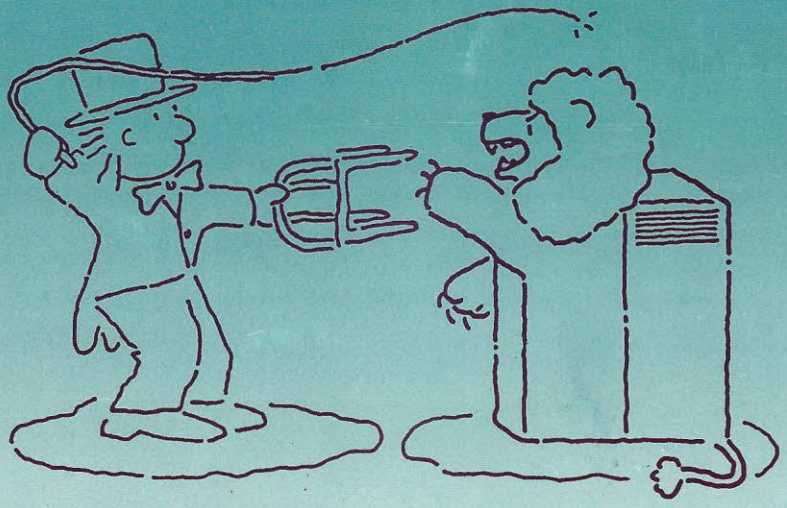
volume two

THE TAMING HP 3000

The Theory and Practice of Successful Performance Management for Hewlett-Packard HP 3000 Computer Systems



by Robert A. Lund

TAMING THE HP 3000

Volume II

TAMING THE HP 3000

Volume II

The Theory and Practice of Successful Performance
Management for Hewlett-Packard
HP 3000 Computer Systems

Robert A. Lund

Performance Press Publishing
Albany, Oregon 97321

TAMING THE HP 3000 - Volume II
The Theory and Practice of Successful Performance Management for
Hewlett-Packard HP 3000 Computer Systems

Manufactured in the United States of America

First edition 1992

Manuscript preparation by Performance Press, Albany, Oregon.
Printing by Commercial Bindery Werks , Salem, Oregon.

ISBN 0-945325-02-9 (cloth)
0-945325-03-7 (paper)

Library of Congress card catalog number: 87-092025

First Printing 8/92 - 200
Second Printing 9/92 - 1800

Printing: 9 8 7 6 5 4 3 2 Date: 92 93 94 95

For ordering information, call or write the publisher at:
Performance Press, P.O. Box 151, Albany, Oregon 97321 (503) 327-3800 Fax (503) 327-3276

To My Lord Jesus Christ,
Whom, I Found Out,
Actually Cares About HP 3000 System Performance,
To A Certain Degree,
Because He Cares About Me;
And Who Has
Unceasingly Poured Out On Me,
Joy, Purpose, And Life,
This Volume Is
Gratefully Dedicated.

Foreword

The subject of performance analysis on commercial Hewlett-Packard computer systems is something I have been involved with for the last 12 years. Being somewhat of a philosopher at heart and an observer of the species *homo sapiens*, it has often fascinated me to watch the myths, misconceptions, and plain incorrect information that has circulated regarding HP 3000 performance.

These incorrect perceptions have not been limited to just end-users and System Managers either. HP SEs, CEs, marketing types, sales reps, independent consultants, and many others, who really should know better, have fallen prey to the mass of information about the subject. This mass seems at times too large, at other times too small, and most of the time too confusing to have any sense made of it by mere mortals with IQs of 150 or less.

Fortunately for all those mentioned above, Bob Lund has taken a giant leap toward cutting through the dense fog surrounding the subject of HP 3000 Performance Consulting. This book is the best (heck, the only) concise source of information that exists to my knowledge. Certainly there are myriads of articles, talks, and papers that have been presented over the years that cover much of this information. But never has this material appeared under one cover, presented in a logical sequence, and with the assumption that the reader *does not* already have a Ph.D. in Computer Modeling.

A fundamental assumption that Bob covers very well is a question most of us have overlooked totally, that is, "Why should I care?" The first part of this book covers the reasons why performance analysis and modeling are vital activities in today's HP 3000 shops, and something that *every* System Manager should be able to perform. As I read through the book, I wondered why Bob spent so much time making a seemingly obvious argument to the reader. But as I read on, I recognized the truth in this course. So many people I have consulted for did not perform regular analyses, and got themselves into a big jam that required a hired "gun" to come and fix the problems. Bob, I realized, has had the exact same experience too many times, and saw the need to preach this gospel. For the first time in print, that I can recall, his discussion of the important question of Why? is addressed.

The other thing that impressed me about the book is his treatment of the 16-bit Classic HP 3000s. Bob, being a pragmatist who lives in the real world, recognizes that there are still a large number of Classic machines running out there, doing very useful work with great cost effectiveness, given the current resale prices of "huge" Series 70s and other high end machines. This is in stark contrast to the so-called "Guru's" at HP, both "labby", field SE, and marketing types alike, who were referring to MPE V and the 16-bit 3000s in the past tense five years ago, even when MPE XL was barely able to boot by itself. Far from writing off the Classic, Bob gives even more tips to wring the last microgram of performance from this tried-and-true architecture, thus breathing new life into the Classic computer.

What lies ahead in these pages can be read slowly, a chapter at a time, or can be read at a single sitting. It will certainly be referred to time and again. But for every HP 3000 System Manager, IT MUST BE READ. In these days of cost cutting, "downsizing" etc., getting the most of what you have has become especially important. This book is a concise volume of information that allows the average System Manager, with no advanced degrees or pocket protectors, the ability to get the most of what they have.

Jason Goertz
July, 1992

Preface

I know what you are thinking... It's about time for Volume II. I suppose it is a bit over-due. One reason for this was that I wanted to get thoroughly equipped with the new HP-RISC systems. The MPE/iX operating system (originally MPE/XL) has evolved quite a bit since release 1.0 in the late 1980's. The other reason was that I started my own firm. Gee, it looks so easy when you are outside of a small, growing business looking in... I now know that in order to successfully run a business and keep a happy customer base it takes lots of hard work mixed with a servants heart being willing to give $1.15 of service and product for every $1.00 received. That venture has taken a fair bit of time.

In this volume I have pulled out all the stops and provide a very large quantity of, what I believe to be, very relevant information on HP 3000 performance management. Like the subtitle says, it is not only the theory or just the practice, but the theory and the practice that provides a solid foundation for success in a discipline. My desire is that this book will equip you with such a foundation.

Now, please spare me a short divergence from the system performance discussion. WARNING, if you really don't want to listen to me speak from my heart, please skip to the introduction and ignore the fine print in the margin of each page.

About those funny little quotes on the edge of each page. Lest you think I am crazy, let me explain. As I grow older (I am 34 as I pen these words), I am giving more thought as to life; what is it, how people miss it, how to get it, and make the most of it.

Consequently, I have been observing, really observing not only how to cure the performance of HP 3000s, but also how to deal with dysfunction in a business, raising kids, what makes a marriage work, etc. Now it gets really philosophical... I am saddened by what I see in 1992 happening to people I know, my country, and most nations on earth.

You know what I am talking about...

Marriages are being blown to smithereens (with all the resulting shrapnel), businesses filing chapter 11 or worse, leaders being exposed and brought down, economies faltering and failing, standards of living, for most, steadily declining, children rebelling, overdosing, and dying at alarming rates, genocide of our young (via abortion) taking place at unprecedented numbers as a solution to other very complex problems, our inner cities nearly resembling "Escape From New York", the elderly losing hope, and a general, apparent decline of (particularly) western civilization.

Some results I observe from the above...

People live with fear, fight to survive, become calloused and hard (if someone smiles at you on the sidewalk of some of our bigger cities, you think their up to something...), kill and plunder at the drop of a dime (or a court verdict...), compromise for expediency sake, and lose the desire and drive that once motivated people, especially in the United States of America.

Once again, if any of this offends you simply ignore it; just go ahead and read the computer stuff... My goal is not to offend anyone but perhaps to provide a slight wake-up call.

It is really hard for me to write this book and not speak of some of the concerns facing us in these times. You see, when I have the privilege of meeting you face-to-face, unless it is just for a very brief time, I'll want to know about your HP 3000 and your job, but I'll especially want to know about you. Do you have a family? What are your hobbies? What are you goals and dreams?

Then you'll learn a little bit about me if we have time...

I really like making things work at their optimum potential. I have been through various phases to gain maximum "performance" out of my Yamaha trials motorcycle, my health, gas mileage of my cars, my time, my investments, my blueberry farm and even my antennas for my short wave amateur radio transmitters. I'm into this performance stuff.

As much as I have practically treated HP 3000 performance nearly as a passion, it is still not the most important thing in my life.

As we continue to talk, you would hear about my three little girls (and one on the way) and my wife Colleen, my obsession for Mexican food, how I like to ride my tractor, tear apart my beehives and just watch them for hours, help others in need, and how I met the Lord Jesus Christ in 1974 and continue to walk with Him.

A book, more than anything else invites an author to come in, sit down with you and visit. You have invited me in. One way I can let you know who I really am is to scatter some of my thoughts (represented by quotations of famous historical people) throughout the book. You'll get to know how I feel about such areas as marriage, kids, money, government, successful living, wisdom, integrity, relationships, and self-discipline. I believe in these things and, more importantly, have consistently seen the benefit of living them and of surrounding myself with people who do. You may disagree with some of these ideas; that's OK. Perhaps you could use some liquid paper on the ones you don't like... They are my tiny contribution to provide a measure of healing and restoration to a civilization badly in need of both.

But onward to the subject at hand, the HP 3000. I would be glad to hear your thoughts (phone, FAX, mail) on things you like, dislike, and bugs you may find in this book. Please feel free to contact me at:

Phone: (503) 327-3800
FAX: (503) 327-3276
Address: P.O. Box 151, Albany, Oregon 97321

To Your Success!

Robert Lund
August, 1992

Acknowledgements

As I mentioned in Volume I, my desire is for this work to help you avoid the pitfalls I have encountered with HP 3000 system performance. At the same time I wish for you to experience the successes that I have enjoyed.

Over the last four years many people have assisted me in my continuing knowledge base of the evolving HP 3000. Many of those have been customers, folks that have received consulting, attended my classes, etc. Others have been those of you who research and write papers. Benjamin Franklin aptly said,

"As we enjoy great advantages from the inventions of others, we should be glad of an opportunity to serve others by any invention of ours."

I have benefited by the research, and subsequent papers, of many. To them, I say thanks for your hard work. In turn I have tried to reciprocate by also contributing original research for the advantage of others. This book is a cumulation of much of what I have found to be helpful in managing an HP 3000's performance.

I must say a special thanks to my team at Lund Performance Solutions who freed me up for vast periods of time to do this work. Thanks to Bill Lancaster for his on-going contribution in the science of HP 3000 performance management and commitment to me. Shelly, your days of toil in typing and re-typing manuscripts have not gone unnoticed. Never have I seen a person as dedicated to a job as you. Jane Barr, your ability to take my retarded English and make it civilized was very helpful. Mike McGuire your contribution of the art work is appreciated.

Thanks also go to those who have reviewed the manuscript: Debra Canfield, Chuck Ciesinski, Jason Goertz, Steve Cullen, David Johnson and Curtis Melton (the American Airlines dynamic duo!), Lisa Zenev, Vladimir Volokh, David Hodges and Bill Lancaster.

How to Use this Book

If you are a performance ace, you may want to skip the Introduction and just skim most of the textbook section. You will benefit most from the case studies and performance improvement suggestions. You will also like the Appendix on performance modeling.

If you have been around HP 3000s for some time but have not actively managed performance from a proactive standpoint, please take time to read the Introduction. Its purpose is to motivate you. If you have little or no performance experience, you should take your time through Section One. Read a chapter at a time. Answer the questions at the end of each part. Try to get as much hands on as you can with your performance tool.

If you are very green to the HP 3000, read everything slowly and then re-read! You will particularly like the questions and word definitions in Section Four.

Some Conventions...

- The term "Classic" refers to HP 3000 CISC systems; ones that run MPE V (V MIT, Platform One, etc.). This includes systems ranging from a Series 37 to a Series 70.

- Any reference to MPE/iX includes MPE/XL unless otherwise noted.

- Sometimes I use the terms RISC (Reduced Instruction Set Computing), HPPA (Hewlett-Packard Precision Architecture), PA-RISC, and Spectrum (original internal project name) interchangeably to refer to the same series of systems (917 to 99x).

Table Of Contents

Section One

Chapter 1

Chapter 2

Chapter 3

Chapter 4
Nuts And Bolts Of System Performance.....61

Chapter 5
Primary System Resources.....75

Chapter 6
Measuring System "Pulse Points".....87

Chapter 8
General Performance Management Tools And Techniques.....129

Chapter 9
Tuning System Tables For Optimal Performance: Fact or Fiction?..135

Chapter 10
Some "Gory" Classic Case Studies.....145

Section Three

Chapter 11

Chapter 12

Chapter 13
Monitoring MPE/iX Performance: Issues and Tools.....173

Chapter 14
Managing MPE/iX Performance - Tools and Techniques.....189

Chapter 15
MPE/iX Performance Case Studies....197

Section Four
An Encyclopedia of HP 3000 Performance Terms and Management Questions....209

Chapter 16
Commonly Asked System Performance Management Questions....211

List Of Figures/Tables

Introduction

The Perils of Flying Blind

Why is it imperative to have a system performance monitoring and management game plan? I suppose it is almost a universal human tendency to neglect something unless it becomes a problem. Most people I know tend to cringe when paying the premiums for the multitudes of insurances that surround their lives; some folks even forego insurance. Only a few have foresight enough to sacrifice now in order to make the future a little easier.

The kind of neglect I am talking about with respect to managing the performance of an HP 3000 can be better described as myopia rather than outright blindness. But it is, nevertheless, a form of blindness resulting from poor understanding, lack of time, inadequate tools, or a low commitment level to the process of system performance management.

In this introductory section, I hope to motivate you with a little bit of lecturing (preaching?) illustrated with humorous situations. However, if you are already motivated and committed to the process of monitoring and maximizing the performance of your HP 3000, you may want to dive right into Chapter 1.

As part of the introduction, I discuss some of the perils that can arise from not proactively monitoring and managing your system's performance. In other words, blindly flying your HP 3000. I present six perils of not paying attention to your system's performance vital signs. Conversely, along with each peril mentioned, I explain the benefits that can result by systematically monitoring the health of your system.

Often, I run into HP 3000 sites experiencing system performance problems. Typically, people are in a bit of a panic. All they really know is that

user complaints have escalated and batch job performance has progressed from bad to worse. Much of the time the system's performance has deteriorated primarily due to one thing: neglect. Essentially, there is one goal: gain optimal utilization from a high-performance machine without any, or little, notion of how that system is performing. The net effect is the same as that of a pilot who has lost his way in a cloud bank: flying blind.

The former president of General Motors, Alfred P. Sloane once said:

> *"If you don't measure and you don't evaluate, you don't manage."*

Good HP 3000 system management must involve measurement and evaluation of key performance indicators. Then, follow through with your knowledge by determining what action(s) to implement. This will help you continue to provide acceptable levels of service for your company both now and in the future. The following perils may result when measurement and evaluation of key performance indicators are neglected.

Peril Number One:

You will not be alerted by indicators that may suggest tuning the system for better performance.

System tuning is one aspect of any complex, high-performance machine. It's true whether you're talking about a Porsche or an HP 3000 Series 947. Tuning can mean different things to different people. To a programmer/analyst, it can mean performing an analysis of existing source code with a view to making it perform better. To a system manager, it may mean utilizing a less "intrusive" way to more fairly distribute system resources using, for example, the TUNE command.

On MPE V systems, much attention has been given to reducing the effect of that nasty thing called disk I/O. This has been accomplished in a number of ways: properly caring for and feeding software disk caching (ample memory, spare CPU, and good data locality); utilizing fast data retrieval tools and utilities (MR NOBUF file access, for example); housekeeping TurboIMAGE databases, etc. There has been much written on this subject.

Once I/O was tamed on the Classic systems, many people found that CPU utilization increased dramatically. It is not uncommon to see anywhere from two to eight percent of the CPU spent per software-cached disk drive. Many have found relief by disabling software caching and utilizing hardware-cached disk drives.

With MPE/iX systems there are a few new wrinkles. As Stan Sieler said:

> *"Obtaining optimum performance with MPE/iX is more difficult than on MPE V... there are more things to tune, with much less knowledge."* [1].

Data locality is still important, mapped files create great opportunities for tuning applications, and native mode optimized code all provide new performance possibilities. These, along with other new and only partially explored operating system features, are some of the ways tuning can be accomplished. If you choose to fly blind, you will certainly miss opportunities to tune your HP 3000 from a system and application perspective. These features will be expanded upon throughout the book.

Figure I.1 shows some disk I/O performance indicators on a busy 960 that point to two things. One is a hyperactive transaction manager (notice the spikes in CPU AQ Busy and disk queue length). Not much can be done (yet) to tune the transaction manager (XM), but these bursts of data "hand grenades" have no small effect on system-wide users. If you choose to fly blind, you will not be able to keep an eye on PIN number 9 (the XM checkpoint process). PIN 9 is the culprit for AQ spiking.

As of MPE/iX 3.0, the transaction manager has been improved so that this condition is not so bad. By monitoring your system periodically, you will be able to identify problems such as this and report them to HP, your software vendor or development staff.

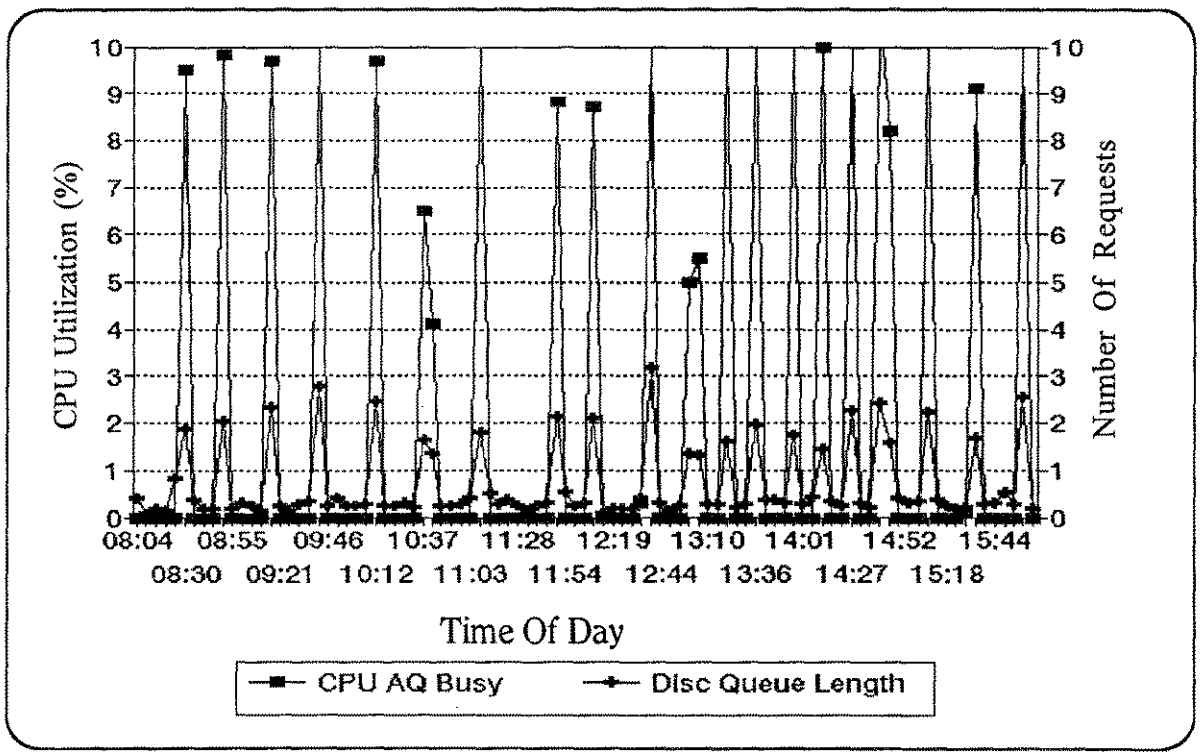


Figure I.1 - Excessive I/O Example and "Hyper" Transaction Manager

Wealth consists not in having great possessions but in having few wants. Epicurus

The second thing Figure I.1 shows is an excessive I/O rate. On this system other confirming indicators showed very poor data locality. This led to a reconsideration of the application design and hardware configuration. Flying blind may leave you at the mercy of hardware vendors and expensive consultants. Perish the thought!

Peril Number Two:

You will miss opportunities to extend the life of your system by not properly attending to housekeeping.

Some of the most pitiful performance problems I have seen relate to the area of system housekeeping. Most notably are the systems that had an operation running fairly smoothly until a crisis of some kind occurred. (What I mean by a crisis is an abrupt increase in user response or batch completion times.)

I remember one occasion when a small company was having trouble completing month-end processing in less than 30 hours. (Other companies with the same HP 3000, memory, software, approximate transaction volume, etc. generally ran month-end processing in less than 15 hours.) We discovered they had a heavily accessed TurboIMAGE master set with a very large number of inefficient secondaries. By changing the capacity of that set from 14000 (not a good choice!) to 14009, their month- end processing time decreased from 30 hours to about 14. The operator was concerned that his overtime pay was now history!

A couple of things that pointed to this problem were excessive CPU pause for disk I/O and high I/O rates--fairly obvious with a monitoring tool.

If you wish to maximize your company's hardware investment, housekeeping issues need to be attended to, especially in the Classic environment. File balancing, load balancing, database fragmentation, and disk space fragmentation are some of the areas to keep up on. More on some of these later.

But how does a monitoring plan ensure that you'll catch these housekeeping chores before they become critical? By proactively monitoring the system for key indications of resource bottlenecking. For example, one of the primary indicators of poor disk I/O performance (often a barometer of how well the "house" has been "kept"!) is how much of the time the CPU is paused for disk I/O. This number is a good one to watch on both the Classic and MPE/iX systems. As it grows, so does your problem. More on this in the "pulse points" discussion later.

Many folks have told me they have forestalled a hardware upgrade for a year or so simply by identifying such problems early on and taking evasive action. When you fly with full visibility, you will be able to make wise decisions pertaining to your company's resources. Often enough, better housekeeping is what the doctor prescribes.

Peril Number Three:

You do not see early warning signs that point to a hardware upgrade requirement.

There comes a point in the life of every growing company when the current system "runs out of gas." Perhaps you have performed all the tuning you know of, housekeeping rates a "ten," and yet throughput is waning and users are whining. A look at a performance blueprint for a typical day or week in the life of your system will help. Such a blueprint can be created if you have historical performance data.

Performance curves are graphical representations of system resource usage and activity. Figure I.2 shows the effect of increasing system activity (adding more data entry users) and the subsequent diminishing returns of flat transaction throughput. Somewhere between five and ten users is where peak performance is obtained in this laboratory load model. Even within this range response times are hardly acceptable by most standards.

The menace of Peril Number Three is that, if you are flying blind, you never quite know where the point of diminishing returns begins on your system. If you have attended to tuning and housekeeping and notice with your monitoring tool that this is happening on your system, the inevitable hardware upgrade may be necessary. A hardware upgrade could involve the addition of more main memory, more disk drive capacity, or even a new processor. But you should never be put into the position of determining your upgrade requirements without obtaining all of the objective, unbiased information possible.

Who would think of driving an expensive automobile any length of time with faulty water and oil pressure indicators on the instrument panel? I think you get my point.

The fear of the Lord is the beginning of knowledge: Fools despise wisdom and instruction. Proverbs 1:7

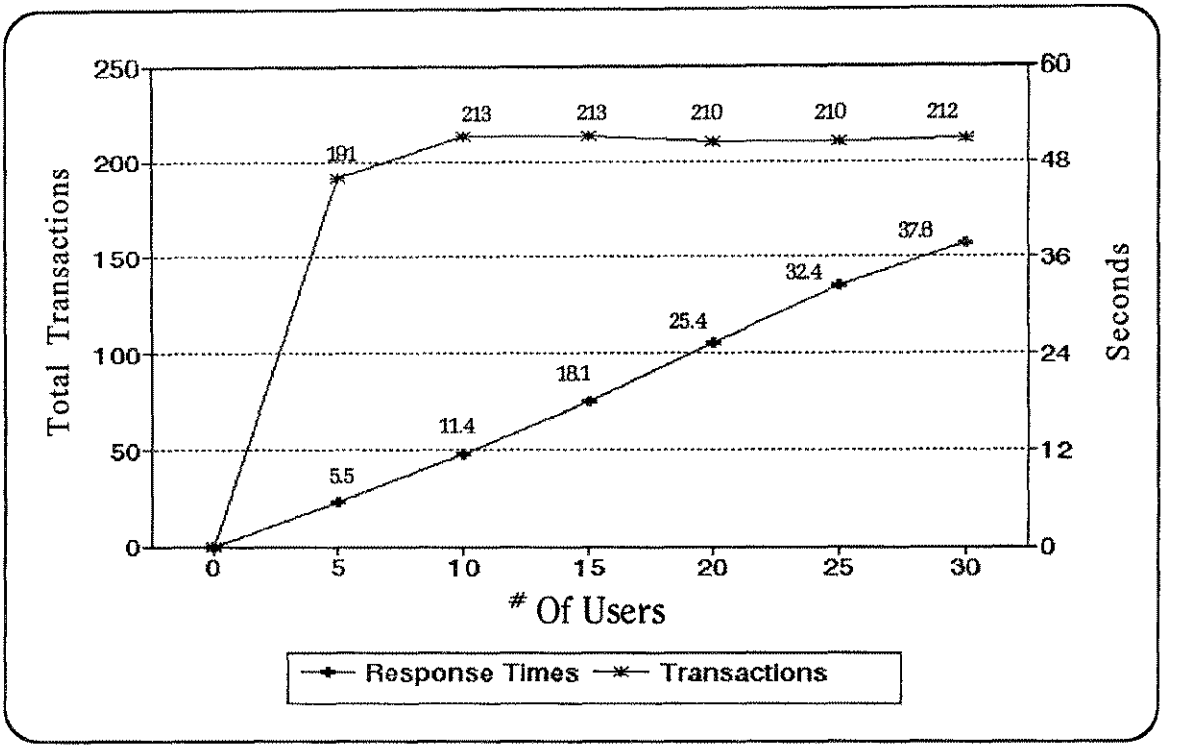


Figure I.2 - Increasing Users versus Throughput

Back to the panicked system manager for one moment. When the system's performance becomes the loudest "squeaky wheel" (amongst the myriad of others that MIS folks deal with), much attention is focussed on the person responsible for the system's operation. That person usually reacts by feeling as if the only choice is to upgrade to a "faster" box.

It has been my experience that people often wait until it is too late to obtain a new system if they have been flying blind. Who has a hundred thousand dollars (or more?) in a moment's notice for a "quick" upgrade? Usually not many. By implementing a sensible performance monitoring plan you can rest assured that you will not be caught by surprise when "yellow" zone indicators begin to show up, pointing to a resource bottleneck.

Just what does the effect of upgrading hardware have on a shop? Figures I.3 and I.4 show a before (925 LX) and after (935) picture of a typical busy day for one company. The activity for each day had minimal variability. This company had no "instrument panel" (performance tool) to provide indication of system performance degradation. If they had not ultimately obtained a monitoring tool and some outside help, they would not have known what the real problem was and which system to upgrade to. In this case, it was primarily a CPU horsepower shortage.

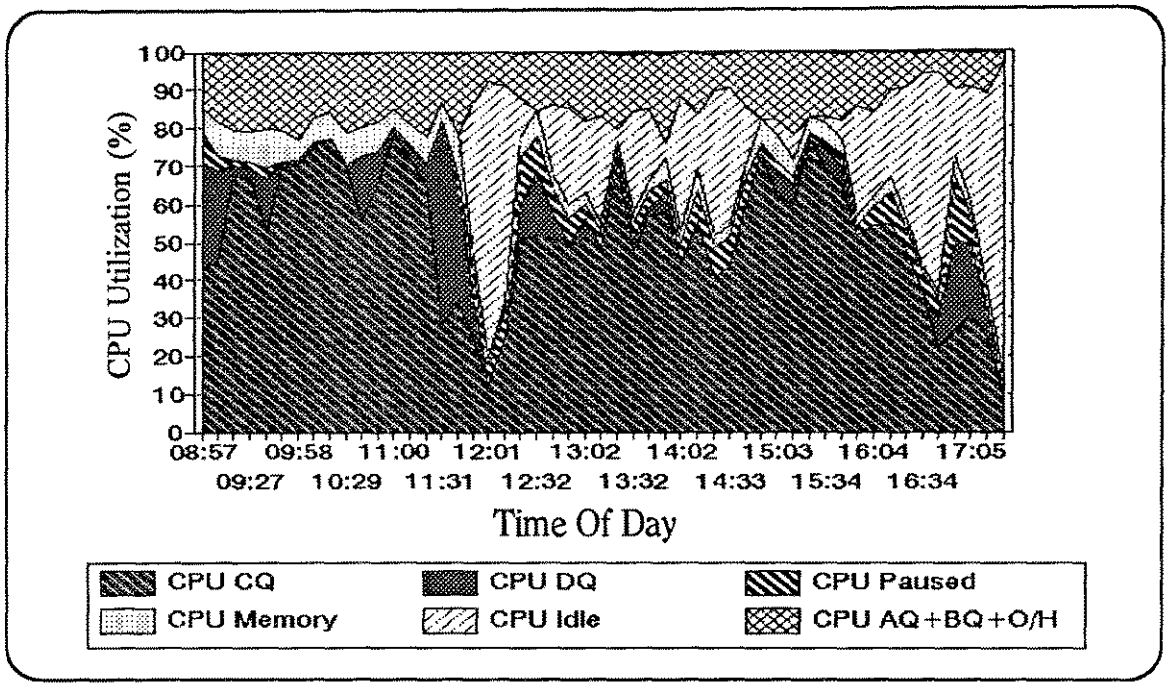


Figure I.3 - Series 925 LX CPU Utilization

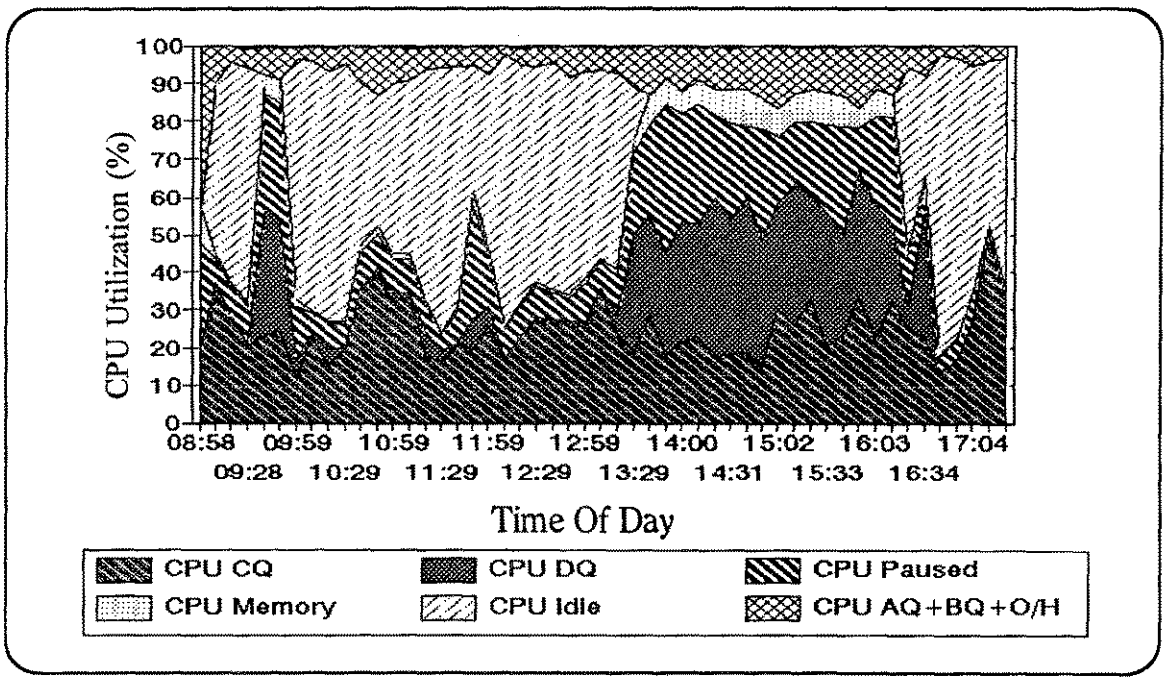


Figure I.4 - Series 935 CPU Utilization

Notice how much idle CPU time is available after the upgrade. This is CPU in the "bank." It should also be mentioned that this company was able to plan for another CPU upgrade beyond the 935 (to a 948) well in advance of its actual required time because they obtained a monitoring tool and some outside, objective consulting help.

There are reliable early warning indicators of hardware bottlenecks. I will be discussing these in detail in the textbook section.

Were we as eloquent as angels, yet should we please some men and some women much more by listening than by talking. Colton

Peril Number Four:

You will not be able to profile and thus improve "hog" applications

If you think for a moment about what causes poor performance, you will inevitably come up with one solution: All of your performance problems will be solved if you simply do not run programs (thanks Bob!). Programs ultimately impact the system, and whether the impact is good, bad, or ugly largely depends on the efficiency of the programmer's code and how the programs are run.

Keep in mind that programmers have a blank system resource "check." By allowing someone to write code on your system, you are, in effect, giving them nearly an unlimited right to consume expensive CPU cycles. Just for fun, figure out how many seconds there are in a typical processing day and divide the cost of your system by those seconds. This will give you a price per CPU second. You can literally assess a cost for each program that executes!

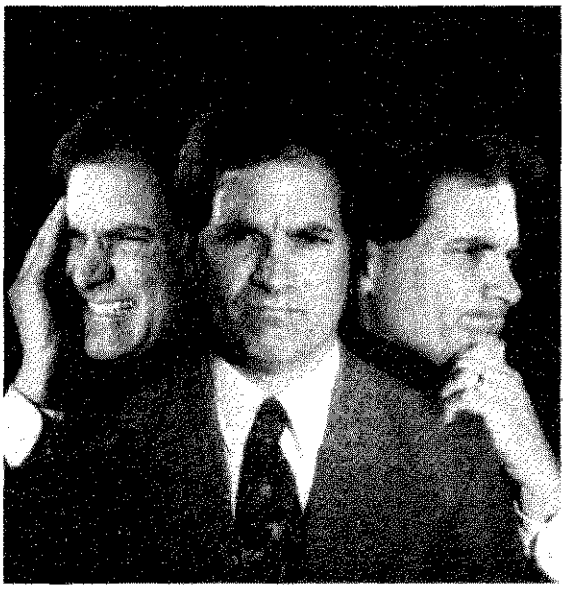
I feel strongly that, as a part of the program development QA process, some resource price tag should be assigned to new or modified programs. (Gee, wouldn't that be great for new releases of the operating system?) A good monitoring tool will provide a few key "pulse points" like:

- How many CPU milliseconds per transaction are necessary?
- How many disk I/Os (average) are required?
- How is main memory going to be impacted? Will we need more?
- What kind of mode switch activity can we expect (MPE/iX)?
- Which files are accessed and in what manner?

A simple way to profile a program is to first fire up your monitoring tool in batch, run the program in question, stop the tool and study the results. Program changes could then be made and the cycle performed again, noting the key indicators in the above list. In order to improve renegade programs, you really need a way to quantify their consumption.

Peril Number Five:

You will not know where you have been, where you are, or where you are going.



Have you ever flown in a small plane when the visibility was zero or close to it? Have you ever been in the same situation but also had instrument failure? Probably not because there's a strong chance you would not have survived. Reliable instruments and avionics are mandatory in flight. The same is true in charting your system's performance history, current utilization and future system usage (capacity planning). A key component in measuring and evaluating your system's "heartbeat" involves historical trending.

By implementing a method to obtain and chart past data, you will begin to detect patterns of system usage. At a minimum, this will help you better balance the system's load, identify seasonal aberrations, and extrapolate future usage.

When in doubt, always collect data on your system. Archive portions of it to save disk space if need be, but collect data. It is very helpful to replay a day of last month's system utilization and identify processes that were acting up (when you weren't there, of course!). If you can collect this data based on a business unit perspective (CPU usage by finance and data entry departments, for example) then you are really in for a treat. If you know that your manufacturing workload will be increasing by, say, 50 percent over the next quarter, you can then perform some simple capacity planning using statistical forecasting or queuing network hardware analysis.

I'll be speaking more later on capacity planning and, in particular, utilizing queueing network analysis to predict future hardware needs. This area of future planning is becoming a more crucial part of HP 3000 system management. Consequently, "serious" capacity forecasting typically has been performed using modeling techniques. Most notably it has been done with queueing network analysis. In the large IBM systems environment, the preferable way to predict the future is with queueing network models.

How can we have a productive society, which depends upon savings, capital goods, and an educated and motivated workforce, when the emphasis of our society is on leisure, entertainment, politics and lawsuits, none of which are productive. R.E. McMaster

It has been said that the only thing we learn from history is that we don't learn from history. However, it is my experience that wise system managers can make great use of historical data on the system to assess current and future courses of action. We'll be addressing the issue of capacity planning more in Chapter 1 and in Appendix A.

Peril Number Six:

You will not be able to maintain a cool head in a performance crisis.

All my possessions for a moment of time. Last words of Queen Elizabeth I

I remember a couple of panic situations when I was taking pilot training. One was an instance in which my instructor made me put on a hood to keep me from looking outside the aircraft. He then placed the plane in a very contorted (and dangerous) situation. What I was supposed to do was NOT trust my feelings. I remember feeling quite dizzy and somewhat nauseated, it was that bad. And I remember an awful feeling of panic as I stared at the instruments and instantly tried to make sense of them. My body said that we were in a steep right bank, but the instruments swore that we were veering left. I responded inappropriately and only made things worse. So much for the learning curve! Things improved later in other situations, as I learned how to understand and trust the instruments.

The lesson here is simple. An ancient proverb states:

> *"If you are slack, in the day of distress your strength is limited". [2]*

If you want to deal quickly and decisively in a system hang situation or some other performance crisis, then you MUST have knowledge and a methodology in place to properly respond. At a minimum, if you do not have an online performance monitoring tool that will help you quickly identify the problems, you are once again: flying blind.

I have witnessed many who have wasted time and money trying to deal with system performance problems on a reactive basis rather than a proactive one. A monitoring tool, knowledge, and a plan are all essential components for your flight bag kit. They will help you avoid the perils of flying blind.

This book is aimed at equipping you with knowledge that will help you avoid the above mentioned perils. Next, we'll look at some scenarios that illustrate the need for a "heads-up" performance management plan.

Common Flying Blind Pitfall Scenarios

What follows are some humorous but life-like situations I have encountered. These different performance scenarios illustrate many of the battles managers face. The intent of each is to provide a relevant plan to identify and resolve such problems. The emphasis will be on the less technical aspects of performance monitoring, managing, and capacity planning. Later on, particularly in the textbook section of this book, I provide more technical detail about problems like these.

For some of the scenarios, I use output and examples from real life problems I have consulted on. Of course, the names have been changed to protect the innocent (and guilty!).

Scenario #1

The Response Time Crisis

You come a bit late into the office (ABC Corp.) after a Monday morning appointment. On your door you notice at eye-level a "yellow sticky" from your accounting manager that says, "System problems, see me ASAP! JL" When you call JL's extension, he explains that the system has been acting sluggish all morning. Further, if that wasn't bad enough, he explains that Brenda (your system manager) is off to vacation land with no beeper. JL explains that user response times have been getting worse over the past four months. This weekend, the system manager (remember, the one who is on vacation) successfully finished installing 20 more users on your 130 user series 960 HP 3000. Now, Monday morning response times are bad and batch activity is even worse. What's the problem?

This situation, and variations of it, unfortunately are not uncommon. What do you do? Some of the options are:

- Call in a consultant to look at the system (probably hit you for at least $1,400 for the day, plus travel time).
- Have a performance analysis done by a third party. Costs anywhere from $1,500 to $5,000.
- Analyze the situation in-house (but your lead performance techy is off to Bermuda).

This is no game, and we want them to know that. We're not going to sit by and watch these guys play an electronic version of "Can you top this?" Richard DeFilippo on the FBI's crackdown on computer hackers

In this situation I advocate that the non-technical manager or operations staff take a stab at identifying what the problem is.

The Sad Typical Company

If ABC Corp. is typical, it has been flying blind in some facet of performance management. From my experience, the majority of performance-"unenlightened" companies can be described as follows:

- They usually have some kind of performance monitoring tool.
- They typically don't track performance data over time for historical trending.
- If they do have a tool, they usually have very little, if any, relevant performance understanding to help them determine what constitutes a performance bottleneck. This is like driving a race car that has no red, yellow, or green indicators for things like oil pressure, water temperature, etc.
- Often, there is very little commitment from upper management for tools and training. Such things are often viewed as: "Hey Dad! Will you buy me a tachometer for my car?" Is a tachometer essential to the running of that car? No, but it is a serious mistake to place performance monitoring systems in the same category. Sadly, many upper managers do.
- Oddly enough, of the companies I have surveyed, most will spend anywhere from $2,500 up to over $10,000 per system per year for performance related support/consulting from outside vendors. Sometimes these monies are bundled (read: hidden) within support contracts, so it is not readily evident.
- Many will upgrade a processor prematurely because they were unable to determine what was happening on the system. One company mentioned to me that they forestalled a recommended upgrade by one and a half years simply by aggressively using a performance monitoring tool to help them schedule activity and tune the system more intelligently. Cost savings and net benefit to the company were tremendous. If you can survive on an existing system for nine months or more, you're virtually guaranteed that there will be new processors that will "smash to pieces" the price/performance of those currently available.

If you tell the truth, you have infinite power supporting you; but if not, you have infinite power against you. General Charles Gordon

Now, I do realize that there are exceptions to the above: But for about 70 or 80 percent of the companies that fall into this "flying blind" category, these characteristics are true.

Let's assume that ABC Corporation has been flying blind on some levels, but at least they have invested in a performance monitoring system that allows them to get at relevant data. There are four "bases" that need to be covered (see Section One, Chapter 2 for the bases). ABC Corporation is faced with what I call "Crisis Mode." What I'm going to do is present the various steps that ABC's manager and staff took to determine what was happening on their system. We'll discuss the other bases later.

The Continuing Saga at ABC. . .

First of all, they activated performance data logging with their performance tool. This was so they could look at the big picture for a day or longer. Someone should have had logging on all along, but it was never a priority. Funny how insurance policies seem to be similar in that respect--quite a nuisance until you need one!

After logging was turned on, they fired up their performance tool interactively. The first item they looked at was CPU activity. Since the CPU is the engine of the system, it makes sense to find out if there is enough available to service the demands of the user population. Since the staff at ABC did not have a long season of data with which to analyze the system (they have thoroughly repented and are now collecting data on an on-going basis), they were limited to a short-term analysis.

A key item to look at when considering whether adequate CPU horsepower exists is how much of the CPU currently being utilized is high priority. By high priority, I mean processing (users and jobs) that cannot suffer worse response times without it being a career decision for the system manager! ABC felt that this meant the majority of all their interactive users. In Scenario #4 we'll talk more about this.

ABC came to the conclusion that since there was very little batch activity and virtually no idle time, they were bumping CPU saturation. They came to this conclusion primarily on the basis of the high priority CPU Busy value along with the CPU queue length. The additional thirty users put them over the edge so that their CPU was swamped during peak periods. There simply was not enough processor power to go around.

The main consideration here was whether this was an anomalous "spike" in activity or a consistent indication. If only ABC had implemented logging...

Scenario #1 Conclusion

Could this situation have been avoided, or at least been predicted? You will probably answer YES, after reading this book. The issues here revolved around a few areas:

- More proactive thinking in regard to performance management.
- Adequate staff training with respect to performance problems.
- A bit more sensitivity on management's part as to the finite nature of system resources (namely the CPU).

Fools rush in where angels fear to tread. Alexander Pope

? Questions/Discussion ?

1. How do you and your staff currently handle situations like the one above?
2. What tools/indicators do you use to measure the health of your system?
3. How important is the issue of the staff's ability to handle crisis mode in light of the multitude of other DP responsibilities?
4. How committed is your company to attracting, developing and keeping quality people (implies a commitment to system education)?

Scenario #2

Batch City Log Jam

The system manager at Un-wise Corporation didn't realize this Friday would end up being called Fry-day--system manager fry-day, that is. The night operator had left just three hours before (he was very tired) and thought he had properly sequenced the final month-end jobs. A number of job aborts during the wee hours of the morning had caused a delay in the batch runs being able to finish just before the users were scheduled to come online. He wasn't aware that a few of the jobs had aborted. So he busily kept reading his book... Later, he discovered that a problem was occurring.

The system manager came in an hour early because the night operator notified her of trouble brewing. Now, she noticed that 10 jobs were running together; someone had inadvertently raised the job limit to 10. She thought, "I'll just let `em run for another couple hours or so; the users can wait that long before they get back into their key applications." She made the decision to allow the development team and all others to do their work. After all, these jobs can't take more than a couple hours to finish, she reasoned.

Time went on...You know that feeling you get when you are sure a job is going to complete any minute? She had that same feeling, but they didn't finish. She must have typed SHOWJOB one hundred times if she did it once. All the while gnawing thoughts hounded her: "I should have aborted them hours ago; maybe they're stuck in a loop, how much CPU time are they getting anyway? I'll let them go another half hour..." And so the story went.

At 1:30 p.m. her phone rang. It was the boss. He wasn't happy. Orders weren't being processed. The users didn't have much to do. But, funny thing, the programmers were singing like larks; no response time problems for them!

Picking up the data processing help-wanted section, she pondered, "Which floppy did I put my resume on?"

She had no way of determining the progress of the jobs. Would it be best to abort some of the jobs? What about sequencing problems between them? Could it be that the happy programmers had something to do with the slow job progress? Could the system queues have something to do with the problem? She wished they had some way to automate this entire job process.

Make all you can, save all you can, give all you can. When I have any money I get rid of it as quickly as possible, lest it find a way into my heart. John Wesley

The moral of the story is that in order to keep from flying blind, you must be prepared to monitor and manage effectively the batch job environment. Poorly scheduled jobs can have a detrimental effect on the overall throughput in a shop. Running too many or the wrong mix of batch jobs can give the appearance of premature CPU saturation. Running too few will, at times, be not taking advantage of available CPU horsepower due to excessive pause for disk activity.

Some folks use MPE commands (schedule function on the STREAM command), home-brewed job schedulers, SLEEPER (contributed library limited, but effective), or commercially available ones. You usually get what you pay for. The commercial ones can generally be cost-justified if your batch load is fairly substantial. Note, though, that the cost-justification process for commercial schedulers often includes staff reduction.

Knowing specific performance indicators, such as the amount of CPU pause time, is invaluable not only for batch scheduling but also for determining some bottlenecking situations. It is possible that online folks are devouring all the available CPU. Without having a monitoring methodology to not only view the behavior of jobs, but also to manage them properly, you will be blindly flying your HP 3000.

A man must first govern himself ere he be fit to govern a family, and his family ere he be fit to bear the government in the commonwealth. Sir Walter Raleigh

? Questions/Discussion ?

1. How does your shop schedule jobs? Is it adequate to avoid fiascoes similar to this scenario?

2. Have you been able to justify purchasing a commercial job scheduling package? Some of the more popular ones I know of are JMS/3000 (Design/3000), Maestro (Unison), OCS/Express (OCS), JOBRSQ (NSD).

3. Does your department deal much with "critical path" batch activity? How does this impact performance?

4. How do you chart the progress of a job? When do you make the decision to abort jobs and re-run them? What is your rerun percentage for the last month?

5. How do you currently know if you are making full use of your CPU (minimizing spare pause time) without overdoing it?

Scenario # 3

The Demanding Board of Directors

It's late afternoon on Thursday at XYZ Corporation. You are preparing to exit for the day. The CEO walks in looking a bit flushed. After returning from a ballistic board of directors meeting he spreads some papers out on your desk like a deck of cards. "The board members wanted to know if our system could handle some more workload due to a proposed acquisition," he exclaims.

He continues, "It seems that we are going to take on the accounting and order processing for the new firm we're going to absorb. Plus, they want to add a new payroll application. They said it is imperative that we know whether a CPU upgrade is necessary. They all looked at me and asked how much more the system could handle. All I could give them was how much total CPU we have been consuming for the last nine months. I honestly couldn't answer the questions..."

Anyway, the heat was on at XYZ Corporation. More accurately, the heat was on the CEO, which meant it was on for the system manager!

This situation has many concerns. Such as:

- Can we consolidate two systems into one? How do we know for sure?
- How do we estimate future hardware needs (at least within the ball park)?
- How much will a new application impact system response, resource usage, etc?

This scenario and these questions lead us into the subject of workload activity and forecasting. This is quite a mature discipline in larger IBM and other big MIP-machine arenas, but not so in the ranks of the HP 3000. The HP 3000 is now capable of competing at the mainframe level and can therefore compel approval of gargantuan DP budgets (millions?). When we are talking about that kind of money, companies have a hard time justifying "another hundred thousand" for excess hardware.

But how do you predict future hardware needs?

The question of predicting future hardware needs based on existing or new application growth can be sticky. Of course there is the obvious problem of knowing for sure how much your company will grow in terms of business activity and, therefore, what the impact will be on your HP 3000.

If you have some understanding of future company workload growth figures, then you know that a workload is some meaningful grouping of programs and users. A workload could be as simple as all text editing activity or as complex as all financial activity (AR, AP, GL, etc.). It is one thing to forecast overall CPU growth and another to break out individual department growth, thereby providing for more granularity and perhaps more accuracy. You'll see more on the subject of workloads in Chapter 3.

In the past, it was assumed that if your company was growing at a certain rate per year, then the required computer horsepower would correspond. Many people have used a simple statistical forecasting method with or without some kind of confidence factors. This method is simple and fairly inexpensive, but provides questionable results.

Later on, in "Base Number Three" of Chapter 3, I go into all the pros and cons of the various capacity planning methods. But for now, if you are even half-way serious about capacity planning, consider the following:

> When a system performance analyst performs a serious capacity planning study, the most common method used is queueing network modeling (accurate and cost effective). This is the science of predicting future resource needs. It is the kind of science that assists AT&T or Burger King, for example, in determining when a larger switching office or more cooks are necessary to handle projected future demands in phone calls or Whoppers.

Queueing network analysis is discussed more fully in Appendix A and in the following references: [3], [4], [5], [6]

Back to our scenario. How could the system manager of XYZ Corporation be better positioned to answer the questions about order entry and accounting? First, a data collection methodology could be implemented that facilitates input into queue network model formulas.

Collecting workload performance data is getting to be less of an option as the data processing plot thickens. Consider Figure I.5. It illustrates the effects of the additional impact of three new applications six months down the road.

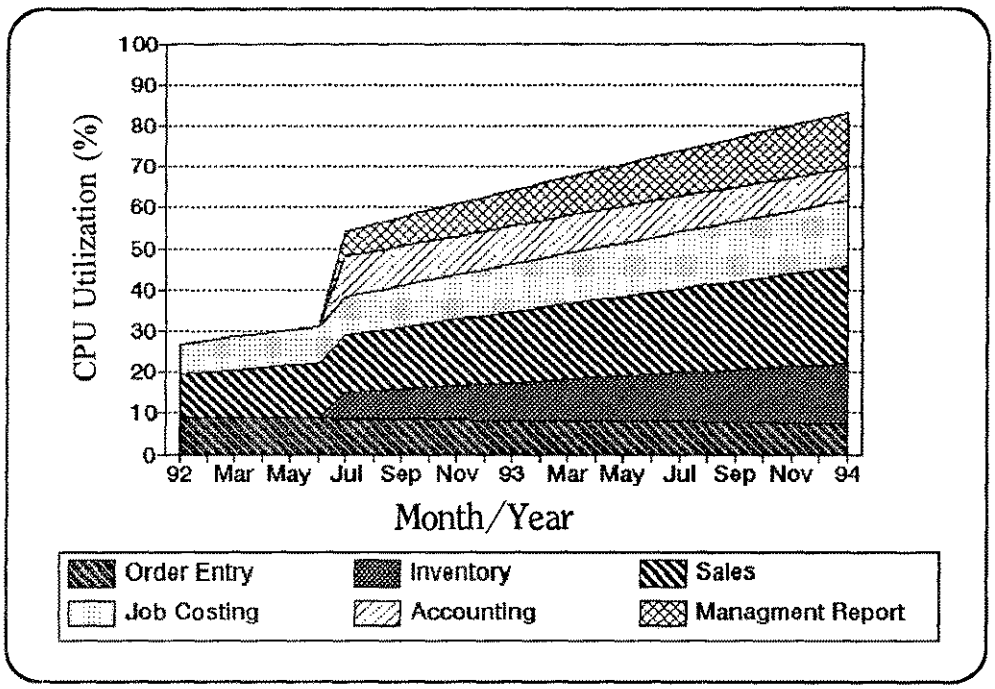


Figure I.5 - Workload Forecasting Using Queue Modeling

By previously ignoring their workload resource utilization, XYZ placed themselves in the embarrassing position of not being able to answer some tough, real-life, questions. Armed with this data and good tools/ methodologies, XYZ Corporation will be able to compete more effectively in the global market place by not over- or under-buying on hardware. More precise planning and, thus, more effective use of increasingly scarce DP dollars will result.

The entire subject of capacity planning can be a bit confusing, but more tools and knowledge about the subject specific to the HP 3000 are becoming available. I deal with this subject more precisely in Chapter 2 and Appendix A.

Before borrowing money from a friend decide which you need most. American Proverb

? Questions/Discussion ?

1. Which capacity planning methods have you used in the past? Which ones worked best?

2. How would you answer the following questions if the CEO drilled you:
   a. Can the current hardware handle an increase of 30 percent more order-entry transactions over the next four months?
   b. What will happen to order entry response times when the new payroll package is implemented?
   c. Which would be better for performance, two 957s or one 967?

3. What are some tangible steps you can take to begin implementing a proactive capacity plan?

4. Take some time to define and describe your workloads. List them in descending order by the amount of CPU they consume.

Scenario # 4

The Alarmed (but naive) System Manager

This situation occurred at Sticky-Wicket company. The old system manager bailed out abruptly--something about a 500 percent pay increase at another firm. Management decided to bring in a new guy. He didn't have a lot of HP 3000 background but had "the right raw material," so the CEO said.

On his first day, the new manager let all the staff know who was in charge. One of the first things dictated was a system performance audit. He asked the lead programmer what kind of performance reporting tools were available on the system. After being shown a performance tool, he proceeded to monitor daily CPU usage. After all, isn't CPU busy the best indicator of whether you have enough horsepower?

Two weeks of monitoring went by. His report to the CEO said that they needed a new CPU because it was at or near 100 percent busy for the majority of their peak processing periods. What is right or wrong with this conclusion? While it is true that the amount of time the CPU is busy can be a good indicator of the need for more, there are others. If these are not taken into account, you WILL suffer varying degrees of pain depending on the situation. Flying blind costs...

How could this manager have more adequately determined the need for an upgrade? The other indicators I referred to can be lumped into subjective and objective categories. First the subjective ones. (These aren't totally subjective, but they are based more on the "gut feel" side of things.) Some of these are:

- Do you have the general feeling that batch jobs are completing in a timely fashion?
- How often do users complain about response times?

The best objective indicator of a CPU shortage is CPU Busy because it indicates how well the system is performing. But there is another side to the story. It is important to realize that if only one batch job is running, the CPU Busy indicator will often push 100 percent unless there is significant disk I/O bottlenecking. (More on this later in the textbook section.) So, the question should be asked, "Are any batch jobs running at the time the high CPU busy was observed?" If so, what is the "interactive only" busy time?

It is generally understood that measuring your high priority busy time is a better indicator of CPU saturation. (See Scenario #1 above for more on this.) It is vital to separate your workloads into high- and low-priority activities whether they are batch or online. High priority can be thought of in this way: With a particular group of jobs or sessions, can you stand any substantial increase in response or throughput time? If the answer is no, then consider those jobs and users as high-priority ones, or put that activity in the low-priority category. The high-priority processes are the ones you want to combine to see if they CONSISTENTLY use 70 to 85 percent of the CPU's time (remember to add system overhead). If so, you could be at or near CPU saturation. Use the low end (70 percent) for MPE V systems; the high end (85 percent) for MPE/iX systems.

Two other supporting data values for CPU load are the process preemption rate and CPU queue length. These two are a bit more technical, but most performance monitors report them.

Success is to be measured not by wealth, power, or fame, but by the ratio between what a man is and what he might be. H.G. Wells

? Questions/Discussion ?

1. How has your company planned CPU upgrades in the past?

   A. Wait until users form a lynch mob.
   B. Proactively monitor high priority CPU utilization.
   C. Over-buy on hardware.

2. What indicators do you use to tell when the CPU is swamped?

3. Do you and your staff understand how to use your performance monitoring tool?

4. Do you have a plan in place to track historical trends in performance?

Scenario # 5

"Heads-Up" Corporation's Performance Management Game Plan

In the previous four scenarios, I have outlined some of the shortcomings many companies experience in performance management. With this last Scenario I want to paint a picture describing what a "heads-up" DP shop would look like with respect to performance management. I've used a hypothetical company, Heads-Up, as our model. The items listed below are not necessarily in order of priority. You may even wish to use this list as a goal sheet.

1. Our model company's DP manager is proactive in his thinking. He has put in place procedures to log performance history for graphical presentation as well as ex post facto problem analysis. Heads-Up collects performance data continually. Its DP staff is constantly looking for new ways to improve what they do. After all, one definition of insanity is doing the same thing again and again... and expecting different results!

2. Heads-Up data processing manager has received tremendous backing from upper management. Some companies' upper management view the Informations Systems (IS) department as a necessary evil--more evil than necessary! The First CEO at Heads-Up was from the "old school" that took a pretty dim view of computers. The new one does not have the same antagonism and even encourages finding new ways to provide cost-effective and efficient information services. He knows that there are tough times ahead for companies that refuse to give serious attention to the role that IS plays in a company's success. If you cannot secure some kind of commitment from upper management for tools, training, etc., you may be fighting a losing battle.

3. This company is committed to attracting, training and keeping people who will not only be able to manage their system's performance, but help improve the overall operation. One way to build staff morale is to provide an individualized training program. Even in lean economic times, it will help you and your firm keep your needed edge by investing in a well-trained, quality staff. It's been said often and is still true, "Work smarter not harder."

4. Our mentor company has a healthy repertoire of tools on hand to manage performance, database issues, general security, etc. In particular, since our subject is performance, they have performance tools that cover the following bases. I'll discuss the bases in more detail later, but for now here they are in a nutshell:

- **Crisis Mode** - Referring back to our earlier scenarios, we remember that crises are rarely forgiving. Your performance tool must facilitate easy identification and resolution of performance bottlenecks. Standard MPE commands really are not adequate here. You must consider investing in a good tool to cover crisis mode.

- **Casual Mode** - By this we mean providing an education for your staff with relevant performance information. Again, your best bet is to use a tool that provides your techies as well as non-techies with helpful information. This makes it a positive experience for DP staff to monitor the system when things are going well, so they get a better feel for what a real problem is.

- **Capacity Planning Mode** - This may be the most difficult. A "heads-up" company will not perform capacity planning as a last minute, right-before-the-budget-is-due exercise. Rather, the process of forecasting future hardware needs based on the shifting sands of new or growing workload demands is woven into the very fabric of what data processing is all about. Again,

When men are most sure and arrogant, they are commonly most mistaken, giving views to passion without that proper deliberation which alone can secure them from the grossest absurdities. David Hume

you will need good data collection and modeling tools/techniques to cover this base.

- **Problem Solving Mode** - By definition, crises occur infrequently. But problems are almost an every day occurrence. Consider the following:

  - Why do Suzie's response times go up 50 percent when sales reports are streamed? She hasn't "gone ballistic" yet, but is greatly annoyed. What can be done?

  - Why does the posting job take twice as long to process the same number of records as it did three months ago?

  - Why does our performance tool report finance application processes as being impeded most of the time?

These and many other problem scenarios raise their ugly heads all too often. It's a good feeling to have procedures in place to deal with such issues.

5. Heads-Up long ago learned the need for good support from experts in any field they were involved in. From their experience, vendor support is almost as important as the product or service itself. How true it is for software utilities, and especially so in the confusing area of system performance, that you need reliable, competent support for the tools you have onboard. Smart companies create a network of "loyalists" who are committed to their success. These folks bring expertise and value that is generally unavailable internally.

There is always the tension of people versus capital resources. Do we simply over-buy on hardware rather than implement good, proactive performance management planning? A little pain all the time, or one large dose? The answer is really for you to decide. If you choose the route I espouse, you and your firm will profit greatly by implementing the principles that I have briefly discussed so far and will elaborate on in subsequent chapters.

I hope that after you read this introductory material you are sufficiently motivated to read the rest of the book. You will be equipped to create a performance game plan for your company that will ensure success in the arena of HP 3000 system performance management.

? Questions/Discussion ?

1. Did Heads-Up Corp. make you feel guilty? If so, which item do you feel the greatest need to change, implement or improve on?

2. Do you get a second opinion from consultants when you are faced with "big dollar" decisions or have you just done the best you could?

3. What action can you take right now to begin performance data collection?

4. Have you ever received a data center audit which reviewed all of your procedures, utilities, etc.?

Notes:

Money is an article which may be used as a universal passport to everywhere except heaven, and as a universal provider of everything except happiness.

Duty as Seen by Lincoln

"If I were to try to read, much less answer, all the attacks made on me, this shop might as well be closed for any other business. I do the very best I know how--the very best I can; and I mean to keep doing so until the end. If the end brings me out all right, what is said against me won't amount to anything. If the end brings me out wrong, ten angels swearing I was right would make no difference".

Abraham Lincoln

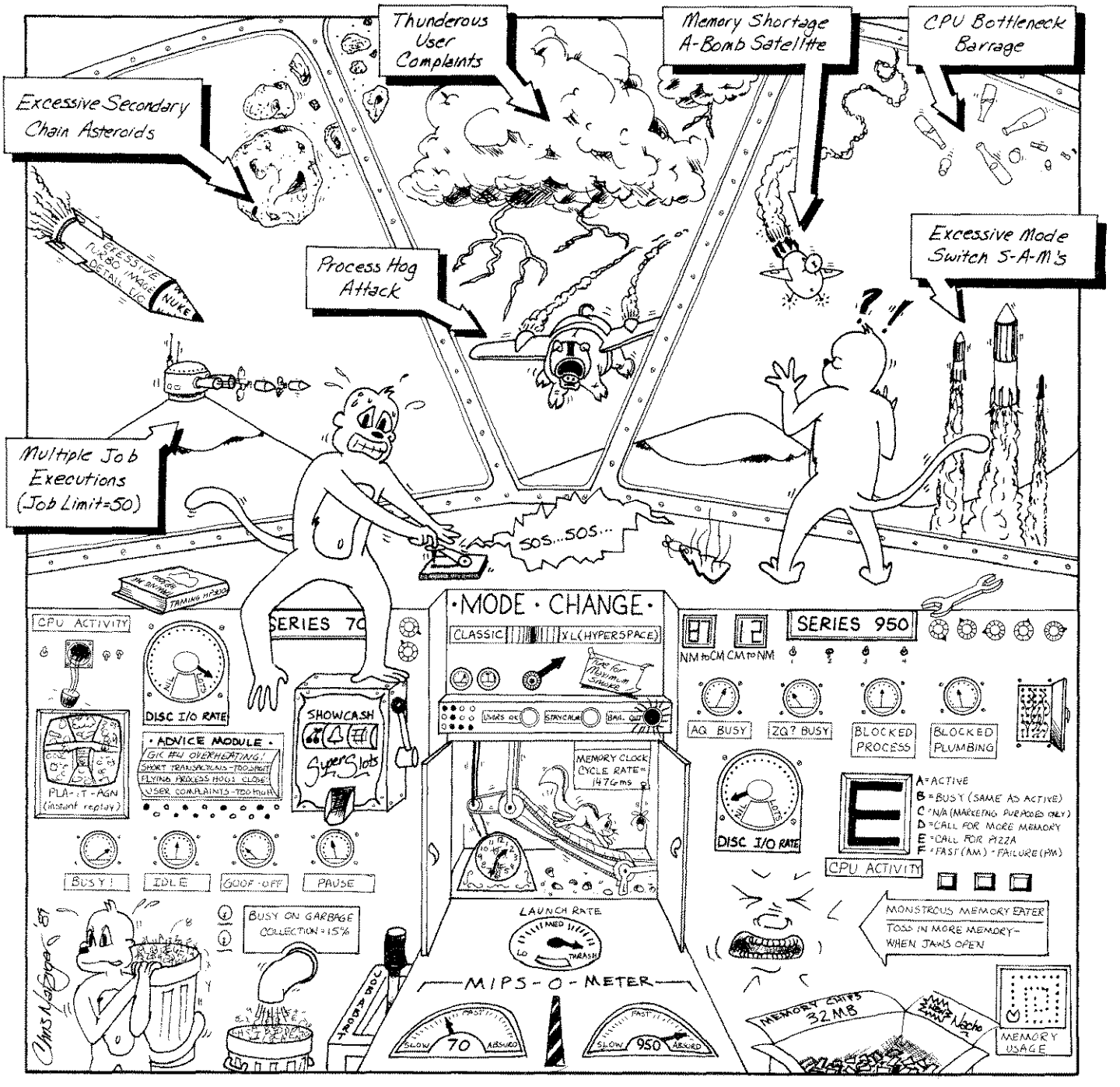
Section One

A Textbook Introduction To HP 3000 Performance

Computer textbooks typically start with a technical view of system performance. This may be useful for some college courses but, like much of academia, may not be terribly relevant in the real world of day-to-day system management. I will therefore first present the practical side of understanding and managing HP 3000 performance. Then we'll explore some of the more technical aspects.

Although technical information can be used practically, often the two terms "practical" and "technical" are mutually exclusive. So, the two parts of this section will focus on the practical and technical aspects of performance management, in that order.

Also be sure you take some time to read the encyclopedia in Chapter 17; it is intended to be a veritable performance "catechism." You will find it useful as you encounter various new words and jargon in the discussions that follow.



Your staff handling performance problems while you are vacationing in the Bahamas.

Chapter 1

System Performance Management: The Basics of Service

The practical side... This is where you live (or die) everyday in your role as a DP professional. This is where all of your book learning and experience (or lack of) are supposed to come together.

The subject of system performance can be very technical. It involves the interaction of an amazing amount of convoluted software and hardware mechanisms. This interactivity is supposed to help you produce information that will "oil" the wheels of your company. The entire process must be manageable and it must be cost-effective. As I stated in the introductory section, you need knowledge and tools to properly harness such high-tech equipment. This part of the textbook section focuses on the practical side of HP 3000 system performance.

My intention here is to equip you with some basic knowledge of how to monitor and manage your HP 3000's performance. Most of what follows are things I wish I knew in my earlier days of HP 3000 management. We'll start with a definition.

What Do We Mean By System Performance?

Regarding this thing called system performance, consider the following quote:

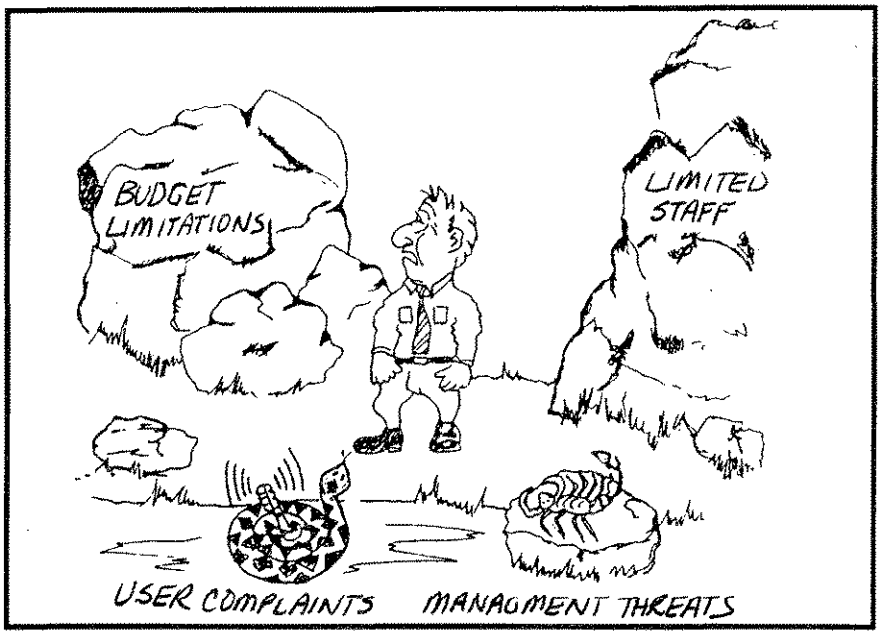
> *"System performance on the HP 3000 is a science of complex variables, interrelated dependencies, and vast engineering mysteries. It is a strange mixture of hardware and software, of pure logic and good guesses, of solid engineering and, yes, maybe even magic... sometimes it works wonders."* [7]

Beyond the maze of hardware and software, system performance is first and foremost a perception of service. If your job or terminal transaction completes in an acceptable time frame, you feel that performance is good. However, when requests are delayed, frustration results. In reality, a system manager is managing perceptions. An important aspect of this perception is consistency. Inconsistent performance (great one minute and terrible the next) is just as much a villain as consistently poor performance. If you provide micro-second response time and then it shoots to ten seconds, users will be frustrated.

On the system side, performance is merely the ability of a set of hardware and software components to complete user requests (transactions). You can think of an HP 3000 as a collection of service centers much like a fast food restaurant. Your order (transaction) is comprised of service from different stations such as the french fry counter, shake area, grill etc. In the same way, a number of servers get involved to provide the ingredients for your transactions. This leads us to the next question....

What is Service?

System service implies a client and a server. In your case, you, along with many pounds of hardware, software, and staff, are the server. You are supposed to provide your clients with service. This service is what your users complain about if they do not get enough. It's what your management uses to threaten your career if they do not get it. But *good* service requires an adequate budget to fund hardware, personnel, tools, and training so that your service meets everyone's demands.



Your IS department provides service in several forms. At the basic level, service is timely access to the processing of meaningful information. This could be as simple as an inquiry to the Jones' August payment record within a couple of seconds or so. Or it could be as elaborate as a complete MRP run in a manufacturing environment, taking six hours of non-stop batch activity.

Here's the problem. Somehow you have to provide access to data within an acceptable time frame and do the whole thing without exceeding the budget. This service you are to provide to your clients consists of four elements:

- Timeliness: How promptly does the client receive requested data?
- Accuracy: How correct is the client's data?
- Cost: How much does the client have to pay for the data?
- Reliability: How available is the client's access to the data?

So, your service covers a fair bit of ground.

Timeliness

Online users want short responses--consistently! Batch jobs have to complete within specified windows. Inconsistent or poor response time affects productivity. Batch jobs that suffer starvation create frustration among your clients and staff. A more succinct way of putting this is: poor service translates into lost revenues either in the form of real money coming into the enterprise or "funny money" received via a chargeback mechanism. Consider this:

> *"Timeliness measures are related to the delivery of results from the computer system within a time period that is acceptable to the end-user in terms of his or her own productivity".* [8]

Accuracy

Not only do your clients want timely response, they want accurate data. Errors due to poor setup or handling of data may result in unusable output.

For on account of a harlot one is reduced to a loaf of bread, And an adulteress hunts for the precious life. Proverbs 6:26

Here are some ways errors occur:

- Mounting wrong tapes.
- Purging wrong files.
- Running a job twice.
- Dropping tapes.
- Entering wrong dates.

In any case, you will hear about it if data is served garnished with errors... even if you served it quickly!

Cost

Your service can be timely and even accurate, but if it costs too much you will be persecuted. Remember that rock and hard place? Keep in mind the following excerpt from the monthly performance column by William Lancaster in the HP Chronicle:

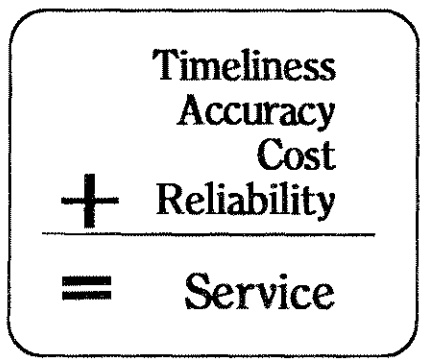
> *"My contention is that from a company management perspective, system performance management is fundamentally a financial activity. A closer look at the primary elements of system performance management will show this to be true. The following list of questions represents a logical flow of concerns for a typical shop:*
>
> 1. *Does my current system configuration meet my users' needs?*
> 2. *Does my current configuration allow for growth matching anticipated company expansion without adversely affecting the users' expectation of performance?*
> 3. *How long will my current system last given anticipated growth?*
> 4. *What is my most cost-effective upgrade path?*
> 5. *How long will my most cost-effective upgrade path satisfy my growth requirements?*
> 6. *Do my growth figures allow for variable growth by application?*
> 7. *How can I predict future capital requirements thereby allowing me to plan ahead enough to obtain the cheapest money (what a concept!) in the most disciplined fashion?*
>
> *Good financial managers will notice that each of these questions can be reduced to dollars and cents either directly or indirectly."* [9]

And the business done in the home is nothing less than the shaping of the bodies and souls of humanity. The family is the factory that manufactures mankind. G.K. Chesterton

Reliability

You must also provide some reasonable access to your client's data. This means they must be able to depend on a pre-determined level of availability. Can they stream jobs? Or is the job limit a constant impedance? When they need to obtain a port, how often can they "connect"? Communication lines, UPS systems, disk drive/CPU redundancy are a few of the areas that comprise service availability.

All of the above elements of service involve resources and expectations. Consequently, organizations must have some way to reliably figure whether the expected service is being delivered or not. Humans usually resort to some form of a contract when such things are involved. Enter SLOs, SLAs and SLM...



SLOs, SLAs and SLM

When relating to the real world, there are three aspects of service we must be concerned with:

- Service Level Objectives (because we're goal-oriented).
- Service Level Agreements (since we need contracts).
- Service Level Management (because all of the above falls down without it).

These three aspects of system service are non-optional ingredients for any proactive shop. But you say, "I cannot afford the time to implement formal service level management." My response is it's more expensive in the long run (time, energy, money) to disregard these ingredients.

A Flatterer never seems absurd: The Flatter'd always takes his word.

Every shop has SLOs. Although they are often unspoken, they do exist. Managers requesting reports have an expectation of when they will be ready. Users often have objectives regarding response times. And, *you* have opinions on all of the above. It is therefore best to take the time to survey your customers objectives and map your results in a way that is realistic given your hardware/application mix.

Consider the following aspects of SLOs. They should:

- Be easy to report.
- Be easy to measure.
- Need little or no user input.
- Be able to be set up by one person in a day.

Few things in life are more difficult for some of us than admitting a mistake.

An SLA is a contract between the DP department and its customers. The implication is that both the customer and the DP department agree to certain levels of batch and interactive service. Following are some components of typical SLAs.

- A signed contract with each user (without a signature, no responsibility is assumed).
- Online response times by application.
- Batch turn around by class.
- System availability by application.
- Accuracy limits.
- Load of each application by transactions per hour and batch jobs by day.
- Application load by resource requirements.
- A plan for reporting SLAs.
- Priorities if service cannot be delivered.
- Penalties if the user exceeds the load.
- Penalties if the data center does not provide service.
- A schedule for follow-up meetings and interface. [10]

One major benefit of pushing an SLA is that you will have some backing (read: resource commitment) from management. By gaining management's "vote," you will more easily be able to perform proactive capacity planning, new application profiling, etc. This allows you to implement a strategy to deal with your department, rather than succumbing to the default, which is reactive.

Whether your SLA is very specific and detailed or simple and broad, it is advantageous to have one. At a minimum, it causes both DP and its customers to be accountable to expectations and responsibilities.

Objectives imply agreements. Agreements, by definition need to be managed. The entire SLO/SLA process is dynamic.

This brings us to service level management. Managing the entire scheme involves:

- A reporting process : Who's on first, what's on second?
- Contingency plans : If plans go awry...
- Scheduled meetings : How are *we* doing? [11]

If service levels are missed for a short time or over a period of time, a contingency meeting needs to take place. How can service be restored? What can be done to squeeze a little more processor "juice" for the highest priority workloads? This will involve analysis and more meetings. Perhaps some arbitration or compromise will be necessary.

If service levels are on target, you'll still want to meet. Albeit, those meetings will be less heated! It is possible that, given the future growth of the system, more workload can be sustained now. Maybe the CPU is under-utilized. Though it is not wise to walk the line of system saturation, neither is it prudent to buy a Peterbilt diesel truck to haul a small tent trailer.

Conclusion

When considering the concept of system performance, most people immediately think of response time. Perhaps batch turnaround time will be a close second for many. The overriding concept, however, is perceived service. If you can weave SLOs/SLAs/SLM into your IS culture, you and your company will be better off in the long run, though it may be a bit painful for the short term.

Although I have sketched a fairly brief overview of this thing called service, the bulk of this book will be primarily concerned with the timeliness aspect of service.

Everybody finds out, sooner or later, that all success worth having is founded on Christian rules of conduct. Henry Martyn Field

Service Level Agreement

Department: ( ) Date / /

Contacts: ( )
( )

Approval: X_____ X_____

Online Service:

		Goal	Actual	Diff
Availability:	8 a.m.- 5 p.m. M-F	98%	100%	2%
Response:	% of responses within .5 seconds	90%	95%	5%
Throughput:	transactions per minute during peak time: 14:00-16:00	600	500	-100
Accuracy:	Operations errors	< 5	5	0
	Application errors	<10	5	-5

Penalties for missed service:
10% reduction in billing for 2% missed service unless missed service caused by user.

Penalties for exceeded loads:
10% increase in billing and no penalty for missed service.

Reporting:
Data center will provide this ( ) report by 8 a.m. each day.
Weekly report will summarize service for the week.
Changes to SLAs must be negotiated with xx and yy.

Priorities:
If resources are unavailable, users of the following applications will be dropped in priority:
Lo-Pri Compiles, all general ledger jobs

Batch Job Service:			
Type A: Turnaround within 45 minutes	90%	80%	-10%
Type B: Turnaround within 20 minutes	90%	90%	0

Figure 1.1 - Sample Service Level Agreement

? Questions/Discussion ?

1. In order of importance, list the four elements of service at your site.

2. Which aspects of timeliness are most important to your customers?
   a. Consistent (but not so great) response times.
   b. Excellent batch turnaround time.
   c. Sub "N" second response time "X"% of the time.
   d. Great online reporting response time.

3. What are some of the best ways to handle the cost aspect of service? Department charge-back? A capacity plan to predict future capital requirements?

4. Take some time to reflect on SLOs and SLAs in your shop. Which aspects are implied? Which are explicit?

.5 Rate yourself on a scale of one to ten (ten being excellent) as to the effectiveness of SLM in your department.

Notes:

He who tells a lie is not sensible how great a task he undertakes; for he must be forced to invent twenty more to maintain that one. Alexander Pope

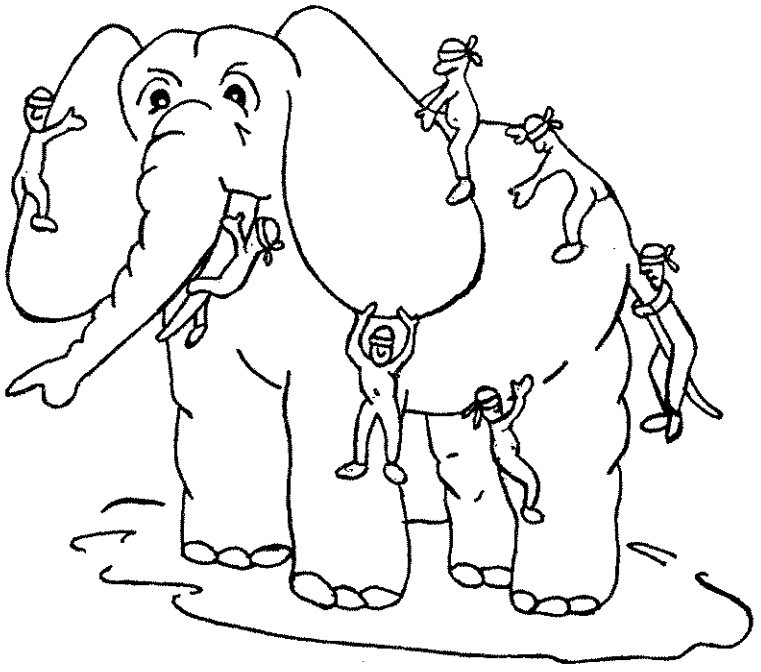
By profession I am a soldier and take pride in that fact. But I am prouder--infinitely prouder--to be a father. A soldier destroys in order to build; the father only builds, never destroys. The one has the potentiality of death; the other embodies creation and life. And while the hordes of death are mighty, the battalions of life are mightier still. It is my hope that my son, when I am gone, will remember me not from the battle but in the home repeating with him our simple daily prayer, "Our Father Who Art in Heaven."

Douglas MacArthur

Chapter 2

Covering the Performance Bases

As you may have gathered by now, system performance is a multi-faceted concept. The way many shops view and thus manage HP 3000 performance may be compared to seven blind men trying to describe an elephant. One has hold of a tail, another an ear, and yet another is tugging on the trunk. Each thinks he has a full view of the elephant.



In this chapter, I present various views of system performance which, when combined will allow DP management to put together a sensible game plan for monitoring system performance. Many system managers have felt as if they were "flying blind," much like the seven blind men. The intention of this chapter is to correct some of this performance myopia or blindness.

First, let's look at the big picture. Any good plan for performance management should, at a minimum:

- Cover all the performance bases.
- Allow effective viewing of performance angles.
- Focus on high-leverage performance Pulse Points.

- Be cost effective.
- Provide data relevant to both technical and non-technical personnel.

We will begin to explore these and other issues in this section and will discuss them in more detail as they are relevant throughout this book.

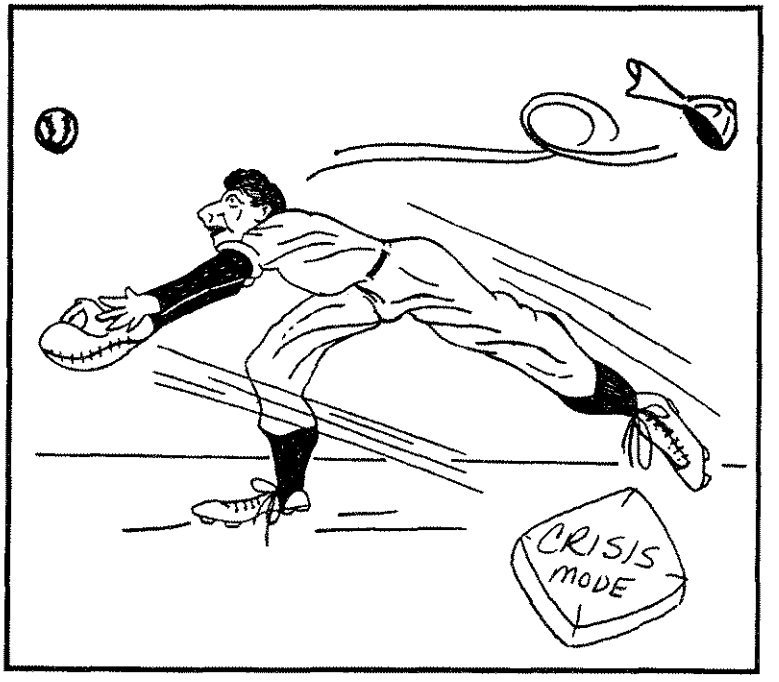
Performance "Bases" to Cover

What are the Bases?

The performance monitoring bases are the "battle zones" the DP manager operates in from time to time. Performance monitoring tools of the past have provided tremendous amounts of technical information, but lacked coverage in some of the more practical areas. For example, the "antique" tool OPT/3000 (MPE V only) addressed a few of the angles and bases of performance management, but left out such relevant data as response times, workload resource consumption, etc. OPT/3000 and other tools may find a cozy home in the toolbox of performance specialists or those concerned with tuning MPE but they have little relevance in today's system manager's repertoire. This is because a full-coverage performance game plan needs to cover all the bases, not just present a screen full of low-level technical data. What follows is a discussion of each of the four bases so that you'll be able to cover them better.

Base Number One: Crisis Mode

You are probably familiar with an occasional system "lockup" or hang. Users may complain about excessive response times, or perhaps a job that normally takes one hour, takes five instead. In these instances, something has happened on the system to cause certain programs to receive inadequate service. Perhaps a process has gone wild at a high system priority and has consumed more than its fair share of CPU. Events such as these can be thought of as crises.

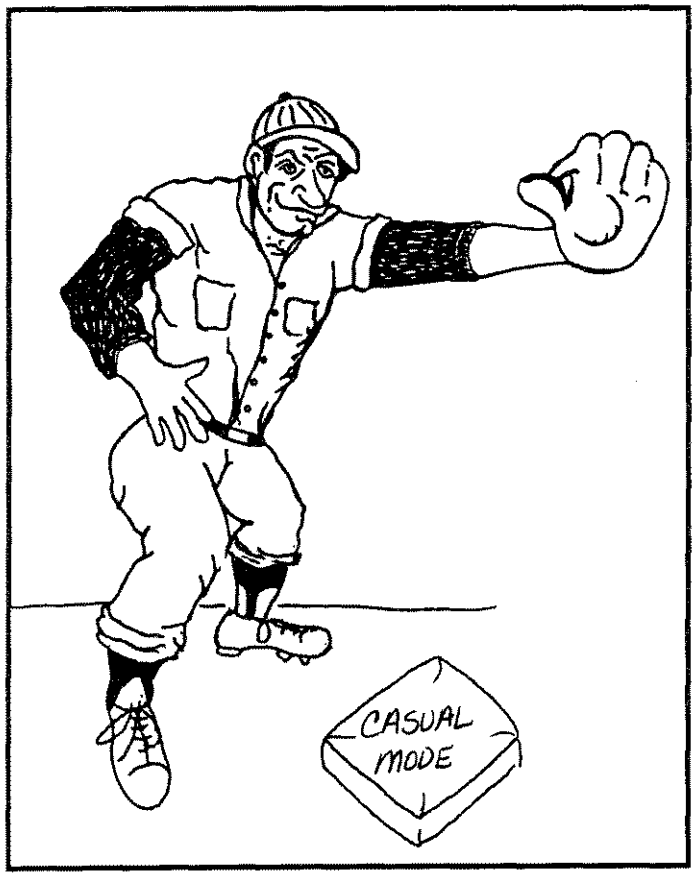


In a crisis, it is often one or two processes that are causing poor performance. Identifying these processes is part of the battle, dealing with them is the next part.. A few performance tools on the market facilitate locating such CPU hogs. Evasive action in a crisis might involve altering the offending process' priority with a feature found in un-supported XCALIBUR on classic systems as well as various HP and third party tools. The ALTPROC command (MPE /iX) will help you here.

In a crisis it is important that you have a plan in place to tackle the problem. Never fun, but always a looming possibility, Crisis mode is very unforgiving if a plan is not in place to tackle such an event.

Base Number Two: Casual Mode

Casual Mode involves a way in which DP staff may easily monitor the system's activity on a nonchalant basis. Essential to this is "real time" data rather ex post facto. If you have never had the opportunity to look around the system from the standpoint of processes, workloads, and global resources, you are in for a treat. Casual observation of the system has the following benefits:



a. It enhances your familiarity with your company's unique workloads and their effect on your particular hardware configuration.

b. It contributes to your company's knowledge base of system performance cause-and-effect relationships. This is essential to weaning your firm from outside performance consultants.

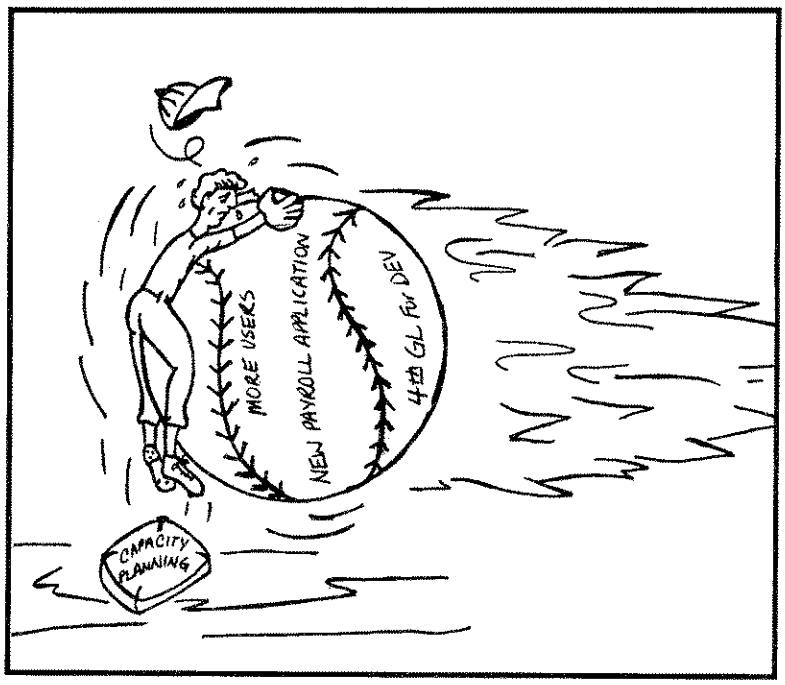
c. It adds to your value as an individual who can provide performance expertise. Many folks claim to be performance savvy, but few deliver.

The budget should be balanced, the treasury should be re-filled, and the public debt should be reduced. The arrogance of officialdom should be tempered and controlled. And assistance to foreign lands should be curtailed, lest we become bankrupt. Cicero, 63 B.C.

Casual monitoring of your system's performance with a friendly performance tool will contribute to your understanding in this reserved-for-gurus arena and will increase the price tag on your head in the job market. You'd be surprised how much firms pay for performance analyses. Here is my recommendation: study one item on the screen of a newly acquired performance monitor for 10 to 15 minutes each day. In a matter of months, you'll be operating like a guru!

Base Number Three: Capacity Planning Mode

This base involves historical trending, workload characterization, and future capacity projection. (I elaborate on workload characterization in Chapter 3.) It is one thing to know where you are on a performance curve and yet quite another to predict how resources will be utilized in the future. There are four types of capacity planning methods utilized by HP 3000 shops. They are:



- Educated Guessing
- Benchmarking
- Queueing Network Modeling
- Historical Trending/Linear Projection

Educated Guessing

Educated guessing involves considering the hardware offerings presented by HP or third-party resellers. If you're on a Series 42, for example, then the next logical step might be to get a Series 52 or 70. But now that the 900 Series systems are here, your sales rep will probably laud the benefits of a shiny new 9xx system. The problem is knowing the right decision. From here, it is simply guesswork. Many system managers who have attended my performance workshops complained that they were told to upgrade the CPU, memory, or obtain a "faster" disk drive only to find little or no perceivable improvement in overall performance.

Analogous to the I-upgraded-but-saw-no-gain-in-performance complaint goes something like this: "You have a car that desperately needs a tune-up. With great diligence, you replace spark plugs and wires and set the timing. To your chagrin you find that the car is still sputtering. Exasperated, you take the car to an expert who, after performing a few diagnostics, informs you that although your heart was in the right place, what the car really needs is a new carburetor!

Thus, capacity-planning-by-assumption oftentimes ends up being pure speculation. The plethora of upgrade choices makes this method less favorable than in the good ol' days of few choices and easy money (relatively speaking). You risk buying too much computer, thereby wasting your company's capital resources or worse yet, you don't buy enough. When you run out of horsepower a year-and-a-half early, everyone is surprised. Guess who they come after?

Benchmarking

Benchmarking, involves taking all or part of your applications and running them on a system of a different size. You simply compare batch run times and, if possible, interactive response times. This is obviously a very accurate, but expensive method.

A more practical and fairly accurate method is merely talking to companies who are using a similar application mix on a larger system than you are on. Many times application vendors have customer reference points for different-size systems. Be sure to tap their knowledge on the performance benefits of various size systems using their application.

Benchmarking may tell you how an immediate upgrade will impact your environment, but it does not allow you to answer difficult or tricky questions about the future.

This triangle of truisms, of father, mother and child, cannot be destroyed; it can only destroy those civilisations which disregard it. G.K. Chesterton

Queueing Network Modeling

Queueing Network Modeling involves forecasting future resource utilization using mathematical models. These models (using various formulas based on Little's Law) can be used to determine the throughput of everything from complex telecommunication networks to fast food restaurants (how is burger "response time" at 12:00 noon vs. 10:45?). Performance modeling involves a few steps:

a) Data Collection and Validation Phase

- Workload characterization. What are your application workloads, (finance, order-entry, telemarketing, etc.)? How do you define them?

- Workload measurement. What are the service requirements of your transactions? What are some of the characteristics of your users and batch jobs behavior (think times, arrival rates, etc.)?

- Input data into model and calibrate. Based on inputs such as service and think times, number of users, etc., the model should be able to predict current utilization, throughput, and response times. If it cannot do this successfully, you will have to "tweak" the model in such a way that it will re-analyze your assumptions and perform another collection. At that point you will have a reliable baseline model with which to predict future changes on your system. Model validation essentially compares actual with predicted performance.

b) Projection Phase

Change model inputs with "what if" questions. Examples are:

- What if my current order-entry transaction volume doubles?
- Can I survive a system downgrade from a 950 to a 927?
- What will the net impact be of an additional new payroll system six months from now?

All these questions and more may be easily addressed by queue network models.

c) Verification Phase

The actual performance of system changes are compared to model forecasts.

Queueing Network Modeling can be very accurate, but cannot be accomplished unless workload information is available. In order to create your own model, you will need to "hit the books." One book you may find helpful (though it is not for the faint-hearted) is "Quantitative System Performance" by Edward D. Lazowska, John Zahorjan, et al. [12] You will find it contains some of the best information available on computer performance modeling.

Hewlett-Packard uses its own "internal-use-only" model called CAPLAN to perform capacity planning consulting for customers. Although there are many modeling tools commercially available for IBM systems, the only commercially available tool that uses Queueing Network Modeling for HP 3000s is FORECAST/3000 Capacity Planner from LPS (see vendor reference).

Keep in mind that certain assumptions exist with models; this means there are inherent limitations. A capacity planner needs to be aware of a particular model's assumptions before going "full speed ahead." For a simple introduction to the math of this discipline and a practical look at the power of Queueing Network Models, see Appendix A, "An Overview of Capacity Planning with Queueing Network Models.

Historical Trending/Statistical Forecasting

You can perform Historical Trending/Statistical Forecasting yourself with any good spreadsheet or statistics program. You will need to capture CPU statistics by workload and then find the average use for that period. A minimal amount of statistical functions will help.

This type of capacity planning first involves gathering data on system resource usage over a period of time. This data is then loaded into a spreadsheet/graphics program. A graph is then made, similar to the one shown in Figure 2.1. The assumption is that if you have had a past growth in CPU usage of a certain rate, and if you expect such a growth rate to continue, then by extrapolating with a dotted line, you can form cross hairs that allow you to target a certain date on which your system will have hit a saturation point (see Figure 2.2). Well, you better have a plan prior to this date to accommodate the projected increase in CPU demand!

You can capture workload CPU consumption (or just total CPU usage) and plot the data according to the procedures summarized below:

1. Measure current system performance for extended period of time (minimum three to six months) broken out by workload.
2. Download relevant performance data to spreadsheet/graphics program.
3. Use linear regression on graph to obtain average capacity utilizations.
4. Extrapolate with projected growth rates; locate intersection of resource saturation (usually CPU) and date.
5. Track projected utilization with actual (down-the-road).

This form of capacity planning is limited to projecting CPU utilization and does not take into account priority interaction between workloads as well as other resource constraints that could wreak havoc in a forecast. It also does not show the impact of new applications or modification of existing ones. But, frankly, it is better than nothing. If you use this method, be alert for indications of excessive disk activity and data structure impedances, such as poor database locking, etc. Your forecast could be predicting ample CPU, thus leading you to think all is well. All the while, a massive disk I/O bottleneck could be taking place. Consistent day-to-day system monitoring is always advisable to perform reality checks on your forecasts.

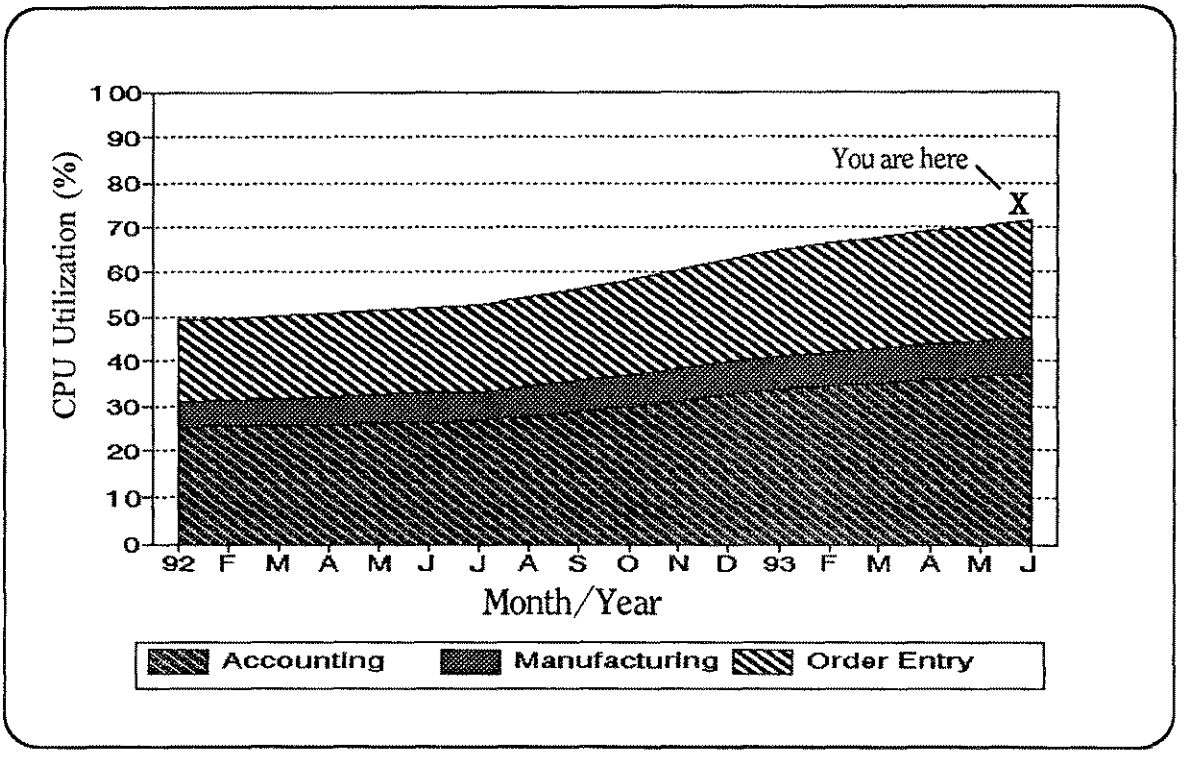


Figure 2.1 - Past CPU Utilization Trend

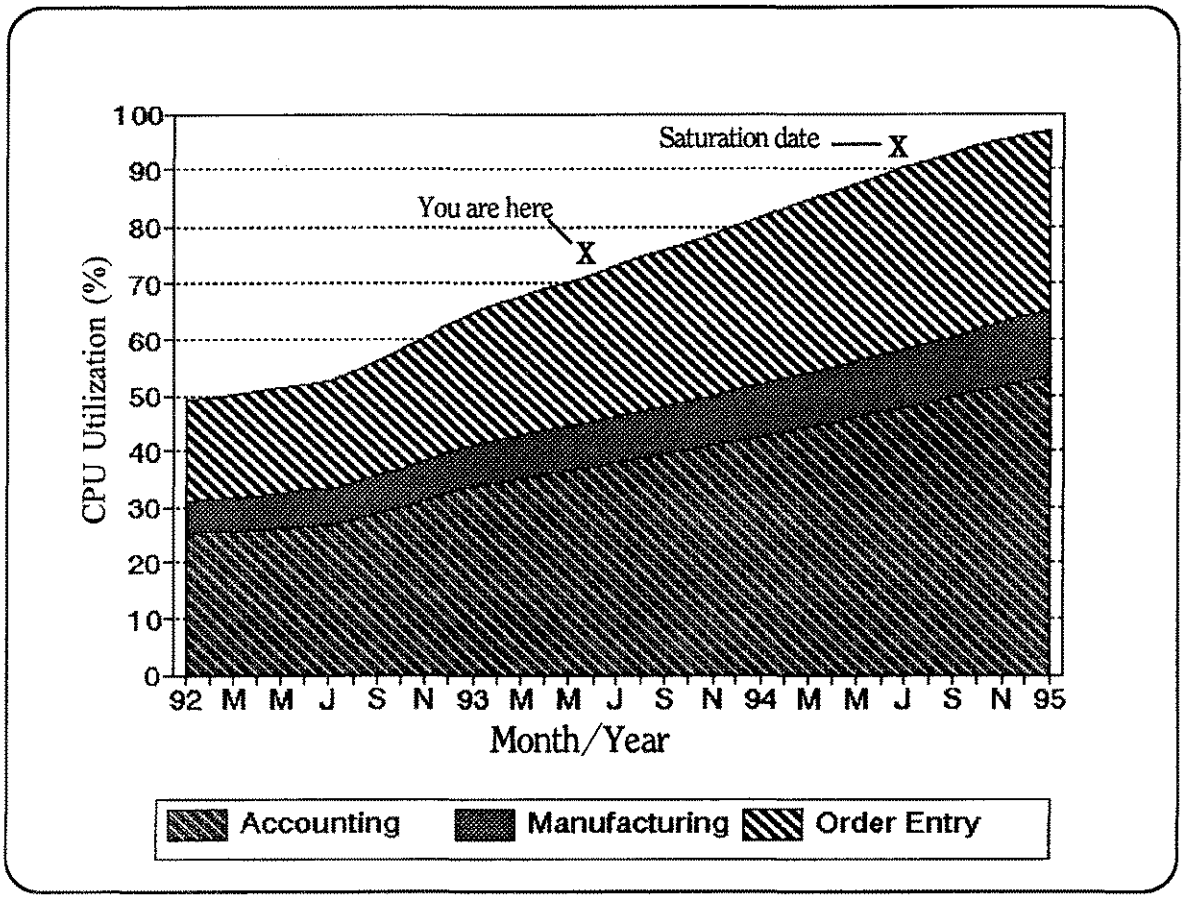
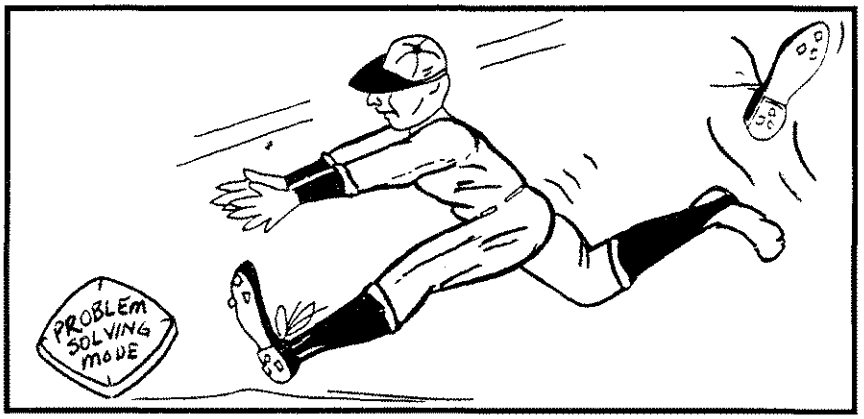


Figure 2.2 - Projected CPU Utilization with Saturation Date

Base Number Four: Problem Solving Mode

Problem solving mode involves less-than-crisis situations, which include identifying why certain programs are experiencing worse performance since a few new "enhancements" were added. Any well-rounded performance management game plan should include software application profiling. This is sort of a performance certification for a new application or enhancement to an existing application. I strongly feel that a resource consumption statement ought to be included along with every quality assurance checkout for programs.



Mankind is still in need of a Savior, for sin is still sin though we call it by many new psychological names. F.M. Swaffield

A resource consumption statement might include such things as:

- Disk space requirements for program(s) and data files.
- Average amount of CPU necessary for a single online or logical batch transaction (usually expressed in CPU milliseconds).
- Average number of disk accesses for a typical transaction.
- An estimate of memory requirements per program user.
- A statement of the approximate percentage of overall CPU that will be consumed given a certain number of users per day.

By combining this information (especially the last item) with what you know about the system from a historical and current workload usage basis, you are less likely to be "bitten" by the new application. By implementing such a preemptive problem solving approach, you will have, at a minimum, an estimate of the kind of impact you can expect when the new application is launched. Having worked in various support capacities for HP and software vendors, I have witnessed the embarrassing effects of focusing on application functionality and virtually ignoring performance implications.

Practical Tips for Covering the Bases

For crisis mode you should rely heavily on an online monitoring tool. Ideally this tool should "come up quickly" and have process information immediately available. If you don't have a tool, the MPE/iX command SHOWPROC can be helpful (Figure 2.3). I often recommend that system managers "burn" a single port and attach a low-grade terminal (remember those 26xx and 2382 terminals from Lost In Space?) to allow your performance tool to be always running. This will allow you to zero in on performance problems as they occur and not worry about bringing up your performance tool!

```
PUB: showproc   ;job =@

QPRI  CPUTIME     STATE JOBNUM      PIN   (PROGRAM)  STEP

C152  0:04.909    READY       S4    46    :SHOWPROC  ;JOB=@
C152  0:15.457    WAIT        S2    48    :RUN SOS.PUB
B100  0:06.318    READY       S2       36    (SOS.PUB.LPS)
C152  0:03.978    READY       S2          63    (SOSLOGX.PUB.LPS)
C163  0:02.392    READY       S2            64   (SOSCHART.CHART.LPS)
D202  0:01.461    WAIT        J6    53    :SOSLOAD
D202  0:08.910    WAIT        J6      40    (SOSLOAD.PUB.HOG3000)
C154  0:25.151    READY       J6          65    (INTER1.PUB.HOG3000)
C153  0:24.871    READY       J6          66    (INTER1.PUB.HOG3000)
C153  0:24.257    READY       J6          67    (INTER1.PUB.HOG3000)
C153  0:25.939    READY       J6          68    (INTER1.PUB.HOG3000)
C154  0:24.315    READY       J6          69    (INTER1.PUB.HOG3000)
```

Figure 2.3 - Sample Output from MPE/iX SHOWPROC

Also, it wouldn't hurt to run a few "fire drills" and perhaps outline a simple plan to assist yourself and others when a performance crisis occurs. Here are some essentials for a crisis checklist:

- Which processes are the high CPU, disk I/O, and terminal I/O consumers.
- Is there any response from user terminals? How about the console?
- Is your performance tool terminal (remember, the one that is always running) responding? If not, there could be a very high priority process (MPE?) that has gone berserk. It is for this reason that I also recommend that you run your performance tool in the background *continually*. This will provide an audit trail of past performance and could provide you with valuable data for a crisis post mortem.
- Was there any datacomm occurring at the time? Sometimes a renegade modem or MUX can flood the system with interrupts.
- Which users' terminals locked up first? Are they the only ones running the same applications...? Perhaps touching the same data bases?

The moral law is written on the tablets of eternity. For every false word or unrighteous deed, for cruelty and oppression, for lust or vanity, the price has to be paid at last. J.A. Froude

Do not minimize the importance of casual mode! Be sure to make time for you and your staff to sharpen your understanding of your system. It is really helpful, for example, to perform some application profile tests. Get with some of the data entry users and coordinate their actions with another person taking online measurements with a performance monitor. I have led many clients through such a test as part of a performance evaluation. You may gain a new appreciation of why system "viscosity" gets worse at times. On more than one occasion, I have helped clients discover newly added utility programs taking a large percentage of the CPU! If you were to occasionally perform some casual monitoring of major workloads, you might not have to call in a performance doctor! Here are a few tips for casual mode:

- Read at least one performance related article per month.
- Create specific performance "problems" and study them—after hours of course! I use a set of programs called Pig and Piglets. With these programs I can create mild, moderate, and severe performance situations. You could write a simple program yourself that might perform repetitive math calculations, disk I/O, etc. for specified quantities. This kind of simulation can be helpful in analyzing MPE idiosyncrasies.
- Study one item on the screen of your performance tool per day and monitor that item. Watch how it fluctuates.

For capacity planning we could suggest a number of things, but really the essential tips are:

- Always collect data! You will thank yourself when you need to go back and extract various resource and workload information. Think of this data as a future planning insurance policy.
- Use a good spreadsheet or graphics program on a PC that performs some simple averaging functions. This is valuable for looking at trends.
- Obtain a performance tool that allows for data trending and capacity planning.
- Attend a capacity planning class or consider trying a self study course in modeling by Dr. Arnold Allen (refer to vendor reference at the end of the book).

- Take the initiative to present weekly, monthly, and/or quarterly utilization reports. This will gain favor in the eyes of those you report to, since they can factor some of this bottom-line data into their budgeting and planning equations. If they have not had previous access to such graphically presented data, they *will* be impressed!

The best tip for problem solving mode is to begin profiling your various applications right away. Include the five areas noted under the Base Four discussion above.

? Questions/Discussion ?

1. Which base do you think is the most important at your shop?
2. What do you typically do in a crisis? How could you improve crisis resolution based on the above discussion?
3. Of the capacity planning methods listed, which one is most practical for your shop?
4. What steps could you take to procure more management commitment to effectively handle the bases?
5. Do you have written performance procedures to follow if the primary system manager is not available?

Notes:

I believe that banking institutions are more dangerous to our liberties than standing armies. Thomas Jefferson

"You cannot bring about prosperity by discouraging thrift.

You cannot strengthen the weak by weakening the strong.

You cannot help the wage earner by pulling down the wage payer.

You cannot further brotherhood by encouraging class hatred.

You cannot establish sound security by spending more than you earn.

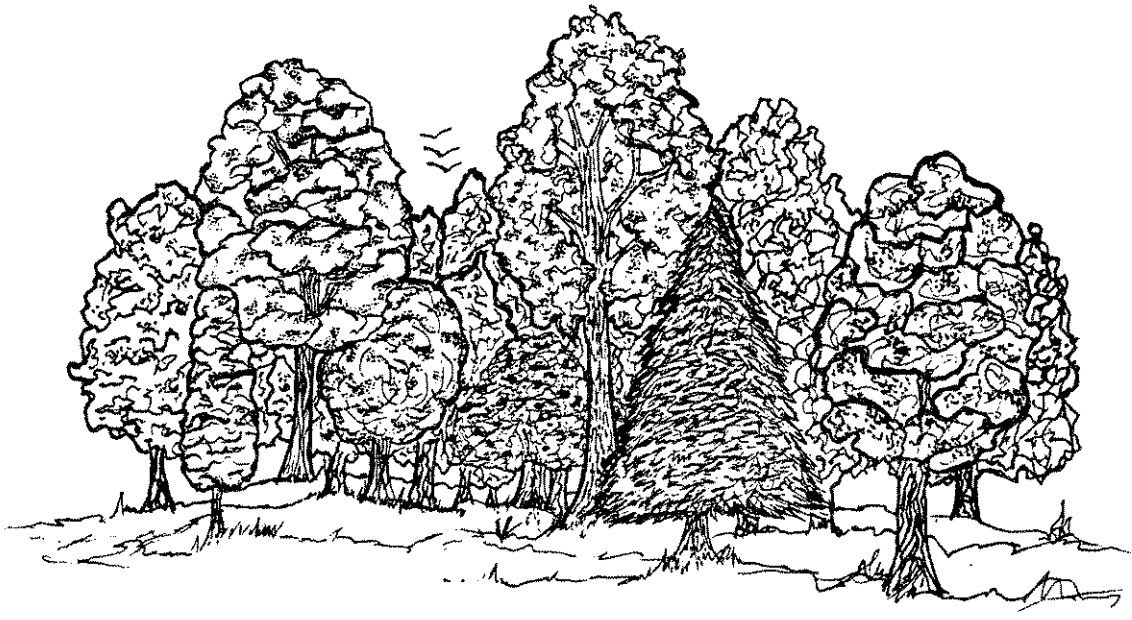
You cannot build character and courage by taking away man's initiative and independence."

Abraham Lincoln

Chapter 3

Performance Angles: The Forest, Trees and Groupings of Trees

There are three basic angles from which to view your system's performance: a view of the forest, a view of the trees, and a view of groupings of trees. From the "forest" perspective we see global resource activity. From the "tree" perspective, we see individual processes. And from the "groupings of trees" perspective, we look at workloads.



When you are viewing the forest perspective you are looking at global resource utilization. This might include overall CPU activity, memory usage, disk I/O traffic, etc. This data can be measured and reported with nearly every performance monitoring tool. The activity at this level provides you with the big picture. Every high performance system,

whether it's a hydroelectric plant, a Porsche or a multi-user computer, needs a way to communicate the vitality of its performance to the operators. Typical performance statuses we take for granted on a daily basis might be oil or water indicators on automobiles. An HP 3000 is no exception.

Face it, unless you have some way to measure the availability of your system's resources, you are flying blind. The ability to measure the supply and demand for the central processor is mandatory for any well-run HP 3000 shop. I am constantly dealing with clients who get caught by surprise when, all of a sudden, their CPU is "out of gas" and they are forced into an upgrade decision. Many times this occurs to the embarrassment of DP management. If a simple plan had been in place to take the system's global "pulse" on an on-going basis, then management could have taken decisive action well ahead of the crisis.

Figure 3.1 below shows a graphic, high-level view of some of the more important performance indicators such as:

- **CPU Bar:** Various busy, wait, and idle states of the CPU.
- **QLEN Bar:** The number of processes waiting and ready to use the CPU.
- **TRN Bar:** The number of terminal reads (often this is equal to user terminal transactions) occurring per minute.
- **RESP:** Global average response time of session activity.
- **PFLT and OC Bars:** Page fault rate and overlay candidate memory indicators.
- **I/O and QLEN Bars:** The number of disk I/Os occurring per second and the average number of I/O requests waiting in line to be serviced.

The Soviet Commissar of Education once said: "I find that religion is like a nail: the harder you hit it, the deeper you drive it in."
Watchman-Examiner

```
SOS/3000   A.39a (C) LPS   THU, FEB 14,   1991,   1:43 PM E:  00:07:55 I:  00:30
      2   10   20   30   40   50   60   70   80   90   100          2   10   20
CPU   |B          BC                                  CMMO   0%|   QLEN |          >|
TRN   |                                                    |  /min   RESP |     |
PFLT  |                                                    |  x100   CC   |     |
I/O   |R RW  WMM                                           |  /sec   QLEN |     |

PIN  J/S#      Session/User Name     Cmd/Program     CPU%   QPRI    #Rd  #Wr  LDV  #Tr  PRsp
 79  J27       PIGHALL, BOB.LUND      PIGLET1        6.4   CS154     0    0   10    0
 90  S30       BOB.LUND               CI             5.9   ------    1    0   21    2   1.7
104  S30       BOB.LUND               SOS            1.2   BS100     1    0   21    3   1.0

*****************************Process Summary By Application Workloads*****************************
No. Group Name   %CPU          %Disc I/O        Prompt Resp   #Transacts   CPU/Tr    I/O/Tr
  1:SYSPROCS    7.0 [  1.1]  79.1 [ 16.0]    - [   .0]     0 [   4]      -         -
  2:SESSIONS   21.5 [  7.8]  20.9 [ 32.8]  1.2 [   .4]     6 [217]     586         3
  3:JOBS       59.0 [ 48.2]   .0 [ 51.2]    - [    -]  59.0 [   0]      -         -

The CPU was used a total of 100.0 of it's capacity during this interval                <G101>
Compat. Mode to Native Mode Switch rate during this interval was EXCESSIVE             <GE03>
CPU Queue length indicates a SEVERE CPU bottleneck                                     <GE05>
Process CPU use by Sub-queue: AQ-.2 BQ-24.9  CQ-62.8  DQ-.0  EQ-.0                    <PI01>
This interval's "hog" process is S30 (PIN 62) and used 7.9% of the CPU                 <PI02>
This interval's highest disk I/O user was <sys> (PIN 6)with 49 I/O's                   <PI03>
This interval's highest Term I/O user was S30 ( PIN 104) 3 Term Reads                  <PI04>
Enter Command:
```

Figure 3.1 - Graphic View of the "Forest" Using SOS/3000 Performance Advisor

Some folks prefer not to be swamped with performance data. They prefer to view data in a graphic format as illustrated in Figure 3.1. Others prefer enough detail to "zero in" on specific current and cumulative data, such as that shown in Figure 3.2 below. There a few commercially available tools that can display this type of data in a variety of ways.

Ill-gotten gains do not profit, But righteousness delivers from death. Proverbs 10:2

```
SOS/3000 A.39a (C) LPS THU,FEB 14, 1991, 1:43 PM E: 00:07:55   I: 00:30
-------------Global CPU Statistics--------------------Global Misc Statistics----
             TOTAL BUSY:  100.0 [ 100 ]              #Ses 129  #Job 6  #Proc 614
AQ  .<[ 5]   Memory   5.8[ 5]    CPU QL    8[ 7]     CM to NM Switches  75[ 20]/s
BQ 6.3[ 5]   Dispatch  .3[0]     Launch/s 91[80]     NM to CM Switches  14[ 7]/s
CQ 42.5[28]  ICS/OH  15.8[15]    CPU CM%   3[ 1]     Transactions 641[8809](1243)
DQ  .0[ 0]   Pause     .0[ 0]    SAQ        17       Avg First Resp    .<[ .0]
EQ 29.2[42]  Idle      .0[ 0]                        Avg Prompt Resp   .3[ .1]
--------------------------------Global Memory Statistics-------------------------
Page Fault Rate  12[12]/s  Memory Cycles  0[ 3]     Overlay Rate   18[ 30]/s
Lbry Fault %      1[ 2]      Read Hit %   82[84]     Swap/Launch   .32[ .33]
--------------------------------Global Disk Statistics---------------------------
<Ldv>Rt/IO%QL <1> 3/ 6/1.08  <2> 2/ 4/.82 <3>  1/ 2/3.34  <4> 2/ 4/.03
<Ldv>Rt/IO%QL <5> 0/ 0/ .00  <9> 2/ 4/.33 <11> 1/ 2/ .13 <12> 2/ 4/.66
```

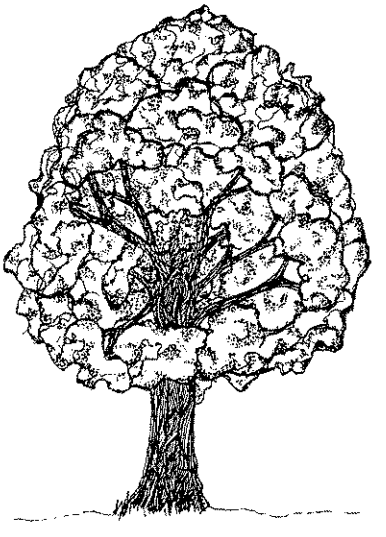
Figure 3.2 - Tabular View of the "Forest" Using SOS/3000 Performance Advisor

At the global-view level you are not asking who is using how much of the CPU but, rather, what is the impact of net CPU usage on main memory, total number of disk I/Os, etc?

Once you have determined the various green (OK), yellow (caution), and red (problem) zones for the global resources, the next question you will want to ask is, "Who's the Hog?"

A View of the Individual Trees

Individual trees represent processes within the forest of system activity. Since global resource usage is generally a sum of individual processes on the system, it is vital to also look at these consumers. This is particularly true if you are migrating from MPE V compatibility mode to MPE/iX native mode. There can be a significant waste of resource (CPU especially) by not compiling programs into native mode, and, to do this, you would want to know which processes are performing the most NM to CM and CM to NM switches. Figure 3.3 below shows some items to keep your eye on when tracking the performance of individual processes.



How sharper than a serpent's tooth it is to have a thankless child. Shakespeare

```
SOS/3000  A.39a   (c) LPS THU, FEB 14, 1991,    1:43 PM     E:00:07:55   I:00:30
PIN  J/S#   Session/User Name       CMd/Program  CPU %  QPRI  #Rd  #Wr  LDV #Tr PRes
  6  <SYS>  <system process>                       .4   CL152   0    0    -   0
 35  <sys>  <system process>          VTSERVER     .7   CS152   0    0    -   0
104  S30    BOB.LUND                  SOS          .8   BS100   0    0   21   1  2.8
 43  S28    KEN,BOB.LUND ci:DISCFREE              3.3   CS152   8    0    -   1   .6
 54  J27    PIGHALL,BOB.LUND          PIGLET1     8.5   CS154   0    0   10   0
 75  J27    PIGHALL,BOB.LUND          PIGLET1     9.0   CS154   0    0   10   0
 77  J27    PIGHALL,BOB.LUND          PIGLET1    12.5   CS154   0    0   10   0
 45  J27    PIGHALL,BOB.LUND          PIGLET1    18.9   CS156   0    0   10   0
 78  J27    PIGHALL,BOB.LUND          PIGLET1    19.2   CS156   0    0   10   0
 73  J27    PIGHALL,BOB.LUND          PIGLET1    20.5   CS158   0    0   10   0
```

Figure 3.3 - Resource Utilization View of the "Trees"

Some of the more important process items are:

- **CPU%** - The amount of CPU that is being consumed by the process.
- **QPRI** - The scheduling queue and priority of the process.
- **#Rd #Wr** - The number of disk I/Os this has process incurred.
- **#Tr** -The number of terminal transactions the process has performed.
- **PRes** - The prompt response time the user is experiencing.

It is also important to know the usage thresholds that represent slight, moderate, and severe bottlenecking for various processes. This will provide the visibility you need to determine the type of action necessary to combat a particular problem. The discussion on system pulse points in Chapter 6 of this section should prove invaluable in this area.

A View of Groupings of Trees

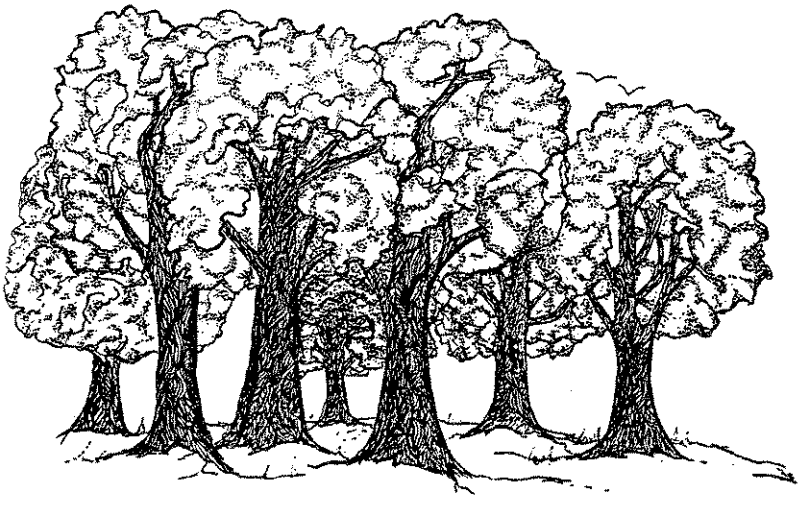
After viewing system performance from the global and process angles, many system managers are still left with a sense of unease when confronted with difficult (but relevant) questions from upper management, such as:

- How much CPU is being spent on our finance department?
- Since the order entry department folks have been complaining about excessive response times, just what is their average response time as an aggregate?

- What percentage of all the disk activity is due to general ledger batch job runs at night?

- Just what would be the net impact on the CPU if we increased the number of accounts receivable transactions by 50 percent?

Questions like these are concerned with the often-neglected aspect of "business units" of performance. How do logical business groupings of processes (workloads) impact the system's performance? In the forest of system performance, you want to know about the vitality of the pine, oak and fir tree stands.



It is extremely important to know the aggregate impact of certain users and programs so that you may plan for growth, charge appropriate departments and so forth.

```
SOS/3000  A.39a  (c)  LPS  THU, FEB 14, 1991, 1:43  PM  E:  00:07:55  I:00:30

----------------------Process Summary by Application Workloads-----------------
No. Group Name      % CPU       %Disk I/O     Prompt Resp   #Transact   CPU/Tr  Io/Tr
1:ORDER ENTRY     9.6[ 6.8]   7.5[ 8.0]      .<[  .1]    233[8979]      13      0
2:AR              5.9[ 7.6]   8.0[10.2]      .1[  .1]    143[7232]      14      1
3:MANUFACT.       4.4[ 2.3]   7.1[ 3.7]      .2[  .2]     55[1286]      24      2
4:PAYROLL         3.6[ 2.4]   3.0[ 4.8]      .1[  .2]     30[1433]      24      2
5:UTILITY         3.2[ 2.6]   8.4[ 4.8]      .4[  .2]     25[1475]      40      4
6:SYSPROCS        2.2[ 3.9]   2.0[ 4.6]      -[   -]       0[   0]       -      -
7:SESSIONS        2.6[ 6.4]   4.1[10.0]      .1[  .1]     59[3985]      14      1
8:JOBS           47.1[45.2]  59.8[53.8]      -[   -]       0[   0]       -      -
```

Figure 3.4 - Resource Utilization View of Groups of Trees

Figure 3.4 shows resource usage for various groupings of programs and users. It illustrates current resource usage for an interval along with an on-going average (in brackets). This figure shows eight examples of multiple user-defined workloads. The last three account for miscellaneous system processes (SYSPROCS), session processes that did not qualify in our major application categories (SESSIONS) and miscellaneous job processes (JOBS). Groups one through five represent the major applications for one company's system. The ORDERENTRY workload represents the most activity from both a CPU and disk I/O standpoint. Even the number of terminal transactions (terminal reads) is head-and-shoulders above the rest.

Workload data provides decision making information. On your system, you may have one application that consumes a majority of system resources. Wouldn't you like to know how applications are using resources on your system?

A graph to help you present this data to management is pictured in Figure 3.5. This application workload summary could represent a day, week or even a year. With information like this it is often easier to convince management of the need to off-load applications to a second system, procure more hardware, rewrite or optimize applications or just illustrate the effect of additional applications.

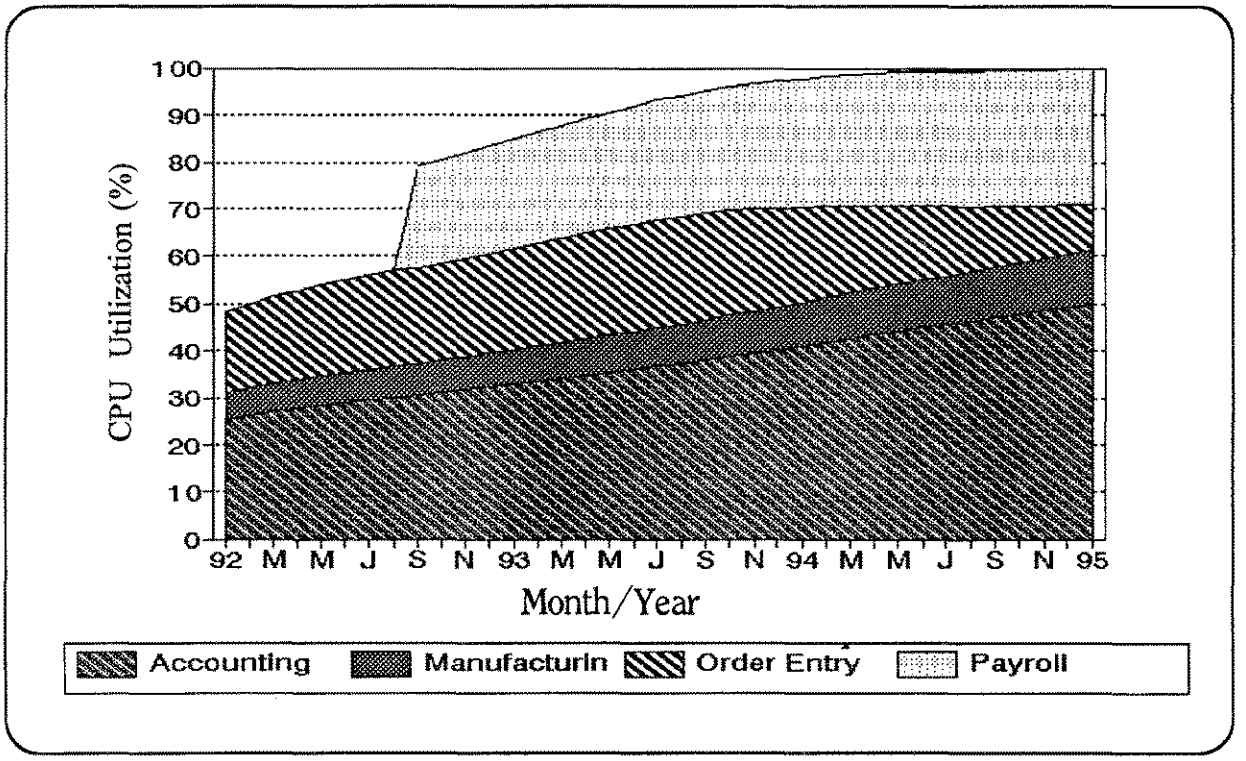


Figure 3.5 - Workload Trend Utilization

He who reigns within himself, and rules passions, desires and fears, is more than a king. Milton

Without knowing workload information it is difficult to perform "heads up" capacity planning. In fact, with queue network modeling and some statistical forecasting, it is necessary to have workloads defined properly. Oftentimes, strategic planning involves asking questions like, "What would be the approximate CPU utilization if we acquired a small company and take on all of their financials?" Showing performance from a business unit perspective is generally neglected by most but is very helpful information for DP management. Refer to Appendix A for more information on capacity planning utilizing queueing network models.

How can one proactively manage performance? The best way is to be sure all of the bases and angles are covered. Many shops I have encountered through the course of consulting or training have a monitoring methodology, but it is generally limited to one or two of the bases and angles. If you are serious about effectively managing performance, you need to cover all of the bases and angles.

Due to the "down-sizing" of our economy (and many DP shops as well), it behooves system managers to implement a thorough monitoring/managing game plan, one that delivers relevant information for management decisions, deals effectively with crises, etc. Cost effectiveness is also important. The heart of performance management is a monitoring tool. A good word of advice is to not lock into a contributed library tool or even a purchased tool that does not deliver all of the angles, bases and metrics essential for the long run. A well-rounded performance methodology will increase the "mileage" of any computer system by forestalling costly capital expenditures for new equipment and more accurately foretelling upgrade needs.

Everybody thinks of changing humanity and nobody thinks of changing himself. Leo Tolstoy

? Questions/Discussion ?

1. Over time, what are the top 10 "hog" programs on your system? Are they harmful? Do any need to be put into different scheduling queues?

2. What are the most meaningful *global* indicators to you (CPU Busy, Memory Activity, etc.)? How do you currently determine when a resource is saturated?

3. How do you currently view the angles? MPE commands? Tools? How could you more effectively do this?

4. Take a few moments to describe the major business functions of your system. Now define and describe each workload by users and programs for each. This will help you in your implementation of workload tracking.

Chapter 4

Nuts And Bolts Of System Performance

or

Just Enough Internals To Be Dangerous

To deal with system performance in a down-to-earth fashion, it is helpful to gain a basic understanding of some system internals as well as familiarity with the basic resources necessary to process data. In this chapter I cover functional components of the hardware and operating system. Consequently, this chapter is a bit windy... take a few breaks and you'll make it through just fine.

The discussion in this chapter applies primarily to HPPA (Hewlet-Packard Precision Architecture) systems (MPE/iX and MPE XL). However, much of the information presented applies to classic systems (MPE V) as well. I will note the differences. The goal of this chapter is to provide you with just enough internals to be dangerous!

In Chapter 5, I cover the primary system resources. This discussion will be an invaluable foundation if you wish to gain an understanding of significant performance "pulse points".

An MPE Process - The Basis of all Life on the HP 3000

A process is the basic entity of activity on an HP 3000. An orthodox definition of a process is: *"The unique execution of a program by a user."*

A user can be a real person or a pseudo person (a batch job user). This means that someone logged on as SUZIE,MGR.FINANCE can be running a program

A.P001.PUB.FINANCE. This user/program combination is considered to be a process. Consequently, this process gets assigned a PIN number that distinguishes it from all other processes. Each process can receive CPU service and this means that all other functions (disk I/O, terminal activity, etc.) can be performed on behalf of this process.

Throughout this discussion it is helpful to keep in mind the simple analogy of standing in line at a fast-food restaurant or at a store that has some kind of queueing system (take-a-number). Just as the person at the head of the queue or lowest number receives priority service, so a process must have the highest priority (signified by a low number) to gain attention from the CPU.

A process receives CPU service based on a couple of things. First, it must signify that it requires CPU attention. Interactive processes do this by pressing the RETURN or ENTER key. Batch jobs, since they have a seemingly never-ending appetite, almost always desire CPU time. For either type of process, all of the necessary pieces for that process (sometimes called the working set) must be resident in main memory in order for it to receive service. This means that the program code, data area, any libraries, etc., must reside in main memory. If any piece is missing, the operating system simply moves on to the next qualifying process.

An example of a very basic process is the Command Interpreter, affectionately known as "CI". Every user and job has a command interpreter process created on its behalf at logon time. This process really lies in a dormant state (a "SON" wait) for much of its life. Since it is dormant, there is really no impact on performance; there is no performance "cost" (memory, CPU, I/O) associated with having an idle process logged on to the system. However, a few table entries will be the only resource used.

It is the Command Interpreter process that provides you with the colon (":") prompt. At this prompt you then execute MPE commands (processed by the command interpreter itself) or run programs. Let's say you run EDITOR. A separate process is created. This son process is what interfaces with you and performs your bidding (writing letters, etc.). It is this son process with which we are usually most concerned with respect to performance. Other son processes could be things like order-entry, payroll, etc. These are the processes that do the "meat-and-potatoes" work on the system.

MPE Tries to be Fair But...

The operating system attempts to fairly distribute the CPU among all processes on the system. But it depends on the types of processes and any special rules that system management has implemented (such as the TUNE command or third-party utilities). In reality there are times when some users seem to get less CPU while other users who do not need as much get more. Next, we'll talk more about how this happens in the context of the dispatcher's policies.

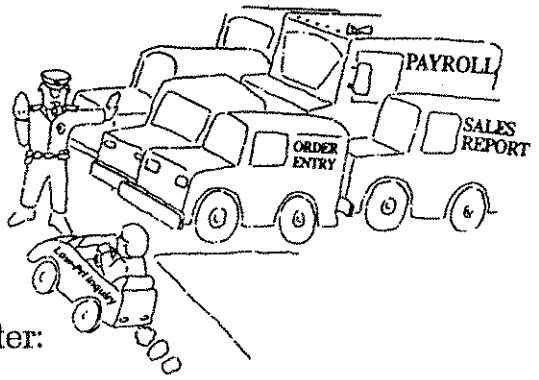
But How Does all of This Impact Performance?

As you may have guessed, any computer system has a finite amount of resources no matter how much CPU or memory is available. Even if only one process is running on the system, it will require some amount of CPU and probably other resources to complete its business. Obviously, when other processes are introduced to the system, the pool of unused resources diminishes to meet the increase in demand. The bottom line is: As processes that require attention increase, target resources become more scarce. So dealing effectively with this competition for system resources is what system performance management is all about.

MPE will try to satisfy the requests for service from all competing processes via a scheduling scheme. The heart of this scheme is an operating system agent called the dispatcher.

The MPE Dispatcher– Traffic Cop of the HP 3000

Anyone who is serious about HP 3000 performance must have some understanding of the dispatcher mechanism. Here we cover some of the important issues regarding process scheduling.



Consider this comment from one writer:

> *"...the dispatcher plays a large role in determining how well an HP 3000 ... meets user expectations of performance."* [13].

Today's music probably fits the times - savage, confused, unsentimental, often hysterical, highly sexual, mostly violent and hostile.
Richard Russell

The Dispatcher's Playing Field

In order to provide CPU service to each process on the system, MPE assigns priority numbers (which represent special regions) on a process-by-process basis. For example, if a user logs on, the priority of the subsequently created process will be 152 unless someone has altered the defaults. The region this process is assigned to is called the "C" subqueue. This process, at a 152 priority, will receive fairly high CPU attention. Other processes, depending on various criteria, will compete with this new process for the CPU's time.

But how does the dispatcher affect performance? Let's assume for a moment that all processes on the system are at the same priority. This means that, all other things being equal, they will share the CPU in a round-robin fashion. That is, each process will obtain a "slice" of the CPU while others wait for their turn. Back to our fast food analogy for a moment. Obviously, the more folks waiting in line for their order, the longer each one will wait. Their response times will increase. Accordingly, the perceived response times are somewhat a function of the queue length of customers (processes) waiting in line for the cook's (CPU's) attention.

Figure 4.1 shows the effect that increasing user requests for CPU service has on response time divided by service time. The bottom line is that perceived response time increases linearly up to a certain point, then the response time increases disproportionately with more demand.

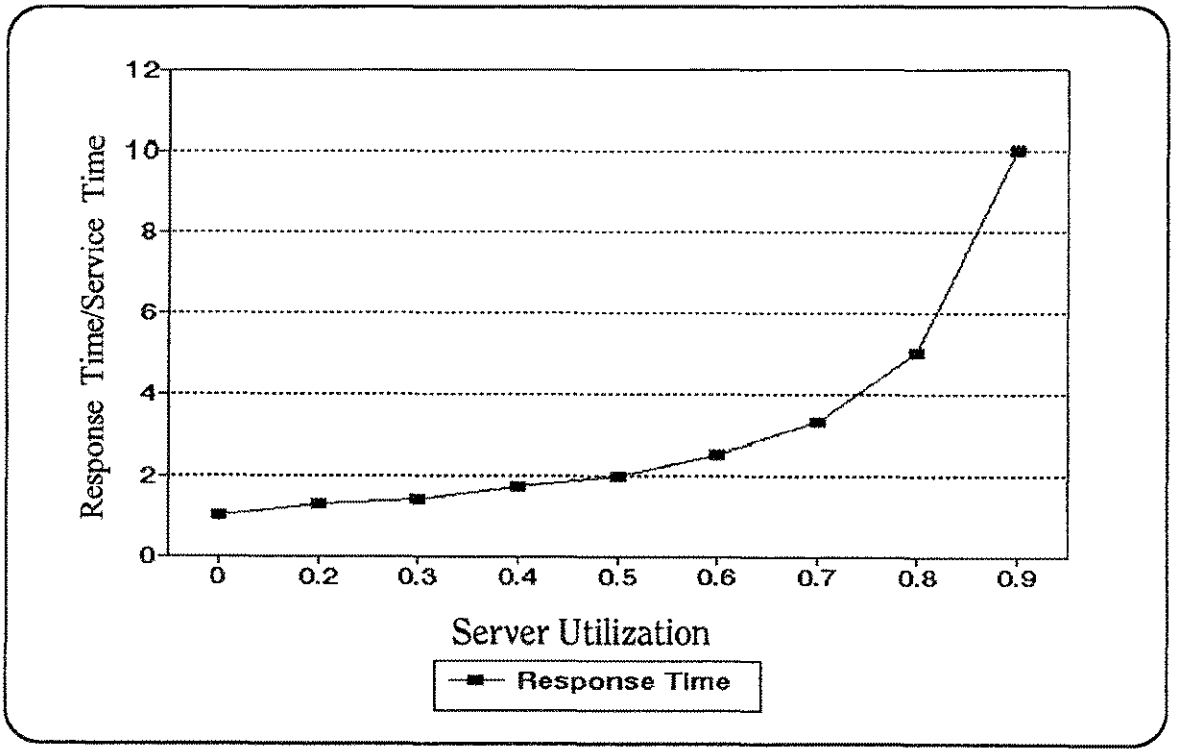


Figure 4.1 - User Response Increase With Demand

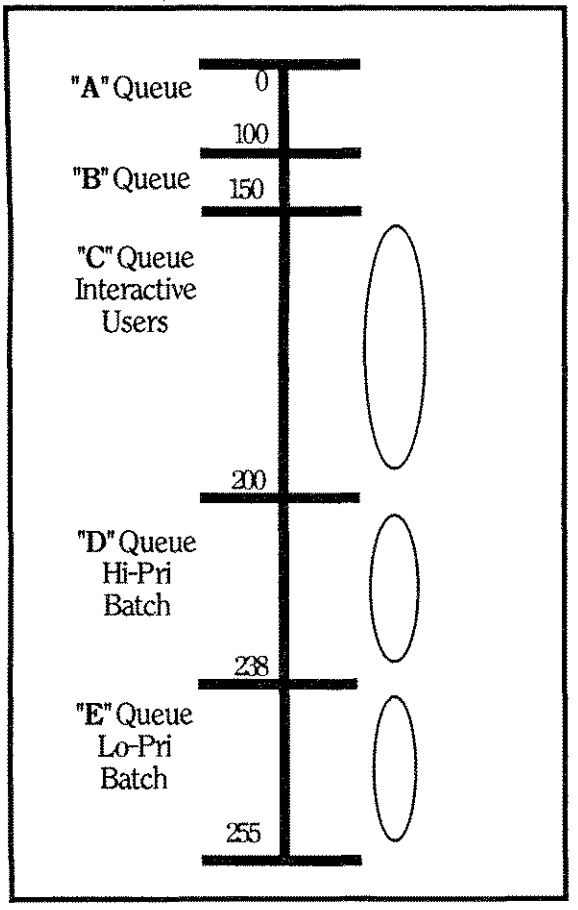
The question you should ask with respect to this graph is: "How does the dispatcher favor some processes while allowing others to receive less attention?"

The answer to this question lies in understanding the principles of the dispatcher. While an entire book could be devoted to this subject alone, I will provide a distillation of the dispatcher's basic functions.

The MPE Dispatcher—The Rules

In this section I discuss the MPE dispatcher mechanism. This discussion applies to MPE, MPE/iX, and MPE XL systems, but I use MPE as the catch-all term, unless otherwise noted.

MPE has five special subqueues that comprise one large queue with a numeric priority range of 0-255. Each subqueue has its own default set of boundaries called the base and the limit. Figure 4.2 illustrates this "playing field."



As mentioned, each process on the system is assigned a priority number. It is also assigned a characteristic of either exempt or non-exempt. This exemption has to do with whether the process is subject to the decay rules of the dispatcher. An exempt process retains its initial priority for the duration of its "life," while a non-exempt process may have its priority adjusted (or "boosted") many times during the execution of a single transaction. This exemption and decay/boost scheme allows the CPU to be shared amongst all processes in a fashion that reflects some underlying assumptions. Hopefully, these assumptions will cause the dispatcher to be a blessing rather than a curse to your environment! Some of the assumptions beneath the dispatcher's rules are as follows:

- Processes with high priorities (assigned to the A and B subqueues) are typically system processes. They get the CPU when they want it with few exceptions. They need priority attention in order to keep the "big wheel" of MPE turning.

- User processes that exhibit "hoggish" behavior are to be penalized in accordance with their CPU appetite. This explains why some processes receive increasingly worse response time for some transactions they execute.

- It is not in the best interests of the CPU to wait around while processes do not have necessary data. The CPU will move on to processes that are ready to use it.

Based on the above assumptions, there are different rules that the dispatcher follows. Though some of these rules are cast in concrete, many can be manipulated into altering the behavior of the dispatcher's activity. But why would you want to do such a thing? The simple reason is that there are so many different application environments. The default dispatcher cannot please everyone all the time. Consequently, HP has provided a number of "tunables," that is, options to cause the dispatcher to perform better/differently.

The bulleted items below contain the pertinent rules regarding the dispatcher's activity. Later in the book I discuss the various "knobs and dials" you can fiddle with to fine-tune the system's performance behavior.

- The AS queue ("A" subqueue) is reserved for high-priority system processes; these are exempt from dispatcher decay. The numeric range of this queue is approximately 0-99. It could be a career decision for a system manager if a user process were put in this queue!

- The BS queue contains both system and some prudently-placed user processes. Processes in this queue are also exempt from dispatcher decay. The numeric range of this queue is 100-150.

- The CS queue has mostly interactive user processes with a few low-priority system processes. Online users are typically operating within the range of 152-200. These processes typically "circulate" between this base and limit, based on subsequent rules below.

- The DS queue houses primarily high-priority batch jobs. Its priority range is 202-238. Typically, most jobs begin at the base priority of 202 and fall down to 238, where they compete with one another for the CPU.

- The ES queue is used by some as a background queue for lower priority jobs. The priority range of this queue is 240-253. On a busy system, it is quite a miracle for jobs to get any time in this queue since only "table scraps" of CPU will trickle down.

- For the CS, DS, and ES queues, a process begins its life at the base priority of the queue it is assigned to. These queues are sometimes referred to as circular queues since processes typically circulate between the boundaries (base and limit) of a particular queue. It is this circulation or "decay" feature that often causes radically different performance to be felt by users and jobs.

- A process within a circular queue is lowered in priority (by incrementing its priority number) when the process uses more than a certain amount of CPU time (a "quantum") before completing a transaction. This activity is referred to as dispatcher decay. If a process completes a transaction (batch job completes, terminal read issued, message file read, etc.) then its priority is reset to the base of the queue it resides in.

Be sure you refer to the section on MPE/iX performance for specific ways to manipulate the dispatcher's actions on MPE/iX systems. If you are on a classic machine, refer to Appendix C (although a number of the ideas presented in the MPE/iX section still apply).

Man--despite his artistic pretensions, his sophistication, and his many accomplishments--owes his existence to a six-inch layer of topsoil and the fact that it rains.

A Day in the Life of a Process

Now that you understand some of the basic rules of the dispatcher and its queueing structure, let's take a look at a few functions of the dispatcher as applied to process activity.

A process is considered "launched" when the dispatcher gives the CPU to that process. A process can only gain use of the CPU if all of the following are true:

- It desires CPU attention (signified by pressing RETURN key or if the CPU recognizes it as a perpetually hungry batch job). This implies that a process is not waiting for a resource.
- All of the process' working set (code and data) are resident in memory.
- It has the highest priority of all ready processes.

Once these three items become true for a process, the process is launched. After the process has been launched, it is said to be executing on the CPU. The process will continue to use the CPU until its transaction is completed or it hits one of many "brick walls." The frequency and type of these stop conditions can help you determine performance problems on an HP 3000. Some of the reasons a process will stop are:

- Another process "steals" the CPU away. This activity is called preemption—the process is suspended so that a higher priority process can use the CPU.
- It uses up its allotted time quantum (also known as a "timeslice"). For MPE/iX systems, a timer is started when a process is launched. When the timer expires the dispatcher decides whether the process is considered to be a "hog." A hog is defined (by the dispatcher) as any process that uses more CPU time than the quantum value for its scheduling queue. Thus, a process will have its priority increased numerically (less priority) depending on how much it exceeded the queue quantum value.
- The process voluntarily gives up use of the CPU to wait for some event to occur. This is called "blocking" (think of hitting a block wall!). Some examples are waiting for memory, semaphores, message files, disk I/O, etc.

- It is interrupted due to some completed operation (typically a disk I/O or terminal activity).

Using Wait States to Diagnose Problems

During the course of a process receiving CPU time and then waiting for events, it can be very helpful to quantify the amount of time spent waiting for events to occur. Often, I will glance down a column of process wait percentages to glean a bit more information as to why performance is not optimal. For example, if I frequently see processes within one application workload waiting on memory to become available, this provides another data point to determine whether there is enough memory on a system. Excessive process preemption can be a good indicator of CPU shortage. Processes waiting on TurboIMAGE access will show up impeded. This can point to the need to consider application design changes.

On systems exhibiting performance problems, it is important to determine whether the problem is global (every user is complaining), specific to a workload group (the finance department), or specific to a single process (just one user or job). If the problem is limited to one workload or process then it could have something to do with the dispatcher.

If an online report process is being starved of CPU (indicated by excessive preemption), it could be that dispatcher tuning might solve the problem. On MPE/iX, an OSCILLATE feature of the TUNE command will allow such CPU-famished processes to get another chance at the CPU. By implementing this OSCILLATE feature, when a process falls to the bottom of its queue, the dispatcher will bump it up to the top of the queue. The default action of the dispatcher is to use the DECAY rule, which basically causes a hog process to fall to the bottom of a queue and stay there until it completes a transaction.

If an entire department is feeling a performance "crunch," it could be that these processes are being punished too much, priority-wise. For example, let's say that there are one hundred processes executing short transactions. Let's also say the processes for the suffering department require a bit more CPU in order to process transactions and there are only 15 users for this application. In this example, many users are causing a few to be penalized with the default dispatcher rules.

Enter the System Average Quantum (SAQ -MPE/iX) and Average Short Transaction Time (ASTT - MPE V).

America, where, thanks to Congress, there are forty million laws to enforce ten commandments.

The dispatcher keeps a running calculation of the average amount of CPU used to complete interactive transactions (SAQ or ASTT respectively). If a process uses more than this average, its priority is reduced (numerically increased). In the above example, one hundred processes using a small amount of CPU keep the SAQ value unusually low. This causes a hog process to easily exceed this average value and therefore be penalized.

But what if these hog processes are involved in taking phone orders and the other ones are not extremely time sensitive? It makes more sense to allow all the processes to compete more fairly for the CPU or even to favor the order-entry workload. This is where the MINQUANTUM and MAXQUANTUM values can help. These are two TUNE command parameters. They control the lower and upper bounds for the SAQ or ASTT values.

Using the above example, let's say the 100 processes average 110 CPU milliseconds per transaction and the 15 order-entry processes need 350. The SAQ value will be approximately:

((100 * 110) + (15 * 350)) / 2 = 141.3

It is clear that when a hog process "comes up to bat," it will soon exceed the 141 SAQ value since it needs nearly twice as much CPU time to complete its transaction. At this point the dispatcher will begin to penalize the process by changing its priority. This is when you receive phone call complaints from users trying to take orders from customers! It is also obvious that the 100 other users are probably quite happy when it comes to response time. So what can you do?

To solve this problem you can use the MINQUANTUM and MAXQUANTUM parameters of the TUNE command to artificially change the SAQ value. Remember, the SAQ is a calculated value, ever changing. If you were to issue the following command:

TUNE ;CQ=152,200,*350,500*

The order-entry hogs would not be penalized until they use at least 350 CPU milliseconds (the MINQUANTUM value). This will cause a more equal sharing of the CPU. It is instructive to notice the SAQ value with your performance tool. You will find that it is mysteriously stuck at 350, even though the real calculated average is probably around 140. The general rule of thumb with respect to these two values is:

"Raising the MIN and MAX quantum values causes more equal sharing of the CPU between light and heavy CPU processes; lowering the pair causes the hogs to fall faster."

Take time to help your child set priorities and establish goals. If you don't help that child set schedules or make decisions on what is truly important, someone else will.

For a more practical look at some of the things you can manipulate the dispatcher into doing, be sure to read the material in Taming the HP 3000 Volume I and the MPE/iX section in this book.

Memory Management and Swapping Related Internals

Computer systems cannot function without memory. Memory acts as a scratch pad for the CPU's work. Nothing of permanent value stays in memory but it must visit memory before any useful work can be accomplished on behalf of users or jobs. MPE V systems usually have a range of 3 to 16 megabytes (Micro/3000 through Series 70), while MPE/iX systems have much more (bare functional minimum is 32 up to hundreds of megabytes).

Within the heart of the MPE operating system lies a memory manager mechanism. While both MPE V and MPE/iX have fairly sophisticated memory managers, the MPE/iX's has been greatly enhanced. The memory manager has as its goal to provide memory room to as many processes as possible, while at the same time minimizing waste.

The MPE/iX memory manager has the following characteristics and duties:

- Keeps track of the busiest pages of memory; these get to stay in longer.
- Kicks out infrequently accessed pages to make room for more urgent requests. Data gets written to virtual memory (transient space). Code simply gets overlaid (since program code acts as its own virtual memory).
- Utilizes CPU time to do its duties; thus, the more CPU spent on this activity, the more likely memory pressure exists. (Refer to the memory pulse points discussion in Chapter 6 for trouble thresholds.)
- Allocates permanent memory for the operating system.
- User processes vie with one another for all other available memory. These processes gain control of memory space based on how often the dispatcher gives them CPU control. So, the more often a process runs, the more likely it will retain its code and data in main memory. This can provide significant performance gains.

Poor is he who works with a negligent hand, But the hand of the diligent makes rich. Proverbs 10:4

- When a piece of data is missing from memory, the memory manager will issue a page fault. Then a swap will occur, pulling the desired page into memory.

- Since all pieces of memory on an MPE/iX system are the same size (a page equals 4096 bytes), MPE/iX does not have to find a large enough area to make a "fit." On MPE V systems, a "chunk" of data can be just about any size. Correspondingly, if an adequate area is not found, memory pressure increases; the memory manager has to create overlay candidates (pieces to kick out if necessary) to make room for requests.

- On both MPE V and MPE/iX systems, pieces of memory can be swapped out by marking them as overlay candidates. They can also be easily recovered before they are actually overwritten by turning off the overlay bit used to mark such pieces.

- The time it takes the memory manager to scan memory for available space is known as the clock cycle rate. If the clock cycle rate is fast, this is a bad sign; free memory cannot be found so the memory manager keeps hunting. The clock rates for MPE V and MPE/iX systems are radically different. For MPE V, I analyze the clock rate in terms of milliseconds per cycle (see Figure 4.3). For MPE/iX, the clock is expressed in cycles per hour. Chapter 6 will provide good, bad and ugly thresholds for these two values.

A few interesting ratios are key memory metrics. One is the ratio of the amount of time the CPU is paused for memory swapping. This value increases as memory pressure intensifies. See Figure 4.4 for a graph representing this value for an MPE V system (noted by more demand by processes). Anything over 5 percent is very bad.

Researchers continue to discover more and more about the importance of touch. It lowers blood pressure, increases resistance to pain and sickness, leads to a significantly more positive attitude, and can even add months or years to a life! G. Smalley

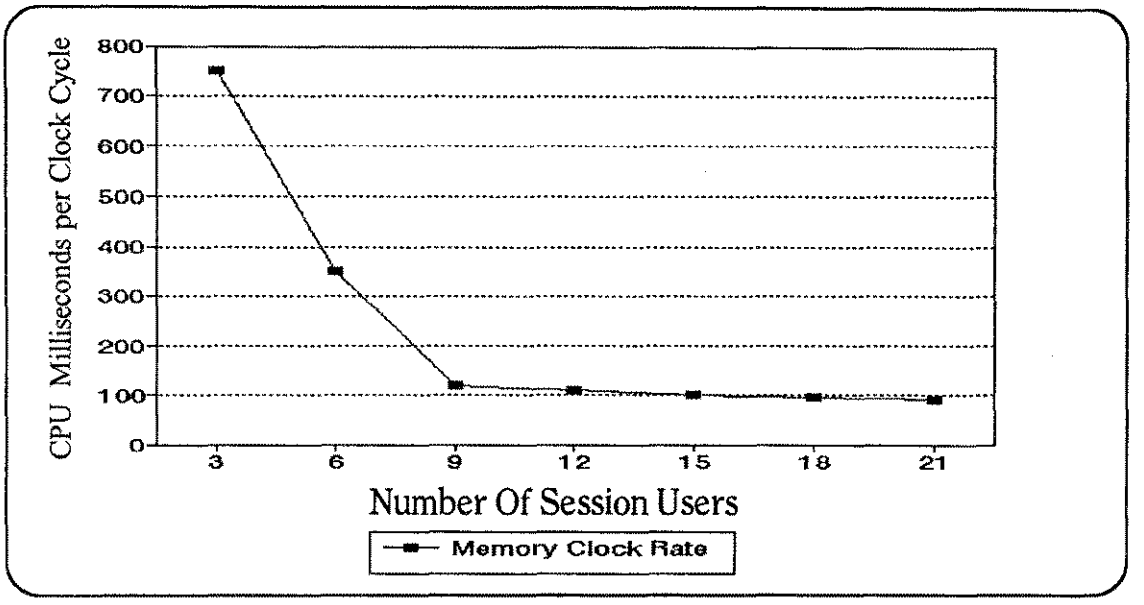


Figure 4.3 - "Classic" Memory Clock Cycle Rate

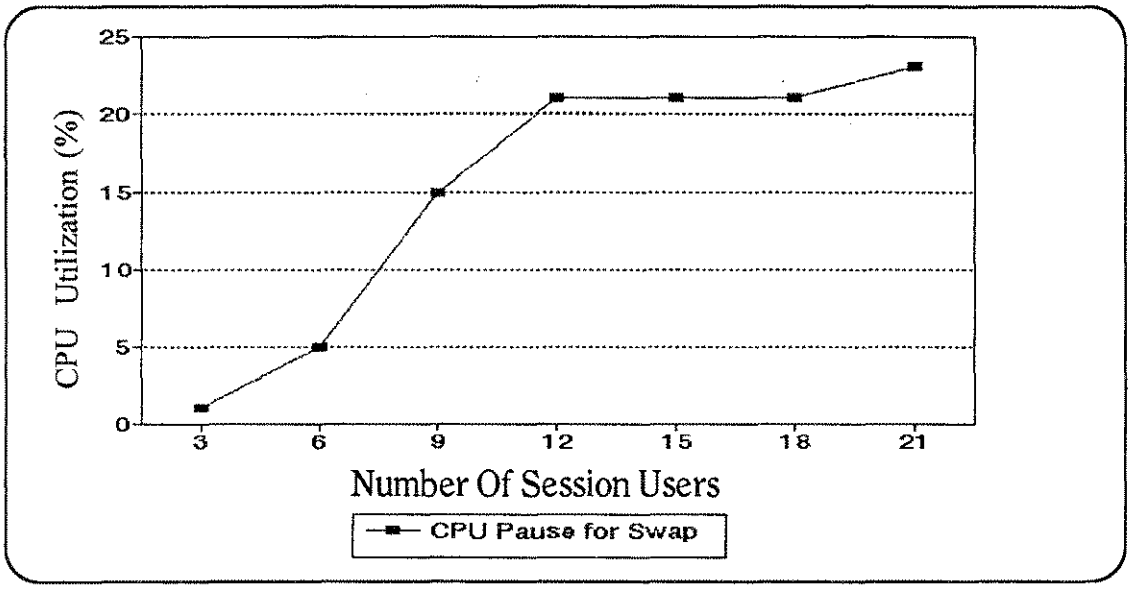


Figure 4.4 - "Classic" CPU pause for memory swap indicator

On MPE/iX systems, page fault activity and the percent of CPU spent on memory management can also be good indicators of the adequacy of main memory.

Conclusion

I hope this very brief discussion on internals will help you to better understand your system and prudently apply appropriate management actions. You will want to combine what you have learned here with the discussions in Chapters 5 and 6.

? Questions/Discussion ?

1. How would you "map" the priorities of your primary workloads? Are the majority of them at the top of the CQ? Do many spend time at the middle or bottom?

2. Are there any processes that you would like to favor or hurt? Which workloads could stand to suffer worse service in order to give better treatment to essential processing?

3. Could your system benefit from use of the TUNE command?

4. What are the wait states of your more important workloads? Do they indicate CPU, memory, I/O or other pressure?

Notes:

Chapter 5

Primary System Resources

Although there are usually many resources necessary to complete transactions, the three main ones are the CPU, primary storage (main memory), and secondary storage (disk devices). Some amount of each of these resources can be thought of as ingredients needed to perform a unit of useful work. This completed unit of work is generally referred to as a "transaction."

Thus, posting a general ledger record, querying an inventory level, or adjusting a receivables payment are all considered transactions. We'll discuss the nitty gritty of a transaction in Chapter 6, but for now keep in mind that some service from each of these resources is usually essential in order for transactions to complete.

To begin, let's turn our attention to the most primary of the primary resources: the CPU.

The CPU

The CPU is considered to be the engine within a computer system. The CPU consists of one or more hardware chip components. Its main job in life is to perform what it is told--fast and consistently. Since hardware components cannot think by themselves, they must be told what to do and when to do it. Thus, a programming language must instruct the CPU how and where to spend its horsepower.

Just as automobile engines are rated by the amount of horsepower they are able to produce, work performed by a CPU is usually rated in terms of Millions of Instructions Per Second (or MIPS). This term has been popularized in recent days and to some degree, misused as the sole

comparative indicator of one computer system over another. (For this reason, MIPS has been redefined as a "Meaningless Indicator of Performance" or "Meaningless Indicator of Processor Speed.")

In order to understand your system's performance, it is crucial that you learn to characterize how the CPU is spending its time. It is important to monitor these values over a period of time and not make conclusions with data from just one interval. By now are you sufficiently convinced about the importance of monitoring tools?

CPU Activity "States"

The various activities that the CPU performs have been referred to as CPU states. At the highest, least complex level, these states are divided into three categories: busy, waiting on events (paused), and idle.

CPU Busy

The busy state of the CPU occurs when the CPU is actually performing work; it is spending horsepower on behalf of some beneficial or overhead tasks. There are five active tasks usually reported by performance monitoring tools for MPE V systems and three for MPE/iX. They are as follows:

1. Busy on User Processing. For both MPE V and MPE/iX architectures, this category is considered to be useful CPU time spent on behalf of system or user tasks. Some examples might be calculating a dividend, executing spooler code to generate a report, or invoking a TurboImage DBGET intrinsic to retrieve a data base record. This time is broken down into interactive, batch and system process categories.

2. Busy on Allocating Memory. On MPE V machines, this is mostly time the CPU spent allocating room for necessary memory segments. On MPE/iX systems, it's time the CPU spent allocating and moving pages of memory.

3. Busy on Overhead Tasks. For both types of systems, this time is broken down into dispatcher activity (the process traffic cop) and handling interrupts from the world outside the walls of the CPU box itself (some terminal I/O, disk request completions, etc.). To better understand one of these, try this: When nobody else is on your system, simply run your performance monitoring tool on one terminal and hold down the RETURN key on another. Watch the CPU Busy on Overhead rise to great heights.

These fake transactions amount to a slap in the CPU's face. The CPU has to spend time actively monitoring and managing all such outside interrupts. These terminal reads are especially harmful to MPE/iX systems. Character mode applications can be costly in an MPE/iX environment. And many of these overhead events are not cheap, either.

4. Busy on Managing Memory Fragmentation. This event is only applicable for MPE V classic systems. As various sizes of disk data are brought into main memory, a checkerboard effect occurs. As in the case of disk space fragmentation, someone has to condense the many pieces of small, often unusable, areas into more useful large ones. Memory fragmentation must be dealt with in a similar fashion. Fortunately, you and I don't have to manage this problem directly since memory can become fragmented in a matter of seconds! MPE V has to spend some of its time dealing with this housekeeping chore. Of course, more time is spent doing this when a memory shortage is suspected. These events are described in more detail in the Primary Storage discussion later in this chapter.

On MPE/iX systems, all "chunks" of memory are exactly the same size: one page. For the purpose of our discussion, a page is defined as 4096 bytes. Since the smallest unit of memory (for swapping to and from disk) is a single page, and since any request can be broken up into an almost infinite number of pages, every single page area in main memory is usable. On one hand, this is very efficient; CPU management time is not wasted since memory fragmentation does not have to be dealt with! But on the other hand, some memory is wasted in a paging system. A one-kilobyte memory request would "burn" three kilobytes since the smallest addressable unit of memory is four kilobytes.

5. Busy Managing MPE Software Disk Caching. This CPU event occurs on MPE V systems only. If MPE software disk caching is enabled for at least one disk drive (via STARTCACHE), the CPU has to deal with searching cache domains (transplanted pieces of disk in main memory), managing placement of those domains, and a few other activities. Hopefully, the CPU expense is well worth the alleged reduction in physical disk I/O. Keep in mind that this overhead is *not* incurred for systems that utilize disk drive controller caching only (onboard within the disk drive cabinet). In contrast to software caching, controller caching does not consume host CPU time (except when due to the elevated disk interrupts). Therefore, I/O reduction does not steal CPU from user-productive tasks, but rather off-loads them onto the intelligence within the disk drive itself.

He that pursues two Hares at once, does not catch one and lets the other go.

CPU Pause

The second major category in which the CPU spends its time is considered to be passive yet anticipatory. This is time in which the CPU could have been busy performing productive tasks but was hindered in doing so because necessary pieces of data were missing. Usually the missing data is on disk. A process will lose usage of the CPU if it encounters a "block wall". One such block wall is a disk I/O request. MPE/iX combines the three following events into one category, which is referred to as "Paused". This paused time means processes would have been able to continue using the CPU but necessary information from disk needed to be retrieved.

The three possible CPU pause states for classic systems are as follows:

1. Paused for Disk I/O. This wait state occurs when a process requests a piece of data from disk on its own accord. An example of this might be: A user wishes to inquire on the August payment record for the Jones account. Internally, the file system searches memory buffers for the needed record. Either MPE V or MPE/iX will do this before it makes the decision to take the long, arduous journey to disk. The time that the CPU could have been busy but was prevented from being so on account of missing user requested I/O is logged to this category. Excessive time here may point to disk I/O bottlenecking or locality problems (see the discussion on Locality in Chapter 12).

It is important to realize that while the presence of this condition can be indicative of a disk I/O bottleneck, an absence may not prove the opposite. Here's why. A certain batch job cannot use the CPU until a disk I/O completes. The resultant CPU idle time is not counted toward pure CPU idle time, but rather CPU-paused-for-disk time. But what if another batch job wants to use the CPU and it has the disk data it needs to continue? That paused time will now evaporate and the second batch job will cause the CPU busy counter to be incremented. Some system managers actually keep streaming jobs until the paused-for-disk value is driven close to zero!

2. Paused for Memory I/O (MPE V Only). Wait time in this category is counted when a necessary code or data segment is found missing. Excessive CPU time spent here may point to memory shortage. The disk request involved was initiated by the memory manager.

3. Paused for Both Memory I/O and File System I/O Request. A third wait state is logged when both of the previously described events take place for a process simultaneously. That is, when a process is lacking both a user-requested disk I/O and a memory-manager requested one, the "both" category is updated. Excessive time in this category may point to memory shortage.

CPU Idle

The last state in which the CPU will spend some of its time is simply an idle condition. This is time that the CPU had nothing to do and it is not waiting on any events. Consider idle time to be CPU time in the bank.

To conclude, the CPU is the engine of any computer system. It is most important that you understand how it spends its time. In terms of having enough CPU, you can either use less (write better code), allocate it differently, or get more via an upgrade. Managing the CPU is perhaps the heart of performance management, especially on MPE/iX systems. If you are new to system performance management, you should pay particular attention to discussions about the CPU in subsequent chapters.

Primary Storage - Main Memory

Main memory is used by the CPU as a scratch pad or workbench. In order for any work to be performed, components of that work (program code and data) must be resident in main memory. For any process to execute, a minimum amount of that process' data and code must be brought into main memory from disk.

Since main memory is a finite resource (especially on MPE V machines), it has to be managed very efficiently to maximize its availability. Good user response times depend on an adequate amount of main memory being available, otherwise elevated disk activity could result (see the discussion of virtual memory in the Secondary Storage section of this Chapter). The result can be elevated user response and batch throughput.

The memory manager on MPE V spends some of its time making sure there are large enough free pieces of memory available to accommodate incoming data and code segments. Memory fragmentation is a constant concern for the memory manager. When demand for memory space becomes excessive, memory fragmentation becomes an acute problem, draining productive resources.

This checker-boarding effect can be combatted in a way similar to the way disk space is compacted. Since all memory activity is done on the basis of a page on MPE/iX, memory fragmentation is not an issue for these systems. Three types of memory compaction occur on MPE V systems, depending on prevailing conditions.

The first of these is called "neighboring." This compaction mechanism is the process of knocking out the wall that separates two adjacent small memory regions in order to make them into a single, larger one. This event occurs almost routinely as part of the system's housekeeping chores. I know of no supported way to measure this event.

The second way MPE V handles memory fragmentation is sometimes referred to as "local garbage collection." This event occurs as a matter of course when the memory manager is cycling through memory, marking little-used memory segments for removal. The memory manager will look to either side of a segment it is thinking about removing and try to combine any small, free pieces into one large piece. In performing extensive memory studies, I have not found any real significance in this event. Even under known stressed-memory shortage conditions, there does not seem to be much significance to the absence or presence of local garbage collection, apart from the fact that some CPU time must be burned to drive this task.

The third way in which MPE V handles memory shortage due to fragmentation is called "global garbage collection." This action occurs when the system is under memory pressure, *and* when idle CPU is available, *and* MPE software disk caching is not enabled (for systems with two megabytes of memory or greater). Global garbage collection is the process in which MPE V locates the largest free piece of memory and tries to make it bigger (within a bank of memory only). Its consistent presence is usually indicative of a memory shortage.

The bottom line with memory on either architecture is to have enough! Memory for classic systems is available very inexpensively. For either system, it pays to shop around for deals from third-party memory dealers.

A good rule of thumb for necessary memory on MPE V systems is simply to max it out! This memory is available for practically give-away prices, so if you see hints of memory shortage, add some. HP typically recommends between 16 to 30 megabytes for MPE/iX systems plus .5 to 1 megabyte per user and 2 to 4 megabytes per batch job. The following formula developed by Michael Hornsby might even be better:

 # Megabytes = (#Users * UA) + (#Disks * DS)

Where

UA is the user-application factor and equals:

.90 - CM applications.
.85 - OCT major applications.
.80 - MIX of CM and NM.
.75 - NM major applications.
.70 - CM single application system.
.65 - OCT single application system.
.60 - NM single application system.

DS is a disk-space factor and equals:

4 - Mostly batch applications.
5 - Mix of batch and interactive applications.
6 - Mostly interactive applications. [14]

Figure 5.1 - MPE/iX Memory Formula

Secondary Storage - Disk Drives

Since main memory is usually limited to a dozen or so megabytes for MPE V systems (perhaps a few hundred megabytes or more on a mid-size MPE/iX system), permanent data has to be kept elsewhere, generally on disk devices. Access to the data residing on those devices has been the subject of untold performance enhancement articles and mechanisms.

The real problem lies in the fact that the time it takes to retrieve data from an electro-mechanical device can be much slower when compared to purely electronic, speed-of-light transfers.

Another use of disk drives is to keep overflow data that cannot be contained in main memory. Dynamically sending out data temporarily to disk is referred to as a virtual memory scheme. On MPE/iX systems, the area reserved for this disk activity is called transient space. These regions are reserved on disk for the temporary containment of data that has not been recently used. Basically, a virtual memory scheme is a cheap way to make main memory look bigger to the system. But the major pitfall to this mechanism is that disk I/Os are incurred whenever access is required. This can slow down performance.

He who walks in integrity walks securely, But he who perverts his ways will be found out. Proverbs 10:9

When an MPE V system runs into memory shortage, response times become increasingly worse, but on MPE/iX systems, a lack of memory can usher in bad response times very quickly. This is partly due to the fact that HPPA systems were designed with large amounts of memory in mind. A shortage will incur disk accesses for transient space paging. Keep in mind that when a secondary storage I/O occurs, your transaction response time is negatively impacted.

Disk Space

Who isn't aware of this resource? You can be short on CPU, memory, and even have disk I/O bottlenecking, but the system will still continue operating, albeit much slower. But if you are out of disk space, life comes to a stop on planet Earth. Of course, an adequate amount is necessary to sustain system activity. Your job, as with the other resources, is to ensure there is enough now and into the future.

To make sure you have enough, you will need to monitor and manage it just as with the other resources. Some things involved in managing disk space are:

- Perform periodic checks on total free space as well as fragmentation levels. Keep in mind that you might have enough total space, but it could be broken into so many pieces it becomes, effectively, unusable. This is especially true for MPE V systems.

- Regularly search and destroy files that have not been recently used. Using a tool such as MPEX (VESOFT) will help you here.

- Remember to plan for future resource capacity in tandem with all resources. This means when you need more of one resource, there is a good chance others also could be getting scarce. When considering a CPU upgrade, for example, be sure to consider more memory and disk space.

Other System Resources

There are other resources on the system that impact transaction turnaround times. While they are not properly thought of as resources, you might loosely think of them that way. These include, but are not limited to, system table entries, hardware constraints, datacomm delays, application and system locks and latches, etc.

System Tables

MPE V systems have suffered system table limitations. Improper configuration of certain tables could negatively impact performance by causing job aborts, system failures, etc. Though probably not the cause of general slow downs, they do need to be configured high enough to prevent an impedance on the system. If you use a classic system, I recommend that you read Chapter 9, "Tuning System Tables For Optimal Performance - Fact Or Fiction?" Table management is performed by the operating system for HPPA systems. Tables are still a problem for MPE/iX but the limits are simply higher! Though of questionable value for improving performance, table tuning can sure be fun for "hacker" types!

Hardware Configuration

Hardware configuration is more of a concern for classic systems than for HPPA. I covered many of these issues in Volume I (see Appendix C).

DataComm and Networks

What about network (NW) and datacomm (DC) bottlenecks? There are two issues involved. First, NW or DC processes actually are running on the system. Consequently, they use CPU and memory. Often, they execute at a very high priority and can certainly impact CPU usage.

The second issue involving particularly NW and DC performance impact is external to the Host HP 3000. When a remote node initiates a transaction, the request travels through the NW "cloud," receives host attention, goes back through the NW and, finally, returns to the user. While this entire process can typically transpire in a few seconds, network performance is becoming an increasing area of concern for many sites. Traditional

Hugging never goes out of style. The next time you see someone you care about, bless them with your embrace.
G. Smalley

performance monitors do well in measuring response times, but are unable to cope with measuring responses outside the "four walls" of the host. To measure network response, you'll need to place network "sniffer" devices in line (best: one box at the user end and one at the host end). Proper synchronization of these devices can produce three important statistics:

- Network traversal time.
- Host processing time.
- Total end-to-end response time.

I have encountered situations in which the host was blamed for a poor response environment when the network was the guilty party, and vice versa.

Locks and Latches Performance Constraints

One fairly common area of performance bottlenecking is that of application locking. Locks represent logical impedances to transaction throughput. Though technically not a resource, application and data structure locking needs to be considered at the performance engineering stage of an application's design.

Also, it is important to watch for bottlenecking when adding more users to an application. This is because each data structure and application have a theoretical upper limit as to the number of transactions that can be processed with reasonable response times.

I consulted at one site that could sustain great response times when the transaction volume was less than 1000 per day. But when business grew to where 5000 transactions were necessary, a good response was 15 to 20 seconds. The primary cause of this was a poor data base locking strategy. Processes had plenty of CPU and memory available, but were literally locked out from getting data since other processes had the data base locked. Adding more CPU or memory would only provide a small gain in performance.

Three areas you need to consider for MPE V and MPE/iX systems are:

- **Process Handling** - Spawning too many processes and/or improper communication between them can create a logical application constraint.

- **Character Mode Terminal I/O** - The terminal I/O subsystem (especially MPE/iX) can be swamped in such a way as to create a logical bottleneck for other users. Consider the following:

Miracles and truth are necessary, because it is necessary to convince the entire man, in body and soul. I should not be a Christian but for the miracles. Blaise Pascal

"A good metric to use if the terminal I/O subsystem is being over- used by your application is the terminal read-to-write ratio. A common terminal read-to-write ratio for character mode applications is five to seven terminal writes for every terminal read. When applications far exceed this ratio, it often means that the terminal I/O subsystem is being over used and possibly becoming a bottleneck for the application."[15]

- **Database Locking** - A locking strategy that does not accommodate the throughput desired for a particular application will produce poor response times. Diagnose this problem by using a performance tool that reports the percentage of time a process is waiting due to impedes. These are typically database locks. Any consistent amount greater than 30 percent or so should be considered suspect. An application change may fix the problem.

Conclusion

This short overview of primary resources will equip you to better diagnose and remedy performance problems and is virtually required reading prior proceeding with Chapter 6.

? Questions/Discussion ?

1. Do you have enough "engine" to drive your workloads?
2. Could you quickly show the average CPU Busy, Pause and Idle for a given day?
3. How much pause time occurs during your heaviest non-batch processing period? What does this imply?
4. Based on the memory discussion, does your system(s) have enough?
5. What plan do you have in place to keep from being "bit" by a shortage of disk space?

Paul McCartney has stated that all the Beatle music recorded after 1965 was produced while either on LSD, alcohol or a combination of both.

Writers of novels and romances in general bring a double loss on their readers, they rob them both of their time and money; representing men, manners, and things, that never have been, nor are likely to be; either confounding or perverting history or truth, inflating the mind, or committing violence upon the understanding.

Lady Montague

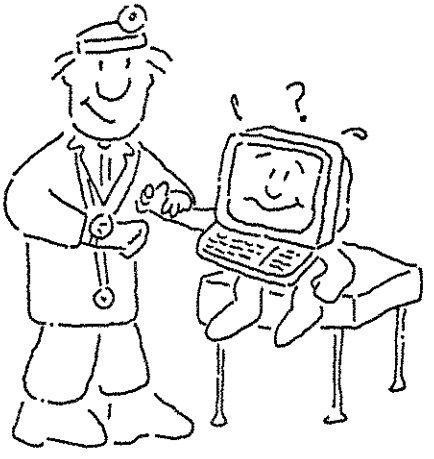
Chapter 6

Measuring System "Pulse Points"

The heart of performance management is measuring and interpreting the HP 3000 pulse points--key resource indicators. Your success in HP 3000 self-care stands or falls with your understanding of these pulse points. The ICCM Capacity Management Handbook says:

> *"Just as it is necessary to measure other processes and equipment as a means of knowing how well they are operating, so too it is necessary to measure computer systems to better understand their operation and performance... Thus, it is typically one of the responsibilities of the performance analyst to determine what to measure, when to measure, and how to apply the results."*[16]

In this chapter I will discuss what to measure and how to apply the results. It is important to realize that some of the pulse-point recommendations may vary over time. As operating systems evolve, resource indications can change their meaning and value. At this point in the game (1992), I do not expect any changes with MPE V, but perhaps some with MPE/iX. So, to some extent this information will be a moving target! If you are reading this book sometime significantly after 1992, you may want to try to contact me at the number listed in the back of the book to get a current status on these indicators.



System Performance "Self-Care"

In recent years it has become popular for people to monitor the health of their own carbon-based modules (human bodies). Popular magazines abound with all sorts of new high-tech devices and remedies designed to assist the amateur self-care enthusiast in his or her quest for optimum health. Most folks these days are aware of the importance of occasionally checking one's blood pressure, since this indicator can be a helpful sign of health or the lack thereof. In the same way, monitoring a computer system's pulse points has become essential to properly managing an HP 3000.

If you are committed to getting top performance at your company, you must learn how to identify current and impending HP 3000 resource bottlenecks. In this section, I focus on a few critical pulse points that deserve periodic monitoring. I discuss some that are important for response time, CPU, disk I/O, and memory; both MPE V and MPE/iX systems are addressed. It must be said that the values discussed are based on talking to other peers in the industry, lots of original research, and performing many performance and capacity analyses on both sick and healthy HP 3000s.

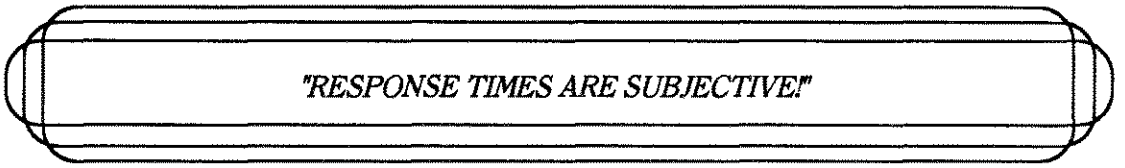
I have always said, I always say, that the studious persual of the Bible will make better citizens, better fathers, and better husbands.
Thomas Jefferson

Response Time Performance Measurements

The discussion of response time could be very lengthy since there are many different angles from which to view it. But I'll keep this discussion basic and to the point. Response time is generally thought of as the kind of service users are experiencing. Again, think of this the same way you would standing in line at a fast food restaurant. Response time is the time from when you place your order to the time you receive the goods.

The question is, how do we measure response time? One way is to measure from the time a user presses the ENTER key to the time a prompt appears. This may involve a little bit or a lot of screen paint time, which could obviously skew the response reading. If users are performing online reporting, is it more accurate to define response time as the time the ENTER key is pressed to when the first character appears on the screen? Some think so. At any rate, you need to decide for yourself which measuring stick is best for your purposes in assuring less than "n" seconds of response time for your shop. Is it the prompt or first response that is most meaningful for your environment?

Overall average transaction response times (ENTER to first character on the screen or to prompt if not much data is printed) are generally acceptable when they are less than two or three seconds. But keep in mind that Goertz's First Law of Response Times states:



Jason Goertz (HP 3000 guru and Chinese food connoisseur) stated this emphatically in a class I took from him years ago, so I wrote it down! What is acceptable response time at one company may mean a career decision for the system manager at another. Keep in mind that your primary goal is not just to ensure an under "n" seconds response, but rather a consistent response that meets your corporate DP goals. I find that users can actually work a greater-than-five-second response time into their operating style. If responses are long, they will simply shuffle some paper, etc. in between prompts. Consistent responses will keep users relatively happy, while a one-second-then-ten-second oscillation could be grounds for a kangaroo court, with you as the defendant!

Another wrinkle in measuring response times involves character- or block-mode transactions. In the case of block mode (ENTER key), a user fills in a form on the screen and then presses ENTER. The entire "data hand-grenade" is sent to the host. A performance monitor would report one terminal read for that user process. Conveniently, the user thinks of that activity as one transaction also. The response time for that transaction will be quite the same as if you timed it with a stop watch (assuming no significant datacomm delays). So far, so good with block mode.

The response-time plot thickens with character mode activity. Typically, a user will enter a field of data, press RETURN, enter more data, etc. This continues until all the data has been entered and the final RETURN is pressed. At this point the user has completed one logical transaction. Herein lies the problem. MPE may have seen a number of transactions because one terminal read is equated to one transaction. So, while the user thinks that one logical transaction has completed, your performance tool will report many physical transactions. Each of the transactions will have relatively short response times, except for the final one. The final RETURN initiates the entire transaction's execution. The entire response map for such a character mode application might look like what is shown in Table 6.1.

Response Time (sec)	Transaction Count	Comments
.05	1	Field-to-Field
.1	1	Data Entry
.075	1	
.05	1	
.1	1	
.1	1	
3.5	1	Final RETURN
.57 (average)	7 (total)	

Table 6.1 - Character Mode Response Times

So you see, an average response time of .57 seconds for our example in Table 6.1 is somewhat meaningless. The user perceived 3.5 seconds as the response time for the transaction, while a performance monitor reported an average response time of .57 seconds for seven transactions.

Practically, how can we make sense out of response measurements? If your applications are block mode, you will find very accurate transaction counts and response times reported by your performance tool on MPE/iX systems. For block mode MPE V workloads, VPLUS terminal status reads are counted as transactions and, thus, skew the true values.

If, on the other hand, your applications are based on character mode, you are stuck with somewhat meaningless response numbers and transaction counts (at least from the users' perspective). For purposes of monitoring and tracking relative changes, the response time values are fine. A daily average response time of .75 seconds may equate to 2 seconds for the users' final RETURNs. Six months later, an average reported response of 1.9 seconds may be enough of an increase to warrant a performance analysis.

The Anatomy of a Transaction

A transaction occurs when all the input from a user is obtained and sent to the host and then a prompt is returned. As I mentioned in the response discussion, the plot thickens when your application uses carriage RETURN input rather than the ENTER key since each press of RETURN

signifies to the operating system that a transaction (terminal read) was performed.

Performance tools will simply equate a terminal read with a transaction because the operating system does.

A superior way to report actual transaction counts and response times is to provide instrumentation for such measures within an application. You might try creating STARTTRAN and STOPTRAN routines, which could increment counters and keep very accurate response and transaction counts. These routines could be quite efficient, and they can be enabled/disabled at will.

Transactions can be thought of as logical units of work that we are trying to perform. But how is a transaction defined, what are the components of a transaction from a system versus user standpoint?

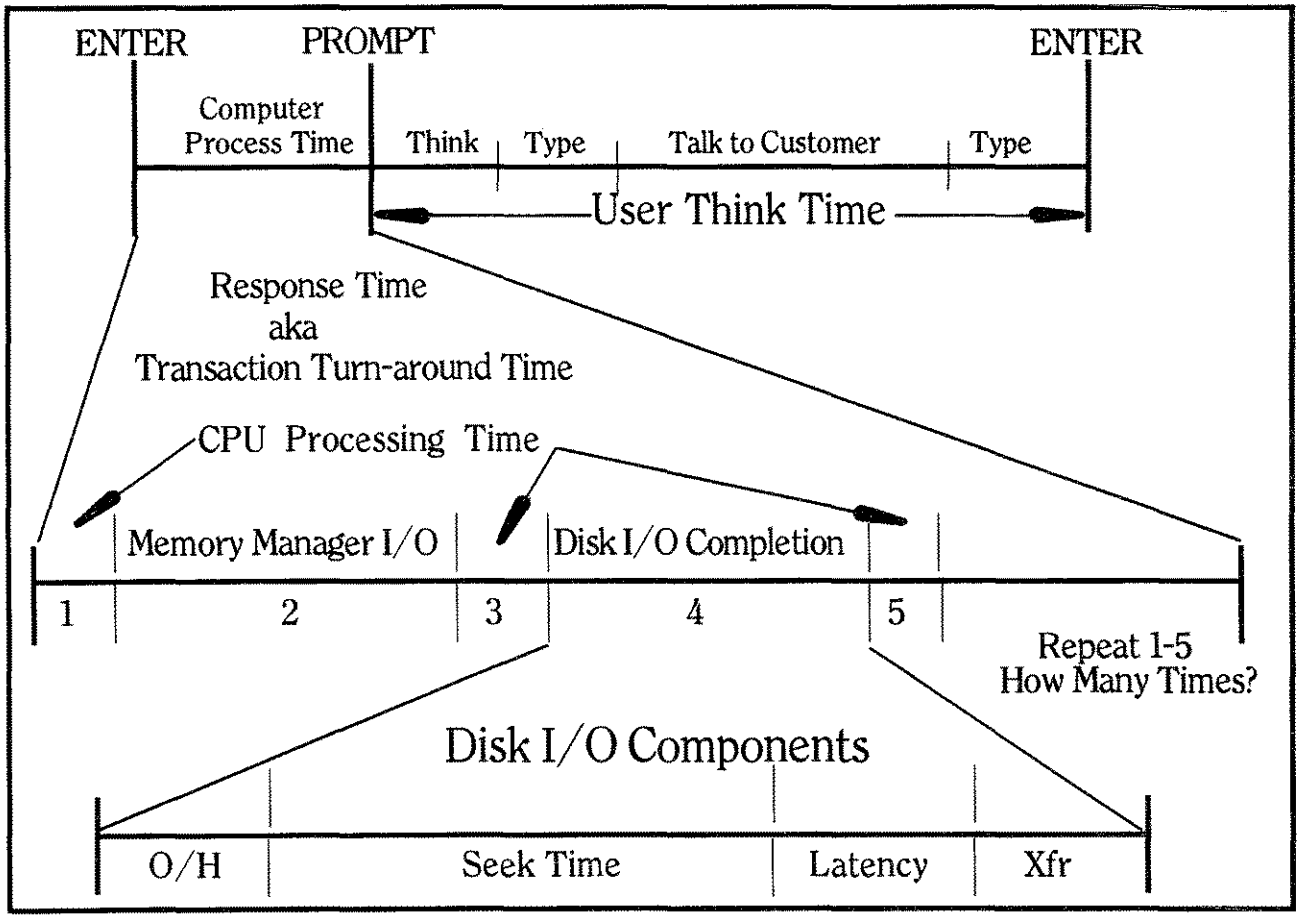


Figure 6.1 - The Anatomy of a Transaction

Thomas Jefferson in 1801, as president: 1) Cut taxes 50%; 2) Reduced the federal debt 50% in eight years; 3) Fired all tax collectors.

Figure 6.1 is a detailed breakdown of an interactive transaction. Notice the various types of services it requires. CPU, memory and disk I/O are all usually necessary to complete a transaction. Many other events also may be involved that stop and re-start service on behalf of the transaction many times prior to the user actually receiving back his prompt.

Transactions and response times are what we base our service levels on. Before we go on with the discussion of resource pulse points in detail, it is important that you clearly understand the elements of a transaction and response times.

Typical HP 3000 Resource "Measuring Sticks"

In this section I present some of the highest leverage performance bottleneck indicators. Please keep in mind that some of this discussion will be debatable. Specialists differ somewhat on this subject (in case you didn't know!). The following metrics are a distillation of my time personally supporting over 150 HP 3000 sites (working for HP and VARS), doing scores of in-depth performance analyses, countless hours of research using simulated loads, training hundreds of HP 3000 system managers at my two-day system performance optimization workshops (performing a cursory analysis on each of their systems), and supporting hundreds more on an on-going basis with my team at Lund Performance Solutions. Still, I am very much open to having the following opinions challenged and adjusted, since we all have limitations on our understanding of this most precarious science.

In the discussion that follows, I cover some of the more reliable and easy to understand pulse points--the measuring sticks of performance management. This data is the building block of the rules-of-thumb used by performance specialists. I use sample output from the program SOS/3000 Performance Advisor for my discussion.

CPU Pulse Points

The CPU - Too Much Payload or Not Enough Horsepower?

One of the first places I turn to when faced with a performance problem is the CPU global activity. Refer now to Figure 6.2 (SOS/3000 screen). This figure represents data from an MPE/iX system. Zero in on the top left CPU section. This table of data shows the time the CPU was busy and non-busy, accordingly; note the average values in brackets. Remember, earlier I said that consistent (average) indication over a period of time is the important issue. The two left-hand columns show the main areas that indicate the CPU's activity.

The Total Busy statistic is a summary of all the time the CPU spent being productive and doing housekeeping (overhead) work. Though it is desirable to keep the CPU busy nearly 100 percent of the time, to avoid saturation, it is important that interactive processing is not 100 percent responsible.

One general rule-of-thumb I use to determine CPU saturation is the 85 percent rule (for MPE V systems use 75 percent). If high-priority processing approaches or exceeds either respective value, it is quite likely your system is approaching saturation. That is, an increase in terminal CPU demand (more transactions, users, or both) could cause a disproportionate increase in user response times. This would then cause batch activity to receive only "table scraps" of CPU time (if any at all).

```
SOS/3000 A.39a  (C) LPS FRI, FEB 28, 1992, 10:30 AM   E: 10:26:37  I: 05:00
-------------Global CPU Statistics----------------+----Global Misc Statistics----
             TOTAL BUSY:  100.0 [ 78 ]             #Ses 100 #Job 14 #Proc 601
AQ  4.8[ 1]   Memory  2.3[ 2]     CPU QL  12[ 2]   CM to NM Switches 635[855]/s
BQ  5.7[ 4]   Dispatch .1[ 0]     Launch/s 41[38]   NM to CM Switches  14[ 69]/s
CQ 80.2[12]   ICS/OH  6.3[ 4]     CPU CM%  8[28]    Transactions1178[60709] 235)
DQ   .6[55]   Pause    .<[ 7]     SAQ       200    Avg First Resp    .<[  .3]
EQ   .0[ 0]   Idle     .0[16]                      Avg Prompt Resp   .2[ 1.7]
```

Figure 6.2 - SOS/3000 Global Screen - A Busy 960

Humor is often an important part of a closely-knit family, but pushed to an extreme, it can become anything but funny. If it becomes the common way of communicating, conversations will become shallow. G. Smalley

But how do you tell if the total busy time is due to terminal or batch activity? Figure 6.2 represents a busy series 960 with 192 MB memory and 100 users. Notice in the top left column of Figure 6.2 the CPU activity breakdown by subqueue. The sum of the AQ, BQ, and CQ percentages shown there, along with other overhead values (memory, dispatcher, overhead), is the number to use for high-priority activity. In our example, the sum of these numbers is 99.4 percent, and the total amount of CPU being spent on interactive processing is 80.2 percent. In this case, especially if 99.4 percent is fairly consistent, we might safely conclude that this system is experiencing CPU saturation.

Keep in mind that the key for pulse-point diagnosis is *consistent* indication. It is quite normal for systems to experience spikes in performance indicators. We are not so concerned about a few, short bursts of CPU busy, but rather, what a key processing zone's profile might look like (say, 14:00 - 17:00). CPU saturation has a couple remedies. You could reduce demand for the CPU; an example of this might be to migrate to native mode for HPPA systems or perhaps re-write a fourth-generation language application in Cobol. Or, you could increase the supply by buying bigger/ better hardware.

Figure 6.3 illustrates the breakdown of CPU total busy into high- and low-priority components. It is sometimes deceiving to see a performance tool reporting a high total busy. One is led to think the CPU might be inadequate. But a closer look reveals that high-priority processing is only a small part; batch activity is consuming the most CPU.

```
SOS/3000 A.00 (C) LPS THU, FEB 14, 1991, 1:40 AM E: 00:05:19  I: 00:30
--------------Global CPU Statistics-----------------+------Global Misc Statistics----
              TOTAL BUSY:  100.0 [ 99 ]              |  #Ses 129 #Job 6 #Proc 612
AQ   .<[ 0]   Memory  5.0[ 5]    CPU QL    6[ 5]    |  CM to NM Switches   6[ 14]/s
BQ  3.1[ 5]   Dispatch .3[ 0]    Launch/s 79[80]    |  NM to CM Switches   6[  7]/s
CQ 29.2[25]   ICS/OH 14.7[14]    CPU CM%   1[ 1]    |  Transactions 538[5841](1135)
DQ   .0[ 0]   Pause    .<[ 1]    SAQ         13     |  Avg First Resp     .<[  .0]
EQ 47.7 50]   Idle     .0[ 0]                       |  Avg Prompt Resp    .1[  .1]
                                                    |
-----------------------------------------------------+-----------------------------
```

Figure 6.3 - High Priority vs. Low Priority CPU Busy

In this example (Series 960, 129 users, 256 MB memory) the system is 100 percent busy. Should the manager panic and upgrade? *No!* Why? To find out, first calculate hi-priority busy (we'll assume batch is less important than sessions).

AQ + BQ + CQ + Mem + Disp + ICS = 52.4

Batch activity is consuming nearly 50 percent of the system. There is no pause or idle time. Consequently, the 52.4 percent hi-pri busy does not indicate a CPU bottleneck (especially since the CPU QL is 6-- see the QL discussion later in this chapter).

Keep in mind the CPU can be compared to an automobile engine with a finite amount of horsepower. Thus, at any point in time, the CPU could be spending a little or a lot of its horsepower units. As stated in Chapter 5 it is common to think of this expenditure in three ways: CPU busy, CPU paused for disk I/O, and CPU idle.

The busy state of the CPU involves both useful, user-productive work and, unfortunately, housekeeping tasks. When housekeeping tasks rise above certain thresholds we begin to think that something is not quite right! And rightly so. Just think what it would be like if most of life's activities consisted of sweeping floors! The table below provides some caution zone indicators for various overhead CPU busy tasks:

	—Caution Zone—		
CPU	**MPE V**	**MPE/iX**	**Meaning/action**
Memory	7-10%	8-12%	Possible memory shortage (esp. iX).
Disk caching	20-30%	n/a	Is it worth it?
ICS management (term & disk activity)	15%	15%	Find offending device or process producing excessive interrupts. (If ADCCs, then the caution zone should be 25% .)
Global garbage collection	2-3%	n/a	Possible memory shortage (must have software caching off to see).
CPU queue length	>10	>10	Be sure that there is not a log jam of batch jobs artificially inflating this value.

Table 6.2 - CPU Pulse Points

An able man shows his spirit by gentle words and resolute actions; he is neither hot nor timid. Chesterfield

You can see from these values that a significant part of your CPU's horsepower could be spent on indirectly productive activity. If your system consistently exceeds any of these values, you may want to delve further into the problem to see why you might be spending so much of your system's power yet not seeing the productive throughput that you might like to (or were told that you would!).

The numbers referenced above reflect what might be referred to as a caution zone (yellow) busy indicators. "Good" CPU busy is time spent performing what we might consider useful processing such as calculating a dividend, searching a TurboImage buffer for the Jones payment record, etc. If the busy time you consider high-priority consistently goes over 75 percent for MPE V systems or 85 percent for MPE/iX, you could be saturating the CPU resource. Response times could suffer due to a paucity of CPU.

High-priority processing is generally thought of as activity (be it terminal or batch) that you would not be happy with if completion times grew worse than current levels. For example, if you could not sustain an increase of say, 10 to 15 percent in the run time for nightly batch processing, then you might consider this to be high-priority processing. If you add to this load, your batch "window" would be exceeded.

Increasing throughput for the CPU resource (or any other for that matter) involves doing one or more of the following:

- **Decrease the demand** - Use the CPU resource more efficiently (e.g., write programs that accomplish the same task with less CPU time).

- **Distribute the supply** - Allocate the CPU resource amongst processes via a more intelligent, priority-based scheme (e.g., redefine the "E" subqueue to live in the "C" subqueue and run very high-priority batch jobs there).

- **Distribute the demand** - Spread demand for the CPU resource over a longer period of time (e.g., run as much batch as possible during user lunch and break times).

- **Increase the supply** - Obtain more CPU resource horsepower (e.g., upgrade your Series 70 to a Series 967).

Another indicator of CPU shortage is the CPU queue length. This number tells how many customers are standing in line, waiting to be serviced by the CPU. The reasoning here is that if at any time there are more than a certain number of processes waiting to use the CPU, it could be over-worked. This is what happens at most fast food restaurants at 12:00 noon! If

you see fifteen people standing in line, it's a good indication that the cook is not able to keep up with the demand. A faster cook or multiple cooks might solve the problem. A consistent CPU queue length greater than seven to 10 on either MPE V or MPE/iX systems points to a yellow condition. This statistic, coupled with the CPU busy numbers, can help determine whether you are bottlenecked on the CPU.

Although not 100 percent definitive by itself, the CPU QL indicator (see Figure 6.2) can be a supportive sign of CPU trouble. We know, for example, that if the CPU busy indicator (high- or low-priority) is large, but the QL is very small (say less than three or four), it could be that the only problem is that a couple of interactive or batch jobs are running. This doesn't necessarily mean you're out of gas, especially if the jobs are completing within a reasonable time frame. But if those jobs are taking too long, you might need to upgrade the processor, or determine if there is some other resource causing the bottleneck.

If, however, the QL is high and the CPU busy is high, this is quite probably indicative of a CPU bottleneck. I have found in most instances where the QL is consistently greater than 10 or so, along with a swelling CPU busy time, the system is running at overload.

An exception to the queue-length rule is worth noting. If there are, say, 20 batch jobs running, the queue length will be 20. This does not necessarily mean that the system is overloaded. But it would indicate saturation *if* you had to have all those jobs running. If, however, an operator inadvertently streamed these jobs, you would be looking at an artificially puffed-up queue-length number.

For MPE/iX systems, it is also good to look at the percent of the CPU spent handling compatibility-mode activity. This may help you to determine whether it would be worthwhile performing the full "second migration." This refers to recompiling your programs with MPE/iX native compilers to take full advantage of the Hewlett-Packard Precision Architecture. If a good portion of the CPU busy (greater than 30 percent or so) is being "burned" in compatibility mode, you will save some CPU (maybe a lot) by enduring the migration to native mode. You will also want to refer to Appendix B, which contains two pulse point charts that will help you to diagnose resource bottlenecks.

He is a governor that governs his passions, and he a servant that serves them. Benjamin Franklin

Process Preemption as an Indication of CPU Adequacy

Another important factor in determining CPU adequacy is process preemption. This has to do with how often processes have the CPU taken by other processes. With the right metrics at hand, it is possible to break down the percentage of a process' response time (or batch completion time) into various wait states. If a large percentage of the time a process cannot continue to handle user requests is because the CPU was stolen by other processes, this can be a strong indicator of CPU shortage. For an excellent example of this, see the CPU Bottleneck Case Study at the end of this chapter.

In any case, the CPU is the heart of the system. HP has, with some exceptions, shifted most resource bottlenecks to the CPU for MPE/iX systems. For Classics (and some MPE/iX systems), it is still possible to have plenty of CPU, but be short on something else. I cover these arenas next.

Disk I/O Pulse Points - From Hyperspace to Snail's Pace

I will say at the outset that, with few exceptions, Classic systems tend to bottleneck on disk I/O while MPE/iX systems tend to be short on CPU. While performing numerous consulting projects, I have seen far fewer I/O bottleneck situations with MPE/iX systems than with MPE V systems. This is due primarily to the "turbo-caching" nature of HPPA; disk I/O requests are generally satisfied in main memory a high percentage of the time. While disk caching does pretty well on MPE V, with few exceptions, I/O elimination tends to be awesome on MPE/iX.

One strong indicator of I/O bottlenecking is the amount of time the CPU pauses for disk. This indicator tells how much time the CPU has to wait because disk I/O data was not available in memory. This value, if consistently above 10 percent or so can indicate a disk I/O "choke-point" on either architecture.

However, you may have an I/O problem with little or no CPU pause- for-disk showing. Let me explain. Let's say the interactive user's activity is showing a 20 percent CPU pause for disk condition. If you were to stream a batch job that would attack disk files other than those of the user's processes, the pause-for-disk condition would probably evaporate. Why is this? Does this mean there is any less of an I/O bottleneck condition? No. It's just that the batch job is able to utilize spare CPU time that was previously sitting idle. Other fundamental indications of I/O bottlenecking

would still be there. So, in any case, you shouldn't think that just because there is no CPU pause for disk, there is no I/O problem. Take a good look at the disk queue length, I/O rates, etc., which I explain next. Table 6.3 provides a list of important I/O pulse points.

-- Caution Zone --

Disk I/O Indicator	Classic	MPE/iX	Notes
CPU Pause for disk	10%	10%	I/O or memory bottleneck
I/O rates	20/sec	15/sec	I/O bottleneck
Avg disk queue length	>1	>1	Consistent Indication!
Read Hit%	n/a	85%	

Table 6.3 - Disk I/O Pulse Points

A couple of other numbers to consider are the disk queue lengths and I/O rates. The queue length on a disk drive is similar to that of the CPU queue length described above. An average queue length of one or greater is generally not acceptable. Disk I/O rates should be less than 20 or 30 per second (per drive) unless the drive has controller caching, in which case the value will often be higher. On MPE/iX and Classic systems with caching, disk I/O rates should be quite low (under 10 per second per drive). As rates go up, one or more of the following may be true:

1. Not enough memory exists to perform caching (MPE V) or pre-fetching (MPE/iX).

2. Data locality is poor. That is, data being retrieved is scattered all over a disk or within a particular file.

3. The read-to-write ratio is low. Too many writes can hurt an HP 3000. This is true even for MPE/iX. With pre-fetch and caching mechanisms, some reads can be eliminated. Writes on MPE V systems can, at times, be delayed but not entirely eliminated. On MPE/iX systems, the transaction manager can actually combine and therefore eliminate write I/Os as a function of gathered writes. And it generally does this very well. A 3:1 or greater read-to-write ratio is generally desirable.

The Wise and Brave dares own that he was wrong. Benjamin Franklin

It has been found recently that excessive writes can virtually cripple even a hefty HP 3000 Series 960. It is good advice to monitor disk I/O indicators over time and not get too excited if they spike temporarily. If disk I/O bottlenecking is indicated, then you might consider one or more of the following courses of action:

1. Check file structures for inefficiency. We often see TurboIMAGE database design or housekeeping problems on classic systems as being culprits. Detail and Master sets ought to be attended to or else you may be a victim of the second law of thermodynamics (left unmanaged, they will get worse)!

2. Check for I/O balancing across disk devices. Use FILERPT for this; you'll find it in the contributed library.

3. Be sure disk caching is on and efficient (classic only). Much has been written on caching. You may want to reference the Managing Performance Column in the HP Chronicle, for September through November of 1990 for some caching performance tips.

4. Do anything possible to improve data locality. Item 1 above will improve data locality within data bases. A couple of ideas are:

- Archive history. The more data you have, the more your applications may have to search through.
- Examine users' data entry/inquiry habits and see if, from an application standpoint, you can help minimize any data re trieval randomness.
- Utilize chain re-packing features in tools such as ADAGER, DBGeneral, and FLEXIBASE

The price of greatness is responsibility. Winston Churchill

MPE/iX Read Hit Percentage

An important indicator of MPE/iX health in particular is the Read Hit%. This metric lets you know how effectively disk I/Os are being eliminated by being satisfied in main memory. (It could be considered a *memory* pulse point just as well as a disk I/O pulse point.)

Essentially, this value says that in "x" percent of the time, I/O requests were satisfied without having to make a trip to the disk drives. Table 6.3 shows the caution zone for this indicator. Appendix B has a more complete list of the ranges of this value.

It is interesting to note the fairly common correlation of this value with the CPU pause-for-disk number. This stands to reason; the CPU is waiting for I/Os to complete since the I/O requests were not satisfied in main memory. Thus the Read Hit% is lower.

The Read Hit% is one of top 10 indicators for MPE/iX health.

MPE V Disk Caching Efficiency

Much has been written and debated over MPE V disk caching. I don't wish to re-hash all that has been said regarding caching. But I'd like to provide some pulse point indications so you can tell if disk caching is helping or hurting you.

I covered software caching from a management perspective, in Taming The HP 3000 Volume I (see Appendix C).

Figure 6.4 depicts a typical output from the SHOWCACHE command. While this table has a wealth of information, one of the most valuable indicators is: User I/Os Eliminated. This indicator is the total average number of read I/Os eliminated due to software caching's efficiency. Healthy ranges for this value are:

Green - >55%
Yellow - 45-55%
Red - <45%

DISK LDEV	CACHE REQUESTS	READ HIT%	WRITE HIT%	READ%	PROCESS STOPS	K-BYTES	%OF MEMORY	CACHE DOMAINS
1	3844694	76	66	91	973209	641	4	174
2	6145772	62	94	67	2044746	1235	8	407
3	4153906	95	89	80	242571	253	1	75
4	2220354	37	78	81	1185509	439	3	135
15	4896996	65	94	73	1433997	2246	15	616
18	3680174	71	88	75	890066	221	1	63
5	22939	39	95	88	12427	625	4	222
16	6967	40	80	93	3968	115	0	35
17	32405	34	93	92	19921	158	1	46
Total	25004207	70	90	76	6806414	5931	41	1773

53% of user I/Os eliminated.
Data overhead is 466K bytes.
Sequential fetch quantum is 96 sectors.
Random fetch quantum is 16 sectors.
Block on Write = YES

Figure 6.4 - MPE V SHOWCACHE Display

A dripping hot water faucet wastes an average of 40 kilowatt-hours of electricity per month - the equivalent of running a color TV 8 hours a day for 31 days.

If you have an undesirable total value, it is important to determine which disk(s) is experiencing the worst read I/O elimination. The same ranges above apply for individual drives. Simply multiply the READ% by the READ HIT%. These values for Figure 6.4 are as follows:

LDEV	% of I/Os Eliminated	Notes
1	69	
2	42	Turn cache off
3	76	
4	30	Exceptionally poor;turn cache off?
15	47	
18	53	
5	34	Turn cache off
16	38	Turn cache off
17	31	Exceptionally poor;turn cache off?

Table 6.4- MPE V SHOWCACHE Individual Disk Performance

In the above example, many of the drives exhibited poor read elimination. This was most likely due to poor data locality and/or insufficient main memory. In these instances, it is better to simply turn caching off and return CPU and main memory back to the system for useful processing than to waste these resources on inefficiently cached drives.

Some performance tools also report MPE V caching statistics and save you from having to perform the math yourself.

Main Memory Pulse Points - "Pa, We Need More Cupboard Space!"

Adequate main memory is necessary in order for the CPU to process data, much like adequate desk space is necessary for you to perform productive work in your office. I have found that if classic systems run out of memory, response times may suffer somewhat; they will become gradually worse. However, on MPE/iX systems, when the system runs out of memory, performance goes from bad to worse much more rapidly.

Some of the signs of memory shortage can be excessive CPU time managing memory (MPE/iX primarily), a high page-fault rate (MPE/iX), a fast clock rate, and a high swaps-per-launch ratio (primarily Classic). Values greater than those listed below in Table 6.5 are progressively worse signs except where noted.

-- Caution Zone --

Memory Indicator	MPE V	MPE/iX	Meaning/action
CPU busy on memory Management	7-10%	8-12%	Possible memory shortage (esp. XL).
Swaps/Launch	25	.6	Possible memory shortage or individual application memory misuse.
Page-fault rate	n/a	15/sec	Possible memory shortage.
Memory clock rate	<2000 ms/cycle	>25/hour ms/cycle	Possible memory shortage; Some variability if software caching is enabled.
Global garbage collection	2-3%	n/a	Possible memory shortage (must have software caching off to see).

Table 6.5 - Memory Pulse Points

These memory numbers tell much of the story. The key for these as well as other indicators is whether they are consistently at or above the stated range. The bottom line for memory is you need enough! If you have any way to procure more memory than what you think you need, get more! The fabulous 9xx systems perform very well with ample memory, but are virtually crippled by a memory shortage.

MPE/iX Mode Switch Pulse Points

One source of resource drain (CPU primarily) on MPE/iX systems is due to mode-switch activity. In order to accommodate non-native mode program execution, MPE/iX must translate code into its "native tongue." This costs some CPU time. Consequently, there are three metrics you'll want to monitor occasionally. They are:

- **CM to NM switch rate:** the number of compatibility mode to native mode switches per second. In comparison with NM to CM below, this rate can be quite high without a substantial cost.

- **NM to CM switch rate**: the number of native mode to compatibility mode switches per second. These incur much more CPU than do CM to NM switches.

- **CPU CM%**: the amount of CPU spent handling compatibility mode services.

How much is too much for these activities? A pretty good rule-of-thumb is as follows.

CM=>NM: greater than 200 per second for small systems and 500 per second for large ones.

NM=>CM: greater than 25 on small systems and greater than 75 on large ones.

CPU CM%: anything over 50 percent should result in some serious investigation, though this metric is almost purely subjective, and you may not have any control over this at all. You may simply be at the mercy of your vendor.

You'll want to refer to the Pulse Points Charts in Appendix B for a complete look at pulse point values. These charts are a handy reference for performance indications on your system.

There are many other pulse points to monitor, but the ones I have outlined in this chapter are the most important in terms of solving performance problems. It would be best for you to utilize your performance monitoring tool and observe these values, even become an expert at knowing how they stand on your system. Graph various indicators to see their interrelationships.

Because of the downsizing of our country's economy, I have found many of our clients need to extend the life of their current HP 3000s. More than ever before it behooves system managers to implement a thorough monitoring/managing game plan—one that delivers relevant information for management decisions and deals effectively with crises, etc. Cost effectiveness is also important. Consequently, at the heart of performance management is a monitoring tool that allows you to effectively cover the angles, bases and pulse points I have discussed thus far.

A well-rounded performance management game plan will increase the mileage of any computer system by forestalling costly capital expenditures for new equipment, and more accurately foretelling upgrade needs. Understanding the critical areas of resource utilization as defined above will allow you to more effectively meet this goal.

Performance Bottlenecks

What Is A Bottleneck?

A bottleneck, by definition, is a choking off or a restriction in the flow of a transaction's progress. Computer performance bottlenecks can be likened to what happens all too frequently on an automobile expressway (Figure 6.5). Due to an accident or road repair work, one or more lanes are eliminated. The number of vehicles able to pass through that "choke-point" per unit of time is decreased. Of course this slowdown is noticed only if there is sufficient demand. As the demand increases for lane space, more crowding takes place and net traffic throughput drops.

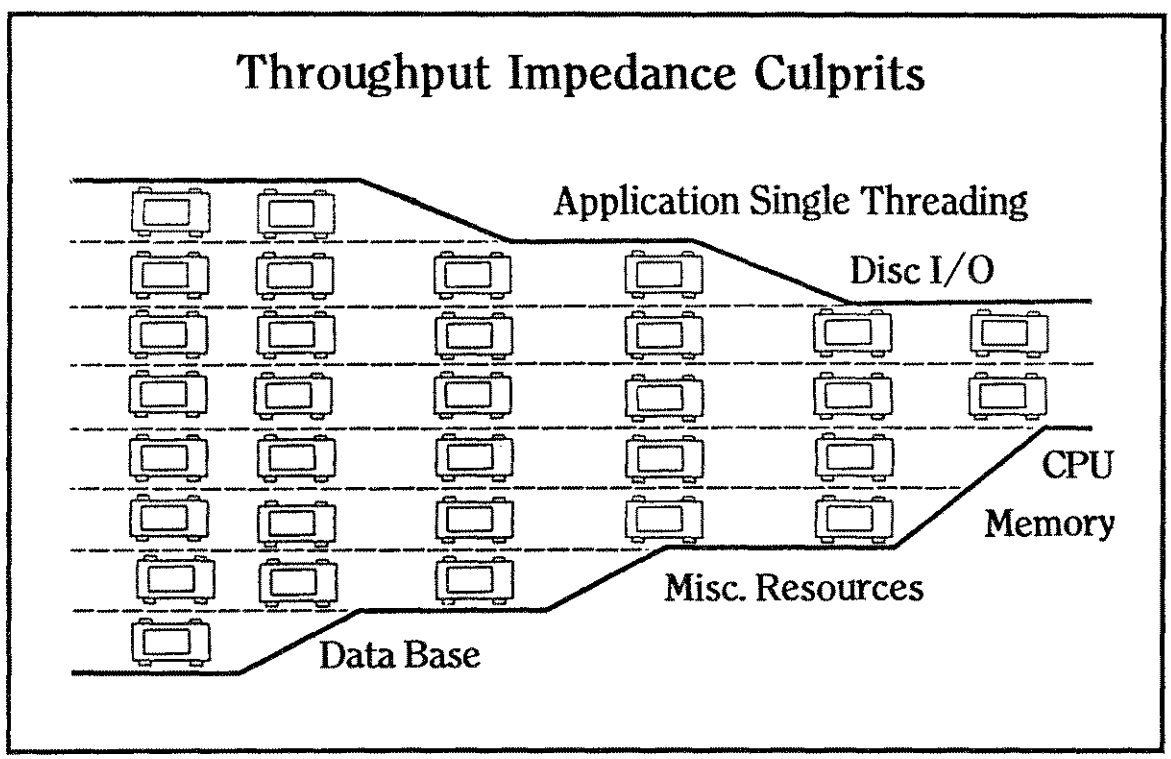


Figure 6.5 - Transaction Bottleneck Illustration

Simply put, when a necessary system resource becomes short in supply, batch and user requests must queue or wait until that resource becomes available. Keep in mind that a single CPU within an HP 3000 is a single-threaded resource; only one request can be processed at one time, albeit very fast.

However, on multi-processor systems such as the 980-200, multiple tasks can be performed at one time. One advertisement illustrating data base single-threading featured a single-horse merry-go-round with one

A two-year-old's temper tantrum is a mark of the child's age. A twenty-five-year-old's tantrum is a mark of the persons' upbringing.

child riding the horse and others standing in a long line. A multi-CPU system would have two "horses" capable of processing more children (transactions). Indeed, the CPU on a multi-programming computer (which has a single CPU, in contrast with multi-processing multi-CPUs) makes customers wait in line for servicing. If the demand for CPU processing becomes too great, the time necessary to service transactions becomes excessive and poor terminal response and/or batch completion times result.

If only one bottleneck exists on a computer, the solution may be easy. Unfortunately, when one performance problem is solved, another often surfaces soon thereafter.

Shifting bottlenecks can be compared to sitting on an old spring mattress. As you apply pressure to push down one unruly performance "spring," another one jumps up at another location. For this reason, it is essential to understand that resources within a computer system do not exist in a vacuum. There is an interdependency between each that sometimes causes them to mask one another. I'll use one example from the dark ages (not bleak, just primitive!) in the history of the HP 3000.

Ancient HP 3000 History

When the HP 3000 Series 40 and 44 were introduced, they provided many folks with a much needed performance increase. More CPU horsepower, a seemingly never-ending supply of main memory (compared to the Series III, 30, 33), and an increase in the number of device connections gave a performance boost to many HP 3000 shops.

However, as more programs were run (concurrently), and as the number of terminal devices swelled, the increased disk I/O traffic became a bottleneck for many of these machines. CPU saturation typically wasn't the primary problem, nor was memory shortage. A common problem was difficulty in squeezing more disk accesses through per unit time once a disk device began to reach a certain saturation point. The increase in CPU and memory capacity in the Series 40 and 44 provided better performance, but eventually the electro-mechanical interface (disk drives) between data and the ultimate user became a choke point, which caused increasingly poorer response times.

The previous memory bottleneck on many early HP 3000s, then relieved by the Series 40 and 44, merely allowed disk I/O to surface as a new bottleneck! Push down the memory spring and up comes the I/O spring. Disk I/O bottlenecking became an acute problem for many. Hewlett-Packard recognized this I/O problem and came to the rescue with a product called MPE Disk Caching.

The concept of a caching mechanism is actually quite simple. As data is requested from disk, the disk device is instructed to retrieve considerably more data than is necessary to satisfy your current need. These "pre-fetched" records are then stored in a memory buffer with the hope that one or more will be needed by other users. If a disk request is satisfied in this newly created memory cache, a physical disk access is eliminated. The addition of this software mechanism along with more main memory (don't forget the nameplate!) turned a Series 40 into a 42 and a 44 into a 48.

I remember when software caching was first installed at a number of sites. At the time, I was a field systems engineer for Hewlett-Packard. If the criteria for caching was met (ample CPU, memory, etc.), the installation would all but make the installing engineer a hero. Some customers likened the effect of disk caching to "hyperspace" in the movie "Star Wars!" Response times decreased for many (but not all) sites.

This benefit did not come cheaply, however. Disk caching consumed hordes of memory and CPU horsepower in order to store and manage the massive influx of data. Systems with an abundance of CPU and memory often found themselves with a new bottleneck or two: CPU and memory! Solve the I/O problem and perhaps a CPU problem surfaces. This story out of the HP 3000 history book illustrates how bottlenecks shift.

HP 3000 Bottleneck Evolution

Let's turn our attention to see how HP 3000 bottlenecks have evolved in recent years.

With the introduction of the HPPA systems, Hewlett-Packard has attempted to create the "mother of all bottlenecks." But why would they want to create a bottleneck with the introduction of a new super-whammy computer? Here's why. Every computer system has at least one bottleneck; most have quite a few. One of HP's design goals for the Series 900 was to shift as many I/O, memory, and miscellaneous constraints to become CPU dependent. If they could do this, then the vast majority of customer bottleneck problems would be solved very simply: with a CPU

41% of the 581 plants and animals on the Department of Interior's list of threatened or endangered species are stable or increasing in numbers.

upgrade. This would be good for profits and provide a straightforward solution to customer growth constraints.

One of HP's goals was to create a one-bottleneck system. That is, to have a computer that was limited only by the speed of the CPU. With the advent of HPPA systems, HP has nearly achieved this objective because, in many cases, it really is a matter of upgrading the CPU to improve throughput.

With the introduction of the 900 series, my experience has been that the CPU is often made out to be the bottleneck culprit. However, my files are full of case studies and consulting projects in which upgrading the CPU would do very little, if anything, to solve the user's performance problem.

Situations that involve inadequate memory, poor data locality, (either micro-- within a data structure; or macro-- data structures across disks), excessive use of character-mode terminal activity, and improper use of process handling features are all examples of when a CPU upgrade would not solve response time problems.

So, a design goal to gain maximum throughput on a 900 series HP 3000 must include optimal utilization of the CPU. Use less, use it more efficiently, and spread the demand over more time, or as a last resort, you'll have to bite the bullet and buy more! See Chapters 12 and 13 for ways to utilize the least amount of CPU on an MPE/iX system.

But how do you identify a performance bottleneck?

Will the Most Prominent Bottleneck Please Stand Up?

To identify bottlenecks you'll want to be equipped with the pulse points information discussed earlier (see Appendix B). Let's use a case study that illustrates one kind of bottleneck and how it is best identified.

Figure 6.6 illustrates how we would like a performance analysis to be broken down. This diagram simply illustrates the fact that poor batch or user responses are due to the time it takes to perform various activities on behalf of user transactions. Although we cannot obtain the kind of detail shown in Figure 6.6, we can derive something almost as helpful.

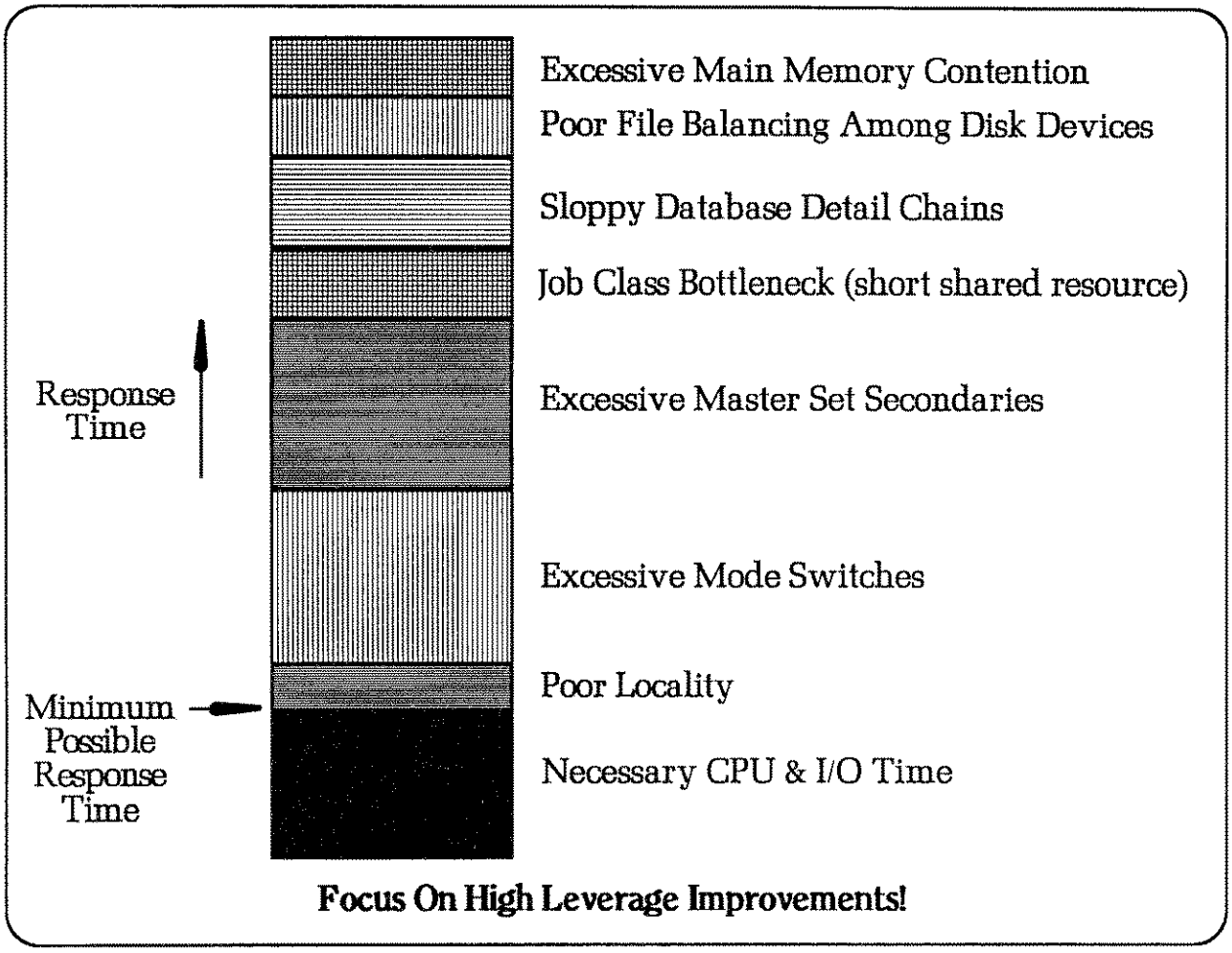


Figure 6.6 - Performance Culprits

CPU Bottleneck Case Study

Figure 6.7 shows data for a real user situation (series 960, 169 users, 144 MB memory) in which a company called me to perform a performance evaluation. They thought a processor upgrade would help. Remember, I said that the majority of the bottlenecks on MPE/iX systems involve CPU shortage.

Atheists put on a false courage in the midst of their darkness and misapprehensions, like children who, when they fear to go in the dark, will sing or whistle to keep up their courage. Alexander Pope

Let's take a look at some of the pulse points on this system. Notice the CPU Total Busy number at the top of the figure; look also at the CPU queue length (CPU QL=13). It's interesting to note that this is almost exclusively an online application (no batch CQ vs. DQ) with only 3.8 percent CPU pause-for-disk I/O time. Remember, pausing for disk means that there is some disk action impeding transaction progress.

```
SOS/3000 A.33 (C)   LPS FRI,   FEB 28, 1992,  10:30 AM   E: 10:26:37   I: 05:00
--------------Global CPU Statistics--------------------Global Misc Statistics----
              TOTAL BUSY:  96.2 [ 99 ]                #Ses 169 #Job 6 #Proc 853
AQ  .<[ 1]    Memory  7.6[ 6]    CPU QL  13[19]       CM to NM Switches 28[ 54]/s
BQ  4.3[ 6]   Dispatch .2[ 0]    Launch/s 70[65]      NM to CM Switches 10[ 21]/s
CQ 65.2[74]   ICS/OH 10.9[10]    CPU CM%  3[ 5]       Transactions 604[90580 (594)
DQ  8.0[ 1]   Pause   3.8[ 1]    SAQ       300       Avg First Resp    .<[  .0]
EQ   .0[ 0]   Idle     .0[ 0]                         Avg Prompt Resp   .6[ 1.2]
-------------------------------Process Information--------------------------------
PIN J/S#  Session/User Name       Cmd/Program  CPU%  QPri  #Rd  #Wr  LDV  #Tr  PRes
842 S1208 KAC3157,CLOCLK.CONTRACT    REACTOR     6.9  CS158   58   29  540   15   .9
896 S1210 JMB3155,CLOCLK.CONTRACT    REACTOR     6.1  CS152   17    6  537   17  1.9
878 S1217 JSK3214,TSR,CONTRACT       REACTOR     4.5  CS152   20   15  142    5  4.7
63  S1214 DAB3201,TSR.CONTRACT       REACTOR     3.8  CS152    8    5  122   19   .9
70  S1213 RLR3104,CLOCLK.CONTRACT    REACTOR     3.4  CS152   22   15  516   18   .3
435 S1212 MKM3160,RELCLK.CONTRACT    REACTOR     2.1  CS152   10    0  651   15  1.2
260 S1208 AAE3151,CLOCLK.CONTRACT    REACTOR     1.8  CS152   52   17  345    8  1.0
752 S1226 SKD3212,TSR.CONTRACT       REACTOR     1.2  CS152   27    4  125    7  2.8
```

Figure 6.7 - MPE/iX CPU Bottleneck Case Data

Let's take a look at some of the pulse points on this system. Notice the CPU Total Busy number at the top of the figure; look also at the CPU queue length (CPU QL=13). It's interesting to note that this is almost exclusively an online application (no batch CQ vs. DQ) with only 3.8 percent CPU pause-for-disk I/O time. Remember, pausing for disk means that there is some disk action impeding transaction progress.

Notice the absence of CPU idle time. Keep in mind that if no non-trivial batch jobs are running and if there is no CPU idle time, then we can say the system is really out of gas in terms of the CPU resource. There were many other processes not shown, but no significant batch activity. If this application were on a Classic system, there is a good chance that there would be significantly more I/O bottlenecking-- CPU pause for disk, long disk queue lengths, etc. This is a classroom case illustrating HP's bottlenecking evolution strategy: transforming as many bottlenecks as possible to become CPU bound. A CPU bound problem can be addressed in the ways outlined in earlier discussions.

Another very revealing aspect of their problem is seen in Figure 6.8. This figure represents the wait states (brick walls) that are hindering user processes (transactions) on this system.

Notice that a majority of the time processes are stalled (waiting) due to an event called "Preempt." Preemption occurs when a process steals the CPU away from another process. Thus, in our example, many of the processes have the majority of their response times increased due to preemption activity.

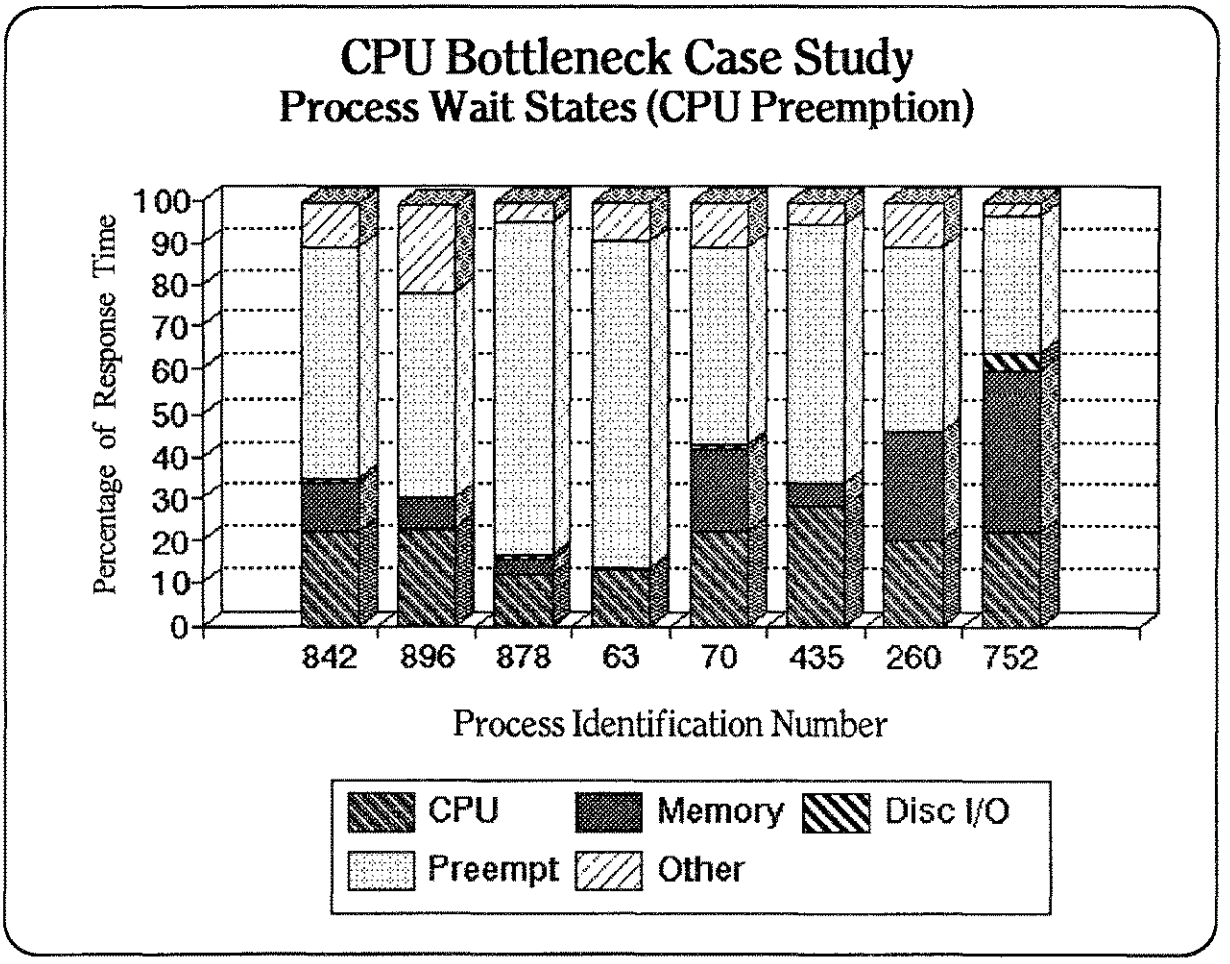


Figure 6.8 - Transaction Throughput "Brick Walls" - Wait States

But what does process preemption mean in simple terms? An illustration will help. Let's say you are at a bank being served by a teller. In the middle of your transaction, the teller allows another person to break in and be served, either in person or via a phone call. You have just been preempted. Then, when the other person's transaction is finished,

The average cost of the average home garden is $19.00; the average yield is worth $325.00. The average home garden is 595 square feet.

the teller resumes dealing with you. I don't know about you, but I am almost always a bit perturbed when this happens. Some preemption is normal within a time-share computer system. If you were preempted often while being served, would you get the feeling that the bank could use more tellers? Probably. And if the situation didn't change, you would most likely find another bank. Our case study is no different.

In our example, these users' response times are impacted heavily due to preemption activity. But how much? Look at process number 878 in Figure 6.8 on the previous page. Its average response time for five terminal transactions (character mode in this case), was 4.7 seconds (not shown). If preemption activity occurs 78 percent of the time, then the component of the 4.7 seconds due to this activity is 3.67 seconds (4.7 * .78). So, if we could solve the preemption problem and all other things stayed equal, that user's response time could go down to as low as 1 second per transaction.

In conclusion, this case study is a good example of what many people could (and do) experience on HPPA systems. This is one reason why I chose it for this discussion. If you have an HPPA system, it would be good for you to study this example and learn the above-discussed CPU metrics; there's a good chance they'll come in handy someday! There are other case studies later in this book to help you gain skill in diagnosing performance problems. Try looking at the output and graphs first before you read the text and see if you come to the same conclusion.

MPE/iX Memory Case Study

Figure 6.9 is a good example of an MPE/iX system (922, 32 MB memory, 16 users, various jobs) that is experiencing memory pressure.

```
SOS/3000 A.27 (C) LPS THU,APR 2, 1992, 1:53  PM     E: 10:13:41:36     I: 10:29
-------------Global CPU Statistics----------------+----Global Misc Statistics----
             TOTAL BUSY:  99.1 [ 41 ]              | #Ses 16  #Job 5  #Proc 145
AQ   .1[ 0]  Memory 14.5[ 4]    CPU QL   1[ 1]    | CM to NM Switches 131[ 55]/s
BQ  2.8[ 2]  Dispatch 5.8[2]    Launch/s 43[15]   | NM to CM Switches  61[ 19]/s
CQ 24.1[10]  ICS/OH 11.3[ 4]    CPU CM%  22[24]   | Transactions 376[18852] (36)
DQ 40.6[20]  Pause    .7[ 5]    SAQ        45     | Avg First Resp   3.4[ 1.2]
EQ   .0[ 0]  Idle     .1[53]                      | Avg Prompt Resp  5.8[ 6.0]
---------------------------------Global Memory Statistics-------------------------
Page Fault Rate 10[ 4]/s  Memory Cycles 15[382]     Overlay Rate  0[ 0]/s
Lbry Fault %    27[22]    Read Hit %    77[ 93]     Swap/Launch .51[ .43]
---------------------------------Global Disk Statistics---------------------------
<Ldv>Rt/IO%QL <1>20/70/2.05  <2> 6/ 19/.23  <3> 4/ 11/ .20
```

Figure 6.9 - Series 922 Memory Bottleneck

The data in Figure 6.9 was fairly typical of their peak processing periods. The key pulse points indicating memory shortage are:

- High CPU % busy on memory management (14.5 percent)
- Moderate page fault rate (10 per second)
- Very high number of memory cycles (15 in a 10-minute period = 90 per hour!)

The solution for this site was simply to procure more main memory.

Job Throughput Case Study

This case study involves an instance on an MPE/iX system in which critical daily batch activity was not completing on time. In fact, jobs that had taken minutes prior to some new applications being loaded were taking hours to complete. Figure 6.10 illustrates why a.m. jobs were virtually roadblocked from completing. Notice how the vast majority of each interval sample show preemption as the cause for the job's delay. Preemption, if you will remember from earlier discussions, occurs when other processes take the CPU away from the currently executing process. In this case, it was many online user processes that were hogging the CPU.

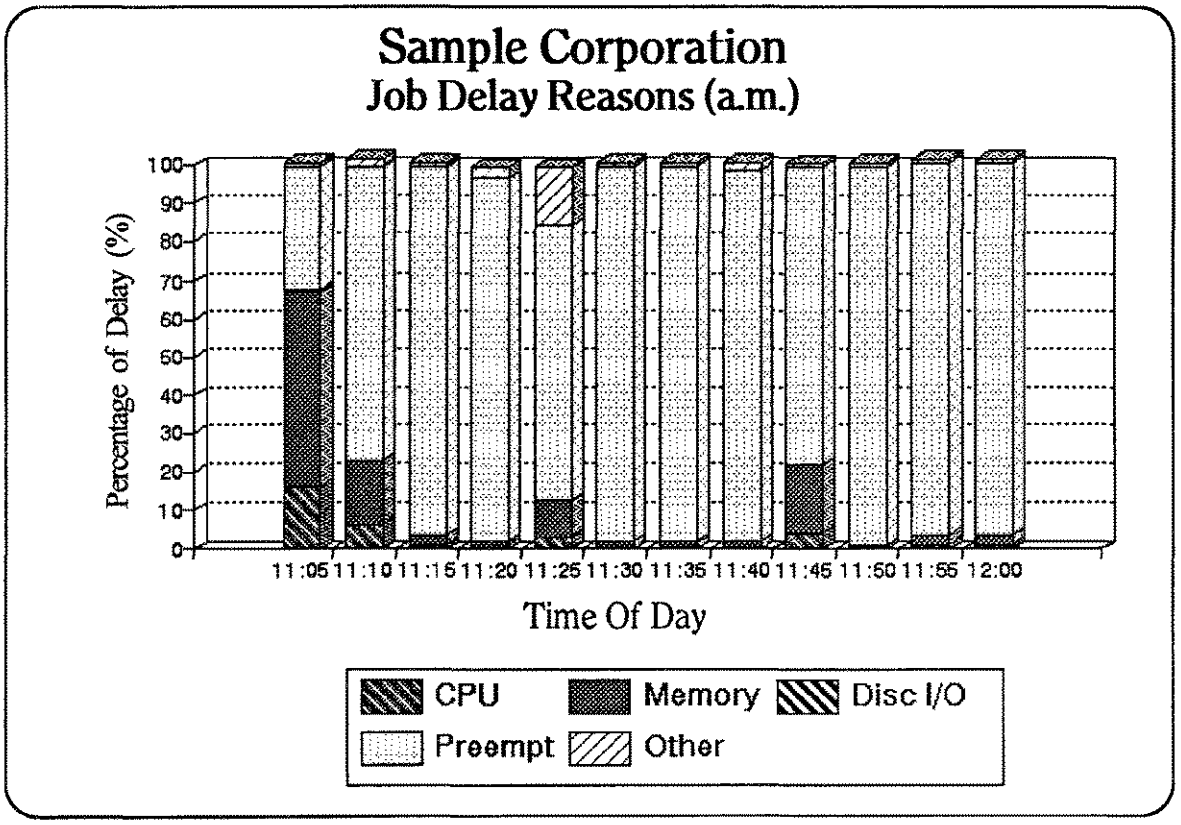


Figure 6.10 - Daily Job Preempted

Tell the American people never to lose their guns. As long as they keep their guns in their hands, whatever happened here will never happen there. A female student from Beijing, Red China, describing her parent's last words to her.

The solution for this situation was to isolate a new application that seemed to be using much more CPU than was expected. Ah! Another reason why you should profile applications in the design stage to identify their resource demands. The same is true when you are evaluating new vendor applications: Make the vendor provide you with a statement of CPU and disk I/O for typical online transactions and batch jobs. This data can then be put into a capacity modeling tool, which would pinpoint the net impact on your system when the application is fully implemented.

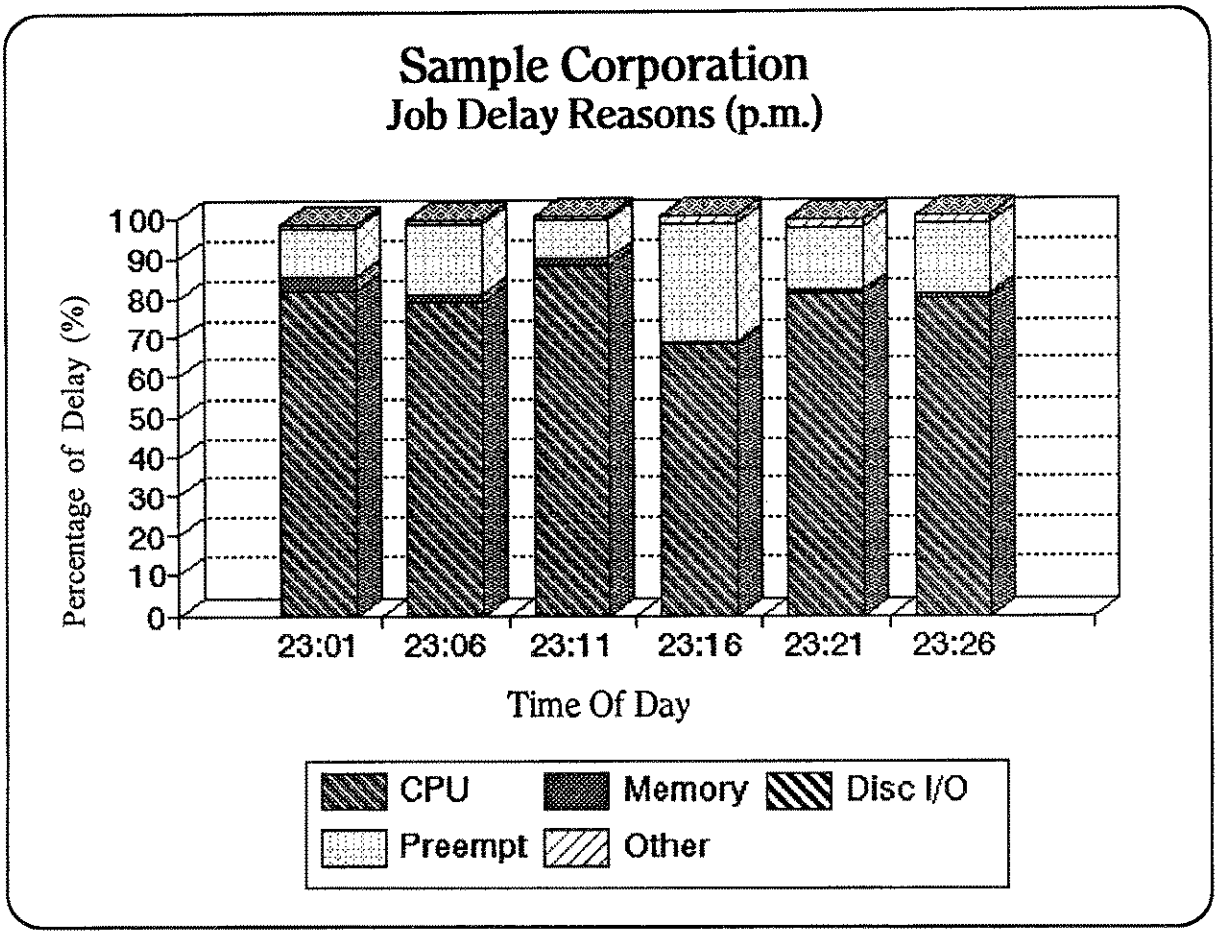


Figure 6.11 - Night Job Unhindered

Figure 6.11 illustrates what we wish all of our batch activity could experience: virtually unhindered use of the CPU. Batch jobs on this system completed very quickly at night!

In Shakespeare, Romeo says, "He jests at scars, that never felt a wound." A child who has never known difficulty will have little capacity to help someone who is hurting. G. Smalley

MPE/iX Locality Bottleneck Case Study

This customer had a very busy series 960 with more than 100 users involved in "heads down" data lookup and entry. Response times were not acceptable and there were occasional system hangs.

We wrestled with this problem for some time and explored many possible solutions. Notice the 40 percent CPU pause for disk in Figure 6.12! This is highly unusual for HPPA systems (which are not supposed to have I/O bottlenecks!). Most attempts to solve this problem were standard remedies for high pause-for-disk situations. Memory was added, database housekeeping was checked, applications were looked at, etc.

```
SOS/3000 A.00 (C) LPS  THU,FEB 14, 1991,  2:11 PM   E: 00:35:39   I: 00:30
-------------Global CPU Statistics---------------------Global Misc Statistics----
            TOTAL BUSY:  59.4 [ 85 ]                 #Ses 127  #Job 6  #Proc 599
AQ  .<[ 1]  Memory  2.4[ 4]    CPU QL   2[ 4]       CM to NM Switches  21[ 18]/s
BQ 3.1[ 4]  Dispatch  .1[0]    Launch/s 43[70]      NM to CM Switches  13[  8]/s
CQ25.4[27]  ICS/OH  6.5[12]    CPU CM%  7[ 2]       Transactions 395[36103] (771)
DQ  .0[ 0]  Pause  40.6[15]    SAQ       17         Avg First Resp   .<[  .0]
EQ21.8[36]  Idle     .0[ 0]                        Avg Prompt Resp   .1[ .1]
---------------------------------Global Memory Statistics--------------------------
Page Fault Rate  6[12]/s  Memory Cycles  0[ 9]     Overlay Rate  15[  24]/s
Lbry Fault %     1[ 2]                            Swap/Launch  .34[ .35]
---------------------------------Global Disk Statistics----------------------------
<Ldv>Rt/IO%QL <1> 1/ 4/ .00  <2> 0/ 2/ .07 <3>  0/ 2/ .06   <4> 7/23/ 3.20
<Ldv>Rt/IO%QL <5> 0/ 0/ .00  <9> 0/ 1/ .00 <11> 7/ 23 9.99  <12> 1/ 5/ .11
<ldv>Rt/IO%QL <13> 0/ 2/ .06 <14> 3/ 8/ .27 <15> 1/ 3/1.29  <16> 2/ 5/ .24
```

Figure 6.12 - Excessive CPU Pause for Disk

What ultimately tipped us off to the real problem was the Read Hit% value. Figure 6.13 shows this value over a typical day, correlated with the CPU pause-for-disk value.

The best thing one friend can do for another is to refrain from giving advise. McKenzie

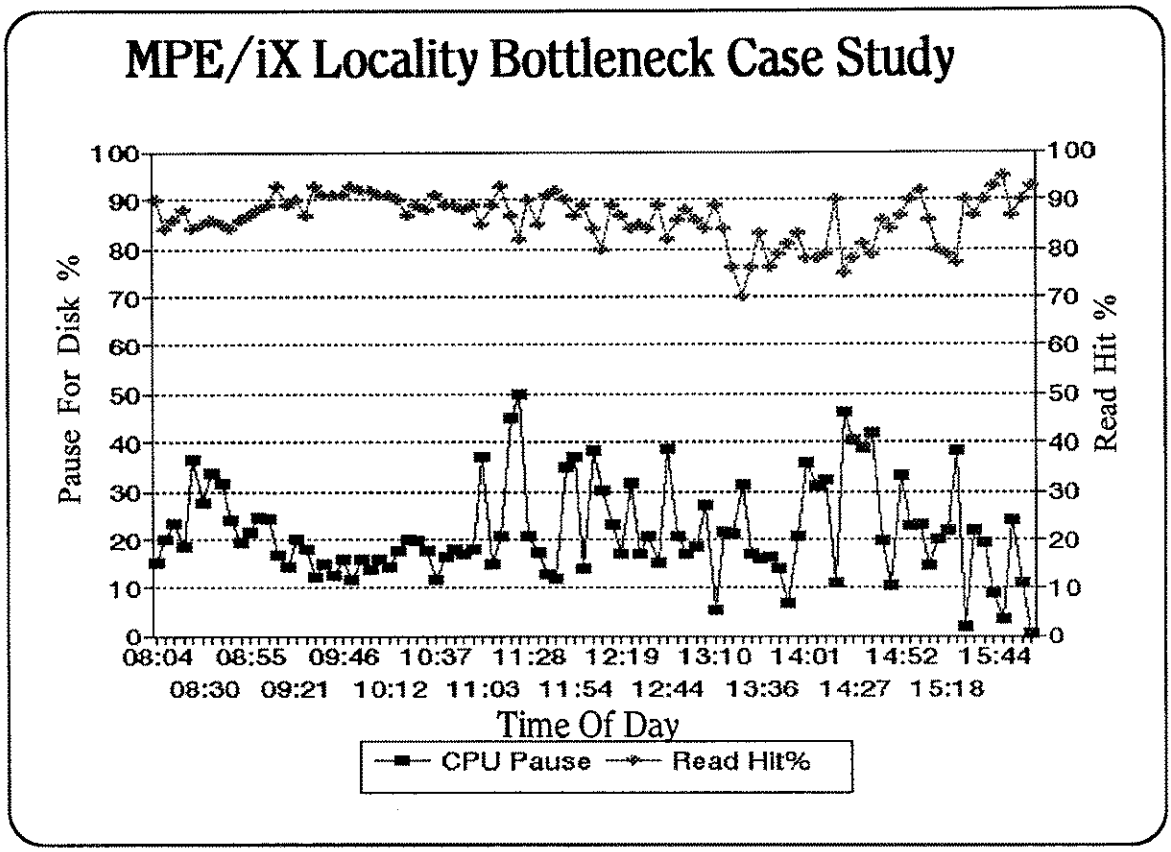


Figure 6.13 - Read Hit % Correlated to CPU Pause for Disk

Close inspection of Figure 6.13 reveals an inverse correlation between the two values for Read Hit% and CPU Pause-for-disk. The problem on this system was the data lookup habits of the users in relationship to the sample space of data possible. As it turned out they had over twenty different data bases, any of which were targets for user access. Consequently, there was very little opportunity for the system to allow multiple users to share data retrieved by MPE/iX's pre-fetch mechanism. Keep in mind that users attacking a single data base share a lot more than just the data itself. There are plenty of "shareables" within data structures, such as a TurboImage data base.

This system exhibited incredible random data access traits. Interestingly enough, adding more memory did not produce any perceivable benefit. I believe that this example is one in which you will either have enough memory (minimal amount) but have locality problems or you will put in as much RAM memory as you have disk space and will have no problems!

Conclusion

I hope these few case studies have been helpful for you to better identify performance problems. If you are even half-way serious about performance management, please consider digesting the pulse points discussion. I really am sincere when I say I wish someone had given me this information earlier in my HP 3000 career. If you have other pulse points that you find interesting and would like to discuss them (or anything else regarding performance) I would like to hear from you. Also, for future versions of this book, please let me know of any "juicy" case study situations you have run into.

? Questions/Discussion ?

1. How "thick" is the response time plot at your site? Have you determined which applications are character- versus block-mode? If so, how are you going to deal with response time reporting?

2. Do you feel that consistent response time is more important than "under-one-second" response time?

3. Which CPU indicators have you used for monitoring? After reading about CPU pulse points, which ones will you use now?

4. Have you considered the issue of locality on your system (Read Hit %)? What step can you take to improve it?

5. Which bottleneck(s) do you think are prominent on your system? If none, which do you think will be first?

Forgiveness is a funny thing. It warms the heart and cools the sting.

Money never made a man happy yet, nor will it. There is nothing in its nature to produce happiness. The more a man has, the more he wants. Instead of its filling a vacuum, it makes one. If it satisfies one want, it doubles and trebles that want another way. That was a true proverb of the wise man, rely upon it: "Better is little with the fear of the Lord, than great treasure, and trouble therewith."

Benjamin Franklin

Section Two

Yet More Ways to Monitor, Manage, and Maximize Performance on the "Classic" HP 3000

HP has made it clear that MPE V systems will be supported until there are no more customers to support. Many shops (and I do mean many) will stay with the tried-and-true Classic system for quite some time. Some folks are forced to stay on MPE V due to severe budget limitations, others stay because of application difficulties, etc. With this in mind, I thought it would be good to include material devoted to Classic "life extension." This section will include ideas for performance monitoring and management that have emerged since the writing of Taming The HP 3000 Volume I or were not included in that book. I have included a chapter on some classic tools that I did not cover in Volume I. And, there is also a chapter of classic case studies that will help you to monitor and manage your systems better. Just for the fun of it, I have included a chapter on tuning system tables for MPE V systems.

Be sure you take time to review Appendix C, which is essentially an outline of key ideas from Taming The HP 3000 Volume I.

Your users and management trying to get your attention with respect to poor system performance.

Chapter 7

Classic Performance Tools

In this chapter I discuss the tools for MPE V that are new or have significantly changed since the first volume of this book. It is my understanding that some of the tools I discussed in that volume are still available for MPE/iX. You may wish to refer to that volume to review some of their features. Vendor information is available in the Vendor Reference section in the Bibliography. The information here is accurate as of early 1992.

There have been no operating system improvements on MPE V to allow one to monitor performance. Most of the contributed library tools still work, except a few (like SURVEYOR — died at VMIT; a moment of silence...), which could crash your system. Many folks who have been using tools like SOO, OPT/3000, and SURVEYOR have been upgrading to some of the few commercial tools available. What follows is a list of such tools.

Although I have said it many times already, it is essential that you have adequate tools to cover the performance bases and angles. Don't forget the perils of flying blind.

Performance Monitoring Tools

SOS/3000 Performance Advisor

I originally conceived SOS/3000 Performance Advisor for my own needs as a consultant. I had used SURVEYOR for years to perform system performance analyses. Simple as it was, SURVEYOR had some excellent data. Then, when the VMIT release of MPE V hit the street, SURVEYOR began spewing "flying bytes" into main memory. Many sites reported all kinds of strange system failures. I noted nearly a dozen different system failure types--usually indicative of memory corruption. SOS/3000 Performance Advisor came about so that I could supply answers to my customers.

While I wish to speak without bias, I am the original designer and author of the tool. However, hundreds of my customers, many who have performed extensive comparisons with other tools, tell me that it is the most comprehensive, cost-effective performance tool available for the HP 3000.

Some of the design criteria I had in mind for this tool are as follows:

- It had to be extremely low in overhead. It was written in the "C" language, and am I glad it was. Many clients have brought to my attention that SOS/3000 has the lowest overhead of all the tools they compared. With one tool, CPU overhead was consistently 350 percent to 450 percent more than with SOS/3000. It also takes full advantage of the HP Architected Interface (AIF).
- It had to cover all the performance bases (see Chapter 2).
- It had to cover all the performance angles (see Chapter 3).
- It had to be easy to use (allowing some help in interpreting difficult technical data).
- It had to have enough depth for even a seasoned performance specialist to be comfortable with.

I believe that we have accomplished all of these goals, and more. Here are some of the features of SOS/3000:

- A global screen allows you to view global, process, and workload data and then use over a dozen detail screens to zero in on performance problems. Nearly every screen allows a graphical or tabular view of the data.
- Full logging capability and host graphics creation are standard; an infinite number of graphs may be user-defined and printed or exported to PC spreadsheets. Data is also compatible with Performance Gallery, a Windows PC-based 3D graphical presentation package (see below).
- Performance data can be gathered, extracted and presented on the host in a fully automated fashion. Historical performance information may then be reviewed by technical staff and management on the HP 3000.
- Users may define department workloads and monitor them in batch or online, or export them.
- An Advice module comes standard with SOS/3000 that alerts you when critical performance events take place. You can also configure your own messages and have them sent to a variety of places (logged on users, console, $STDLIST, or a flat file).

Vendor: Lund Performance Solutions

HP Glance

HP Glance is Hewlett-Packard's successor to the antiquated OPT/3000. It has a much simpler feature set than its parent. It allows you to see key global and process information on one screen, which is a feature I really liked about good ol' SURVEYOR. Contextual performance data is crucial. Glance allows you to see the forest (global resources) and the trees (processes) together. I like Glance's function keys and help subsystem. There are function keys missing for some detail screens, but you can access these by issuing single letter commands.

Vendor: Hewlett-Packard

Performance Gallery

Performance Gallery is a graphical performance management tool. It is a Windows-based, color graphics package that utilizes performance data from a host collector. Performance Gallery provides key insight into your HP 3000's resource utilization and workload activity. It greatly simplifies data manipulation and analysis, giving you free time to devote to proactive system management.

Performance Gallery gathers performance data with a collector on the HP 3000 and stores it in a log file. The data is then transferred from the HP 3000 to the PC. From this data you can create clear graphic presentations of your system's CPU utilization, system bottlenecks, response times, disk and memory activity, transaction throughput and much more. Graphics may be viewed on the screen and printed out for graphic presentations.

Additionally, you may look at multiple perspectives of your system's performance by viewing many graphs at one time on the screen. To solve a new or recurring problem, you may need to analyze previous performance data from many different angles. With over 40 different graphs to choose from, during a single working session, you can evaluate the different aspects of your system's health.

Performance Gallery's richness and versatility allow you to satisfy a full range of performance reporting requirements. Its syntax-free reporting gives users the ease of working from cascading point-and-click pull-down menus. It requires an IBM PC-compatible (at least a 286), and windows.

Vendor: Lund Performance Solutions

It's a recession when your neighbor loses his job; it's a depression when you lose your own. Harry S. Truman

HP Laser/RX

HP Laser/RX is a Vectra-based reporting tool that allows color-graphic representation of historical performance data. Data is collected with a perpetually running program (SCOPE) that creates logfiles. Logfile data is then downloaded to the Vectra and analyzed.

Laser/RX boasts of nice color graphs that represent various key performance indicators such as CPU utilization, disk I/O activity, memory management, etc. It requires a Vectra (at least a 386), Windows, a CD-ROM, and a mouse.

Vendor: Hewlett-Packard

Performance Management Tools

Q-Xcelerator Resource Manager

Q-Xcelerator is a CPU resource management tool that allows you to define unique scheduling queues apart from the default MPE C, D, and E subqueues. Workloads are then assigned to these special queues. A perpetually running background process ensures that high-priority processes receive the best response times. Other processes will suffer a bit more when there is CPU pressure.

Figure 7.1 illustrates a sample logical map for one system utilizing Q-Xcelerator. Typically, all batch jobs fall to the bottom of their scheduling queue and fight with one another for use of the CPU. Given the queue map in Figure 7.1, payroll batch processing would get the most attention, while weekly status reports would get CPU time when it becomes available.

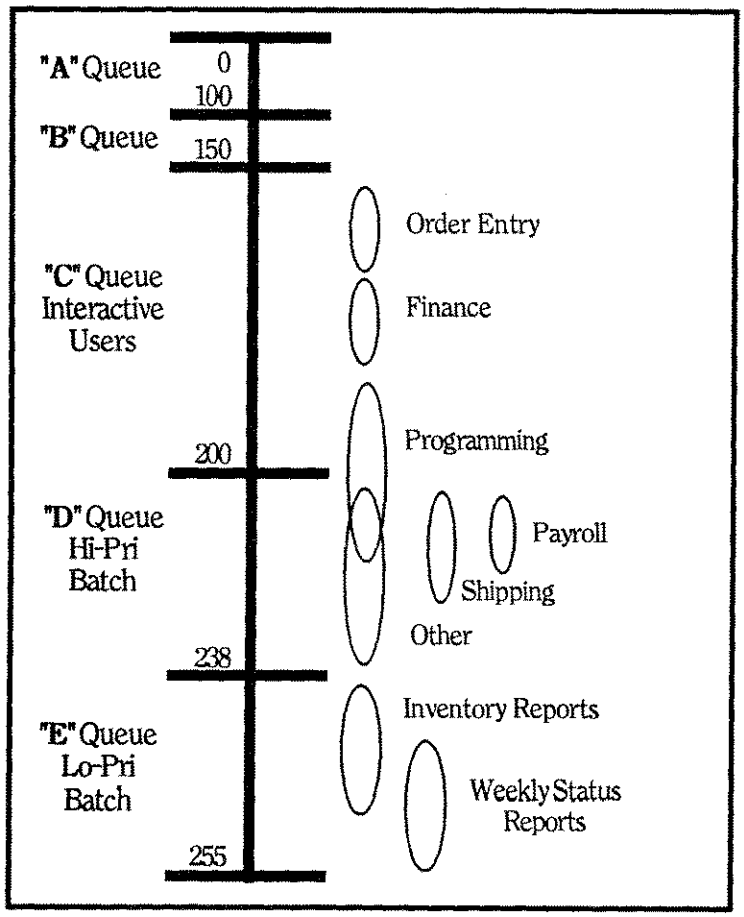


Figure 7.1 - Q-Xcelerator Sample Queue Configuration

Q-Xcelerator can be used to supplement the actions of the MPE dispatcher. The effective life of a system can be extended by "reshuffling" the resource deck.

Vendor: Lund Performance Solutions

Capacity Planning Tools

FORECAST/3000 Capacity Planner

This tool utilizes queueing network analysis to forecast future hardware needs and the effects of workload growth and modification. Forecast/3000 is the only tool of its kind commercially available for the HP 3000.

Forecast/3000 takes the science of queueing theory and implements it in such a way that even a novice system manager can easily:

- Characterize workloads.
- Validate models.
- Determine the effects of upgrading hardware.
- Analyze the effect of program modifications on response times.
- Know when the system will run out of CPU and which model will be the best upgrade choice.

Forecast/3000 utilizes data gathered by its own host-based collector. Within the PC analyzer portion, it calculates and reports present and future system performance based on the estimated departmental growth of your business. Full color graphics allow you to predict such things as future hardware requirements, the impact of future applications, and changes to existing programs.

Based on input parameters regarding application workloads and business growth rates, Forecast/3000 allows you to address a variety of "what if" questions. Some of the questions this tool can address are:

- What will be the effect of upgrading to a larger system?
- Can an upgrade be postponed?
- What response service levels can be guaranteed?
- Will job completion deadlines be met?
- What will be the effect of a prototype application on the system once it is in production with its full number of users?
- How will a new payroll package affect our CPU usage?

Forecast/3000 produces numerous reports (graphic or tabular) that profile system performance metrics such as response time, throughput, CPU utilization, performance degradation, and transaction service time. Forecast data may also be exported to your favorite spreadsheet or graphics package.

Vendor: Lund Performance Solutions

RX Forecast

RX Forecast is an add-on tool for Laser/RX that allows future planning based on business growth. It utilizes data collected by SCOPE over a period of time. Trends are analyzed and extrapolation lines are drawn to indicate future CPU utilization. Confidence limits are drawn, and it displays line graphs.

Vendor: Hewlett Packard

? Questions/Discussion ?

1. List the top five most important features to you of an online performance monitor.
2. What tools and/or methods are you using to monitor performance trends?
3. How important is the amount of overhead incurred by a performance monitor? How much CPU utilization are you willing to spend on monitors? Two percent? Five percent?

Notes:

He who lives only to benefit himself confers on the world a benefit when he dies. Tertullian

In the Beginning...

God created the Heaven and the Earth. Quickly He was faced with a class action suit for failure to file an environmental impact statement. He was granted a temporary permit for the heavenly part of the project, but was stymied with a cease and desist order for the earthly part.

Appearing at the hearing, God was asked why He began his earthly project in the first place. He replied that He just liked to be creative.

Then God said, "Let there be light: and immediately the officials demanded to know how the light would be made. Would there be strip mining? What about thermal pollution? God explained that the light would come from a huge ball of fire. God was granted permission to make light, assuming that no smoke would result from the ball of fire, and to conserve energy, the light would have to be out half the time. God agreed and said He would call light "Day" and the darkness "Night." The officials replied that they were not interested in semantics.

God said, "Let the Earth bring forth green herb and such as may seed." The Environmental Protection Agency agreed so long as native seed was used. Then God said, "Let the waters bring forth the creeping creatures having life; and the fowl that may fly over the Earth." Officials pointed out that this would require the approval of the Game and Fish Commission coordinated with the Heavenly Wildlife Federation and Audubongelic Society.

Everything was OK until God said He wanted to complete the project in six days. Officials said that it would take at least 100 days to review the application and impact statement. After that there would be a public hearing. Then there would be 10 to 12 months before...

At this point, God created hell.

Chapter 8

General Performance Management Tools and Techniques For Classic Systems

This chapter may be the last hurrah on ways to manage and maximize performance on the Classic HP 3000. These ideas, along with those in Taming The HP 3000 Volume I, should equip you to squeeze the last few drops of processing "juice" out of your MPE V machines.

General Philosophical Tips and Recommendations

I want to provide a few brief ideas for you to think about if you own a Classic HP 3000 and are considering spending big dollars to upgrade to HPPA:

- Be sure your system has as much memory as the box will support. Let's face it, used Classic memory is selling at near give-away prices. There is really no good excuse for not maxing out your system unless, of course, you absolutely do not need more. Utilize your performance tool and the pulse point recommendations in Chapter 6 to know for sure.

- There is really no excuse for staying on a Series 42 or 48 if you have some CPU overload. Series 52 and 58 upgrades, like memory, are selling at bargain basement prices. You should be able to realize a 15 to 30 percent increase in raw CPU horsepower by doing so. This does not mean, however, that you will automatically experience response time improvement. Once again, it depends on whether you have an indication of CPU bottlenecking. If you do, this inexpensive upgrade could save your life for another six months to a year, depending on your growth forecast. Beware of latent demand, which may effectively "eat up" any additional CPU provided.

- Series 70 systems are also selling at unbelievable prices, so if you have no other compelling reason to migrate to HPPA (and there are plenty of good reasons to do so), upgrade your smaller Classic system to a 70. Realize, however, that if you do procure a 70, you still have to contend with escalating hardware maintenance costs.

- Consider procuring a second or third Classic system. Especially good candidates for these are Micro/GXs. These are selling at unbelievably low prices and could effectively off-load some of your performance headaches. Of course, you'll need more disk drives, software licenses, etc. For some of the prices I've been quoted, you could actually buy a spare and simply throw the broken one away! If you are interested, give me a call for some of these sources (503) 327-3800.

System Management Tips for Improved Performance

With classic systems there are many ways to tune and housekeep them to ensure optimal performance. The following are "hot spots" you'll want to keep your eyes on.

- **Be brutal with the disk I/O monster!** Disk I/O bottlenecking of some sort is probably the most common performance problem I have seen in recent years. This performance killer is quite insidious. Not only does it prevent full utilization of your CPU's horsepower, but its effect worsens if you don't "nip it in the bud." Here is a check list to help you attack disk I/O problems victoriously. I have discussed many less high-leverage disk I/O tips in Volume I of this series. Appendix C will give you some good ideas here.

  1. **Monitor and housekeep your data bases.** Without question, this is the most common Classic killer. Keep those prominent detail sets packed. Be sure the primary path is the most important one to you (as far as optimal retrieval goes). With few exceptions, the most heinous performance problems I have seen on MPE V systems usually had something to do with data bases. Master sets need to be checked for efficiency. Appendix C outlines most of the data base-critical areas to stay up to date on.

  2. **Properly care for and feed MPE software Disk Caching.** Unless you are short on CPU and/or memory, make good use of this great MPE V subsystem. Know when to use it and, perhaps

more importantly, when not to use it. Steven Smead has some great tips on Disk Caching Optimization [17]. Also, check Appendix C for "headliners" from Taming the HP 3000 Volume I.

3. Be sure file I/O is balanced across disk drives. See Appendix C for more here. Make use of the most excellent contributed program FILERPT (written by Stan Sieler). If you think you're serious about Classic performance but don't use this program, you are not really serious.

- **Be sure that your hardware configuration is conducive to good performance.** Again, see Appendix C.

- **Based on an analysis of your system's performance for typical days, be sure you are balancing your batch/interactive mix to take advantage of the "valleys".** Excessive batch activity can put the squeeze on memory and I/O resources. Even though you run batch in a lower queue, interactive users can still be impacted by their presence (both accessing the same data base, for example). The main point here is to try to smooth out the demand for system resources.

- **Take advantage of spare PC "MIPS."** Many folks use a PC download utility (Reflection, PC2392, Advancelink, etc.) to send data to PCs for processing. As far as raw CPU is concerned, take a look at the following results from studies performed by Chris Bartram:

Platform	CPU Time (sec)	Benchmark
i286/8Mhz	203.407	(1)
i286/12.5Mhz	131.702	(1)
i386/16 w/187 Co-proc.	90.604	(1)
i386/25	50.934	(1)
i386/33	38.676	(1)
HP9000/835	31.760	(1)
HP3000/37	653.440	(2)
HP3000/MICRO	402.873	(2)
HP3000/52	256.584	(2)
HP3000/70	136.530	(1)
HP3000/70	124.008	(2)
HP3000/917LX	17.204	(1)
HP3000/917LX	152.341	(2)
HP3000/917LX	52.246	(3)

Table 8.1 - CPU Performance Benchmark

A sincere compliment is one of the most effective teaching and motivational methods in existence. Zig Ziglar

Benchmark Notes:
(1) "C" code, compiled in native mode on the target machine.
(2) "SPL" code equivalent to (1) (compatibility mode/classic system).
(3) Same as (2), but after code was "OCTCOMP"ed on MPE/iX.

Realize that even though raw CPU horsepower is greater on some PCs, this does not automatically mean greater throughput. Even so, you can obtain great benefit by taking advantage of PC power by downloading data to be processed by them in the evening when folks go home.

- **Implement proper dispatch queue tuning.** Much has been written on this subject. With the TUNE command you can make users really happy or really angry. Refer to Chapter 4 for more on this. Used properly, queue tuning can also extend the useful life of your system.

- **Beware of potential network rattlesnakes.** Activities like remote data base access and copying files across DS lines seriously affect performance. This is because the datacomm processes that serve these functions run at very high priorities (often times with decay-exempt priorities).

- **In general, minimize the use of UDCs, and user/batch logons.** Use process handling (SUSPEND/ACTIVATE) rather than the RUN command, be sure your system tables are configured large enough (see Chapter 9), and ensure that you have configured an adequate amount of virtual memory.

- **Implement a RAMDISC mechanism for I/O-intensive systems.** This jewel from Kelly Computer Systems can be a life-saver.

- **Utilize fast I/O retrieval techniques.** Commercial tools like SUPRTOOL (Robelle), OMNIDEX (DISC), SUPERDEX (Bradmark), SORTMATE and I/O MATE (Running Mate) provide various ways to retrieve data from data bases, flat files, etc., in a much faster and efficient fashion than traditional ways.

How do You Know When to Say "Uncle" and Upgrade?

There are a number of good reasons to migrate to an HPPA platform. If your system is out of gas (or projected to be out soon), that can be a good reason! This is especially true if the philosophical comments above don't apply to your site. HP has been enhancing the attractiveness of upgrading to the Spectrum systems and will continue to do so.

Some of the advantages to upgrading are:

- Potentially less hardware support costs.
- A most probable increase in performance.
- Less cost of ownership (lower price/performance ratio).
- Less square footage, electricity, and air conditioning requirements, and general environmental hassles (monstrous power conditioners and power panels, etc., are not required with MPE/iX systems).
- You'll have the latest "toy" on the block!

If the above items represent the good news, there is a potential "down side" to HPPA migration:

- Some software vendors have really gouged customers in uplift fees when they migrated from Classic to HPPA.
- If you're not willing to go to native mode, you might not experience the great performance improvement everyone is talking about.
- Even though the price/performance ratio is incredible when compared to Classic systems, there is still a significant outlay of capital necessary to make such a migration. Think of this in light of the fact that your Classic system might not be costing you much to keep.

To be humble to superiors is duty, to equals courtesy, to inferiors nobleness. Benjamin Franklin

Conclusion

Obviously the Classic systems have a time-fuse to their demise. Not to be too corny but I feel like saying one hurrah for Hewlett-Packard and the HP 3000: This system has been one of the most popular "workhorses" across all vendor lines of mini-computers. If you have been around awhile, of course you have taken some lumps in helping HP debug the hardware and operating system. I think you'll agree that the Classic has been an awesome system, which has contributed to the success of tens of thousands of companies around the world.

? Questions/Discussion ?

1. Which resource(s) are stressed on your system? How have you dealt with them?

2. Have you recently analyzed your system's memory requirements? What is the average clock cycle rate? What is the percentage of CPU utilization spent waiting for memory swapping?

3. Have you checked into an upgrade to a 5x from your 4x system? If not, why?

4. How would you rate the effectiveness of software caching on your system? Is there much difference in CPU or memory indication when you turn it on or off?

5. What are the most compelling reasons for you to migrate to MPE/iX? How important are the "environmentals?"

Chapter 9

Tuning System Tables For Optimal Performance: Fact or Fiction?

I suppose that most fields of study have a unique set of truisms that are not only believed by those involved, but are vigorously practiced and even defended when questioned. The realm of computers is no exception, nor is the HP 3000. Working with the HP 3000 family for over 13 years has given me the opportunity to acquire a plethora of information, tips, and yes, even some old wives tales.

I have a particular interest in optimizing HP 3000 performance. Having such an obsession, I began to search high and low for as many techniques as possible to maximize system performance. In my quest I uncovered many ways to seemingly accomplish this goal. To help propagate these ideas, and to perhaps dispel some of the myths surrounding the subject of system performance optimization, I wrote the first volume, Taming The HP 3000--Over 101 Ways to Monitor, Manage and Maximize System Performance on the Hewlett-Packard HP 3000.

One long-cherished arena of performance optimization has historically focused on "tuning" some or all of the system tables which lay in the viscera of the HP 3000. This optimization has had a dual focus: making the tables small enough to not waste memory, and allowing enough room to not impact performance. Anyone who has worn the hat of system manager or technical support has had, as a cherished duty, to be the "keeper" of the tables. If an operator runs into, say, an OUT OF VIRTUAL MEMORY message on a job listing, it has been the keeper's duty to figure out if virtual memory is the culprit or if this is merely a "smoke screen" for a paucity of DST entries. Ah, if the simplicity of this duty were exposed, the fact might seep out to the user population that the technical guru is really a mere mortal after all!

In this chapter I focus primarily on the findings of a study of the effect of various system table settings and the resultant performance (throughput) on one HP 3000. I attempt to show how much difference it makes to spend time adjusting various system tables. A secondary goal is to provide you with some of the undesirable side effects of having a particular table configured too low.

The Effect of System Table Tuning on HP 3000 System Performance

The question you are probably asking is whether it is time- and cost-effective to perform an analysis of system table levels and then start the system numerous times to make those changes. What's a system manager to do?

A bit of HP 3000 history will help our discussion. In the early days of the HP 3000 (circa early 1970s), main memory was the resource most often found lacking. Many techniques evolved to minimize usage of this precious commodity by various applications. Having only a quarter megabyte of memory or less was not uncommon and cost many thousands of dollars. (How much does your PC have? Did it cost less than one hundred bucks?)

System managers would toil for hours, shaving entries from the various system tables. If the PCB table, for example, was fat by 100 entries, nearly one percent of the total memory usage could be rescued from misuse by deconfiguring those entries. If a "fatness audit" on each table revealed a surplus, a possibly significant amount of main memory could then be used by MPE to minimize overhead due to memory shortage. Even with the best tuning efforts, many early HP 3000s were memory starved. An upgrade of memory, many times, resulted in a "hyperspace" improvement of response times.

As memory prices decreased, so did the era of memory-short systems, except for those on tight budgets. We then moved into the next generation bottleneck: disk I/O... but that's another story. Remember I said earlier that one aspect of tuning the tables is the impact on memory availability. The bigger the tables, the more main memory is needed for them to occupy. In the old days, five kilobytes of memory could have been a few percentage points of the aggregate. Now it's barely a "drop in the bucket." Suffice it to say that if you have three megabytes or more, maxing out your tables will have little effect on the system's overall ability to obtain sufficient memory for your processing needs (rare cases notwithstanding).

To illustrate this point, an HP System Engineer acquaintance of mine recently told me this story:

> As a performance specialist, he is frequently invited as a guest lecturer in the advanced system manager classes taught at HP. The question arose on how to tune system tables for maximum performance (he says it nearly always does). He immediately pulled out a memory dump to show the class how much main memory is on their machine. (Have you seen one printed out on a four-megabyte system? What better way to burn a box of "blue bar"!). He then ripped off one page. This represented the amount you could save, on average, by carving a few words from various tables.

Some third parties helped break artificial, maximum main-memory (marketing) barriers that were set up by HP to encourage shops to upgrade boxes rather than to insert more memory. In these days of cheap memory, you should not be a cheap-skate. As a side note, an abundance of memory will also allow MPE disk caching to do its best job if you are memory-fat.

So, what really happens when system tables are too high or too low? If they are too high, the only real downside is a waste of main memory. This is because most of the tables are memory resident; that is, they stay in memory at all costs. In contrast, data segments and a few system tables can be swapped out to virtual memory on disk. As mentioned, the waste is quite negligible for most systems these days. If system tables are too low, performance is impacted, but not necessarily as one might think.

System performance may be impacted in three ways:

1. An interruption occurs (system failure, halt or hang that forces a restart). It is generally thought that system performance slowdowns mean long response times and/or tardy batch job completions. Well, how is the response time when the "M>" prompt is showing on the console? So, I suppose we would say that performance is impacted quite severely by a table overflow that results in an interrupt.

2. An impedance occurs. If a particular resource is in short supply, applications compete unnecessarily and incur a slowdown. Such resources might be data base locks, Resource Identification Numbers (RINs), certain wait situations, and even system table entries. Sometimes applications will wait until a resource is freed up. Life then continues as normal.

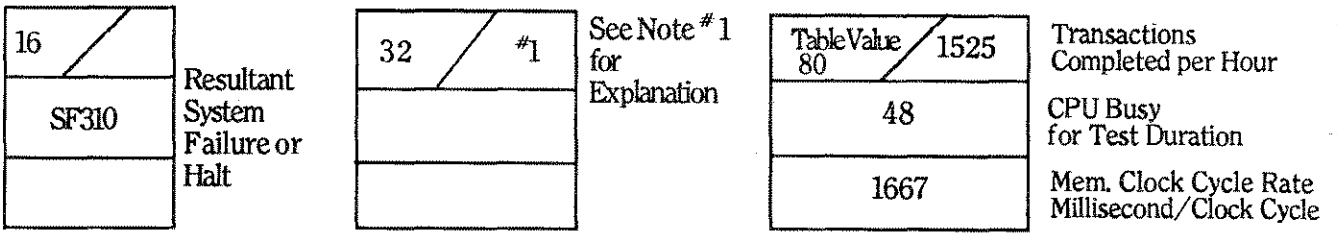
As a ring of gold in a swine's snout, So is a beautiful woman who lacks discretion. Proverbs 11:22

3. A slowdown occurs due to primary resource saturation. This is somewhat likened to item two above, but it is usually categorized separately by performance specialists. The primary resources are generally thought of as being CPU horsepower, main memory availability, and disk I/O throughput. Though system tables affect these resources indirectly, they are impacted. For example, a scarcity of Disk Request Table entries will make some applications wait, causing an artificial lowering of the overall I/O rates from what they would be if more entries were available.

The charts that follow are the results of a study that involved hundreds of system start-ups and configuration changes. I will now discuss some of the specifics as well as the guidelines followed during the study.

Table	Minimum Allowed	Initial Value	First Run	Min. Value	2X the Min.	Various Table Settings			
						A	B	C	D
CST	80	156	1525	1549					
			48	49					
			1667	1707					
XCST	16	1024	1525		32/#1	64/#1	128/#1	256/#1	512/1558
			48	SF310					50
			1667						1743
DST	70	2048	1525			280/1556			
			48	#2	#3	48			
			1667			1586			
PCB	12	128	1525		24				
			48	SF623	#4				
			1667						
IOQ	20	256	1525	1557					
			48	51					
			1667	1674					
DRQ	20	700	1525		21	22/1644	23/1634	40/1600	
			48	SF623	SF623	#5/54	#5/50	#5/52	
			1667			1936	1586	1807	
TBUFS	1	7	1525	1543					
			48	#6/51					
			1667	1646					

Figure 9.1 - Study Results - Part 1



20 Simulated Sessions- Series 37-Caching on-2MB Memory- MPE VE UB-Delta-2

Run Legend

Let's be clear about what was to be accomplished and what was not to be accomplished (uh-oh here come the disclaimers...). Although I made many system configuration changes and applications run, I know of many other scenarios that could occur in other environments that might shed a different light on a particular table. That is to say, I don't claim this study to be exhaustive even though it was exhausting!

First, I set out to see if varying the system table values and running a set of applications for a certain period of time (15 minutes) would result in better, worse or the same overall transaction throughput. I also wanted to observe some of the effects on the system and the applications of running out of table values. But it is possible that some people may have had special situations that might disprove a point I make.

The study was done on a Series 37 with two megabytes of main memory and two 7945 disk drives running under MPE V (UB-Delta-2). The application was a set of twenty programs, which simulated user input with specified think times, CPU load, and disk I/O rates. Overall, the applications placed quite a load on the system. Refer to Figures 9.1 and 9.2 for most of the discussion, as I explain how to interpret the two charts.

I have listed some of the more common system tables on the far left. The minimum value allowed by MPE is next as well as the initial values that I started with for the "First run" column. Each box represents the results of a unique configuration and the application run result. In the top left-hand corner is the table value (except for the first run and minimum value reading columns, which are obvious). If a "#" and a number appear anywhere in the box, there is a note associated with that scenario. Refer to Figure 9.4 on following pages for specifics there. If an "SF" and a number appear, this means that a system failure resulted after making the configuration change. All changes were made during COOLSTARTs. For most of the boxes, the top line refers to the total number of transactions completed during the 15-minute period. The middle line is the average percentage of time the CPU spent on pure user-busy activity. The last number is the Memory Clock Cycle Rate expressed in CPU milliseconds-per-clock cycle. This number is fairly sensitive to many situations involving memory shortage. My experience is that this value is a good measuring stick of a memory problem.

Now refer to Figure 9.1 above. This figure includes many of the more common system tables. Notice that neither the number of transactions, clock-cycle rates nor CPU busy figures vary by much of a statistical deviation. Try comparing the "First Run" column values with the results from the various configurations. Keep in mind what I am trying to point

Most people, when they come to you for advice, want their opinions strengthened, not corrected. McKenzie

out. Application throughput is generally not affected negatively with either a high or low setting for the various system tables. The system might fail, or processes might deadlock (read: hung system) or impede on resource shortage (DRQ, IOQ and so on) but generally will not run slower or faster by varying the number of table entries.

The same is true for values reflected in Figure 9.2. I also tried various combinations of very low table values to see if the number of transactions and/or memory pressure increased or decreased, to no significant avail. However, I noted some interesting instances. Notice, for example, the UCOP table. The results in column A show that the application ran OK, but when the job tried to logoff (refer to Note # 11 in Figure 9.4), a system failure #1 occurred. Compare this to column B where a system failure occurred even before the application could finish! I can't explain this...

Table	Minimum Allowed	Initial Value	First Run	Min. Value	2X the Minimum	Various Table Settings			
						A	B	C	D
SBUFS	8	20	1525	1590					
			48	51					
			1667	1668					
SWAP	128	1200	1525		256/1561	130/	132/1583		
			48	SF602	50	SF602	49		
			1667	#7	1708	#7	1706		
PRIM. MSG.	10	64	1525	1558					
			48	50					
			1667	1651					
SEC. MSG.	16	32	1525	1554					
			48	48					
			1667	1488					
SPEC. REQMSG.	10	256	1525	1553					
			48	50					
			1667	1573					
ICS	256	1024	1525		512/-	538/	544/1594		
			48	#8	#9	#9	45		
			1667				1414		
UCOP	1	128	1525		2/1540	3/1554	4/	5/1545	
			48	#10	49	#11/49	SF1	49	
			1667		162	1568		1620	
TRL	6	128	1525		12/-	18/1836	24/1633	64/1563	
			48	#12	#13	#14/55	#15/51	50	
			1667			1619	1676	1605	

20 Simulated Sessions- Series 37-Caching on- 2MB Memory- MPE VE UB-Delta2

Figure 9.2 - Study Results - Part 2

Figures 9.1 and 9.2 are full of interesting but perhaps not practically useful data. Overall, they tell us that setting table values very low does not significantly impact performance (under these circumstances at least). This, of course, is with the exception of hangs and halts. But, they do give us a bit of information regarding what happens if certain tables overflow. I caution you not to experiment with this data on your systems. I did, and got into a couple of reload situations as a result (boo, hiss!).

Figure 9.3, below, attempts to show whether any pattern outside of an expected statistical deviation would develop by varying the number of Disk Request Table entries. Again, nothing significant.

# of Entries	20	40	60	80	100	120
Memory Clock Cycle Range (CPU MS/MEM. CLOCK CYCLE	1739	1795	1657	1657	1648	1822
CPU Busy	50	50	48	48	50	47
# of Transactions per Hour	1618	1586	1555	1546	1542	1582

20 Simulated Sessions-Series 37-Caching 2MB memory-MPE VE UB-Delta2

Figure 9.3 - The Effects of Varying Disk Request Table Entries on Throughput

In the August 1984 issue of Interact magazine, Sharon Fisher, states, "It' ll be interesting to see if someone can propose supply-side configuration, where configuring all tables to their maximum would improve performance." I think that the results of this study will lend credence to the theory that it doesn' t impact performance significantly to over-configure tables. Basically, you have enough or you don' t. The only exceptions that have been mentioned to me could be the Disk Request Table, I/O Queue and, under some conditions, Terminal Buffers. Some processes might have to wait for free entries to become available, but in this study I did not observe symptoms of this problem. Perhaps various I/O rates were not high enough to become a problem. Whether your experience is the same or different, I would appreciate a quick (or not so quick) note stating your findings.

An atheist's most embarrassing moment is when he feels profoundly thankful for something, but can't think of anybody to thank for it. Mary Ann Vincent

#1 System came up OK, but when program was launched message read: "UNABLE TO OBTAIN CST ENTRIES."

#2 Just after part 5 of the system start-up, a stack marker trace appeared with: "ERROR #301--OUT OF DST ENTRIES"--HALT 4.

#3 System up OK. When trying to run application, received error: "LOAD ERROR 66--UNABLE TO OBTAIN PROCESS DST ENTRY."

#4 System up OK, but received: "ERROR #31-- LACK OF SYSTEM RESOURCES."

#5 Message: "MPE DISK REQUEST TABLE HAS OVERFLOWED", application kept running.

#6 At start-up received error message: "NO. OF ATP TERM BUFFERS FOR DRT 8 HAS BEEN INCREASED TO A MINIMUM OF 56." This is a most bothersome table error message to follow. I have had many occasions of confusion with this table. HP admits it could use some reworking!

#7 System came up OK, but System Failure #602 resulted when trying to run application.

#8 System hung just before the date/time prompt on start-up.

#9 Just before process creations began in master application a HALT 4, ICS stack overflow, occurred.

#10 System came up, but QUAD could not exit or abort; had to re-start.

#11 Job ran OK but when it tried to log-off, a System Failure #1 resulted.

#12 Four of 20 processes created, then received message: "MPE TABLE TRL OVERFLOW;" System failed with a System Failure #27.

#13 No system failure; "TRL OVERFLOW," printed out 20 times; program aborted with an integer overflow. Tried same setting two more times, received many overflow messages and a System Failure #3 both times.

#14 Tried this setting twice. First time, no system failure but system hung and CNTL B would not work; had to power off and restart.

#15 Many TRL overflow messages.

Figure 9.4 - Notes for study

Conclusion

To answer those who like spending lots of time fiddling with various table values to perhaps squeak out a few more transactions, you might spend your time more profitably turning over other rocks, like: properly caring for and feeding your data bases and disk caching; balancing your workload; penalizing online CPU hogs; shaking out configuration issues, and balancing file I/O between drives and so on.

Given the schedules of most people I know in data processing, time might be more effectively spent in areas other than table tuning. Unless of course, you enjoy myth chasing...

? Questions/Discussion ?

1. What are the settings of the more critical tables (system critical tables) on your system?

2. Have you noticed the message, "disk request table overflow"? Does it seem to slow batch or user activity? Does turning caching on/off make a difference?

Notes:

In America an average of 4 out of 5 books are read by 12% of the population, and half the people have never read a book at all.

"Change comes in two ways, by decision or trauma. Change by decision is based upon planning; change by trauma is due to a lack of planning and execution".

R.E. McMaster, Jr.

Chapter 10

Some "Gory" Classic Case Studies

This chapter on case studies concludes the discussion on Classic systems. Keep in mind, however, there also are many other references to MPE V systems throughout Sections One and Four.

The intent of this material is to familiarize you with some of the many gruesome situations people have found themselves in regarding Classic system performance. Of course I have sufficiently disguised the data to protect both the innocent and the guilty! My underlying motive in presenting this material to you is to put the "fear of God" in your bones (not a bad thing even if you don't have a computer!) in regard to properly monitoring and managing your Classic system.

I cover five unique situations involving bottlenecks that include such resources as CPU, memory, and disk I/O.

Case 10A: Honey, Somebody Shrunk The CPU...

One of my customers called recently for some consulting help. User response times were hurting bad. Figure 10.1 shows two significant CPU indicators: CPU busy on overhead (ICS and dispatcher) as well as the CPU dispatcher queue length (also known sometimes as the "ready queue").

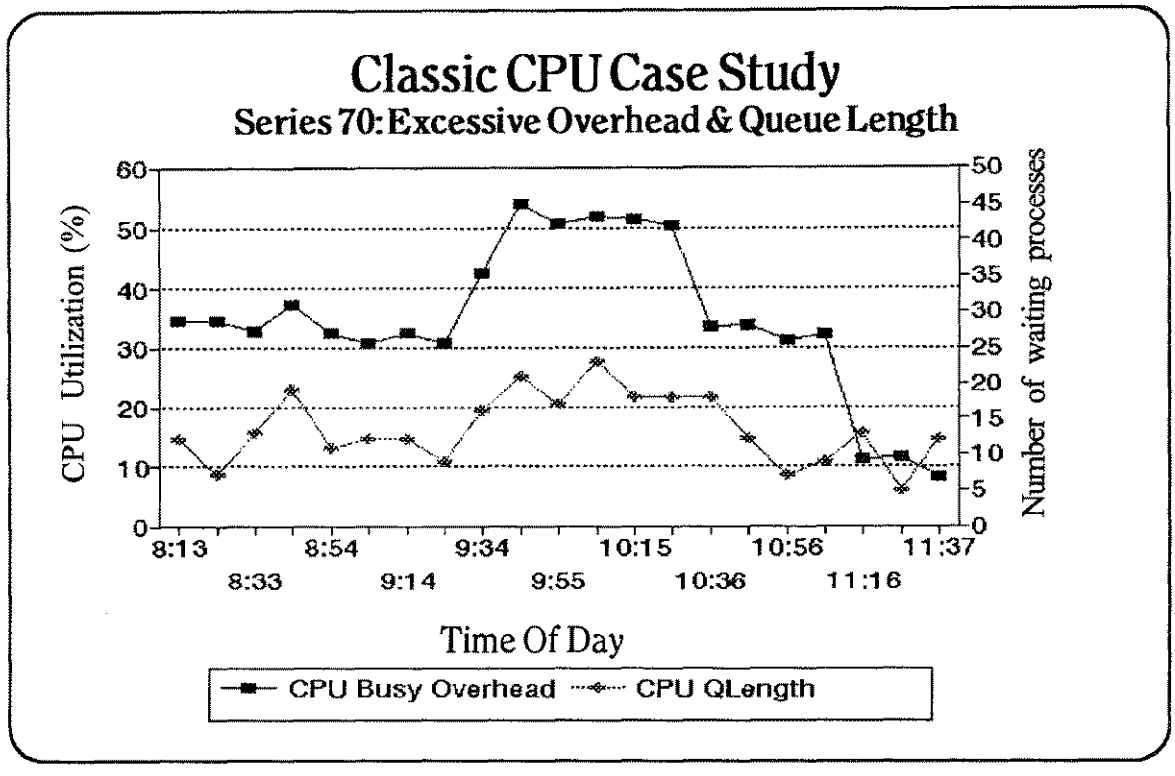


Figure 10.1 - Series 70 CPU Bottleneck Indicators

Overhead for a Classic system consists of memory management, caching, handling interrupts from disks and terminals (ICS), memory garbage collection, and dispatcher activity. ICS and dispatcher activity are inferred values on Classic systems as there are no counters for such events in the operating system Measurement Interface. All other CPU activity states are measured directly. Figure 10.1 is charting the ICS and dispatcher CPU busy time. For a healthy Classic system (with ATP terminal controllers), this value should rarely exceed 15 percent. For this system, an average of 30 percent would be on a good day! Simply put, this is like having only two-thirds of a Series 70 available for processing! It seems that a number of factors contributed to this high value.

The primary problem on this system was too much demand for CPU. Unfortunately, there is no way to distinguish between interrupts and dispatcher management. My hunch was that both were to blame. They had added a new version of a vendor's software package, which included items such as new screen handling capabilities (read: more terminal interrupts!). This, coupled with the tremendous demand for system time, caused the high overhead. However, I still had a gnawing feeling that some flaky hardware device inside or outside of the HP 3000's walls were spewing out interrupts. I have seen berserk modems, (MUX or other hardware devices) impact Classic systems similarly.

The system manager said that if he kept the number of logged on users under 100, kept data bases blocked and performed regular housekeeping, the response times were barely tolerable. He did, however make the decision to upgrade to a 957!

If your overhead and/or CPU queue length rise near these heights, get help quick!

Case 10B: Oops, Somebody Forgot To Configure The Extra Memory Board...

This case illustrates the important of adequate memory for a smooth running system. It also underscores the importance of being sure all the physical memory in a Classic system is truly configured! We were contacted by a site because they were experiencing severe performance problems on their Series 70. Online response times were absurd. Things were getting bad... fast. After some investigation, it was determined that main memory was the culprit. Here's why.

Figure 10.2 shows some of the key memory activity for a typical busy day in the life of this Series 70. Notice how the Memory Clock cycle rate is consistently less than 400 milliseconds per cycle. The Classic pulse points chart in Appendix B says that if this value is less than 2000, you are in deep weeds. This indicator is quite definitive.

Also notice how the percentage of CPU time paused for memory I/Os to complete is at least averaging 5 percent (also the bright red zone).

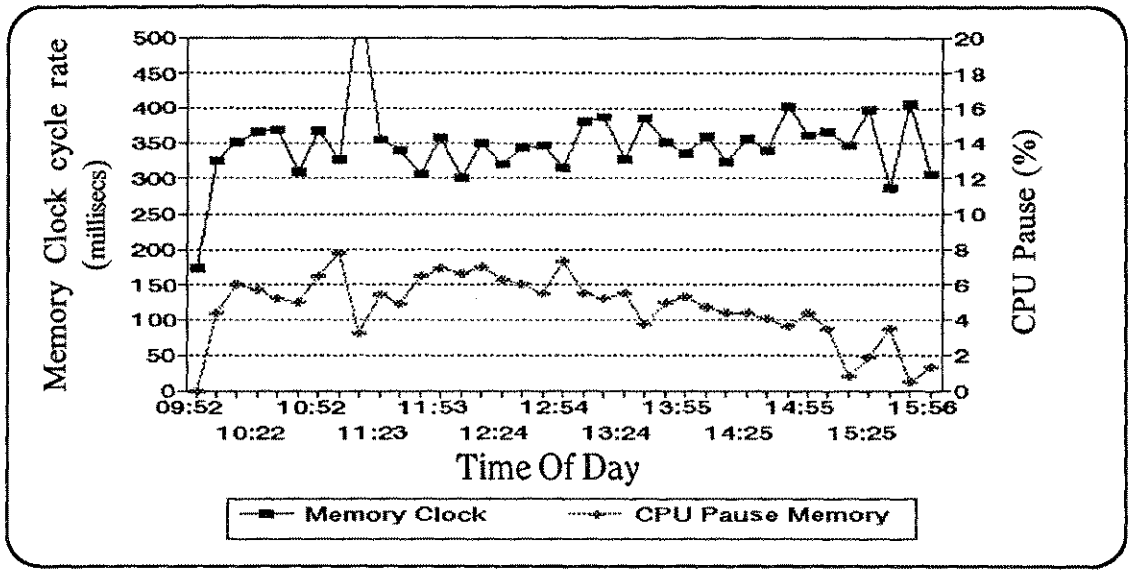


Figure 10.2 - Series 70 with Four Megabytes of Memory

A man who spends ten thousand a year will do more good than a man who spends two thousand and gives away eight. Samuel Johnson

After adding four megabytes of memory, Figure 10.3 shows the dramatic improvement of the memory pulse points. But what was the effect on the user community? Incredible, to say the least! Response times went from barbaric to civilized.

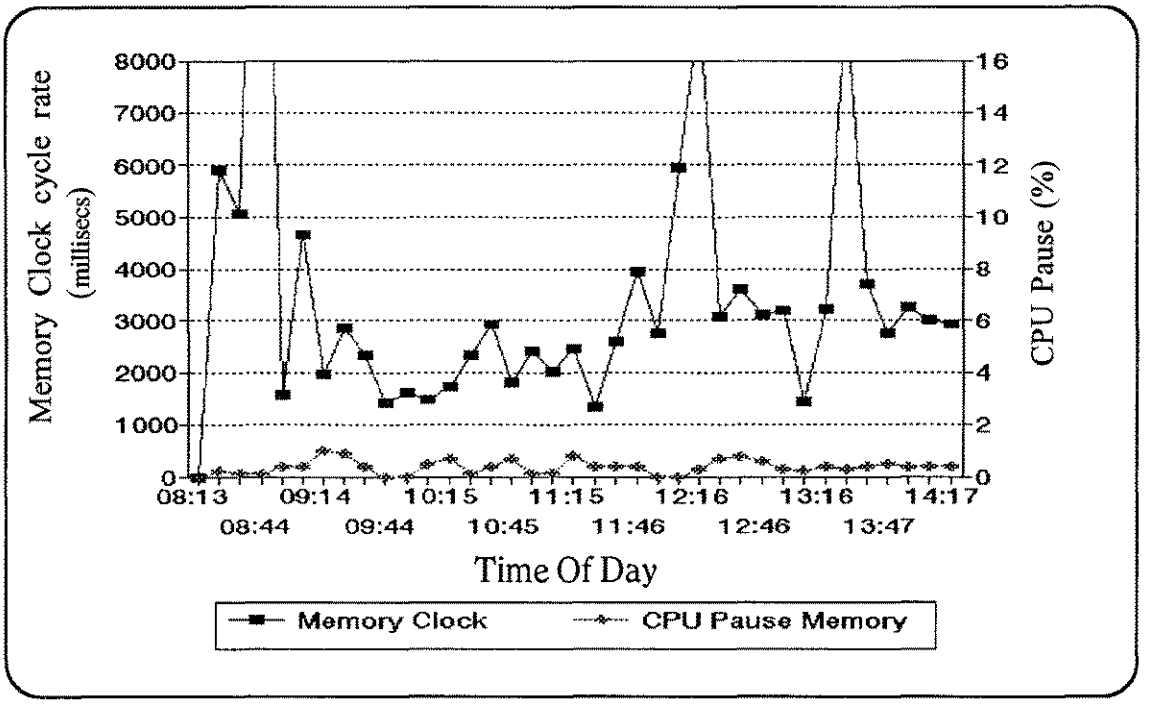


Figure 10.3 - Series 70 with Eight Megabytes of Memory

It appears that everyone thought they had eight megabytes of memory in the system. Indeed, they did—at least the other four-megabyte board was in the card cage. It's just that, well, someone forgot to actually configure it in via SYSDUMP! This site had relied upon a software vendor to perform most of their support and had apparently dropped the ball.

Sufficient memory is important!

Case 10C: Do You Think 15 Disk Drives and 16 Mb Memory is Enough to Solve Our I/O Problem?

I received a call from a frantic system manager stating that his company was in dire straits. Something had happened to their data bases over a weekend, and transaction response times increased from 15 seconds (barely tolerable) to nearly 20 *minutes!*

I went on-site for this company since it was close to having to close up shop because critical processing wasn't getting done.

Figure 10.4 illustrates one of the most hideous I/O bottleneck situations that I have ever encountered. Notice the CPU pause time, especially in the

morning. This value hovers around 40 to 50 percent most of the time. During the entire day, the CPU is either busy or paused (with a couple short idle exceptions).

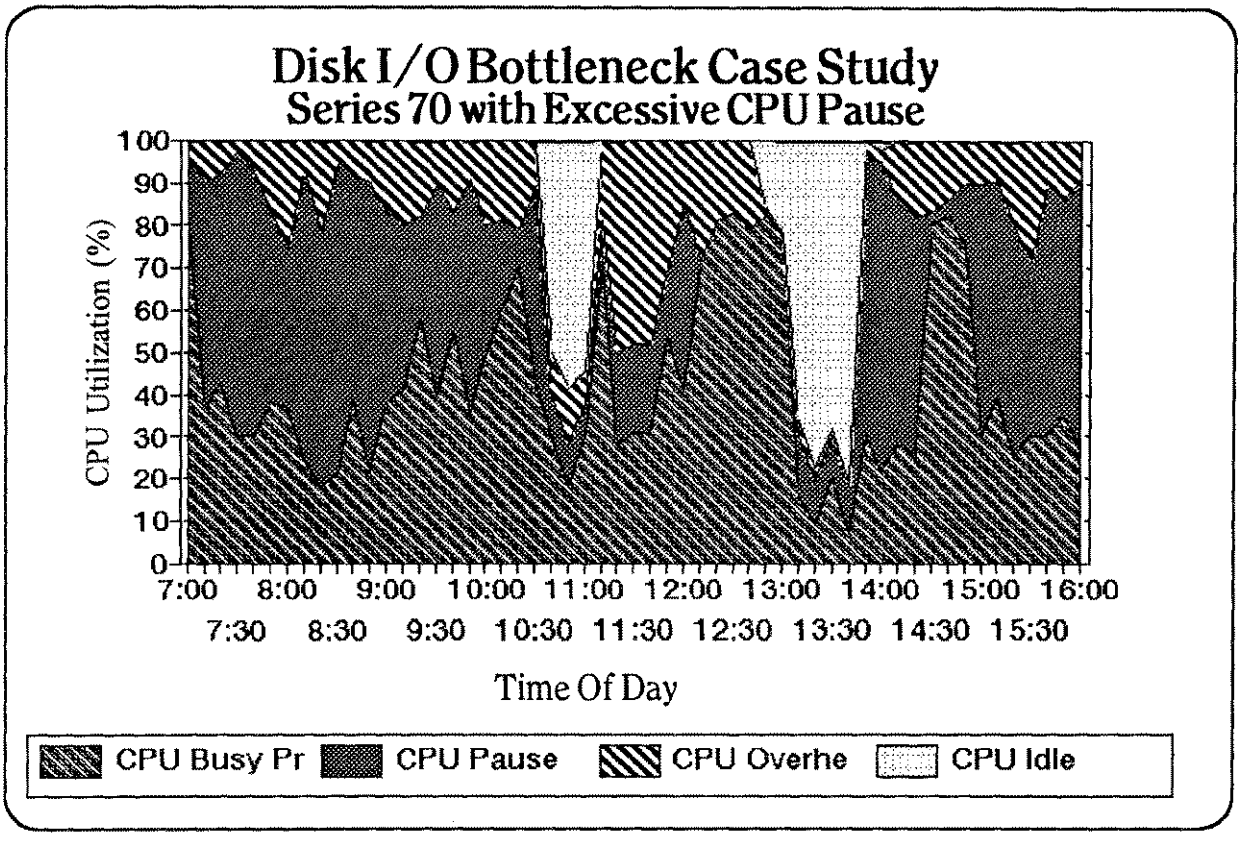


Figure 10.4- Series 70 with Severe I/O Bottleneck

I remember sitting in the computer room watching two disk drives perform the "HP Rhumba" for minutes to service a single transaction! After performing some diagnostics, I determined that one TurboIMAGE master set was in pretty bad shape.

Figure 10.5 shows that this set was full of master secondaries that pointed outside of their block. My experience is that anything around 15 percent is quite bad. You derive this number by multiplying the percentage of secondaries by the inefficient pointers and dividing by 100. In this case the result was 16.8 (36.6 * .459). Also the MAX BLKS value should be less than 50 or so. Here it was a whopping 6838! This means the master set had a very large cluster of entries.

When you sow an action, you reap a habit; when you sow a habit, you reap a character; and when you sow a character, you reap a destiny. Zig Ziglar

```
HOWMESSY (Version 1.6)           Data Base:  MYDB.PUB.XYZ            Run on: SAT, FEB  25 1990, 9:37 AM
for IMAGE/3000 databases         By Robelle Consulting Ltd.                     Page:      1

                Type                   Load   Secon-  Max   Blk                    Max    Ave   Std  Expd   Avg    Ineff Elong-
Data Set          Capacity Entries    Factor  daries  Blks   Fact  Search Item     Chain  Chain Dev  Blocks Blocks Ptrs  ation
Set1       Man    650001   478951      73.7%   36.6%  6838    10   CLAIM              9    1.58 0.84  1.00   1.66  45.9%  1.66
```

Figure 10.5 - HOWMESSY Listing for Key Master Set

The generous man will be prosperous, And he who waters will himself be watered. Proverbs 11:25

In essence, any time a record was posted, because of their poor key value selection (julian date), nearly the entire cluster of 68,380 records (6838 x 10) might have to be searched to find a free position. The cluster needed to be broken up. The only hope for immediate relief was to try increasing the capacity of the set. If given enough room, this would allow easier placement of new records. This explained the "dancing" disk drive problem!

A capacity re-size was scheduled for the evening. We simply guessed and approximately doubled the amount of space in the data set. In the morning, users were back to their barely tolerable response times of 15 or 20 seconds!

Case 10D: Oh No, No More Overtime During Month-End Processing. . .

A small company was experiencing very poor month-end batch processing. Other companies using the same applications with the same Series 42 (and memory) were completing their month-end processing in approximately 14 hours. The company under study took nearly 30 hours to close the month. I regret that I do not have performance data (screens or graphs) for this one, so I'll just describe what I recall.

When first asked to help, I began to look for the typical resource bottlenecks. They had adequate CPU and plenty of memory (a whopping three megabytes!), but the system was exhibiting some pretty awful disk I/O bottleneck symptoms. CPU pause for disk was very high. More snooping led me to a look at their data bases.

As is somewhat common on Classic systems, TurboIMAGE housekeeping was the culprit. This company had a very important master set with a capacity of 14000. (I'm sure a red flag went up in your head when you saw

the capacity...this is IMAGE 101 stuff.) Since this set was accessed for nearly all transactions, any inefficiency could have large repercussions. This set exhibited 98 percent secondaries, with secondaries x inefficient pointers being nearly 50 percent!

All I did was change the capacity from 14000 to 14009. Month-end processing went from 30 hours to just under 15! Of course you can' t necessarily expect to obtain such results on your system, but you never know until you monitor your system as a whole, and also keep an eye on those key data sets.

The operator complained that he had no more overtime during month-end processing!

Case 10E: Pig and Piglets Laboratory Simulation Example

This example is one of my all-time favorites, illustrating principles regarding:

- Increasing users versus response time and transaction throughput.
- The effects of adding memory in a bottleneck situation.
- "Knees" in performance curves.
- How systems behave when under saturated conditions.
- The asset or liability of MPE software disk caching.

A brief word about the simulation. We set up our Pig and Piglets simulator to create a load on a Series 37 HP 3000. This load consisted of session processes, all of which used some CPU time, simulated think time, and performed disk I/O. The study consisted of 28 separate experiments (whew!). Each simulated user attempted to perform as many transactions as possible within a 10-minute period. I started with three simulated interactive users (no batch jobs) and increased the users by three each time up to 21 total. For each of these I tested four different scenarios:

- One megabyte of main memory with caching off.
- One megabyte of main memory with caching on.
- Two megabytes of main memory with caching off.
- Two megabytes of main memory with caching on.

If the only diet your family had were loving words, would they be gaining weight--or losing? G. Smalley

Diminishing Returns

Refer now to Figure 10.6. Notice how the total number of transactions increase from three to six users for all of the experiments. For three experiments (1 mb mem, cache on/off; 2mb mem, cache on), transaction throughput (number of transactions accomplished within a 10-minute period) actually declined after jumping to nine sessions! This is simply the good ' ol law of diminishing returns. No matter how much you scream at the person taking your order at McDonalds, if there are people in front of you, you simply have to wait.

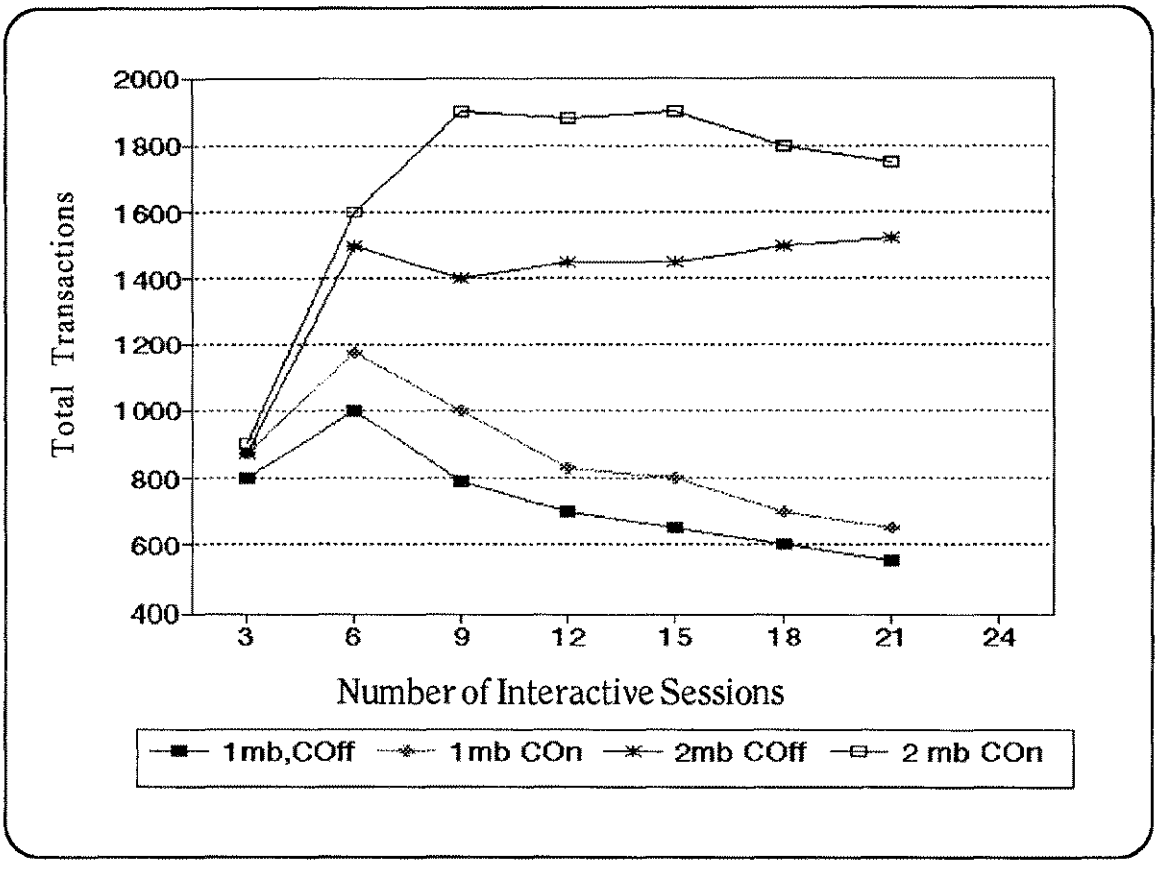


Figure 10.6 - Series 37 Pigs Simulation (Throughput)

The Effects of Adding Memory

Figure 10.6 also illustrates the effect of adding more memory to a system that is presumed to be swamped. Transaction throughput literally doubles by doubling the memory. Memory allows greater throughput since the system does not have to make the arduous journey to disk as often.

Better a grateful dog, than an ungrateful man. Turkish Proverb

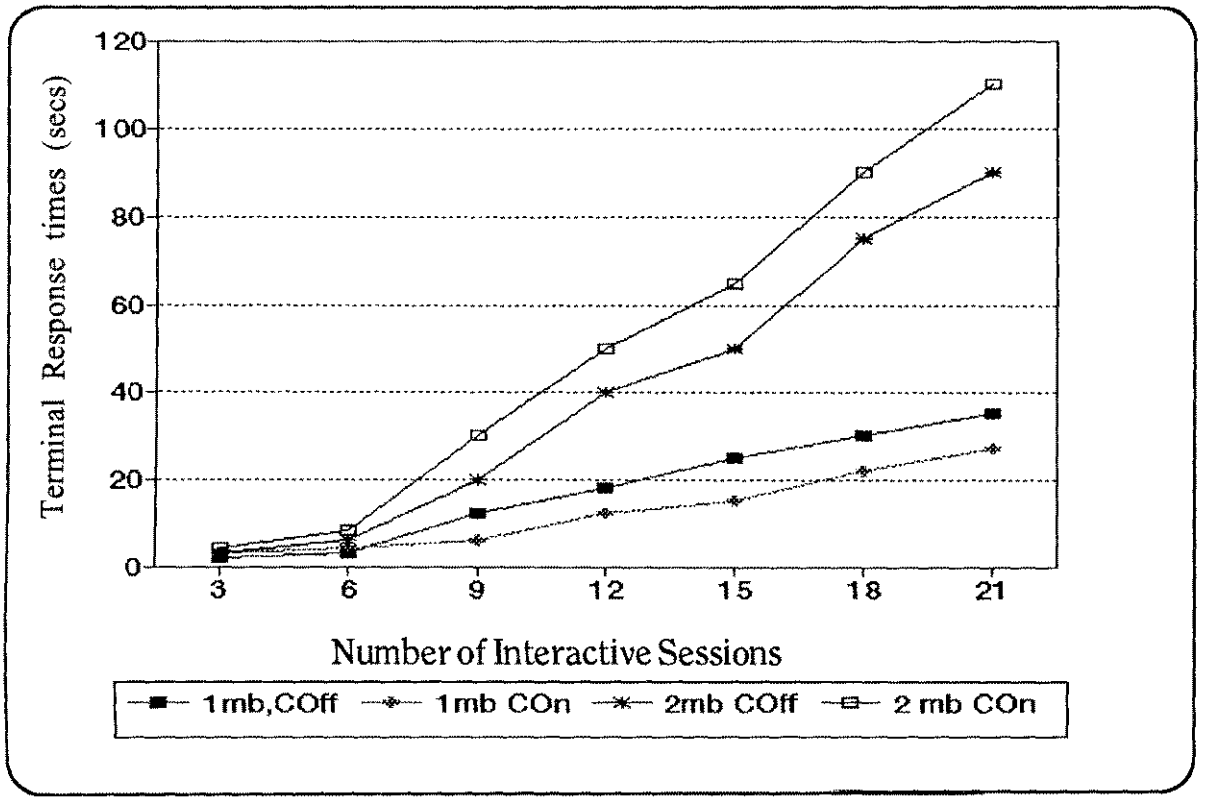


Figure 10.7 - Series 37 Pigs Simulation (Response Times)

Figure 10.7 shows what happens to users' response times coincidentally with the transaction figures from Figure 10.6. Notice the subtle "knee" on this figure that occurs at six sessions. From there, response times go from bad to worse quite steeply in two of the experiment series. This illustrates the fact that though transaction throughput stays flat, response times become ludicrous.

What about the effect (or non-effect) of MPE Disk Caching? I must say that Disk Caching, in the vast majority of situations, works wonderfully. (Note my earlier comments in Section One regarding this I/O savior.)

However, in this particular study, Disk Caching performed worse on than off! (Well, one itty-bitty exception was the two megabytes with six users-- only a squeak better that with Caching off.) This is due primarily to a paucity of CPU, certainly not enough memory, and one other reason which is not apparent from the information given (you had a hunch that I wasn't showing all the cards...). One thing that Caching absolutely needs for proper operation is good data locality. This simulation program didn't have good data locality.

In a discreet man's mouth a public thing is private. Benjamin Franklin

By the way, here's what *not* to conclude from this study:

- That a Series 37 can only handle six users! Certainly it depends on what those users are doing (i.e. their transactions' appetites).
- That MPE software Disk Caching is evil! Again, it depends on how much CPU and memory are available as well as the condition of data locality.
- Doubling main memory doubles throughput! In this experiment, yes; on your system quite probably not. Again, it depends on how much memory pressure the system is under to begin with.

Can you see why I say that this experiment is one of my favorites? There are numerous lessons to be learned from it.

Conclusion

I hope you are able to implement some of the monitoring and management ideas outlined in this section. The case studies have helped many to understand the gravity of what we are dealing with. Some of these bottlenecks are like rattlesnakes. They cripple your company's ability to move forward with timely processed information.

If you are keeping your Classic, good for you. I think it's a great thing to see these tried-and-true "work horses" last as long as is practical. I would be grateful if you would call or write me with any Classic "war" stories you have to tell.

? Questions/Discussion ?

1. In Case 10A, does the CPU queue length correlate with CPU Busy time? Under what conditions would they not correlate? Batch? Interactive?
2. Have you checked the ICS overhead on your system? At what points during the day is it the highest? Why?
3. Is all your physical memory configured into the system via SYSDUMP?
4. If the presence of CPU pause for disk indicates I/O dysfunction, is it also true that the absence of such means no I/O bottleneck exists?

Thy life is no idle dream, but a solemn reality; it is thine own, and it is all thou hast to front eternity with. Thomas Carlyle

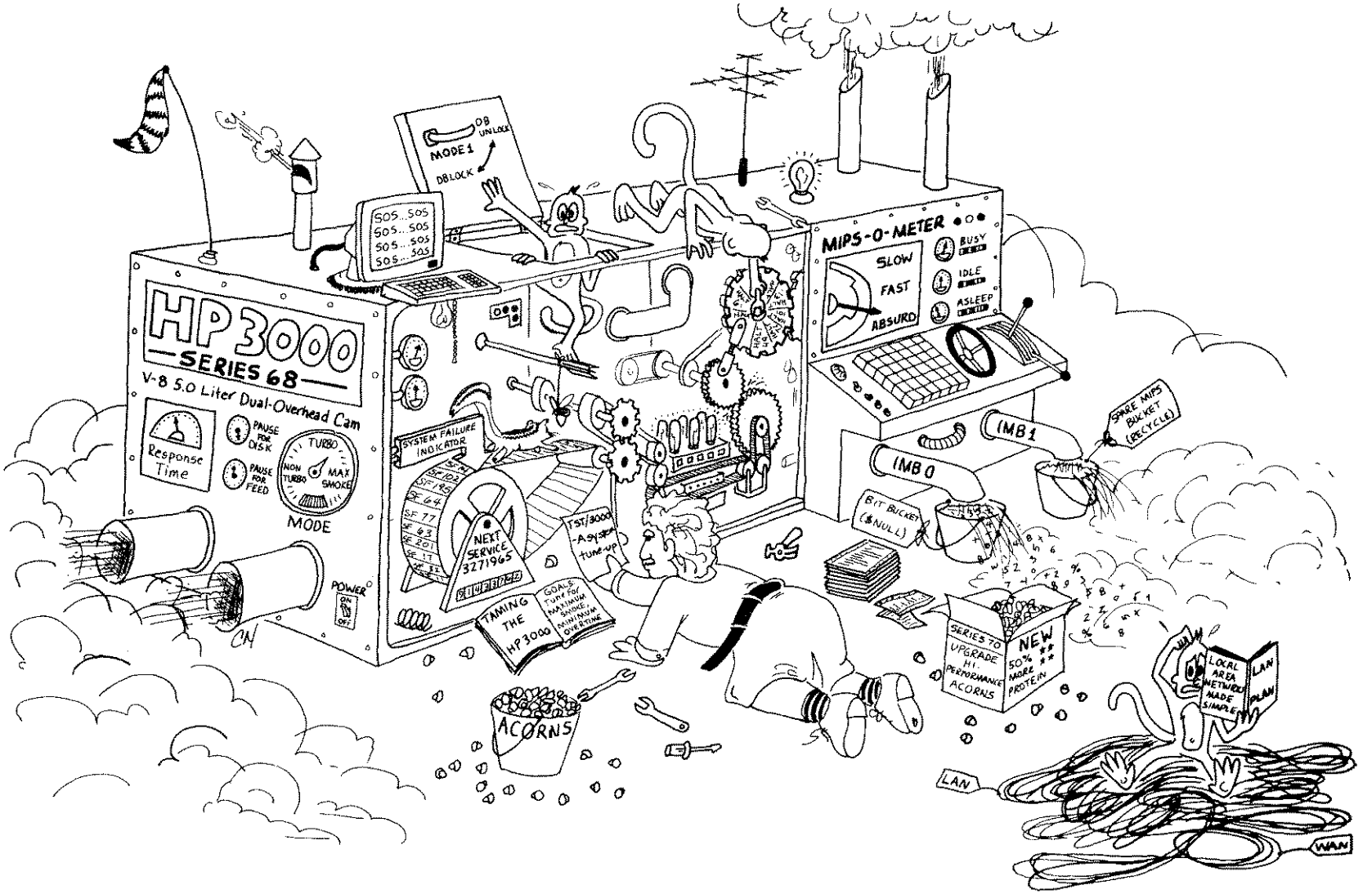
Section Three

Performance Of HPPA - RISC Machines...What We Know So Far

With the introduction of Hewlett-Packard Precision Architecture, a new era for commercial computing was inaugurated. At a minimum, these new RISC-based systems literally saved Hewlett-Packard from near doom since many customers had "maxed out" on their Classic systems. When I worked at HP as a field systems engineer (ah, the good ol' days...), I attended a meeting at which the VISION project was announced, and then un-announced. This generation of computers was to be, as I recall, a CISC-based series of systems with a bit more performance. The entire project was scrapped in favor of a new one: RISC. This decision allegedly had something to do with an ex-IBMer coming over to HP. Hmmm...

Anyway, there is no doubt that the HPPA series of systems has been a great success for HP and their customers. HP has truly done a remarkable thing by creating such a relatively easy and unobtrusive upgrade path from the Classic systems.

In this section I outline some of the basic architectural features, as well as key performance areas. With the new architecture comes a change in some of the rules when programming with performance in mind. So, I have included a chapter on some application tips for taking advantage of HPPA's performance. You'll also find performance monitoring tools and management tips in this section. Be sure to digest the material in Section One (the textbook) since most of it focuses on HPPA systems. Last but not least come case studies for HPPA systems. I think you'll find these will help you get the most out of your company's hardware investment.



You, desperately trying to tune your Classic HP 3000 to avoid the inevitable move to MPE/iX.

Chapter 11

HPPA Architectural and Performance Features

Much has been written about Hewlett-Packard's implementation of RISC, so this chapter presents a distillation of key Hewlett-Packard Precision Architecture (HPPA) features. I delineate these for the purpose of giving you a basis from which to understand the more relevant performance features of MPE/iX.

HPPA Fundamental Concepts

I cover just a few of the performance-oriented HPPA features in the first part of this chapter, then discuss the more specific operating system performance benefits. Here are some of the more prominent characteristics of HPPA.

RISC vs. CISC

For some time now, traditional system architectures have been based on Complex Instruction Set Computing (CISC). CISC uses a plethora of instructions ranging in execution time from one machine cycle to many cycles. RISC, on the other hand, is based on an Italian (Pareto) economist's 80-20 rule. Applied in a computing environment it states that 80 percent of all the instructions are performed by 20 percent of the time. RISC limits the instructions to those that execute in one CPU cycle (HPPA implementation has about 130 instructions). This is contrasted with the Classic instruction set, which is very large.

One performance advantage of RISC is that each instruction should be able to execute in one cycle. Though there are more instructions needed to perform a task, optimization (reducing the number of instructions necessary to perform an operation) is possible through "smart" compilers. This scheme allows the elimination of microcode.

No Microcode

Microcode is a normal part of CISC systems. Microcode is like complex subroutines. That is, at run time these microcode subroutines are called and have to be parsed, so to speak. This costs performance. Note this comment by HPPA expert Jason Goertz:

> *"No microcode exists in HPPA! This single fact is the most fundamental difference between an MPE/XL-based HP 3000 and an MPE V-based HP 3000, and has far reaching ramifications to the rest of the machine, both hardware and software."*[18]

HP's implementation of RISC, then, is HPPA. HPPA utilizes the reduced set of hardware instructions in lieu of microcode. With the RISC technology, HP was able to create very fast processors. Indeed, in mid-1992, the Series 990 and 992 were announced, which are hundreds of times faster than the lowly Series 37!

Large Memory Addressing Capability

Classic systems are limited to a 32,000-word data stack size. This limitation has been a real thorn in the flesh of MPE V programmers. With HPPA, this limitation has been removed. Gigantic arrays are now possible. An addressable piece of memory, called a "space," can be four gigabytes!

HPPA utilizes a concept called an address space. Think of an address space as potential memory space; until it physically gets mapped to real memory and disk addresses it is only a potential. The concept of memory space has been treated in numerous papers and articles. The main reason I call your attention to it here is because it forms the basis of Mapped Files, a neat performance and convenience enhancement over MPE V.

Support for Multiple Processors

Another great feature of HPPA is that it provides support for multiple processors. That is, up to four CPUs can actually reside in one cabinet. Until the introduction of the first of these systems (980-200), an HP 3000 could perform only one operation at a time. On a 980-200, for example, the two CPUs can actually be performing two hardware instructions at one time. This does not effectively translate into double the performance of a 980-100, however. The performance increase is certainly much better than that of the single processor 980-100, but the actual performance improvement realized depends on the kind of CPU, memory, and I/O constraints that exist. HP says that the performance rating of a 980-200 over a 980-100 is a factor of approximately 1.58.

I wouldn't be surprised if HP has a fault-tolerant system in the works. HPPA certainly could accommodate such an implementation. Overall, multiple processor support is a great way to increase performance because it is frugal in terms of dollars, square footage, ect.

MPE/iX Performance Features

With the advent of HPPA, a number of new features have been added to the HP 3000 line. These are by no means Band-Aid additions to the Classic systems. On the contrary, the majority of the operating system algorithms have been completely rewritten. I will outline some of the key performance features present in the MPE/iX operating system as of mid-1992. The hardware and operating system combine to provide for immense performance gains over the Classic systems.

Intelligent Pre-Fetching

On MPE/iX systems surplus data is brought into memory with nearly every disk I/O retrieval. This extra data will hopefully be used by the same process or others in the near future. This scheme, referred to as pre-fetching, has as its aim to eliminate disk I/O activity as much as possible. Pre-fetching is similar to Disk Caching on MPE V systems, only better. You can think of pre-fetching as "hyper-caching."

On MPE V, the amount of data brought in by Caching is calculated quite simply. The quantum (amount of data) is determined by whether the operation is random or sequential and how much the system manager says the amount should be. Typically, 16 sectors of data is brought in on a random

Let not it be imagined that the life of a good Christian must be a life of melancholy and gloominess; for he only resigns some pleasures to enjoy others infinitely better. Blaise Pascal

MPE V file read. This amount could be altered by the manager, but is not dynamically determined as is the case with MPE/iX systems.

For MPE/iX systems, the amount of data being read is determined by a couple of things. Typically, if the operation is a result of the file system requesting data, it is possible that quite a few pages will be brought in. If the memory manager "asks" for data, only a single page is retrieved. The memory manager requests data when a page fault occurs. This event is measured by most performance tools and can be a good indicator of system health. The more often page faults occur, the more often data must be requested from disk. Possibly more memory will help in a situation like this.

Since the primary purpose of pre-fetching is to reduce/eliminate disk I/O activity, how do we know how well it is performing? The bottom line for pre-fetching efficiency is how many read operations are eliminated. The Read Hit% indicator is one of the best pulse points to use in measuring pre-fetching efficiency. If you can believe it, I commonly see Read Hit% in excess of 95 percent! This is really quite a feat and gives inquiry activity a boost in performance.

You can also enable/disable pre-fetching for TurboIMAGE data bases with DBUTIL. See the notes on TurboIMAGE in Chapter 14 for more on this great feature.

Gathered Writes

Another way MPE/iX improves performance is via a mechanism called gathered writes. With gathered writes, the operating system waits until one of the following conditions takes place before physically posting data to disk. Those conditions are:

- The transaction manager reaches a "checkpoint." This means that the transaction manager's buffer has reached saturation. Data is then posted to the actual disk files themselves.
- The program uses FLOCK and FUNLOCK during file access.
- The program issues an FCONTROL(6) intrinsic. In effect, this flushes all writes to the target disk file.
- Memory space is needed by a higher priority process.
- The program closes the file (FCLOSE).

The gathered writes scheme actually eliminates write operations. On MPE V systems, all writes must be written to disk, independent of one another. There is really very little possibility of eliminating writes on the classic architecture. MPE/iX waits as long as possible, "corralling" as many contiguous pages of data as possible before posting data. One consequence of this activity is that the length of time to post an average I/O takes much longer on MPE/iX than on MPE V systems.

Compiler Optimization

If an application is to take advantage of the HPPA features, it must be compiled with a native compiler. These compilers have different levels of optimization available. Level 0 provides no optimization apart from what the program designer put into the code. Optimization levels 1 and 2 provide increasingly efficient use of CPU time.

What do we mean by optimization? Optimization involves some of the following mechanisms in reducing the net amount of CPU necessary to perform an application task. Without going into too much detail, some of the ways in which this frugality is accomplished is with:

- Global data flow and alias analysis (knowing which data items are accessed by code and which data items may overlap).
- Constant propagation (folding and substitution of constant computations).
- Loop invariant code motion (computations within a loop that yield the same result for every iteration).
- Strength reduction (replacing multiplication operations inside a loop with iterative addition operations).
- Common subexpression elimination (removal of redundant computations and the re-use of the one result).
- Dead code elimination (removal of code that will not execute).
- Branch optimizations (transformation of branch instruction sequences into more efficient instruction sequences).
- Live variable analysis (removing instructions that compute unnecessary values). [19]

These and many other transformations are performed against source code during compilation with optimization levels enabled. On the practical side there are a few things to keep in mind when using optimization. First, certain assumptions are made about data alignment and placement. This adds a burden on the programmer who is trying to debug such code. Since some optimization features actually move code around, a programmer could be quite frustrated when trying to find code that should be there (using DEBUG) but is not. It is best to start with a level 1 optimization and fully compile the application before proceeding to the next level. It is possible to attain a 5 percent or so reduction in the amount of code executed for some applications. Depending on the criticality of the particular application in question, it may or may not be worth the added "up front" time to make the application compile with another level of optimization. Be sure to apply rigorous testing of applications after another optimization level has been implemented.

MPE V Instruction Emulation (Compatibility Mode)

The MPE/iX operating system has beautifully accommodated those wishing to migrate from MPE V. The vast majority of programs that run on the Classic systems will be able to run under MPE/iX without re-compiling with a native compiler. HP has done this by providing run-time translation of MPE V program code. While this is not a performance feature per se, it is a wonderful way to migrate to HPPA relatively pain-free. The down side of such a migration is that you will not be able to take full advantage of HPPA performance gains. In fact, some cases I have seen barely broke even with respect to performance. Hewlett-Packard has termed this run-time translation feature "compatibility mode."

Native Mode Program Optimization

If you really want your applications to get the most performance possible (i.e. quick batch turnaround time, short user responses, etc.), you really need to compile programs with a compiler that "speaks" the language of the HPPA systems. Each of these compilers offers some code optimization (see Compiler Optimization) that carves CPU time from your programs. Unless you have some other reason for migrating to an HPPA system, you will be short-changing yourself of some performance by not taking advantage of native mode (NM).

Object Code Translation for Compatibility Mode Applications

If you do not have source code available for your applications, HP has provided an intermediate level of performance optimization within MPE/iX. If a fair percentage of a given program's instructions are due to user code (as opposed to system code), the OCTCOMP program may give a performance boost. The Object Code Translator (OCT) takes your object program as an input and creates a new binary program as its output. This new file is fully executable. Be forewarned that this new program will consume a tremendous amount of disk space (10 to 15 times more than the original!).

What kind of performance gain can one expect by OCTCOMPing programs? From the tests I have seen, you could expect anywhere from five to 20 percent less CPU to perform similar compatibility mode operations. If your program consists primarily of system intrinsic calls, which are already in native mode, then you will not realize such a great gain by OCTCOMPing that programming. On the other hand, if your program consists of lots of computational activity, you could expect some pretty good results from utilizing OCTCOMP.

Mapped Files

The introduction of mapped files in MPE/iX has increased the potential for two major items: the potential for better performance and larger (more elaborate) applications. Mapped files have the following attributes:

- They always use less CPU to perform identical operations when compared to the file system. Keep in mind that this does not necessarily mean performance is better!
- The file system performs better when sequentially reading from or writing to large files. However, if many files are reading the same file, mapped files could be better (unless you are short on memory).
- Randomly read large files have the best performance when they are mapped.

Pick out a character trait that you appreciate in your child. Is that child tidy? Patient with a brother or sister? Truthful? Give praise today for that quality that has nothing to do with performance, and you will have planted a seed of unconditional love. G. Smalley

You can think of a mapped file as a very large array. A mapped file is really a tremendous feature for programmers because file data can be accessed as easily as if you were reading an element within an array. Note the following comment from Eugene Volokh:

> *"A mapped file is actually not a type of file but rather a type of file access. Almost any file can be opened as a mapped file; once your program opens a file with the mapping option, it will be able to access the file as if it were an array in its own data area. Instead of accessing a file using FREAD and FWRITE (or the equivalent language constructs, such as Pascal's READLN or WRITELN), you'll be able to access the data of the file just as you'd access any array (or record structure)."* [20]

I also like the following comment by Steven Smead summarizing access strategies utilizing mapped files:

> *"...when designing an application, be sure to consider the user requirements of the program. If response time is always the goal, use the approach that delivers the lowest wall times. If low CPU utilization is the goal and transaction turnaround time is less important, the access strategy that uses the fewest CPU seconds should be employed. When in doubt, test your application using both access strategies to see which method best meets the program performance criteria."* [21]

If you are serious about investigating the use of mapped files, you must read his article (note the reference).

Conclusion

In this chapter I have briefly outlined some of the more prominent features of HPPA and MPE/iX from a performance standpoint. Use this information, along with the contents of Section One and the Case Studies in Chapters 13 and 14 to gain a better understanding of how to best monitor and manage your system.

? Questions/Discussion ?

1. Have you taken advantage of some of HPPA's features, such as:

   - Large memory addressibility
   - TurboIMAGE pre-fetch
   - Transaction management
   - Full native mode

   If not, why?

2. If you are only utilizing CM, what prevents your firm from going to full NM? or at least OCTCOMPing?

3. What are some good reasons for not utilizing compiler optimization?

4. How could mapped files be best utilized at your shop?

It is far better to forgive and forget than to hate and remember.

"Politicians around the world are breathing hard... hoping citizens remain blind to the truth... the State, however almighty, however seemingly powerful, is no more than a state of mind, a fiction. The State exists only so long as citizens believe it exists. That is the message of the Soviet collapse."

Dr. Antony Sutton

Chapter 12

Programming Considerations For Optimal MPE/iX Performance

In this chapter I present some ways to optimize MPE/iX performance from a programming standpoint. If you have read Taming the HP 3000, Volume I, you will recall that one of the goals was to present non-intrusive performance improvement ideas. That is, many shops either do not have source code, or would rather not invest time, effort and money in re-vamping existing applications. Most would like to know how to maximize performance without tampering with "unbroken" code. That philosophy is still a driving force behind this volume. Having said that, I do think it is important that you have a basic understanding of what things to do, or not do with respect to program development in order to take advantage of the HPPA and the MPE/iX operating system.

Here you'll find some of the more prominent strategies for application design. Many papers have been written on the subject of MPE/iX programming for optimal performance, some of which I have cited in the bibliography as recommended homework assignments.

Foundational Programming Philosophy Behind MPE/iX Programming

As I mentioned, HP desired to shift performance bottlenecks onto the CPU. This has been described by one system engineer I know as HP's desire to create "black boxes" with minimal or no "tunables." If performance became a problem, the cure would simply be to upgrade the CPU or memory. While this is not entirely a bad philosophy, it can be frustrating to customers who like to fiddle with various performance knobs and dials. I suppose I am one of those "tune-aholics." Nevertheless, HP has made serious moves toward shifting bottlenecks to the CPU.

One philosophical strategy for MPE/iX programming is to focus on minimizing the amount of CPU necessary to perform desired activities. Most of the ideas cited here focus on minimizing a program's CPU appetite, but I make a few comments on the issue of data locality.

As it turns out, as fabulous as HPPA is in reducing disk I/O bottlenecks, these systems still are not immune to some I/O diseases. So, I will preach a little about program and data locality.

Utilize Native Mode as Much as Possible

It is probably obvious to you by now (having plowed this far through the book), that if you are serious about maximizing your application's performance, you will need to compile your programs with a native compiler. Those available are COBOL, FORTRAN, PASCAL and C. There may be others, but these certainly are the most popular. Taking your applications from compatibility mode (run-time translation... reminds me of run-time uncompiled BASIC!) to native mode typically will buy you very noticeable performance gains.

Should you convert all of your MPE V applications to native mode? Not necessarily. This question is addressed in Chapter 16 (Commonly Asked Performance Management Questions).

Take Advantage of Native Compiler Optimization

In the first part of this chapter I referred to the concept of compiler optimization. If performance is an issue at your site and you are able to perform some program changes, you might consider utilizing the various optimization levels found within the native compilers.

Compiler optimization simply shaves CPU cycles from applications. Most of the compilers have three levels of optimization. Level 0 performs no optimization. Levels 1 and 2 gradually decrease the amount of CPU. You should check current bug reports for various compilers (especially COBOL- it doesn't have level 2) with respect to optimization since some reportedly have had problems. Why not simply have just the third level? The catch here is that to properly take advantage of compiler's optimization, you will probably have to perform some code changes.

Sid Smith said it well with regard to compiler optimization:

> *"Various levels of optimization are supplied to allow the programmer to effectively debug the code before fully optimizing it. Each level of optimization makes certain assumptions about data alignment and placement to optimize register usage and reduce code, but in so doing adds complexity to the debugging process."* [22]

The optimization process actually plays tricks with your code. It is possible to be using DEBUG on your program and find that some code has been moved or removed! The optimizer functions try to be smart indeed; they do quite well. It's just that you will have to get used to code that has moved!

It is possible to see the amount of executable code reduced by 5 percent or more.

Avoid the Use of Extra Data Segments (XDS)

Due to some extreme overhead, MPE V applications that use extra data segments (and many do!) should be re-tooled to utilize mapped files instead. Compatibility mode switching is especially ugly when an XDS is involved. Remember, the XDS was designed to provide you with more memory data stack space. Due to its virtually unlimited stack space, an XDS is not necessary.

If you are using extra data segments for interprocess communication, then just use mapped files. Contributed library routines are available to take the place of XDS.

Avoid Message Files

At the time of this writing, HP's message file facility is still in compatibility mode. This means if you have an application that is in native mode, it will have to switch back to compatibility mode every time the message file facility is accessed. This means performance degradation. But how much does this cost? That depends, of course, on how often message files are used in an application. It also depends on how much of a critical link that application is to your overall operation.

As an alternative to using message files, HP has a facility called PORTS. PORTS is part of the operating system architected interface (AIF). It could be purchased for about $2,000.00 in early 1992.

Avoid Compatibility Mode Services

By the time you read this, there may only be a few pieces of the operating system that are still in compatibility mode. But in general, try to avoid using any utilities or operating system services that have not been compiled with a native compiler. Check with your application vendor!

Make Use of Large Arrays

With the advent of plentiful and cheap memory and the removal of most (if not all) of the various memory-related limitations on Classic systems, it makes good sense to utilize large memory resident arrays instead of a disk file when possible. Remember, memory data transfers are much faster than disk I/O transfers.

Ensure Optimal Data Locality

Just as with Disk Caching for Classic systems, MPE/iX will function best when data is localized. This means that the operating system pre-fetch mechanism will be able to eliminate disk accesses if data is grouped in such a way that many users share much data. But how do you improve data locality?

Optimizing data locality involves a number of issues that I'll present in the form of some questions. These are:

- How do users typically access data?
- Do applications open up a data base and stay in it all day?
- What is the probability that two users will be attacking the same data bases? How about data sets? How about data items?

It is hard for me to underscore how important good locality is to an HP 3000's performance. Believe it or not, it is common for MPE/iX systems to exhibit a 95 percent Read Hit. Try to fathom this. This means that 95 percent of the time any data needed will be found in memory. Although I've seen, studied, and lectured on this many times, it never ceases to amaze me!

A high Read Hit% is a function of adequate main memory, users' access habits, and how localized necessary data is.

The issue of data locality is both a housekeeping and design issue.

Locality Lessons

To properly care for and feed the I/O elimination features of MPE/iX, one needs ample memory and good data locality. The following will help keep locality optimal:

- Keep data sets (masters and details) clean and tidy; this means eliminate excessive secondaries and keep detail set chains short.
- When you design an application, keep in mind that the more you can facilitate the localizing of data on disk, the easier life is for you and those who have to manage the system later.

Avoid Character Mode When Possible

As I mentioned earlier, block mode is preferable on an MPE/iX system due to excessive terminal activity causing elevated CPU usage by individual processes. An HP "Labby" was very adamant about this when he stated that character mode was "lethal" to MPE/iX.

? Questions/Discussion ?

1. Since much of the strategy for performance improvement on MPE/iX systems involves CPU frugality, how has your development staff been equipped to do so? Have they read this chapter?
2. After seriously studying locality, what strategy have you developed to ensure good data locality within your data bases?
3. Have you measured the impact of character terminal I/O on your system?

It is only when men begin to worship that they begin to grow. Calvin Coolidge

I know men; and I tell you that Jesus Christ is no mere man. Between Him and every other person in the world there is no possible term of comparison. Alexander, Caesar, Charlemagne, and I have founded empires. But on what did we rest the creations of our genius? Upon force. Jesus Christ founded His empire upon love; and at this hour millions of men (and women) would die for Him.

Napoleon Bonaparte

Chapter 13

Monitoring HPPA Performance Issues and Tools

In this chapter I cover some MPE/iX monitoring issues, utility commands and monitoring tools that help you view performance. Basically the same principles apply to monitoring HPPA systems as the Classic systems (or any other multi-user computer, for that matter), but with a few new wrinkles. Let's cover the issues first.

Same Ol' Monitoring Issues...

Basic performance monitoring and management of MPE/iX systems are very similar to that of MPE V systems. The primary resources still need to be checked periodically to see if they are at or close to saturation.

So, you will want to keep your eye on the following "normal" areas:

- **CPU utilization** - What percentage of the CPU is being consumed by high-priority processing? What is the average CPU queue length? What percentage of process delay times are due to preemption?

- **Memory utilization** - How much CPU is being spent on managing page-fault activity? How fast is the memory clock cycling through memory?

- **Disk drive utilization** - How much of the CPU's time is paused waiting for disk I/Os to complete? What is the average disk queue length? What is the disk I/O read/write rate?

This is fairly standard stuff. And I might add, essential. You should consider reading again the textbook section of this book to review the basics.

Although there are similarities in the kind of metrics that need to be monitored, many of the thresholds that indicate good, bad, and ugly resource utilization have changed. Some have changed substantially. Note the differences on the two Pulse Points charts in Appendix B.

. . .With Some New Wrinkles

With the advent of MPE/iX came some pretty substantial rewriting of good ol' MPE. Many new major functions were added, but HP was able to keep a very large subset of MPE V functionality. Although many of these monitoring issues are covered in the textbook section, I will briefly outline them here.

The Switch Facility

If any substantial workload on your system is not compiled into native mode, you will certainly want to monitor compatibility mode switch activity. Keep in mind that switches occur as a courtesy to you so that you do not have to re-compile your source code. Cost for these switches are incurred by the process that performs a switch. A switch from native mode to compatibility mode (and vice versa) simply costs CPU. That CPU time is charged to the process that called the switch intrinsic.

At the global level, you want to be sure NM to CM switches are minimized because they are expensive (they cost CPU time). CM to NM switches can be higher, but still are not desirable. You also should periodically scan or chart the percentage of all CPU time spent in compatibility mode. If this number is high, you could reduce CPU utilization a fair bit by compiling your busiest programs into native mode.

At the process level, you also can keep an eye on programs that perform a disproportionate amount of mode-switch activity. These are good targets for native mode compiling.

Transaction Management Activity

The MPE/iX transaction manager intercepts write activities on behalf of data structures (TurboIMAGE is the most common) and keeps these unposted data in main memory. It also keeps a special disk journal file until it posts the data to the target files. When monitoring an MPE/iX system, you will want to watch for bizarre transaction manager behavior. Most of the strange behavior I have seen was prior to release 3.0. One case study in Chapter 15 notes this activity.

Transaction manager activity manifests itself in a couple of ways. First, you might see a couple of processes (PINs 5 and 9 prior to 3.0 and PINs 9, 10 and others from 3.0 and beyond) periodically taking a lot of CPU time. Prior to 3.0, this activity was pretty ugly. After 3.0 you will want to watch for a number (not sure how many) of processes (PINs 10, 11...?) occasionally becoming active at a fixed priority at the top of the C queue. Just be sure these processes do not become hogs. You may want to create a separate monitoring workload for them (SOS/3000 Performance Advisor or LASER/RX) to watch their cumulative activity throughout a busy day.

Since the transaction manager is an integral part of MPE/iX systems, you may want to watch some of its internal activity. You can view this information with SOS/3000 Performance Advisor ("N" hotkey) or writing routines to access the Measurement Interface Architected Interface.

Also, watch the disk queue length throughout a busy day. It would be good to chart it if you question how active your transaction manager is. Prior to 3.0 you will see massive disk queue lengths occurring every five to 45 minutes (depending on how much write activity is taking place). After 3.0, this was somewhat smoothed out.

Keeping an Extra Watchful Eye on Memory

In previous discussions, I explained the importance of memory for HPPA systems. With this in mind, I want to doubly encourage you to watch for memory yellow zone indications. Refer to Chapter 6 and the Pulse Points charts in Appendix B for more on memory activity.

Stan Sieler explains why memory is so important for MPE/iX:

> *"Main memory is vital to the performance of the system. Unlike MPE V, which tended to degrade slowly, MPE/iX will suffer a very sharp drop in performance when not enough memory is available. Economize on everything else... and buy memory."*[23]

Plough deep while sluggards sleep; and you shall have corn to sell and to keep. Benjamin Franklin

Don't Be as Paranoid about Disk I/O Bottlenecks

MPE V systems regularly suffer from disk I/O bottlenecking. An I/O bottleneck on an MPE/iX system would be the exception rather than the rule. The elaborate memory management scheme, along with pre-fetching and gathered writes, tends to mitigate physical disk writing activity. Consequently, if you were extremely diligent about attacking areas of disk I/O hindrance (data base housekeeping, file balancing, blocking factor optimization, etc.), you can back off a bit...but not too much! Certainly I/O can become a problem. All MPE/iX has done is push out the knee in the I/O bottleneck curve a bit further than it was on MPE V systems. Keep a watchful eye on I/O, but not too watchful.

Monitoring Indications of Data Locality

As mentioned previously, good data locality is critical for a properly performing HPPA system. The best indicator I have found for this is the Read Hit%. It is very instructive to chart this value, correlated with various workload activities. This will give you a hint as to which applications need better data locality management (data set re-packs, etc.).

Performance Monitoring Tools

Here I outline the tools and commands available to help you monitor your HPPA system's performance. Many of the commercial monitors reviewed in Volume I are available for MPE/iX. Consult Chapter 6 in Volume I of this series for descriptions of such tools.

SOS/3000 Performance Advisor

I originally conceived SOS/3000 Performance Advisor for my own needs as a consultant. I had used SURVEYOR for years to perform system performance analyses. Simple as it was, SURVEYOR had some excellent data. Then, when the VMIT release of MPE V hit the street, SURVEYOR began spewing "flying bytes" into main memory. Many sites reported all kinds of strange system failures. I noted nearly a dozen different system failure types--usually indicative of memory corruption. SOS/3000 Performance Advisor came about so that I could supply answers to my customers.

While I wish to speak without bias, I am the original designer and author of the tool. However, hundreds of my customers, many who have performed extensive comparisons with other tools, tell me that it is the most comprehensive, cost-effective performance tool available for the HP 3000.

Some of the design criteria I had in mind for this tool are as follows:

- It had to be extremely low in overhead. It was written in the "C" language, and am I glad it was. Many clients have brought to my attention that SOS/3000 has the lowest overhead of all the tools they compared. With one tool, CPU overhead was consistently 350 percent to 450 percent more than with SOS/3000. It also takes full advantage of the HP Architected Interface (AIF).
- It had to cover all the performance bases (see Chapter 2).
- It had to cover all the performance angles (see Chapter 3).
- It had to be easy to use (allowing some help in interpreting difficult technical data).
- It had to have enough depth for even a seasoned performance specialist to be comfortable with.

I believe that we have accomplished all of these goals, and more. Here are some of the features of SOS/3000:

- A global screen allows you to view global, process, and workload data and then use over a dozen detail screens to zero in on performance problems. Nearly every screen allows a graphical or tabular view of the data.
- Full logging capability and host graphics creation are standard; an infinite number of graphs may be user-defined and printed or exported to PC spreadsheets. Data is also compatible with Performance Gallery, a Windows PC-based 3D graphical presentation package (see below).
- Performance data can be gathered, extracted and presented on the host in a fully automated fashion. Historical performance information may then be reviewed by technical staff and management on the HP 3000.
- Users may define department workloads and monitor them in batch or online, or export them.
- An Advice module comes standard with SOS/3000 that alerts you when critical performance events take place. You can also configure your own messages and have them sent to a variety of places (logged on users, console, $STDLIST, or a flat file).

Vendor: Lund Performance Solutions

HP Glance

HP Glance is Hewlett-Packard's successor to the antiquated OPT/3000. It has a much simpler feature set than its parent. It allows you to see key global and process information on one screen, which is a feature I really liked about good ol' SURVEYOR. Contextual performance data is crucial. Glance allows you to see the forest (global resources) and the trees (processes) together. I like Glance's function keys and help subsystem. There are function keys missing for some detail screens, but you can access these by issuing single letter commands.

Vendor: Hewlett-Packard

Performance Gallery

Performance Gallery is a graphical performance management tool. It is a Windows-based, color graphics package that utilizes performance data from a host collector. Performance Gallery provides key insight into your HP 3000's resource utilization and workload activity. It greatly simplifies data manipulation and analysis, giving you free time to devote to proactive system management.

Performance Gallery gathers performance data with a collector on the HP 3000 and stores it in a log file. The data is then transferred from the HP 3000 to the PC. From this data you can create clear graphic presentations of your system's CPU utilization, system bottlenecks, response times, disk and memory activity, transaction throughput and much more. Graphics may be viewed on the screen and printed out for graphic presentations.

Additionally, you may look at multiple perspectives of your system's performance by viewing many graphs at one time on the screen. To solve a new or recurring problem, you may need to analyze previous performance data from many different angles. With over 40 different graphs to choose from, during a single working session, you can evaluate the different aspects of your system's health.

Performance Gallery's richness and versatility allow you to satisfy a full range of performance reporting requirements. Its syntax-free reporting gives users the ease of working from cascading point-and-click pull-down menus. It requires an IBM PC-compatible (at least a 286), and windows.

Vendor: Lund Performance Solutions

HP Laser/RX

HP Laser/RX is a Vectra-based reporting tool that allows color-graphic representation of historical performance data. Data is collected with a perpetually running program (SCOPE) that creates logfiles. Logfile data is then downloaded to the Vectra and analyzed.

Laser/RX boasts of nice color graphs that represent various key performance indicators such as CPU utilization, disk I/O activity, memory management, etc. It requires a Vectra (at least a 386), Windows, a CD-ROM, and a mouse.

Vendor: Hewlett-Packard

Performance Management Tools

Q-Xcelerator Resource Manager

Q-Xcelerator is a CPU resource management tool that allows you to define unique scheduling queues apart from the default MPE C, D, and E subqueues. Workloads are then assigned to these special queues. A perpetually running background process ensures that high-priority processes receive the best response times. Other processes will suffer a bit more when there is CPU pressure.

Figure 13.1 illustrates a sample logical map for one system utilizing Q-Xcelerator. Typically, all batch jobs fall to the bottom of their scheduling queue and fight with one another for use of the CPU. Given the queue map in Figure 13.1, payroll batch processing would get the most attention, while weekly status reports would get CPU time when it becomes available.

All the branches of knowledge are connected together, because the subject matter of knowledge is intimately united in itself, as being the acts and the work of the Creator. John H. Newman

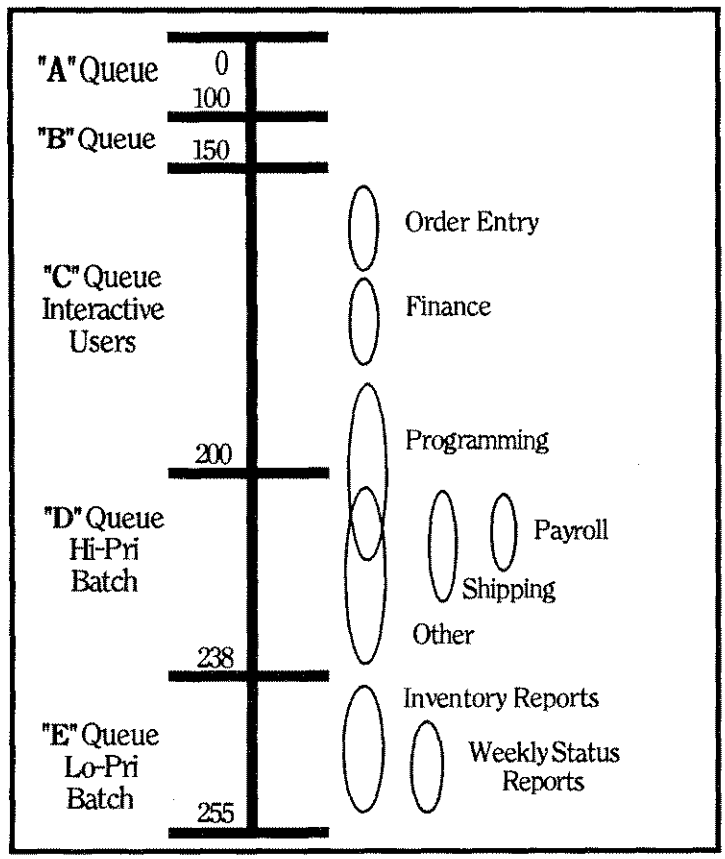


Figure 13.1 - Q-Xcelerator Sample Queue Configuration

Q-Xcelerator can be used to supplement the actions of the MPE dispatcher. The effective life of a system can be extended by "reshuffling" the resource deck.

Vendor: Lund Performance Solutions

Capacity Planning Tools

FORECAST/3000 Capacity Planner

This tool utilizes queueing network analysis to forecast future hardware needs and the effects of workload growth and modification. Forecast/3000 is the only tool of its kind commercially available for the HP 3000. Forecast/3000 takes the science of queueing theory and implements it in such a way that even a novice system manager can easily:

- Characterize workloads.
- Validate models.
- Determine the effects of upgrading hardware.
- Analyze the effect of program modifications on response times.
- Know when the system will run out of CPU and which model will be the best upgrade choice.

Forecast/3000 utilizes data gathered by its own host-based collector. Within the PC analyzer portion, it calculates and reports present and future system performance based on the estimated departmental growth of your business. Full color graphics allow you to predict such things as future hardware requirements, the impact of future applications, and changes to existing programs.

Based on input parameters regarding application workloads and business growth rates, Forecast/3000 allows you to address a variety of "what if" questions. Some of the questions this tool can address are:

- What will be the effect of upgrading to a larger system?
- Can an upgrade be postponed?
- What response service levels can be guaranteed?
- Will job completion deadlines be met?
- What will be the effect of a prototype application on the system once it is in production with its full number of users?
- How will a new payroll package affect our CPU usage?

Forecast/3000 produces numerous reports (graphic or tabular) that profile system performance metrics such as response time, throughput, CPU utilization, performance degradation, and transaction service time. Forecast data may also be exported to your favorite spreadsheet or graphics package.

Vendor: Lund Performance Solutions

RX Forecast

RX Forecast is an add-on tool for Laser/RX that allows future planning based on business growth. It utilizes data collected by SCOPE over a period of time. Trends are analyzed and extrapolation lines are drawn to indicate future CPU utilization. Confidence limits are drawn, and it displays line graphs.

Vendor: Hewlett Packard

Application Design Performance Tools

SPT/XL Software Performance Tuner

This tool is the greatly improved child of APS/3000 (Application Program Sampler), which runs on the Classic HP 3000. SPT/XL runs only on MPE/iX systems. It allows you to evaluate an application's design as a function of its resource requirements. This tool provides the ability to create a prototype application, measure key resource usage, make changes and see how you did.

SPT/XL provides the following information for a program:

- CPU usage.
- Wait times for disk, memory, software locks.
- File system and HP TurboIMAGE activity.
- Intrinsic use.
- Transaction response times and transaction counts.

The tool allows you to identify parts of an application's code that would benefit from tuning efforts. This way, before an application goes into production, you'll be able to quantify its resource requirements.

Vendor: Hewlett-Packard

MPE V Tools Not Available For MPE/iX

All of the tools I discussed in Volume I of Taming the HP 3000 are available for MPE/iX with the exception of the following:

- **SOO** (Son of Overlord) - Was not rewritten for MPE/iX, to my knowledge.
- **SURVEYOR** - Was rewritten for MPE/iX and became SURVEYOR XL, which then gracefully passed away with MPE/iX 2.1 (see tribute below).
- **TUNER** - System table monitor; not as necessary with MPE/iX.
- **LOSTDISC** - Was not rewritten for MPE/iX.
- **OPT/3000** - Was not rewritten for MPE/iX.
- **HPTREND** - HP dropped support of this for MPE V and does not provide this service for MPE/iX.
- **SURVEYOR XL** - Gone the way of the steam engine, as of MPE XL 2.1. The author of the Classic version of SURVEYOR wrote a version of the program for MPE/iX. Then he was conscripted into the deep, dark caverns of Hewlett-Packard: he will not be updating SURVEYOR XL. We have used this program many times to perform some initial "snooping around" in MPE/iX performance. The early releases of SURVEYOR XL included global CPU, memory management, disk I/O, and miscellaneous statistics. Later process information was incorporated by merging it with SCOUT, a process monitor. Since SURVEYOR XL depended on the measurement interface deep within MPE/iX, and since that interface has changed, SURVEYOR/XL no longer works on MPE/iX 2.1 and greater. A moment of silence...

MPE/iX Commands and Methods Helpful in Diagnosing Performance

SHOWPROC

This MPE/iX command became available as of release 2.1 of the operating system. It is helpful for a quick look at the processes on the system. Figure 13.2 illustrates an example of SHOWPROC.

```
PUB: showproc   ;job =@

QPRI  CPUTIME     STATE JOBNUM      PIN    (PROGRAM)   STEP

C152  0:04.909    READY       S4    46     :SHOWPROC   ;JOB=@
C152  0:15.457    WAIT        S2    48     :RUN SOS.PUB
B100  0:06.318    READY       S2       36     (SOS.PUB.LPS)
C152  0:03.978    READY       S2          63     (SOSLOGX.PUB.LPS)
C163  0:02.392    READY       S2            64     (SOSCHART.CHART.LPS)
D202  0:01.461    WAIT        J6    53     :SOSLOAD
D202  0:08.910    WAIT        J6       40     (SOSLOAD.PUB.HOG3000)
C154  0:25.151    READY       J6          65     (INTER1.PUB.HOG3000)
C153  0:24.871    READY       J6          66     (INTER1.PUB.HOG3000)
C153  0:24.257    READY       J6          67     (INTER1.PUB.HOG3000)
C153  0:25.939    READY       J6          68     (INTER1.PUB.HOG3000)
C154  0:24.315    READY       J6          69     (INTER1.PUB.HOG3000)
```

Figure 13.2 - SHOWPROC Output

Using PIN 46 as an example, some of the things you can tell about your system's performance are:

- A process' queue and priority (C152).
- The amount of CPU time used by the process (0:04.909 seconds).
- Its current wait state (READY).
- Its job (session) number (S4).
- It's PIN number (46).
- The last command issued (or program run) by the process (:SHOWPROC ;job=@).
- Any son processes (PIN 48 spawned PIN 36).

The command is similar in some ways to the old MPE V SOO (Son of Overlord) program and can be helpful for a quick look at process information.

DISCFREE

This command is especially helpful in tracking such things as disk space usage, virtual memory usage (transient space), and disk space fragmentation. Figures 13.3 and 13.4 show the outputs for the B and A options of the DISCFREE command. Below is a brief explanation of each tracked item, followed by a bit of exhortation concerning disk space management.

```
PUB: discfree b

DISCFREE A.01.03 Copyright (C) Hewlett-Packard 1989. All rights reserved
                        TUE, JUL 7, 1992, 4:05 PM

ALL MEASUREMENTS ARE IN SECTORS.

-------------------------------------------------------------------------

LDEV: 1 - - (MPEXL_SYSTEM_VOLUME_SET:MEMBER1)

            DEVICE SIZE : 2619776
            TRANS SPACE :  184704          PERM SPACE  :  1853936
        MAX TRANS SPACE : 1964832      MAX PERM SPACE  :  2252992

             FREE SPACE :  581136
AVAIL TO     TRANS SPACE :  581136  AVAIL TO PERM SPACE  :   399056
```

Figure 13.3 - DISCFREE "B" Output

DEVICE SIZE - Total disk capacity in sectors.

TRANS SPACE - Amount of transient space being utilized on this disk.

PERM SPACE - Amount of permanent space being utilized on this disk.

MAX TRANS SPACE - Maximum amount of transient space configured for this disk.

MAX PERM SPACE - Maximum amount of permanent space configured for this disk.

FREE SPACE - Amount of free space remaining on this disk. This value is calculated by subtracting TRANS SPACE and PERM SPACE from the DEVICE SIZE.

AVAIL TO TRANS SPACE - Amount of transient space available for use on this disk. This value is calculated by subtracting TRANS SPACE from MAX TRANS SPACE.

AVAIL TO PERM SPACE - Amount of permanent file space available for use on this disk. This value is calculated by subtracting PERM SPACE from MAX PERM SPACE.

One thing to note regarding the amount of space actually available for use is revealed in Figure 13.3. It shows that 581136 is available for transient space and 399056 for permanent space. Figure 13.4 tells us that total free space is 581136. You don't have the sum of 581136 plus 399056 available. The most you can use is 581136 for transient and 399056 for permanent. If you use 200000 more for transient, this would leave only 381136 (581136 - 200000) for permanent or transient space.

You may also find DISCFREE C, D, and E displays to be useful.

Let's examine the issues with disk free-space. The first and most obvious is that if insufficient space exists, applications will not run. Second, without adequate free space, users will not be able to logon and jobs will enter the WAIT condition seen occasionally on the SHOWJOB display. Keep in mind that with each logon, 10,000 sectors of transient space is set aside for memory swapping. In general, an ample amount of disk space is necessary for a healthy system. DISCFREE will help you monitor its usage.

Consider this comment from one writer:

> *"Sufficient free disk space available on the system is extremely important in reducing I/O demand. The way in which the file system manages free space and candidate locations for the placement of new file extents is completely different from MPE. If there is a lack of sufficient free space, the file system will have to work harder in locating candidate free space locations. The secondary storage manager must fulfill the requests demanded by the volume manager for placement of data on disk. The harder the file system has to work, the greater the delay to the process. The free space must be available to both permanent and transient structures. The free space must also be distributed to multiple disk drives. If the requested disk type is restricted to only one disk drive then all requests will be queued up and processed serially through that drive."[24]*

Figure 13.4 is a DISCFREE A listing. Use this display to monitor how fragmented disk drives become. While I have not seen any concrete evidence, disk fragmentation, in theory, can contribute to disk I/O impedance.

As an alternative, some people like the VOLUTIL SHOWSET command, which provides much of the same information in a different format.

He is poor whose expenses exceed his income. La Bruyere

```
PUB: discfree a

DISCFREE A.01.03 Copyright (C) Hewlett-Packard 1989. All rights reserved
                          TUE, JUL 7, 1992, 4:05 PM

ALL MEASUREMENTS ARE IN SECTORS.

-------------------------------------------------------------------------

LDEV: 1 - - (MPEXL_SYSTEM_VOLUME_SET:MEMBER1)

LARGEST FREE AREA :        310048     TOTAL FREE SPACE:      581136

     0 BLOCK (S) OF      1-     9  CONTIGUOUS SECTORS =       0  FREE SECTORS.  0%
   487 BLOCK (S) OF     10-    99  CONTIGUOUS SECTORS =   21056  FREE SECTORS.  4%
   149 BLOCK (S) OF    100-   999  CONTIGUOUS SECTORS =   37488  FREE SECTORS.  6%
    46 BLOCK (S) OF   1000-  9999  CONTIGUOUS SECTORS =  114656  FREE SECTORS. 20%
     5 BLOCK (S) OF  10000- 99999  CONTIGUOUS SECTORS =   97888  FREE SECTORS. 17%
     1 BLOCK (S) OF 100000- AND UP CONTIGUOUS SECTORS =  310048  FREE SECTORS. 53%
```

Figure 13.4 - DISCFREE "A" Output

Utilize the HPPA "Instrument Panel" to Measure CPU Activity

Few people realize that the second character from the left on the four character display panel actually indicates the level of system activity. This is measuring global CPU utilization. Here's the idea:

- F0FF indicates an idle CPU.
- F5FF indicates a CPU that is 50 percent busy.
- FAFF indicates a CPU that is 100 percent busy.

Although you should not come to any substantial conclusions, this display will tell you a few important things. If users are complaining about response times for example, and the display was indicating F2FF, you could safely conclude that the CPU was not the bottleneck. It could be excessive CPU pause for disk I/O, or perhaps an application impede problem, but certainly not due to a lack of CPU. On the other hand, if the display reads F9FF and users are whining, it could very well be that the CPU is the culprit.

Conclusion

While HP has provided a few more "free" ways to monitor some aspects of performance, in order to be proactive in performance management you will need to implement a monitoring plan that covers all the bases and angles I spoke of in Section One. This will involve tools. Such tools are available now for HPPA systems that have more flexibility and features due to more information being available and a lack of MPE V architectural limitations.

? Questions/Discussion ?

1. Refer to the Questions/Discussion box at the end of Chapter 7.

Notes:

The king's heart is like channels of water in the hand of the Lord; He turns it wherever He wishes. Proverbs 21:1

U.S. students finished near the top internationally in one category - the number of hours that 13-year-olds spend watching TV, second only to Scotland. U.S. students once again ranked near the bottom in the most comprehensive test ever to compare their math and science skills with those of other countries. In science, U.S. 13-year-olds finished ahead of only Ireland and Jordan among 15 countries rated. In math, U.S. 13-year-olds were second from the bottom among all countries, according to the Educational Testing Service.

Chapter 14

Managing MPE/iX Performance- Tools and Techniques

Throughout this book I have endeavored to highlight performance features peculiar to MPE/iX. In this chapter I cover some ways you can actually manipulate the behavior of your system's performance. This is not an exhaustive list. Appendix C contains ideas, from Volume I of Taming the HP 3000, which are still very much applicable to HPPA systems. In fact, over 75 percent of all the ideas presented there are, in some way, useful in managing the performance of an MPE/iX-based system.

I must say that you should approach any system tuning with caution, since it is possible to negatively affect certain workloads over the entire system. It is good to experiment, but if you notice any erratic behavior, simply undo what you did (use the TUNE command to re-implement the prior values you changed).

"Knobs and Dials" for Tuning MPE/iX Performance

Here, I present some ways to alter the system's performance from a CPU scheduling standpoint. This mostly involves priorities, so a refresher reading of Section One will help you understand the basis of some of the ideas that follow.

Utilize TUNE Min/Max to Favor or Penalize Applications

The min/max parameters of the TUNE command allow you to give CPU favor to applications that might otherwise not receive enough. If you wanted to penalize CPU-intensive applications a bit more, you would decrease the numeric value of the min/max pair to something like:

```
TUNE;CQ=152,200,0,100
```

The "0,100" basically says that any transactions that use more than 100 milliseconds of the CPU's time before completing will be knocked down in priority. Basically, the hogs fall faster! More CPU will be available for other, perhaps higher priority, users. This forces the System Average Quantum (SAQ) value to be frozen at 100, unless the actual SAQ value is calculated to be less than 100. If that were the case, the SAQ would be the actual value computed by the system.

To cause a more equal sharing of the CPU by processes regardless of whether they are CPU-intensive, simply raise the min/max pair to something like:

```
TUNE;CQ=152,200,500,1000
```

This would force the SAQ value to be 500 unless the actual system computed SAQ falls between 500 and 1000. This technique allows processes that would otherwise be penalized greatly by the dispatcher to have an equal opportunity at getting CPU time.

Alter Queue BASE and LIMIT Boundaries to Provide Better Response

If you would like to provide more of a boost to batch applications or penalize some interactive hog programs, this technique could be quite useful. Remember that processes start their lives at the top of their queues and then decay down to a worse priority position. Therefore, it stands to reason, that if you could make some processes fall farther than their default limit or start higher than their default base, you could change the behavior of your system.

To provide worse response for unruly interactive processes (thereby giving other users better response), simply perform:

```
TUNE;CQ=152,238
```

This would cause CPU-intensive CQ processes to fall to the same level as batch jobs. At the bottom of the DQ (238), both batch and any interactive processes sent there will fight for CPU time. At the same time, CPU-frugal CQ processes will have more of a chance to get adequate CPU time.

To give batch jobs a "shot in the arm" so they will complete faster (possibly harming online applications in the meantime), simply enter:

```
TUNE;DQ=160,238
```

This would overlap the DQ's base to the middle of the CQ. Batch jobs may get a temporary boost in performance, assuming there were processes using CPU at the bottom of the CQ.

Relocate the EQ for High Priority Batch Jobs

One phenomenon observed by many on MPE/iX systems, is queue starvation. Since I/O bottlenecking is infrequent on MPE/iX, interactive user processes will not have to wait as long for disk data retrieval. While this is a blessing on one hand, it can be disastrous for batch jobs. On MPE V, batch jobs receive CPU time while interactive processes wait for disk I/Os to be fetched. In an MPE/iX environment, it is possible that CQ processes could monopolize the CPU, causing batch jobs to starve.

One remedy for this is to relocate the EQ itself to within the CQ's boundaries. One way to do this might be:

```
TUNE;EQ=156,160
```

After redefining the EQ in this manner, you simply launch important jobs with the PRI=ES parameter or utilize ALTPROC or a commercial tool to relocate the target job from the DQ to the new EQ. This technique allows important batch jobs to be second only to very high-priority CQ processes. The best setting can be determined by experimenting with various values of the EQ's base and limit.

Utilize the Dispatcher Oscillate Boost Feature to Give Hog Processes a Second Wind

A CPU-intensive process will fall to the bottom of its queue. This can cause the process to become CPU-starved. A relatively new feature of MPE/iX is to give such processes more of a chance at some CPU time. Choosing the OSCILLATE option for the TUNE command (vs. the DECAY

Often, small children prefer a touch to spoken words. A touch tells them, "I'm available if you need me. I care about you." G. Smalley

option) will cause a process to bounce back up to the top of the queue and start over again. This can provide some temporary, but recurring relief for some Hog processes. But be aware because a CPU intensive (serial read) batch job can oscillate rapidly and take over the queue that it's in. Be careful!

Use ALTPROC or a Tool to Alter a Process' Priority or Queue

Occasionally, you may wish to move a process to another queue or to a fixed priority position. You may utilize the ALTPROC command or a commercial monitoring tool to accomplish this. In effect, you are saying that you wish to bypass the initial queue assignment for that process or change the dispatcher's treatment of a certain process. You'll need OP or SM capability to use this command.

Other Improvement Tools and Techniques

Upgrade to Fiber-Optic Disk Drives

Unless you are experiencing *a lot* of I/O activity, this idea will not buy you much in the way of performance. Fiber-Optic disks have three benefits:

- The data transfer rate is five megabytes per second versus one megabyte for HPIB devices. Remember, though, that because of the extreme elimination of physical disk accesses, this is rarely a problem. Even if it is a problem, it would be indicated to perform different I/O optimization techniques.

- The distance limitation is expanded to 500 meters versus 15.

- You are allowed eight disks per device adapter versus six for HPIB. Keep in mind that data transfer time accounts for usually less than 10 percent of the total time to complete an I/O. The majority of the time is still due to disk seek and latency time. Nevertheless, if any of the above limitations are holding you back from adequate disk I/O throughput, you might consider fiber-optic as a solution. Before you make a purchase decision I encourage you to engage a consultant to ensure your decision is right.

Utilize a RAMDISC for I/O Intensive Applications

Since I reviewed the Kelly RAMDISC in Volume I of Taming the HP 3000, there have been some significant improvements in this product. A RAM disk is essentially a large quantity of extended main memory with software intelligence that allows you to place key data files in it. The access times are at the same speed as other memory accesses, which means it can be very, very fast.

This product can shave a significant amount of time off of jobs or user response times if the bottleneck is disk I/O related. All of the initial concerns I had regarding RAMDISC have been remedied since I first beta-tested it back in 1987.

Vendor: Kelly Computer Systems

TurboIMAGE Data Base Performance

First off, I must say that Hewlett-Packard has done a great job in causing TurboIMAGE performance to "scale with the HPPA architecture." This means that as systems get more CPU horsepower and memory, these bottlenecks are lessened, while the potential for an application or data structure bottleneck increases.

The potential problem with TurboIMAGE (developed initially in the early 1970s) was that it would not be able to keep up with the new gains in systems performance, such as with HPPA. This is due to the "serialization" of TurboIMAGE. This is a problem due to the need for a deadlock-free environment. To accomplish this, certain internals of IMAGE need to be locked while performing disk I/O. The shear power of the HPPA systems only exacerbated the serialization problem.

Hewlett-Packard performed a number of surgeries on TurboIMAGE in order to provide better throughput on the higher-end MPE/iX systems. Some of these improvements included:

- Reduced path length.
- Reduction of Disk I/O traffic.
- Improved concurrence of data access.
- Ported to MPE/iX native mode (release 1.X).
- Integrated with transaction manager (1.X).
- Improved concurrence of buffer access (1.X).

- Added mapped file access to data sets (2.X).
- Reduced lock area serialization (2.X).
- More intelligent buffer manager (2.X).
- Minimized semaphore avoidance (2.X).
- Increased lock area size (2.X).
- Increased buffer area limits (2.X).
- Hashed access to buffer area (3.X).
- Improved lock area manager (3.X).
- Reduced DBPUT/DBDELETE semaphore contention (3.X).
- Reduced DBPUT/DBDELETE semaphore hold time by I/O pre-fetch (3.X).
- DBPUT page prefetch. [25]

All this to say that HP has kept up TurboIMAGE performance quite well. You may want to review some of the tuning recommendations discussed in Volume I of Taming the HP 3000 (Appendix C). Then keep in mind that much has changed for HPPA systems.

The age-old housekeeping recommendations for TurboIMAGE on Classic systems have not changed. These include:

- Re-packing your detail sets when indicated.
- Keeping a close eye on master set metrics (secondaries and inefficient pointers).
- Making sure the indicated primary path for details is the best one for performance.

As I stated in Volume I, tools such as DBLOADNG, HOWMESSY, DBGENERAL, ADAGER and FLEXIBASE will help you diagnose and maintain good performance in these areas. I must say, however, that I have not seen many MPE/iX problem cases that were due to data base housekeeping neglect. But the ones I have seen were pitiful. You see, the awesome ability of MPE/iX to eliminate I/O problems can lull us into thinking that all is OK with data base performance. Really, the day of reckoning does not come as soon as it does on Classic systems.

Following are a some specific recommendations regarding TurboIMAGE performance.

Utilize the Pre-fetch Feature for TurboIMAGE

Prior to 3.0 of the MPE/iX (MPE/XL) operating system, data pre-fetching was not performed on behalf of TurboIMAGE databases. This was a significant performance disadvantage in most cases. Enabling this feature (through DBUTIL.PUB.SYS) allows data to be retrieved in larger quantities. One benefit, as noted in preliminary testing, is that PIN-boosting occurs less often with pre-fetching enabled. This is when low priority process, which is holding a resource that a higher priority process needs (like a data base lock), is temporarily boosted to that higher priority in order for it to get enough CPU attention to release the lock. With TurboIMAGE pre-fetching enabled, this seems to occur less frequently. This is a great help to a potentially nasty problem since this is precisely how batch jobs can negatively impact interactive users.

It also seems that the only time pre-fetching hurts is when there is a single batch job running. The CPU and memory required to manage the pre-fetch overhead can be better spent by turning off this feature.

Without extensive testing, I cannot conclude all the up- and down- sides of this feature, but my initial impression is that it will benefit heavier write intensive applications that all attack the same data bases. You will definitely want to experiment with this new TurboIMAGE attraction.

Use Data Set Numbers Instead of Names

When programming TurboIMAGE applications, Evan Rudderow has found that using data set numbers instead of names utilizes 11 percent less CPU when accessing such sets. This can be significant for large applications.

Design with Data Locality in Mind

In general it is vital that you design data bases to have optimal data locality. With reference to other discussions regarding locality in this book, this means not only micro locality (data within a data set), but also macro locality (multiple data bases vs. few). With the gains in TurboIMAGE performance outlined above, it will, in many cases, be better to have one large data base rather than multiple data bases to accomplish the same task. This is primarily because more than data is shared when multiple pages of data are retrieved by the operating system. So, with one large data base it is possible that other data structures could be shared (disk I/O being avoided) with each page of data brought in.

Conclusion

Although a number of things can be done to improve the performance of an MPE/iX system, HP has made it easy and difficult at the same time. By attempting to shift all bottlenecks to the CPU or memory, to get better performance out of your system, you simply put your money down and upgrade a hardware component. Even though this is true for many situations, there still remains much that can be done to "tune" an MPE/iX system. Some of the new TUNE command options attest to this.

While disk I/O will not be a significant concern for most cases, it still needs to be addressed or at least diagnosed periodically. More than ever, it is important to plan effectively for future upgrades. This is why the subject of capacity planning using queueing models has garnered my attention over the past few years. If getting more hardware is the primary way to improve performance, then we had better focus on planning—capacity planning that is.

Nearly all men can stand adversity, but if you want to test a man's character, give him power. McKenzie

? Questions/Discussion ?

1. Do you have multiple workloads on your system where some are high-priority and some are lower? If so, does it make sense from a performance and political perspective to utilize the TUNE command to favor one or more groups over the others?

2. Are there times on your system where batch jobs are "starved"? If so, can you see the benefit in overlapping the C and D queues? To what extent will this help?

3. When would ALTPROC (priority altered) be a dangerous tactical manuever?

4. Which bottleneck(s) do you think are prominent on your system? If none, which do you think will be first?

Chapter 15

MPE/iX Performance Case Studies

In this chapter I illustrate some of the important performance monitoring issues with some disguised but real-life MPE/iX case studies. This material will familiarize you with some system performance situations others have encountered. Once again, I remind you that in order to get the most out of these examples, you'll need to become comfortable with the material presented in Section One.

CASE 15A: Could Someone Please Turn off the Switch(es)?

This Series 955 MPE/iX system was really being pushed. The manager was caught between the software vendor and his management. The system just wasn't able to keep up with the demands placed on it. Figure 15.1 illustrates its burden. One thing you'll immediately see is the absence of a CPU label for idle time. That's because there wasn't any! But if you remember my comments in the textbook section, this alone is not a definitive indicator of CPU bottlenecking.

Paucity of CPU pause time and CPU busy on memory management are also worth noting. Overhead, combined with AQ and BQ processing, was moderate (averaging about 20 percent). The system did not exhibit memory shortage (note the small amount of CPU time spent on memory management), which was concluded by looking at other primary memory pulse points.

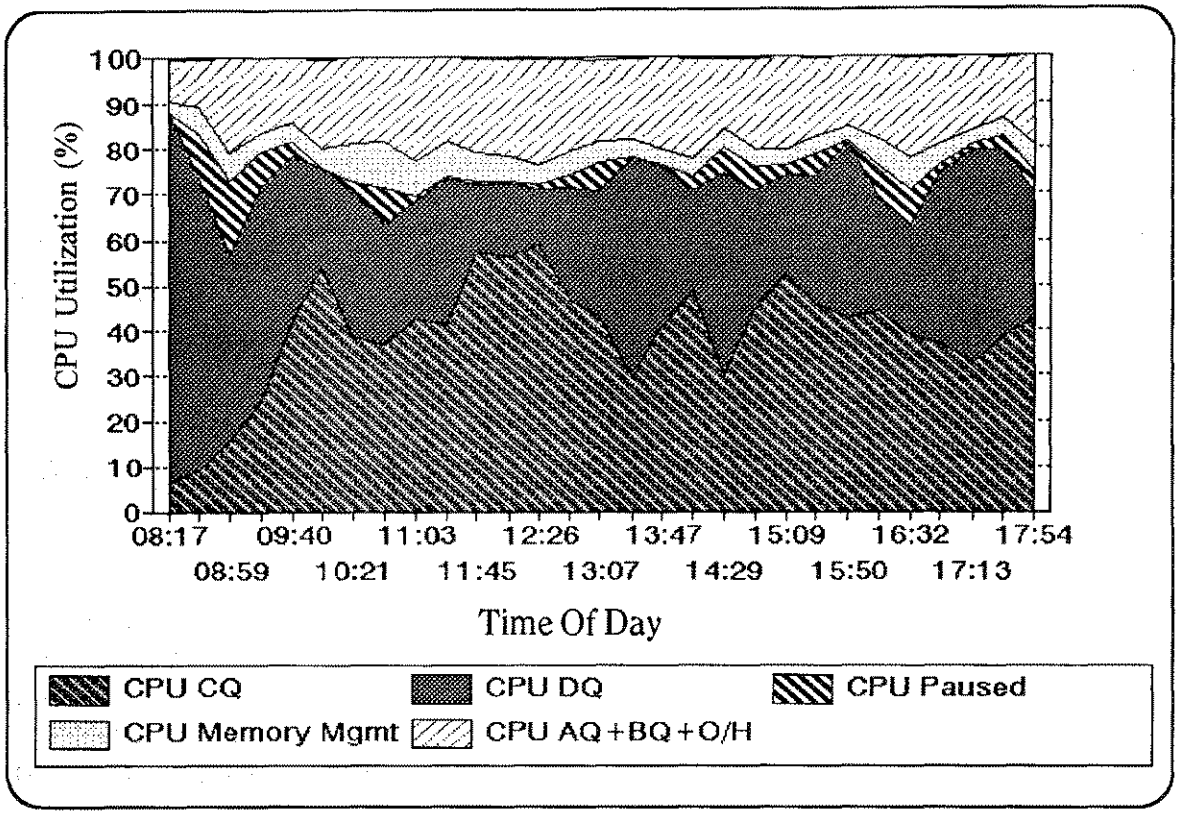


Figure 15.1 - CPU Utilization for Busy Series 955

Figure 15.2 helps to zero in on the heart of the matter. You'll notice throughout the day the incredibly high CPU queue length. Remember, this means that, on average, there were over 15 processes waiting to get CPU service time.

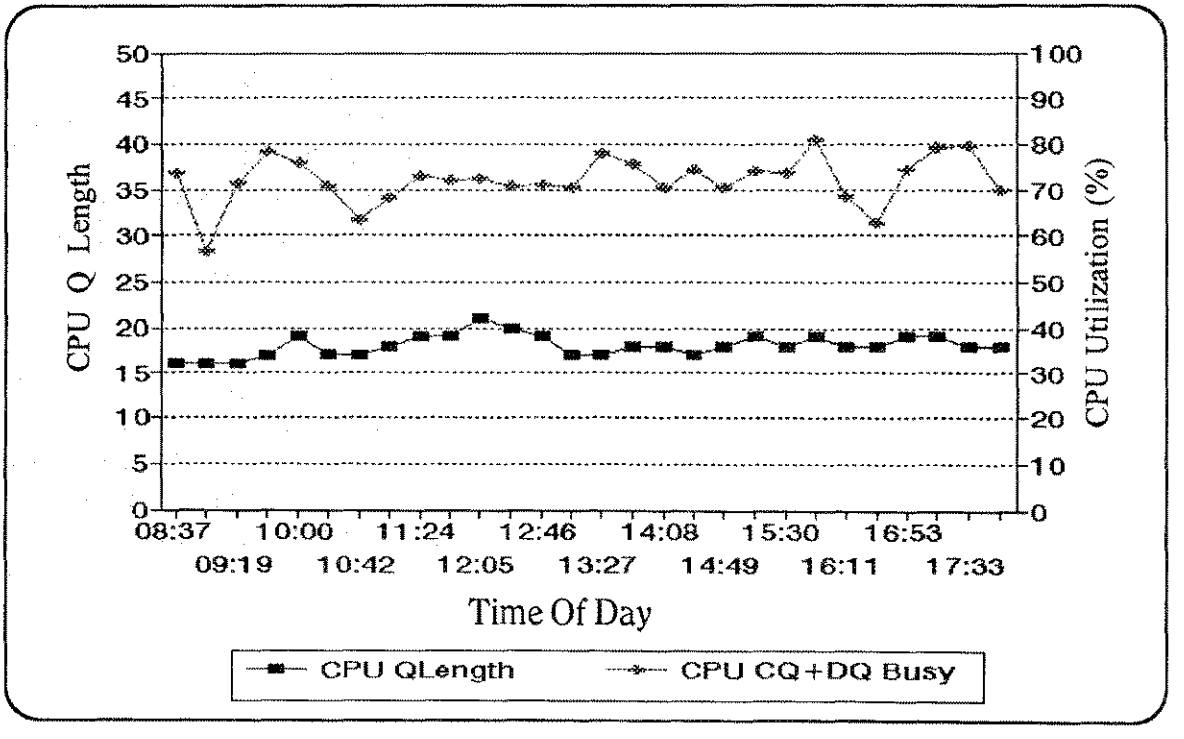


Figure 15.2 - CPU Busy on Processing Contrasted with CPU Queue Length

Needless to say this is certainly a red-zone CPU case. The remedies for this situation are few: use less CPU (write more efficient code), spread out the demand (staggered user/batch shifts), or buy more (upgrade and/or buy a second system).

Figure 15.3 sheds a bit of light on some of the high CPU usage. This chart shows how many compatability to native (C/N) mode switches were being performed per second. The amount shown is not crippling (especially since the switches are not native to compatibility ones), but it certainly did add a fair chunk of overhead to the system. This situation could be improved by converting the applications to native mode.

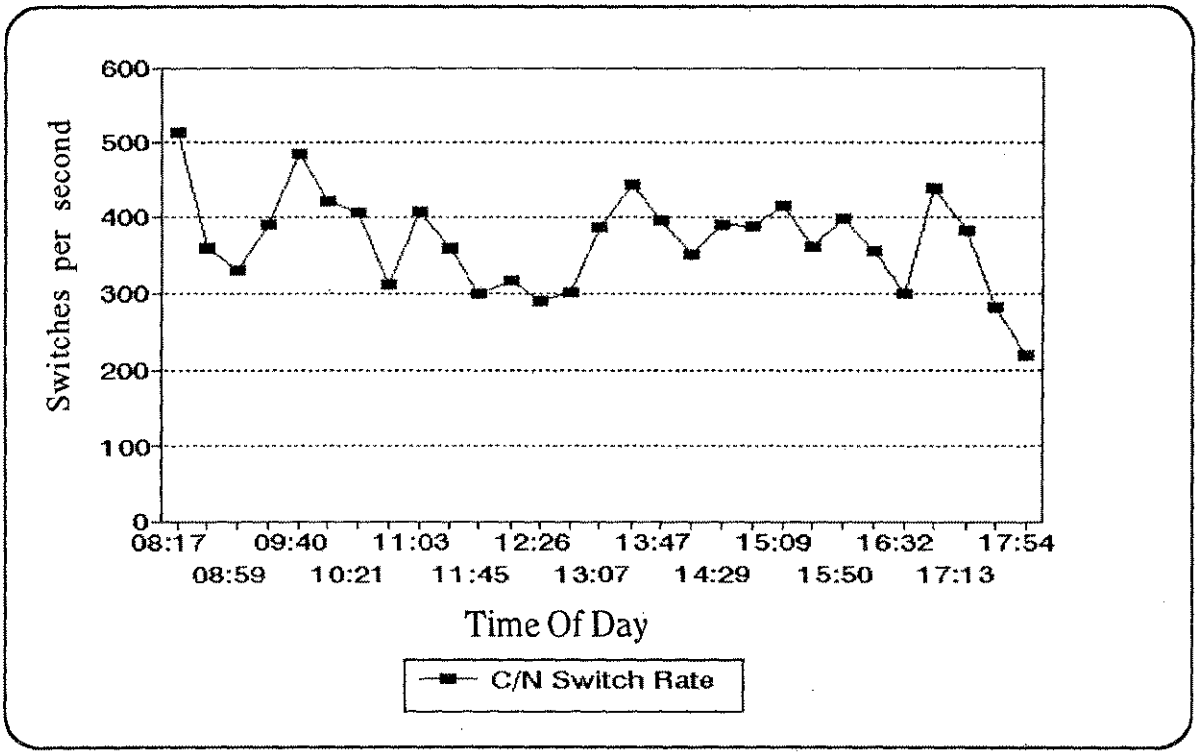


Figure 15.3 - Compatibility Mode Switches

He who leaves God out of his reasoning does not know how to count. Italian Proverb

CASE 15B: Biting the Bullet and Upgrading

Figure 15.4 presents us with another CPU pressure situation. A few observations show this system was causing poor user and batch responses. First of all, the amount of time this Series 925 spent on interactive processing was averaging at or over 50 percent throughout the day. This, coupled with 4 to 10 percent memory management and 15 to 20 percent AQ+BQ+Overhead, pushed the CPU well near (at times over) the 85 percent saturation mark. Although there was a bit of idle time, when a system approaches 85 to 90 percent busy on high-priority processing, response times degrade rapidly.

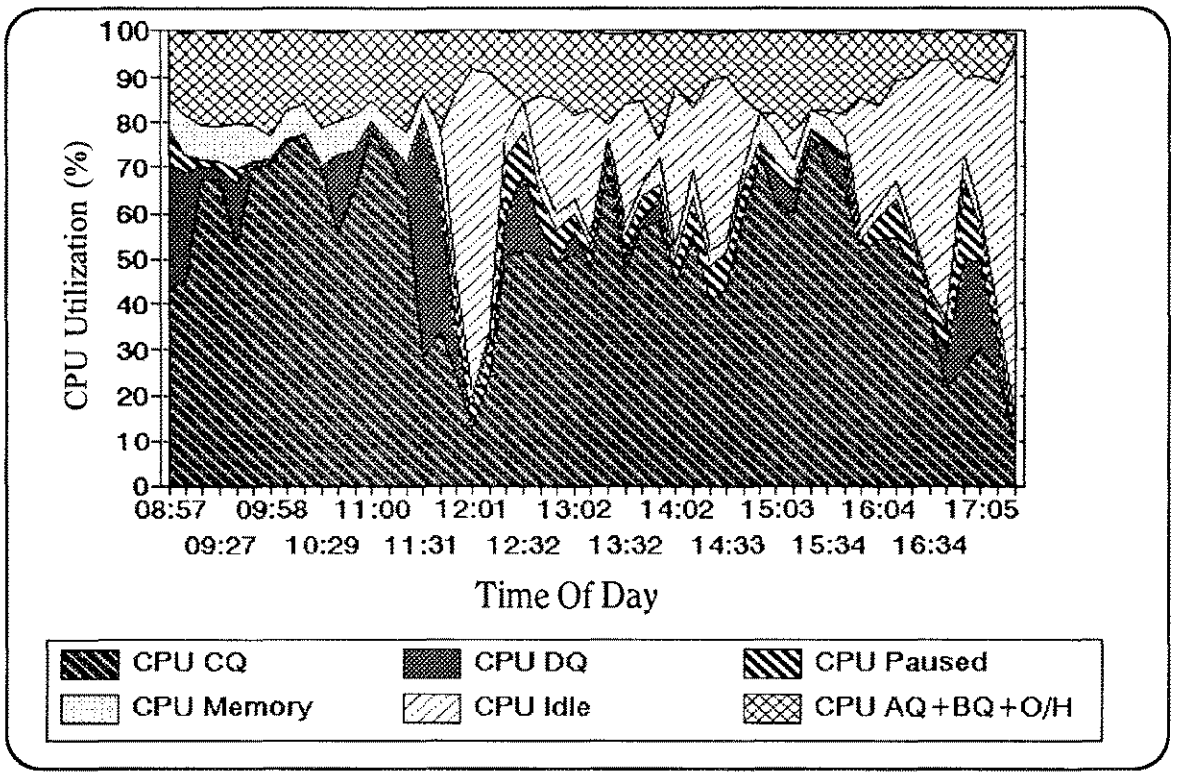


Figure 15.4 - A Busy Series 925

Figure 15.5 shows the upgrade to a 935 on a typical processing day (with a bit more memory, and 45 users compared to 30 on the 925). It is apparent that the amount of CPU spent on interactive processing is considerably less; memory management time was also less (72mb vs. 48mb).

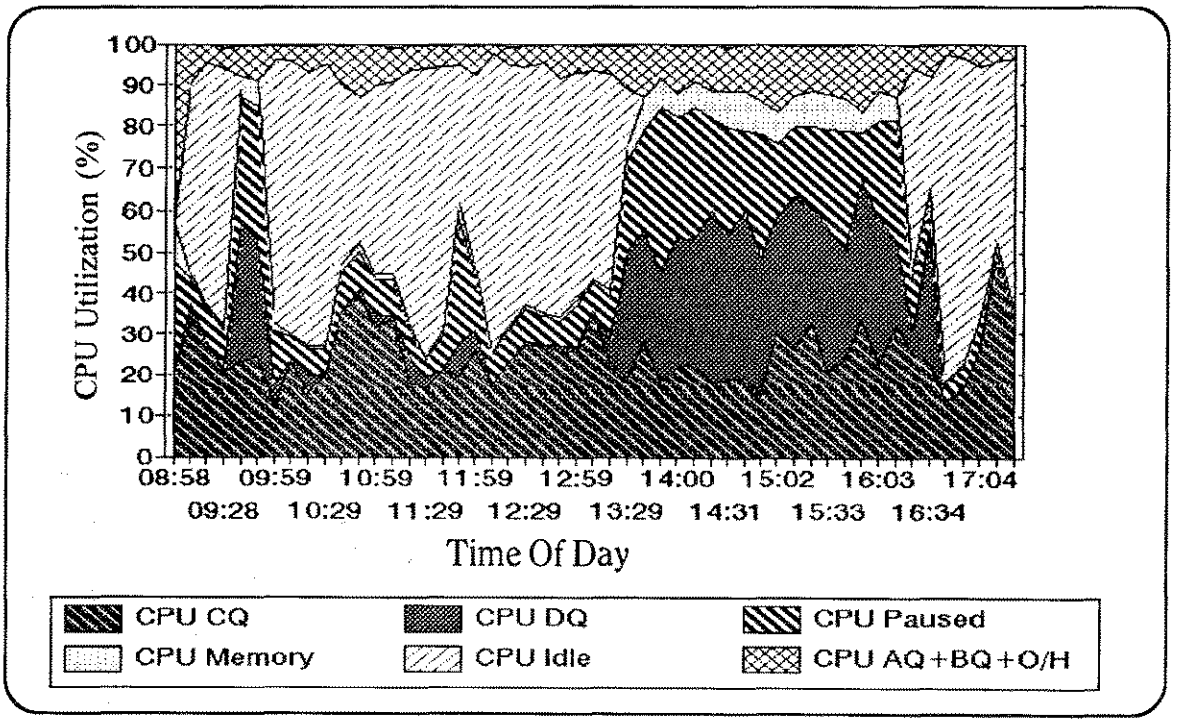


Figure 15.5 - The "Un-Busy" Series 935 (upgraded)

The chart shows noticeably more idle time and CPU overhead is less. The dark period between approximately 13:00 and 16:00 represents some batch processing. This job apparently accessed some data files, which caused a large amount of the CPU' s time to be wasted waiting for data to become available (CPU paused).

CASE 15C: When the Read Hit% " Hits" the System Hard

This situation illustrates what happens when data becomes difficult for the disk drives to round up efficiently. In earlier chapters I discussed the relationship of the Read Hit% with the amount of time the CPU has to wait for data to become available. Figure 15.6 demonstrates this.

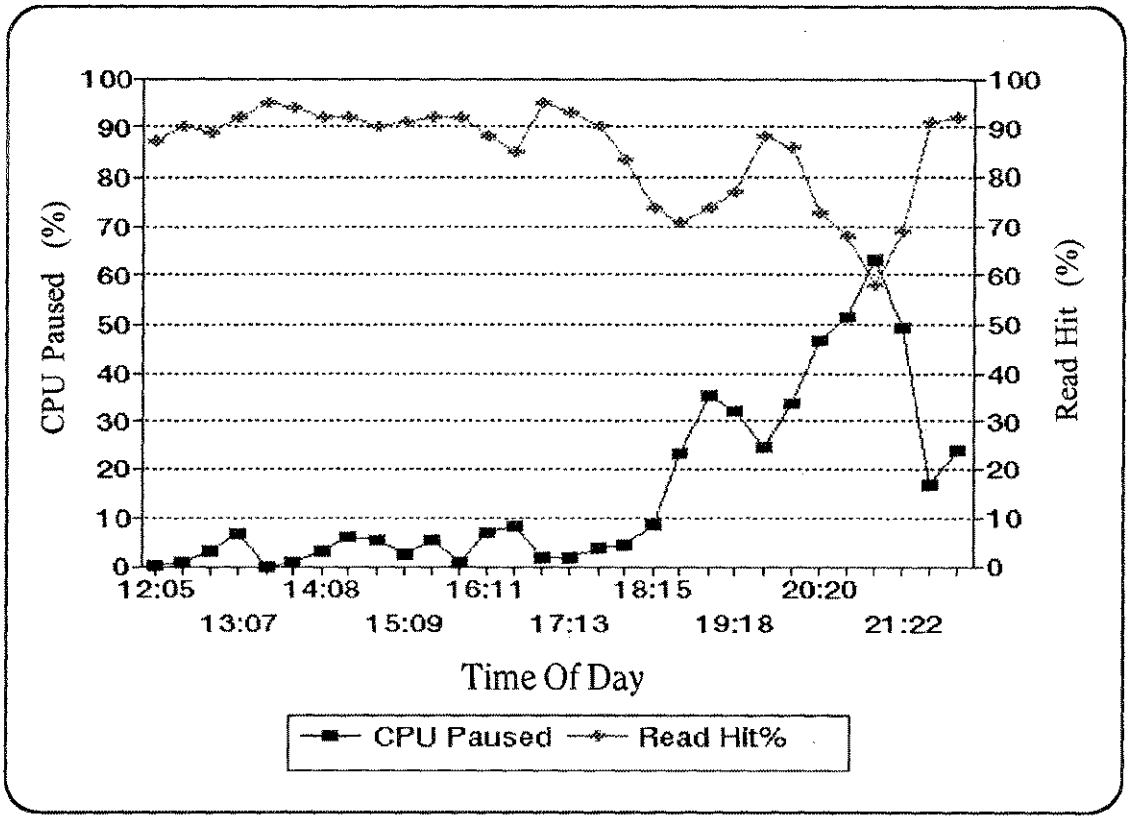


Figure 15.6 - The Inverse Relationship of Read Hit% and CPU Pause for Disk

You' ll notice the nearly perfect inverse correlation between these two values. When disk data becomes difficult to access in a timely manner, both the CPU and user response times suffer. This is usually a result of data base records becoming fragmented, performing backward chained reads, etc.

Needless to say, the period of time at approximately 21:00 must have been interesting in terms of response times!

Even if you came from a home where affection was absent, you can change. You can be a "person of blessing" in your own home by deciding to provide meaningful touch. G. Smalley

Figure 15.7 provides another view of the situation. Right after 20:20, this graph shows something that could be considered confusing. The I/O read rate is high but the disk queue length is fairly short. Perhaps this points to a long data transfer time and possibly some I/O channel delay (although it is tough to tell from the data given).

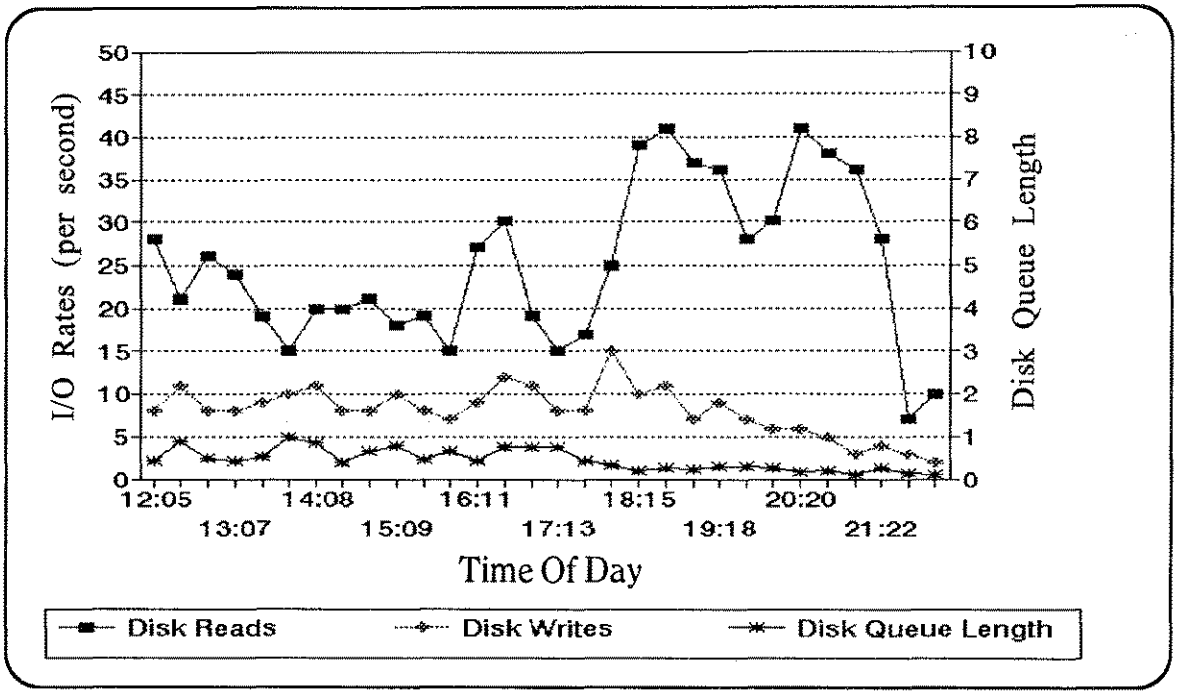


Figure 15.7 - Disk I/O Activity and Average Queue Length

In any case, it is worth monitoring the Read Hit%, CPU pause, and Disk I/O rate values. These, along with other disk indicators, will help you understand the conditions under which your system is choked on I/O activity.

CASE 15D: The Memory Bottlenecked One User Series 920

This case study could be funny if it weren't a real situation! It is a great classroom example of what happens to a system when memory is drastically underconfigured.

Figure 15.8 shows nearly a full day's worth of activity. Mind you, there was just a single user on this system. Notice when the CPU became busy (the user was doing something), the CPU pause time skyrocketed! Does this mean a disk I/O bottleneck? Maybe yes, maybe no. This illustrates how a lack of memory can give the initial appearance of being an I/O problem.

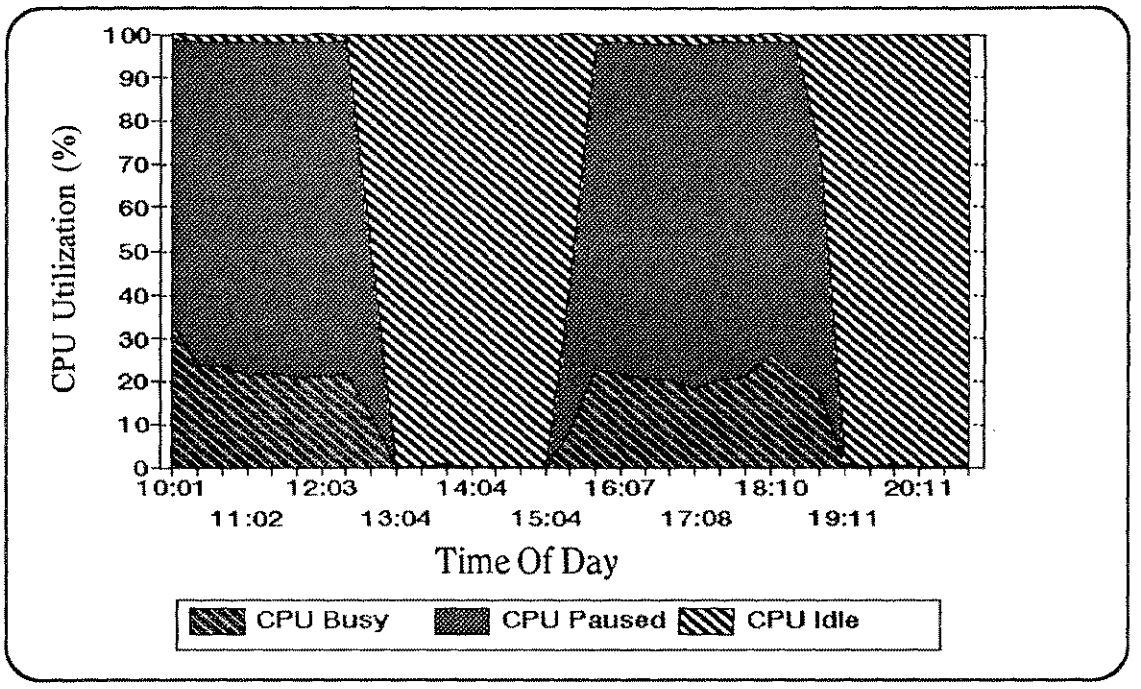


Figure 15.8 - Radical CPU Pause for Disk on a 24mb Series 920

The real question for this situation was, what is the causative agent? Was it I/O or memory? Figure 15.8 alone cannot answer this question.

Figure 15.9 displays a Read Hit% that we would expect to see given the nearly 80 percent CPU paused-for-disk condition. Have you ever seen a 5 percent Read Hit value? I had not... until this pitiful example!

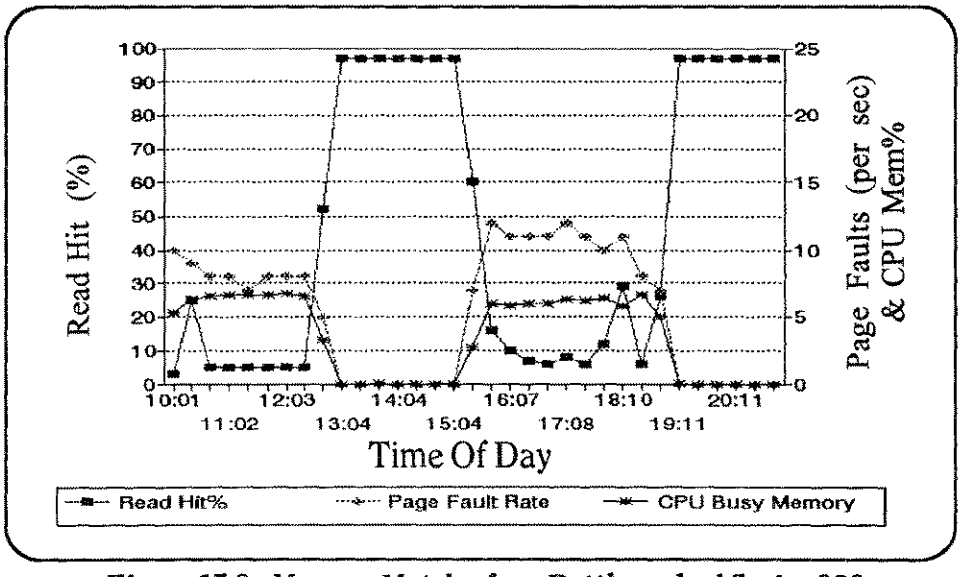


Figure 15.9 - Memory Metrics for a Bottlenecked Series 920

The page-fault rate for this system is not good. Although it is difficult to quantify, some of the pulse point values vary depending on the system size. My suspicion is that both the page-fault and memory-management values are quite indicative (on this small system) of a bad memory shortage.

Of all the tyrants the world affords, Our own affections are the fiercest lords. Earl of Sterling

CASE 15E: The Disappearing CPU Pause Act

This Series 935 illustrates a concept I mentioned in Section One regarding CPU pause-for-disk time. Figure 15.10 is really quite boring if you think about it. The system is virtually 100% busy for 24 hours! No pause time and no idle time here.

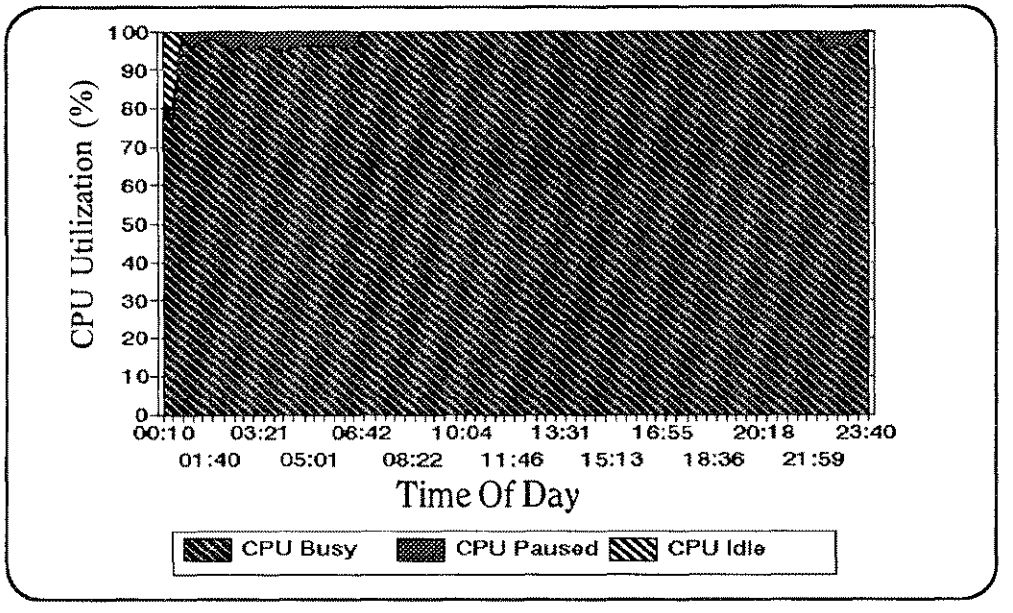


Figure 15.10 - A Boring, but Busy, Series 935

We really must ask ourselves the next question when faced with such a wall of CPU activity: Just what is the breakdown of the CPU's activity? Figure 15.11 helps to answer this.

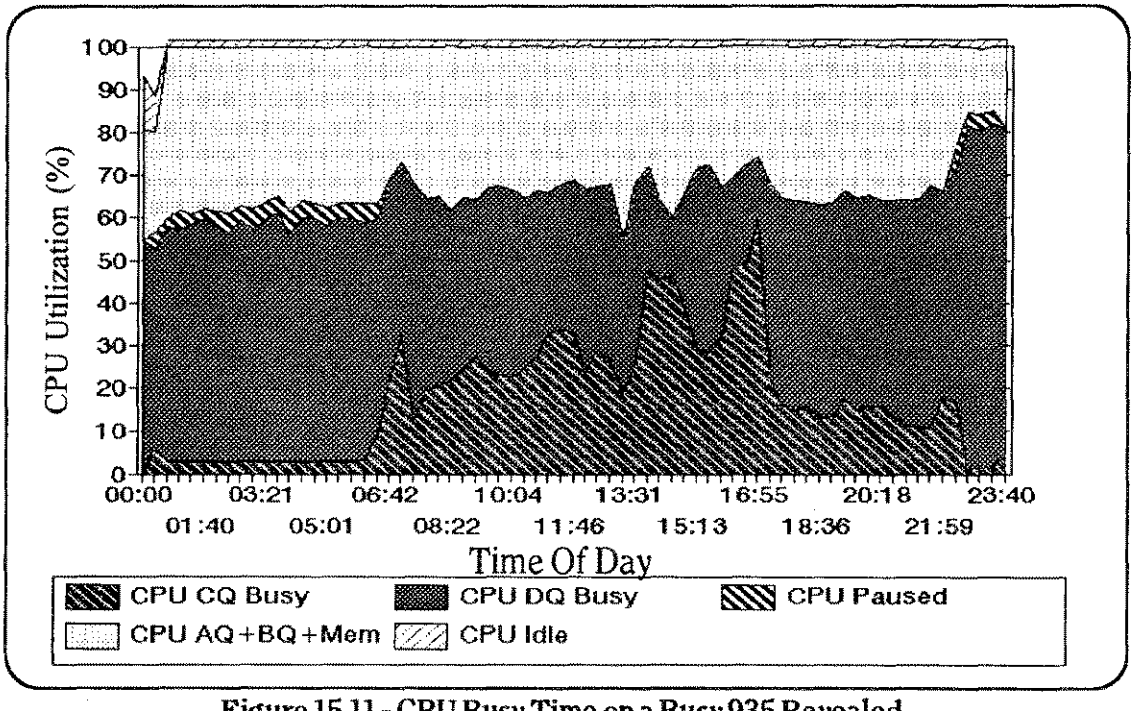


Figure 15.11 - CPU Busy Time on a Busy 935 Revealed

He who walks with wise men will be wise, But the companion of fools will suffer harm. Proverbs 13:20

Three observations come to mind:

- The DQ seems to be taking most of the CPU's time.
- There is a fair amount of CPU overhead.
- The CQ is really quite frugal in its CPU consumption.

The lack of CPU pause-for-disk time is almost scary! If you regularly monitor your system, you'll often see the CPU pause time jump up when certain batch jobs are active. Refer to Figure 15.5 in Case Study 15B (15:00 and following) for a common example. This often related to the fact that batch applications many times do not read data from data bases by efficient (primary) paths.

Does the absence of CPU pause time mean that there is absolutely no problem with disk I/O? Perhaps. The only way to know is to confer with data such as is found in Figure 15.12.

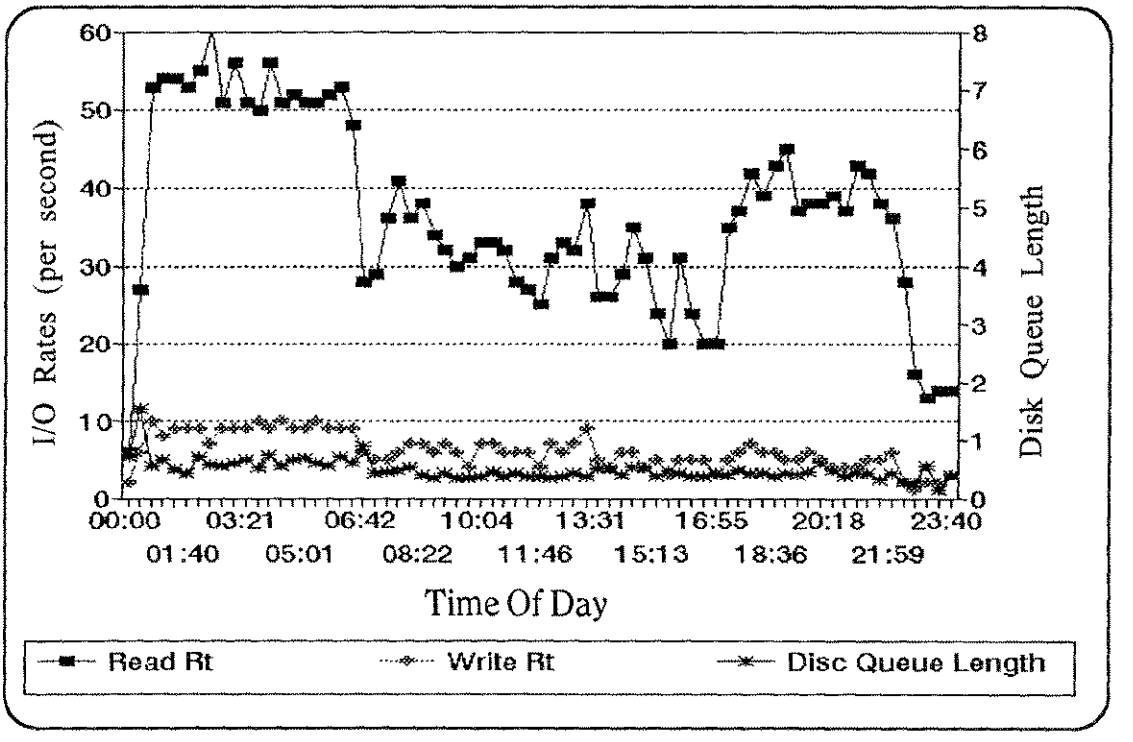


Figure 15.12 - Supportive Disk I/O Indicators for a Busy Series 935

While I/O rates in Figure 15.12 are not hideous, they indicate an impact on the system. The average system disk queue length hovers between .5 and 1. This is inherently a yellow condition for disk activity. Write activity was not substantial.

It so happens that the technical support team at this site had the habit of streaming jobs until the CPU pause time went away! While I haven't given it deep, ponderous thought, the idea strikes me as a positive one, unless of course other indicators point to compromised service for the customers of the system.

Jean Paul Sartre, a life-long atheist philosopher, said at the close of his hedonistic life: "I do not feel that I am the product of chance, a speck of dust in the universe, but someone who was expected, prepared, prefigured, In short, a king whom only a Creator could put here; the idea of a creating hand refers to God."

Based on the data in Figures 15.10, 15.11 and 15.12, we can conclude the following:

- This system was busy, but perhaps not out of gas. We would need to ask the manager how much of the batch activity is considered vital and high-priority (could not sustain a substantial increase in completion times without sacrificing service level agreements--remember those?).

- There was fairly substantial I/O pressure observed throughout most of this day, even though CPU pause for disk is absent.

CASE 15F: Some General MPE/iX Operating System Weirdness

Figure 15.13 reveals a fascinating aspect of MPE/iX performance prior to Release 3.X. First, if you ever see the AQ CPU activity reach 47.2 percent, you have permission to flip out! Even if this only represented a 30-second snapshot of the system, average AQ busy at this level means at least one thing: from the BQ down to the EQ, life stopped on planet Earth! Look closely at the SOS/3000 display and notice the only process being shown is PIN 9. This process consumed the majority of the AQ CPU time. It did so because it needed to post data from transaction manager logs to target data base files.

```
SOS/3000 A.01 (C) LPS       WED,MAR 6, 1991, 12:43 PM   E: 04:40:12  I: 00:31
-------------Global CPU Statistics--------------------Global Misc Statistics----
             TOTAL BUSY:  90.5 [ 70 ]                #Ses 119  #Job 3  #Proc 587
AQ 47.2[ 2]  Memory  3.8[ 7]     CPU QL   9[ 3]      CM to NM Switches  9[ 47]/s
BQ  3.1[ 5]  Dispatch  .2[0]     Launch/s 55[78]     NM to CM Switches  4[ 18]/s
CQ 23.1[27]  ICS/OH 11.4[10]     CPU CM%  1[ 5]      Transactions 533[270K](1001)
DQ   .0[ 0]  Pause  9.5[30]      SAQ       11        Avg First Resp   .<[  .0]
EQ  1.7[18]  Idle     .0[ 0]                         Avg Prompt Resp   .2[ .1]
-------------------------------Global Memory Statistics--------------------------
Page Fault Rate  5[11]/s  Memory Cycles  0[ 53]     Overlay Rate  0[  3]/s
Lbry Fault %     1[ 2]    Read Hit %     82[84]     Swap/Launch  .36[ .49]
-------------------------------Global Disk Statistics----------------------------
<Ldv>Rt/IO%QL <1> 1/ 3/6.46  <2> 1/ 1/9.99 <3>  1/ 2/9.42  <4>  1/ 3/9.99
<Ldv>Rt/IO%QL <5> 1/ 3/9.99  <9> 5/ 10/5.11 <11> 3/ 5/ .08  <12> 3/ 6/ .11
-----------------------------------Process Information-----------------------------
PIN  J/S# Session/User Name   Cmd/Program CPU %  QPri   #Rd   #Wr   LDV   #Tr   PRes
 9  <SYS> <System Process>                44.2   AL30    0   983    -     0     -
```

Figure 15.13 - Bizarre Transaction Manager Activity

To call this figure bizarre is really not fair. This was "normal" for the transaction manager prior to Release 3.X of MPE/iX. At 3.X, HP smoothed out this spiking activity and hid most of it in the CQ.

The main reason I mention this brief case study is to underscore the need for you to keep a lookout for system behavior that seems to be out of bounds. Each new release of an operating system, or for that matter, any application, could present some undocumented "features," such as the one above. With a good monitoring game plan in place, you will not get caught off-guard.

Conclusion

These case studies will help you begin to diagnose your own system's health. They are by no means an exhaustive set of the performance problems I have seen on MPE/iX systems. They merely represent an introduction to performance diagnosis. You will certainly be ahead of the vast majority of technical support and system manager folks if you can master the majority of the textbook material and case studies.

Notes:

"All stock prices depend upon the hopes and fears of the buyers. Their hopes and fears translate into their decisions to buy or not to buy. Therefore, it is the hope and fears, human emotions, of the buyers that drive the markets. Those human emotions are created and fired by publicity: news, advertising, and even rumor. Fundamentals, statistics and technical data have no effect on the market unless those elements are publicized to generate an emotional response from the buyers. It is the tenor of that news, not the news itself, that causes the market to react."

The Burning Match Principle, by Jim Straw

Section Four

An Encyclopedia of HP 3000 Performance Terms and Management Questions

This section will serve as a performance reference manual. The questions and definitions should help in your understanding of HP 3000 performance.

A Picture of Perfect Health

With Appropriate tuning, any "system" (carbon or silicon based) can be healthy.

Chapter 16

Commonly Asked System Performance Management Questions

During the course of consulting and talking with hundreds of system managers at seminars and user-group shows, I have been asked many questions regarding performance. This chapter addresses many of the most common questions.

How do I know when it will be time to upgrade my CPU?

This question must be answered first by asking two others. Are your service levels being exceeded? If so, which resource is the bottleneck? All too often I have seen individuals procure a larger system not really knowing if they have truly exhausted the useful life of their existing system. Stressing any resource (CPU, memory, I/O) can negatively impact transactions. Many times managers have not implemented any kind of capacity plan, not to mention a realistic monitoring strategy that will provide them with early warning resource saturation indication.

Notwithstanding the above lecture, knowing when to upgrade the system really boils down to a business decision. This decision should take into account the impact of things like:

- Which resource(s) are we really short on?
- How much book value remains for the system and peripherals?
- Will adding memory help?
- What do we do with the lease?
- What kind of trade-in incentives are there? When will they expire?
- What about software license upgrade costs?
- Will we save on electricity, floor space, air conditioning, etc.?

The business decision should also address the service level issue. If you cannot deliver appropriate service (response time) for online users and batch processing, then you may have to upgrade. But maybe not. If you have poured through the bulk of this book, you probably understand that the answer most frequently given by performance specialists when confronted with nearly any question about performance is, "It depends!" An upgrade from purely a service level standpoint really depends on someone (you or a consultant) determining the problem.

Keep in mind that poor computer service may involve issues more complex than simply an overloaded CPU. I have seen numerous occasions when a move to a faster CPU would not have substantially increased throughput for a given application mix. As stated before, transactions need various types of services performed. CPU, memory, disk I/O, etc., all are involved. Other issues, such as application design, data locality, data base housekeeping, and others, also come into play. If one or more necessary resources are in short supply, the operation may suffer. If the problem is data locality, simply upgrading the CPU is unfruitful, imprudent and is a waste of resources.

In a nutshell, an upgrade decision (any hardware component) should be made only when the bottleneck has been determined. The severity of the bottleneck also should be outlined. I recommend that you re-read my discussion on system performance pulse points. Learning when a particular resource is exhausted will help you make decisions to tune or housekeep the system better/differently, or to simply bite-the-bullet and upgrade a hardware component.

Current poor service usually gets more attention than future needs! But good system management also involves ensuring adequate hardware is available down the road. Therefore, a decision to upgrade your system must involve some level of capacity planning (see the discussion on capacity planning in Chapter 2 and Appendix A). Knowledgeable managers take into account information regarding resource utilization, business factors, and future requirements before making hardware upgrade decisions.

Does it make sense to merge two systems onto a larger single box?

I am often asked this question. Once again, the answer is, "It depends." There are an immense number of business-related issues that need to be addressed here, and the performance side of things may not be much simpler. Here are some of the reasons why.

Two processors are able to perform two things at one time. A single CPU is really limited to one task at a time. It is for this reason that I most often advocate the opposite course of action: splitting applications off of one system and onto two (or more). It would be ideal to dedicate a processor for each major application; a lot of system overhead and paging activity can be avoided by doing this. But a veritable Pandora's Box may be opened with all the business and operational questions involved.

It is best to perform some sort of capacity modeling to determine if a consolidation would be best. But even the best modeling available will only put you in the ball park due to the complexities involved in such a move. For this reason, it is best to perform an actual benchmark by placing the application mix of both systems on such a target system.

How do I know what will happen if I add more users to my system?

There are three approaches to answering this:

First, simply take a risk and connect 'em up. This might seem like the epitome of flying blind, but sometimes it works. It is initially inexpensive to take such a gamble because you don't have to invest time and money into an insurance policy that will cover you in the event that your system's capacity is exceeded.

The second approach is a little more prudent. It requires that you put some time into determining the current condition of your system's resources. This involves applying good rules of thumb (see the Pulse Points discussion in Chapter 6) to what your performance monitor shows you, and from there, determining the impact of adding x number of users to your system.

A "workload view" of your system is crucial in order to understand the impact of additional demand. For instance, if you know that the Order Entry application (consisting of, say, 40 users) is currently consuming on average 40 percent of your CPU, then you may be able to roughly equate one effective user assuming similar activity with one percent of the CPU. Most cases are not this simplistic, however. But if you know exactly how your workloads are sharing the CPU, you might be able to perform some simple spreadsheet math and project how busy the CPU would be with the additions.

The bad news with this method is that it really cannot predict response times, disk I/O increase, any unusual behavior that results in "knees" in performance curves, etc. However, if you are not close to CPU saturation (85 percent for MPE/iX and 75 percent for MPE V), you can probably get away with this method.

The third method involves the application of time and possibly money. Professional capacity planning usually involves some sort of modeling. For instance, when HP performs their HP CAPPLAN service, they utilize a queueing network modeling tool. This method has all-around accuracy, but is still not a magic wand. You could build a model yourself utilizing queue network algorithms or by investing in a commercial tool. Many such tools exist for the IBM environment, but only one such tool exists for the HP 3000: Forecast/3000 Capacity Planner. This modeling tool will be able to answer a vast array of "what if" questions regarding the addition of more users and other scenarios.

How do I know where the next bottleneck will occur?

Sometimes it is difficult to predict such a thing. Earlier in this book I referred to the old bed spring analogy. That is, when you press one spring down on an old mattress, another one pops up elsewhere. The best way for you to arm yourself with any prediction capabilities is to carefully review material that discusses the inter-relationships between system resources. A good place to start is the textbook section of this book.

It is also a good idea to equip yourself with some of the yellow zone performance indicators referenced in Appendix B. When your system is consistently exhibiting symptoms of marginal resource pressure, you should begin to keep your eye on those resources. Early warning will give you a chance to put together a plan to deal with the problem.

Below are a few principles to keep in mind when trying to forecast the next system "choke point".

1. If your system is exhibiting signs of memory shortage, and you " fix" the memory problem, you may end up with a much busier CPU. This could thrust you into a CPU saturation condition if you were close to one initially.

2. A high CPU pause-for-disk indication can be a sign of a disk I/O problem (database, locality, etc.). If you are able to reduce the pause for disk by addressing disk bottlenecks, most likely all of that pause time will now be translated into CPU busy time. Again, this could produce a CPU "pinching" condition.

3. Upgrading the CPU may reveal a single-threading phenomena within an application that could result in poor response times for users. I have seen instances, especially on the high-end systems, in which applications that ran fine on a smaller system became bottlenecked on a larger system. There

might be ample CPU, memory, and disk I/O capacity but something to do with the application or data base design causes a one-horse merry-go-round effect. Simply put, do the application and supporting data structures scale with the higher horsepower systems? If you or your vendor are not relatively confident in answering this question, it would be advisable to engage a consultant to assist you.

How do I know what size CPU is needed when I upgrade?

This may be a relatively easy question to answer since the hardware vendor often places various marketing restrictions to move you onto a particular system. For example, if you are on a 920, you probably will want the horsepower/price advantage of the 9x7 systems. How about the 917? HP states that the 917 processor has a relative performance factor of 5.4 times that of the 920. But only a limited number of terminals and peripherals may be connected to the 917. The 947 has the exact same horsepower rating as the 917, but allows more terminals. A 957 would be too big (and expensive), and a 932 may be obsolete shortly, so your decision has been made for you. Admittedly, this was a simple example; many other situations are not this simple.

The best way to know how much horsepower is necessary for an operation is to put together a capacity plan and perform some kind of forecast. This may include a simple linear regression, such as described in Chapter 2, and putting it into a spreadsheet or statistical package for analysis. Or it may involve a more sophisticated (flexible and accurate) analysis by implementing modeling techniques or tools (Forecast/3000 Capacity Planner, HP CAPLAN). With modeling, you can evaluate many different upgrade scenarios to determine CPU power, but you still need to consider business issues such as those described in previous discussions. In any case, you need to beware of the avenues that may exist, taking into account factors other than simply CPU power. Think about memory constraints, connectivity, etc.

How much memory is enough?

A few formulas exist to help you determine approximately how much memory you will need to support your operation. See the main memory discussion in Section One. An MPE V system will use as much as is available, up to a point. Generally speaking, as long as you are within supported limits, MPE V and MPE/iX systems will typically use all they can get. This assumes,

Nothing is quite as embarrassing as watching your boss do something you assured him couldn't be done. McKenzie

of course, there is sufficient demand. It is possible to overkill; too much memory will be a waste of money and perhaps not provide the incremental gain that you would have expected.

I have seen situations involving large MPE/iX systems that exhibited poor data locality. At first glance, it seemed that these systems could use more main memory (as evidenced by moderate page faulting, etc.). After adding more memory, it was discovered that the faulting didn't change appreciably. Further investigation concluded that a data locality problem existed, which caused the system to have a high CPU pause-for-disk condition, and which was a function of the extremely randomized data retrieval habits of the users. This was coupled with an incredible number of very large data bases (all of which were fair game for inquiry). So, except in this extreme kind of situation, as long as your memory pulse point indicators are in the green zone (see the pulse points discussion in Chapter 6), you are safe.

Why don't user transaction counts or response times match what my performance tools say?

When users enter transactions, they are usually thinking in terms of logical units of work. When the operating system thinks of transactions, it thinks in terms of terminal reads (every time a RETURN or ENTER key is pressed). So, if a user is entering a new record, the entire screen form will be filled in and the final RETURN or ENTER pressed. The system processes the transaction and returns with a prompt. While the user assumed one transaction was posted, the system could have "seen" many transactions (terminal reads).

Herein lies the problem with performance tools that record terminal activity. A character-mode application will generate many system transactions to accomplish only one user transaction because multiple RETURNS are issued when traversing from field to field. On the other hand, a block-mode application will generate one system terminal read transaction (MPE V will show more than one due to not stripping out status reads).

If your application is block mode, then your performance tool will be reporting fairly accurate transaction counts (from the users perspective). With character mode, the best you can do is to try to get a feel for what certain terminal read counts translate into in terms of user-perceived transactions.

What is the SAQ ? How Can it help me?

This acronym stands for the System Average Quantum (SAQ). While this value applies to MPE/iX systems, its cousin on MPE V is called the ASTT. For all intents and purposes, they are the same.

The SAQ is a value that represents an average of the amount of CPU used per interactive transaction by user processes. For MPE V it is the average of the last 100 transactions; while I'm not positive, I believe that this is true for the MPE/iX also. At minimum, the SAQ value can give you a feel for the interactive transaction "hog factor" at any one time. If the SAQ value is 100, this means that the average transaction on the system used 100 CPU milliseconds to accomplish its work. The SAQ is also used by the dispatcher as the basis for decaying unruly processes. Transactions that exceed the SAQ value before completing a transaction will be decremented in priority, with obvious performance implications. As you watch this value fluctuate over the course of a day, it will give you a feel for which applications have a more vociferous appetite for CPU.

Practically speaking, you may utilize the SAQ as a basis for some system tuning with the TUNE command. Here's an example. Let's say you have two applications, order entry and accounts receivable. The order entry application is very important to the company since customers should not have to wait to give an order. Let's also say that the average transaction for order entry consumes 400 CPU milliseconds. An AR transaction only needs 100 CPU milliseconds. If the SAQ value were 225, then order entry users would be penalized quite often.

To combat this problem you could implement the TUNE command's feature of raising the SAQ value artificially by entering:

```
TUNE;CQ=152,200,500,500
```

This will cause order entry processes to compete more equally with AR processes.

While not the most important performance metric on the system, the SAQ can provide insight into your system's activity that is not easily obtainable elsewhere.

Everyone wants to go to heaven, but nobody wants to die.- U.S. Senator Phil Gramm on the balanced budget

My system is running at or near 100 percent CPU busy most of the time. How can I be sure it is time to move up to a larger system? Will the CPU busy change or will everything just run faster?

If your system is running at 100 percent CPU capacity, you need to ask yourself a couple of other questions. First, what percentage of this busy time is due to high-priority, interactive and batch activity? Second, are users happy with response times and job turnaround times? If a large majority of the workload is high-priority, meaning you could not sustain a substantial increase in service time, and users are generally unhappy, you either have to upgrade, use less, or put up with what you have.

With a faster system, if CPU really was the only (or at least the primary) bottleneck, batch jobs and user activity will certainly speed up. The CPU busy percentage will go down for user transactions. On the other hand, you still will probably see the CPU "pegged" at 100 percent for batch activity. The important thing to note is that the duration of the jobs will be much shorter.

What are the most appropriate ways to view and measure my system's performance?

Simply put, you need to master the Angles, Bases, and Pulse Points spoken of in Section One. You need to be equipped with a tool that will help you monitor a crisis situation. Analysts will benefit from being able to quantify and characterize application changes. Managers will need future system usage information to make prudent decisions. Programmers, analysts and managers alike will appreciate being able to casually monitor the system and learn about its performance idiosyncrasies.

A view of the system that leaves out global, workload, or process data is incomplete. Knowing that your system is 95 percent busy provides you with good visibility. Finding out that a single department (workload) is taking 75 percent of that total busy time is also revealing. Identifying a program that is the culprit for that workload will allow you to potentially buy more time with the existing hardware you have. This is really a large component of what proactive system management is all about.

Confidently measuring various performance pulse points will give you assurance that you will not be caught in the middle of a surprise upgrade.

All the scholastic scaffolding falls, as a ruined edifice, before one single word--faith. Napoleon Bonaparte

How can I differentiate between an operating system and application bottleneck?

This can be tricky. My experience has been that operating system bottlenecks can hide in a number of places. One common area is CPU overhead. I have seen faulty hardware (boards and external devices) cause excessive CPU overhead (ICS) on both MPE V and MPE/iX systems. Watch this value (CPU busy on overhead).

Some "sneaky" system processes can consume inordinate amounts of CPU while doing their duty. To trap these, watch them over a period of time with your performance tool and perhaps even create a separate workload for them so that you could monitor their aggregate impact on the system. This is also a good idea for third-party utilities! Just how much CPU do all those neat utilities use on the system? You might be surprised.

It is also a good idea, while searching for operating system bottlenecks, to identify its process' wait states. If processes are delayed, you might want to talk to the HP response center and find out what the issue is. I would also be suspicious of new operating system releases. If your system is acting up right after an operating system install, start asking questions.

As far as application bottlenecks are concerned, it is important to find out what is causing response time delays (interactive and batch). Once again, this is best addressed by identifying the various wait states for that application's processes. If the application is waiting for an operating system event or is being preempted by system processes, then the operating system might be to blame. If not, then a quick analysis of the wait states will reveal more. It could be that processes are delayed by a number of things like: CPU, memory, disk I/O, father/son message files, preemption, terminal I/O, semaphores, impedes (locks), etc. If substantial time is found in any one of these categories, start asking questions of the application developer.

How often should I be collecting performance data and charting it? Continuously? Monthly?

I'll be brief here. You will never regret collecting and archiving relevant system performance data. The benefit of reporting historical trends far outweigh the trouble of making sure the performance collector is running and occasionally archiving some files. I have talked to numerous people regarding this subject, encouraging them to collect data when they upgraded to a new system. However, the lull of a new system's awesome performance sometimes keeps us from being proactive. Many of these people have contacted me a year or so later wanting to know what to do since performance had "gone sour."

Young children don't understand abstractions such as love. For them, things become real only when they touch them. Your touch will make real the words, "I love you." G. Smalley

It is a good idea to create simple, weekly summaries of system activity. Not the presentation quality type, just some report card of system activity. Then, monthly, you should create some fancy graphs that illustrate key performance indicators. These graphs could be presented to management to keep them appraised of system happenings. If the charts say that the system is being taxed heavily, then this might cool their plans for new applications. This is much better than adding the application and wondering what happened to performance.

What are some good performance preventative-maintenance techniques?

One: Always collect performance data!

Two: Begin to chart historical performance right from the beginning of new system installation.

Three: Every month or so create a suite of relevant tabular or graphic reports for a typical busy day and match this data with the pulse point values in Appendix B.

Four: Keep your performance tool running continuously on a terminal. This will allow you to avoid the sometimes long start up time for some tools (especially if the system is hung!). You will always have the last minute or so of data right on that screen. This data, just prior to the negative event, will be valuable to ensure that the problem will not happen again.

Five: Make sure that your software vendor and application programmers provide you with a statement of program consumption prior to implementation. Remember they have "blank checks" to spend your system's resources as they please. Here's an idea: try to quantify how much it costs if a new software release uses 10 percent more CPU.

Six: Create a flowchart of steps for you or operators to go through in a performance crisis.

What is the first thing to look at in a performance crisis?

When the system as a whole or an individual workload is experiencing severe performance problems, I usually start asking questions. Here's some to ask yourself if crises ever occur at your site:

- How much idle time is there? If none, then how much pause time? If the answer is still none, then the CPU is probably choking.
- How much of the CPU's time is being consumed by overhead tasks (ICS, dispatcher, memory management, AQ time, etc.)? If this is a substantial amount, there could be a faulty device causing interrupts, a shortage of memory or high-priority processes consuming CPU time.
- How many mode switches are occurring on an MPE/iX system? If a lot, then how much CPU is spent being utilized on compatibility mode?
- What is the dispatcher launch rate? If high, why are processes stopping so often? Investigate the wait states.
- What are the wait states for the various workload groupings of individual processes?
- Who is the CPU hog?
- Who is the disk I/O hog?
- Who is the terminal read hog?
- Which disk drive(s) is being hit the hardest?
- Which queue is receiving the most CPU time?
- Which user workloads have high terminal responses? If they do, does it take a long time to get a prompt? If so, what delay events does the wait state analysis point to? If only a group of users has problems, which data bases are they accessing? Could be a file-locking problem.
- What is the Read Hit% for an MPE/iX system? If high, it's probably not disk or memory (some exceptions). If low, it could be data locality (data base) and/or a memory shortage.

The important thing in a crisis is to be prepared. Maybe it's time for you to reread the textbook section again! Also, be sure to get a screen dump of the data from the performance tool you are continuously running!

It's extremely difficult for a child to live right if he has never seen it done. McKenzie

What are the most important performance indicators?

Our country would become fabuously prosperous if, beginning tomorrow morning, everyone would do a full day's work. Howard Kershner

If I were on a desert island and could only have a handful of HP 3000 performance metrics, here are the ones I would pick and why:

- **CPU Idle time** - If there is no CPU in the bank, I cannot spend any!
- **CPU CQ and DQ Busy time** - This will tell me whether my high- or low-priority workloads are dominating the system.
- **CPU Pause for disk** - If there is a lot, this points to a disk bottleneck. If there isn't any, the CPU is being utilized quite well.
- **CPU Queue length** - Just how many processes are waiting to receive CPU service? High is bad, low is good.
- **CPU Busy on memory management time** - This will help ascertain the condition of main memory (MPE/iX).
- **Memory clock rate** - Good indicator of the health of main memory.
- **Read Hit%** - A high value indicates the I/O subsystem, memory resources and data locality are doing pretty well.

If you just focus on the above indicators, you will have a larger knowledge base of HP 3000 performance diagnosis metrics than a majority of the system managers and technical support people I meet.

When does it make sense to say "Uncle" and upgrade my Classic system to HPPA?

This is primarily an economic decision. But you must also take into account the current health of your system and your reasonable upgrade path. Notice I said "reasonable." If you are on a Series 52, is it reasonable to upgrade to a Series 70? Perhaps. Series 70s are inexpensive. The performance would be great. But what about hardware maintenance costs? And the environmental requirements? And the electricity? What about the increase in productivity on a much faster MPE/iX system? HP is making it more attractive than ever to make the jump.

Maybe a second Classic system would be better. As much as I am for the Classic systems' survival, there are more and more compelling reasons to upgrade to HPPA. The bottom line is, it will make sense to upgrade when the economics make sense. It is wise to take into account lost business, excess payroll, etc., that all could result from a slow computer.

Do I need to re-compile all my MPE V programs into native mode?

Probably not. Paretos principle of the 80-20 rule, when applied to computers, states that 20 percent of all the programs will account for 80 percent of all the system activity. So then, it really depends on which programs comprise this 80 percent group. What are the most frequently used and mission-critical applications on the system?

If a single batch job, run once at night has no trouble completing in its window, why bother re-compiling it? Busy user programs are the most likely candidates for native migration. Daily run batch jobs are prime targets also. The key is to identify the 80 percent.

Education makes a people easy to lead, difficult to govern, but impossible to enslave. Lord Brougham

If there lurks
in most modern minds
the notion that to desire our own good
and earnestly to hope for the enjoyment of it
is a bad thing,
I submit that this notion
has crept in from Kant and the Stoics
and is no part of the Christian faith.
Indeed, if we consider the unblushing promises
of reward
and the staggering nature
of the rewards promised
in the Gospels, it would seem that
Our Lord
finds our desires not too strong, but too weak.
We are half-hearted creatures,
fooling about with drink and sex and ambition
when infinite joy is offered us,
like an ignorant child
who wants to go
on making mud pies
in a slum because he cannot imagine
what is meant by
the offer of a holiday at the sea.
We are far too easily pleased.

C.S. Lewis

Chapter 17

HP 3000 Performance Encyclopedia

Active Process - A process is said to be active when it is allowed to use the CPU. After all the necessary pieces of its working set (data and code) are in memory and it has the highest priority, then that process is launched. This means that it has the CPU's undivided attention for a small amount of time. This continues until the process is blocked by some event (see Process Stop). At this time the once-current, active process loses the CPU, is re-scheduled, and is put in line to wait until the resource or event it needs becomes available.

Allocated Program - (see Program Allocation).

AS Queue - (see Scheduling Queue).

Average Short Transaction Time - ASTT (see SAQ).

Background Garbage Collection - (see Garbage Collection).

Batch Job - A type of process that does not utilize terminal input and is assigned a "job" status. The fundamental difference between a batch job and an interactive user is that a job really has no "think" time between transactions. In fact, the system sees the job as being one long transaction consuming large amounts of resource. It is like a very quick user simply hitting the RETURN key as soon as a prompt appears.

Block Mode - This is a form of terminal flow control that allows local user edits. With block mode the host itself does not have to be "bothered" with every entered character, field, or line. The ENTER key is used instead of RETURN. One implication of block mode with respect to performance is less

CPU is typically needed to handle data entry activity. With each press of a RETURN or ENTER key, the CPU is interrupted because the operating system needs to provide service. On MPE/iX systems, block mode is preferable. In fact, extensive character mode activity can drain the CPU. CPU time for terminal activity on Spectrum systems is allocated at the user process level, although there does appear to be some counted on the ICS-see ICS. For Classic systems, nearly all time for terminal interrupts is counted on the ICS. Just for fun, hold down the RETURN key and observe system CPU overhead with a *performance tool.*

Blocked - (see Process Stop).

Blocked I/O - When a request for a disk I/O is made, the calling process must wait until the I/O is retrieved. The process is considered to be blocked on an I/O wait. It is this time that causes the CPU to be in a paused-for-disk state when there are no other processes waiting for the CPU.

Blocking Factor - The value that determines how many logical records (actual file entry records) are contained within one physical record (one disk I/O).

Bottleneck - A constriction of transaction progress. A bottleneck occurs when some necessary resource becomes scarce. Common bottlenecks are CPU, disk I/O, memory, disk space, etc. Keeping bottlenecks under control is one aspect of system performance management. Refer to the discussion on bottlenecks in the Section One of this book.

Brother Process - Also known as a "sibling" process. A process is a brother of another process if they both have the same father ("parent").

BS Queue - (see Scheduling Queue).

Buffer - A temporary residence for data, usually referring to main memory. Data incoming from, say, a terminal device, will reside in various buffers during its journey from the user to being processed at the CPU.

Busy - A resource that is currently being used or a process that is doing something. When your performance tool reports a resource metric, such as CPU busy, this usually means that over a period of time (an interval), the resource was actively performing work on behalf of system or user processing.

Cache Domain - For MPE V systems only. A cache domain is basically a transplanted piece of disk data. It occupies a portion of main memory and is generally available to all users globally, subject to the normal security restrictions imposed by MPE. The number of cache domains found on a system is a function of demand and the amount of main memory available. See Taming the HP 3000 Volume I for an extensive discussion of MPE software disk caching.

Caching - (see Disk Caching).

Circular Queue - The three user process queues (CS, DS, and ES). These queues are so named because processes in these queues typically drop in priority and are periodically raised again; hence the "circular" concept. With the advent of the MPE/iX OSCILLATE function of the TUNE command, CPU intensive processes are able to truly circulate back to the top of their queue once they hit the bottom. (see Oscillate and Scheduling Queue).

> ***Performance Implication:*** *Processes having circular queue characteristics usually are decayed and boosted back up to the base of their queue. The extent of this activity largely determines the kind of service that various workloads receive. That is, within a circular queue, hog processes receive increasingly worse CPU time than non-hogs, thereby favoring CPU-frugal ones.*

CISC - Complex Instruction Set Computing. This is the term used to describe the architecture of Classic HP 3000 systems (as well as many other non-HP systems). This older architecture used a large number of instructions in its command set. HPPA employs fewer instructions in its architecture, termed RISC (see RISC).

Clock Cycle - This term usually refers to some hardware device or operating system function that performs activity on a time-driven basis. One CPU clock cycle, for example, may execute one hardware instruction. The speed of the clock usually determines the raw speed of the CPU. The term can also be used to refer to an action of the memory manager, which scans memory to find free areas and marks files to be overlaid (see Memory Clock Cycle).

Command Interpreter - Also known as "CI." The CI is a system program that executes commands that users or jobs submit via the MPE colon prompt. Thus, the CI is the bridge between users/jobs and the operating system. When a user or job logs onto the system, a process is created. This process is considered the command interpreter process. At the MPE prompt, commands may be entered at a terminal or within a batch job. As an alternative, a program may execute commands via the COMMAND intrinsic. Thus, at a CI prompt you may execute one of the nearly 200 MPE commands available (SHOWME, TUNE, etc.). If a RUN command is issued, then a son process is created. Typically, users will have two processes, their CI process (generally stays dormant) and the meat-and-potatoes process (the one that does the work). Sometimes it is desirable to filter out the CI processes from a monitor's display since their response time values are often quite bizarre. This is because the father's (CI) response time encompasses that of the son (EDITOR, accounts payable program, etc.).

Compatibility Mode - Also known as CM. MPE/iX systems allow programs written for MPE V systems to run via compatibility mode. Most programs that execute on MPE V will run on MPE/iX systems without source re-compilation. This is accomplished by providing run-time translation of MPE V-based programs. Although providing a painless migration, CM is not a free lunch. In terms of economics, you either pay up front by compiling source code with a compiler native to the Spectrum architecture or you pay (CPU time) each time a program is executed.

Configuration - This refers to a unique union of hardware peripherals and software variables. It also can refer to setting various system table values. System table configuration is more of an issue on Classic HP systems than on the Spectrum series.

Controller Caching - This is a mechanism usually resident within a disk drive that performs anticipatory pre-fetching of data. That is, when an I/O is requested, instead of retrieving only the data desired by the calling process, much more data is brought into a memory area within the disk drive itself. Then, if the operating system requests more data, this memory "cache" area is searched first to see if an I/O could be eliminated. The operation of controller caching is very similar to software Disk Caching (MPE V). One advantage of controller caching is that CPU or memory resource from the host is not required to reduce disk I/O.

> ***Performance Implications:*** *Can be an excellent performance turbo-charger for I/O bound systems, especially for Classic systems that have a paucity of CPU. Occasionally, it is a good decision to enable both cache mechanisms on MPE V. Use of controller caching on MPE/iX systems is of questionable benefit except under severe I/O bottleneck situations.*

Core - This is also known as main memory.

CPU - Central Processing Unit. The CPU is the heart of any computer system. It is responsible for following the orders given by a computer program.

CPU Cache - On HP 3000 systems there is a portion of memory that holds the most recently used data or instructions. This memory is very high speed and is one way HP has increased performance on HPPA systems.

CPU Queue Length - This is also known as the Ready Queue. This value, reported by SHOWQ (number of processes in top right of display) and performance monitors, is the number of processes waiting to receive CPU attention.

> ***Performance Implications:*** *A high CPU Queue length may indicate a shortage of CPU. Be sure there are only necessary jobs and sessions running at the time you measure this value. Running a large quantity of unnecessary jobs could skew this number.*

CPU States - The CPU, technically speaking, has three "states." These states are often referred to as Active, Paused, and Idle. By Active we mean that the CPU is actually expending horsepower on behalf of some demand from either the operating system or user processes. Idle CPU time is "money in the bank"; this is truly available horsepower. Ample idle time implies the CPU is not being asked to do much. Paused is similar to Idle, but is the state in which the CPU has *nothing* to do (is idle) but at least one disk request is outstanding. This means, of course, that if the necessary data were available there *would* be something for it to do.

CS Queue - (see Scheduling Queue).

Decay - (see Priority Decay).

Device - This usually refers to an external piece of hardware typically involved with I/O activity. It could also include printers, tape drives, modems, etc.

Directory - This is an operating system data structure that is similar to a telephone directory. It allows MPE to locate data files on physical disk devices by keeping track of their physical volume/sector addresses.

Dirty Page - A block of main memory that has been modified but has not actually been posted to disk. The actual data will be posted out to disk upon request or if the space the data is occupying is needed by other processes.

Disk Caching - This is an operating system feature that retrieves more data than was actually requested. The surplus data is then placed in memory locations called cache domains (MPE V). The operating system tries to satisfy subsequent I/O requests by first checking cache domains.

> ***Performance Implications:*** *Can make a "life or death" difference to a Classic system. There are very few reasons why one would not utilize some sort of caching utility on MPE V systems.*

Disk I/O - This is a activity performed by the operating system to retrieve or post data from/to a physical disk device. Data must be retrieved from such devices and placed in main memory before processes can make use of them.

Disk Queue Length - This is the number of processes waiting to be serviced by a disk device.

> ***Performance Implication:*** *A long disk queue may indicate a disk I/O bottleneck.*

Dispatcher - The Dispatcher is the "traffic cop" of the system. Its responsibility is to distribute CPU time to processes based on a "fair" allocation scheme. Fairness is determined partly by MPE and partly by the system manager. There are numerous "knobs and dials" that can be twisted to alter the way the dispatcher apportions the CPU resource. Its overhead is measured by the operating system measurement interface and is reported as the amount of CPU time spent on dispatching activity. For MPE/iX, some tools report it as a separate indicator, but on MPE V, its time is mixed with ICS overhead.

DS Queue - (see Scheduling Queue).

ES Queue - (see Scheduling Queue).

Execution - A process begins execution when it is launched by the dispatcher. One of the following events will cause an executing process to lose use of the CPU:

1. The process is "bumped" by a higher priority process (see Process Preemption).
2. It is interrupted.
3. It exhausts its allotment of CPU time.
4. It stops due to the need for some event to occur; this is usually considered to be one of many process stop events (see Process Stop).

Father Process - A process is said to be a father (parent) when it spawns another process. This new process is called a son (child).

Fault - A fault occurs when a necessary code or data object (segment) is absent from main memory.

File Label - The part of a file that contains specific characteristics relating to that file.

Frozen Memory - Portions of main memory that cannot be swapped out to disk by the memory manager. Usually these are high-priority pages (segments) that are part of the operating system or critical subsystems.

Garbage Collection - An action of the MPE V memory manager, which attempts to create larger free areas of memory. Garbage collection occurs in two primary ways: Local and Global (also known as background collection). Local collection is routine de-fragmentation and is performed as the memory manager is serially scanning memory. If it finds two pieces of empty memory that could be combined (knock the logical barrier out from in between them), it does so. Global collection occurs as a semi-emergency effort to find room for a large incoming request. This action takes place when the memory pressure flag is on (the current request for memory is larger than the largest available area) and the CPU has some idle time available, and if Disk Caching is off or is on with less than two megabytes of memory (whew!).

HPPA - Hewlett-Packard Precision Architecture. This is HP's implementation of RISC.

ICS - The interrupt control stack. This is a system table that holds hardware requests. For example, when a disk device retrieves an I/O on behalf of a process, an ICS entry is posted that interrupts the CPU. This activity costs CPU time just to handle the interruption (not counting actually communicating the I/O to the requesting process).

> ***Performance Implications:*** *ICS activity can be a source of CPU overhead. Excessive interrupts can impact performance negatively by increasing the amount of CPU time spent handling ICS requests.*

Idle - A condition of the CPU, in which it has nothing to do and is not waiting for any events.

I/O - Input/Output. This refers to data flowing between the CPU and hardware devices.

Impede - An impede is a special process-stop event that takes place when a process cannot gain access to a necessary data structure due to another process having prior exclusive access. An example of this might be an interactive user who wishes to read a record from a TurboIMAGE data set, but a batch job already has the record/set/data base locked. The user would have to wait a finite amount of time until the job issues a DBUNLOCK. At times, the operating system will temporarily raise the priority of the batch job to that of the user process so that it can get enough CPU time to actually issue the unlock.

> ***Performance Implications:*** *Excessive process impede activity can be detrimental to performance, especially within a workload (multiple processes) that accesses common data structures. High process impede values can be indicative of a need to rethink a locking strategy.*

Interactive Activity - This refers to online user terminal processes (as opposed to batch activity). Such processes usually have "think" time between transactions.

> ***Performance Implications:*** *Unless users are performing online batch-job-like functions (online compiles, reports, etc.), interactive sessions usually impact system resources less than batch jobs due to think time.*

Interprocess Communication (IPC) - A facility to allow processes to communicate with one another, using a "message" file. Data is written to and read from such files by processes within or outside of the same job or session process (family) tree.

In one year Erno Rubik, the Hungarian inventor of the Rubik's Cube puzzle, brought in more profit and hard currency than all of Hungarian heavy industry combined. Mr. Rubik is a millionaire. Hungarian heavy industry is a mess. John Naisbitt

Performance Implications: *This facility has overhead similar to that of normal MPE file activity, but for MPE/iX, since the IPC code is still in compatibility mode, switch overhead can be harmful to performance.*

Interrupt - An event that occurs to inform the operating system (on behalf of a waiting process) that a requested operation has completed. Think of interrupts as hardware signals of device activity. Typically, this will be a disk I/O completion, but could be almost any other event from the world outside the CPU box itself. This includes, but is not limited to terminals, datacomm devices, disk drives, etc. Hardware handlers take control when an interrupt occurs, often stopping a process that is currently executing. Some of the effect of interrupts can be measured by looking at the amount of CPU time spent managing them (see Overhead).

Job - (see Batch Job).

Junk Wait - A process-stop event that refers to system processes that have no activity to perform.

Launch - (see Process Launch).

Linear Queue - (see Scheduling Queue).

Loader - This is a system process responsible for preparing a program file for execution. Some of the actions of the loader are:

- Assigning system table entries.
- Assigning code and data space (segments).
- Attempting to resolve external SL (system library) references.
- A number of other preparatory services.

Performance Implications: *The loader is a single-threaded process. This means that only one program can be loaded at a time. If many users were trying to run programs simultaneously, they would become bottlenecked on the loader. The loader also can take some significant CPU resource. It is instructive to monitor its consumption over a typical day if you think it might be a performance culprit. If so, you might consider a different strategy than running and terminating programs. Sometimes a better solution is to utilize the process handling functions of SUSPEND and ACTIVATE, which do not utilize the loader (after initialization).*

Locality - This term refers to the proximity of data likely to be accessed by user processes. If commonly used data is kept geographically close to one another on disk, then that data is said to exhibit good locality. Data that is

scattered across disk has poor locality. Numerous discussions and case studies relating to the issue of data locality are presented in previous chapters.

> ***Performance Implications:*** *Good locality is vital for both MPE V and MPE/iX systems. You can have plenty of CPU, memory, and fast disk devices, but if you have poor data locality, your system will pay a performance price. Both MPE V disk caching and MPE/iX pre-fetching depend on good data locality in order to eliminate excessive disk activity.*

Main Memory - This refers to electronic chips within a computer used to temporarily store data on behalf of various operating system components and the CPU. Usually this memory is volatile; that is, it needs voltage applied in order to keep its contents intact. When the power goes out, internal batteries (and perhaps an external UPS system) will keep the memory refreshed. The operating system utilizes memory to store "surplus" data that is pre-fetched (MPE/iX) or cached (MPE V). Having enough memory will help ensure that costly disk I/Os do not have to take place as frequently.

> ***Performance Implications:*** *You better have enough of this resource!*

Mapped File - A disk file that is accessed by virtual memory instructions (loads and stores) rather than the normal file system intrinsics (like FREAD, FWRITE, etc.). It is treated by the programmer as if it were a memory array. Mapped access to a file creates a one-to-one relationship between each byte in virtual memory and the file itself. See the discussion about mapped files in the MPE/iX section.

Memory Clock Cycle - This is an activity of the MPE memory manager. When there is a request for memory space, the memory manager begins to search memory where it last left off. The time it takes to cycle through all of main memory is referred to as a clock cycle. This time is an important performance metric for both MPE V and MPE/iX systems, although the indicators have radically different good/bad thresholds. See Chapter 6 for a complete explanation of this event.

> ***Performance Implications:*** *A fast memory clock cycle rate is indicative of a shortage of main memory.*

Memory Fragmentation - A "checker board" effect that occurs on MPE V systems when areas of memory become unavailable for use due to their small size. MPE V deals with this problem much like you would perform a RELOAD or VINIT CONDENSE to de-fragment disk space. The memory manager performs an action called garbage collection (see Garbage Collection) to solve the problem. Fragmentation does not occur on MPE/iX systems so there is no need for such action.

Run not into debt, either for wares sold, or money borrowed; be content to want things that are not of absolute necessity, rather than run up the score.
Sir M. Hale

Performance Implication: *If memory is in short supply and is being fragmented at a high rate, the operating system will have to spend more overhead time dealing with the problem.*

Memory Manager - This is an MPE module that performs various memory-related tasks, such as making room for incoming data, managing fragmentation (MPE V), checking to see which memory objects are candidates to remove from memory, searching to see if necessary data are already in memory (avoiding an I/O), etc.

Performance Implications: *If you do not have enough memory to support the current workload, the memory manager will have to work harder. This simply means more of your CPU's time will go into managing such activity (overhead).*

Minquantum/Maxquantum - These values represent borders for the range of the calculated System Average Quantum for MPE/iX (see SAQ) or the Average Short Transaction Time for MPE V (see ASTT).

MIPS - Millions of Instructions Per Second. This acronym has been used to rate the raw horsepower of a particular CPU. MIPS is not the best rating of overall throughput and performance because it does not take into account other areas of performance bottlenecks (I/O, memory, etc.). A better indicator would be something like the TPC/A benchmark (see TPC Benchmark A).

Performance Implications: *The more of these your CPU has, generally speaking, the faster your programs will execute.*

Monitor, Monitoring Tool - A program that is used to provide indication of computer performance activity. Typical tools will provide technical trouble-shooting data for support staff as well as business-relevant information for managers.

Performance Implications: *Read (re-read?) "The Perils of Flying Blind" in the Introductory section of this book!*

MPE - Multi-Programming Executive. This is the operating system for HP 3000 systems. It refers to MPE V, MPE/XL, MPE/iX in a broad sense.

Multi-Processing - A type of computer system that actually has more than one CPU in it. This kind of system is capable of performing more than one task at a time.

Performance Implications: *This system is capable of more throughput than a single-processor system.*

Multi-Programming - Describes a computer system that is capable of managing multiple user and batch tasks. This kind of computer is not necessarily capable of performing two activities at the same time but, rather, uses time-slicing techniques to provide a portion of CPU for processes, giving the appearance that all programs are being serviced simultaneously. The term "MPE" stands for Multi-Programming Executive.

> ***Performance Implications:*** *As the number of processes needing service increases, the response time for each process tends to increase accordingly.*

Oscillate - (see Priority Oscillation).

Overhead - This term is used to describe activity on the system that produces no useful benefit except indirectly on behalf of user processes. CPU ICS overhead is one example of this (see ICS). When a disk device, for example, retrieves an I/O, it informs MPE by means of an interrupt. This takes a finite amount of the CPU's attention to handle. Although the I/O is important to some processes, the interrupt itself is necessary but not directly useful to that process.

> ***Performance Implications:*** *Excessive overhead can prevent CPU resource from being spent on useful activity.*

Page - A unit of memory on MPE/iX systems consisting of 4096 bytes. Disk data are placed into pages in main memory before the CPU is able to use them.

PIN - This stands for Process Identification Number. MPE assigns a unique PIN number to every process on the system. PIN numbers appear in performance tools, SHOWPROC and SHOWQ commands, as well in many console messages.

Preemption - (see Process Preemption).

Primary Storage - (see Main Memory).

Priority Base - This is the highest priority that a process may possess within its scheduling queue. When a process begins its life via the RUN command or process CREATE intrinsic, it is assigned base priority of the queue in which it was born.

Priority Boost - This is an MPE dispatcher activity that causes the priority of a process to increase (the priority number actually decreases numerically). This can occur for several reasons. It is usually performed as a temporary measure to allow a higher priority process access to some resource "owned" by a non-executing process. A typical resource might be a data set that is

locked. The lower priority process is uplifted in priority, providing it enough CPU time to release the held resource. If you are lucky, you may actually see evidence of this with an online monitoring tool. A batch job that temporarily assumes a priority of, say, 152 would be a good example of this. After the resource is relinquished, the lower process is then returned to its original low priority. A more-than-temporary boost can also occur when process oscillation is enabled (MPE/iX) via the TUNE command (see Priority Oscillation) or when other events occur (such as a terminal read, message file read, timer wait, etc.).

Priority Decay - An action by the dispatcher that causes the priority of a process to be reduced (numerically increased). The dispatcher performs this action to keep hog (CPU intensive processes) from monopolizing use of the CPU. This is the default option for the TUNE command.

Priority Limit - This is the lowest priority a process can assume within its "home" scheduling queue. Processes in the CS, DS, and ES queues will "decay" down to the limit of that queue if they continually exceed the SAQ or ASTT (MPE V) in the amount of CPU time they consume per transaction. The limit of each queue can be affected with the TUNE command.

Priority Oscillation - This is an (MPE/iX) option assigned on a queue basis that instructs the dispatcher to raise a process' priority back to its base (a higher priority) when it reaches the limit of its queue (see Priority Base and Priority Limit). This feature prevents heavy CPU transactions from being locked out of receiving any CPU time. With the DECAY option enabled (see Priority Boost and Priority Decay), these transactions would simply sit at the bottom of the queue and perhaps not receive much CPU attention.

Process - The execution of a specific program by a specific user. There are three types of processes: system, session, or batch job. A process may be created by the RUN command, or by the CREATE and CREATEPROCESS intrinsics. Each process is uniquely assigned a PIN number (see PIN).

Process Launch - This is the activity that refers to a process receiving exclusive use of the CPU. A launch occurs when the MPE dispatcher has determined which process is ready to run and has the highest priority if there are many such processes ready. Typically, this activity will occur many times in the life of a process. A launch implies that a process stop occurred (see Process Stop). After a process is launched, it is considered to be executing (see Execution).

> ***Performance Implications:*** *Excessive launch activity implies excessive process stops. Each launch incurs CPU overhead (especially due to dispatcher activity). A low launch rate is desirable.*

Process Preemption - This situation occurs when a higher priority process (the "thief") steals the CPU away from a lower priority process (the "victim"). Not only must the thief process have a higher priority in order for preemption to occur, but at least one of the following must be true:

1. The thief process is in a higher scheduling queue.
2. The priority of the thief process was temporarily raised (see Priority Boost).
3. The victim process (currently executing) has used CPU time that is greater than the SAQ (MPE/iX) or ASTT (MPE V).

For MPE/iX version 3.0 and greater:

4. The victim process must not be classified as "un-preemptable."
5. The priority of the thief process has to be greater than the priority of the victim process by a certain amount.

Process Priority - Each process on the system is assigned a priority number from 0 to 255. The MPE dispatcher uses a processes priority to determine if it is eligible to obtain CPU time. The lower the number, the higher the priority. Often, processes are assigned to queues that are logical ranges of priority numbers (see Scheduling Queue, Priority Base, Priority Limit). A process priority can be designated as linear, which means the priority stays the same for the life of the process. It may also be designated as circular. This means that the process will be subject to the decay and boost action of the MPE dispatcher (see Priority Boost, Priority Decay, Priority Oscillation). The Process with the highest priority of a given list of processes waiting for CPU time will be launched (see Launch).

Process States - A process will assume various states during its life. The two primary states are active (using the CPU) or waiting. Some of the possible wait states a process can be paused on are:

BI/O	Waiting for blocked disk I/O to complete
CPU	Currently active in the CPU resource.
Dead	This process has terminated.
Fath	Waiting for activation by its father process.
GRIN	Waiting for a global RIN to become available.
I/O	Waiting for NOWAIT I/O to complete.
Imp	Waiting due to some resource being unavailable. Some examples are data base locks, lack of system table entries, etc.
JUNK	Waiting for a miscellaneous system wait.
LRIN	Waiting for a local RIN to become available.
Mail	Waiting for a mail transaction to complete.
Mem	Waiting for a segment(s) to be brought into main memory.
Mour	Waiting for a son process to terminate (MOURning its death).

If God were not willing to forgive sin Heaven would be empty. German Proverb

Msg	Waiting for message file I/O.
SIR	Waiting for a SIR to become available.
SON	Waiting to be activated by its SON.
Tout	Waiting for time-out to complete (pause).
Time	Waiting for a time-out to complete.
UCOP	Waiting for either service by UCOP or a reply.

Process Stop - A dispatcher action that causes a process to lose use of the CPU. A process will stop executing when one of the following categorical events is encountered: the process is preempted, uses up its timeslice, or encounters a block.

Program Allocation - This is a feature of Classic systems that allows much of the initial overhead to be resolved prior to actually executing a program. Program initiation incurs quite a bit of overhead. Allocating a program ahead of time (via the ALLOCATE command) allows preliminary overhead to be resolved ahead of time. The performance benefit of using this feature is only realized by the first run of a program file. Each subsequent user running that program will not have to suffer the initiation cost.

Quantum - Usually refers to a block of CPU time used by a process before it loses control of the CPU. Each dispatcher queue (see Scheduling Queue) has its own CPU quantum for processes. The quantum is set by the TUNE command.

> ***Performance Implications:*** *If processes are allowed too large of a quantum there may not be enough timely, equal sharing between processes. If the quantum is too small there may be excessive process launch activity (see Process Launch).*

Read Hit Percentage - This is the number of disk requests satisfies in main memory compared to the entire number of requests.

> ***Performance Implications:*** *This value is important for understanding the efficiency of disk caching (MPE V) and pre-fetching (MPE/iX).*

Ready Queue: This is a holding queue for processes waiting to use the CPU. Most performance monitors report this as the CPU Queue Length.

> ***Performance Implications:*** *It is widely known that the more requests there are in a queue the slower the response times will be for customers waiting in the queue. It is important to give attention to your system's CPU queue length as part of routine performance monitoring. See Chapter 6 for more on this.*

Resource: This refers to any hardware or software entity that provides service for processing data. Typical resources are CPU, memory, disk devices, etc.

Performance Implications: *Ample resource is what you need in order to provide service to customers. Performance management involves finding ways to properly match resource demand with resource supply. Not enough of one resource means user requests will be bottlenecked on that resource.*

Response Time - This is usually considered to be the amount of time measured from when a user presses the RETURN or ENTER key to the time when the user can resume entering data. For batch jobs, this time is considered to be the entire time for the job to complete. This subject is covered extensively in Section One.

Performance Implications: *If response times are high or inconsistent then users' productivity will decrease. Short-user and batch-response times are usually one primary aspect of good system performance.*

RIN - Resource Identification Number. This is a number assigned to control access to various resources such as files, printers, etc. Local RINs are used within a process tree, while Global RINs are available when spanning a process' tree. If a process has a RIN locked, this prevents other processes from gaining access to the locked resource.

Performance Implications: *Processes can experience response time problems if RINs are used improperly.*

RISC - Reduced Instruction Set Computing. This is the term applied to computer architectures that utilize a reduced set of hardware instructions. Hewlett-Packard's implementation of RISC is called HPPA (Hewlett-Packard Precision Architecture). Some of the main features of this architecture are:

- Few, simple instructions of the same size, which simplifies the fetch mechanism.
- Complex instructions are eliminated but complex operations are performed by running many simple instructions together.
- Optimizing compilers reduce the number of instructions necessary to perform complex activities.
- All instructions are implemented in hardware so that each one would execute in one CPU clock cycle.
- Three-level instruction pipelining, which increases instruction throughput.

The impersonal hand of government can never replace the helping hand of a neighbor. Hubert H. Humphrey

- Memory mapped I/O with very large virtual memory.
- Support for co-processors (HP 3000 990 model 300, for example).

SAQ - The System Average Quantum for MPE/iX. It represents an average amount of CPU time used by the last 100 interactive transactions (terminal reads). It is used by the dispatcher to make decisions to hurt or help processes by either lowering or raising their effective priorities. On MPE V the equivalent of the SAQ is the ASTT (Average Short Transaction Time).

Scheduling Queue - A physical queue of priorities (0–255) is logically divided into five separate subregions. These regions are called scheduling queues. Each queue may have special policies assigned to it that allow you or MPE to introduce processes with unique requirements. There are special "non-optional" rules that the dispatcher uses that cannot be changed. But there are also flexible options that you may utilize to change the behavior of the dispatcher. The TUNE command can be used in such a way to create an environment that better suits the needs of your department. The five subqueues are designated AS, BS, CS, DS, and ES. The AS and BS queues are usually reserved for high-priority system processes (some user processes), while the three lower queues are used for interactive and batch process activity.

Secondary Storage - This refers to disk devices. Data and programs are kept on disk until they are needed by the system.

Server, Service Center - (see Resource).

Service Time - This is the amount of time a transaction needs at a given service center (see Resource). Throughout the life of one transaction it will need time from the CPU and disk devices. Each of these devices will provide service for the transaction until the transaction is satisfied.

> ***Performance Implications:*** *The lower the service time required for transactions, the faster they complete. This may mean providing faster servers or writing more efficient program code.*

SHOWPROC - The MPE/iX command that displays the execution state and scheduling characteristics of processes. It shows a minimal amount of process data, such as the queue and priority, CPU time consumed, current wait state, etc. This data, though limited, can be useful mostly because it is quick to get at.

SHOWQ - The command that shows the current state of processes (dormant or running) and lists scheduling parameters for the various dispatcher queues.

SIR - System Internal Resource. MPE provides a queueing facility to ensure single-threaded access to special system resources. These resources include the system file directory, system tables, TurboIMAGE structures, etc.

> ***Performance Implications:*** *Although not very common, excessive SIR activity can create a bottleneck to resource access.*

Software Caching - (see Disk Caching).

SON Process - A process created by another process. The "spawning" process is the father.

Swapping - This is an activity of the MPE memory manager that involves issuing a request for disk data to be brought into main memory. A swap will occur when a process cannot continue without a necessary piece of code or data.

> ***Performance Implications:*** *Since swapping almost always involves disk I/O activity, this is not a desirable event.*

System Process - This is a process that operates as a function of the MPE operating system. Most system process activities are considered to be a necessary evil since they take high priority CPU time and generally do not perform activities of obvious benefit to user processing. It is helpful to monitor the aggregate activity of such processes to see their impact on your system. Some of these processes involve: spoolers, transaction management, the loader, device recognition, network agents, etc.

> ***Performance Implications:*** *These processes are generally frugal in their CPU consumption but occasionally can contribute to a CPU bottleneck condition. Extensive data communication activity can have a negative impact on the system as a whole.*

Terminal Read - This is an activity in which data from a user process (at a terminal) is retrieved and transmitted to the HP 3000. The system perceives this as a transaction, though the user may have to issue many such terminal reads in order to complete one real transaction.

> ***Performance Implications:*** *Excessive terminal reads can negatively impact a system's performance (especially MPE/iX and MPE V systems with ADCCs).*

Think Time - This is time that transpires between an application prompt and depression of the RETURN or ENTER. It is so named because a user is usually thinking or actually typing prior to pressing RETURN.

Performance Implications: *Think time is a vital component in performance capacity planning and also can be a deciding element in whether a system is utilized at capacity. The more think time per transaction, the less burden on the CPU (or other devices). Lower think times cause the system to be busier. You might think of a batch job as being a series of many transactions with no think time in between.*

TPC Benchmark A - This is a computer systems performance benchmark performed by the Transaction Processing Performance Council. This benchmark is intended to provide a comparison between various computer systems. This benchmark, "... Exercises the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing update-intensive database services. Such environments are characterized by: Multiple on-line terminal sessions, significant disk input/output, moderate system and application execution time and transaction integrity." [26]

Performance Implications: *A benchmark such as TPC A is a much better measuring stick of total system performance than MIPS (see MIPS). TPC-A takes into account all the variables necessary to effectively measure real-life performance for business computer systems. HPPA systems consistently rate very high on these.*

Transient Space - (see Virtual Memory.)

Turnaround Time - (see Response Time.)

Transaction - A transaction is typically thought of as a logical unit of work issued by an online user. The system defines a transaction as a terminal read, but users generally see transactions as the completion of the amount of service performed between terminal reads. Refer to Section One for more discussion on transactions.

Performance Implications: *Because transactions are units of work performed by the system, they cost performance. If transactions are complex and require great amounts of service, then resources will be taxed. Frugal transactions will have less impact on performance.*

UCOP - This is an operating system process whose job it is to handle user logon requests. This process is invoked when the HELLO or JOB commands are entered by potential processes.

Virtual Memory - This is specially reserved disk space used to temporarily off-load main memory data. MPE V uses the term virtual memory while MPE/iX uses Transient Space (a rose is a rose...). Effectively, a virtual memory

scheme is an inexpensive way to make main memory look bigger. The most active processes in main memory will have their data stay in memory, while less active processes' data will be temporarily sent out to virtual memory areas on disk devices.

> ***Performance Implications:*** *It is important to have enough virtual memory configured on your system to prevent program aborts or delays. A paucity of main memory will cause the memory manager to swap data out to disk, which, of course, will impact a transaction's response time.*

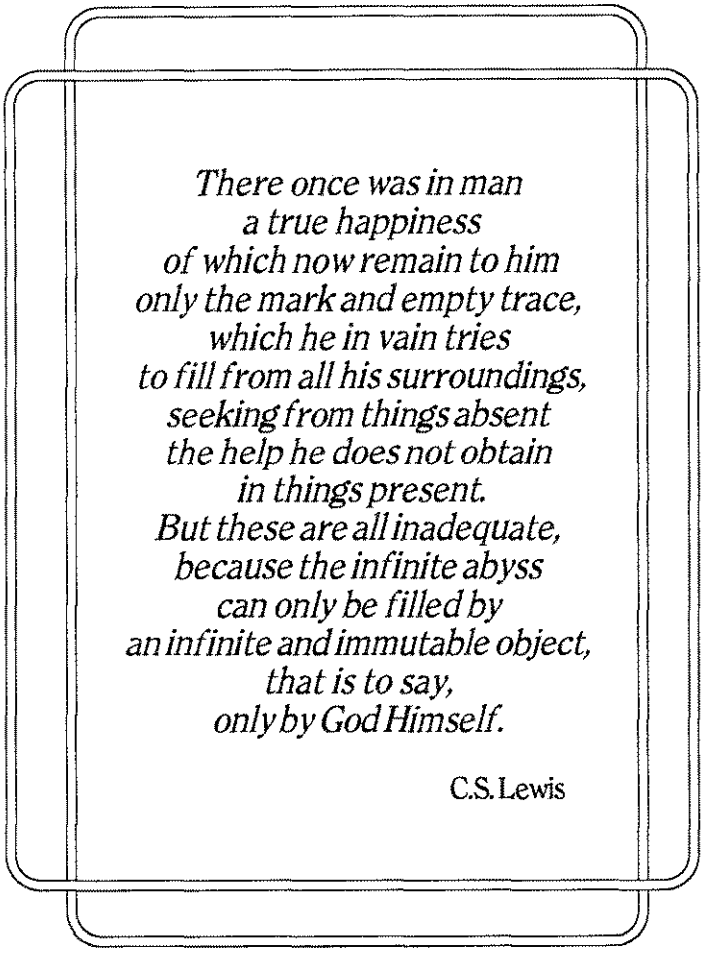
Workload - This refers to a description of an application's demand on a computer system. A workload can be thought of as a grouping of processes accomplishing similar tasks. An example might be the finance workload which could consist of AR, AP and Payroll processing.

> ***Performance Implications:*** *Without workload definitions and descriptions for a system, it is nearly impossible to perform accurate capacity planning. Also, workloads have differing service requirements. The more a workload demands of a resource, the more utilized that resource will be. This, in turn, affects not only processes associated with that workload, but other workloads as well.*

Users - A user is usually associated with online terminal activity. Users, in contrast to batch jobs, generally have less impact on system performance due to their think time between transactions. This is true unless, of course, an "online batch job" is being performed (compile, long report, etc.).

> ***Performance Implications:*** *System utilization will be in proportion to user activity. The more think time between transactions, the less CPU utilization will be observed. "Heads down" data entry users can have nearly as bad an effect on performance as batch jobs.*

You cannot bless the members of your family without meaningfully touching them. G. Smalley



There once was in man
a true happiness
of which now remain to him
only the mark and empty trace,
which he in vain tries
to fill from all his surroundings,
seeking from things absent
the help he does not obtain
in things present.
But these are all inadequate,
because the infinite abyss
can only be filled by
an infinite and immutable object,
that is to say,
only by God Himself.

C.S. Lewis

Appendix A

An Overview of Capacity Planning With Queueing Network Models

In this Appendix, I present ways modeling techniques can be used in predicting the future performance of an HP 3000. Modeling techniques can be used to help ensure that the existing hardware is truly out of gas. These techniques can provide you with some of the information you need to help you avoid over- or under-buying for a new system. Managers are often faced with questions like, "What will be the impact of a new application?" or " How do I know for sure how long an upgrade will last my company?" I present some of the problems involved in selecting an upgrade, and how to know for certain when an upgrade will be necessary. These times, many new, and often confusing, choices for hardware upgrades are available.

This area of capacity planning is particularly exciting for me. For some time now, I have had an interest in helping managers predict the future performance of their HP 3000 systems. In this Appendix I briefly cover some common models used for performance evaluation and capacity planning, but focus mainly on queueing network modeling. This area is a fairly immature science in the HP 3000 market, though it is quite developed in the larger IBM mainframe arena. There are quite a few commercial modeling products available for "big blue" systems, but only one for the HP 3000. If you are not faint-hearted with math and queueing theory, you could build a usable model yourself.

In this Appendix I describe some of the basics of the math involved and provide some examples. Suffice it to say, for these introductory comments, queueing network modeling is undoubtedly the most cost-effective and accurate way to predict the future performance of your systems.

Consider this comment by one capacity planning analyst:

> *"Significant savings, often millions of dollars, can result from capacity management software packages. As a result, senior management's views about capacity planning efforts are improving. Many companies are finding that the savings come from three major areas: maximization of computer resources, better planning for new computer resources in both what is needed and when it is needed, and increased productivity of both data center personnel and end users.* [27]

Definition and Importance of Models

Let's first turn our attention to a definition of queueing network modeling.

"A Queueing Network is a particular approach to computer system modeling in which the computer system is represented as a network of queues which is evaluated analytically. A network of queues is a collection of service centers, which represent system resources, and customers, which represent users or transactions. Analytic evaluation involves using software to solve efficiently a set of equations induced by the network of queues and its parameters". [28]

If you think of queue modeling in light of the preceding definition, it really is quite a simple concept. There are many day-to-day queue networks that we are involved with. I'm thinking of things like fast food restaurants or making a phone call from the east to west coast on Mother's day ("I'm sorry, all circuits are busy..."). Analysts in charge of strategic planning for these types of systems often utilize queueing network modeling to reduce their growth pains.

There are a few major benefits of using an analytic model to perform capacity planning. Queueing network models:

- Provide an improved understanding of the physical system.
- Handle non-linear interactions of many variables.
- Have proven to be the most accurate way to predict future capacity needs.
- Relate capacity, throughput and service.

It is very difficult to evaluate in your head the trade-offs of many different system relationships. Consider the effect on user response times if you were to place an interactive workload at a lower priority while at the same time adding a new application and applying staircased business growth rates for the next two years. About this time your brain goes tilt... But this is precisely the type of real-world questions that business planners have to answer. The answer to these questions and more is queueing network models.

Such modeling can provide significant information to financial analysts.

> *"Analytic modeling packages can be used to explore further when the next upgrade will be required as well as the life-expectancy of each upgrade. Cost of the equipment, when it will be needed, and most importantly, how long it will last, are the three factors needed by the financial analysts of the organization. The more advanced notice of equipment needs that the financial side of the organization receives, the more negotiation leverage it has with the hardware vendors. More accurate information on how long equipment is expected to last gives the financial side the information that it needs to determine how the equipment should be financed and depreciated."* [29]

Types of Performance Models

There are four types or levels of models available to make decisions about system performance. These are:

- **Implicit Models** - You probably do not realize you're using these. An example might be something as simple as, "A Series 937 is faster than my 922". Or, "The users begin to complain about response time just after lunch time; perhaps we should move that payroll batch job to after hours." You have simple models like those constructed in your brain. These are examples of empirically created models that we trade with one another. Implicit models represent a knowledge that you have acquired that takes into account such subjectives as the user population, application quirks, hardware configuration, management/political constraints, etc.

- **Rules of Thumb** (ROTs) - These typically consist of threshold metric information. An example of an ROT might be, "A sustained page-fault rate of greater than 30-per-second may be an indication of memory shortage." Overall, these models are fine for simple situations, but really do not address some of the tough futuristic business planning required of DP management these days.

- **Simple Queueing Models** - These models utilize relatively simple math (similar to some that I will be discussing shortly) to construct a picture of your system and then allow prediction. They are typically low-cost and come with low accuracy. Also, much skill is necessary since you'll probably be programming them yourself.

- **Detailed Queueing Theory** - These models are accurate and easy to use. Someone has built the accuracy and user-friendliness into the models. Subsequently, they cost more and they are usually commercial (versus shareware, etc.) packages.

Modeling Functions and Steps

Some of the questions that can be answered by a queue network model created for your HP 3000 are:

- What happens to response times if we upgrade?
- How much will our batch financials improve if we decrease the amount of CPU necessary for on-line order entry?
- Will the addition of a particular application six months from now kill our three-year plan for the expected life of the current system?
- At what date will our CPU be saturated given a 20 percent annual growth in orders?

Basically, models help answer most of the questions relating to the how, what, when, where and why of performance capacity planning. The actual steps involved in forecasting performance with a queueing model are as follows:

1. Define and set up workloads. This step requires a knowledge of your applications and much patience. You might have to re-define and re-measure them numerous times before they are right for modeling.

2. Run Data Collector on Host.

3. Perform analysis of collected data to find appropriate modeling windows and obtain estimates of service demands (CPU and Disk). Identify the peak period best for modeling.

4. Enter data into modeling formulas.

5. Validate initial model results with actual observations during analysis phase.

6. If validation is within acceptable limits, proceed to forecast. If validation is not good, you must consider the following:

a. Dispatcher queue aberrations (third-party queue-management tools), changes in queue overlap, multiple C or D queues. Modeling assumes no queue tricks.

b. Miscellaneous or unknown process wait-time for resources such as tape drives, father/son waits, data base locks, etc.

7. Create forecast of system and try various "what if" scenarios.

It is important to keep in mind that modeling is most accurate when considering relative changes in response time, throughput, and utilizations. That is, even though a model may speak of response times as two seconds now and four seconds nine months from now, the actual numbers may not be exactly what the users are experiencing. It is the relative change in response that is important.

The HP 3000 as a Queueing System

The behavior of a queueing system, as applied to the HP 3000, is really quite simple. Customers (transactions) arrive at the system from terminals or batch jobs. These customers then request service at the CPU, then at other devices. They must queue if those devices are busy. When their processing is finished, customers depart the system (back to terminal or batch spool). Figure A.1 illustrates a technical aspect of a queueing network model.

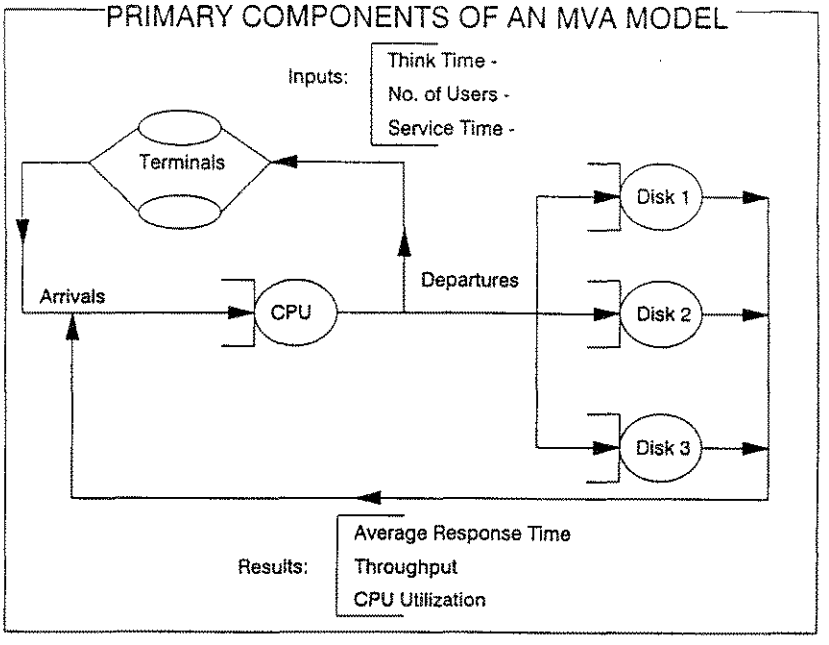


Figure A.1 - Queueing Network Model

Basics of Queueing Theory

If you don't like math, even basic math, you better skip the following section! However, if you can handle simple algebra, and if you are intrigued by the subject so far, you're in for a treat. I first lay out some formulas and then illustrate how simply queueing theory can be used to solve some basic problems.

Modeling Assumptions And Formulas

Single Server Model (M/M/1):

Assumptions:

A = Average Arrival Rate (per second)

S = Average Service Time (seconds)

U = Server Utilization (percent)

R = Average Response Time (seconds)

Q = Average Queue Length

Formulas:

R = S/ (1-U)

Q = U/ (1-U)

U = A * S (Utilization Law)

For Instance:

A = 10

S = .05 seconds

U = 10 * .05 = .5 = 50%

R = .05/ (1-.5) = .1 second

Q = .5/ (1-.5) = 1

Figure A.2 - Modeling Assumptions and Formulas

Given some of the basics in Figure A.2, we can calculate some pretty exciting answers to many otherwise unanswerable capacity planning questions. What HP 3000 IS folks care about most are things like utilizations, throughputs, and response times; predictions of these things, that is. By simply changing a few variables in such a model, you can come up with some very relevant information. Although it is very basic, the following example will help illustrate this point.

Simple Queue Model Example: XYZ Company's Order Entry Growth

The MIS director of XYZ company wants to know what will happen to its HP 3000 if 60 percent more data entry activity is added to the system, effectively raising the number of transactions per minute to an estimated 200. The current number of orders per minute is 125. The development team concludes that the appetite of the average order transaction is .27 CPU seconds.

Given this scenario:

- What is the current average CPU utilization and response time?
- What will the new average utilization and response time be?
- Will the current system be less than 95 percent utilized?
- Will the response times be less than the three-second target?

Let's use some of the basic math in Figure A.2 to answer some of these questions.

First we need to obtain the current CPU utilization and response time. Given the arrival rate of user requests and the service time of each transaction, we can do just that.

Transaction arrival rate:
A = 125 tran/min = 2.1 tran/sec

CPU service time per transaction:
S =.27 sec/tran

CPU Utilization:
U = A * S U = 2.1 * .27 U = 56.7%

So, the current CPU utilization is 56.7%. Now for the current response time:

R = S/(1-U)
R = .27/(1-.567)
R = .62 seconds

The current average response time of each user transaction is .62 seconds.

Now we can calculate the proposed impact of the above stated growth on this system.

Projected arrival rate:
A = 200 tran/min = 3.3 tran/sec S = .27 sec tran
the appetite of each transaction hasn't changed

Projected CPU utilization:
U = 3.3 * .27 U = 89%

Projected response time:
R = S/(1-U)
R = .27/(1-.89)
R = 2.45 seconds

The answers to the above questions are:

- YES, the CPU will be less than the 95 percent target (barely!)
- YES, the response times will be less than three seconds (barely!)

What happens if a new version of the order-entry application is proudly introduced with all the new enhancements you asked for, but there is a slight increase in the amount of CPU needed by each transaction in amount of a mere er... uh... 32 percent? Now what happens to service level targets? A little math will show that the CPU will not be able to handle this new load since the utilization goes above 100 percent.

You can see that even with simple queueing theory math, some real-life answers can be obtained. We obviously haven't taken into consideration a number of issues that cannot be ignored for the most accurate queue models. Some of these are:

- Multiple workloads.
- Priority interaction between workloads.
- The effect of other servers (disc drives, most notably).

More complex models require fairly complicated math. To illustrate the power of such models, I use Forecast/3000 Capacity Planner. Various "what if" situations are presented for a hypothetical company's growth estimates.

Capacity Model for a Typical Company

Let's use a hypothetical company, **Progress Corporation**, to demonstrate the power of queueing models for capacity planning.

Progress had three workloads initially, each with the following characteristics:

Workload	#Users	Priority	CPU Hog
Order Entry	28	High	Medium Heavy
AR-AP	9	High	Medium
Shipping	16	Medium	Light
Payroll (fut.)	4	High	Heavy

Table A.1 - Workload Characteristics

Progress has just upgraded from a Series 925 HP 3000 to a Series 947. They experienced considerable performance improvement. The users were ecstatic. However, knowing that this was only a temporary euphoria, the system manager decided to implement a capacity plan. He felt that such a plan was warranted based on some pretty aggressive business growth plans and an additional application that management wanted to implement in about eight months.

Figures A.3, A.4, and A.5 show the initial state of their 947. They wanted to be around 50 percent CPU utilization. As you can see, their response time targets for each workload (less than 2 seconds) are attained with room to spare.

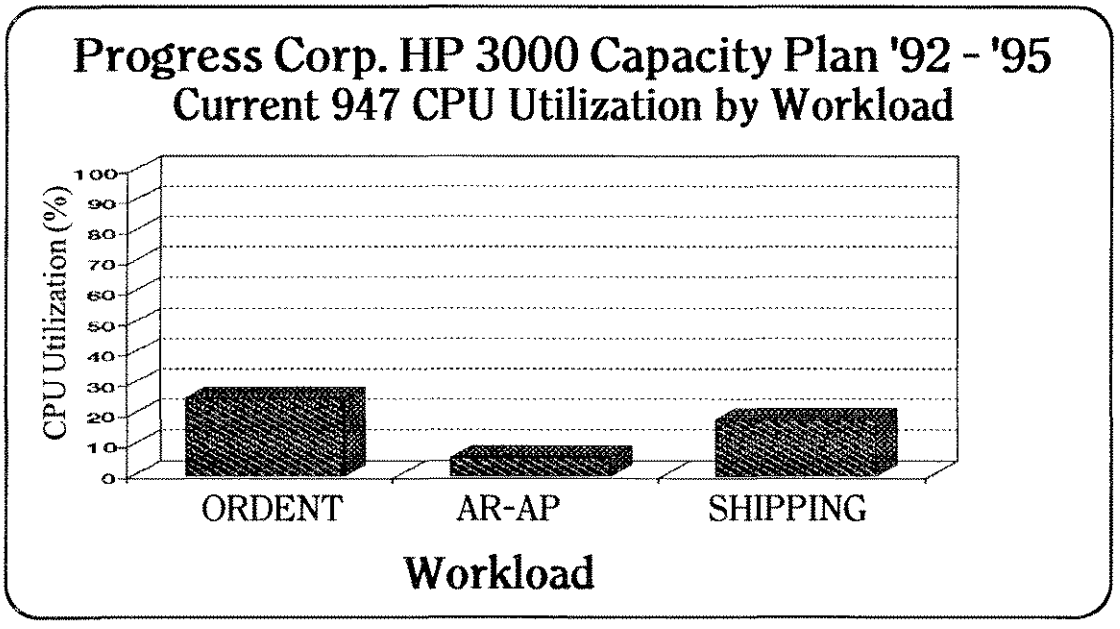


Figure A.3 - Initial 947 Workload CPU Utilization

Of the 11 strongest earthquakes in the U.S. in this century, four have occurred in 1992. And six--more than half--have occurred in the last three years.

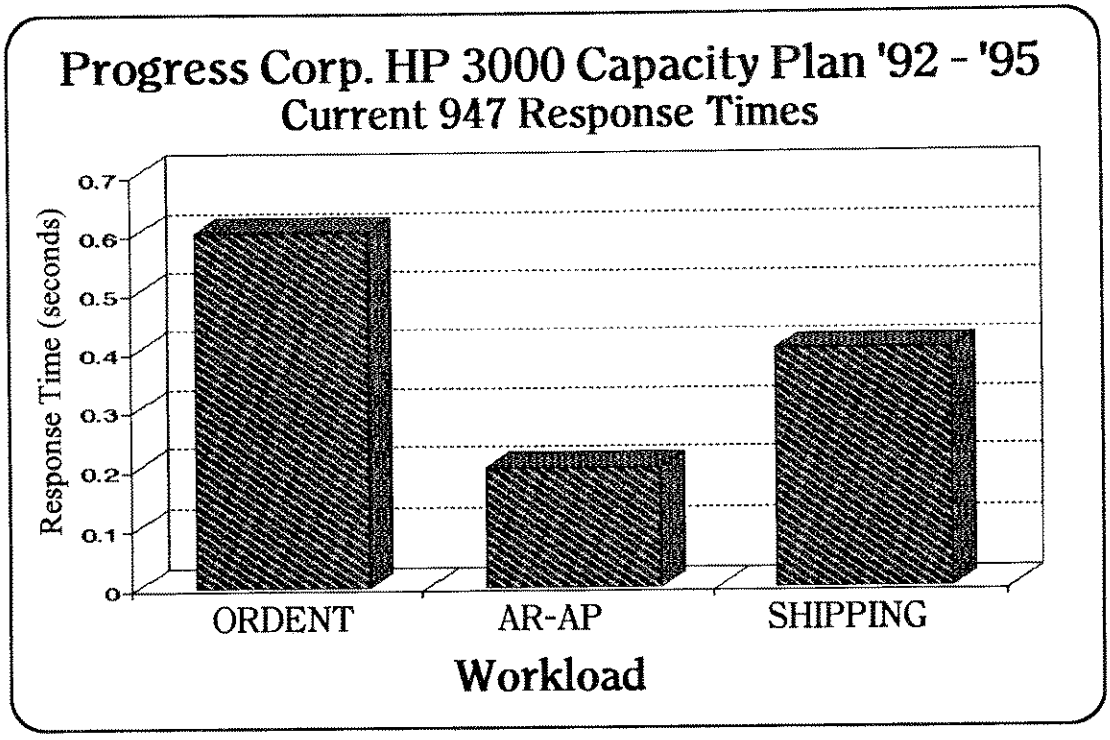


Figure A.4- Initial 947 Workload Response Times

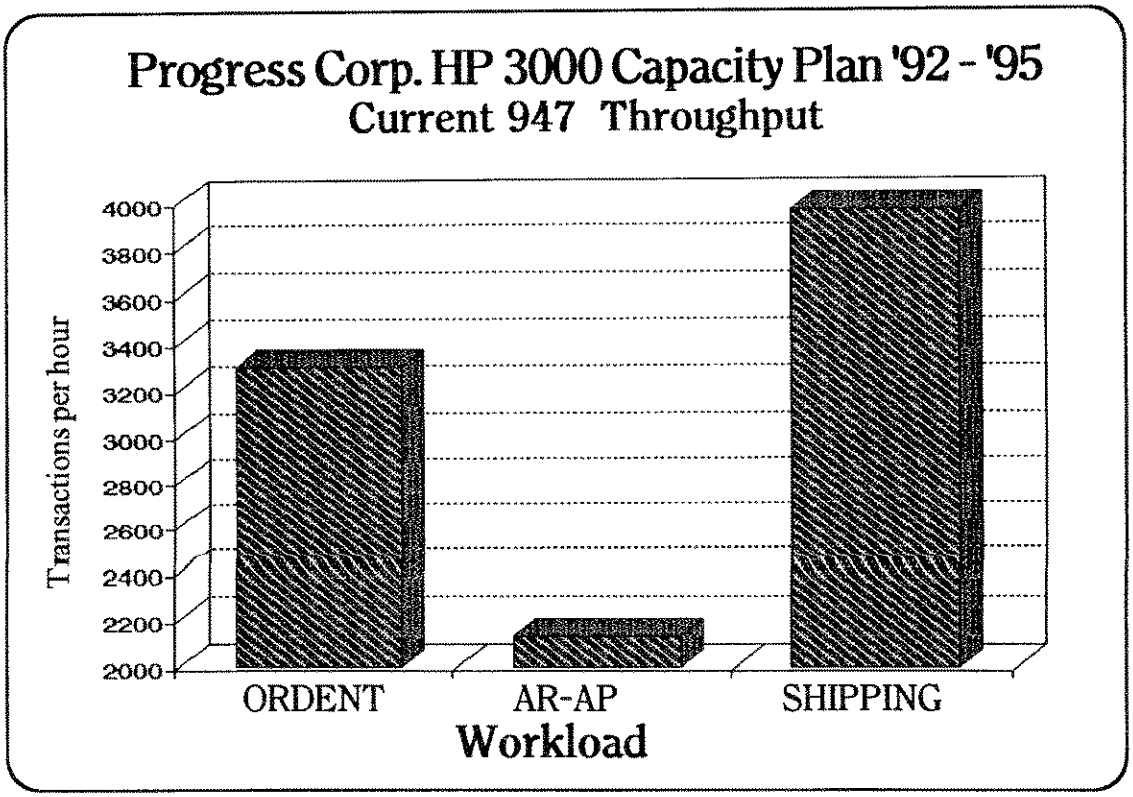


Figure A.5 - Initial 947 Workload Throughput

The Company's Growth Plan

Progress hoped that the new system would last until the end of 1994. Their growth plans for the business were set to take off in about six months at a fairly steep rate then level off again. Figures A.6 and A.7 show their growth and resulting CPU utilization along with response times. More notes? The CPU does not pierce the 85 to 90 percent level until sometime after 1994. Notice how response times (Figure A.7) for all workloads stay under two seconds until shortly after June 1994. The shipping workload experiences a "knee in the curve" phenomenon, which shows its responses piercing well above the service level target of two seconds. This is due to the fact that the shipping application is treated by MPE (on this system) as a lower priority (see Table 1.1). This means that, in a pinch (CPU getting scarcer), shipping will receive less service.

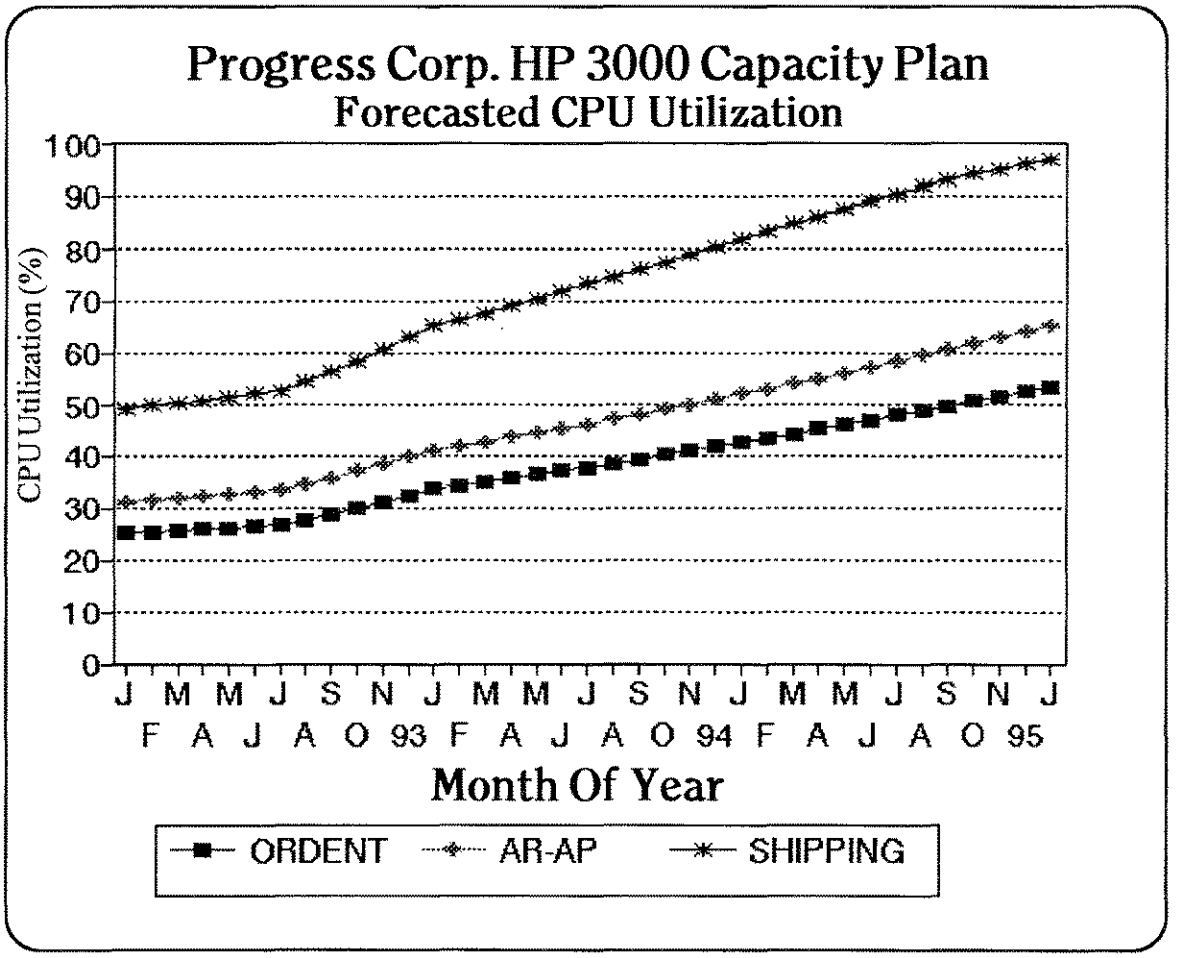


Figure A.6 - Three Year 947 CPU Forecast

Wine is a mocker, strong drink a brawler, And whoever is intoxicated by it is not wise. Proverbs 20:1

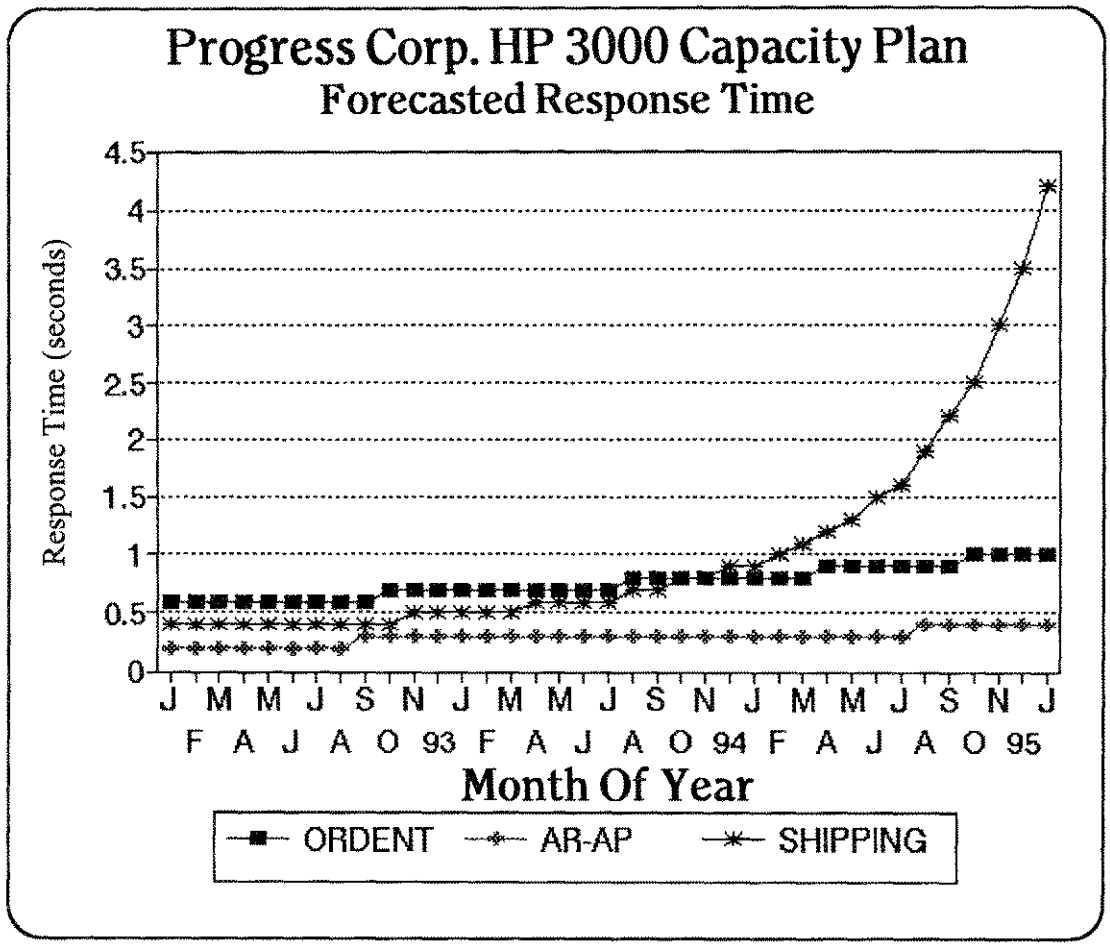


Figure A.7 - Three Year 947 Response Time Forecast

The Plot Thickens

What about that payroll application management wanted to implement in eight months? As an unknown quantity, the system manager was a bit nervous about the impact of this application, and rightly so. If you have little or no control over the source code, new applications can produce incredible surprises. Management said that it couldn't be that big of a deal especially since there was only going to be about four users or so. How could it impact the system that much?

When you look at Figure A.8, you may first think some kind of tumor grew at the eight-month mark! That tumor is the payroll application. Does its presence affect the don't-die-till-'95 plan? YES! How much an application will affect a system is purely an issue of the application's appetite. Queueing models portray these surprises, and more, quite nicely.

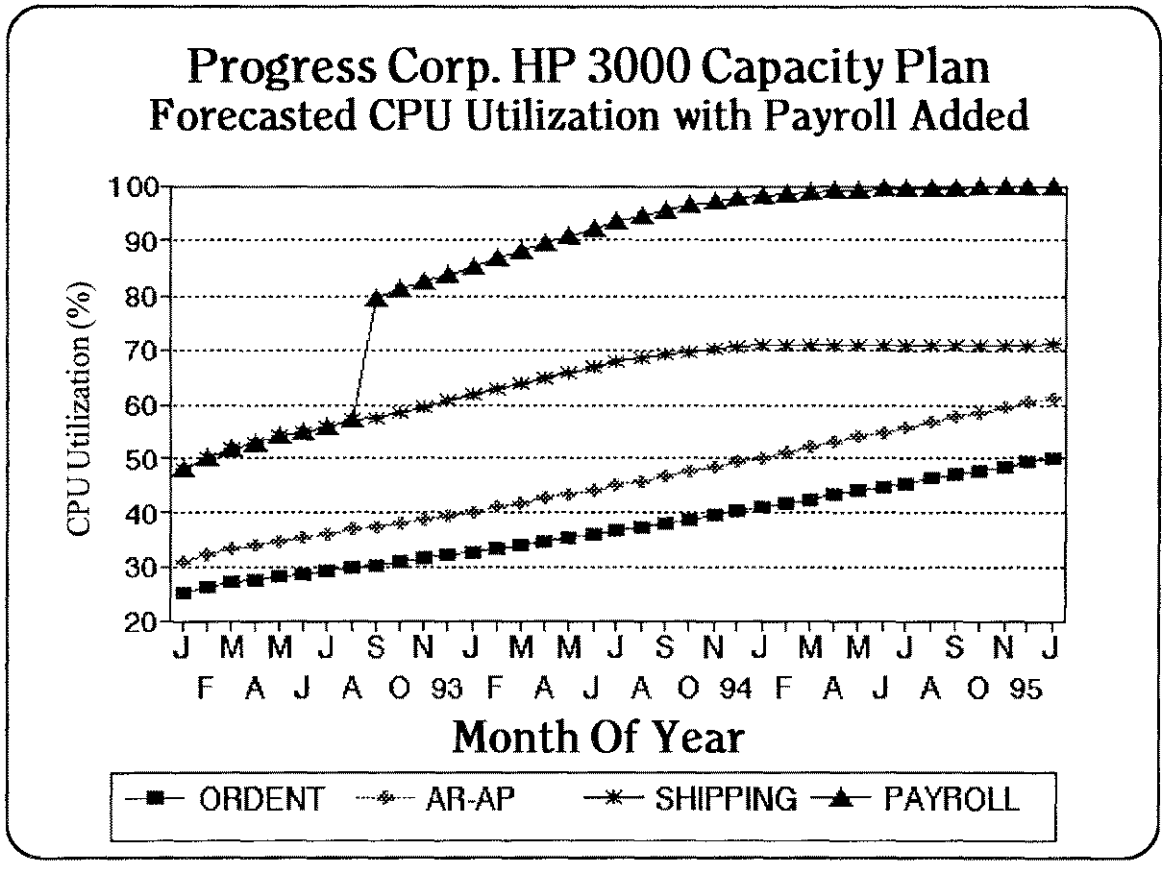


Figure A.8 - CPU Utilization With Payroll Added

For **Progress**, the CPU approaches saturation right around the January '93 time frame.

Figure A.9 illustrates the severely impacted response times for both shipping and payroll. Right out of the gate, payroll's response time is not so great!

O taste and see that the Lord is good; How blessed is the man who takes refuge in Him! Psalms 34:8

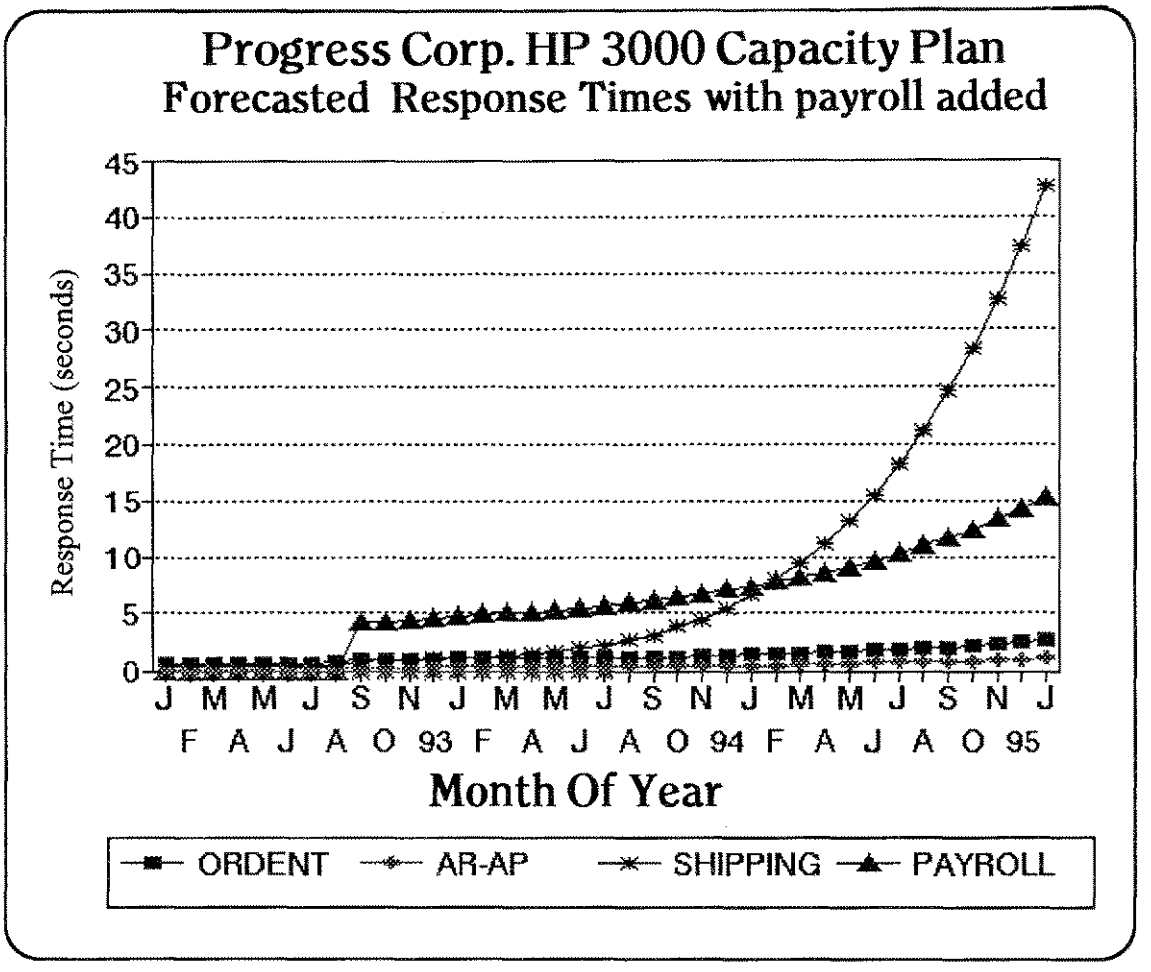


Figure A.9- Response Times With Payroll Added

Programmers to the Rescue

At a staff meeting the manager presented the above graphs and asked if there were any ideas how to last longer on the 947 even with the payroll implementation. A couple of programmers mentioned that there had been some known performance bugs with the order-entry application. Spending about a month-and-a-half would reduce the amount of CPU per order transaction by about 35 percent. This might help.

After some "what-if "modeling, the manager decided it would help enough to give them some more time. Figures A.10 and A.11 are the result of modeling 35 percent less CPU for each order. The net effect of this effort is that it would provide about seven more months of useful life for the system as a whole. Such modeling will allow you to determine the effect on key performance indicators before investing the actual time and effort necessary to implement them.

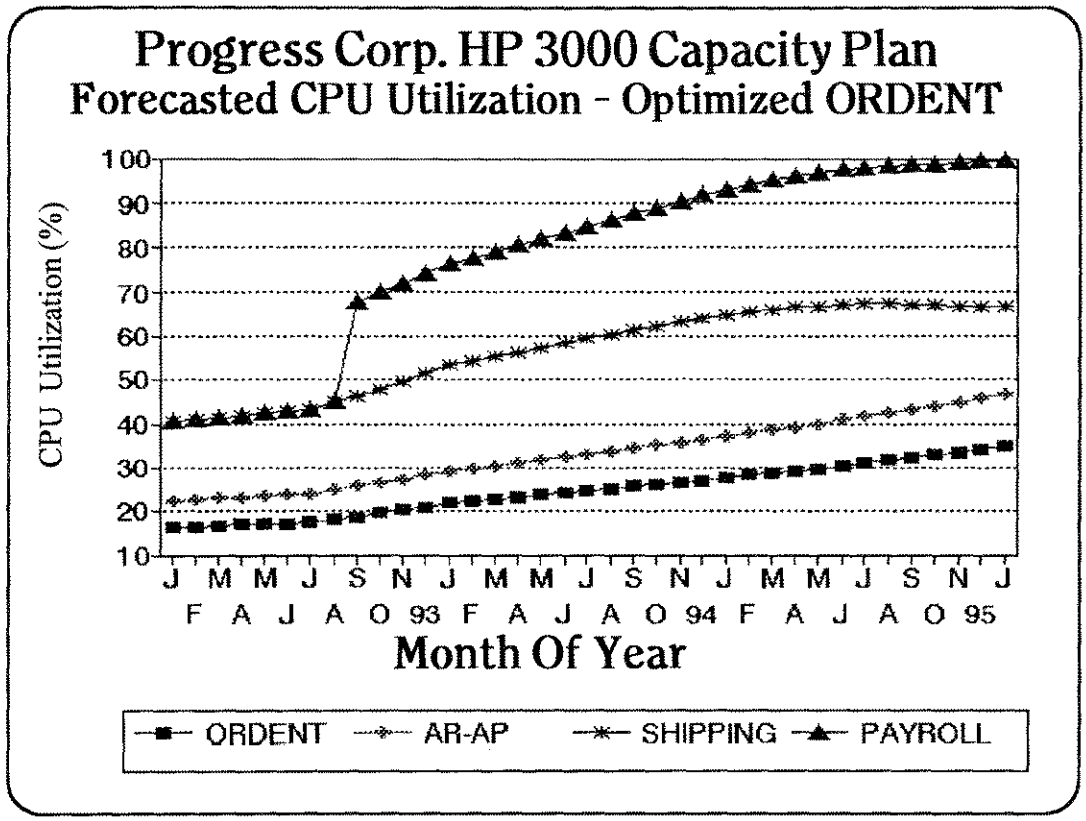


Figure A.10 - CPU Utilization After Optimizing Order Entry

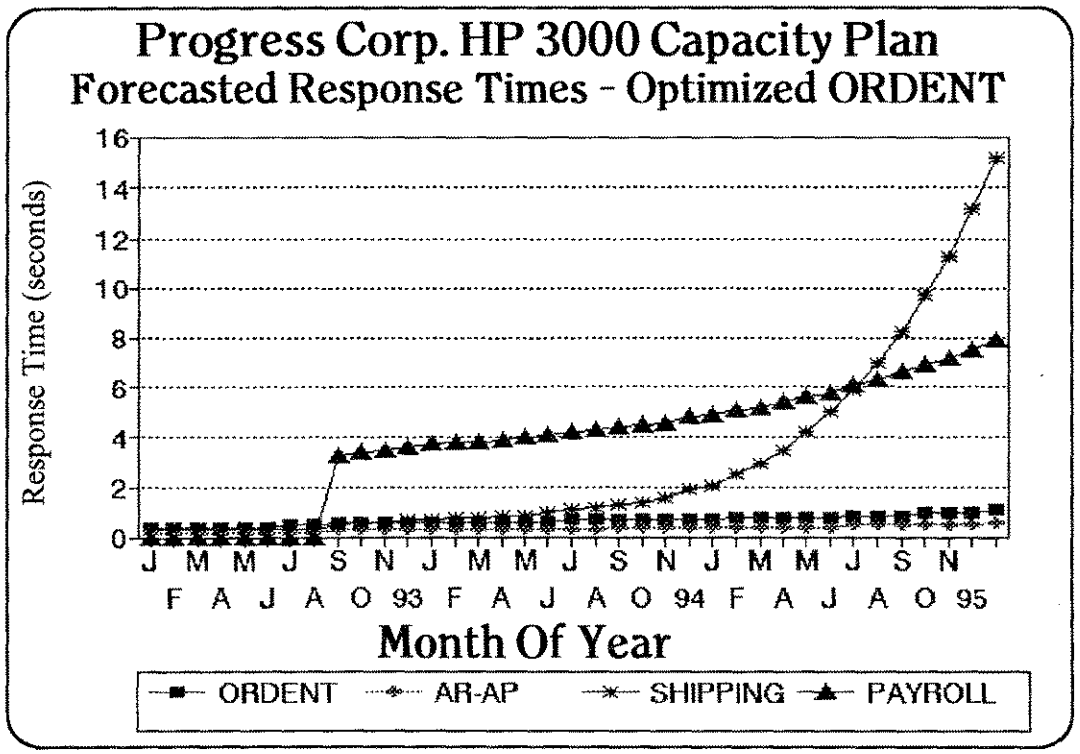


Figure A.11 - Response Times After Optimizing Order Entry

About 21 billion gallons of sewage are produced in the U.S. each day - an average of about 100 gallons per person.

A Last Resort

Progress thought they had better also consider the impact of upgrading to a larger system. They chose to model the effect of a Series 967 HP 3000. Figures A.12 and A.13 show how much improvement in CPU utilization and response times could be expected with an upgrade to the 967.

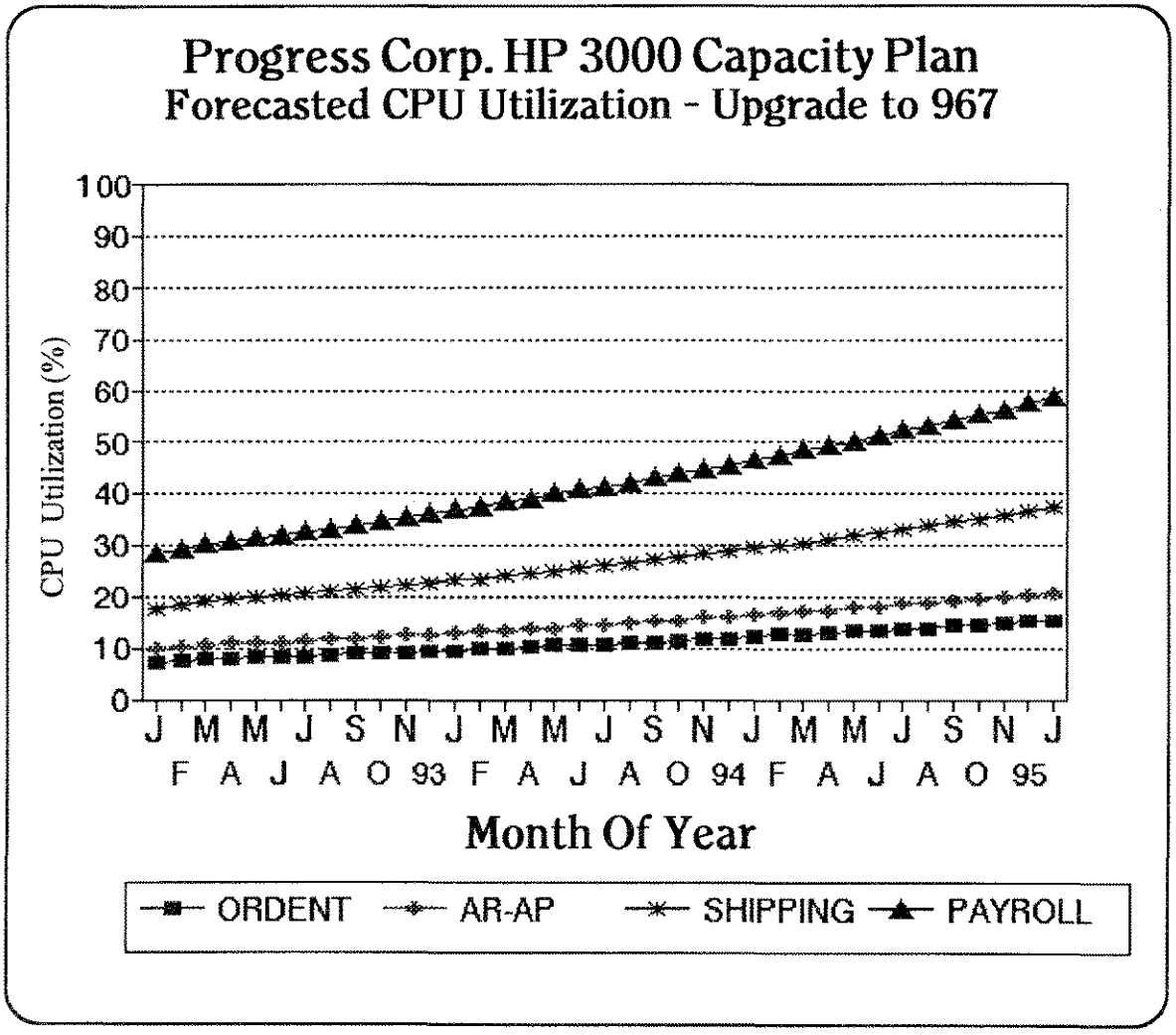


Figure A.12 - CPU Utilization After 967 Upgrade

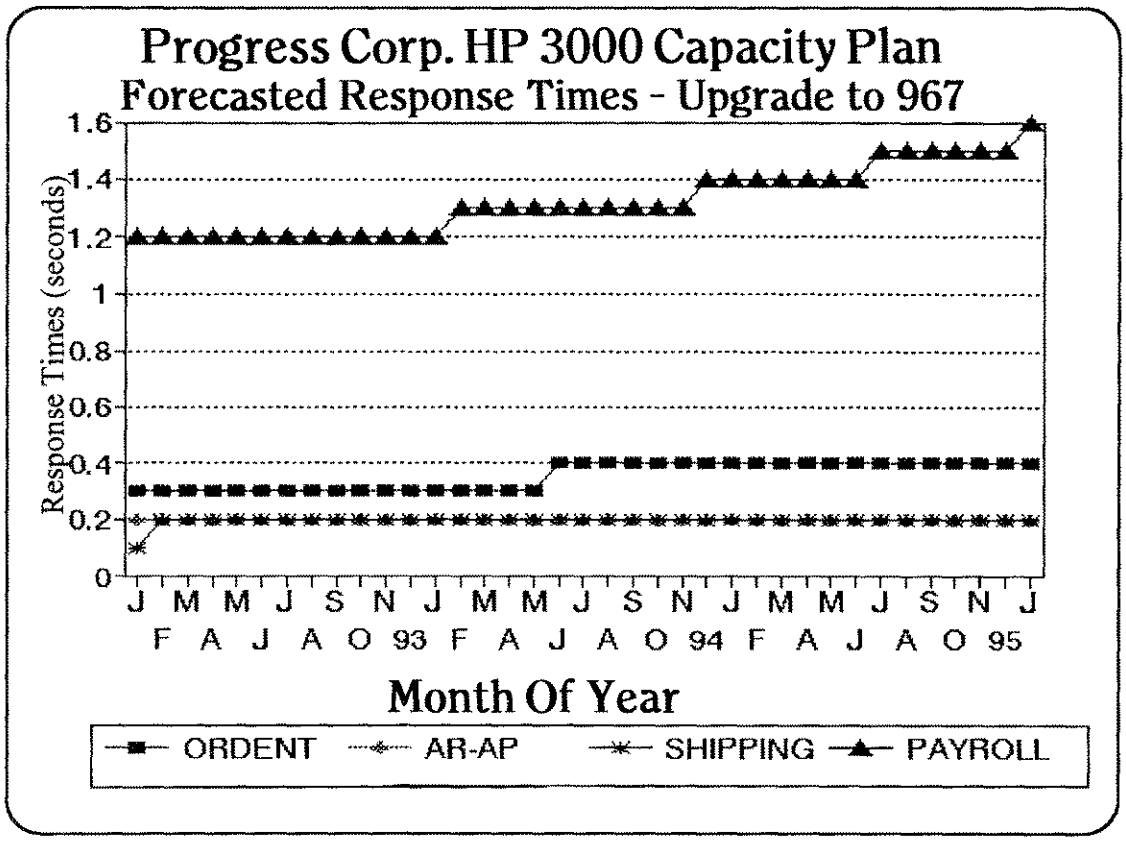


Figure A.13 - Response Times After 967 Upgrade

Conclusion

I hope this brief overview of queueing network models has been helpful in showing you a few things:

- Models are increasingly finding a place in the HP 3000 market. This is mostly due to the vastly expanding options within the HP 3000 product line.
- Models are extremely flexible and powerful in their ability to forecast. A model can predict the effect of numerous changes to an application, upgrade scenarios, downsizing, etc., without investing time and/or money (apart from the time it spent playing with a model on a PC).
- Models can be complex to "home-brew" but commercial packages are fairly easy to use.
- Models allow IS managers to chart a fairly predictable course that will allow them to know way ahead of time when the next performance "bump" will be encountered.

The fundamental basis of this nation's law was given to Moses on the Mount. The fundamental basis of our Bill of Rights comes from the teachings which we get from Exodus and St. Matthew, from Isaiah and St. Paul. I don't think we emphasize that enough these days. If we don't have the proper fundamental moral background, we will finally wind up with a totalitarian government which does not believe in rights for anybody except the state.

Harry S. Truman

Appendix B

MPE Performance Diagnosis Pulse Points Charts

These charts are intended to be used for HP 3000 performance diagnoses. I advise you to use them in conjunction with the information presented in Chapter 6.

It is also important to note that the "color value" (green, yellow, red) of various indicators on these charts could differ, depending on certain conditions. Keep in mind that just because one memory indicator points to a red condition, this doesn't necessarily mean that the system is saturated. The critical thing to remember is that it is the consistent indication over a period of time that substantiates a resource bottleneck.

Also, if your experience disagrees with data on these charts, I'd like to hear from you, especially if you have found other pulse points not listed to be of value.

INDICATOR	GREEN	YELLOW	RED	COMMENTS
CPU:				
Total Busy (%) P+C+M+G+I	<50*	50-75	>75	High-priority processing only Low-priority batch excluded (*it depends)
CPU Queue Length	<5	5-15	>15	Exclude "perpetual" batch jobs from consideration; check high job limit
ICS (%) ATPs	<10	10-15	>15	
ICS (%) ADCCs	<15	15-25	>25	
Software Caching (%)	-	-	-	Observed normal range 2-8% per disc drive
Memory:				
CPU Pause/Memory (%)	<1.5	1.5-3	>5	
CPU Pause/User I/O & Memory (%)	<1.5	1.5-3	>5	Both disc and memory
Clock Cycle Rate (ms)	>10000	2000-10000	<2000	Milliseconds per cycle, somewhat variable by CPU model
Swaps/Launch Ratio	<.1	.2-.3	>.3	
CPU Busy on Memory Management (%)	<8	8-12	>12	Measurement of memory activity
Cache Memory (%)	>20	10-20	>10	Busy system assumed
CPU busy on Global Garbage Collection (%)	<2	2-3	>3	Must have software Caching off to see
DISK I/O:				
CPU Pause for User I/O (%)	<5	5-10	>10	With caching on
CPU Pause for User I/O (%)	<10	10-15	>15	With caching off
I/O Rates (per second)	<10	10-20	>20	Per drive (caching on)
Avg. Queue Length	0-.5	.5-1.0	>1.0	
%I/Os Eliminated	>55	45-55	45	
TurboIMAGE:				DBLOADING/HOWMESSY TOOLS
Master x Secondaries	<20	20-30	>30	Depends on inefficient pointers
Secondaries Inefficient Pointers	<8	8-13	>13	
Master Maxblocks	<10	10-40	>40	
Detail Elongation	<1.5	1.5-3.0	>3.0	Most important overall indicator of data base performance

Figure B.1 - Classic Pulse Points

Every man's way is right in his own eyes, But the Lord weighs the hearts. Proverbs 21:2

INDICATOR	GREEN	YELLOW	RED	COMMENTS
CPU:				
Total Busy (%) (a+b+c+m+disp+isc)	<50*	50-85	>85	High-priority processing only Low-priority batch excluded (*it depends!)
CPU Queue Length	<5	5-15	>15	Exclude "perpetual" batch jobs from consideration; check high job limit
ICS + Dispatcher (%)	<10	10-15	>15	
CPU CM (%)	<10	10-50	>50	Subjective
AQ + BQ (%)	<5	5-8	>8	Operating system version Dependent
Process Preemption (%)	<20	20-40	>40	On most active processes; % of wait states
Memory:				
CPU busy on memory management (%)	<8	8-12	>12	Very reliable indicator
Page Fault Rate/Sec	<10	10-25	>25	Possible CPU model dependency
Library Fault (%) (as % of Page Faults)	<10	?	?	LIBRARY =XL,NL,SL
Swaps/ Launch	<.4	.4-.8	>.8	
Clock Cycles/Hour	<10	10-25	>25	
DISK I/O:				
CPU Pause for User I/O (%)	<5	5-15	>15	Reflects data locality; can reflect I/O efficiency
Read Hit %	>95	85-95	<85	Measurement of locality
Disc Queue Length	<.5	.5-1.0	>1.0	Overall average; Discount XM spikes (pre 3.0)
Disc I/O Rate/Sec	<10	10-25	>25	Per drive
TurboIMAGE:				Use DBLOADING/HOWMESSY
Secondaries *Inefficient Pointers	<8	8-15	>15	
Master Secondaries	<20	20-30	>30	Depends on inefficient pointers
Master Maxblocks	<10	10-40	>40	
Detail Elongation	<1.5	.5-3.0	>3.0	
Miscellaneous				
CM/NM Mode Switches (per second)			>200 - small >500 - large	Ranges vary by processor model
NM/CM Mode Switches (per second)			>25 - small >75 - large	Ranges vary by processor model

Figure B.2- MPE/iX Pulse Points

He who would be great must be fervent in his prayers, fearless in his principles, firm in his purposes, and faithful in his promises. McKenzie

Everyone who is seriously involved in the pursuit of science becomes convinced that a spirit is manifest in the laws of the universe--a spirit vastly superior to that of man, and one in the face of which we with our modest powers must feel humble.

Albert Einstein

Appendix C

Taming The HP 3000 (Volume I) Overview Of Topics Covered

This appendix is a summary of the first five chapters of Taming The HP 3000 Volume I-Over 101 Ways to Monitor, Manage and Maximize Performance on the HP 3000. While the majority of these improvement ideas apply to both Classic and Spectrum systems, items without any asterisks apply only to classics. A "?" means that I do not have a concrete feeling for whether an idea will have much effect on an XL system. A double asterisk means that an idea may be especially appropriate to MPE/iX.

To get the entire explanation for each idea, you will want to procure your own copy of Taming The HP 3000 Volume I. It is available by calling **(503) 327-3800 or (503) 926-3800.**

Chapter 1 - Load Balancing Techniques

*	#LB1	Schedule the Quantity of Jobs and Sessions Intelligently
*	#LB2	Schedule the Types of Batch Jobs Intelligently
*	#LB3	Defer All Batch Jobs to After Hours
*	#LB4	Avoid/Reduce Session Logons and Logoffs
*	#LB5	Stagger Programmer/User Work Hours
*	#LB6	Move Online Programs Which Produce Paper to Batch
*	#LB7	Use a Batch Job Scheduling System to Schedule Jobs
*	#LB8	Use the Simple Scheduling Facility of the MPE STREAM Command to Schedule Jobs
*	#LB9	Avoid/Reduce Running Programs
*	#LB10	Move Program Development Onto a Separate CPU
*	#LB11	Move Graphics and Word Processing Applications to a Separate CPU or PCs

* #LB12 Avoid Simultaneous Runs of Programs which Access the Same Data Files
* #LB13 Force PIG/3000 Programs into Lower Sub-queues by Writing Programs which Intercept Associated RUN Commands

Chapter 2 - Operational and Management Considerations

#OM1 Allocate Frequently Used Programs
#OM2 Enable AUTO-ALLOCATE
* #OM3 Use UDCs to Avoid Unnecessary Keystrokes
* #OM4 Limit the Number of UDCs Enabled Per Level
* #OM5 Keep Appropriate UDC Information at an Appropriate Level
* #OM6 Prohibit Complex UDCs at the System Level
* #OM7 Attempt to Limit one UDC File per Level
* #OM8 Limit the Use of File Equates Within UDCs
** #OM9 Provide Ailing Batch Jobs More CPU Attention by Overlapping the CS and DS Sub-queues
** #OM10 Use the TUNE Command to Penalize Heavy Online CPU Hogs
* #OM11 Use the TUNE Command to Favor short Transactions
* #OM12 Sep Up a Game Plan for Disc Free Space Management
#OM13 Use the VINIT Utility to Compact Disc Drives
* #OM14 Perform a Full System RELOAD to Compact Disc Drives
#OM15 Perform a RECOVER LOST DISC SPACE to Maximize Disc Freespace
* #OM16 Build Large Dummy Files to "Hide" Disc Space
* #OM17 Inflate Virtual Memory to Provide for Future Disc Space Needs
#OM18 Locate and Shrink Files Which Contain Wasted Space
* #OM19 Facilitate Mass File Purging with MPEX
#OM20 Disallow Indiscriminate use of FREE5
* #OM21 Restrict the Use of PURGEGROUP and PURGEACCT
* #OM22 Use Faster Copying Utilities than FCOPY
* #OM23 Use KLA/EXPRESS to Improve System Performance
* #OM24 Implement a Faster Backup Scheme then MPE's Store
* #OM25 Remove IA Capability from CPU Hog Programs

Chapter 3 - Disk I/O Optimization & Reduction

	#IO1	Use MPE Disc Caching
	#IO2	Experiment with the BLOCKWRITE Parameter of the CACHECONTROL Command
	#IO3	Alter the Sequential Fetch Quantum of the CACHECONTROL Command
	#IO4	Alter the Random Fetch Quantum of the CACHECONTROL Command
	#IO5	Selectively Enable/Disable Caching on Specific Disc Drives
	#IO6	Use the Program DCO (Disc Cache Optimizer)
	#IO7	Use an Appropriate Blocking Factor for Each File to Reduce Physical I/O
	#IO8	Minimize File Allocation Time and Wasted Disc Space by Resetting the EOF of Files Which have Empty Space in Them
	#IO9	Block UDC Files with a Large Blocking Factor
*	#IO10	Use a Sorting Tool that is More Efficient then SORT/3000
	#IO11	Utilize DISC RPS when Appropriate
*	#IO12	Utilize Disc Controller Caching When Appropriate
*	#IO13	Spread Files Across Discs
*	#IO14	Place Complementing Files on Separate Drives
*	#IO15	Insure I/O Locality by Placing Heavily Accessed Files Close Together on Disc Towards the Front/Center of the Drive
*	#IO16	Insure Optimal Disc Speed by Maintaining A Cool Temperature in the Computer Room
*	#IO17	Keep VPLUS Forms in a Single File

Chapter 4 - Image/3000 Performance

*	#IM1	Re-pack Detail Sets After Mass Deletions
*	#IM2	Choose an Optimal Capacity for Master Sets
	#IM3	Select an Optimum Block Size
?	#IM4	Recalculate Blocking Factors When Upgrading From IMAGE to TurboIMAGE
?	#IM5	Choose an Optimal Number of Image Buffers
	#IM6	Upgrade (downgrade?) to TurboIMAGE
*	#IM7	Utilize AUTODEFER Mode When Appropriate
*	#IM8	Split Large Databases into Multiple Smaller Ones
*	#IM9	Perform Some Maintenance Activities After Hours
*	#IM10	Spread IMAGE Sets Across Discs to Minimize Disc Drive Contention
*	#IM11	Disable QUERY's Locking Mechanism
*	#IM12	Forbid Serial Searches of Any Kind for Online Activity
*	#IM13	Create Paths for Frequently Accessed Items

Doing your best is more important than being the best. Yes, desire is the ingredient that makes the difference between an average performer and a champion. Zig Ziglar

* #IM14 Minimize the Number of Paths in the Most Actively Accessed Sets
* #IM15 Put Much Thought into Which Path is the Primary One
* #IM16 Consider Utilizing DBUPDATE in Lieu of DBPUT and DBDELETE
* #IM17 Utilize Integer Keys when Appropriate
* #IM18 Use Byte Keys Unless Integer Ones are Appropriate
* #IM19 Minimize the Use of Sort Chains
* #IM20 Use DBGET Mode 1 to Initialize Field Lists
* #IM21 Minimize DOPENs
* #IM22 Use the "*" in Image Lists Whenever Possible
* #IM23 Use DBLOADING to Monitor Your Databases
* #IM24 Utillize HOWMESSY to Monitor Database Efficiency
* #IM25 Use DBGENERAL's Diagnostics to Track Database Performance
? #IM26 Use the IMAGE Profiler
* #IM27 Implement OMNIDEX for Rapid Database Searching

Chapter 5 - Hardware and Configuration Considerations

* #HC1 Upgrade the Central Processing Unit (CPU)
* #HC2 Distribute Major Applications Among Separate CPUs
* #HC3 Add More Main Memory
* #HC4 Obtain Another Disc Drive Upgrade to Faster Disc Drives
* #HC5 Upgrade to Faster Disc Drives
#HC6 Upgrade Slave Disc Drives to Masters
#HC7 Add General I/O Channels (GICs) if You are GIC Limited
#HC8 Add an Inter-Module Bus (IBM)
#HC9 Upgrade from ADCCs to ATPs
* #HC10 Distribute I/O Devices Over as Independent Path as Possible
#HC11 Spread Discs Over GICs Using a Weighting Method
#HC12 Specify an Appropriate Number of "Disc" Classes per Disc Drive
#HC13 On Certain Systems, Keep Device Class DISC OFF if the System Disc (LDEV 1)
#HC14 "Hide" Memory as a Reserve for Peak Periods

Bibliography

Research References

[1] Sieler, Stan. "MPE XL and Performance: Not Incompatible", *Proceedings of the Interex HP Users Conference, San Francisco,* (1989), pp. 4238-1 through 4238-13.

[2] Proverbs 24:10.

[3] Sternadel, Dan. "Capacity Planning", *Interact Magazine,* (August 1991), pp. 112-125.

[4] Allen, Arnold. *Probability, Statistics and Queueing Theory with Computer Science Applications, Second Edition* (San Diego, California: Academic Press, 1990).

[5] Lazowska, Edward. *Quantitative System Performance,* (Englewood Cliffs, New Jersey: Prentice-Hall 1984).

[6] Lund Performance Solutions (1992). *Forecast Capacity Planner reference manual.*

[7] Dean, Paul. "System Wizardry", *Interact Magazine,* (August 1988).

[8] *IS Capacity Management Handbook Series, Volume 1,* (Milpitas California: The Institute for Computer Capacity Management, 1990).

[9] Lancaster, William. *"Managing Performance", HP Chronicle,* (April 1992).

[10] Watson, Cheryl. "The Three Phases of Service Levels", *Mainframe Journal,* (July 1989), pp. 42-43.

[11] Ibid., p.82

[12] Cf. *Quantitative System Performance,* op, cit.

[13] Chambers, Mike. "The Evolving MPE XL Dispatcher: Axioms and Options", *Interact Magazine*, (August 1991), pp. 62-75.

[14] Hornsby, Michael. "$Locality-Memory Requirements for MPE XL", *Interact Magazine*, (September 1989), pp. 98-106.

[15] Caroll, Bryan. "Performance Analysis on MPE XL Systems", *Interact Magazine*, (April 1990), pp. 44-45.

[16] *IS Capacity Management Handbook Series, Volume 1*, (Milpitas California: The Institute for Computer Capacity Management, 1990).

[17] Smead, Steven. "Getting the Most Performance From MPE V", *Interact Magazine*, (June, 1992), pp. 69-83.

[18] Goertz, Jason. "A Technical Evaluation of HP Precision Architecture" (Part 1 of a 2 part series), *Super Group Magazine*, (June 1992).

[19] Hansen, Robert. "New Optimizations for PA-RISC Compilers", *Hewlett-Packard Journal*, (June 1992).

[20] Volokh, Eugene. "The Truth About MPE Disc Files", *Interact Magazine*, (September 1989), pp. 118-140.

[21] Cf. *Getting The Most Performance From MPE V*, Op. Cit.

[22] Smith, Sid. "Making It Fly, Optimizing Applications for MPE XL", *Proceedings of Interex HP Users Conference, San Francisco*, (1989), pp. 4061-1 through 4061-9.

[23] Cf. *MPE XL and Performance: Not Incompatible*, Op. Cit.

[24] Pierson, Scott. "How to Win Memory and Influence CPU: A look at MPE XL System Performance", *Proceedings of the Interex HP Users Conference, San Diego*, (1991), pp. 3134-1 through 3134-19.

[25] Cheng, Tu-Ting. "TurboIMAGE XL" *Interact Magazine*, (March 1991), pp. 99-110.

[26] "Standard Specification," *TPC Benchmark*, (1989).

[27] Porter, Dave. "Modeling, The Key to Effective Capacity Planning", *Technical Support*, (August 1988). p. 30.

[28] Ibid. p. 31.

[29] Ibid. p. 31.

Vendor References

Adager - Sun Valley, Idaho, (208) 726-9100. **ADAGER - Database utility.**

BradMark Computer Systems - Houston, Texas (713) 621-2808. **DBGENERAL - Database utility.**

Design/3000 - Salem, Oregon. (503) 585-0512. **JMS/3000 - Job scheduler.**

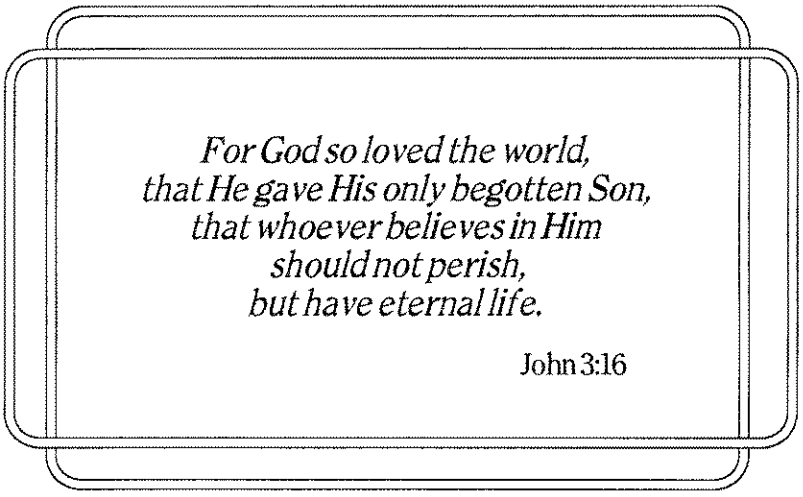
Kelly Computer Systems - Mountain View, California (415) 960-1010. **XL/3000 RAMDISC, Memory.**

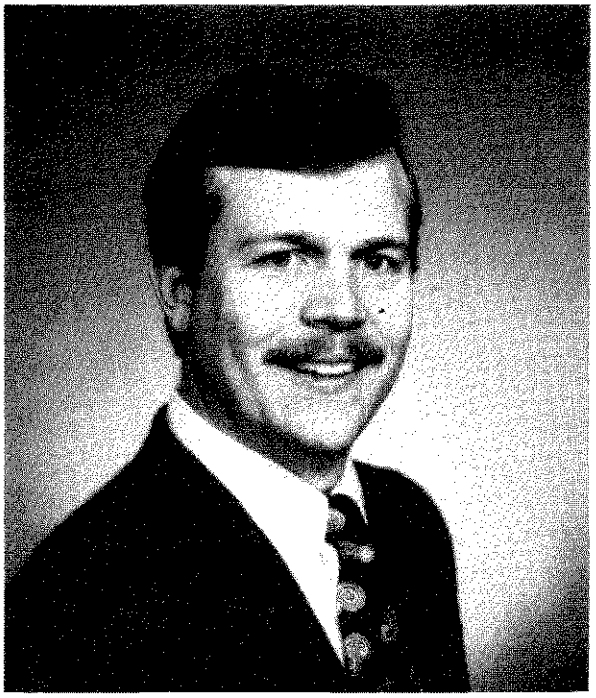
Lund Performance Solutions - Albany, Oregon. (503) 327-3800. **SOS/3000 Performance Advisor - Performance monitor; Performance Gallery - Graphical performance analyzer; Q-Xcelerator Resource Manager - CPU Resources Manager; FORECAST/3000 Capacity Planner - Capacity planning package.**

Operations Control Systems (OCS) - Palo Alto, California (415) 493-4122. **OCS/3000 - Job scheduler.**

Running Mate - Sacramento, California (800) 824-9046. **SORTMATE - Sort Utility, IOMATE - Data extraction.**

VESOFT - Los Angeles, California (213) 282-0420. **MPEX - Extended MPE utility.**


For God so loved the world,
that He gave His only begotten Son,
that whoever believes in Him
should not perish,
but have eternal life.
John 3:16



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 (Summit Information Systems), 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 HP 3000 journals and magazines. Robert has provided technical support to hundreds of HP 3000 sites throughout the United States. Currently he is the president of Lund Performance Solutions, the leader in HP 3000 performance software, training, and consulting. He and his wife and three daughters live in rural Oregon.

** WANTED! HP 3000 Performance War Stories **

If you would like to discuss matters relating to material in this book, feel free to call Robert at (503) 327-3800. Also, if you have any "war stories" or interesting facts regarding HP 3000 performance, please let him know by calling the above number, FAXing to (503) 327-3276 or writing him at: P.O. Box 151, Albany, Oregon 97321

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Did You Borrow This?

There's nothing like having your own personal copy of Taming The HP 3000 Volume II. You can make notes in it, keep it with you, and refer to it quickly whenever you have a question. To order, just complete and mail or FAX this form. Mail to: Performance Press P.O. Box.151 Albany, Oregon 97321.

Name_____

Company_____

Address_____

City_____ State_____ Zip_____

Country_____ Telephone ( ) _____

In the U.S.: $79.95 per book plus $6.95 shipping and handling.

Outside the U.S.: $79.95 per book plus $11.95 shipping and handling. *Orders must be paid in advance in U.S. funds.* You may pay by credit card, by check drawn on a U.S. bank or by postal money order.

QTY	Title	Price	Total
	Taming The HP 3000 Volume I	$49.95	
	Taming The HP 3000 Volume II	$79.95	
	Shipping & Handling In U.S.	$6.95	
	Outside U.S.	$11.95	
	Total Due		

☐ Check Enclosed

☐ Bill company Purchase Order No.:_____

☐ Credit Card: ☐ American Express ☐ Master Card ☐ Visa

Card No._____ Exp. Date:_____

Signature_____

In a hurry? If you want to order by phone or arrange for fast delivery call **(503) 327-3800 or (503) 926-3800 Fax to: (503) 327-3276 or (503) 926-7723.**

Here's What Folks are Saying about Taming the HP 3000 Volume II...

"What affects system performance? How can I begin to analyze and improve performance on my system? Is there any way to use my current performance to plan for the future? If you've been asking any of these questions, you'll want to check out Robert Lund's Taming the HP 3000 Volume II. Whatever your level of expertise with system performance, you'll find ideas to stimulate you to take charge of your system's performance, rather than letting your performance take charge of you. Lund has a way of using simple examples which make complex issues easy to understand. Taming the HP 3000 is easy to read and educational, a winning combination."

Debra Canfield
Director of Information Systems
Dairylea Cooperative, Inc.

"I highly recommend Mr. Lund's new book (Taming the HP 3000 Volume II) to both novice and experienced users. Anyone with an interest in the how and why of the HP 3000's performance will gain significant insight into the functioning of their system and how it can be improved. Bravo, Bob!"

Lisa Zenev
Systems Manager
Pacific Hospital Association

"Taming the HP 3000 offers the most practical and lucid approach to managing HP 3000 system performance that I have read in my 13+ years experience with CISC and RISC based HP 3000s. Armed with the information in this book and a good performance monitoring tool, the guesswork is removed from HP 3000 performance management. Robert has successfully applied case work from his own experiences to demystify HP 3000 performance analysis. "

David Hodges
Systems Manager
Union Camp Corporation

"Robert Lund has taken a giant leap toward cutting through the dense fog surrounding the subject of HP 3000 Performance. To my knowledge, this book is the best, most concise source of information that exists . This is presented in a logical sequence, and with the assumption that the reader DOES NOT already have a Ph.D. in Computer Modeling. What lies ahead in these pages... will certainly be referred to time and again. But for every HP 3000 system manager, IT MUST BE READ. In these days of cost cutting, 'downsizing', etc., getting the most of what you have has become especially important. This book is a concise volume of information that allows the average system manager, with no advanced degrees or pocket protectors, the ability to get the most of what they have."

Jason Goertz
HP 3000 System Specialist

Shouldn't You Have Your Own Copy?

ISBN 0-945325-02-9

LUND
PERFORMANCE SOLUTIONS

(503) 327-3800 • (503) 327-3276 Fax
34130 PARKWOODS DRIVE N.E. • ALBANY, OR 97321-9547