# The VAX/VMS analysis and measurement packet (VAMP) : a case study

EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computing Science

Memorandum COSOR 88-09

# The VAX/VMS Analysis and Measurement Packet (VAMP): A Case Study

by
J. Hoogendoorn, R.C. Marcelis,
A.P. de Grient Dreux, J. van der Wal,
R.J. Wijbrands

Eindhoven, March 1988
The Netherlands

# 1 Introduction

It can arguably be stated that the concept of a computing machine has been the single most important idea shaping modern society. Computers have rapidly developed from the valve-operated ENIAC machine of the 1950's which filled an entire room to the manufacture of silicon chips with immeasurably more computing power which is smaller than a persons fingernail all the way to massively powerful super-computers.

The amazing thing about this development is that, as yet, there doesn't appear to be an end in sight. In order to survive in the modern world, companies need to be able to out-compute their competitors. The call for more computer power, speed and technology keeps growing at an ever increasing rate. Because of this need, where only a few years ago a single personal computer might have stood, there are now entire computing systems to cater for company needs, complete with a large computing staff under the guidance of a System Manager.

Because of the increasing pressure to have access to computing power, existing computer systems rapidly become congested, trying to handle more users and resulting only in longer delays. The System Manager fights a never ending battle to balance the user requirements and existing computer capacity, and tends to be forever asking for a larger budget to increase system capabilities. Because of the phenomenal speed at which computing systems have been created and the rate at which they continue to grow, it is often very difficult to obtain reliable estimates of the current system performance. It is even more difficult for the System Manager to determine precisely what future trends may be and how best to prevent overtaxing an already overworked system.

Many of the problems with existing systems are only discovered by the users themselves, who then send complaints to the System Manager. He in turn relies heavily upon his own personal experience in an attempt to solve these problems.

The area of computer system performance analysis is a relatively new field of study. By the use of various monitoring programs it is possible to obtain information about existing computer systems and to obtain an overview of certain trends over time (e.g. there will *always* be more users in the future!). The major task of a performance analyst is to attempt to find so-called *bottlenecks* within the system, that is those points where longer than expected (or wanted) delays occur. The Performance Analyst can offer certain tools to the System Manager whereby these bottlenecks can be found and some attempt is made to predict future system performance. Through the use of such primitive *Decision Support Systems*, a manager can quantify some of his ideas as to how a system can be bettered, and compare the performance of these theoretical systems in order to determine the best approach to attack performance problems. With the aid of such tools, the manager then has more than just his feelings as to what can best help a system, and with these concrete results can then proceed to alter the existing sys-

tem accordingly. Just as importantly, with some definite answers, the System Manager can put forward a convincing argument to the company directors for an increased budget.

The area of Performance Analysis thus has an important future in helping to reduce the trial-and-error attempts at effective system management.
At the moment, Performance Analysts are very much in the initial stages of developing viable methods which can be used by System Managers. The successful development of a Decision Support System can only occur if there is good communication between the designers (Performance Analysts) and the ultimate users (System Managers) during the design phase.

At the Eindhoven University of Technology (EUT) an attempt has been made to model and analyse the performance of the local VAX/VMS-cluster, consisting of three VAXes sharing background memory. The result of this work is a group of separate programs, which fall under the title *VAX/VMS Analysis and Measurement Packet (VAMP).*

1. The VAX/VMS Operating System includes a standard monitoring program (MONITOR) which measures current system characteristics at regular intervals of 3 minutes throughout the day. This program accumulates data for each VAX computer in the system.
2. At the end of the day, relevant data are extracted from the information gathered by the monitor and are stored on file.
3. An implemented algorithm to calculate models of VAX/VMS-clusters in order to obtain certain performance parameters.
4. Within the context of the user interface, the necessary parameters for the model of the existing VAX/VMS-cluster can be deducted from (amalgamated) daily data files. In this manner the performance in a certain period can be determined.
5. The user interface allows for altering the model of the existing VAX/VMS-cluster in order to make predictions about the performance of a changed system configuration.

In order to use VAMP for the analysis of the existing-EUT cluster, we first had to translate the measurements into parameters for the model. Then, in October 1987, we had an unique opportunity to verify the accuracy of our model. In order to cope with the ever increasing workload, it was decided to change the existing configuration in a number of ways.

In this paper we describe the difficulties encountered in using the VAMP packet to "predict" the performance of the cluster after the change.

In section 2 the cluster before and after the upgrading is introduced. Section 3 discusses the modelling. The actual case study is presented in Section 4. The conclusion drawn in Section 5 is, that even for a relatively small system change as the one studied here, it is very hard to give a good performance prediction.

# 2 Configuration Description

The terminology *cluster* is used by the manufacturer (Digital Equipment Corporation) to describe a system in which a number of individual VAXes are coupled together to allow sharing of a common background memory. The VAX-cluster at the EUT is shown in Figure 1.

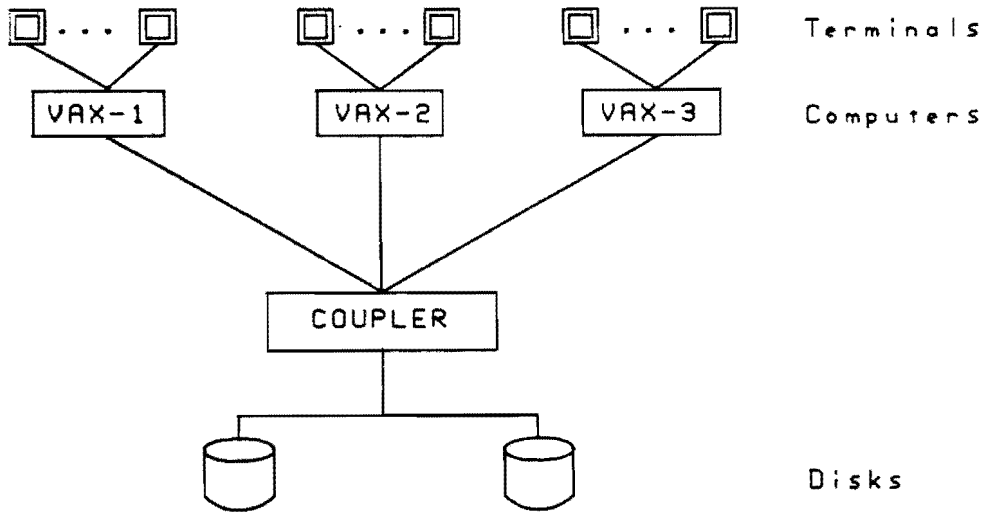


Figure 1: *VAX/VMS-cluster at the Eindhoven University of Technology*

In an effort to avoid confusion, we use the word *configuration* to mean a real existing system, and the word *cluster* to mean a theoretical system which has been created by altering an existing configuration.

The basic structure of the configuration is shown in Figure 1. The old (pre-October) configuration consisted of three identical VAX-11/750 computers each having 6 Mb of main memory. As background memory, there were five faster disk units (RA-81 with a total access time[1] of 38 msec), and two slower disk units (RA-60 with a total access time of 52 msec).

The alterations resulting in the new configuration are as follows:

[1]This access time consists of a *seek* time of 28 msec, a *latency* time of 8.3 msec and a *transfer* time of 1.3 msec based on an average transfer of 4 pages per I/O, see[2].

1. One of the three VAX-11/750 machines (VAX-3 in the diagram) has been replaced by a VAX 8530 machine which is estimated to have a processor speed 5 times as fast as the old VAX-11/750. The main memory of this new VAX has also been extended to 16 Mb.
2. Two of the disks (DISK-6/USER5[2] and DISK-7/POOL) have been replaced by two faster disk units of the RA-81 type with an access time of 38 msec.
3. Two slower disks (DISK-8/USERBD and DISK-9/APPL) of the RA-60 type (access time is 52 msec) have been added.
4. As a result of our long term monitoring of the old configuration (from April '87 to end-September '87) we noticed that one of the disks was quite heavily used (approximately 50% of all disk visits were to this disk). This disk (DISK-4/COMSYS) is a system disk which is used as a sort of temporarily extendible memory. Within the lifetime of a user process, extra *pages* (512 bytes) of memory are allocated when necessary (until a certain maximum limit is reached). If the contents of these pages are altered, and a user is forced to reduce his total available memory (due to a busy system), then the resulting *modified pages* are temporarily written to this system disk until the user needs them or the user process is finished (in which case the modified pages are rewritten to the users own disk space if necessary). In addition to these *paging files* there are also *swapping files* on this disk. When the system becomes busier, a currently non-active (i.e. low priority) process may be temporarily *swapped out* with the contents of the entire user memory space being stored on this disk and retrieved later as necessary. Since, in particular, the page files were in very regular use, we recommended that these files be moved to another disk, along with the swap files (DISK-6/USER5), effectively splitting the workload of the system disk in two (actually reducing the visit frequency by about 40%), with only the normal system command programs left on the original disk.

At the time of the changeover (beginning of October '87), the VAMP packet was still specifically oriented for examination of the old configuration. We rapidly made the packet more general, so that we had measurements from mid-October '87 onwards.

In our attempt to examine the accuracy of our predictions, we used measurements from the old configuration to build our theoretical cluster and compared the predicted performance of this cluster with the measured performance of the new configuration.

[2]When this notation is used, the first name represents the name in our model, and the second is the name in reality.

# 3 VAX/VMS Cluster Modelling

The actual configuration shown in Figure 1 can be modelled as shown in Figure 2. We have ignored the presence of the required coupler, since in practise it is fast enough to handle all traffic without delay.

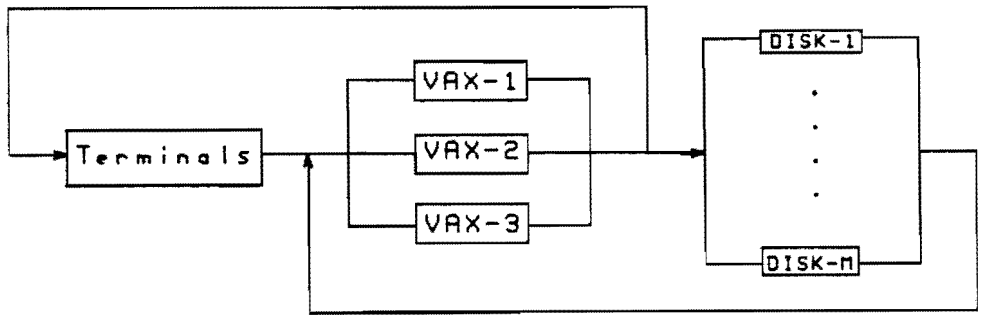


Figure 2: *Initial model for the VAX/VMS-cluster*

Within the VAX/VMS Operating System, it is possible to distinguish between three classes of processes:

1. Interactive Processes - These correspond to users who have logged into the system and are reacting interactively with the computer via a terminal (also known as Real-Time processes).
2. Batch Processes - This class represents user programs which have been started up in a non-interactive mode. That is, they run automatically and require no additional user input from a terminal.
3. System Processes - These are processes which are created by the above two sorts of user processes in order to perform a certain task. Once this task is completed, the system process disappears.

It is necessary to distinguish these classes, because the CPUs also distinguish between them. When processes are being selected for service at the CPU, interactive processes are always chosen before batch processes, since it is supposed that there are users waiting at their terminals for as rapid a response as possible from the computer. Thus interactive processes have a higher *priority* than batch processes. Therefore, as the system becomes busier, with more interactive processes, batch processes receive almost no attention from the CPU, owing to their lower priority. System processes actually have the highest priority of the three classes. However, the VMS monitoring program does not sufficiently distinguish these system processes from the user processes. Since system processes are only in existence when called upon by user processes, we can model the workload of the system processes as merely a part of the user processes, dependent upon their relative

workloads. As a consequence, we have six types of processes circulating through our model - interactive and batch processes for each of the three VAXes.

The different components of our cluster are also known as *stations*. The terminals have been modelled as a single group operating as a so-called *Infinite Server* station (i.e. all processes arriving at the terminal station can be catered for). A process at the CPU receives attention for a fixed length of time (this length of time is known as a *time slice* or *quantum*) and then makes way for the following process, which also receives a fixed service time. Since the amount of time which each active process receives is small (200 msec), we can model the Round-Robin behaviour of the CPU as a *Processor Sharing* station. (the Processor Sharing discipline is an extreme form of Round-Robin, with an infinitely small time slice).

Upon leaving the CPU an interactive process either returns to the terminal or goes to one of the disks. Once the necessary information has been read from the disk, the process returns to the CPU for further service. We therefore have 1 terminal station, 3 CPU stations and a varying number of disk stations (7 disks for the old configuration, and 9 for the new).

As formulated above, we could use a method called Mean Value Analysis[3] to obtain our performance estimates. However, for a problem such as this, with six classes of users, the computational complexity becomes quite large. We have therefore chosen to use a simplified scheme called the Schweitzer-FODI method [4,1]. This method, by the use of some clever approximation schemes has a rapid convergence to a solution for the performance parameters. In order to make use of this technique, however, a number of simplifying assumptions were needed:

1. The disks have a decidedly non-exponential character. To cater for this we have used the Pollaczek-Khintchine formula for $M|G|1$ queues with a coefficient of variation less than one, derived from numerical experiments (see [6]).
2. A problem that we find with monitor programs is that they quite often monitor information which an electrical/computing engineer would need. The problem is that as part of our mathematical analysis, we would like to be able to monitor other system parameters. It proved impossible, for instance to be able to obtain an estimate for the probabilities that, upon leaving the CPU, a process returns to the terminals or goes to a disk. We therefore had to further simplify our model, and this is shown in Figure 3.
3. It seemed impossible to determine a real terminal workload (known as *think time*) from the data collected by the monitor program. We could only obtain a percentage of the time that an interactive process stays at the terminal station. As a consequence, we had to reformulate our problem to make use of relative workloads.

4. Each group of users (for instance a Department at the EUT) has to cope with their characteristic system workload on a specific VAX in combination with space on a specific user disk. This suggests that we should make a further distinction between the interactive processes belonging to a VAX than introduced in Section 3. Unfortunately, information about the connections between VAXes and user disks cannot be deducted from the measurements by the monitor program. Further, the monitor program makes no distinction between the actual tasks of interactive and batch processes (e.g. executing or compiling). Consequently, we had to assume that all interactive processes belonging to a specific VAX consist of one sort of average process. A simular assumption had to be made for batch processes.

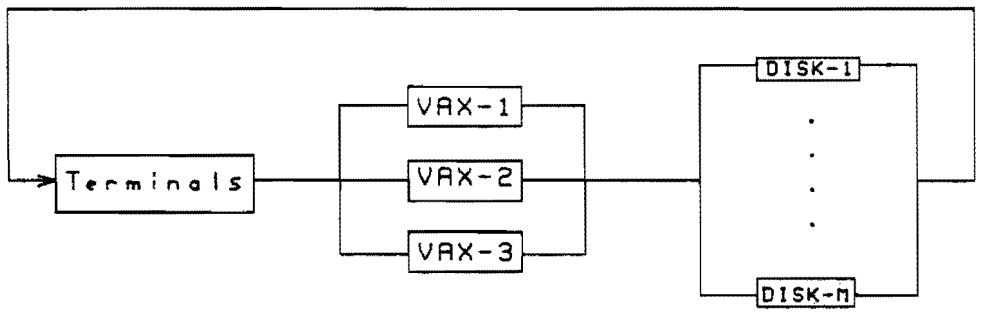


Figure 3: *Actual model for the VAX/VMS-cluster*

# 4 The Case Study of VAMP

## 4.1 Introduction

In the model there are two distinct priority levels at the CPU, separating the interactive and batch processes. At the disks however, both processes are handled on a FCFS basis, and so the interactive processes might be forced to wait for the batch processes. However, because of the low utilization of the disks this hinderance will be minimal. In reality this division at the CPU is not as strict as in our model. Because of the *dynamic* (i.e. varying) priority scheduling scheme implemented by the VMS Operating System (see VAX Software Handbook[5], 296-304), it is possible that batch processes receive a temporary priority boost, giving them precedence at the CPU. This dynamic scheduling

has, unfortunately, proved too difficult to implement in our model.

VAMP uses the concept of the so-called active processes, not to be confused with interactive processes. A process is said to be active if in the last measured interval it used some CPU time or called for disk I/Os. In this way we exclude the inactive interactive processes, in which we are not interested. In the existing version of VAMP, it is necessary for the user to specify the number of active processes to be included in the performance computations. However, it is also necessary to enter the number of batch processes[3]. In order to carry out our performance comparisons we need to duplicate the effect that batch processes have upon interactive. For the purposes of this study, we do not take the batch processes into consideration since, in practise, in any reasonably busy system batch processes receive hardly any attention at the CPU owing to their lower priority. As a consequence, we cannot use our measurements on the configuration (daily day between 8.00am and 5.00pm) in the weekends, because the number of interactive processes is low and hence not representative of a normal configuration. It is of course possible (with a minor adjustment of our program) to study only the data at weekends if we wish to closely study the performance of batch processes.

A performance calculation with VAMP requires a certain date period to be given. In this period there have to be a number of days in which the monitor program has collected data of the systems behaviour. VAMP then determines the necessary model parameters for each VAX per number of active processes (e.g. relative workloads and relative disk visit frequencies). In the performance comparison which follow in Section 4.2, the model parameters used for the performance calculation with the new configuration are determined by data collected by the monitor program for 10 working days over the period October/15/87 - October/30/87. By choosing this fairly small period near the beginning of initial measurements, we hoped that we would have a period in which the system workload (i.e. the type of programs used by users) would stay relatively constant. Over this period we see an average of 3 active processes on VAX-1/TUERC1, 3 on VAX-2/TUERC2 and 13 on VAX-3/TUERC5[4].

The model parameters used for our performance prediction are determined by data collected by the monitor program, followed by an adjustment by VAMP over all measurements with the old configuration, for the period March/28/87 - October/2/87. This period had an average population of (6,5,3) active processes, which deviates quite considerably from the measured average of (3,3,13). Partly this difference will be due to the fact that the System Manager decided that most users on VAX-1/TUERC1 and VAX-2/TUERC2 would be "transferred" to the new faster VAX-3/TUERC5 with the first two VAXes being used almost exclusively for the teaching of computing courses. Also the faster VAX must have attracted some users from other computer systems, such

[3]It should be possible to avoid this in future versions of VAMP.

[4]From now we shall represent the number of active processes on each of the three VAXes by a *population* triple such as (3,3,13), for instance.

as the *Burroughs B7900.* The effects of such an alteration are very difficult to estimate in a surrounding where many users are free to use any computer system they like.
We have not attempted to estimate what the population change would be. Our main interest was whether VAMP would prove to be sufficiently accurate to predict the performance of the new system for the measured population of (3,3,13) active processes.

The main performance measures of interest to us are the *CPU utilization* and the *response time per CPU second* per active process. Since the monitor program does not give us the workload of a job we cannot obtain response times. Fortunately, the notion response time per CPU second, which with some effort we can estimate from the collected data, is quite informative. One of the other performance parameters calculated by our model is the disk utilization.

## 4.2 The Performance Comparison

### 4.2.1 Construction of CLUSTER

VAMP contains a number of options for altering a configuration. With these options we have constructed CLUSTER, with the intention of approximating the new configuration, described in Section 2, as closely as possible. CLUSTER has been built out of the old configuration in the following way :

1. VAX-3/TUEWS1[5] has been sped up by a factor 5. The main memory of this VAX has been made 2.67 times larger than the original, resulting in an increment of the maximum number of 12 active processes, imposed by the number of ports on VAX-3/TUEWS1, to 32.
2. The access times of DISK-6/USER5 and DISK-7/POOL have been decreased from 52msec to 38msec.
3. Two disks, namely DISK-8/USERBD and DISK-9/APPL have been added (access times 52 msec), with a disk workload respectively 5% and 1% of the disk workload of DISK-4/COMSYS. Further we have transferred 40% of the disk workload of DISK-4/COMSYS to DISK-6/USER5. This 40% is to account for the transfer of the page and swap files and is based upon an examination of disk workloads within the first weeks of monitoring the new configuration.

We have carried out the performance prediction with CLUSTER for a population of (3,3,13) active processes.

### 4.2.2 Predicted Performance of CLUSTER versus Measured Performance

In order to get an idea about the differences in behaviour of the users of the three VAXes before and after the upgrading of VAX-3, we first give the measured performance for the

[5]After replacing the third VAX, the system name changed into TUERC5.

population (6,5,3) in the period March- October 1987 in Table 1. Notice that, although the system workload of the interactive users had nicely been spread over the three VAXes, the response times were quite high. This has been one of the reasons for upgrading one of the VAXes.

| | number of active processes | CPU utilization | response time per CPU second |
|---|---|---|---|
| VAX-1/TUERC1 | 6 | 0.54 | 2.34 |
| VAX-2/TUERC2 | 5 | 0.58 | 2.34 |
| VAX-3/TUEWS1 | 3 | 0.54 | 2.09 |

Table 1: Old Configuration - *Measured Performance* in Mar/28/87 - Oct/2/87

The *measured performance* of the new configuration during 10 working days in the period October/15/87 - October/30/87 with the average population of (3,3,13) has been placed in Table 2. It has to be remarked, that when comparing the response time of the first two VAXes with the one of the new VAX-3/TUERC5, the latter response time has to be divided by 5, because all elementary CPU operations are in terms of a VAX-11/750 machine. However, we are mostly concerned with the comparison of the predicted with the measured response times of *corresponding* VAXes so this is unnecessary.

| | number of active processes | CPU utilization | response time per CPU second |
|---|---|---|---|
| VAX-1/TUERC1 | 3 | 0.47 | 1.73 |
| VAX-2/TUERC2 | 3 | 0.27 | 1.86 |
| VAX-3/TUERC5 | 13 | 0.46 | 2.92 |

Table 2: New Configuration - *Measured Performance* in Oct/15/87 - Oct/30/87

Compared with the performance of the old configuration, we see a definite improvement. The performance for VAX-1/TUERC1 gives a good indication that indeed the system workload has changed. Although the average number of active processes has been halved, the CPU utilization is only 15% lower.
As we can see from the CPU utilizations for VAXes 1 and 2 in Table 2 the VAX-1/TUERC1 users have used more computing power than VAX-2/TUERC2 users, but the response time per second CPU is slightly better for VAX-1/TUERC1. This suggests that the measurements are not accurate or that the difference in behaviour of the two user classes is substantial. In order to see the effect of the relatively small measurement period we also give the measured performance for the period October/15/87 - November/4/87, with 3 extra working days, see Table 3.

| | number of active processes | CPU utilization | response time per CPU second |
|---|---|---|---|
| VAX-1/TUERC1 | 3 | 0.44 | 1.73 |
| VAX-2/TUERC2 | 3 | 0.28 | 1.80 |
| VAX-3/TUERC5 | 13 | 0.49 | 2.90 |

Table 3: New Configuration - *Measured Performance* in Oct/15/87 - Nov/4/87

We see that the measurements with three extra working days do differ, but it remains very likely that the users of VAXes 1 and 2 are different. In the remainder of this paper we will use the measured performance in Table 2.
The predicted performance of CLUSTER is presented in Table 4

| | number of active processes | CPU utilization | response time per CPU second |
|---|---|---|---|
| VAX-1/TUERC1 | 3 | 0.32 | 1.87 |
| VAX-2/TUERC2 | 3 | 0.39 | 1.86 |
| VAX-3/TUERC5 | 13 | 0.50 | 5.19 |

Table 4: CLUSTER - *Predicted Performance*

Clearly, there is a substantial difference between the measured and the predicted performance. The CPU utilization for VAXes 1 and 2 together is predicted well, but the split over the two is not good. The response time per CPU second is quite good. For VAX-3/TUERC5 however, the predicted CPU utilization is good, while the predicted response time is completely wrong. Can we find a logical explanation for this?

## 4.3 Explanations for the bad behaviour of the predicted Performance of VAX-3/TUERC5

In search of an explanation, we have looked at the differences between the measured and the predicted performance. We observed a major difference in the disk utilizations, see Table 5.
Looking at the user disks, we see that only DISK-3/USER3 has a rather accurate predicted utilization. The predicted utilizations of the remaining user disks are much too high. At this point we may state that together with the change in the system workload, the disk visit frequencies of several user disks has been altered drastically and that information about the connection between VAXes and user disks (model simplification mentioned in Section 3) is essential for obtaining good estimates for the measured disk utilizations of user disks. Further we see that the predicted utilization for the system

| | measured utilization | predicted utilization |
|---|---|---|
| DISK-1/USER1 | 0.041 | 0.128 |
| DISK-2/USER2 | 0.128 | 0.223 |
| DISK-3/USER3 | 0.086 | 0.101 |
| DISK-4/COMSYS | 0.190 | 0.434 |
| DISK-5/USER4 | 0.041 | 0.110 |
| DISK-6/USER5 | 0.052 | 0.328 |
| DISK-7/POOL | 0.216 | 0.205 |
| DISK-8/USERBD | 0.029 | 0.035 |
| DISK-9/APPL | 0.000 | 0.003 |

Table 5: Disk Utilizations

disk is much too high, while the pool disk has an accurate prediction.

Another way of presenting the difference in disk utilizations, is to look at some model parameters, in particular the quotient of the relative CPU workload and the relative disk workload, see Table 6. The relative disk workload per active process belonging to a particular VAX, consists of an amalgamation[6] of several data items collected by the monitor program and is the same averaged value for each disk. In our model the relative disk visit frequencies ensure that the distribution of processes over the disks is quite accurate.

| | relative CPU workload : relative disk workload |
|---|---|
| prediction | 1 : 2.56 |
| measurement | 1 : 1.06 |

Table 6: Relative Disk Workload Comparison

We see that the relative disk workload used in the prediction is more than twice as large as in the measurement. We now want to find out what the effect of this is on the predicted response time per CPU second of VAX-3/TUERC5 in Table 4 by means of an *adjustment* to this response time. Let us first analyse the components of the response time per CPU second.

$$\textit{Response time per CPU second} = 1 + \textit{CPU wait time} + \textit{disk response time}$$

[6]The relative disk workload per active process consists of the sum of the disk access time times the number of disk I/Os generated by active processes belonging to the VAX under consideration, divided by the total number of active processes belonging to this VAX.

In particular we want to adjust the predicted disk response time for processes belonging to VAX-3/TUERC5, by means of a multiplication with the quotient of the predicted and measured disk response time. Therefore we consider as in our model each disk as an $M|G|1$ queue with FCFS service discipline in order to obtain a measure for the disk response time, which makes use of the disk utilizations in Table 5. Let $m_i$ and $\rho_i$ be respectively the access time of disk $i$ (38 or 52 msec) and the utilization of disk $i$ , $i = 1, \ldots, 9$. Then we can compute the Poisson Arrival intensity at disk $i$ , $\lambda_i = \frac{\rho_i}{m_i}$. The measure for the response time for disk $i$ , $S_i$, becomes

$$S_i = \frac{\rho_i}{1 - \rho_i} R_i + m_i$$

with $R_i$ the residual service time of disk $i$. As a measure for the disk response time, $S$, we obtain

$$S = \sum_{i=1}^{9} \lambda_i \, S_i$$

If we set $R_i = \frac{2}{3} m_i$, as we have also done in the model, we obtain the following quotient

$$S_{predicted} : S_{measured} = 1 : 0.44$$

We are now able to make the actual adjustment, by multiplying the predicted response time with this quotient. After examining some extended results of the predicted performance, we found that the predicted response time equals 3.38. So the adjusted predicted response time becomes 1.49. As adjusted predicted response time per CPU second we get a value of 3.30. This certainly closer approximates the measured value of 2.92 (see Table 2).
We also made a performance calculation with CLUSTER with for VAX-3/TUERC5 a relative disk workload of 1.06, obtained from the measured performance (see Table 6). This gave a predicted response time for one CPU second of 2.53.
Concerning these two adjustments, it is evident that the high error in the relative disk workload contributes significantly to the total error.

We have therefore reduced the problem of the bad behaviour of the predicted response time per CPU second for VAX-3/TUERC5 to the question why the relative disk workload and particularly the number of disk I/Os of the prediction is this high. The ratio of the main memory of the old and new VAX-3 is 6 Mb : 16 Mb = 1 : 2.67. For the number of active processes this is 5 : 13 = 1 : 2.6 (with the performance calculation with CLUSTER the model parameters for 13 active processes are in fact the model parameters of the old configuration for 5 active processes). Concerning this, starting from a proportional number of disk I/Os per unit time, the number of disk I/Os in the prediction should have remained the same as in the measurement. There seem to be two possible explanations. The first one concerns the memory allocation strategy of the system.

- The VAX/VMS Operating System contains a system parameter controlled by the System Manager which is called the PFRATH (Page Fault RATe High). Roughly, this parameter has the following effect. If a process generates more than PFRATH page faults per CPU second, then this process receives an increase of WSINC (another system parameter) pages in main memory. In the old configuration PFRATH was set for each VAX on 12 per CPU second. After the changes in the configuration this parameter has remained the same for each VAX, even though VAX-3/TUERC5 has been quickened 5 times. Since starting processes on this VAX will generate more page faults per CPU second, so they are allocated more pages in main memory and will, in the long run, generate less disk I/Os.

The second argument concerns the increase of flexibility due to extending main memory.

- Let the number of pages, a process would like to have at a certain moment, be denoted by the random variable $X_i$. Assume the $X_i$ are independent and identically distributed. Let $M$ be the number of pages available per process. Then

$$\frac{1}{k}E[X_1 + \cdots + X_k - kM]^+ \geq \frac{1}{k+1}E[X_1 + \cdots + X_{k+1} - (k+1)M]^+$$

  I.e., the average shortage per process decreases if the number of processes is increased, while the average number of pages per process is kept at the same level. Since the average number of active processes on VAX-3/TUERC5 has been increased from 5 to 13 and memory from 6 Mb to 16 Mb the above reasoning might be an explanation.

Of the two arguments above, the first one is the most important. Main memory is usually not completely used, so the limitations per process seem to be more restrictive than the overall limitation due to the total main memory size.

# 5 Concluding Remarks

In this performance study we considered the upgrading of a VAX cluster replacing one of the CPUs by a much faster one. In doing so we have run into two major problems. It is necessary that we are able to predict the change in behaviour of the users once a new configuration has been installed. And it is vital that we understand how the number of page faults depends on the strategy used to allocate pages in main memory. For changes in the system configuration as those studied here, we have to know how to deal with these two problems before sufficiently accurate predictions of future performance will be possible. Only after looking at the measurements it became clear how to use the parameters of the model.

Probably the change in user behaviour will be relatively small in many professional situations. But the memory allocation problem is always present, and it still requires a lot of effort to properly model it.

# References


[1] Doremalen, J.B.M. van, Wessels, J., Wijbrands, R.J., *Approximate Analysis of Priority Queueing Networks* in O.J. Boxma, J.W. Cohen, H.C. Tijms (eds.) *Teletraffic Analysis and Computer Performance Evaluation* (North-Holland, Amsterdam, 1986), 117–131.

[2] Grient Dreux, A.P. de, *Performance Onderzoek naar het TUE VAX-cluster systeem*, Afstudeerverslag, 1987. (In Dutch)

[3] Reiser, M. and Lavenberg, S.S., *Mean-Value Analysis of Closed Multichain Queueing Networks*, J.ACM **27** (1980), 313–322.

[4] Schweitzer, P.J., *Approximate Analysis of Multiclass Closed Networks of Queues*, Lecture presented at *The International Conference on Stochcastic Control and Optimization*, Amsterdam (1979).

[5] *VAX Software Handbook* (Digital Equipment Corporation, USA, 1982).

[6] Wijbrands, R.J., *Queueing Network Models and Performance Analysis of Computer Systems* Ph.D Thesis, Dept. of Math. and Comp. Sci., Eindhoven University of Technology (1988).