (called cached domains) are checked prior to actually performing a physical disc I/O.

Dispatcher - A part of MPE. Its job is to decide which process is to get the CPU's attention next. It always schedules processes to favor high priority ones.

I/O - Stands for Input/Output. It is used to describe the process of either performing a read or a write to disc.

Launch - A process is said to be "launched" when the dispatcher allows the process to begin execution.

Memory Pressure - A specific point that is reached when the memory managing part of MPE has searched through memory looking for large enough areas to satisfy requests in less than a certain period of time. The memory pressure flag, when raised, tells MPE to start doing garbage collection. The memory pressure rules are a bit different, however, if Disc Caching is enabled and you have more than 2 megabytes of memory.

Response time (aka transaction turnaround time) - The amount of time it takes for a user, having pressed the RETURN or ENTER key, to get a prompt for the next input.

Nickel tour through MPE internals

The best performance persons are those who also have a good understanding of what makes life easy or difficult for MPE. I will obviously not be able to spend much time discussing MPE internals as it is a voluminous subject. But, having a basic understanding of some of the overall policies of MPE will help you understand what you see on the performance statistics reports you will be seeing.

In a very real way MPE can be likened to an automobile engine. I am loosely coupling the actual hardware chips, microcode, and operating system software in this analogy. The goal of this engine is to perform work and the amount of work accomplished in a given period of time is what we call throughput. This is just like a car's engine being able to cover a certain distance, given a certain weight, at a certain speed. Obviously the larger the engine, the heavier the load that can be carried at a desired speed. If we were to have a small passenger car attempt to carry a two-ton load, it might be able to arrive at the desired destination, but it would probably take much longer than if we chose a large truck that was suited for this task. It is much the same with differing "horsepower" CPUs. A larger CPU (say a Series 70) will be able to process more transactions in a given amount of time (all other things being equal) than, say, a Series 37.

That much I probably didn't need to tell you; it's relatively obvious. But in our analogy I would like to stress the following: Just as an engine has certain overhead that is required to continue operating as an engine, so does a computer system. The drive belts that power smog equipment, power brakes, water pump, and so forth, all require a finite amount of horsepower to be stolen from what we really want to accomplish in the first place: to produce usable horsepower. If there are worn bearings in power driven components, tight belts, and other inefficiencies, the engine will be limited in the amount of payload that can be moved within a desired time frame. Here is where our picture shines. MPE, when faced with similar overhead drains, has the same problem of accomplishing usable, beneficial work within an acceptable time frame. By monitoring the amount of time the system is spending in each of these overhead areas, we will be able to determine how to minimize this "horsepower" loss.

I suppose we really cannot squeeze much more out of our analogy; it is limited. But if you will keep in mind this concept of an engine operating with some overhead, you will benefit much from our discussion of system performance.

MPE is an interrupt-driven operating system. It does not go looking around for things to do on its own initiative. It makes its decisions and performs work based primarily on "nudges" from within its environment (inside the computer cabinet) and from the outside world (terminal, tape drive, and so forth). Every process on the system is entitled to a certain amount of the CPU's attention. You may have noticed I am using the term "CPU" and "MPE" interchangeably; they are two separate entities, but work together so closely that for our purposes, we will lump them into the one big concept of an engine. Now, here are a few of MPE's policies regarding performing work on behalf of processes:

1) A process will get approximately 300 milliseconds of the CPU's attention (if it needs that much) unless one of the following happens:

   a) A higher priority process preempts the current process

   b) The current process needs a code or data segment that is on disc, but not in main memory

   c) The current process needs either to read or write data from/to disc

   d) The current process needs a resource that is currently not available (RINS, SIRS - don't worry about these)

   If a process is constantly using a resource, we should find out why. Is there a logical problem (an application design problem), or a physical problem (simply not enough memory)?

2) If a memory segment or disc I/O has to be obtained or performed, the CPU must wait for that I/O to finish before it can continue doing what we want it to do: post the accounts receivable entry in the database, request a current account balance, and so forth. This time is tallied as a percentage of time that the CPU could not perform useful work, but had to wait.

3) MPE also has to make decisions on what to launch next. It's the highest priority process ready to run that gets to use MPE's engine.

   You may have seen batch jobs taking inordinate amounts of time to complete. One reason can be because higher priority interactive sessions are consuming all the available CPU resource. This is a very dramatic illustration of MPE's prioritizing scheme in action. Its intent is to insure user sessions are not usurped by batch jobs.

4) Certain garbage collection routines are invoked under special circumstances when main memory is at a premium. This is an overhead operation that involves shuffling unused small portions of main memory into larger pieces that are more favorable to MPE's use.

The main point I want to make from this discussion is that when resources are scarce (memory shortage, excessive disc I/Os, CPU unable to keep up with load), or if processes are inefficient, then MPE will not be able to perform the amount of work in the amount of time deemed acceptable. We usually measure this in terms of terminal response and the number of transactions performed per unit time.

If this last section intimidates you a little, don't worry. We will review some of these concepts when we look at real live performance data.

Getting a feel for your system's performance personality

You might have guessed by now that there are quite a few things that can affect your system's performance. And that is why it is not advisable to make "in concrete" statements regarding what affects good or bad performance. Your system is probably unique in its application mix and hardware configuration. And though there are some guidelines that are true no matter what system you have, each HP 3000 with a novel mix of workloads seems to have its own performance personality.

Probably the best way to get familiar with your system's performance characteristics is to devise a plan for monitoring it on a regular basis. That is what we will deal with next: A tool and a plan to accomplish this goal.

Performance paradigms developed in the laboratory are a great place to start, but each system with its unique set of loads has its own performance idiosyncrasies. The best way to discover what is going to produce the symptom of, say, memory pressure on your system is simply to monitor it over a period of time under the various load mixes that run on it. This will allow you to see which programs and which types of loads produce various performance stress indications. But how to monitor the system?

Learning by observation: a tool and a plan

There is really only a limited number of performance data gathering tools available on the market or in the INTEREX Contributed Software Library (CSL). As a good (inexpensive) way to start, we will discuss the use of a public domain program called SURVEYOR. This little program has been around for a while, and for being a freebie is actually quite nice for not only cutting your teeth in performance analysis, but for continued use in shops that run on "lean-and-mean" budgets. Keep in mind however, that it does not report some of the critical data needed to diagnose performance problems.

If you have the TELESUP account on your system, it may be in the PRV group. HP has removed it from recent TELESUP tapes due to a problem with system failure 16 occuring. Due to this problem, you may have trouble getting a copy. See the discussion as to the occurrence of this problem a little while from now.

I will assume you have SURVEYOR (if not, write Performance Press), and that you have very little or no knowledge of how it works, or how to interpret the results. Let's be clear up front: I am going to be quite vague in regards to the interpretation of the results (with a few exceptions). I am going to say things like "it seems like," or

"this may indicate," or "in this circumstance." It's like words on a printed page. Although they have definitions, much of their meaning derives from a specific context. Our context is your unique HP 3000 with it's quantity and quality of workload.

Enter SURVEYOR

To introduce you to SURVEYOR, I am going to use a synthetic load-inducing set of programs. These are merely programs written to produce various types of "real-life" stresses on the system. We will take note of how the various performance indicators vary with the different quantities and kinds of loads we put on it. Note especially how the number of transactions completed during the 10-minute time frame varies. After all, this is one of the most meaningful measuring sticks of performance we use.

Getting started with SURVEYOR

Check to see if your PCB system table is configured to greater than or equal to 628. If so, because of a bug in SURVEYOR (versions less than or equal to B.00.04), you may experience a system failure. To be safe, configure your PCB down to under 600, that is if your system does not need that many PCB entries. SURVEYOR can be run interactively or in a batch job.

```
------------------------------------------------------------------------------
System Surveyor B.00.04 (JAK)          MPE V - Unsupported Contributed Utility
------------------------------------------------------------------------------

Samples started WED, MAY 13, 1987,  9:40 AM
Mpe G.B2.02 (G.B2.02)  linked memory size- 462080 words

******************************************************************************
System Surveyor B.00.04 (JAK)     WED, MAY 13, 1987,  9:42 AM Elapsed 00:02:00

******* C P U ******   *** Paused for Disc IO ****  Memory  Garbage Collection
Idle    Busy   Cache   Mam   User&Mam  User  Cache  alloc.   Global     local
60.5%  28.3%    0.0%    0.2%   0.1%    6.4%   0.0%    0.0%    0.4%       0.0%
------------------------------------------------------------------------------
Drive-     1      2
R/sec-   0.8    1.0
W/sec-   5.0    7.3
------------------------------------------------------------------------------
CPU:  Avg CPU Busy-      0ms; Avg Shrt Trns Time-  259ms; %Preempts-          9
MEM: Launches/Sec-       5;   'Swapins'/launch-    0.0;  Mem Clck Rate     3806
IO:  IOs/Term read-    546;  CPU msecs/IO-          62;  ** MEMORY PRESSURE **
------------------------------------------------------------------------------
```

Figure A.1 - A Sample Surveyor Screen

After you have logged on and issued the RUN SURVEYOR command, you will be prompted with a series of questions. For now, answer the first two with just carriage returns.

A screen similar to that in Figure A.1 will now appear. If you see more lines of information below the top display, these are process level data. This is a great feature: to be able to display each process on the system along with a limited amount of information pertinent to that process. We will touch on process information only briefly in this article, but this information is valuable in the sense that there are not many ways to look at process states without having to spend a good deal of money for a more advanced monitoring tool.

Where to begin?

I want to focus on the global portion of the display ("Idle", "Busy", etc.), but we will briefly mention a bit about the following two sections. We will refer to the three sections separated by rows of "----" as sections 1, 2, and 3. Notice how the screen is updated about every 30 seconds or so.

Section one is divided into three groups: CPU status, which includes the following:

1) Idle - This is the percent of time the CPU was not doing any productive work on behalf of user programs, and was not doing any "housekeeping" chores. Lot's of idle time (as observed consistently over a period of time) simply means that you have some spare CPU horsepower in the bank.

2) Busy - This is the percent of the time that the CPU spent doing what we would consider productive things for user processes (note that I am using the words processes and programs loosely interchangeably) like sorting a file, calculating a dividend, and so on.

3) Cache - The percent of time the CPU spent managing the Disc Caching facility.

4) CPU pause for MAM (memory access manager) - The percent of time the CPU spent waiting for a needed program or data segment to be brought into main memory from disc.

5) CPU pause for user & MAM - The percent of time spent waiting for needed code/data segments and a specific I/O request to complete from disc.

6) CPU pause for user I/O - The percent of time spent waiting for necessary disc I/Os to complete.

7) Cache - I have never seen this number be other than zero, so I am not sure if it has been implemented in SURVEYOR.

8) CPU busy on memory allocation - This is time spent managing the memory facility.

9) Global garbage collection - This is time spent consolidating pieces of unused main memory into larger, more usable pieces. This is done only under the following conditions:

   a) The CPU has some idle time *and*

   b) MPE senses a memory shortage *and*

   c) Disc Caching is disabled (unless you have less than two megabytes of memory)

10) Local garbage collection - Think of this as routine memory consolidation done as a matter of normal memory housekeeping chores; not usually a sign of memory shortage.

Section two breaks out the number of reads and writes performed to each disc drive. This information is helpful in determining, at a glance, how balanced the I/Os to your files are. If one drive is consistently getting a higher number of reads and/or writes than others, you may want to look at spreading the more frequently accessed files across many drives to improve I/O efficiency (see Chapter four for more information here).

Section three provides some good information, but since this is a primer I will talk about only a couple of them.

1) Launches/sec: This is the number of processes handed over to the CPU to have work performed on their behalf per second. It can be one indicator of how much activity is occurring.

2) Swapins/Launch: This can be one indicator of a memory shortage. With Disc Caching off, if this number is excessive, we might want to look for other memory shortage symptoms. This number means that for every process whose turn came up to run, a certain number of segment retrievals needed to be performed for that process to run.

3) Mem Clk Rate - This is the number of CPU milliseconds (wall time for versions < = B.00.00) performed during the given interval divided by the number of clock cycles. If this number is small, MPE checked memory quickly meaning there wasn't much available space. A large number (thus, a slow time) is desirable. An asterisk here means the speed was extremely slow - a good sign.

4) Memory Pressure - If "*** MEMORY PRESSURE ***" is on, MPE is literally "under the gun" to find free memory space.

Observing performance trends

Armed with a simple understanding of MPE and SURVEYOR, we will now look at various loads that have been placed on a lowly Series 37. In some cases I have purposely exaggerated a bottleneck so you will see what SURVEYOR reports in mild and extreme cases. If there is anyone from the SFPCS37 (Society for the Prevention of Cruelty to Series 37s) reading this, please forgive me. I really push this box. This will help us spot trend indicators that point to CPU overload, memory shortage, and I/O saturation. We will do this by increasing loads in each of these areas until it says "uncle" - until we most evidently see signs of overload. We will look at tables and graphs that illustrate anxiety for the CPU, Memory Manager, and the I/O system.

Tables and graphs...the fun part

The first graph (Figure A.2) illustrates our base run. That is, what we will use as our normal, "operating pretty well" system. This first graph plots the number of users (terminals) versus response time as perceived at the terminals. Notice how the times proceed fairly linearly until a certain point where there appears to be an almost exponential increase in response time by adding more users. This threshold, linear to non-linear, is described sometimes as a "knee in the curve." This is the point where adding more load proves to have devastating effects on system performance.

Looking at the graph of SURVEYOR's data collection during this period, we note the following as sessions were added. What we know intuitively is really true: CPU, memory, and disc I/Os become a premium.

Figure A.2 illustrates a number of interesting traits. This is an example of Disc Caching hurting instead of helping response time. With 1MB of memory, there simply is just not enough; with 2MB the response time is a little better up to the addition of that sixth or so user, when the transition changes from a nice smooth increase to a much steeper slope. This is not quite an exponential relationship at this point, but something radically changes. As we will see in subsequent discussion, MPE gets quite swamped with overhead tasks. Things are going fairly well up to a point, and we reach a point of diminishing returns: adding more users to get more done has a net negative effect on the overall system. If we just referred to the 1MB slopes, adding 1MB of memory would indeed "pull" that response time curve down somewhat. I have a suspicion that more memory would help even further.

It is tempting at this point to conclude a number of things. Remember, the application that I am running on this Series 37 is going to be different from the one on yours; the resulting response time curve will be different. What I am saying is don't let that six or seven user threshold stick in your mind as a maximum user limit.

Isn't it also an obvious fact that Disc Caching hurts rather than helps in this case? There is a short period of time when it does provide a benefit (just prior to the addition of the sixth). But overall, it just adds to the housekeeping chores MPE must be distracted with in this particular situation.

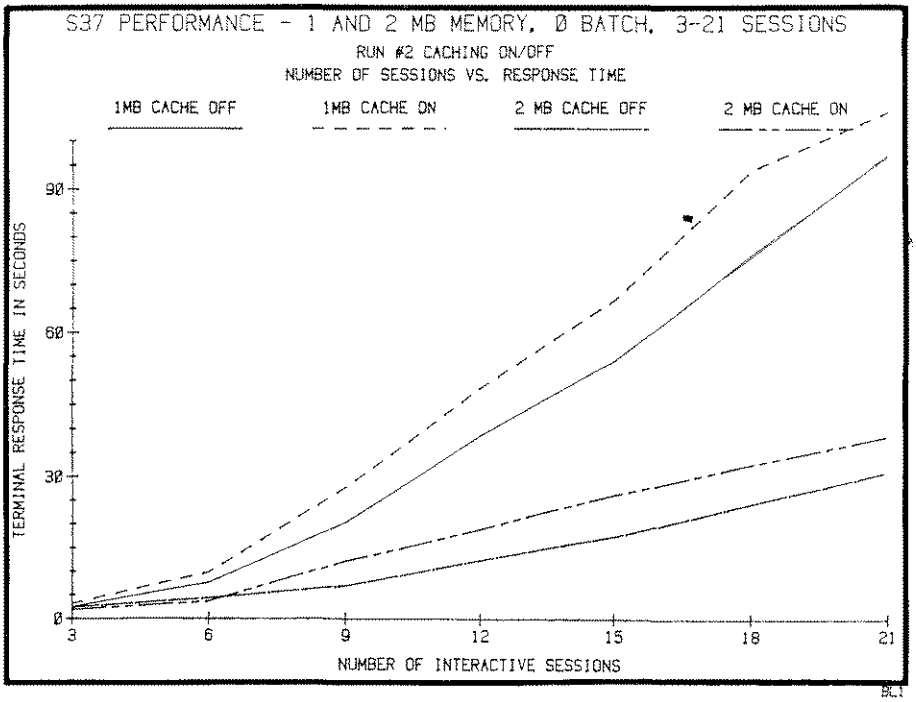


Figure A.2 - Caching and Sessions versus Response Time

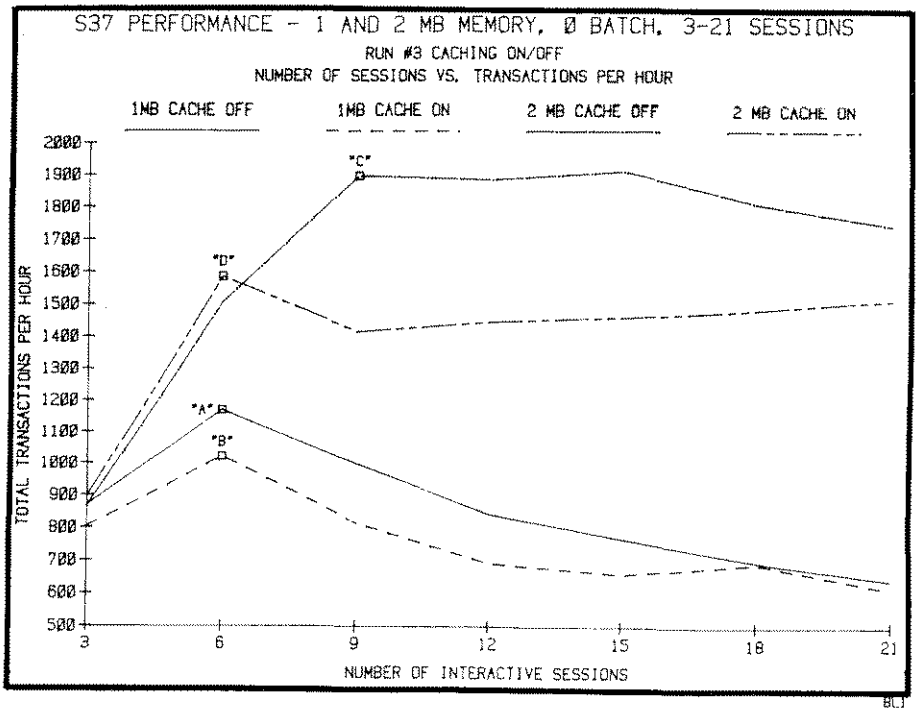


Figure A.3 - Caching and Sessions versus Transaction Throughput

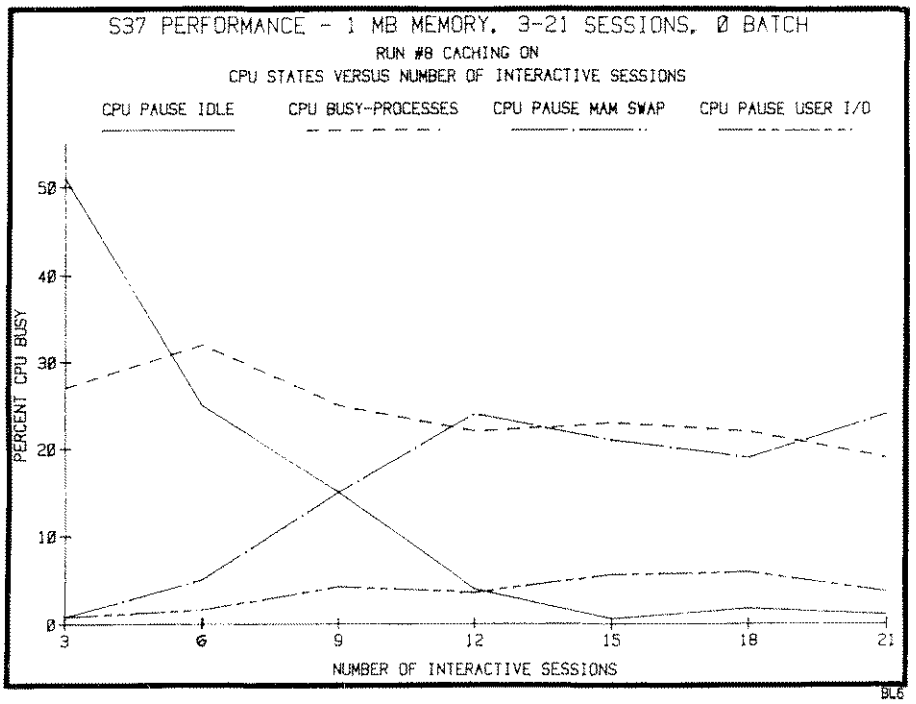


Figure A.4 - CPU States for Increasing Number of Users

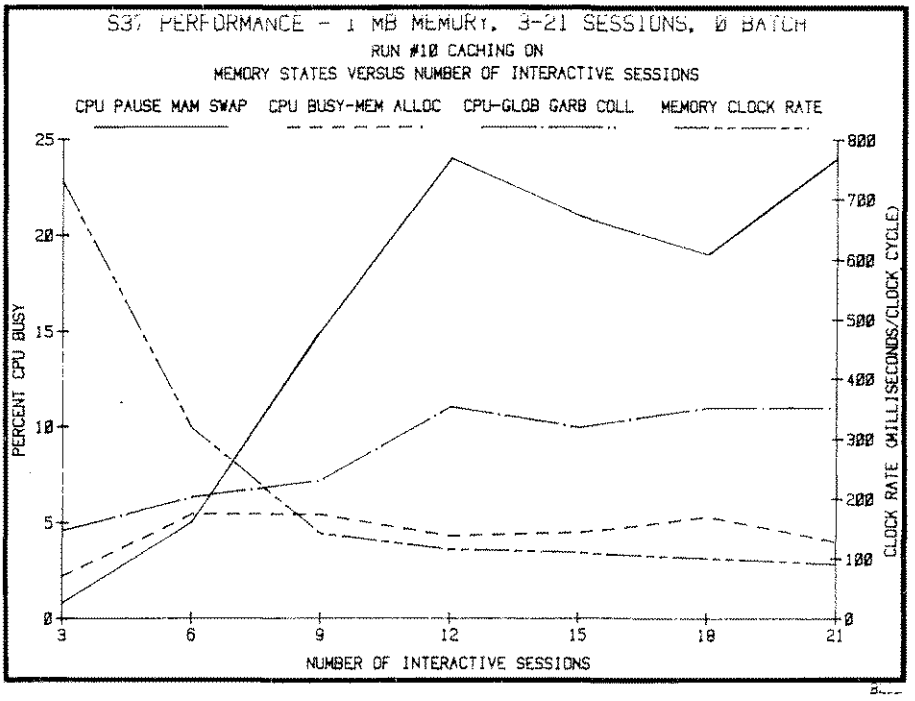


Figure A.5 - Memory States for Increasing Number of Users

Figure A.3 illustrates the efficiency of the system in yet another way: transaction throughput. With the same rules as in Figure A.2, we see a dramatic difference in the number of transactions accomplished by the curves represented by points B and C. Also, we not only see a change at points A, B, C, and D, but we see another example of a negative return for our addition of users. We accomplish nearly twice as many transactions with the addition of memory. Figures A.4 and A.5 are the "pared down" charts of SURVEYOR's data collection referred to earlier. Figure A.4 illustrates some of the major CPU states during this period of time. Notice how idle time drops off, and how the amount of time the CPU spends on doing productive work on user processes begins to drop off at the sixth user point.

Figure A.5 shows a more dramatic issue: our memory bottleneck. Look at the radical valleys and mountains. Hold it one moment! Have you been paying attention to the top end values for the "Y" axis? Remember our previous discussion of the morality of statistics? Well, here is a good example. In Figure A.4 the "CPU PAUSE MAM SWAP" is the same curve as the one in Figure A.5! I simply changed the top end of the scale from 50 percent to 25 percent. I did this to fully use all the white area that would have been on the chart if I had used a scale of 0-100 percent. But didn't I have you reeling when I pointed out those "radical mountains and valleys?" I am making a fuss at this point to help you keep hardware peddlers honest!

Actually, Figure A.5 does point out some extreme memory issues. The amount of time the CPU is waiting for MAM SWAP is horrendous. You probably would not see this amount of garbage collection or such a low clock rate (right "Y" axis) on any but very memory short systems. I won't graph you to death by showing you the other charts for 2MB and caching off, but the ones where caching is off and memory is added all but eliminate these memory pressure symptoms.

What would the addition of one batch job have on the total transaction volume? Figure A.6 shows us. Curves D and C on Figure A.3 are the same as the dashed and solid ones of Figure A.5 (the scale is different!). We do accomplish more transactions, but since we have more overhead, we begin to see a negatively sloping curve for the batch job ones.

The process of varying disc I/Os has a fairly dramatic affect on response time and transactions per hour. Figure A.7 speaks loud and clear to this point. As the number of I/Os requested goes from acceptable to the absurd (the HP 7945 drives simply cannot perform that many I/Os per second), we see a fairly dramatic result in response times and the number of transactions performed. Notice how Disc Caching again helps somewhat, but under this application/memory/CPU mix, we don't see that great improvement that some folks do. The point is that anything we can do to eliminate disc I/Os will help our system performance. Memory for caching, efficient Image chains, faster disc drives, optimal IMB and GIC configuration, and so forth, all contribute to eliminating disc I/O as a bottleneck.

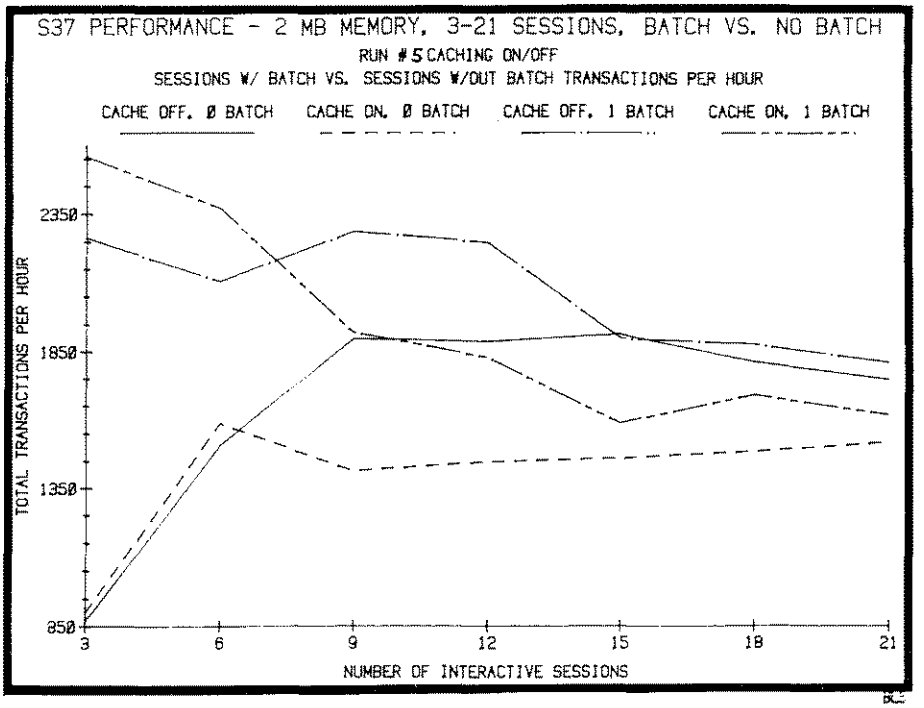


Figure A.6- The Effect of Batch on Throughput

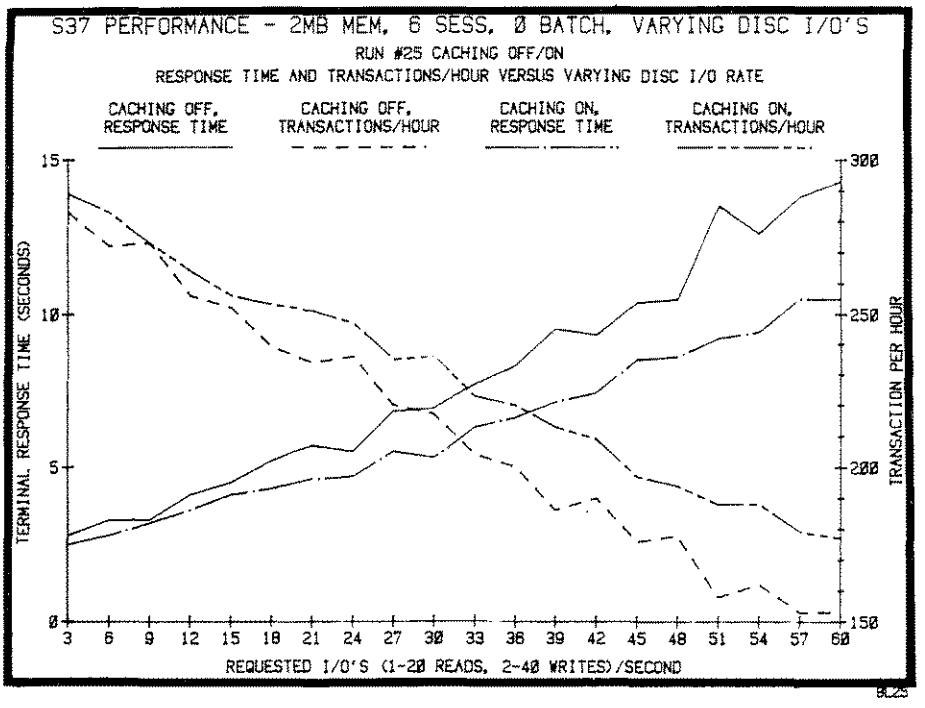


Figure A.7 - The Effect of Varying I/Os on Response Time and Throughput

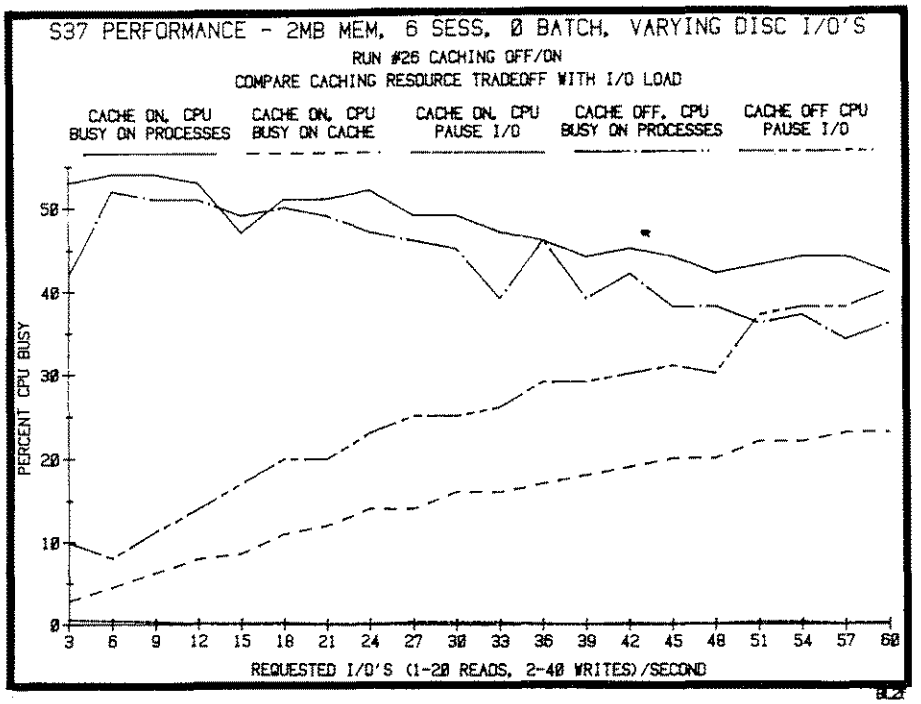


Figure A.8 - The Effect of Varying I/Os on Critical Resources

```
                                  Q &   %Tot   *W A I T   S T A T E S*
 Program Name                      Pin   Cpu   Abs    Dsc
PROG   .SJMC. LUND   S48   C68     1%    15%    26% ..............
PROG1  .SJMC. LUND   S52   C107    0%    35%    12% ..............
PROG2  .SJMC. LUND   S11   C108    2%    18%    11% ..............
```

Figure A.9 - Abbreviated Portion of the Process Display Section in SURVEYOR

Figure A.8 illustrates the resource tradeoff between a cached and non-cached system. With caching on, the CPU is much busier (add the Busy on Processes to the Busy on Cache) than with caching off. Though idle time is not shown here, there is a greater surplus under the cache off scenario. But look at the Pause for I/O while caching is off! It jumps up to 20 percent or so. With caching on, it is barely noticeable at the bottom of the chart. There you have it. We trade slow I/Os for fast memory access at the expense of available CPU and excess memory.

What we have seen in the last seven figures is the result of running lots of jobs. These pictures help us recognize trends in performance. As we look at these kinds of numbers in our own environment, we will begin to learn about what loads cause which stresses. This will also help us make intelligent decisions regarding our own system's performance...like when to call for help!

Where to from here?

Now that you have begun to see trends in performance stress indicators, and after practicing with SURVEYOR, you may want to make it a regular habit to monitor your system in batch. This way you will be able to get a feel for healthy and unhealthy trends in the life of your system. A simple way to run SURVEYOR in batch mode is:

```
!JOB SURVJOB,MGR/pass.TELESUP/pass

!RUN SURVEYOR.PRV.TELESUP;INFO="RUN=480;DELAY=600"

!EOJ
```

This job will execute for eight hours (the "480" is in minutes), with a collection report generated every 10 minutes (delay is in seconds). If you need to monitor a shorter or longer period, you may wish to alter the "RUN" and "DELAY" parameters accordingly.

Interpretation guidelines

Here are some things to keep in mind, and to look for as you monitor your system on a regular basis:

1) CPU busy and idle time are two indicators to look at first. If there is not idle time either a batch job is running, or your system is quite probably overloaded. Remember, since batch jobs do not have "think time" like human user, they tend to skew the idle time a bit by taking whatever CPU is not being used by interactive sessions. Though it depends on your particular growth plans, philosophy of DP management, and so forth, it's nice to have a margin for growth in terms of CPU availability. Total CPU busy on sessions above 75 percent for a consistent period of time will begin to produce negative implications for response times and transaction throughput.

2) As the memory clock rate decreases, the system is experiencing increased stress in finding free memory space for process demands.

3) As the CPU Pause on MAM increases, this can be another indicator of memory shortage. The memory manager must make disc requests to obtain necessary missing segments; we know by now how spell disc I/O: S-L-O-W. The same is true for CPU pause for MAM and User I/O, though it is difficult to tell with SURVEYOR how much is devoted to each of these.

4) If Disc Caching is enabled on your system at the time of a performance study, you will tend to see either low or zero numbers under Paused For User Disc I/O. Remember, you are not getting anything for free with Disc Caching enabled. You are simply exchanging resources: efficient, plentiful ones versus inefficient, scarce ones. So when caching is on, you tend to see the amount of time the CPU spends on pausing for user I/O decrease dramatically, but the CPU busy on Cache increases. And memory usage increases also. This is why a couple of the criteria for determining whether or not caching will help a system are: a) is there excess CPU? and b) is there excess memory? Caching usually is ordered with an extra megabyte of memory for this reason.

5) If garbage collection is enabled on your system, and some number other than zero shows up under that label on the SURVEYOR report consistently, it is a good sign of memory stress. It is important to realize that you may indeed be experiencing memory problems and *never* have Global garbage collection show up. This is because garbage collection is enabled on your system only if the following conditions are true:

   a) The system has some idle time

   b) Disc Caching is NOT enabled and you have two or more megabytes of memory

   c) The system is experiencing memory pressure

   So, if you have less than 2 megabytes of memory and Disc Caching is enabled, you may still see a number under "Global Garbage Collection." If you have more than 2 megabytes of memory and Disc Caching is disabled, you may see a number here, only if memory stress is being experienced.

Summing it all up

If I were starting all over in this field, I would begin by running SURVEYOR (or an inexpensive alternative) and simply observe patterns. I would try to correlate what I see on the reports with my workload. Perhaps setting up a few programs which could be run in batch and comparing completion times with various configuration settings would provide you with a feel for your system's "pulse." Try deconfiguring a megabyte of memory via SYSDUMP and then running the same programs. This will exaggerate a memory problem. Then do various runs with Disc Caching on/off and varying sequential and random fetch quantum values.

You will succeed in taking charge of system performance to the degree that you understand performance theory and gain experience observing real live performance data - and lots of it.

Appendix **B**

A Disc Space Management Game Plan

This article appeared in the October 1986 issue of Interact Magazine by the author.

One of the ongoing concerns of managing system resources is providing adequate disc space for applications to run efficiently. In fact, lack of contiguous disc space can negatively affect system performance.

From time to time a FREE5 should be run to monitor the amount of free disc space. This is accomplished with the simple command:

```
:RUN FREE5.PUB.SYS
```

The display that follows provides a chart for each disc drive telling you the total system free space, the total free space for each drive, and how fragmented each disc is.

The purpose here is to provide you with an explanation of disc space management and a game plan. FREE5 is explained in the *MPE Utilities Reference Manual* which I encourage you to read before continuing.

First, a little explanation of how MPE interacts with disc space. MPE operates most efficiently when it has large areas of contiguous free space available on disc. The larger numbers in the >10000 or >100000 rows in the FREE5 listing indicate a healthy amount of available space (for example, if the >10000 row reads as in Figure B.1).

Figure B.1 shows two chunks of free space of an average size of 25,052 sectors. (Remember, a sector is a measurement of disc space equal to 256 bytes. So to figure free space in terms of bytes, simply multiply the total free space for that drive by 256. Thus, 88556 times 256 equals 22,670,336 bytes or 22.67 Megabytes.) The total space in the 10000 to 100000 category is 50104 sectors.

Generally the numbers in the "count" column become larger as you go down the column. The larger numbers toward the bottom of the count column tend to indicate an increasingly fragmented system. That is, many little pieces are available but no large ones are.

```
                SIZE  COUNT  SPACE     AVERAGE
              >100000 0      0         0
----------->> >10000  2      50104     25052
              >1000   12     25062     2088
              >100    28     8695      310
              >10     122    4020      32
              >1      181    675       3
              TOTAL FREE SPACE=88556
```

Figure B.1 - FREE5 Listing

Also a large piece is sometimes necessary to perform a particular operation. For example, using some database capacity changing utilities requires a lot of free space. Sometimes you will not be able to resize the database because MPE could not find enough room to complete the task. The following line with asterisks serves to illustrate this business of contiguous space. If each asterisk below represented one sector of data on disc, and each period is free space, a simple representation might look like this:

```
..*.**.*****..***.**.*.*.*.***.****..*.......**********.**.
```

If we somehow could "scrunch" all the free space into one big contiguous section, it would look like this:

```
<           free space          ><           file space     >
.................................*****************************
```

This is exactly how your disc drives look after you perform a full system reload.

Solutions for a Fragmented System

Now to solve the problem of a fragmented system. There are five alternatives to choose when dealing with this situation. The first two are merely temporary solutions and don't really solve the fragmentation problem, but they will probably get you through a crisis situation. Option one is easiest and most temporary, and number five, which takes longest, is the most effective.

The solutions:

1) Purge off as many files as you possibly can. This is not a great solution especially if the files you are purging are very small. You may want to follow with step four if you didn't gain enough contiguous space. It's a good idea to store to tape anything you will permanently purge.

2) Store off to tape large databases and files you don't need for the moment, perform your operation, and then restore the files back onto the system.

3) Perform a RECOVER LOST DISC SPACE by doing a coolstart. (See the HP SYSTEM OPERATION AND RESOURCE MANAGEMENT REFERENCE MANUAL on how to do this.) This does not really do any condensing per se. It merely recovers space that may have been lost when the system went down (spoolfiles and temporary files). So if your system has not failed recently (since you've done a reload or RECOVER LOST DISC SPACE), this will take less than four minutes per 1000 files and may gain you absolutely nothing. Sometimes, however, this option will gain just enough free space to solve your problem. Just be warned; it takes time.

4) Do a VINIT CONDENSE. This is a slick way to solve your fragmentation problem without having to reload. This operation can be performed only under the following conditions:

   a) A full backup is done first and everyone is off the system

   b) You have reasonable assurance you have no outstanding problems with the power company or power conditioner (I realize you cannot realistically know this, but if the power company says it will be monkeying with the power overnight, don't do a VINIT). You must be reasonably sure your system is stable; that is, you haven't had any hardware/software problems and you haven't had any system failures recently

   Vinit is actually moving files around the system and updating the directory - a very delicate operation. That is, if a system failure or power failure were to occur, or someone were to log onto the system (I don't even think the system would allow this) and purge or attempt to access files, it's likely directory corruption would occur.

   The procedure for doing a condense with VINIT is:

   a) Perform a full backup

   b) Get a FREE5 listing (for a before and after scenario)

   c) Log on as MANAGER.SYS or OPERATOR.SYS

   d) Make sure all users are off the system; set limits and the job fence accordingly to keep folks and jobs off

   e) Type VINIT. When you get a > prompt, type CONDENSE ldn. You must do this for each disc drive on the system, replacing ldn with the logical device number for that particular device

   f) Get a FREE listing once again

If VINIT is to do its stuff, it needs some contiguous space to use as a buffer. So if you are really fragmented, you might perform the steps just outlined and notice little or no difference in the FREE listings from steps b and f. If that's the case, doing steps e and f once again will probably do the final cleanup. This will happen only in rare cases, however. In really bad situations, your only recourse is to plunge into step five.

5) Do a reload. This is no fun. It is expensive, because somebody has to babysit hanging tapes.

6) A technique to avoid steps 1 through 5 is to address the problem before it occurs. One way is to create a large disc file on each disc device with the BUILD command. This allows you to purge such files if you run out of disc space. The drawback is that space is always tied up as a wasted file. The following command will build a file of 40,000 contiguous sectors on ldev 1:

```
BUILD BIGFILE;DISC=40000,1,1;LDEV=1
```

If you really want to get fancy and you want to avoid a reload, doing a RECOVER and a VINIT CONDENSE accomplish essentially the same thing as a reload. These options would also be faster.

Now that you have a plan of attack against fragmentation, here's when to perform these options:

1) If system response time is getting progressively slower and you haven't added any more users or jobs **AND** a FREE5 listing exhibits a paucity of large free pieces of disc space.

2) As a part of your weekly backup schedule, you might consider doing a VINIT CONDENSE just before going home; that is, let it condense while you are not there. (Perhaps you could do one disc each evening or week, depending on your batch job scheduling, staff, and so forth.)

3) After you experience a system failure or a halt. (It's best to do a RECOVER LOST DISC SPACE shortly afterward to regain any space that might have been lost to spoolfiles or temporary files.

VINIT and RECOVER LOST DISC SPACE

Let's examine a real-life run of VINIT and RECOVER LOST DISC SPACE. The example given was performed on an HP 3000 Series 37 operating under MPE G.01.03 (T-DELTA-3). Figure B.2 is the freespace "before" snapshot to be used as the basis for comparison after condensing and recovering.

First we'll do a RECOVER LOST DISC SPACE. This is accomplished by performing a coolstart and choosing the RECOVER LOST DISC SPACE option (remember, this is going to take some time and may not gain you any space). The FREE5 listing in Figure B.3 is our result.

```
FREE5 G.01.00 (C) HEWLETT-PACKARD
VOLUME  MH7914D2            LDEV 2
LARGEST FREE AREA= 21599
  SIZE  COUNT  SPACE   AVERAGE
>100000 0      0       0
>10000  2      35867   17933
>1000   0      0       0
>100    1      205     205
>10     8      166     20
>1      249    399     1
TOTAL FREE SPACE=36637
```

Figure B.2 - Free Space "Before" Snapshot

```
FREE5 G.01.00 (C) HEWLETT-PACKARD
VOLUME  MH7914D2            LDEV 2
LARGEST FREE AREA= 35903
  SIZE  COUNT  SPACE   AVERAGE
>100000 0      0       0
>10000  1      35903   35903
>1000   0      0       0
>100    6      2026    337
>10     23     1162    50
>1      245    395     1
TOTAL FREE SPACE=39486
```

Figure B.3 - Free Space "After" Snapshot

```
FREE5 G.01.00 (C) HEWLETT-PACKARD
VOLUME  MH7914D2            LDEV 2
LARGEST FREE AREA= 38529
  SIZE  COUNT  SPACE   AVERAGE
>100000 0      0       0
>10000  1      38529   38529
>1000   0      0       0
>100    1      180     180
>10     10     328     32
>1      272    449     1
TOTAL FREE SPACE=39486
```

Figure B.4 - Results of a VINIT CONDENSE

The main thing to note, comparing before and after (Figures B.2 and B.3), is that the TOTAL FREE SPACE line has increased from 36,637 sectors to 39,486; this is space that had been essentially unavailable to MPE for use because of a system failure or halt. Before a failure, spoolfiles and temporary files are flagged as used space. When the system is restarted, the space those files occupied is not released for use by MPE (unless you perform a WARMSTART and purge spoolfiles). The flags for those files are reset only by doing a RECOVER LOST DISC SPACE or a full system reload. So a RECOVER LOST DISC SPACE does not really compress the files on disc although sometimes it may look that way (compare the 10000 line in figures B.2 and B.3). It resets the "used space" flags for temporary files and spoolfiles and therefore regains any lost space.

Now let's see the effect of a VINIT CONDENSE. Figure B.4 is the result of condensing that disc drive. Notice that the TOTAL FREE SPACE line for Figures B.3 and B.4 is unchanged. A condense of a disc drive will not regain lost space but will attempt to consolidate all the free space into a large single area. The LARGEST FREE AREA line is changed, however. And as you compare the other SIZE lines, you will notice a further consolidation up toward the 10000 line.

Now let's run VINIT again and see if some of the smaller portions of free space can be packed up into the larger pieces. Figure B.5 is the result of running VINIT CONDENSE again. Figure B.5 shows we haven't gained anything. A mere "reshuffling of the deck" has occurred between the 1 - through 100-sector region. Apparently the VINIT designers thought a thousand or so sectors wasn't worth the addition of logic to handle these small pieces.

```
FREE5 G.01.00 (C) HEWLETT-PACKARD
VOLUME  MH7914D2              LDEV 2
LARGEST  FREE AREA= 38529
  SIZE   COUNT  SPACE   AVERAGE
>100000  0      0       0
>10000   1      38529   38529
>1000    0      0       0
>100     1      104     104
>10      12     404     33
>1       272    449     1
TOTAL FREE SPACE 39486
```

Figure B.5 - Results of Running VINIT CONDENSE Again

In this example, the results aren't dramatic; we started with a small disc drive that wasn't very fragmented to begin with. I encourage you to try this on your system and keep a running scorecard of how fragmented the drives become over time. After performing a number of condenses, you will begin to get a feel for how often you should run VINIT.

Managing disc space can be bothersome if you are not prepared to monitor it consistently. But a very fragmented system can affect system performance. By keeping on top, you will be able to eliminate disc fragmentation; a thief of precious disc resource.

Appendix C

XL/3000 Ramdisc

XL/3000 is a relatively new product for the HP 3000 that could be a step beyond disc caching. Finally, someone has taken advantage of rapidly dropping memory prices to do what has been done at the microcomputer level for some time now; that is, produce a RAM memory pseudo disc drive. This author beta tested the product prior to the first release version and noted incredible disc I/O rates for a "disc" drive.

In reality, this disc drive is simply a grouping of random access memory chips seen by MPE as a private volume. The memory, which is nothing other than regular main storage, is configured above that which is reserved for MPE's use (as noted in SYSDUMP). By using memory as a data storage device, transfer rates of up to ten times that of a disc drive are possible. This is possible because memory access time is performed in microseconds as opposed to the 25 to 30 milliseconds associated with conventional disc drives. Though MPE "sees" it as a disc device, it is merely RAM memory disguised as a private volume.

FEATURES OF XL/3000

The RAMDISC boasts of the following features:

1) File reads are instantaneous. The processes accessing these files do not have to wait for the disc drives to locate the data and therefore do not give up the CPU on account of pausing for an I/O to be serviced. If a file resides on the RAMDISC then the access takes place almost immediately.

2) Reads from the RAMDISC are actually faster than reads from a disc cache domain in memory. XL/3000 performs a single arithmetic calculation to determine the memory address of a desired sector, resulting in a fifty percent improvement in reads per second vs. disc caching.

3) Writes to a file in the RAMDISC are virtually instantaneous. Product evaluators have observed write rates of over 400 per second.

4) More control is given to the user. Disc Caching contains a finite amount of disc data, while a RAMDISC is capable of containing, say, an entire IMAGE database.

5) XL/3000 provides performance and capabilities beyond Disc Caching in four ways:

   a) The user can guarantee that a file will be present in the RAMDISC.

   b) The user has control over XL/3000 to a level greater than just the LDEV number as in the case of disc caching.

   c) There are virtually no MPE resources required to manage the RAMDISC.

   d) The XL/3000 RAMDISC is able to provide storage size larger than that of Disc Caching (the author tested it with twenty four megabytes for the RAMDISC and eight megabytes for MPE on a series 70).

RAMDISC APPLICATIONS

Some of the possible applications for XL/3000 are listed below:

1) Session temporary "tag" files are built on the RAMDISC if a file equation directs them to the RAM group of the logon account, or if the RAM group is the one that the job logged onto initially. Data is read from and written to permanent hard disc resident files while the session temporary work file enjoys the benefit of lightning fast I/O throughput.

2) The author has seen program compile times decrease by thirty to fifty percent.

3) Sort scratch files, resident in the RAMDISC have caused sort times to decrease by as much as fifty percent.

4) Kelly has claimed that office productivity tools, such as graphics intensive applications, spreadsheets, and editors have seen greatly improved response times when utilizing XL/3000 as compared to hard discs.

REAL LIFE TESTS

In order to observe XL/3000's impact under various system stress situations, the author ran a series of tests. The tests consisted of observing XL/3000 under CPU, memory, and disc I/O stress situations. In every case, 25 simulated sessions and two batch jobs were run on a series 70 with eight megabytes of MPE memory and twenty four megabytes of RAMDISC memory. The CPU test consisted of intense calculations and short user think times with light memory and I/O loads. Memory stress was induced by large stacks and extra data segments; light CPU and I/O loads were induced. I/O load was simulated by requesting many times more I/Os per second than the two disc drives could possibly keep up with. In every case IMAGE data base access was occurring. All database and temporary files resided in the

RAMDISC for the XL/3000 test. Each of these stress cases was tried with Disc Caching only, no disc caching/no RAMDISC, and RAMDISC only. Note the total number of transactions completed in each case. Figure C.1 displays the test results.

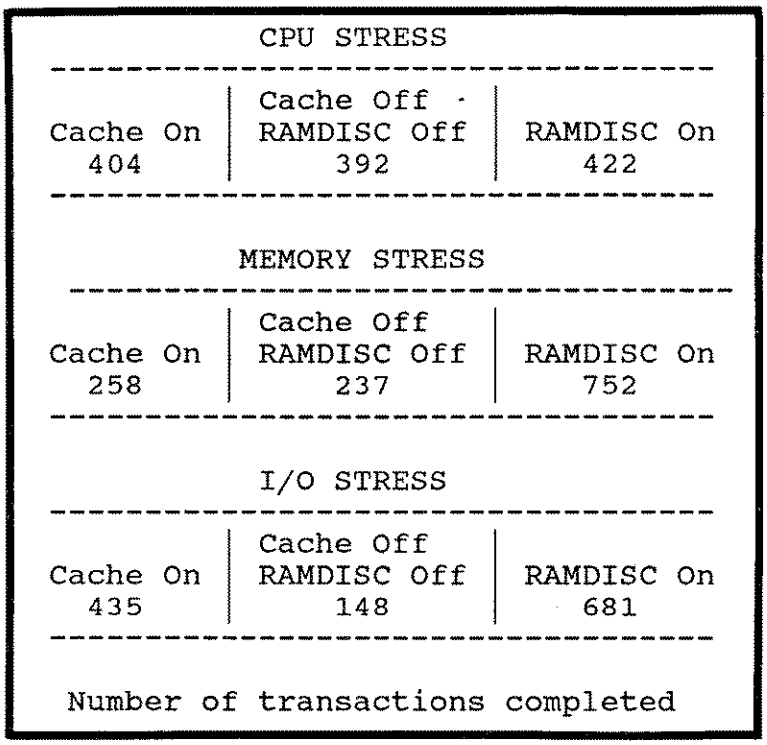


Figure C.1 - XL/3000 Performance Under Various Resource Stresses

SOME CONCERNS

XL/3000 has tremendous potential for many HP 3000 shops. As seen from the brief study above this statement is fairly clear. However, a few concerns do surface when discussing implementation and day-to-day management of the RAMDISC device.

First, it will take some up-front thinking and planning to make the most of XL/3000. Unlike Disc Caching that virtually does it all for you (this has good and bad points as discussed above), the RAMDISC requires that you decide which files are going to be ram-resident. Of course you can change your plan any time, but to fully utilize the amount of extra memory that comes with the RAMDISC, you will want to make the most of it.

Second, if at any time you exceed the RAMDISC memory space, an OUT OF DISC SPACE message will result along with the normal implications of this problem. Some way for operators and other staff to deal with this in a timely manner would need to be worked out so that the overall time wasted in recovering is minimal.

Third, Though the RAMDISC gives more control than disc caching, this may not be as desirable as one would think. Again, this is a managing issue. Some shops try to be as "turn key" as possible. These type of environments may not be a great candidate for the RAMDISC in some situations due to the moderate amount of managing that it needs to be given.

One last concern is the fact that the RAMDISC is not cheap. Of course, one would need to look at how much I/O was needed in order to optimize a system, and then cost justify the purchase based on those figures.

CONCLUSION

In summary, XL/3000 has broken new ground for disc I/O performance on the HP 3000. Though not a new concept, it is new to the world of the HP 3000. Since disc I/O is typically a common bottleneck on many HP 3000s, one would expect a noticeable, overall increase in transaction throughput and a decrease in average terminal response times by increasing disc I/O rates. Indeed, the author has found this to be true. It will most assuredly take some coordination between processes utilizing the RAMDISC to insure that job interrupts do not occur due to RAMDISC storage being overflowed. Managers and operations personnel will need to plan and strategize on how to best utilize XL/3000 in their unique environments. In spite of these concerns, it is expected that XL/3000 has opened a new door for performance improvement on I/O bound HP 3000s.

Bibliography

RESEARCH REFERENCES

Armstrong, Melody. "Disc Controller Cache." *Interact* (Sept. 1987): 106ff.

Beasley, Dave. "How Dispatching Queues Really Work." *Proceedings of the Interex HP 3000 Amsterdam Conference* (March 1985): 786-796.

Blake, Isaac. "System Performance and Optimization." *Interact* (April 1984): 47-50.

BradMark Computer Systems. "Do synonyms degrade performance?: In theory, yes, but what about reality?" *IMAGERY Newsletter* (July/Aug 1986).
_____. "Optimizing detail sets: What are the primary concerns?" *IMAGERY Newsletter* (Sept/Oct 1986).
_____. "Optimizing master sets: Is the solution prime?" *IMAGERY Newsletter* (July/Aug 1986).
_____. *DBGENERAL User's Manual.* California: BradMark, 1987.

Carolian Systems Intl. *SYSVIEW Reference Manual.* Ontario, Canada: Carolian Systems Intl., 1987.

Carroll, Bryan. "Performance Programs and the Measurement Interface." *Proceedings of the 1984 IUG Anaheim, Calif. Conference* (Feb. 1984): 28-1/28-22.

Cooper, Steven M. "Variations On A Tune - Another Look at the Never-Ending Struggle Towards Optimal Performance." *Proceedings of the Interex HP3000 Madrid Conference* (March 1986): 415-425.

Dowling, James F. "An Investigation Into the Effects of Memory Availability on Performance for HP 3000 Systems." *Volz Associates System Performance Handbook* (May 1987).
_____. "Data Center Management: A Methodology For HP 3000 System Performance Management, Part 1." *HP Professional* (May 1987): 42ff.
_____. "Data Center Management: A Methodology For HP 3000 System Performance Management, Part 2." *HP Professional* (July 1987): 64ff.

Duncombe, Brian. "Performance Self-Analysis." *Proceedings of the Interex HP 3000 IUG Amsterdam Conference 1985* (March 1985): 819-832.

Engberg, Tony. "Response Time: Defining it and Speeding it Up." *Interact* (April, 1986): 48ff.

Freeman, Stan. "Capacity Planning on the HP 3000." *Interact* (June, 1986): 56ff.

Green, Robert. "HP 3000: Optimizing Batch Jobs." *SCRUG Letter* (May 1981).

Greer, David. "How Messy is Your Database?" *Interact* (April 1986): 18ff.
_____. "The Price of Empty Space in Detail Datasets." *The Chronicle* (Nov. 1986): 47.

Heidner, Dennis and Russell, Marguerite. "The TurboIMAGE Supplement." *The IMAGE/3000 Handbook.* Seattle: Wordware, 1987.

Hewlett-Packard. "HP 3000 Application Note #30: Disc Cache." *North American Response Center Application Notes* (June 1987) P/N 5958-5824R2725.
_____. *Disc Performance Reference Guide for HP3000 Systems.* Idaho: Hewlett-Packard, 1987. P/N 5953-3680.
_____. *HP 7933XP HP 7935XP Application Engineering Manual,* California: Hewlett-Packard, 1986. P/N 5953-3672.
_____. *HPTREND User's Guide.* 1987. P/N 35136-90001.
_____. *Performance Guide.* 1984. P/N 5954-0401.

Jordan, Arthur. "Application Performance Tuning." *Proceedings of the Interex HP 3000 Detroit Conference 1986* (March 1986): 1-29.

Kemp, Larry. "Predicting System Performance." *Interact* (Sept., 1987): 145-150.

Kramer, Jim. "Series 58 Performance." *Proceedings of the Interex HP 3000 Madrid Conference 1986 (March 1986): 549-557.*

Lund, Robert A. "A UDC Primer." *Interact* (April 1987): 71ff.
_____. "Disc Space Management." *Interact* (Sept. 1986)
_____. "Taking Charge of Performance: The Basics." *Interact* (Sept. 1987): 82ff.

May, Jim. "Programming For Performance." *Publication specifics unknown.*

Merit, David. "Convert to TurboIMAGE in hours, not minutes." *Interact* (Feb. 1987): 34ff.
_____. "IMAGE Performance Myths." *HP Professional* (July 1987): 58ff.

Muntean, Mark and Boles, Sam. "Structured Tuning: Man's Interface to the Machine's Performance Components." *Proceedings of the Interex HP 3000 Amsterdam Conference* (March 1985): 887-896.

Overman, James S. "Improving Application Performance Without Changing Your Programs."*An Unpublished paper given to the author.*" (1986).

Primmer, Paul. "HP Labs Systems Performance Evaluation Project." *Proceedings of the Interex HP 3000 Amsterdam Conference* (March 1985): 897-904.

Robelle Consulting Ltd. *HOWMESSY User Manual.* Canada: Robelle Consulting Ltd., 1986.

_____. *What's UP, Doc? Special Edition* (1986).

Russell, Marguerite. *The IMAGE/3000 Handbook.* Seattle: Wordware, 1985.

Shumko, Green, Greer. "What If...You Didn't Wait for Spectrum?" *HP Professional* (July 1987): 82ff.

Smith, Guy. "Chase busy files to spread HP 3000 load." *The Chronicle* (Dec. 1986): 45-47.

_____. "Database Housekeeping and Configuring Tips." *The Chronicle* (Nov. 1986): 40ff.

_____. "Performance: Beating a Dead Workhorse." *The Chronicle* (Oct. 1986): 45-47.

Smith, Sid. "The Series 70: What is it? What can you expect from it?" *Unpublished paper given to the author* (1986).

Strategic Systems. *PROBE/3000 User's Manual.* Seattle: Strategic Systems, 1987.

Tauber, Andy. "Disc Balancing." *Interact* (Jan. 1986): 60ff.

Trasko, Mark S. "The Future of Database Technology." *Supergroup* (July 1986): 24ff.

Trasko, Mark. "Maximizing Database Performance." *HP Professional*, (Sept. 1987): 44-49.

Van Dijk, Rob. "The poor Man's Performance Measurement." *Proceedings of the Interex HP 3000 IUG Amsterdam Conference 1985* (March 1985): 813-818.

Van Valkenburgh, R.E. "Improving Your Performance." *Interact* (Sept. 1987): 31ff.

VESOFT. *MPEX User's Manual.* Los Angeles: VESOFT, 1987.

Volz Associates. *System Performance Management Seminar Notebook.*

VENDOR REFERENCES

Adager - Apartado 248 Antigua, Guatemala +502(2)324333. **ADAGER - Database utility.**

Atlantis Software - 348 E. Temple, Salt Lake City, Utah 84111 (801) 521-3000. **WATCHDOG - System Optimizer.**

BradMark Computer Systems - One Towne Center, Audubon Parkway/Buffalo, N.Y. 14228 (716) 689-6882. **DBGENERAL - Database utility.**

Carolian Systems - 3397 American Dr. #5, Mississauga, Ontario L4V1t8 Canada (416) 673-0400. **SYSVIEW-Performance Monitor, SYSPLAN - Capacity Planner.**

CSL - Contributed Software Library - Interex, The International Association of Hewlett-Packard Computer Users. 680 Almanor Ave., Sunnyvale, Calif. 94086 (408) 738-4848. **Contributed Software Library Tapes.**

Design/3000 - Salem, Oregon (503) 585-0512. **JMS/3000 - Job scheduler.**

Dynamic Information Systems (DISC) - 910 15th. St., Suite 640, Denver, Co. 80202 (303) 893-0335. **DBMGR - Database utility, OMNIDEX - Fast IMAGE accessor.**

EMC^2 Corp. - Hopkinton, Ma. 01748-9103 (800) 222-EMC2. **Falcon disc drives.**

HI-Comp - 11 Cranberry St., Brooklyn, N.Y. 11201 (800) DBETUNE. **DBTUNE - Database utility, HI-BACK - Fast Store Utility.**

Kelly Computer Systems - 1101 San Antonio Rd. Suite 419, Mountain View, Calif. 94043 (415) 960-1010. **XL/3000 RAMDISC.**

KLA & Assoc. - Box 14854 Clearwater, Fla. 34629-4854 (813) 784-5976. **KLA/EXPRESS - System Optimizer.**

Operations Control Systems (OCS) - 560 San Antonio rd., Palo Alto, Calif. 94306 (415) 493-4122. **OCS/3000 - Job scheduler.**

OPT - 299 W. Foothill Blvd., Suite 230, Upland, Calif. 91786 (800) 858-4507. **QSORT - Sort utility.**

Orbit Software - 175 5th Ave. Suite 2274, New York, N.Y. 10010 (212) 538-2616. **ONLINE BACKUP - Store utility.**

Robelle Consulting Ltd. - 8648 Armstrong Rd., R.R. #6, Langley, B.C. Canada V3A409 (604) 888-3666. **SUPRTOOL - Database utility, HOWMESSY - Database diagnostic.**

Running Mate - Box 160488, Sacramento, Calif. 95816-0488 (800) 824-9046. **SORTMATE - Sort Utility.**

Strategic Systems - 10502 11th Ave. N.E., Seattle, Wash. 98125 (206)325-3309. **PROBE/3000 - Performance monitoring tool.**

Tymlabs - 211 E. 7th St., Austin, Texas 78701 (512) 478-0611. **BACKPACK - Store Utility.**

Unison - 415 Clyde Ave., Mountain View, Calif. 94043 (415) 968-7511. **MAESTRO - Job scheduler.**

VESOFT - 1135 S. Beverly Dr. Los Angeles, Calif. 90035 (213) 282-0420. **MPEX - Extended MPE utility.**

Wordware - Box 14300, Seattle, Wash. 98114 (206) 625-0609 - **The IMAGE/3000 Handbook with TurboIMAGE Supplement.**

About the Author

Robert Lund has worked with the HP 3000 since 1979 in various capacities such as: Programmer/Analyst, System Manager, Systems Engineer (Hewlett-Packard, Fullerton, California), Technical Support Engineer for a large Hewlett-Packard VAR, and a System Performance Consultant.

While working for Hewlett-Packard as a Systems Engineer, Robert taught numerous customer courses, and spent time working in the HP Santa Clara Response Center. Robert also taught Business Computer Systems at Santa Ana College in Santa Ana, California.

He has written numerous technical articles for INTERACT magazine. Robert has provided technical support to over 150 HP 3000 sites throughout the United States. He and his wife and two daughters live in rural Oregon.

You may correspond with Robert by writing the publisher at:

Performance Press
P.O. Box 151
Albany, Oregon 97321
(503) 327-3800

FEEL FREE TO CUT THESE AS NECESSARY.

HEY! I WANT MY OWN COPY OF TAMING THE HP 3000.

TO ORDER BY PHONE - CALL (503) 327-3800

In the US: $29.95 per book plus $2.95 shipping/handling

Outside the US: $29.95 per book plus $4.95 shipping/handling
Orders must be paid in advance.

☐ Check Enclosed
☐ Bill Company: Purchase Order No.: _____
☐ Credit Card: ☐ Master Card ☐ VISA
Credit Card No.: _____ Exp. Date: _____
Signature: _____

NAME: _____
COMPANY: _____
ADDRESS: _____
_____
CITY: _____ STATE: _____
POST/ZIP: _____ COUNTRY: _____
PHONE( ): _____

QTY.	DESCRIPTION	PRICE	TOTAL
	Taming Book	$29.95	
	US S/H	$2.95	
	Outside US S/H	$4.95	
		TOTAL DUE	$

SEND TO: PERFORMANCE PRESS P.O. BOX 151 ALBANY, OR 97321 (503) 327-3800

TO ORDER BY PHONE - CALL (503) 327-3800

In the US: $29.95 per book plus $2.95 shipping/handling

Outside the US: $29.95 per book plus $4.95 shipping/handling
Orders must be paid in advance.

☐ Check Enclosed
☐ Bill Company: Purchase Order No.: _____
☐ Credit Card: ☐ Master Card ☐ VISA
Credit Card No.: _____ Exp. Date: _____
Signature: _____

NAME: _____
COMPANY: _____
ADDRESS: _____
_____
CITY: _____ STATE: _____
POST/ZIP: _____ COUNTRY: _____
PHONE( ): _____

QTY.	DESCRIPTION	PRICE	TOTAL
	Taming Book	$29.95	
	US S/H	$2.95	
	Outside US S/H	$4.95	
		TOTAL DUE	$

SEND TO: PERFORMANCE PRESS P.O. BOX 151 ALBANY, OR 97321 (503) 327-3800

HERE'S WHAT SOME FOLKS ARE SAYING ABOUT TAMING THE HP3000...

"Taming the HP3000 is valuable reading for both the new and experienced HP3000 System Manager. I highly recommend that every HP3000 shop obtain a copy for their technical library."

—Ken Vestal, President Design/3000

"Most Data Processing Managers in today's HP3000 enviroment would agree that system performance is a major effort in any size shop. While no "off the shelf" reference is going to actually do the work for you, one definitely motivates you to do it for yourself! I'm referring to Robert Lund's book, **Taming The HP3000**. The approach is methodical and detailed, yet reads with ease and simplicity. The suggestions are practical ideas that encourage the reader to try them out first-hand. This becomes your motivation as a manager, to decide on their effectiveness, and I can honestly say that many are effective. The section on disc caching was particulary helpful and the IMAGE chapter on masters and details also awarded me considerable performance gains. I enjoyed this book because I could not read it cover to cover! I was too busy stopping to try out a new idea or examine my configurations. **Taming The HP3000** could aptly be renamed **Unleashing the System Manager** because it highlights the possibilities of a finely tuned system; something everyone can benefit from. Robert Lund is a motivator! The goal is reachable and **Taming The HP3000** provides the incentive and direction necessary in attaining "optimum system performance".

—Creighton McCartin, Data Processing Manager, American Baptist Credit Union

"The ideas presented in this book have proven to be very valuable in my position as client support manager of an HP VAR. I recommend it highly to anyone who has a system to tune but has limited time and resources."

—William Lancaster, Advanced Laboratory Systems

"I found it to be informative as well as entertaining. It would be very helpful for the user and System/Manager with limited knowledge, without being intimidating, but also very useful for the User/System Manager who has a very broad knowledge of the system."

—Diana Vaughan, Operations Supervisor, Island County Computer Services

"Taming the HP3000 is an excellent step-by-step introduction to diagnosing and resolving system bottlenecks. This book explains the practical side of performance measurement and tuning. This book is full of examples, illustrations and exercises on how to manage a system with performance in mind. Most of the analyses do not require special software. Taming the HP3000 is both informative and useful, does a very good job of identifying and analyzing performance components. He then gives explicit recommendations for both general and specific performance optimization. Every system manager should be exposed to this information."

—Larry Kemp, Systems Consultant, Hewlett Packard Company

"Bob's book was helpful in our limited budget environment with the suggestions on how to use contributed software to monitor the system. The IMAGE hints were easy to understand and follow. I would recommend the book to novice and experienced HP3000 users."

—Margie Worden, Benton County Central Services

SHOULDN'T YOU HAVE YOUR OWN COPY?

Write: **Performance Press**
P.O. Box 151
Albany, Oregon 97321
(503) 585-3972
(or see the order page)

ISBN 0-945325-01-0