Section I Introduction

Performance management and capacity planning are disciplines which rely on an effective understanding of a computer's operation vis a vis its consumption of resources.

The goal is to maximize the productivity of a given amount of hardware without impacting true user productivity.

To emphasize the importance of this function, remember that hardware and its support remain a major expense. Consider that beyond the purchase price is the cost of:

Maintenance	Communication hardware
Space	Administration
Power and cooling	Resource allocation
Peripherals	Cabinetry

Although the cost for CPU power is declining rapidly, note that the scope and complexity of applications are increasing equally rapidly, thus resulting in the same level of need, measured in money terms, for computing power as in the past.

Achieving improved performance is a matter of knowing how to:

- analyze the operation of the computer to determine its resource utilization
- recognize abnormal/unnecessary utilization of resources
- determine resource availability
- use the system in ways that minimize resource consumption
- effect appropriate changes in the behavior of users or software
- add to or alter existing hardware or system wide software to maximize total performance

Capacity planning is built on much the same knowledge base. As such, the two functions of capacity planning and performance analysis are performed jointly.

Under VMS, the task of performance improvement is largely a job of dealing with the efficiency of users and application software. There is limited potential for system wide actions which produce meaningful improvements.

There are vast amounts of application software which are inefficient and capable of being greatly improved. Similarly, many VAX users have no regard for the volume of computer resources they consume, or for their own productivity.

Consequently, the key word in performance is analysis. The word means "to examine critically, part by part." Each user, each application, each resource, and each time period must be studied to determine the inefficient applications and users and to identify time periods of unacceptable loading.

Several themes are pervasive in this document and some philosophic points underlie it. The following points are meant as a framework for performance analysis on the VAX.

The VAX and VMS

DEC has provided the world with, in this author's opinion, the finest general purpose computing facility, judged on a cost benefit/basis.

This document will focus on its weak points, suggesting areas to avoid in order to get the most from a VAX. It is important to keep in mind its overriding points of excellence.

> It has an excellent basic architecture - an efficient and powerful instruction set and comprehensive data handling capabilities.

It has an excellent implementation of a primary programming language - Fortran. The features of VMS are almost totally available to the Fortran programmer.

VAX/VMS has a vast range of powerful, easy to use features, including:

Efficient system level I/O
Asynchronous control flow mechanisms
State of the art virtual memory management.

(A major performance problem is that programmers become like "kids in a candy store" - there are so many features they feel they have to try them all, appropriate or, more frequently, not.)

A powerful file management system is included in VMS, and the level of security is excellent for a general purpose system.

The VAX hardware line is the widest range of totally compatible machines available today.

Software Efficiency

A contemporary attitude found all too often is that "designing and implementing software for efficient operation as well as functionality is too expensive, given the 'low cost' of hardware, to be justified."

This attitude is promoted by hardware vendors and software developers, particularly developers of program generators, database packages, and other "hi-tech" tools designed to speed software development. It would be justified if the resulting inefficiency was a matter of 10 or 20%, but this is not the case.

In fact:

Most VAX software is two to 100 times slower than it could be if relatively basic priciples of good performance were incorporated in its design.

The above statement is meant to include much of the commercially available software for VAXes. Many of these products have been developed with little or no regard for performance. The responsibility for this lies in large part with the user community which fails to include performance as a purchase criteria.

Usually a few key programs are responsible for the majority of resource usage on any system. Within any program, it is usually only a small fraction of the total code that is responsible for the bulk of resource usage. A relatively small effort, if directed properly, can pay off handsomely.

Software designs which, from the beginning, include efficiency as a design goal, seldom require much extra development effort so long as the developers are skilled professionals.

The payoff from software efficiency improvements or improved user practices grows over time and as usage increases. On the other hand, as usage grows, solving a performance problem with hardware is akin to trying to fill a Black Hole.

Hardware Productivity vs. People Productivity

Another frequently encountered attitude is that software efficiency is accomplished only at the cost of people productivity, both during development and as the software is used.

In fact:

Significant efficiency results from proper design and good programming discipline, and often does not materially extend the development effort.

Efficient software is usually well designed and well implemented, and, therefore, more maintainable and easier to use. It is usually more error free.

Efficient software produces faster response and turnaround, improving user productivity.

Usage practices which are inefficient as to usage of machine resources are usually also inefficient as to user productivity.

People productivity and machine productivity are not opposites. In fact, they are highly related.

Performance Improvement Magic

There is none. Performance improvement comes from attention to detail and discipline. It results from small changes here and there, each of which, in and of itself, produces only a small overall contribution, but collectively add up to sizable gains.

A large measure of performance improvement and personal productivity improvement results from good usage habits on the part of both developers and users. Computers are expensive resources, and organizations need to insist that the users of these resources utilize them efficiently and productively, just as organizations demand efficient utilization of other, more traditional expensive resources. Acquiring good habits requires self discipline.

High Level Tools

All high level tools (high level languages, utilities, database systems, RMS, file handling mechanisms, etc.) impose some degree of performance cost vis a vis a lower level solution. This cost difference is often one or more orders of magnitude. However, high level tools may lower development cost and may make future revisions easier.

The use of high level tools is justified only if the performance cost differential is less than the saved development costs.

The high level tool may not be justified if:

- usage will be intensive
- future revisions are unlikely or the dimensions they will take can be forecast so flexibility relevant to them can be designed into a low level solution.

High level tools usually have inherent assumptions about problem structure. To use them often means pounding a square peg into a round hole. Specifically designed procedures can often take advantage of "natural" structures inherent in the problem yielding more efficient operation and more natural user interfaces.

High level tools solve a generic problem. Each real problem has a few nuances of its own which may not be addressed by the high level tool, and are either left unsolved or solved with tortured procedures and work arounds. Less than desirable functionality is the result.

High level tools should not be used because they represent the latest technology, because they are the subject of the latest magazine articles, or because "everybody else is".

They should be used only if a thorough cost/benefit analysis demonstrates that the savings they entail outweigh the performance costs and solution limitations they will exhibit as compared to a specifically designed, lower level, solution.

