Proceedings of the CMG'89 Conference

Experiences of Hierarchical Workload Modeling in Capacity Planning

Kimmo E. E. Raatikainen†
University of Helsinki, Department of Computer Science
Teollisuuskatu 23, SF-00510 Helsinki, FINLAND

Abstract

In an extensive research project we have developed and analyzed modeling tools and techniques to be used in practical situations of capacity planning. Our objective is to construct queueing network models that can be used to analyze the performance of a VAXCluster-system both with measured and with predicted workloads.
The reported experiences indicate that the hierarchical modeling of workload provides a fruitful basis for integrating the capacity planning into organization-wide general planning. However, the standard measurement tools of VAX/VMS are insufficient to identify all parameters of the queueing network workload. Considerable specialized knowledge of the examined system, carefully planned and executed benchmarks, and subjective adjustments are required in identifying the io-workload.

1. Introduction

In principle, a capacity planning study consists of three primary steps. Current and future business activities identify the needs and opportunities for employing computerized information processing. These needs and opportunities are the external premises of capacity planning. The first step of a capacity planning study is to translate these premises into computing requirements and performance objectives. When the requirements and objectives are specified, the second step is to predict and analyze the performance of possible computer system configurations. Finally, the third step utilizes performance predictions and cost-benefit analysis. They are the basis of constructing the decision alternatives for the development of the information processing. In this paper we consider the computing requirements or more precisely how activities of an organization can be translated into workload of a queueing network model.

We regard capacity planning as an integral part of general organization-wide planning. This implies that business activities of the organization must be translated into the consumption of various hardware resources. Since business activities have no direct correspondence to the consumption of hardware resources, we use a hierarchical model of the workload. Business activities are first translated into a *functional workload*. The characterization of workload as service demands of hardware resources is necessary in performance predictions that are based on queueing network analysis.

The characterization of workload involves three phases. Usually the workload is so heterogeneous that the queueing network model must have multiple classes of customers. This classification is the first phase. The other two phases consist of identifying the *functional* and *queueing network workload*.

The second practical factor that has affected our modeling and identification is that the *monitoring* of the current workload and performance *should not create additional load*. Therefore we have decided to use primarily the data that is automatically collected. In the examined system measurement data is provided by the VAX/VMS Accounting and Monitor Utilities.

In this paper we begin by discussing modeling issues. We review briefly some specific features of the examined system and queueing network modeling. In Section 3, we summarize the measurement data obtained by VAX/VMS Accounting and Monitor Utilies. In Section 4 we concentrate on workload modeling. In this paper we consider all three phases of workload characterization: functional classification, identification of functional workload, and identification of queueing network workload.

2. Modeling Issues

In modeling the basic concern is the system under consideration. The system's specific features; the hardware, workload, and measurement tools; determine to a great extent possibile methods used for modeling. In this section we summarize the specific features of the examined system and discuss queueing network modeling.

Our approach to modeling is based on two fundamental principles. The first one is decomposition of the performance modeling effort. The second one is hierarchical

†This work was supported by Emil Aaltonen Foundation

modeling of workload, which will be considered in a later section.

We decompose the performance modeling into three main parts. The effects of cpu/io-capacity, main memory, and permanent system load are analyzed using separate models. This decomposition allows us to use simple models. The drawback, however, is that the interactions between the parts must be considered separately. In this paper we consider only the effects of cpu/io-capacity. A closed multiclass queueing network model is used to study these effects.

2.1 Examined System

The examined system is a VAXCluster of one VAX 8650 computer and one HSC50 storage controller. At the time of our measurements the operating system was VAX/VMS, version 4.5. The system is primarily used for academic research. The main functions are: 1) a national scientific computer for rarely used and expensive software products; 2) a front-end to systems having some databases of the government; and 3) a gateway to national and international networks.

The most important site-specific features of the workload are: 1) there are no dominating applications; 2) the workload is very heterogeneous; and 3) a significant part of workload consists of programs written by end-users.

2.2 Queueing Network Modeling

Queueing network modeling has turned out to be a powerful tool in performance analysis. A queueing network model has natural ways of representing how some performance indices depend on computing capacity and workload. A queueing network model can be interpreted as a combination of two descriptions: a description of computing capacity and a description of workload. The description of computing capacity or the *hardware model* consists of the service centers and their properties. The properties include queueing discipline and service rate. The description of workload or the *workload model* consists of workload classes, workload volume for each class, and service demand for each workload class on each service center.

A hardware model describing how the system performs under the given workload should include all the hardware resources having influence on the performance indices of interest. In principle, there are three basic hardware components, which affect the performance: CPU, IO-SUBSYSTEM and MAIN MEMORY. In our queueing network model we consider only the effects of CPU- and IO-CAPACITY. The effects of main memory are considered separately. The omission of main memory simplifies the analysis but also reduces the range of the model quality.

In performance modeling the interactive usage of a computer system is usually represented as a closed queueing network model; see for example [2, Ch. 1]. Our model (Figure 1) is a closed queueing network model of four service stations. The *thinking delays* between program executions are represented as the **Class Switch and Delay** node. The node is an *infinite capacity server* or an IS node whose service rate is same as the number of customers in the center.

In the **CPU** node we assume that the scheduling policy is *processor sharing*, i.e. that the node is a fixed-rate PS server. In VAX/VMS the actual processor scheduling is based on dynamic priorities. Since most of the interactive users have the same priority parameters, we assume that the processor sharing discipline is a reasonable choice.

The *io-subsystem* is modeled as two io-servers: **DIO** and **BIO** nodes. The division is based on the available measurement data. The VAX/VMS Accounting data contains separate *counts* for the direct and buffered io-operations.

The **BIO** node serves the buffered io-requests. It is also used to solve the problem of the *thinking delays* during the program execution. In the "standard" model the tasks leave the *Computer System* and enter the *Terminals* node, when they start to wait for terminal input. This is an ideal situation but we can not use such a model, since we do not have any information of the waiting times due to terminal input. In view of the available data, two delays of very different nature must be represented in the **BIO** node: the time elapsed in terminal output and the time elapsed in terminal input. There-

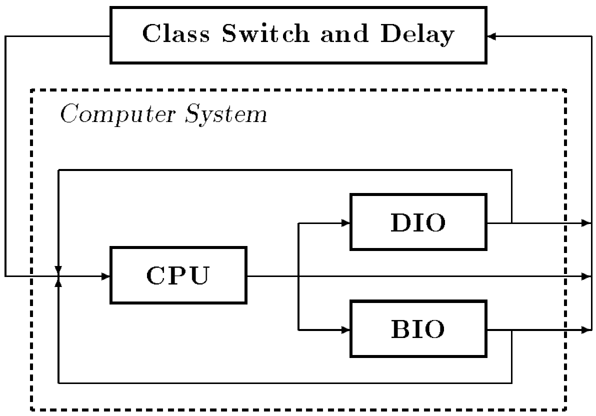

Figure 1: Queueing Network Model for Examined System

fore the best choice is to model the **BIO** node as an IS server.

The **DIO** node serves the direct io-requests. We assume that all the direct io-operations are disk io-operations. In our system, interactive tasks seldom use tape io-operations. In addition, the tape operations include delays for manual operations that are independent of the available computing capacity. Therefore we have excluded the tasks involving tape operations from the workload.

In VAXCluster systems the disk io-operations are highly optimized. The HSC50 storage controller, actually a PDP-11 computer, performs all disk management functions. The controller may store up to 120 transfer requests, and provides buffering for 128 Kbytes of data. In addition, the VAX/VMS File System has its caches in the main memory. A cached and optimized disk system behaves under low or moderate load as an IS server. Since the io load is far from the saturation point, we model the **DIO** node as an IS server.

3. Available Measurement Data

The capacity planner must have adequate knowledge of the workload and performance of the current system. Such knowledge can be obtained only through systematic monitoring. Systematic and continuous measurements of the workload and performance provide the necessary background to capacity planning. We assume that the available measurements are the daily data obtained using VAX/VMS Accounting and Monitor Utilities.

There are two major problems with Accounting and Monitor data. The one is its relevance to workload identification. The other is its enormous volume. Although daily accounting and monitor files are extensive, they are by no means complete for modeling. All important indicators of the performance and workload cannot be calculated exactly but must be estimated.

Even in medium-size installations thousands of tasks are executed daily during prime usage times. In other words a daily accounting file contains thousands of records. The VAX/VMS Monitor Utility collects aggregated performance data using a fixed, predefined collection interval. In order to see dynamic changes the interval should be relatively small, for example a couple of minutes. As a result a daily monitor file will also be large.

In this project we have divided the characteristics to be analyzed into two main groups: characteristics used in *model specification* and characteristics used in *workload identification*. The model specification or the decisions of how the system is modeled are based on the characteristics of *system activity*. Most indicators of system activity are dynamic, that is they are analyzed as functions of wall-clock time. The indicators of *resource usage* are used in workload identification. These indicators are static in the sense that they are considered as random variables, each being identically distributed during the examined period.

3.1 System Activity

The trace analysis of daily accounting and monitor data is our primary tool for examining the characteristics of system activity. The ultimate objective is to achieve an understanding of what is happening in the examined system. The second objective is a rough analysis of trends to be used both in *workload forecasting* and in studying the quality range of the model. Some indicators of system activity are also used in identifying the workload. We use two groups of characteristics to describe the system activity: *Sessions & Tasks* and *CPU & IO*.

3.1.1 Sessions & Tasks

This group consists of three characteristics: *Number of Active Sessions*, *Number of Active Tasks*, and *Task Departure Rate*. These characteristics are used in determining parameters of functional workload.

We need two views for the numbers of active sessions and tasks. Firstly, we need a dynamic view that describes how these numbers change in time. Secondly, we need a static view; that is the distribution of load during the examined time. Both the dynamic and static behaviour can be accurately constructed from the accounting data.

The task departure-rate or throughput is an unobservable quantity. It must be estimated using the point process of task departures. As usual, we estimate the rate by the number of departures during a time interval of constant length.

3.1.2 CPU & IO

The accounting files contain only aggregated information about cpu- and io-activity. The monitor data also contains information about these characteristics. However, the monitor data is less relevant for our purposes, since we have no way to extract the activity due to interactive tasks from the activity due to other tasks.

We summarize the cpu- and io-activity with four characteristics: *cpu-utilization*, *rate of buffered io-operations*, *rate of direct io-operations*, and *rate of disk io-requests*. These characteristics are used in model specification and in qualifying the specification. The cpu-utilization is characterized using only the accounting data, and the disk request-rate using only the monitor data. The buffered and direct io-rates are analyzed using both types of data.

When the characteristics are analyzed using the accounting data, we assume that the resource usage is distributed uniformly over the residence time. However, this assumption is unrealistic, if we are interested in the cpu- and io-bursts. Nevertheless, for our purposes the assumption is reasonable, since we obtain a smoothed view of the "average" dynamics.

3.2 Resource Usage

The VAX/VMS Accounting Utility provides the primary data for workload identification. We use the *cpu-time* and *counts* of direct and buffered io-operations in identifying the current workload.

1) CPU Time. The accounting data contains the cpu time in units of 10 milliseconds. In our queueing network model the measured cpu-time is the service demand of the **CPU**-node (the sum of service times over all visits).

2) Dio Count. The dio count is the number of direct io-operations. In the queueing network model the count corresponds to the number of visits in the **DIO** node.

3) Bio Count. The bio count is the number of buffered io-operations. In the queueing network model the count corresponds to the number of visits in the **BIO** node.

4. Modeling the Workload

Before a queueing network model can be solved, the workload must be described as workload classes, workload volume for each class, and service demand (in units of time) for each workload class on each service center. We call this description the *queueing network workload*. In capacity planning, the business plans of an organization provide the primary description of workload. We call this description the *business workload*. It is a task of capacity planning to translate the business workload into queueing network workload.

In general, these two descriptions of workload are quite different. The difference is usually so great that the queueing network workload cannot be directly derived from the business workload. Therefore it often becomes necessary to adopt an intermediate description of workload based on some user-oriented measures.

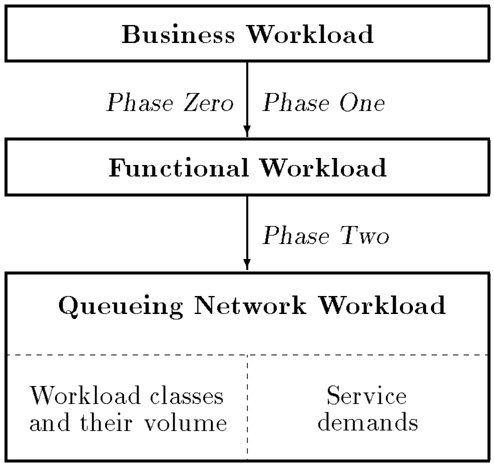

Figure 2: Descriptions of Workload

Typical user-oriented measures for interactive workloads are related to the number of users and the types of programs executed. We call our intermediate description the *functional workload*. The functional workload characterizes the workload as the types of programs to be executed and their volume. Now the workload model can be presented as three descriptions of workload and two phases of translation as shown in Figure 2. In addition to those two phases, an initial phase (*Phase Zero*) specifying the types of programs or *functional classes* is an essential part of workload modeling.

4.1 Functional Classification

The objective of functional classification is to create a bridge between user activities and functional and queueing network workload. We have four fundamental factors that affect the functional classification:

1. The available measurement tools must know the items, of which the functional classes consists. We assume that the functional classes consist of tasks identified by the name of an executed program.
2. The end users must understand the relationship between their work and the functional classes. The number of executed programs belonging to each class must be predictable from business plans and available history of computer usage.
3. The capacity planner must be able to predict the changes in service demands of the functional classes. The three basic problems of forecasting are:
   - Different programs may have different rates of service demand growth.

   - The proportions of program executions in a functional class may change.
   - A new program may replace an old one.

   These problems can be solved only if we maintain the usage statistics for each (common) program separately.

4. The number of classes should be reasonable. If there are too many classes, the reliability of the forecasts and the accuracy of identifications will decrease. Typically, the number of functional classes is moderate or large, say 10–50.

We base the first-level classification on the accessability of programs as follows:

1. **Common Images** are installed by the staff of the computer center as *known images*. The computer management staff knows (or should know) the functions of the programs. This knowledge is used in dividing these images into functional classes.
2. **Private Images** are maintained by the end-users. The management staff does not necessarily know the functions of these programs. As a consequence their classification is extremely difficult and can only be based on extensive user surveys. In a research environment many user-written programs are of an ad-hoc nature and used only a few times. Therefore we take the private images or user programs as one functional class.

The classification of the common images is based on program names and their purposes. We divided the common images into eleven classes:

1. **Basic Services**: small system services that are widely used. Typically they require very little computation. The class contains the VAX/VMS-commands login, logout, directory, delete, rename, set, show, submit, and create; computerized calendar; and local extensions to login and show.
2. **Compilers**: Fortran, Pascal, Macro, C, and Cobol. The Linker is also included in this class.
3. **Editors**: EDT, TPU, and Emacs are used.
4. **File** utilities: VAX/VMS-commands copy, difference, librarian, search, sort/merge, type, and analyze/rms; Jnet commands receive and sendfile; and file archival commands archive and retrieve.
5. **Communication** utilities: VAX/VMS-commands mail, phone, reply, and request; Jnet command send; computer conference utility PortaCom; Kermit; and various PAD-, SNA-, and X.400-services.
6. **Help services**: VAX/VMS help, help systems of application programs, and a local information system.
7. **Maintenance**: programs that help the users in maintaining their own computing environment and in troubleshooting. In addition, programs that are used by the system management staff to maintain global system services.
8. Application programs for **Scientific Computation**.
9. Application programs for **Graphics**.
10. Application programs for advanced **Text Processing**.
11. **Other Application** programs: APL, Lisp, Prolog, Macsyma, Matlab, database products, statistical packages, e.g. SAS.

4.2 Identifying Functional Workload

The business workload describes (or should describe) the activities of different organizational units that require computing services. By analyzing these activities together with each organizational unit or *user group* the capacity planner can identify the major functional classes of computer services or *generic tasks* and their volume.

In addition to the major generic tasks that are directly related to business activities, these activities create utility tasks: such as set, submit, directory, copy, delete, or rename commands in VAX/VMS-environment. The end-users do not usually recognize these utility tasks as computing services. Moreover, volumes of utility tasks can not be predicted from business plans. The best we can do is to predict these volumes subjectively using available usage statistics. The prediction is not as difficult as it seems, if weekly or monthly usage statstics are constructed for each user group. An example of user group statistics is given in Table 1.

The description of functional workload includes functional classes and the volume of each functional class. For example, the volume can be expressed as the number of executions in an hour.

4.3 Identifying Queueing Network Workload

The description of the queueing network workload includes several parameters that can be divided into two sets: *customer description* and *service demands*.

4.3.1 Customer Description

The customer description consists of customer chains, workload classes, the volume of workload classes in each chain, and thinking delays in the center representing terminals for each chain. With the exception of thinking delays these parameters are easily identified using the functional workload and objectives of performance analysis.

Table 1: Executed Generic Tasks by Some User Groups During One Week

User Group	Basic Services	Compilers	Editors	File utilities	Communication	Help services	Maintenance	Sci. Comput.	Graphics	Text Proc.	Other Applic.	Private Images	All
C	4910	271	650	942	440	47	31	33	14	6	71	344	7759
G	116		3	24	4	6	1						380
L	2631	77	149	439	256	70	20	33	15	9	12	138	3849
N	814	108	61	207	8	3	3				1	64	1269
S	288	2	2	31	55	12	11					2	403
T	573	55	75	152	34	5	16	4			4	30	948
U	3414	151	520	719	570	41	85	107	63	142	103	56	5971
V	3790	119	508	550	501	45	453		22	72	3	43	6106

1) Customer Chains. In our approach the determination of chains is based on profiles of user groups. The user profiles are specified using the proportions of executed generic tasks. Since the user profiles are quite similar, we decided to use only one chain.

2) Workload Classes. The functional classification provides the functional classes. In analysis we are usually not interested in all the functional classes. Therefore, in analysis we can reduce the number of classes. The workload classes are the functional classes of interest and one class consisting of all other functional classes.

We are interested in functional classes of *Basic Services* and *Compilers*. Hence we have three workload classes: *Basic Services*, *Compilers*, and *Other Tasks*. The class of *Basic Services* primarily represents small tasks that are used extensively. Users assume that tasks of *Basic Services* will execute quickly. The class of *Compilers* represents tasks having greater service demands.

3) Volume of Workload Classes. The volume of workload classes is described using two parameters: the number of customers in each chain and the proportions of executions of each workload class in each customer chain.

3a) Number of Customers. The summaries of active sessions and tasks can be used to select suitable values of N. The number of customers corresponds to the number of active sessions.

In the examined system the number of active sessions was in the range of [28,53] during the daily periods of the peak load. Reasonable choices of N are close to the observed maximum, since we must balance the modeling error of using a closed model even though the actual system is open. We found that $N = 55$ produces a load distribution closest to the measured proportions.

3b) Proportions of Executions. Proportions of executions are obtained directly either from measurements or from predicted functional workload. These proportions (π_c's) are needed in specifying the service demands of customer chains:

$$D_{device} = \sum_{class} \pi_{class} D_{class,device} \; .$$

We have measurted the following proportions:

Class	π_c
Basic Services	.6170
Compilers	.0329
Other Tasks	.3501

4) Thinking Delay. The actual times between successive tasks of the same user can be analyzed from the VAX/VMS accounting data. However, these delays are difficult to predict. Therefore, we do not calculate the actual thinking delays, although they have a clear real-life correspondence. We use the thinking delay only to adjust the throughput of the model.

In product-form models only the mean delay affects the throughput. The determination of the mean delay Z

(over all the classes) is based on the operational analysis of the queueing network models; see [1]:

$$\left.\begin{array}{rcl} N & = & (R+Z)X \\ \bar{N} & = & RX \end{array}\right\} \Longrightarrow Z = \frac{N-\bar{N}}{X},$$

where N is the number of customers, Z the mean thinking delay, $\bar{N}$ the mean active load, and X the throughput of the system. The measured throughput was 0.3621 tasks per second, and the measured mean active load 19.30. With $N = 55$ we obtain $Z = 98.59$.

In the case of future workloads we cannot use direct adjustment. We propose that the future thinking delays are predicted using a time-series of adjusted thinking delays.

4.3.2 Service Demands

The measurements or predictions provide the mean service demands directly. However, the units in which the means are obtained are those of VAX/VMS accounting data. The io-demands that are avaialble as operation counts must be converted into delay times.

1) CPU. The determination of the CPU demands is straightforward. The accounting data provides the CPU demands for each task. The means of measured total cpu-times were:

Class	D_{cpu}
Basic Services	.1951
Compilers	4.113
Other Tasks	2.275
All Tasks	1.023

2) DIO. The VAX/VMS Accounting Utility provides only the count of *direct io-operations*. In the queueing network model the count corresponds to the number of visits in the **DIO** node. We use benchmark results to convert the counts into delay times.

The determination of the mean request-delay is not simple. The delay depends on the type request in question. We must determine the means separately for different kinds of requests. We have found that in the examined system it is sufficient to consider three types of requests: the *sequential write*, *sequential read*, and *random access*.

In addition to the means we must identify the proportions of the three request types (β_{sw}, β_{sr}, and β_{ra}) for each workload class. Using the knowledge of the io-behaviour of the system images and the contributions of each image to the direct io-count of the class, the β's can be determined. The determination of the β's is not accurate, but it is the best we can find.

We have determined the following β's:

Class	β_{sw}	β_{sr}	β_{ra}
Basic Services	.10	.40	.50
Compilers	.40	.60	—
Other Tasks	.30	.40	.30

In the determination of the β's and in the benchmarking we should have more accurate measurement tools available than currently available in the VAX/VMS. We should have the request log from the HSC disk-server. The log should contain the initiator, operation code, and arrival and departure times of the request. Such a trace could not be collected continuously. For example, traces measured once a month and analyzed off-line would remove much of the current uncertainity in modeling the disk subsystem.

Benchmarking an optimizing and caching file system like a VAXCluster is not a simple task. We have three important factors to be considered:

1. The measured delays should be only due to io-request.
2. The request arrival-rates should be on the planned level. We must adjust the rate levels for the whole disk subsystem, each HSC disk-channel and each individual disk unit.
3. The hit ratios to the caches of the file system should be on the planned level.

The VAX/VMS Monitor data was used to identify the current levels. In addition, the Monitor data is the basis of predicting the future levels.

We did not obtain an ideal situation for the benchmarks, since special instrumentation tools were not used. The test programs performed the io and measured the response times using VAX/VMS system services. In addition to the test program, we simultaneously executed two other programs. Their objective was to balance the request rates. Among the other things, the actual rates and hit ratios were measured by the monitor during the benchmarks.

The best solution we found was to run the test programs as ordinary user tasks when the cpu-load is moderate. If we run the programs under heavy load, the measured delays will include cpu-contention. During the light load we found that the cache hit-ratios tend to be too high, and we concluded that the test programs had an almost perfect fit.

The measured mean request-delays were:

Sequential Write	.18016
Sequential Read	.04406
Random Access	.04055

When the proportions and mean delays of dio-requests and the mean number of requests, V_{dio}, are identified, the mean total request delays, D_{dio}, can be easily calculated:

Class	V_{dio}	D_{dio}
Basic Services	12.15	.6794
Compilers	113.9	11.21
Other Tasks	73.60	6.170
All Tasks	36.20	2.948

3) BIO. In our system most of the buffered io-operations are due to either terminal input or terminal output. The terminal output operations do not cause serious problems. However, input operations are problematic, since they include thinking delays of interactive users.

We have used the mean buffered io-delay as an adjustable parameter. In the case of current workload we can calculate (using Little's Law; [1]) the mean system response-times:

$$R_c = \bar{N}_c / X_c \ . \tag{1}$$

The next step in determining means of buffered io-delays (for each class) is to determine the grand mean (over all the classes) of delays in BIO node and of residence times in CPU node. We use iteratively the *Mean Value Algorithm* [2, p. 115] to determine the grand mean of BIO delays, D_{bio}, so that the MVA gives the same mean response time as Equation (1). In addition, the MVA provides the mean residence time in CPU node, R_{cpu}. Since each class is (on average) delayed in CPU by the same factor, R_{cpu}/D_{cpu}, we obtain the means of residence time for each class:

$$R_{c,cpu} = \frac{R_{cpu}}{D_{cpu}} D_{c,cpu} \ .$$

Therefore the means of BIO-delays for each class can be calculated as

$$D_{c,bio} = R_c - R_{c,cpu} - D_{c,dio} \ .$$

The measured and calculated parameters that adjusted BIO-delays are based on are summarized in Table 2.

In the case of predicted workload we must adopt a different method to determine the means of BIO-delays. Using the measured counts of buffered io-operations, $V_{c,bio}$,

Table 2: Summary of Measured and Adjusted Parameters of the Queueing Network Workload

	Basic Serv.	Com-pilers	Other Tasks	All Tasks
$\bar{N}$	1.3315	0.4076	17.5609	19.30
X	0.2234	0.0119	0.1268	0.3621
R	5.9602	39.546	138.49	53.300
R_{cpu}	0.3048	6.425	3.55	1.598
D_{dio}	0.6794	11.214	6.17	2.948
D_{bio}	4.9760	21.907	128.77	48.754
V_{bio}	21.20	72.47	233.0	95.50
S_{bio}	.23472	.30229	.55266	0.5105

and adjusted mean delays, $D_{c,bio}$, we can determine the mean delay per request: $S_{c,bio} = D_{c,bio}/V_{bio}$. We propose that $V_{c,bio}$ and $S_{c,bio}$ are forescasted using corresponding time-series. The means of future BIO-delays are then determined as $\widehat{D}_{c,bio} = \widehat{S}_{c,bio}\widehat{V}_{c,bio}$.

Summary

We have considered tools and techniques to be used in modeling the workload for practical situations of capacity planning. We have proposed a hierarchical approach to construct the workload of a queueing network model from business plans of an organization.

We have shown that the workload modeling can, to a great extent, be based on the VAX/VMS Accounting data. However, the io-workload is measured in VAX/VMS as io-operation counts. This creates problems that can be solved only using additional measurement facilities and/or adjustments based on operational analysis of queueing networks. We have shown that a combination of a few benchmarks and operational analysis produces an identification of reasonable accuracy with the required amount of detail.

The overall conclusion of our experiences is that the hierarchical workload modeling provides a sound basis for workload prediction and for identification of the workload model. The standard measurement facilities of VAX/VMS are extensive, but by no means complete. In particular the Disk IO-Subsystem of VAXCluster should have better monitoring facilities. In the current situation considerable specialized knowledge of the examined system is required in identifying the io workload. We believe that as the starting level of modeling for capacity planning the hierarchical workload modeling is too complex and requires too much cumbersome background work. However, if the data center is systematically managed, and the workload is systematically monitored and reported, we see no reason, why the hierarchical workload modeling should not be productive.

In this paper we have primarily considered identifying the model of current system and workload. We have proposed some possibilities to identify future models of workload. Our research project is currently working on workload forecasting. We hope to be ready in the near future to report our experiences of hierarchical workload forecasting.

References

[1] Denning, P. J. and J. P. Buzen. 1978. "Operational Analysis of Queueing Network Models". *ACM Computing Surveys* **10**, *no. 3*: 225–261.

[2] Lazowska, E. D.; J. Zahorjan; G. S. Graham; and K. C. Sevcik. 1984. *Quantitative System Peformance*. Prentice-Hall, Inc., Englewood Cliffs, N. J.