Introduction

The PATHWORKS network operating system software provides remote file service to desktop computing devices across a local area network. Integration of personal computers (PCs) on a network allows users to share applications, files, and printers. Most applications available on the desktop can be used in a manner that consumes widely varying amounts of that single-point resource known as the file server.

Some of this variation is due to the intentional part-time nature of the server's resource utilization, and some is caused by innocent changes in the user community's work techniques. Since desktop applications are used by novices and experts alike, small changes in the levels of skill, experience, and thus technique can significantly affect the performance of the server.

Capacity planning is a method of estimating the changing hardware needs for a computer system due to changes in workload. It can also be used to explore "what-if" alternatives for existing workloads.

Changes in user work habits such as running macros can increase a server computer's response time by as much as an order of magnitude. In addition, simplistic rules of estimating the consumption of server resources, such as number of users per VUP (VAX–11/780 unit of performance), can be very misleading. The use of applications in ways that increase individual productivity can slow server response time for the user community. These issues should be considered when selecting a file server system. Because the number of active users is often unknown in client-server environments and the user application technique may vary, capacity planning uses a model of the actual workload to predict server performance and help define configuration alternatives.

This paper describes a queuing analytical model that was used to gain knowledge about resource consumption on the PATHWORKS server computer. The paper discusses the special modeling process required for the client-server environment. It describes data capture and workload classification using DECperformance Solution software. Finally, the paper presents the results of a performance analysis of a PATHWORKS server with response-time constraints.

Some of the terms found in this paper have specific definitions. Many of the "correct" terms for network file serving are not the terms used by users of these systems. Network file serving has acquired the name "networked." Server computers are often referred to as "the network," and getting access to one's files on the server is usually called logging into "the network." In this paper, we refer to MS–DOS-based PCs and Macintosh computers generically as desktop computing devices. In addition, the word "workload" refers to the cause of the resource consumption, which is the combination of client application and user technique within that application. The term "workload class" has a specific definition in DECperformance Solution software. It refers to a group of VMS processes that the modeler wants to manipulate differently from other processes.

Defining the Question

PC users on an integrated PATHWORKS network need to determine which server computer system is appropriate to their workloads today, and which will be appropriate as their numbers increase in the future. The system they choose must deliver sufficient performance today and allow a method to plan for expanded needs in the future. Users of desktop computing devices, which are not networked, can benefit from a series of anecdotal model case studies which describe other workloads and the file servers which were recommended. This paper gives the results of our efforts to gain insight into the reasons for and symptoms of server resource exhaustion (bottlenecks) on PATHWORKS file server systems.

Analytical Models

PATHWORKS software takes advantage of the expanded computational power of the client-server architecture, which requires special modeling techniques. Two of Digital's analytical modeling tools can be used in our capacity modeling process, however, DECperformance Solution was the primary tool. The model was used to answer questions about the need to enhance file server computer resource requirements as a result of changes in hardware or workload.

Performance models can answer at least two questions.[1] First, "How is performance affected if we change either the number of users or the amount of hardware?" Second, "How can we maintain performance if we add users doing the same kinds of tasks?" Of the two, the second question is the one we seek to answer when we model PATHWORKS client-server workloads.

Data Collection

Data can be collected with the VAX Performance Advisor (VPA) version 2.1 or the DECperformance

Solution version 1.0 or later. DECperformance Solution software is an integrated product set that provides performance and capacity management capabilities for computing systems. This layered software product runs on the VAX VMS operating system and uses a queuing analytical model to answer questions. This process requires collection of two kinds of information.

1. A detailed record of the cause of resource consumption, including which process is causing each disk or CPU activity. Processes should be combined into like groups, called workload classes, which may be manipulated independently. For example, some workload classes may be reduced or eliminated and some may be increased.

2. As detailed a record as possible of the effect of resource consumption, including the effect on multiple remote clients. Changes in performance are typically measured by the elapsed time from the carriage return to the return of the prompt. In the case of a timeshare user, this is a closed loop since almost the entire process is visible to the data collector.

In a PATHWORKS environment, such data capture is not possible. A data collection device running on the server computer cannot determine the number of users for whom the PATHWORKS server process is consuming resources. Furthermore, the collector cannot detect the response time seen by the users of the desktop devices.

We have developed a general process that can be applied to all client-server workloads. These include applications such as VTX or VAX Notes, in which the number of users initiating the server process' resource consumption are unknown to a data collector.

Figure 1 illustrates a simplified closed queuing model of a PATHWORKS transaction. The user initiates the transaction through a keyboard or pointing device. The application running on the desktop computer performs the initial local processing and issues a call to the server requesting I/O. The server performs some remote computing, and the I/O request is satisfied when the server transmits either the data or acknowledgment that the data has been written. This travels back to the user's desktop device and some further computing leads to a graphic indication to the user to proceed to the next step.

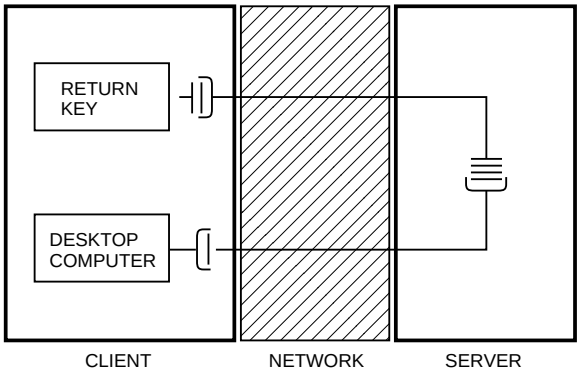

Figure 1 Simple PATHWORKS Queuing Model

If these three sequential queues—client, network, and server computer—were equal in response time, the server would have only a one in three influence on the responsiveness the desktop user sees. Of course in reality the three queues are never equal, and the two local queues are highly dependent on the local desktop computer's capabilities. Each queue can have a request backlog if the service time is not faster than the arrival rate. The response time of any queue is the queue wait time plus the actual time to be serviced. The total response time of the workload class, as modeled on the server, is the analytic sum of all its queues' response times.

In reality, the analytical model of the PATHWORKS environment is more complex than the one shown in Figure 1 and involves disk, memory, and CPU queues. The response time calculated for a PATHWORKS server computer workload class is the calculated sum of the response times of all server process queues for that workload class. As stated earlier, this is only an indicator of a desktop user response time.

Cause and Effect

A data collector, running on the server computer is not aware of the response time perceived by the user at the desktop device, nor can the server's data collector process know how many users are generating the current workload. Server response time is a subset of the response time as seen at the desktop and if the server's response time improves, the user's will improve as well, as shown in Figure 1.

A model that is built from a data collector which has only a partial definition of the whole loop (i.e., the server computer portion as shown in Figure 1) is called an open model.[2] The models described in this paper are open models. Since the most

likely bottleneck is the shared resource known as the server, this is a useful way to model client-server workloads.

Uniform Service Level

Model analysis of a PATHWORKS client-server computer workload cannot predict the increase or decrease in response time seen by the user. A model can determine the effect of any change in hardware configuration or arrival rate (number of users). Capacity planners can use this method to add more users by incrementing arrival rates. Then hardware can be upgraded until an equal or faster server response time is reached. This method can be used to increase the number of users at the same performance or split users into smaller groups with the same or better performance.[1]

Not all desktop transactions require server intervention. In fact, the success of the client-server architecture depends on infrequent access to servers. Obviously, file servers are required when a file is saved. However, many applications perform disk I/O without any obvious or explicit user action. For example, WordPerfect software provides a temporary file that is a type of journal file. Periodically, the application updates this file with data stored in memory. When a user's input reaches a predefined buffer limit, the next keystroke causes the file to be written. The capabilities of this application, and many others, must be considered when planning the capacity of a PATHWORKS file server installation. In this example, the load per client on the server can be significantly reduced by placing the temporary file on a local hard disk.

Performance of a file server computer can also be affected when expert users employ macro techniques or when users generate automated output. Macros read each instruction from the macro file one record at a time, thereby continuously doing I/O. Most expert users provide a save as the last instruction in the macro, which allows them to be absent when the work is being accomplished and then saved. This increases server I/O as well. Most desktop applications permit automated output. For example, some allow form letter generation; some computer-aided design (CAD) applications provide Bills of Materials. This capability also increases server I/O.

The use of either macro techniques or automated output can impact server computer utilization. A server that was intended to be a part-time file server can become a full-time I/O device which can rapidly exceed it's capacity.

To illustrate how a small change in environment can affect file server performance, we employed a Markov model, using a SHARPE queuing model of a server environment. Figures 2 and 3 show the results. We asked the question "If we had 120 users each randomly filing once an hour and each file action took 5 seconds, how often would a user wait for another user to complete a file transaction?" We discovered that only 14 percent of the time another transaction would be running in the server process. Then we asked, "What would happen if 5 of the 120 users started running a macro and this macro did I/O for 5 minutes at random intervals within the hour?" The remaining 115 users continued working as before. In this case the possibility increased to 28 percent that a job request would be on the queue, 24 percent that two job requests were waiting, and 20 percent that three job requests were present.

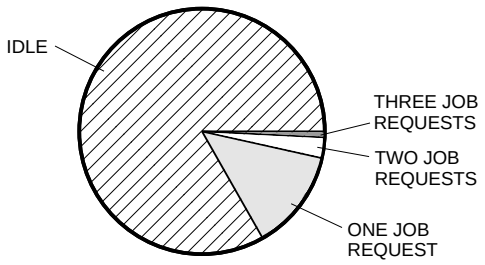

Figure 2 Low Use with Infrequent Saves

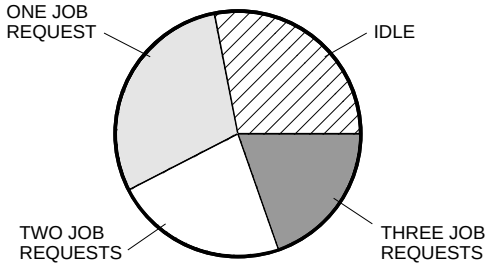

Figure 3 High Use with Few Macros Running

In the same study, less than 5 percent of the users changed the way they were working. None of the applications was changed. Almost any PC or Macintosh application can reasonably be used in this way. As the smaller group of users became more productive, the other 95 percent experienced a significant delay in response time. The system capacity must be sized to allow for a situation in which user activity lessens overall response time.

Modeling Process

The modeling process we describe in this paper was developed over a two-year period. Before discussing the modeling procedures, we list the benefits and limitations of the process.

Benefits

- Determinations can be made as to the numbers of PATHWORKS and new workload class users required to maintain the same performance.
- Single-function server computer models, with only PATHWORKS workload classes, can have non-PATHWORKS workload classes added for a more complex environment.
- The server can be upgraded to maintain the performance level of growing user communities.
- Larger user communities can be divided between two standalone servers to maintain an acceptable level of performance.
- Stable user communities can be reduced to provide equal levels of performance with two smaller servers.
- Hardware trade-offs can be explored. For example, some users can be moved to another disk.
- Local site management can be made aware of the magnitude of daily workload variation; understanding this variation is also part of the model process.

Limitations

- The model cannot predict response time changes, at the client, due to changes in server loading.
- Information about the number of users generating the applied workload must be collected by methods other than using DECperformance Solution software. These methods are detailed in the section Capturing Workloads.
- Although memory can be modeled, increased PATHWORKS read or record management services (RMS) cache requirements cannot be anticipated by the model. When adding users to a PATHWORKS server computer, adequate spare memory must be allowed to provide the same or better cache hit rates. The RMS cache hit rates can be determined, without software tools, by executing a program at the Digital command language (DCL) prompt: @SYS$UPDATE:AUTOGEN SAVPARAMS TESTFILES FEEDBACK, and then reading SYS$SYSTEM:AGEN$PARAMS.REPORT.
- Available modeling tools only allow PATHWORKS workloads to be modeled onto VAX VMS servers.
- Prior to data collection, the server must be checked to see if it is tuned for use today and for the future, or the recommended server system may be incorrectly sized.[1]

Capturing Workloads

DECperformance Solution software requires VAX Performance Advisor version 2.1 or later collector files named nodename_date.CPD. In addition, either a VPA$SCHEDULE.DAT or a PSDC$SCHEDULE.DAT file is required to define the cluster configuration and collection schedule. Either a VAX Performance Advisor version 2.1 or DECperformance Solution version 1.0 Data Collector, or the DECperformance Solution Service Delivery Software kit may be used to collect data. All three require a license and product authorization kit.

Enough data must be collected to represent the range of a typical workload. The sum of the subjective user opinion of performance must be collected as well as the tasks the users were performing. If this data is not collected, the planner may mistakenly model equal levels of user dissatisfaction rather than equal levels of user satisfaction. Subjective performance evaluation is always gathered by interviewing or monitoring users.

Collections should be made over a series of normal workdays to avoid gathering misleading data. We have observed two normal workdays with only a 5 percent difference in the number of desktop users logged into the server, yet five times more server resources were used.

Additional data on user activity that is consuming resources must be collected by methods other than the DECperformance Solution collector. Both the Macintosh and MS–DOS server products have interactive DCL software utilities that provide some information about the condition of the current server process. Command procedures can call these utilities with a brief DCL command string. For example, ADMIN/PC SHOW FILE COUNTERS displays the current cache misses and request rates, and ADMIN/PC SHOW FILE SESSIONS shows the client device ID, client connections, and open files. The size of the server process cache configuration can be gathered using the ADMIN/PC SHOW FILE

CHARACTERISTICS command. If analysis is performed offsite, a DCL procedure can gather information about volumes and system logical names, which allows user disk assignments to be defined. Finally, user authorization resource limits on the server process can be extracted from the system. The Macintosh server software has similar commands using the ADMIN/MSA SHOW CONNECTION command.

When the size of the user community is unknown, the above data must be used to characterize the number of users being modeled. Specific customers with large installations or many remote sites need quantitative user characterization. In all cases the cause of the observed performance characteristics must be determined at some quantitative level.

The data gathered by using the ADMIN/PC SHOW FILE COUNTERS and ADMIN/PC SHOW FILE SESSIONS commands can be invalidated if desktop devices include automated procedures to attach to file services when the desktop device is booted. The simple act of activating the client power switch should not count that user as explicitly intending to use the server computer. On the other hand, explicitly connecting to file services and being interrupted for an unexpected event should not exclude that user from the total active user count. Ultimately, a combination of the total possible and the total active connections is needed.

Defining Workload Classes

With the DECperformance Solution data collector, workload classes are defined prior to starting the modeling process. They are defined either by specifying the anticipated logical divisions or by determining them from the observed performance data. DECperformance Solution software provides many ways to group processes, e.g., user identification code (UIC), resource usage, image name.[3]

The DECwindows interface to the DECperformance Solution performance tool provides an excellent way to review the data.[4] The graphic display of the server process by day along with the subjective user characterization can help select the day or days to be modeled. The same method can be used to determine peak usage hours. Finally, this technique can help categorize workload classes by applicable processes. Table 1 lists the workload class groupings we used.

Table 1
Workload Class Groupings

Workload Name	Image Name Selection Criteria
FILESVS	NETBIOS, PCFS_*, PCSA$*
OVERHEAD	AUDIT_SERVER, NETACP, EVL, ERRFMT, OPCOM, JOBCTL, REMACP, CONFIGURE, IPCACP, TPSERVER, FILESERV, CSP, SMISERVER
ABNORMAL	PSDC*, VPA$DC_V5, DECC*, SPM, MONITOR
MAC_FILESVS	ATK*, MSAP*, MSAD*, MSAF*
LAD	LAD$KERNEL
OTHER	(All Else)

Workload Family	Workload Member(s)
PW_DOS	FILESVS, OVERHEAD, ABNORMAL
PW_MAC	MAC_FILESVS, OVERHEAD, ABNORMAL
PW_BOTH	FILESVS, MAC_FILESVS, OVERHEAD, ABNORMAL
PW_LAD	LAD, FILESVS, OVERHEAD, ABNORMAL
PW_THREE	LAD, FILESVS, MAC_FILESVS, OVERHEAD, ABNORMAL

Workload families are groups of workload classes that the data collector can expect to see. The PW_DOS workload family characterizes a system as a PATHWORKS file service environment. It includes PATHWORKS server processes, required system overhead functions, and processes needed to collect data that are not normally part of the system. All other processes are automatically placed in a category called "other." This suits the needs of our general-case, single-function PATHWORKS server computer, but any server can be used for tasks unrelated to the PATHWORKS print and file service. If the tasks in the default (other) category need to be subdivided for separate scaling, the workload class definitions have to be added to a family which calls each workload class explicitly, as indicated for the PW_LAD workload class family in Table 1.

For example, to answer the question "As groups of ALL–IN–1 system users change to PCs, how many users can the PATHWORKS server computer support?" This determination requires defining another workload class by UIC for the ALL–IN–1 system users. The workload class could be moved by UIC to the FILESVS workload class. This method assumes the current collection of FILESVS workload classes reflects the mix of the remaining ALL–IN–1 system users.

Prior to the model building step, the PSDC$DATABASE logical must be pointing to the location of the VPA$SCHEDULE.DAT and the VPA$PARAMS.DAT files. The model building step generates a model with the workload class groupings given in Table 1. The workload class and family definitions are made using the DCL command ADVISE PLAN EDIT in the VPA/VME (VAX Performance Advisor /VAXcluster Modeling Environment) utility and are written to a file named VPA$PARAMS.DAT. (If the DECperformance Solution tool is used, the files are named PSDC$SCHEDULE.DAT and PSDC$PARAMS.DAT.)

If this logical is defined while using the DECperformance Solution DECwindows interface invoked from the session manager, the logical may not take effect in the DCL session in which the model is to be built. The command to generate a model can include the time selected to be representative and the workload class family definition name. A report can be generated which describes the newly built model. The command used is:

```
ADVISE PLAN BUILD/CLASS=
(USER=PW_DOS)/BEGIN=9-DEC-1991:10:30 -/
END=9-DEC-1991:11:30/REPORT/
OUTPUT=MYMODEL.RPT MYMODEL.MDL.[3]
```

At this point the model must be validated by typing ADVISE PLAN REPORT MYMODEL.MDL VALIDATION/OUTPUT=MYMODEL_VALID.RPT at the DCL prompt. All predicted values should be within 10 percent of the calculated values.[2,3] A CPU validation report for a collected workload includes data on throughput, queue length, average service time, average response time, and percent of

utilization. For the FILESVS workload, the measured utilization was 67.7 percent as compared to 64.7 percent for the model. This 3 percent difference is 4.4 percent of the measured value and thus well within the 10 percent range.

Normalizing the Environment

The next step is to return the system to the normal environment. Even though data collectors are typically designed to utilize a small amount of system resources, they are not normally part of the server workload. Grouping abnormal processes into a workload makes it easier to remove them during the DECperformance Solution model process. Access to the DECperformance Solution model interface is achieved through the command ADVISE PLAN MODEL MYMODEL.MDL.[3]

Recording Response Times

The next step is to solve the model and view the calculated response times for the remaining workload classes. These are FILESVS, OVERHEAD, OTHER, and any custom-defined classes. The OTHER workload class can be used as a defined workload class provided it contains no unexpected processes that are using significant resources. The calculated response times for the remaining workload classes should be considered maximum times, and model manipulations should always seek to attain these numbers or less.

If the intention is to capture the PATHWORKS workload class for use elsewhere and if the same system had significant OTHER workload classes, these classes should be removed (turning the server computer into a single-function PATHWORKS server).[3] This reduces the response times of the remaining workload classes and requires increasing the PATHWORKS workload class until the response time returns to the observed value. The increase in throughput is proportional to the increase in PATHWORKS users accommodated at the same performance, without the competition of the OTHER workload class.

Model Manipulation

Basically, the response time can be manipulated (1) by decreasing the usage of a significant resource (model resource utilization percentages help locate the bottlenecks) or (2) by increasing the capacity of that resource.

There are two ways of decreasing the resource utilization. If the resource is single-threaded on the critical path, as a CPU would be in a nonsymmetrical multiprocessor (SMP) machine, the method is to reduce the number of users by decrementing their arrival rate (called throughput or transactions per second [TPS] in various menus) or by increasing the speed of the bottlenecked device.

The model allows for workload class manipulation to remove arrival rates of the workload class. As this is being done,the original arrival rate must be noted so the same changes can be applied to the number of users that caused the workload.

If the bottleneck is not on a single path, its capacity can be increased by spreading the load across another similar device. This can be achieved with multiple disks.

In the ALL–IN–1 system case discussed earlier, 100 percent of the workload class from the first UIC group of ALL–IN–1 system users can be removed from the model.[3] If the model is solved at this point, all the workload class's response times should diminish. If the FILESVS workload class throughput is incremented in proportion to the additional PATHWORKS users and the model is solved again, the response times of all workload classes increase.

The question is: "Has the removal of the ALL–IN–1 system users decreased critical resource usage sufficiently that their addition to the PATHWORKS FILESVS workload class does not increase any of the remaining workload class's response times beyond their target?" The answer depends on the per capita usage of the critical resource of each workload class. The nature of each workload class may be different. For example, PATHWORKS workloads do not scale well over SMP processors. The workload class being removed may use more CPU time per user than the PATHWORKS FILESVS workload class.

Findings

We analyzed a large PATHWORKS workload class from a VAX 6000 model 510 system whose CPU utilization averaged 72 percent. The subjective user evaluation was that this system was very near performance capacity limits, and a fair amount of dissatisfaction was associated with the level of performance. The question was asked "Could this community be split in half across two VAX 4000 model 300 systems with the same or better performance?" We immediately agreed this would work, but went about proving it with a model. After the workload class was normalized and the response times were noted, the workload class arrival rate was reduced by 50 percent and the CPU and disk systems were changed to the VAX 4000 model 300. The new model was solved, and the response times were significantly worse than with the VAX 6000 model 510 system. The workload class was halved again, and

the resulting response time was still slightly over the target.

This finding was difficult to understand since the VAX 4000 model 300 system CPU was now down to 36 percent utilized, and only one quarter of the users remained. The reason for the inadequate response time was found by studying the queuing model. Figure 4 is a simplified model showing two CPUs and their queues displayed on a time scale. The first is a slower CPU and the second a faster one. Since we did not allow the response time (total queue plus service time) to vary, the queue length (measured in number of waiting jobs) on the slower CPU was shorter. The service time of the slower CPU was larger, in proportion to its queue wait time, and therefore an interruption by an overhead process caused significant loss of processing time (response time) to be available for the critical workload class.[5]

Therefore, the general rule became: Slower CPUs will be less utilized at the same workload class response time. This result has been seen on two different customers' workload classes (one with DOS and one with Macintosh clients) which were modeled by different engineers using different modeling tools.

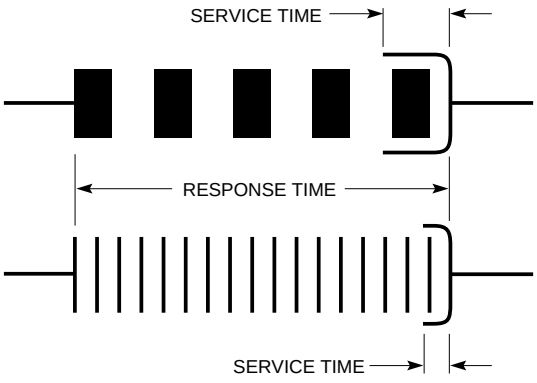

Figure 4 Server Queue Comparison on Different CPUs

Another surprising result became evident in the day-to-day variation at a customer's installation. The same two workload classes were analyzed across several days to examine typical workday variations in workload class resource utilization. Two normal workdays were selected by the customer. The most intense hours of these two days were different by a significant factor. On one workday, three to five times as many users applied the same workload class as on the other day, yet all experienced the same response times. This wide variation is typical of client-server workloads.

Library of Workload Classes

After we had captured a series of data, we created a small library of real workloads that represented various conditions. The actual workloads consist of a model file that is devoid of user-specific information. Other non-PATHWORKS workloads can be added to these models. Alternatively, the numeric workload characterization can be added to existing models. Using the above methodology, the model can be manipulated to determine what system is appropriate for this more complex environment. As additional installations are analyzed, their model files will be added to the library.

With either the DECperformance or DEC Capacity Planner modeling tool, the process is the same: Change the hardware and modify the throughput to maintain or lower the response times of the model during iterations. The changes to throughput are then applied to the original number of users to determine the acceptable number of users in terms of server computer capacity.

Although both modeling tools exhibit similar mapping of the quantitative workload class characterization, we do not know the units of some of the key metrics used. Therefore, entering a workload class captured in one model to another model is not recommended.

Summary

The PATHWORKS network operating system software provides remote file service to desktop computing devices across a local area network. Capacity planning of client-server environments requires special modeling techniques. DECperformance Solution software provides performance and capacity management capabilities for computing systems; it uses a queuing analytical model to answer resource consumption questions. The modeling process depends on the collection of enough data to represent the range of a typical workload. Additional data on user activity that consumes server resources must also be collected. Analysis of workload models reveals the reasons for and symptoms of bottlenecks. Capacity planning depends on the results of these analyses to predict server response times.

Acknowledgements

I would like to thank Prashant Bhabhalia, who helped me ensure that the modeling process is correct, and Dick Dunnington, who checked my queuing theory. Also, I would like to thank Frank Caccavale, who helped me understand the PATHWORKS server architecture. Melur Raghuraman helped me grope toward the Uniform Service Level model described here. Karl Friedrich, Ann Bousquet, and Lindsey Stephens helped me transition to DECperformance Solution software. Finally, I would like to thank Pete Stoddard for applying his technical reviewer skills to this paper.

References

1. *Guide to DECcp Methodology* (Maynard: Digital Equipment Corporation, Order No. AA-NA34A-TE, 1989).

2. R. Jain, *The Art of Computer Systems Performance Analysis* (New York: John Wiley & Sons, 1991).

3. *DECperformance Solution Capacity Planner User's Guide* (Maynard: Digital Equipment Corporation, Order No. AA-PH6LA-TK, August 1991).

4. *DECperformance Solution Performance Advisor User's Guide* (Maynard: Digital Equipment Corporation, Order No. AA-PH6SA-TK, August 1991).

5. F. Hiller and G. Lieberman, *Operations Research* (San Francisco: Holden Day, 1967).

Trademarks

Author Biography

Christopher E. Methot Chris Methot has been analyzing client-server performance since joining Digital in 1986. He has worked in performance characterization of LAVc systems and has contributed to VAX Performance Summaries. He is currently the supervisor of Capacity/Performance Engineering in the Personal Computing Systems Group. In addition to developing the PATHWORKS client-server modeling process, his group is developing a standard performance test for Macintosh servers and has benchmarked many of Digital's hardware servers. Chris holds a B.S. (1967) in industrial design from the University of Cincinnati.