# Predicting Performance in UNIX Systems From Portable Workload Estimators Based on the Terminal Probe Method.

by
Luis Felipe Cabrera
and
Germán Rodriguez-Galant

## INDEX

# Predicting Performance in UNIX‡ Systems From Portable Workload Estimators Based on the Terminal Probe Method

Luis Felipe Cabrera †
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720, USA

Germán Rodriguez-Galant
Departamento de Estadística
Facultad de Matemáticas
Pontificia Universidad Católica de Chile
Casilla 114-D, Santiago, Chile

*Labor omnia vincit.*
Roman proverb.

## Abstract

The characterization and prediction of the behavior of a computer system is one of the main goals in any performance evaluation study. Moreover, if one is interested in comparing different computer installations, which may have diverse user communities and hence different natural workloads, workload estimation and comparison, as well as user utilization, become central issues of discussion.

In this paper we present the results of using a portable workload estimation technique, based on the terminal probe method, to characterize the natural workload of interactive computer installations and predict selected performance indices of interest. Studies performed on 12 computer systems show that there are system independent statistical characteristics of the observed performance indices which allow us to model system behavior and to compare installations. These results can be used as a basis for load balancing strategies in distributed systems. We have found a family of statistical models which fit our measured data in a comprehensive way: not only the mean and the variance of our distributions are well approximated but the modelled distributions fit the observed ones. Thus, order statistics can also be obtained.

All of the observed systems operate under the UNIX operating system. The methodology utilized is easy to understand, quick to implement, and may be applied without bringing the systems down. The more than 16,000 measurements used for this study were obtained over a span of two and a half years. Detailed analysis done with these data enable us better to control the duration of the data gathering period and, what is more, achieve predetermined statistical confidence in the modelled distributions.

**Key Words:** Benchmarking; installation comparisons; performance; performance indices; terminal probe method; UNIX operating system; portable workload estimators; load balancing; generalized

---

† On leave from the Departamento de Ciencia de la Computación of the Escuela de Ingeniería of the Pontificia Universidad Católica de Chile. This research has been partially financed by the research project DIUC 47/82 of the Dirección de Investigación of the Pontificia Universidad Católica de Chile.

‡ UNIX is a trademark of AT&T Bell Laboratories.

linear models; linear predictors.

## 1. Introduction

The utilization of portable software tools has become not only an economic necessity but also an invaluable asset in any multiple-installation computing environment. In particular, the possibility of using hardware independent portable performance monitoring software is a reality since the advent of operating systems which run on several machines from different manufacturers and possibly multiple models from any given manufacturer. We have used a UNIX portable set of benchmarks to gather statistics in all of the systems.

In this paper we will present a study which shows that there exist system and task independent statistical properties of performance data gathered using the Terminal Probe Method (described in Section 2.1) which enable us to draw firm conclusions from studies based on approximate workload estimators. In fact, our results can be used as a basis for establishing load balancing, or task allocation, strategies in distributed systems. Moreover, this study also shows how one can limit the duration of the data gathering period, and how, when modelling a system, one may select a few predetermined values of the workload estimators as the sole objects of measurement. These considerations enable us to reduce the data gathering overhead.

Over the years the so called Terminal Probe Method has been utilized in a variety of contexts. Indeed, not only has it been used to assess a specific configuration under a controlled workload, but there have also been studies which used it as a tool for comparing different installations operating under their "natural" workloads. Thus, questions regarding the representativeness and validity of these comparisons have been posed. These questions have remained unanswered. In this paper we address the problem of inferring that a behavior observed through imprecise workload estimators corresponds to the real behavior of an installation. Moreover, we are also able to use this observed behavior to rank different installations according to predetermined performance criteria, and to model the behavior so as to estimate the impact of load growth.

Our results are based on 16,583 measurements taken on 12 systems over a span of two and a half years. All of the systems were presented with the same benchmark, which runs on any UNIX system. The methodology used is easy to understand, quick to implement, and does not require bringing the systems down for its application. The set of benchmarks which were developed may be installed in a few minutes.

Section 2 briefly explains the computer systems studied, the data gathering method, and the tasks chosen for measurement. Section 3 presents the analysis of the observed data. Section 4 the approximation and the modelling of our data using Generalized Linear Models. Section 5 contains a study of the robustness of our modelling. In Section 6 we present some "characteristic" surfaces for our systems, and, finally, we present our conclusions in Section 7.

## 2. The Experimental Environment

All of the installations used for our measurements ran versions of the UNIX operating system. UNIX is the name of a family of operating systems, first developed at Bell Laboratories, which has evolved over the last 15 years [12,18]. In 1969 a first version was implemented on a PDP-7, and since then the system was ported to PDP-11s, VAXes, IBMs, Amdahls, Interdatas, NCRs and many others. In 1979, a group at the University of California at Berkeley implemented a paged virtual memory extension to UNIX for the VAX [1], which is now part of what is known as "Berkeley UNIX". Most of our measurements were done on this last type of system.

On logging into a UNIX system, each user is assigned a special process containing a command interpreter, known as the *shell* [3], which listens to the terminal. The shell parses each input line and decodes the command requested along with its flags and arguments. Then it *forks* and *execs* the command. Shells may also read commands from files. Thus, users may define sequences of shell commands, known as *shell scripts*, and store them in files for later invocation. The versatility of these scripts is greatly enhanced by the fact that the shell language also contains control-

flow primitives, string-valued variables, and arithmetic facilities. Since UNIX automatically handles all file allocation decisions, the portability of these scripts is greatly facilitated.

### 2.1. The Data Gathering Method

Our strategy for monitoring the responsiveness of each system was the same as that used in [4]. It consists of running a script which invokes a set of predefined benchmarks together with commands to gather statistics about the workload and measure the time it takes the benchmark to terminate. The script runs periodically in a totally automatic way. Each time the script cycles through its commands, it executes a *sleep* command that suspends its execution and then wakes it up after a predetermined number of seconds. We used 1200 seconds, so that there would always be 20 minutes between any two measurement points. We decided to use the *time* command because of our commitment to use standard UNIX tools. *time* has a rather low resolution and truncates its measurements, it does not round them off. More details can be found in [4,5,18].

This data gathering technique can be categorized as a time-sampling method [10], and is in fact very similar to Karush's terminal probe method [11]. By using it with a system in normal operation, one evaluates the performance of an installation (i.e., the hardware configuration plus the workload being executed).

Table 2.1.1 presents a summary of the number of measurements for the various systems. Hardware changes were made to some installations during the measurement period. For statistical reasons we had to consider the systems which existed before and after the change as being different. (For example, in Table 2.1.1 VAA and VAB correspond to two such systems. This change, a fairly trivial one, consisted of the addition of some ports to the installation.) The most important mnemonic element in the names is given by the first letter. Those which begin with a P are the names of PDP-11 systems. Those with a V are VAX systems. All VAXes were 11/780 [7], with the exception of V50, which was an 11/750. If a 7 appears, the system was an 11/70; a 4 means an 11/40.

| System | No. of Measurements |
|---|---|
| VI | 3427 |
| P7A | 3206 |
| V50 | 2718 |
| VE | 2052 |
| VM | 1467 |
| P7B | 1016 |
| VAA | 844 |
| VC | 791 |
| VB | 435 |
| P4A | 301 |
| P4B | 235 |
| VAB | 91 |
| Total | 16583 |

Table 2.1.1: Systems measured and total number of observations.

### 2.2. The Systems Measured

There were a total of five different installations, four at the University of California, Berkeley, and one at Purdue University. Those measurements labeled VC come from Purdue, where the VAX 11/780 had, at that time, a configuration with 3 megabytes of main memory, 56 ports, three RM03 disk drives on Massbus 0, and one TE 16 tape drive on Massbus 1. V50 was our only VAX 11/750-based installation. It had 2 megabytes of main memory, three 330 megabytes disks on a

Massbus controller, 23 ports, and connections to three Ethernets. It ran Berkeley UNIX 4.2 BSD.

Measurements from P4A and P4B came from a PDP 11/40 which had 200 kilobytes of main memory, one DIVA disk controller, and three DIVA disk drives with 50 megabytes disks. This installation had 23 ports and no floating point arithmetic unit. The P4B measurements were made on the installation without a cache memory, and the P4A with a 2 kilobytes cache memory. The P7B measurements were made on a PDP 11/70 with 1.3 megabytes of main memory, a 2 kilobytes cache memory, 81 ports, one DIVA disk controller with four DIVA disk drives, and an RS04 fixed head disk used as a swapping device. The P7A measurements were made on the same installation with 2 megabytes bytes of main memory and with the drum used for storing temporary files instead of as a swapping device.

All of the other measurements were made on the same installation, which went through successive changes. VB was an 11/780 with 512 kilobytes of main memory, two RP06 disk drives, one TE 16 tape drive and 16 ports. This was a "swapping" UNIX system. VAB was the same installation but running a paging version of UNIX. VAA was VAB with 8 more ports installed. VM, VI, and VE, all ran paging versions of UNIX. Only VE ran the 4.2 BSD version of Berkeley UNIX. VM had 2 megabytes of main memory, 2 RP06 disk drives, two CDC 300 megabytes disk drives, and 32 ports. VI had 4 megabytes of main memory, 4 RP06 disk drives, two CDC 300 megabytes disk drives, and 72 ports. VE, finally, had 8 megabytes of main memory, three 300 megabytes disk drives on a Unibus controller, two 167 megabytes disk drives on two Massbus controllers, one tape subsystem, 72 ports, and one connection to an Ethernet.

### 2.3. Choosing Tasks and Work Load Estimators

Throughout our data gathering period we ran a script which contained three basic tasks: a C compilation, a CPU bound job, and a text formatting job. This last task was the command *man man* which retrieved and formatted the manual page entry of the on-line manual. † We also measured three workload estimators: the number of users logged in (*nu*), the number of processes in the process table (*np*), and the number of active users (*nau*). *nau* was obtained by counting the number of ports which had more than one process associated with them. This generated a bias in systems where there were several port-dependent daemons running. No correction was made because the same number of them existed during the entire data gathering period. These "shifts" by a constant do not affect statistical correlations.

At a later stage, and only for the systems VI, VE, V50, and P7A, we appended two new tasks to exercise aspects of the systems which were found not to be well observed previously. These new tasks did not alter the measurements of the initial set of them. They were the copying of a 60 kilobytes file within the same disk, and an editing session involving a series of commands made to a 60 kilobytes file. These measurements were used to validate hypotheses about the systems, but are not included explicitly in our subsequent discussion.

The selection of these portable workload estimators and of the tasks has been justified elsewhere [4,5,6]. The main tradeoffs in choosing the tasks were resource consumption versus workload representativeness of the (original) set of activities. The low granularity of the *time* command played a role in that it forced us to make our tasks longer than we would have liked. As we wanted less than 5% error in each individual measurement, our tasks had to run for a sufficiently long period of time. *time* returns three values: user time, system time, and response time. The first two are accurate to one-tenth of a second. Response time is accurate to the second. Thus, each of our tasks was chosen so as to take, in most cases, more than 5 seconds to complete. For the workload estimators the criterion was semantic simplicity. However, as *nu*, *nau*, and *np* are

† Some recent optimized versions of *man* format the data only if it has not been retrieved for a long period of time by preserving the formatted versions in files. Berkeley UNIX 4.2 BSD does this, as can be observed in Table 3.4.1.

related to each other, it was of interest to determine whether we could explain the same phenomena with just a subset of them. Thus, we studied their correlations. These results are presented in Section 3.1.

## 3. Analysis of the Data

The explosive combinatorial characteristics of the amount of data which was collected and analyzed for this study precludes us from explicitly presenting all the different cases. We have chosen, instead, to select appropriate representative displays for each of the analyses done. We have heavily relied for these analyses on SPSS and Minitab [14,17].

Using the SPSS facility called *scattergram* [14], we were able to plot each of the observed performance indices for our different tasks as an individual function of each of our workload estimators. Moreover, scattergram also gives us an idea as to how each cross section distribution looks like, i.e., the distribution obtained for each individual value of the workload estimator. It is unfortunate that scattergram prints values only through the number 9, thus preventing immediate sizing of samples which have more than 9 points. Further analyses enabled us to plot the exact histograms of these distributions, and hence see their actual shape.

### 3.1. Correlations Between Work Load Estimators

The choice of *nu*, *nau*, and *np* as portable workload estimators has been justified primarily in terms of their semantic simplicity and ease of measurement [4,5,6]. Given the amount of data available, a question worth answering was whether a subset would suffice. Can we explain (in the statistical sense of this term) the same phenomena with one of them? With two? Thus, the correlation between pairs of them and their prediction power were studied. Table 3.1.1 presents the results obtained for the correlation coefficients between each of the three pairs of workload estimators used, computed using all the available measurements in each system.

| | Correlation Coefficients | | |
|---|---|---|---|
| System | *nu* vs *nau* | *nau* vs *np* | *nu* vs *np* |
| VI | .899 | .799 | .689 |
| P7A | .917 | .849 | .791 |
| V50 | .954 | .942 | .933 |
| VE | .901 | .885 | .868 |
| VM | .880 | .854 | .783 |
| P7B | .946 | .903 | .884 |
| VAA | .852 | .838 | .746 |
| VC | .881 | .769 | .686 |
| VB | .842 | .855 | .771 |
| P4A | .677 | .828 | .660 |
| P4B | .410 | .345 | .470 |
| VAB | .492 | .906 | .606 |

Table 3.1.1: Correlation coefficients between workload estimators for the different systems.

From Table 3.1.1 we observe that there exists substantial correlation between the different pairs of estimators. The coefficients are certainly larger than those observed in most of the social sciences studies, which are usually on the order of .4, and smaller than those obtained in economics, which are usually on the order of .95. Moreover, we also see that the pair which tends to be less correlated is $<nu,np>$, which suggests that these two estimators measure different aspects of the workload. We also observe that, for P4A and VAB, *nau* and *np* are much more highly correlated than the other two pairs. In Section 4 we shall see that, for most systems and tasks, *nau*

adds no significant additional information to that given by *nu* and *np*.

The fact that P4B presents values which are very different from those of all the other systems is explained by us in terms of hardware configuration. This system was a PDP 11/40 with no cache memory. Response time because of this suffered a tremendous degradation. We believe the user community changed its work habits while the system temporarily operated under these conditions. P4A was the same installation with cache memory.

Our scattergram analysis told us that the high correlation values between workload estimators came from data with almost no outliers. All of the data points were in one compact cloud. The exception to this rule was observed in P7A, and is depicted in Figure 3.1.1. In this figure, the secondary cloud, that parallel to the horizontal axis, contains a small number of points, and depicts the system on a large number of users logged in but with very few doing work.

We also analyzed the dependency of our workload estimators on the time of day. The regularity observed for all of them, reflected by the low dispersion of values and the periodicity, supports the claim of stability for the natural workloads. User communities seem to achieve steady state working habits soon after hardware changes have occurred. Figure 3.1.2 depicts, for system V50, the number of processes as a function of the time of day. Figure 3.1.3 presents, for system VE, the number of active users as a function of the time of day.

### 3.1.1. Robustness of These Measurements

The correlation between workload estimators is a function of the working habits of the installation's user community. Table 3.1.1.1 presents, for each system, the correlation coefficients between pairs of our workload estimators obtained from random subsamples of the totality of the measurements of our workload estimators. Indeed, the file containing all sample points was sequentially scanned, and for each measurement a 'random' decision was made as whether to

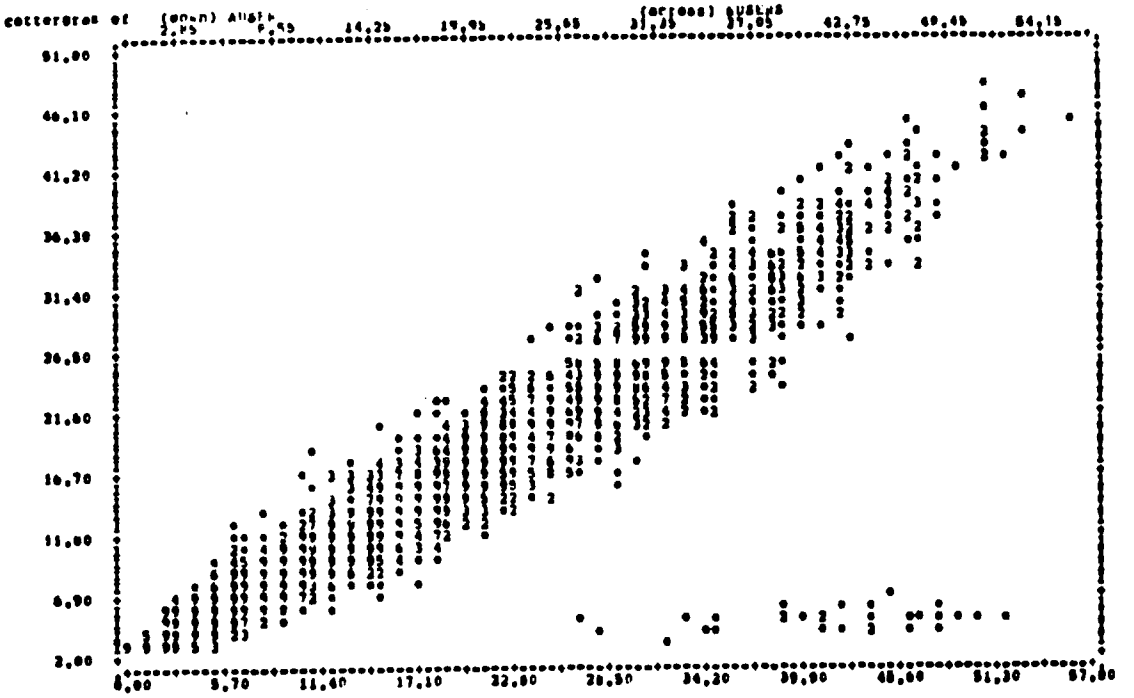

Figure 3.1.1: System P7A. Scattergram of AUSER (*nau*) versus NUSERS (*nu*).

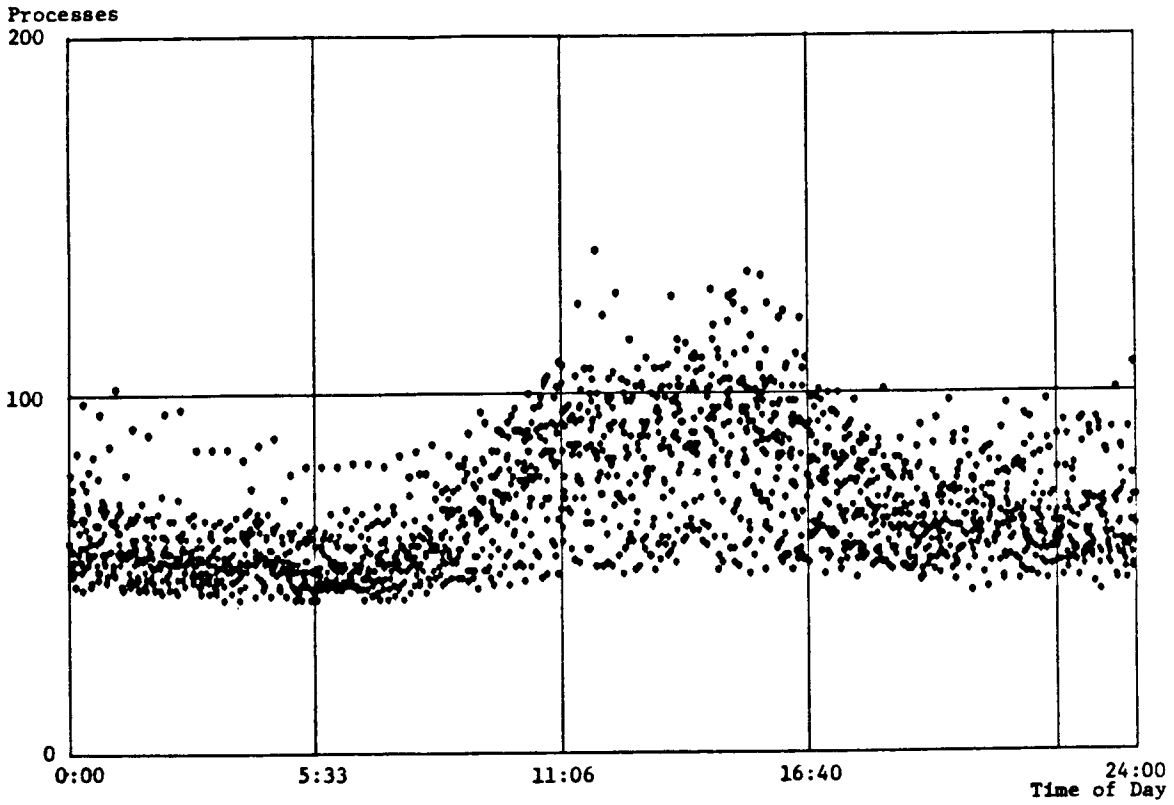


Figure 3.1.2: System V50. *np* versus the time of day.

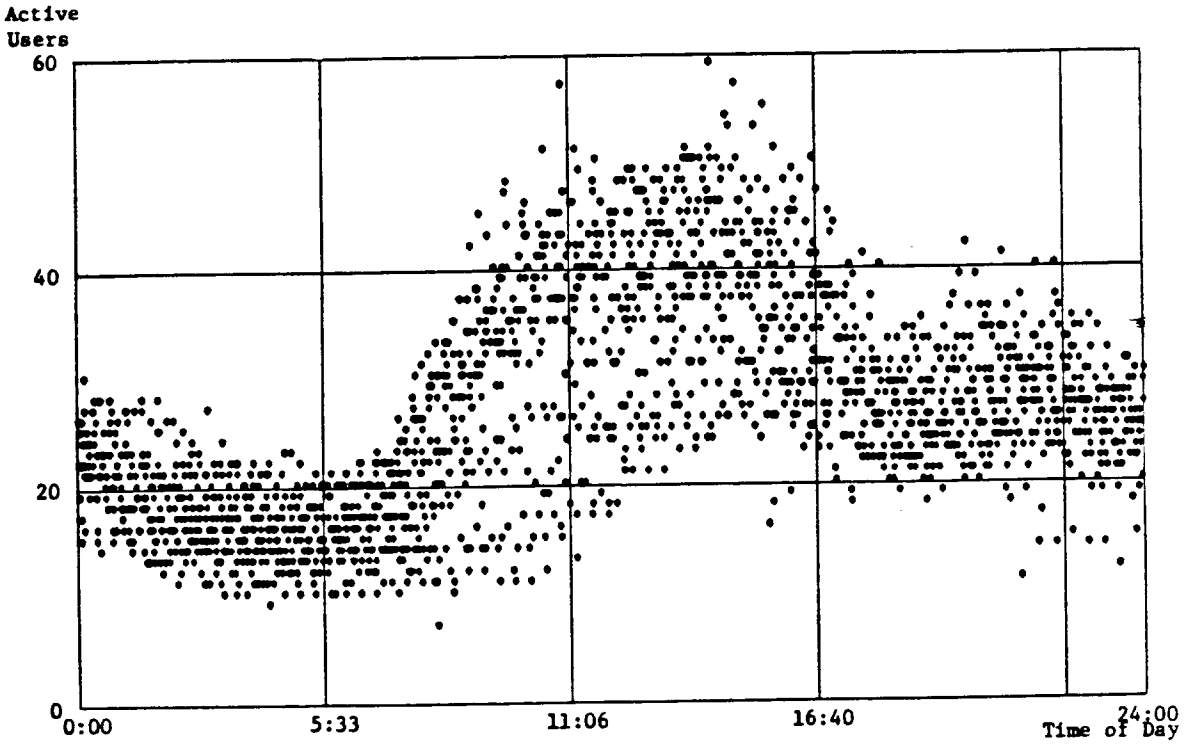


Figure 3.1.3: System VE. *nau* versus the time of day.

| System | Number of measurements | % of total measurements | Correlation Coefficients | | |
|---|---|---|---|---|---|
| | | | *nu* vs *nau* | *nau* vs *np* | *nu* vs *np* |
| VI | 1708 | 49.78 | .896 | .785 | .668 |
| P7A | 1471 | 49.71 | .909 | .849 | .795 |
| V50 | 1366 | 50.25 | .954 | .933 | .941 |
| VE | 1031 | 50.24 | .902 | .884 | .877 |
| VM | 740 | 50.04 | .880 | .860 | .789 |
| P7B | 515 | 50.06 | .948 | .899 | .885 |
| VAA | 425 | 50.03 | .842 | .821 | .740 |
| VC | 398 | 50.31 | .887 | .776 | .684 |
| VB | 225 | 51.72 | .836 | .861 | .769 |
| P4A | 156 | 51.65 | .705 | .834 | .698 |
| P4B | 114 | 48.30 | .384 | .343 | .489 |
| VAB | 39 | 42.85 | .339 | .856 | .481 |

Table 3.1.1.1: Correlation coefficients of random subsamples.

incorporate or not that measurement into the subsample. The size of the subsample and its percentage of the size of the whole sample are indicated.

By comparing Table 3.1.1 with Table 3.1.1.1, we see that there is indeed great stability in these numbers. Another remarkable fact is that all relative orderings among coefficients are preserved. In all systems the orderings of the correlations coincide. This clearly indicates a high degree of uniformity, or stability, in user habits. On the other hand, some hardware changes do bring different behavior patterns, as can be seen with P4B and P4A, VAB and VAA. From an overall analysis of the behavior of each installation which underwent hardware changes, we were able to detect changes in the user habits only when the hardware additions significantly altered the responsiveness of the system at all levels of the load.

### 3.2. User Time Measurements

User time, as defined by the UNIX programmer's manual [18], is the time spent in user mode during the execution of a command. As our tasks were always run using the same set of inputs, their behavior does not change between runs. The same user mode instructions are repeatedly executed. Thus, a precise clock should always give the same reading. For the same reason, user time measurements should not depend on the load of the system. Thus, for them, the accuracy of the workload estimator is not relevant regarding the precision of user time measurements. The same measurement should be observed at all levels of load. Nevertheless, because of the way *time* is implemented (sampling at clock "ticks") and of its resolution (only to the tenth of a second), we expected to see measurement errors. Because of their random nature, we believed they should be normally distributed. This turned out to be the case for most tasks and systems. In all cases, though, the normal distribution was the best (and statistically valid) approximation to the observed distributions for user time.

We have chosen to display scattergrams of that task which was most amenable to sampling error: the CPU bound job. Other tasks had even more normal user time distributions. Figures 3.2.1 and 3.2.2 display scattergrams of our CPU bound task in two different systems. Figure 3.2.2 has a good number of 'lower' outliers, but the central trend is clearly that of normal distributions. In both figures we verify that the median and mean values of each cross section remain practically constant at all values of our workload estimator. As expected, user time is observed to be practically independent of load.

User time measurements also shared the following characteristic: the means and variances of the distributions corresponding to a task in a system showed no real correlation. Figure 3.2.3 depicts the variance (VAR) plotted against the mean (MEAN), for the task *man man* in the system P7A, where *nu* was the underlying workload estimator. That is, for each value of *nu*, the mean

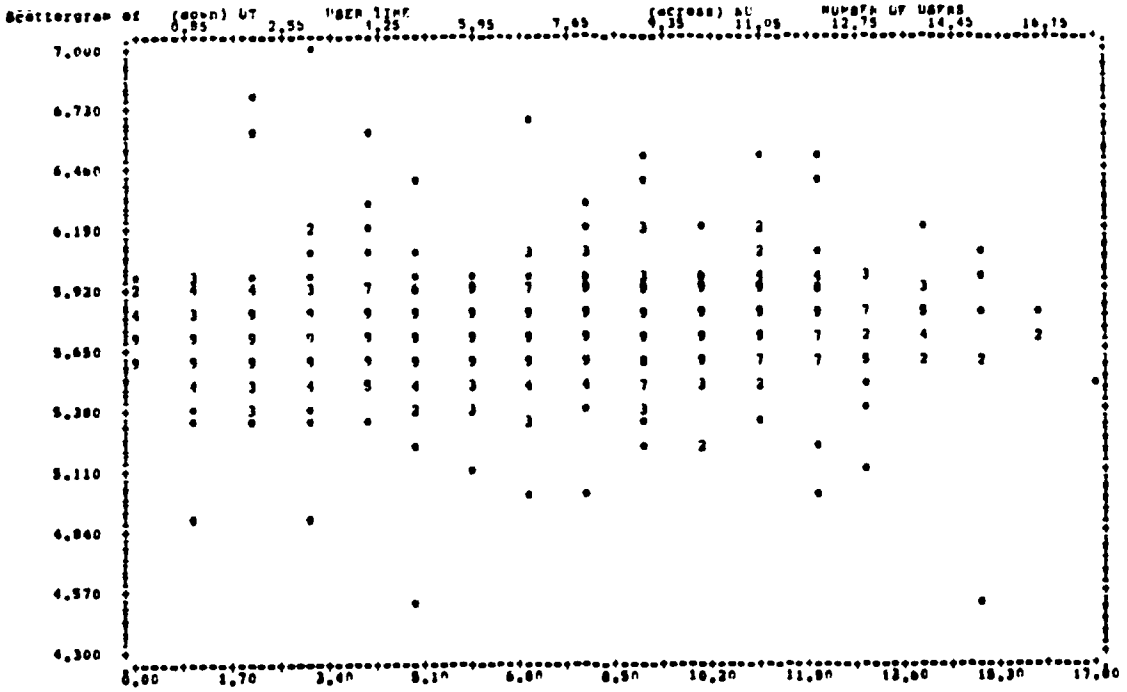

Figure 3.2.1: System VM. Scattergram of UT (user time) versus *nu*. Task: CPU bound job.

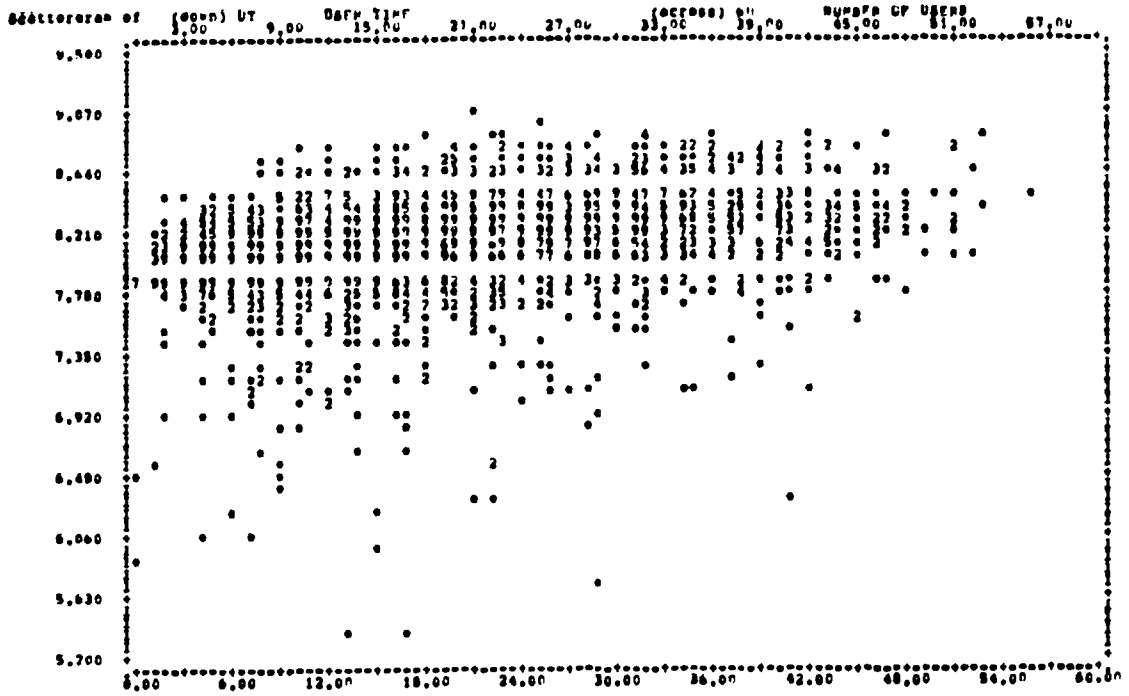

Figure 3.2.2: System P7A. Scattergram of UT (user time) versus *nu*. Task: CPU bound job.

and variance of the distribution of user time (a cross section of the scattergram) was computed. Then, these values were plotted against each other. We may also observe in Figure 3.2.3 that most of the means fall between 0.17 and 0.21. This is almost constant, since *time* is accurate to the tenth of a second, and the range of *nu* in this system was from 0 to 56. In this example the

medians were almost always 0.2.

### 3.3. System Time Measurements

System time, as defined by the UNIX programmer's manual [18], is the time spent in the operating system while executing a user command. The accounting implemented in UNIX, however, allows a process to be charged not only for those system activities done on its behalf (like the opening of a file) but also for activities done on behalf of the community at large (like the servicing of a page fault interrupt). System time is load dependent.

Figures 3.3.1 and 3.3.2 present scattergrams of system time for two of our tasks in different systems. We have displayed *nu* as the workload estimator. There are two main differences between the cross section distributions in these scattergrams and those for user time. First, we clearly see that, towards higher values of our workload estimator (and this was observed for all three of them), the distributions 'shift' upwards. Second, the distributions themselves show a much larger degree of skewness towards the high end than those observed for user time.

The shift of the distributions is just the load dependency of system time. Even our inaccurate workload estimators account for this dependency. The biased shape of the distributions, however, appears to be the combined effect of our measurement errors and of the nature of the distribution. Not only does the precision of *time* play a role, but also the inaccuracy of our workload estimators. The ideal workload estimator in an interactive computer system is that which exactly characterizes all the activities and their interrelationships within the system. System time measured against an ideal estimator would behave as a function: i.e., the variance of the measurements would be zero. The problem then is to find suitable statistical distributions to fit our data. From them we may obtain estimates of our desired performance parameters.

Inspection of the system time distributions revealed that they could not be approximated by normal distributions. The observed asymmetries were non negligible. While some of the

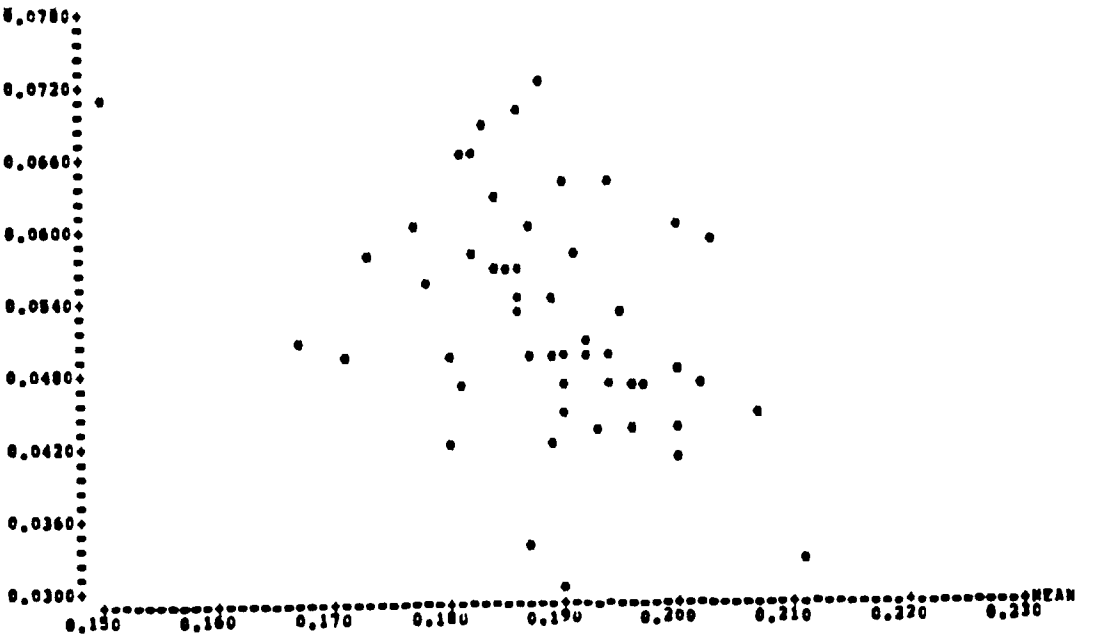


Figure 3.2.3: System P7A. Variance (VAR) versus mean (MEAN).
Task: Man man. Workload estimator: *nu*. Performance index: user time.

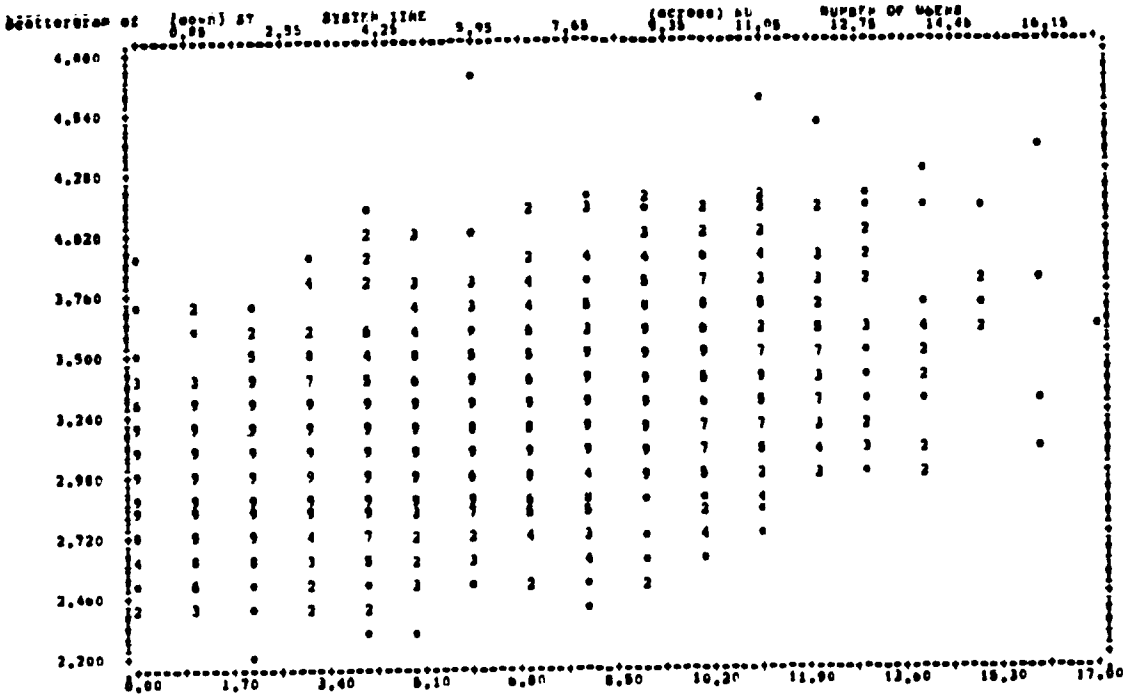

Figure 3.3.1: System VM. Scattergram of ST (system time) versus *nu*. Task: C compilation.

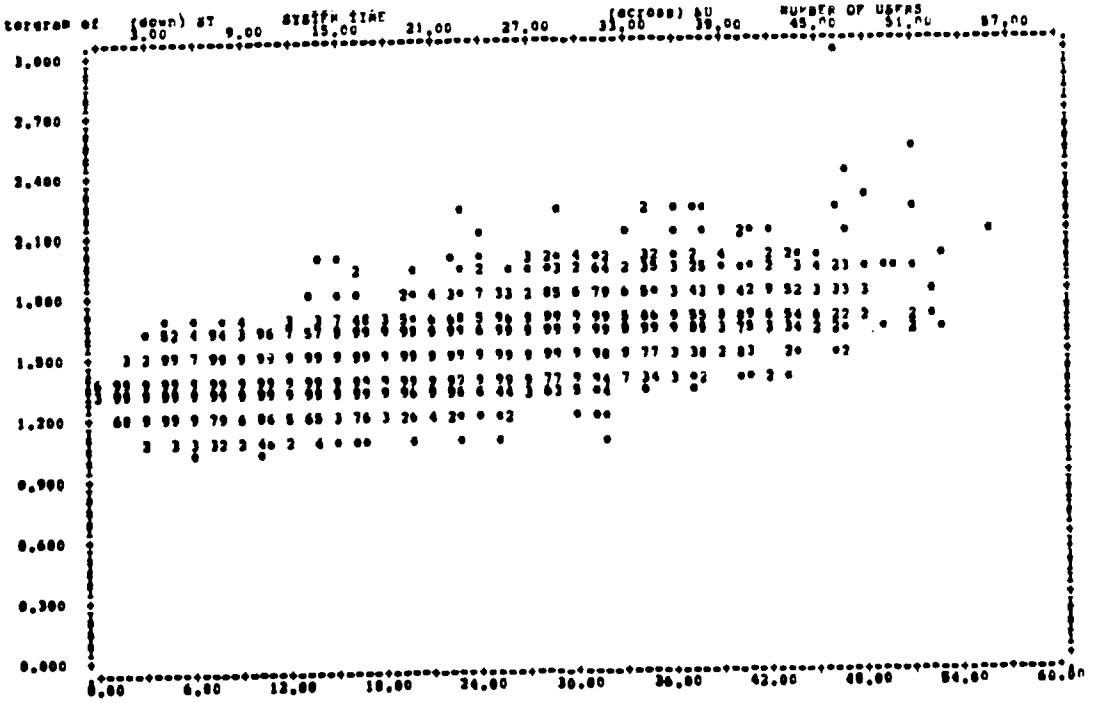

Figure 3.3.2: System P7A. Scattergram of ST (system time) versus *nu*. Task: Man man.

distributions looked like exponential, others showed a left-truncated bell shape with a long tail towards the high values of the observed variable. A family of distributions which could be used as suitable approximations were the Gamma distributions.

In contrast with Figure 3.2.3, the means and variances of system time distributions for each of our tasks in the different systems could be approximated very well through linear relationships. The 'slopes' of these relations varied, but in general the variances grew slowly (with coefficients between 0.05 and 0.45) as linear functions of the means. Figures 3.3.3 and 3.3.4 present the variance (VAR) plotted against the mean (MEAN) for the C compilation task in the system VM, and for the *man man* task in the system P7A, respectively. As in Section 3.2, *nu* is the underlying workload estimator. If in Figure 3.3.1 one does not take into account those distributions which have very few points, those above 13 users, in Figure 3.3.3 we are left with those values which have means less than 3.5. We see that the variance remains almost constant as a function of the mean. In fact, for this reduced range, the variance remains almost constant as a function of load as well.

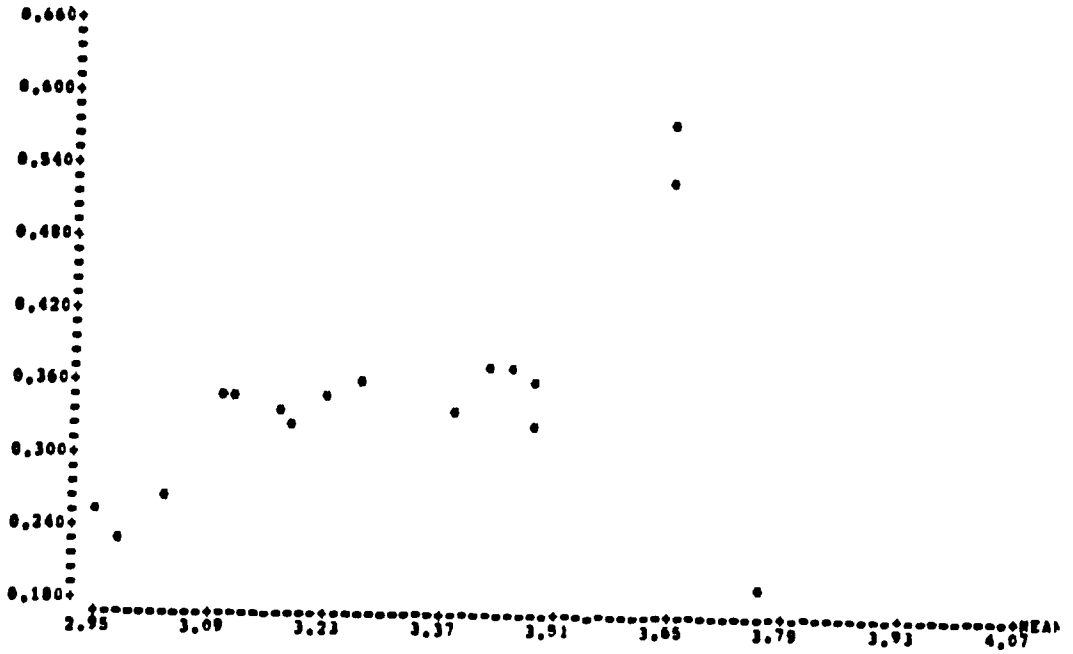


Figure 3.3.3: System VM. Variance (VAR) versus mean (MEAN).
Task: C compilation. Workload estimator: *nu*. Performance index: system time.

In Sections 3.2 and 3.3 we have chosen to display the same tasks and systems to allow a detailed analysis of these tasks by the reader. The inherently different behaviors displayed by user time and system time should be clear. In particular, Figures 3.2.3 and 3.3.4 should be contrasted to see the patterns regarding the relationships between variances and means.

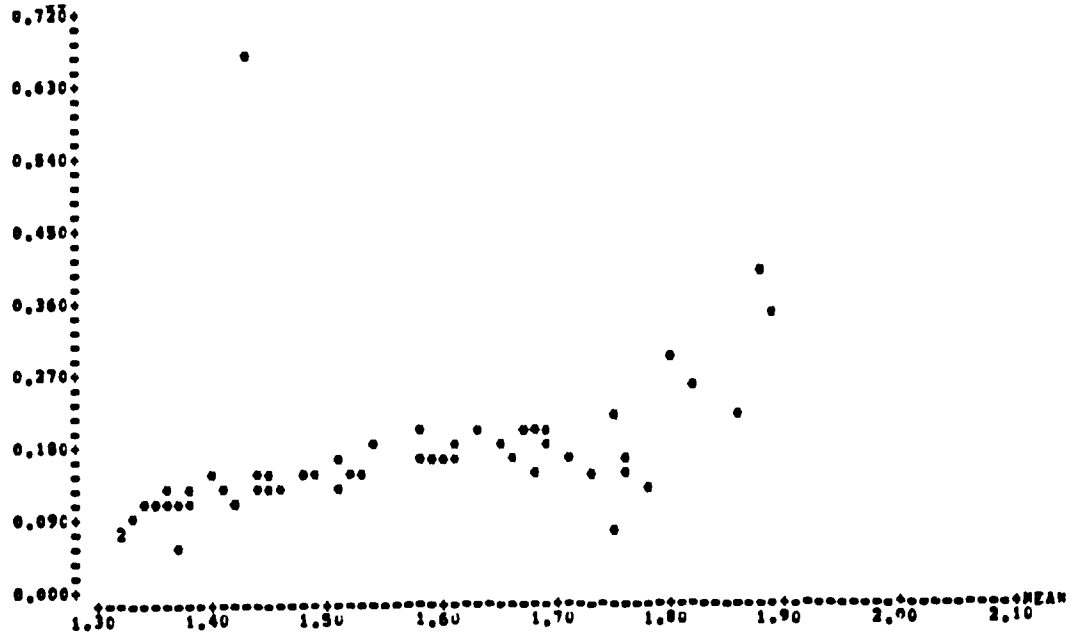


Figure 3.3.4: System P7A. Variance (VAR) versus mean (MEAN).
Task: Man man. Workload estimator: *nu*. Performance index: system time.

### 3.4. Response Time Measurements

Response time, as defined by the UNIX programmer's manual [18], is the total elapsed (wall clock) time spent by the system executing a command. As the events we were measuring never took less than 3 seconds to complete, with the exeption of *man man* in systems V50 and VE as explained in Section 2.3, see Table 3.4.1, and most of them always took more than 5 seconds, workload estimation inaccuracies had the potential of affecting more the response time

| System | Tasks | | |
|---|---|---|---|
| | C compilation [seconds] | CPU bound [seconds] | *man man* [seconds] |
| VI | 4 | 5 | 7 |
| P7A | 5 | 8 | 3 |
| V50 | 6 | 12 | 0 |
| VE | 4 | 6 | 0 |
| VM | 11 | 5 | 10 |
| P7B | 7 | 8 | 4 |
| VAA | 9 | 5 | 13 |

Table 3.4.1: Minimum response time observed.

measurements. Deviations from an ideal workload estimator produce distributions of measurements which have to be handled through statistical analysis. As it is clear that our estimators are fairly rough, what is of interest is to see whether they produce results which are amenable to statistical analysis.

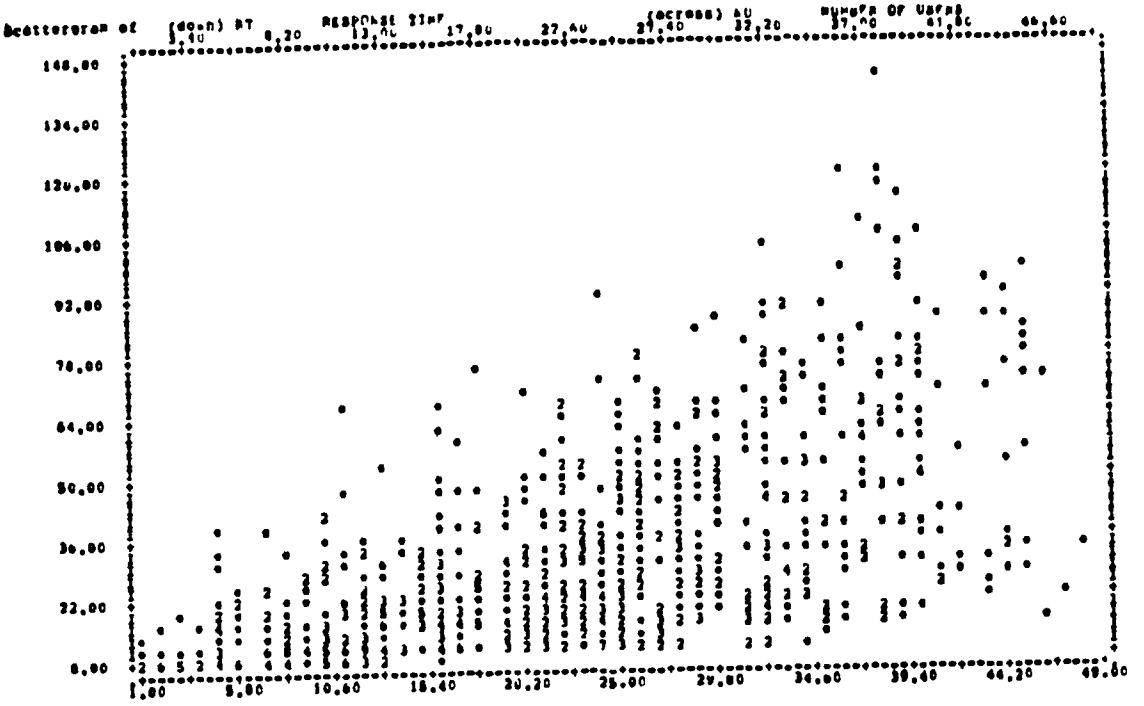

Figure 3.4.1: System P7B. Scattergram of RT (response time) versus *nu*. Task: CPU bound job.

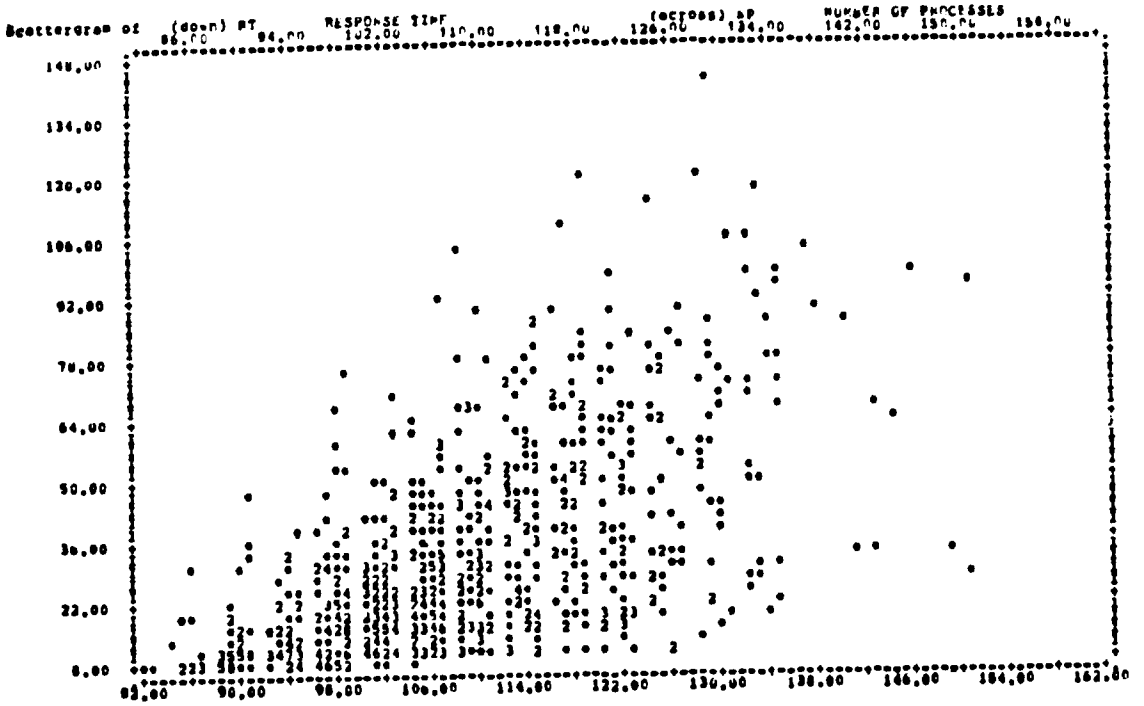

Figure 3.4.2: System P7B. Scattergram of RT (response time) versus *np*. Task: CPU bound job.

It should be remarked that the main advantages and interest of the chosen workload estimators arise from their easy definition, their portability, and the low overhead involved in their measurement. The problem is to find suitable statistical distributions which fit our data. From them we may obtain reliable estimates of our desired performance parameters.

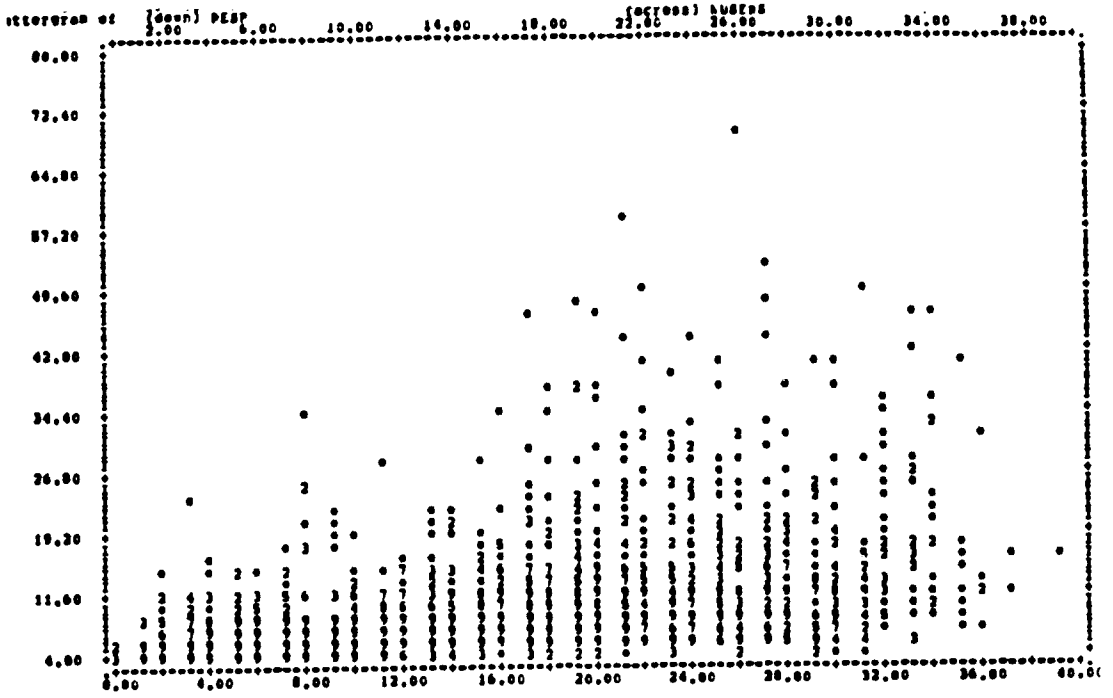

Figure 3.4.3: System VI. Scattergram of RT (response time) versus *nu* (NUSERS). Task: C compilation.

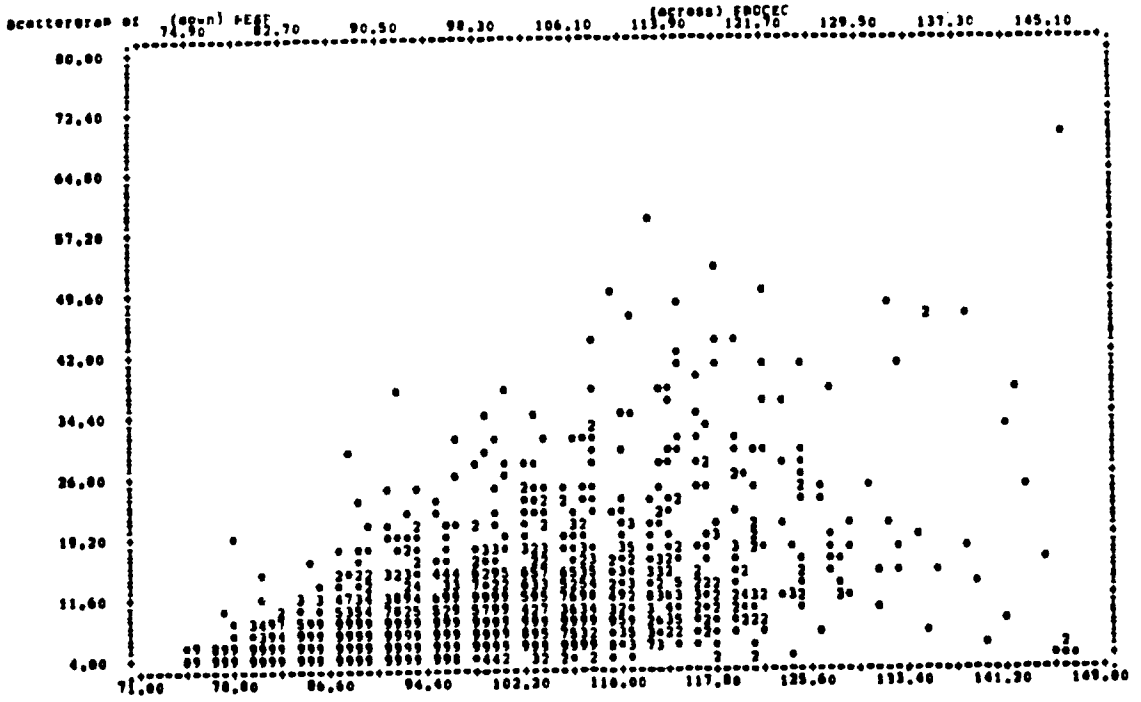

Figure 3.4.4: System VI. Scattergram of RT (response time) versus *np* (PROCEC). Task: C compilation.

In [6], where a preliminary version of this subsection and of Subsection 4.2.3 were presented, we displayed the scattergram of the *man man* task, in the System VI, as a function of each of our three (single variable) workload estimators. Here we have chosen to display scattergrams of the C compilation and CPU bound tasks, in the systems P7B and VI, respectively. This set of figures is representative of the response time behavior of all measured systems, with the sole exception of P7A.

In P7A, *np* produced scattergrams with some anomalies towards the higher values. This did not occur for *nu* or *nau*. One explanation is that new software which altered the number of processes continuously running was installed. A hardware change such as the addition of network ports could also have had this effect.

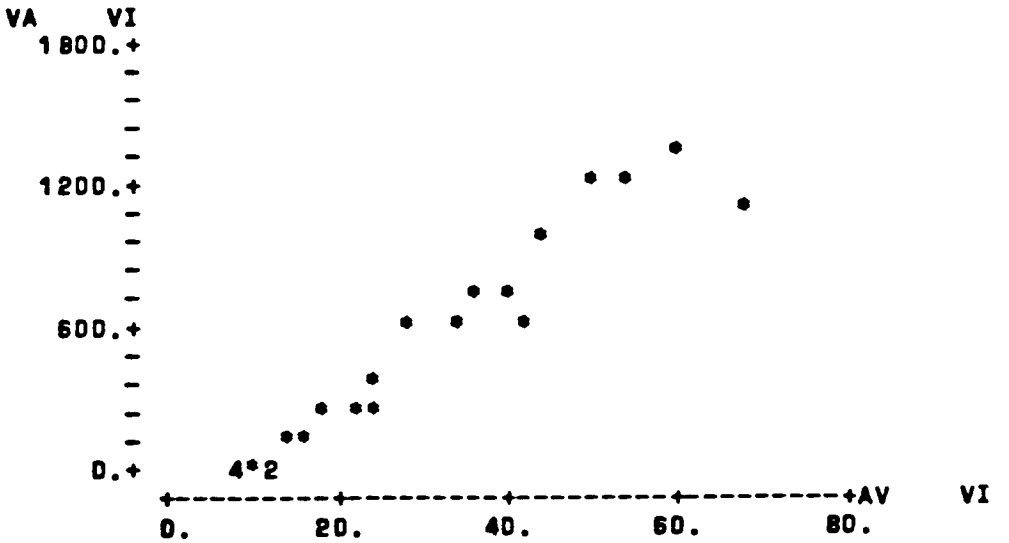


Figure 3.4.5: System VI. Variance (VA) versus mean (AV).
Task: CPU bound. Workload estimator: *nau*. Performance index: response time.

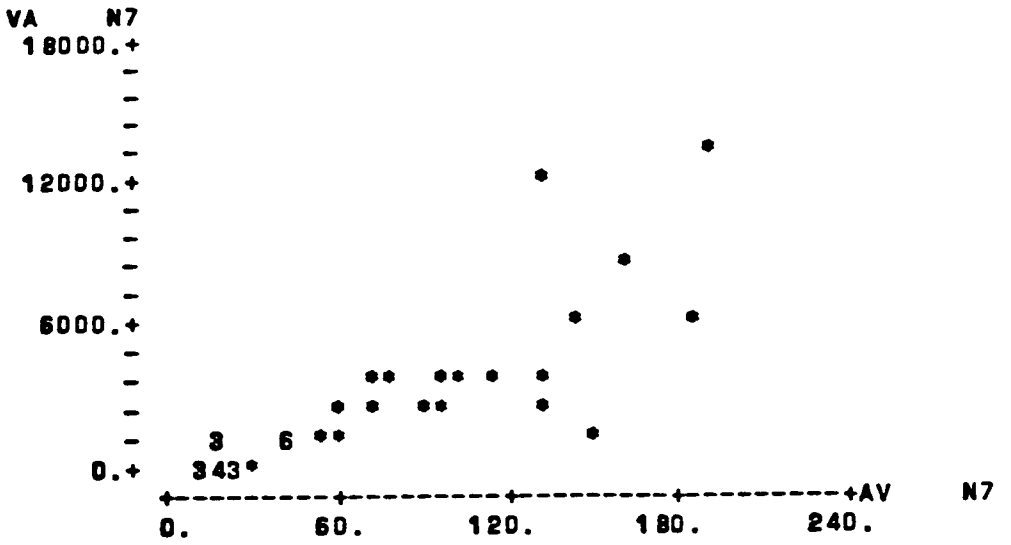


Figure 3.4.6: System P7A. Variance (VA) versus mean (AV).
Task: CPU bound. Workload estimator: *nau*. Performance index: response time.

In contrast with what has already been observed in user time and system time, the response time scattergrams always show that the distributions have much larger variances for higher values of the workload estimator. Never we observed a response time scattergram where the variances were constant or grew very little as a function of load. As with the system time samples, they have larger degrees of skewness towards the high values of load. We also observed in all systems that the minimum response time, per task and per system, is achieved throughout a substantial part of the range of each of the workload estimators. This points to the inaccuracy of our workload estimators. However, filtering out outliers from the scattergrams, we see that the maximum response time is a clear function of the workload estimator. Moreover, the behavior of the means of these distributions is best explained as an exponential function of the load [4,10].

As with system time, close inspection of response time histograms revealed that response time does not follow a normal distribution. As before, our measurements indicate that response time may always be accurately approximated by a Gamma distribution. This distribution offers the appropriate flexibility to adequately fit our data. One has only to characterize the distribution's parameters.

We then looked for possible relationships between the first and second moments of the response time distributions. We found that there were as strong linear relationships between them as those observed for system time. Moreover, we also found high correlation between them. Figures 3.4.5 and 3.4.6 depict the variance (VA) plotted against the mean (AV) for the systems VI and P7A (N7). The measurements were taken from the CPU bound task using *nau* as workload estimator. The associated regression analysis showed that, in system VI, 92% of the variation in the variance could be explained in terms of the variation in the mean. This was 68% for system P7A. Other tasks and systems exhibited the same behavior. Their correlation coefficients were always above 60%.

This empirical relationship provided the final reduction of complexity needed to appropriately fit a Generalized Linear Model [13].

## 4. Using GLIM to Model Five Systems

GLIM is an interactive package for modelling data through generalized linear models [2,13]. Its use allows fitting different parameter combinations to the data and obtaining the modelling results very efficiently. The theory behind the models is presented in Section 4.1. From the user's point of view, one defines a dependent variable, "yvar", which for us will correspond to user time, system time, and response time, and fits its expected value in terms of other observed variables. Whenever more than one such observed variable exists, as in our case (where we have *nu*, *np* and *nau*), linear combinations of them can also be fitted, as in ordinary regression analysis.

Moreover, GLIM permits choosing the underlying error distributions and has different modelling assumptions for each type of them. For example, normal errors are assumed to have the same variance. This is not the case for gamma errors, where the variance is assumed to be a fixed multiple of the mean. The analyses presented in Section 3 support these hypotheses.

The other degree of choice which GLIM offers is the type of relationship, called *link* by GLIM, existing between the expected value of yvar and the estimating variables. Possibilities include the identity relation, the inverse, the square root, and the logarithm, among others. If, for example, one chooses the logarithm relationship, then the values of the fitting model, **M**, and yvar are linked through the logarithm; i.e., the expected value of yvar, E(yvar), satisfies:

$$E(\text{yvar}) = e^{\mathbf{M}(\vec{x})}$$

where $\vec{x}$ is the vector of estimators.

As a criterion for evaluating alternative fitting models, we not only considered the statistical deviance given by GLIM, which is the maximized likelihood, but also looked at the sums of squares of the differences between our estimates and the observations. GLIM minimizes this sum of squares in the presence of normally distributed errors. Section 4.2 describes this study.

## 4.1. The Generalised Linear Models

Underlying the concept of a statistical model for a random variable is the idea that the variable under investigation has a definite structure which will explain the values actually obtained as well as predicting future values. The structure is in fact a description of the population and will be mirrored in whatever sample we obtain. This underlying idea postulates that the variable can be expressed in terms of other more basic variables: the components of the structure. If these latter variables have fixed (though possibly unknown) values, they are termed *systematic components*, whereas, if they too are random variables, they are termed *random components*.

### 4.1.1. Definition of a Generalised Linear Model

Even though one may consider any random variable $Y$, with $y_i$ denoting its $i$-th sample value, to be representable by any combination of any number of components, for most practical purposes simple structures suffice. A Generalized Linear Model (GLM) is determined as follows. Let a set of independent random variables $Y_i$ ($i = 1, \ldots, n$) have means $\mu_i$, so that

$$Y_i = \mu_i + \epsilon_i.$$

Then there are three basic properties which define a GLM.

#### 4.1.1.1. The Error Structure

The probability density function p of $Y_i$ is given by

$$p(Y_i) = e^{(Y_iQ[i]-b(Q[i]))/a_i(F)+c(Y_i,F)}$$

for suitable choices of $a_i$, b and c. (Note that F, termed the *scale* parameter, is constant for all $i$.) The mean and the variance of $Y_i$ can then be expressed in terms of $Q_i$ and F:

$$E(Y_i) = b'(Q_i)$$

$$var(Y_i) = b''(Q_i) * a_i(F)$$

where primes (') denote differentiation with respect to Q.

For example, the normal distribution is obtained by setting $a_i(Q) = Q$, $b(Q_i) = \frac{1}{2}*Q_i^2$ and $c(Y_i,Q) = -\frac{1}{2} * \log(2TQ) + \frac{Y_i^2}{F}$, where F would usually be denoted by $\sigma^2$.

It is convenient to write $b''(Q_i) = t_i^2$, the variance function, which is a function of $\mu_i$ only,

$$var(Y_i) = a_i(Q) * t_i^2 = \frac{Q*t_i^2}{w_i},$$

where the $w_i$ are called the prior weights, and the functions $a_i(Q)$ have the form $\frac{F}{w_i}$.

#### 4.1.1.2. The Linear Predictor

The role played by the remaining variables in the structure of each observation is expressed as a linear sum of their effects for the observation, called the *linear predictor*, $n_i$,

$$n_i = \sum_{j=1}^{p} x_{i,j} * b_j$$

where the $x_{i,j}$ are known and the $b_j$ are (usually unknown) parameters. The matrix X, of order 'n x p', is called the *design matrix*. The right hand side of the equation is called the *linear structure*. If an $x_{i,j}$ represents the presence or absence of a level of a factor, then $b_j$ is the *effect* of that level; if $x_{i,j}$ is the value of a quantitative covariable, then $b_j$ scales $x_{i,j}$ to give its effect on $n_i$.

#### 4.1.1.3. The Link Function

The relationship between the mean of the i-th observation and its linear predictor is given by the *link function* $g_i$:

$$n_i = g_i(u_i)$$

where the $g_i$ are assumed monotonic and differentiable. We define $h_i$, where

$$u_i = h_i(n_i),$$

as the inverse of the link function. Although each observation could in theory have a different link function, this is rare in practice and so the subscript is dropped.

In summary, a particular GLM can be identified by specifying the error distribution of the random component, the make-up of the linear predictor, and the function linking the means to the linear predictors. All error distributions must belong to the exponential family, which includes the exponential, normal, and gamma distributions among others.

### 4.2. Summary of the Analyses

In this section we present the results obtained using GLIM on the data of five systems. Presenting all of the twelve systems would have been too long and would not have added any new insight. We selectively present different tasks for user time, system time, and response time. We display seven linear models based on the three workload estimators *np*, *nu* and *nau*. Higher order models of the data were not explored because of the accuracy obtained with the linear models.

We have chosen to include all the information provided by GLIM so that one may independently judge the accuracy of the modelling effort, and, better yet, have the ability to produce graphic displays, as we do in Section 6, any of the modelled tasks in any of the systems. Indeed, if $L$ is the *link* function, and a, b, c, the parameters of the linear model, then:

$$L[\mathrm{E(yvar)}] = \%\mathrm{GM} + \alpha\mathrm{a} + \beta\mathrm{b} + \gamma\mathrm{c}$$

where $\alpha$, $\beta$, and $\gamma$ are the estimates given by GLIM for the corresponding parameters, and %GM the base modelling value.

#### 4.2.1. User Time

As in Section 3.2, we chose to present the results for the CPU bound job, since the processing of this task represents the CPU power of each machine. The GLIM hypotheses used were

| System | | Linear Estimators | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | *np* | *nu* | *nau* | *np,nu* | *np,nau* | *nu,nau* | *np,nu,nau* |
| VI | Normalized sum of sqr | .4097 | .4010 | .3965 | .3928 | .3940 | .3943 | .3916 |
| P7A | Normalized sum of sqr | .1066 | .9837E-1 | .9940E-1 | .9837E-1 | .9930E-1 | .9817E-1 | .9817E-1 |
| VM | Normalized sum of sqr | .3154E-1 | .3156E-1 | .3162E-1 | .3145E-1 | .3152E-1 | .3154E-1 | .3114E-1 |
| P7B | Normalized sum of sqr | .4947E-1 | .4746E-1 | .4748E-1 | .4746E-1 | .4783E-1 | .4743E-1 | .4742E-1 |
| P4A | Normalized sum of sqr | .3408E-1 | .3704E-1 | .3760E-1 | .3392E-1 | .3408E-1 | .3611E-1 | .3408E-1 |

Table 4.2.1.1: Normalized sums of squares for the different systems and linear estimators.
Task: CPU bound job. Yvar: ut. Error: normal. Link: identity.

normal errors and identity link. Both are supported by our data, as seen in Section 3.2.

Table 4.2.1.1 presents the normalized sums of squares for each linear estimator in each system. These quantities were obtained from the values given by GLIM by dividing the sum of squares (of the differences between the fitted points and the observed ones) by the degrees of freedom of the corresponding sample. In this way we take into account the size of the sample. For each system, a smaller table entry represents a better fit.

In Table 4.2.1.1 we see that the single variable estimators never fit better this data than the multiple variable ones, with only one exception. This exception is in system P4A, where *np* fits better than <*nu*,*nau*>. This behavior is probably due to the atypical characteristics that *nu* and *nau* had in P4A. In fact, Table 3.1.1 shows that their correlation was 23% lower than that observed for VM, and 31% lower than the one observed for P7B. We also have that, for VM and P4A, *np* is the best single estimator, while for P7A and P7B *nu* is best. For VI *nau* predicts better. This lack of pattern points to the coarseness of the estimators, as well as to the different workloads of the installations and to the user's habits.

| Parameters | S Y S T E M S | | | | | |
|---|---|---|---|---|---|---|
| | V I | | P 7 A | | V M | |
| | Estimate | Std. Error | Estimate | Std. Error | Estimate | Std. Error |
| %GM | 4.365 | .9366E-1 | 7.635 | .3796E-1 | 5.479 | .4323E-1 |
| *np* | .2221E-1 | .9793E-3 | .3938E-2 | .3048E-3 | .5053E-2 | .9253E-3 |
| scale | .4100 | | .1067 | | .3159E-1 | |
| %GM | 6.047 | .2064E-1 | 7.925 | .1092E-1 | 5.679 | .7963E-2 |
| *nu* | .3117E-1 | .1276E-2 | .1035E-1 | .4974E-3 | .6647E-2 | .1229E-2 |
| scale | .4014 | | .9845E-1 | | .3161E-1 | |
| %GM | 5.826 | .2778E-1 | 7.915 | .1175E-1 | 5.653 | .1273E-1 |
| *nau* | .5373E-1 | .2119E-2 | .1263E-1 | .6335E-3 | .8974E-2 | .1751E-2 |
| scale | .3966 | | .9948E-1 | | .3167E-1 | |
| %GM | 5.130 | .1119 | 7.893 | .3992E-1 | 5.556 | .5713E-1 |
| *np* | .1113E-1 | .1335E-2 | .3171E-3 | .3709E-3 | .2985E-2 | .1370E-2 |
| *nu* | .2097E-1 | .1759E-2 | .1002E-1 | .6301E-3 | .3721E-2 | .1819E-2 |
| scale | .3932 | | .9846E-1 | | .3152E-1 | |
| %GM | 5.294 | .1224 | 7.844 | .3924E-1 | 5.524 | .6225E-1 |
| *np* | .7228E-2 | .1619E-2 | .6930E-3 | .3660E-3 | .3621E-2 | .1707E-2 |
| *nau* | .4093E-1 | .3561E-2 | .1174E-1 | .7878E-3 | .3222E-2 | .3226E-2 |
| scale | .3944 | | .9939E-1 | | .3159E-1 | |
| %GM | 5.878 | .3051E-1 | 7.915 | .1168E-1 | 5.668 | .1498E-1 |
| *nu* | .1172E-1 | .2890E-2 | .7578E-2 | .1237E-2 | .4803E-2 | .2516E-2 |
| *nau* | .3614E-1 | .4827E-2 | .3835E-2 | .1568E-2 | .3008E-2 | .3582E-2 |
| scale | .3948 | | .9828E-1 | | .3161E-1 | |
| %GM | 5.294 | .1220 | 7.892 | .3989E-1 | 5.541 | .6285E-1 |
| *np* | .8027E-2 | .1624E-2 | .2261E-3 | .3726E-3 | .3562E-2 | .1706E-2 |
| *nu* | .1326E-1 | .2897E-2 | .7414E-2 | .1267E-2 | .4707E-2 | .2514E-2 |
| *nau* | .1961E-1 | .5858E-2 | .3736E-2 | .1576E-2 | -0.2531E-2 | .4453E-2 |
| scale | .3920 | | .9831E-1 | | .3154E-1 | |

Table 4.2.1.2: GLIM parameters for three systems and seven linear estimators.
Task: CPU bound job. Yvar: ut. Error: normal. Link: identity.

The pair which best approximates the measurements most of the time is <*np*,*nu*>. However, for P7A and P7B <*nu*,*nau*> is best. This split can be explained in terms of the workload of this PDP 11/70 installation, as its use for instructional purposes does make the user related estimators better predictors than the resource oriented ones. We shall see this again in Section 4.2.3.

In all cases using <*np*,*nu*,*nau*> produces better fits, but Tables 4.2.1.2 and 4.2.1.3 show that the standard error of the *nau* coefficient is not statistically meaningful in four of the five cases. In System VI, the exception, it is only marginally meaningful. From this we conclude that *nau* provides no additional prediction information in the presence of *nu* and *np*.

### 4.2.2. System Time

We have chosen to display the results of modelling *man man*. As in Section 3.3, we observed that the distributions had gamma shapes and the variances had a linear relationship with the means: we assumed that GLIM errors were gamma. We used log as the link function

| Parameters | S Y S T E M S | | | |
|---|---|---|---|---|
| | P 7 B | | P 4 A | |
| | Estimate | Std. Error | Estimate | Std. Error |
| %GM | 7.922 | .6944E-1 | 1.643 | .4440E-1 |
| *np* | .6842E-2 | .6364E-3 | .1712E-2 | .1378E-2 |
| scale | .4959E-1 | | .1004E-1 | |
| %GM | 8.464 | .1770E-1 | 1.695 | .1164E-1 |
| *nu* | .8734E-2 | .7001E-3 | .9147E-3 | .3305E-2 |
| scale | .4758E-1 | | .1009E-1 | |
| %GM | 8.447 | .1933E-1 | 1.675 | .2086E-1 |
| *nau* | .1094E-1 | .9001E-3 | .3943E-2 | .3612E-2 |
| scale | .4796E-1 | | .1005E-1 | |
| %GM | 8.461 | .1137 | 1.624 | .5114E-1 |
| *np* | .3137E-4 | .1308E-2 | .2589E-2 | .1837E-2 |
| *nu* | .8703E-2 | .1469E-2 | -0.3177E-2 | .4394E-2 |
| scale | .4763E-1 | | .1006E-1 | |
| %GM | 8.483 | .1257 | 1.647 | .5006E-1 |
| *np* | -0.4319E-3 | .1503E-2 | .1375E-2 | .2209E-2 |
| *nau* | .1151E-1 | .2162E-2 | .1131E-2 | .5787E-2 |
| scale | .4801E-1 | | .1007E-1 | |
| %GM | 8.458 | .1968E-1 | 1.673 | .2119E-1 |
| *nu* | .7006E-2 | .2636E-2 | -0.2363E-2 | .4301E-2 |
| *nau* | .2295E-2 | .3375E-2 | .5601E-2 | .4710E-2 |
| scale | .4761E-1 | | .1008E-1 | |
| %GM | 8.502 | .1254 | 1.631 | .5392E-1 |
| *np* | -0.5262E-3 | .1498E-2 | .2004E-2 | .2345E-2 |
| *nu* | .7028E-2 | .2638E-2 | -0.3666E-2 | .4566E-2 |
| *nau* | .2956E-2 | .3866E-2 | .2417E-2 | .6007E-2 |
| scale | .4766E-1 | | .1009E-1 | |

Table 4.2.1.3: GLIM parameters for two systems and seven linear estimators.
Task: CPU bound job. Yvar: ut. Error: normal. Link: identity.

because system time is load dependent. For higher values of overhead, the effect tends to behave in a multiplicative way akin to exponential growth.

In Table 4.2.2.1 we have both the statistical deviance and the sum of squares of the differences between the fitted points and the observed ones. The values given by GLIM have been divided by the degrees of freedom of the corresponding samples to take into account the size of the sample. We have named these quotients the normalized deviance and the normalized sum of squares. When GLIM errors are normal, these two coincide. We see that only twice did the two likelihood estimators not agree in selecting the best alternative. In VM for the single variable estimators, and in P7A for the tuple estimators.

As with user time, there was variability when choosing the best estimator, even though *nu* was favored among single variable ones. <*np*,*nu*> was best among the two variable ones. We also had that, for P7B and P4A, the best two-variable estimators matched the fit produced by the three-variable one. In Tables 4.2.2.2 and 4.2.2.3, we see that the coefficient for *nau* is statistically significant only for VI and P7A. This confirms our earlier observation that *nau* provides no additional information in the presence of *np* and *nu*. Moreover, the fact that *nau* is not statistically significant in the modelling of P7B and P4A, explains in part why the three-variable estimators did not outperform all of the two-variable ones.

| System | | Linear Estimators | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | *np* | *nu* | *nau* | *np,nu* | *np,nau* | *nu,nau* | *np,nu,nau* |
| VI | Normalized deviance | .8611E-1 | .8006E-1 | .8000E-1 | .7928E-1 | .7994E-1 | .7889E-1 | .7880E-1 |
| | Normalized sum of sqr | .1203 | .1137 | .1128 | .1124 | .1127 | .1119 | .1117 |
| P7A | Normalized deviance | .1298E-1 | .8622E-2 | .9270E-2 | .8616E-2 | .9261E-2 | .8578E-2 | .8566E-2 |
| | Normalized sum of sqr | .3055E-1 | .2003E-1 | .2144E-1 | .1988E-1 | .2144E-1 | .1993E-1 | .1990E-1 |
| VM | Normalized deviance | .2039 | .2038 | .2055 | .2015 | .2021 | .2036 | .2012 |
| | Normalized sum of sqr | 1.4383 | 1.4541 | 1.4657 | 1.4315 | 1.4397 | 1.4520 | 1.4287 |
| P7B | Normalized deviance | .1345E-1 | .1288E-1 | .1312E-1 | .1267E-1 | .1294E-1 | .1284E-1 | .1267E-1 |
| | Normalized sum of sqr | .5875E-1 | .5680E-1 | .5775E-1 | .5564E-1 | .5678E-1 | .5661E-1 | .5564E-1 |
| P4A | Normalized deviance | .4882E-1 | .5086E-2 | .5120E-1 | .4879E-2 | .4826E-2 | .5080E-2 | .4826E-2 |
| | Normalized sum of sqr | .3795E-1 | .3974E-1 | .3982E-1 | .3795E-1 | .3749E-1 | .3949E-1 | .3749E-1 |

Table 4.2.2.1: Normalized deviance and normalized sums of squares for the different systems and linear estimators. Task: man man. Yvar: st. Error: gamma. Link: log.

| Parameters | S Y S T E M S | | | | | |
|---|---|---|---|---|---|---|
| | V I | | P 7 A | | V M | |
| | Estimate | Std. Error | Estimate | Std. Error | Estimate | Std. Error |
| %GM | -1.033 | .4314E-1 | .1051 | .1262E-2 | .3773E-1 | .1904 |
| *np* | .1115E-1 | .4516E-3 | .2285E-2 | .1001E-3 | .2082E-1 | .2342E-2 |
| scale | .8616E-1 | | .1299E-1 | | .2042 | |
| %GM | -0.2122 | .9174E-2 | .2574 | .3154E-2 | .8505 | .2019E-1 |
| *nu* | .1719E-1 | .5701E-3 | .6961E-2 | .1445E-3 | .2941E-1 | .3117E-2 |
| scale | .8010E-1 | | .8629E-2 | | .2041 | |
| %GM | -0.3209 | .1245E-1 | .2534 | .3490E-2 | .7436 | .2335E-1 |
| *nau* | .2853E-1 | .9532E-2 | .8324E-2 | .1892E-3 | .3878E-1 | .4452E-2 |
| scale | .8006E-1 | | .9275E-2 | | .2058 | |
| %GM | -0.5014 | .5072E-1 | .2765 | .1098E-1 | .3354 | .1439 |
| *np* | .3517E-2 | .6052E-3 | -0.1816E-3 | .9985E-4 | .1244E-1 | .3451E-2 |
| *nu* | .1388E-1 | .7924E-3 | .7146E-2 | .1768E-3 | .1725E-1 | .4593E-2 |
| scale | .7934E-1 | | .8623E-2 | | .2020 | |
| %GM | -0.4063 | .5539E-1 | .2357 | .1118E-1 | . | . |
| *np* | .1161E-2 | .7319E-3 | .1685E-3 | .1017E-3 | .E- | .E- |
| *nau* | .2646E-1 | .1603E-2 | .8119E-2 | .2274E-3 | .E- | .E- |
| scale | .8002E-1 | | .9270E-2 | | . | |
| %GM | -0.2825 | .1366E-1 | .2525 | .3358E-2 | .8139 | .3798E-1 |
| *nu* | .8925E-2 | .1300E-2 | .5607E-2 | .3612E-3 | .2298E-1 | .6390E-2 |
| *nau* | .1517E-2 | .2175E-2 | .1866E-2 | .4561E-3 | .1041E-1 | .9090E-2 |
| scale | .7898E-1 | | .8586E-2 | | .2041 | |
| %GM | -0.4006 | .5499E-1 | .2751 | .1096E-1 | .2328 | .1582 |
| *np* | .1621E-2 | .7304E-3 | -0.2163E-3 | .9990E-4 | .1652E-1 | .4296E-2 |
| *nu* | .9192E-2 | .1306E-2 | .5773E-2 | .3687E-3 | .2486E-1 | .6355E-2 |
| *nau* | .1188E-1 | .2631E-2 | .1941E-2 | .4573E-3 | -0.1877E-1 | .1126E-1 |
| scale | .7889E-1 | | .8576E-2 | | .2018 | |

Table 4.2.2.2: GLIM parameters for three systems and seven linear estimators.
Task: man man. Yvar: st. Error: gamma. Link: log.

| Parameters | SYSTEMS | | | |
|---|---|---|---|---|
| | P7B | | P4A | |
| | Estimate | Std. Error | Estimate | Std. Error |
| %GM | -0.3028E-1 | .3585E-1 | .8841 | .3126E-1 |
| *np* | .6597E-2 | .3284E-3 | .4283E-2 | .9711E-3 |
| scale | .1349E-1 | | .4916E-2 | |
| %GM | .5066 | .9211E-2 | 1.002 | .8339E-2 |
| *nu* | .7810E-2 | .3641E-1 | .6213E-2 | .2371E-2 |
| scale | .1292E-1 | | .5122E-2 | |
| %GM | .4911 | .1009E-1 | .9890 | .1495E-1 |
| *nau* | .9822E-2 | .4694E-3 | .5757E-2 | .2591E-2 |
| scale | .1315E-1 | | .5155E-2 | |
| %GM | .2987 | .5764E-1 | .8781 | .3591E-1 |
| *np* | .2428E-2 | .6635E-3 | .4571E-2 | .1289E-2 |
| *nu* | .5379E-2 | .7519E-3 | -0.1042E-2 | .3083E-2 |
| scale | .1273E-1 | | .4931E-2 | |
| %GM | .2800 | .6447E-1 | .8536 | .3510E-1 |
| *np* | .2564E-2 | .7721E-3 | .6560E-2 | .1548E-2 |
| *nau* | .6428E-2 | .1118E-2 | -0.7623E- | .4032E-2 |
| scale | .1299E-1 | | .4874E-2 | |
| %GM | .4996 | .1020E-1 | .9929 | .1513E-1 |
| *nu* | .5701E-2 | .1365E-2 | .4795E-2 | .3088E-2 |
| *nau* | .2792E-2 | .1744E-2 | .2410E-2 | .3363E-2 |
| scale | .1289E-1 | | .5130E-2 | |
| %GM | .2922 | .6390E-1 | .8562 | .3771E-1 |
| *np* | .2518E-2 | .7647E-3 | .6462E-2 | .1640E-2 |
| *nu* | .5650E-2 | .1357E-2 | .5777E-3 | .3189E-2 |
| *nau* | -0.4796E-3 | .1997E-2 | -0.7831E-2 | .4195E-2 |
| scale | .1274E-2 | | .4890E-2 | |

Table 4.2.2.3: GLIM parameters for two systems and seven linear estimators.
Task: man man. Yvar: st. Error: gamma. Link: log.

### 4.2.3. Response Time

For this performance indicator we have chosen to present the modelling of the C compilation task, because of the relevance that the C programming language has in all UNIX systems [12,18].

Since in Section 3.4 we had observed that the distributions had gamma shapes, and that the variances had a high correlation with the means, we could assume that GLIM errors were gamma. It is well known that response time, as a function of workload estimators, behaves exponentially [4,10]. Thus, a logarithmic link function was used.

Table 4.2.3.1 displays the normalized deviance and the normalized sum of squares for response time. In contrast with what was observed for user time and system time, response time had clear best single and tuple estimators. With the sole exception of P7A, for all other systems the best single variable estimator was *np*. For P7A, it turned out to be either *nu* or *nau* depending on the likelihood estimator used. In fact, this was the only time where the likelihood estimators disagreed in the selection of the best single variable estimator. For the two variable cases, this occurred twice: for P7A and P4A. The best estimator pair was <*np*,*nu*>. Again, P7A was anomalous in this respect, in that both choices included *nau*. For both P4A and P7A, if one used

| System | | Linear Estimators | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | *np* | *nu* | *nau* | *np,nu* | *np,nau* | *nu,nau* | *np,nu,nau* |
| VI | Normalized deviance | .1641 | .1719 | .1914 | .1483 | .1587 | .1783 | .1484 |
| | Normalized sum of sqr | 24.47 | 25.66 | 26.79 | 22.22 | 23.26 | 26.68 | 22.22 |
| P7A | Normalized deviance | .4654 | .3237 | .3611 | .3167 | .3330 | .3135 | .3085 |
| | Normalized sum of sqr | 910.9 | 360.3 | 347.5 | 357.5 | 347.5 | 347.7 | 341.53 |
| VM | Normalized deviance | .1750 | .1875 | .1977 | .1609 | .1732 | .1854 | .1563 |
| | Normalized sum of sqr | 179.8 | 207.0 | 209.5 | 179.3 | 181.0 | 203.9 | 178.8 |
| P7B | Normalized deviance | .2408 | .2566 | .2887 | .2334 | .2410 | .2546 | .2157 |
| | Normalized sum of sqr | 491.6 | 540.9 | 574.9 | 490.6 | 490.9 | 537.1 | 447.79 |
| P4A | Normalized deviance | .1132 | .1463 | .1656 | .1091 | .1125 | .1402 | .1069 |
| | Normalized sum of sqr | 210.2 | 237.2 | 245.7 | 210.1 | 209.3 | 236.8 | 208.0 |

Table 4.2.3.1: Normalized deviance and normalized sums of squares for the different systems and linear estimators. Task: C compilation. Yvar: rt. Error: gamma. Link: log.

the normalized sum of squares, the tuple choice would be <*np*,*nau*>.

In P7A, the scattergrams for response time versus *np* show an anomalous behavior for the high values of the workload. This did not occur for *nu* or *nau*. In contrast with all the other scattergrams, the values of response time did not grow above those found half way through the range of processes. This system did not exhibit the same patterns under *np* as the others, which were akin to Figures 3.4.1 to 3.4.4. One explanation is that new software which altered the number of processes continuously running was installed. A hardware change such as the addition of network ports could also have had this effect. We did not record such a change.

In P7A *nau* does group the measurements much better. GLIM is correct when it gives *nau* as a better fit than *np*. The failure of *np* to be the best single estimator may also indicate, in addition to what we said in Section 3.4, that the users of this system behaved differently than those of other systems. P7A was mostly used for instructional purposes. Since many users may be concurrently executing the same code, such as a toy operating system for example, severe distortions from the *np* estimator point of view may exist. For example, the memory utilization of several users executing the same shared piece of code is much less than it would be if each had his own copy. Thus the load on the resources of the machine is less than that suggested by the number of processes. This load is clearly more faithfully represented by the number of active users.

| Parameters | SYSTEMS | | | | | |
|---|---|---|---|---|---|---|
| | VI | | P7A | | VM | |
| | Estimate | Std. Error | Estimate | Std. Error | Estimate | Std. Error |
| %GM | -0.8103 | .5845E-1 | -1.0100 | .7959E-1 | .5254 | .1008 |
| *np* | .3029E-1 | .6104E-3 | .3135E-1 | .6388E-3 | .5679E-1 | .2155E-2 |
| scale | .1642 | | .4656 | | .1750 | |
| %GM | 1.533 | .1329E-1 | 1.736 | .1980E-1 | 2.748 | .1933E-1 |
| *nu* | .3962E-1 | .8237E-3 | .5547E-1 | .8986E-3 | .8015E-1 | .2968E-2 |
| scale | .1719 | | .3237 | | .1875 | |
| %GM | .1721 | .1150E-1 | 1.747 | .2234E-1 | 2.467 | .3164E-1 |
| *nau* | .5028 | .1213E-1 | .6462E-1 | .1200E-2 | .1047 | .4341E-2 |
| scale | .1914 | | .3611 | | .1977 | |
| %GM | -0.3140E-2 | .6737E-11 | .1430 | .7197E-1 | 1.1040 | .1282 |
| *np* | .1865E-2 | .8011E-3 | .5721E-2 | .6688E-3 | .3961E-1 | .3075E-2 |
| *nu* | .2136E-1 | .1057E-2 | .4937E-1 | .1128E-2 | .4032E-1 | .4090E-2 |
| scale | .1484 | | .3166 | | .1609 | |
| %GM | -0.2587 | .7452E-1 | .6764 | .7219E-1 | .8455 | .1446 |
| *np* | .2310E-1 | .8622E-3 | .1001E-1 | .6731E-3 | .4586E-1 | .3966E-2 |
| *nau* | .1800 | .1587E-1 | .5380E-1 | .1435E-2 | .2758E-1 | .7516E-2 |
| scale | .1587 | | .3330 | | .1732 | |
| %GM | 1.544 | .1319E-1 | 1.666 | .2082E-1 | 2.628 | .3608E-1 |
| *nu* | .2973E-1 | .1349E-2 | .3582E-1 | .2208E-2 | .5818E-1 | .6053E-2 |
| *nau* | .1696 | .1882E-1 | .2672E-1 | .2791E-2 | .3467E-1 | .8623E-2 |
| scale | .1679 | | .3135 | | .1855 | |
| %GM | .1097E-1 | .7326E-1 | 1.174 | .7104E-1 | .7469 | .1389 |
| *np* | .1849E-1 | .8680E-3 | .4811E-2 | .6636E-3 | .5351E-1 | .3770E-2 |
| *nu* | .2098E-1 | .1320E-2 | .3161E-1 | .2243E-2 | .6612E-1 | .5560E-2 |
| *nau* | .9277E-2 | .1917E-1 | .2527E-1 | .2783E-2 | -0.6310E-1 | .9850E-2 |
| scale | .1484 | | .3085 | | .1564 | |

Table 4.2.3.2: GLIM parameters for three systems and seven linear estimators.
Task: C compilation. Yvar: rt. Error: gamma. Link: log.

Tables 4.2.3.2 and 4.2.3.3 show all the parameters given by GLIM in the analysis of the C compilation task. In Section 4.1 we saw how to obtain analytic formulae to represent the expected value of response time in terms of these parameters. Section 6 will present some of these surfaces.

It is interesting to notice, in Tables 4.2.3.1, 4.2.3.2 and 4.2.3.3, that in VI the addition of *nau* to the estimator <*np*,*nu*> does not increase the accuracy of the model. In fact, the deviance got slightly worse. We also see that *nau* had a statistically significant coefficient in <*np*,*nu*,*nau*> only in P7A. These observations lead us to conclude again that for most systems *nau* does not add significant modelling information in the presence of *np* and *nu*.

In the case of a long data gathering period, after an exploratory amount of measurements, one could assess whether *nau* is indeed significant for the systems, and, if not, omit gathering statistics about it. This will reduce the overhead of the metering effort and its total cost, while preserving modelling accuracy. However, our evidence does indicate that, for the types of workloads we measured, *nau* will provide no additional information in the presence of *nu* and *np*.

| Parameters | SYSTEMS | | | |
|---|---|---|---|---|
| | P 7 B | | P 4 A | |
| | Estimate | Std. Error | Estimate | Std. Error |
| %GM | -1.229 | .1529 | .6347 | .1491 |
| *np* | .3951E-1 | .1401E-2 | .7122E-1 | .4629E-2 |
| scale | .2408 | | .1133 | |
| %GM | 2.060 | .4108E-1 | 2.4650 | .4433E-1 |
| *nu* | .4390E-1 | .1626E-2 | .1509 | .1258E-1 |
| scale | .2566 | | .1463 | |
| %GM | 2.030 | .4724E-1 | 2.1620 | .8466E-1 |
| *nau* | .5319E-1 | .2201E-2 | .1395 | .1466E-1 |
| scale | .2888 | | .1657 | |
| %GM | -0.1511 | .2514 | .9139 | .1685 |
| *np* | .2589E-1 | .2892E-2 | .5752E-1 | .6052E-2 |
| *nu* | .1728E-1 | .3250E-2 | .5109E-1 | .1448E-1 |
| scale | .2334 | | .1091 | |
| %GM | -1.381 | .2799 | .5103 | .1673 |
| *np* | .4150E-1 | .3342E-2 | .8106E-1 | .7384E-2 |
| *nau-* | .321E-2 | .4794E-2 | -0.3440E-1 | .1934E-1 |
| scale | .2410 | | .1125 | |
| %GM | 2.112 | .4508E-1 | 2.212 | .7903E-1 |
| *nu* | .5862E-1 | .5897E-2 | .1140 | .1604E-1 |
| *nau* | -0.1980E-1 | .7526E-2 | .6519E-1 | .1757E-1 |
| scale | .2546 | | .1402 | |
| %GM | -0.9839 | .2656 | .7749 | .1755 |
| *np* | .3762E-1 | .3166E-2 | .7059E-1 | .7633E-2 |
| *nu* | .5266E-1 | .5435E-2 | .6126E-1 | .1486E-1 |
| *nau* | -0.6370E-1 | .7889E-2 | -0.5600E-1 | .1955E-1 |
| scale | .2157 | | .1068 | |

Table 4.2.3.3: GLIM parameters for two systems and seven linear estimators.
Task: C compilation. Yvar: rt. Error: gamma. Link: log.

## 5. Robustness of the Models

What duration should the data gathering period have? Or, how many sample points should one take? In this section we present a robustness analysis of our GLIM modelling which sheds light on this question. We do it for response time, the performance index which we consider most important at the user level.

We present the parameters given by GLIM in three systems when random subsamples of different sizes of the same set of measurements were analyzed. We considered the same task as in Section 4.2.3: the C compilation. As in Section 3.1.1, these subsamples were made by sequentially scanning the file containing all sample points, and for each measurement a 'random' decision, with a fixed probability for membership, was made about whether to incorporate or not that measurement into the subsample. We created three files with membership probabilities of 0.5, 0.25 and 0.125, to verify the robustness of the modelling technique.

In Tables 5.1.1, 5.1.2 and 5.1.3 we display the parameters given by GLIM when presented with the random subsamples of different sizes. From the appropriate entries in those tables, we

conclude that the coefficients found in the tables of Section 4 are very robust. In almost all of the cases (the only exception being the 170-point subsample in System VM), the estimators for the smaller samples were within their standard error from those found in the larger samples, even though we were cutting the sample sizes in half.

The price paid for fewer data points, as expected, was much larger standard errors of the fitting estimators. Given that each smaller sample was roughly half of the larger one, we see that standard errors diminished approximately 25% when the size of the sample doubled. This rule can be a guide in deciding the duration of the data gathering period as a function of the desired accuracy.

We also observed that, for the best linear estimators in each system, those found from samples which had more than 300 points showed high levels of significance, i.e., the standard errors of the estimators were small. This gives us an empirical global bound on the minimum number of data points one should gather in a system.

## 6. Characteristic Surfaces of the Installations

In this section we have chosen to display the surfaces obtained from modelling with GLIM two tasks in three systems. We call these the "characteristic surfaces" of the installations, because of the stationary behavior of the distributions of performance indices found in our analyses. These plots were obtained using the UNIGRAFIX [19] graphics system.

| | Sample Size 1497 | | Sample Size 729 | | Sample Size 376 | |
|---|---|---|---|---|---|---|
| Parameter | Estimator | Std. Error | Estimator | Std. Error | Estimator | Std. Error |
| %GM | .5640 | .1021 | .6707 | .1404 | .6130 | .1906 |
| *np* | .1120E-1 | .9473E-3 | .9783E-2 | .1304E-2 | .1069E-1 | .1755E-2 |
| *nau* | .5146E-1 | .2020E-2 | .5608E-1 | .2785E-2 | .5173E-1 | .4124E-2 |
| scale | .3393 | | .3146 | | .3395 | |

Table 5.1.1: System P7B. Robustness analysis of <*np*,*nau*>.
Task C compilation. Yvar: rt. Error: gamma. Link: log.

| | Sample Size 1722 | | Sample Size 851 | | Sample Size 428 | |
|---|---|---|---|---|---|---|
| Parameter | Estimator | Std. Error | Estimator | Std. Error | Estimator | Std. Error |
| %GM | -0.1275 | .9481E-2 | -0.6774E-2 | .1405 | .6659E-1 | .1886 |
| *np* | .2020E-1 | .1128E-2 | .1893E-1 | .1652E-2 | .1765E-1 | .2264E-2 |
| *nu* | .2008E-1 | .1502E-2 | .2032E-1 | .2060E-2 | .2341E-1 | .2955E-2 |
| scale | .1506 | | .1422 | | .1309 | |

Table 5.1.2: System VI. Robustness analysis of <*np*,*nu*>.
Task C compilation. Yvar: rt. Error: gamma. Link: log.

| | Sample Size 739 | | Sample Size 365 | | Sample Size 170 | |
|---|---|---|---|---|---|---|
| Parameter | Estimator | Std. Error | Estimator | Std. Error | Estimator | Std. Error |
| %GM | 1.348 | .1828 | 1.428 | .2424 | -0.5394E-1 | .3657 |
| *np* | .3401E-1 | .4387E-1 | .3209E-1 | .5783E-2 | .6752E-1 | .8716E-2 |
| *nu* | .4529E-1 | .5835E-2 | .4458E-1 | .7294E-2 | -0.1081E-2 | .1062E-1 |
| scale | .1742 | | .1553 | | .1186 | |

Table 5.1.3: System VM. Robustness analysis of <*np*,*nu*>.
Task: C compilation. Yvar: rt. Error: gamma. Link: log.

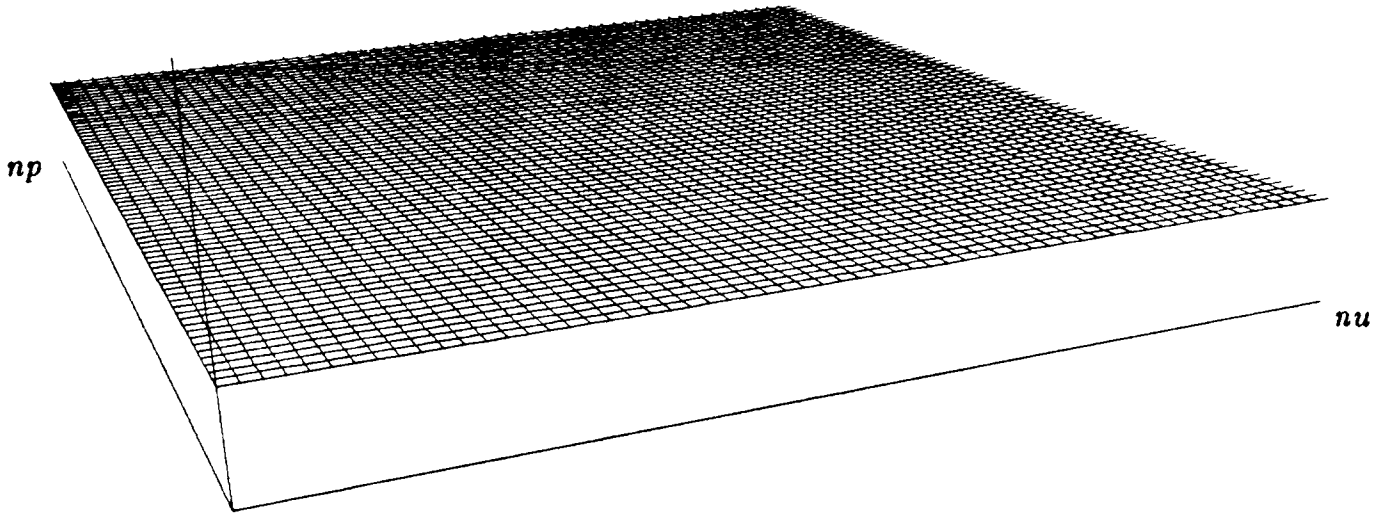


**Figure 6.1: Task**: CPU bound. Performance Index: User Time.
System: VM. Estimator: $<nu,np>$.

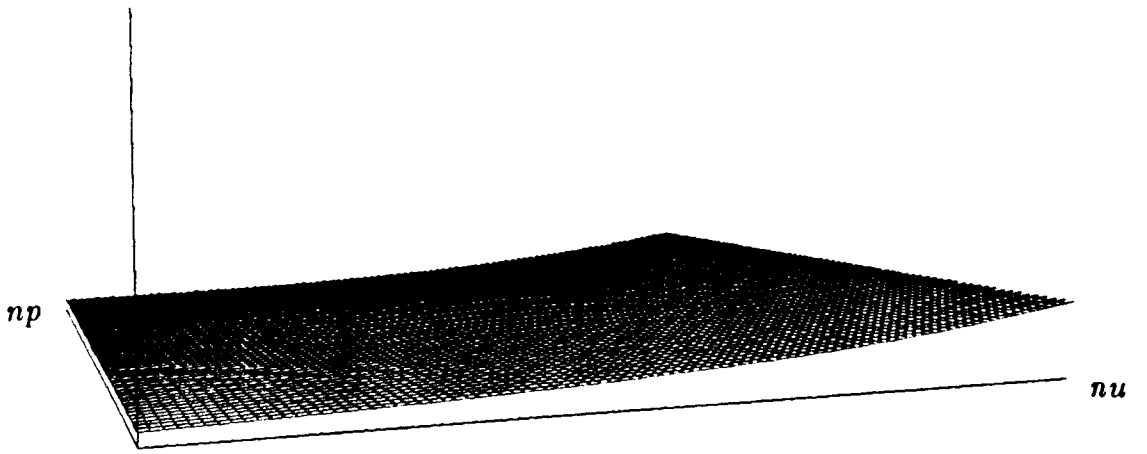


**Figure 6.2: Task**: C compilation. Performance Index: Response Time.
System: VI. Estimator: $<nu,np>$.

These surfaces (or the analytic expressions which define them) may be used to assess, from the user's point of view (or from a task scheduler's point of view in a set of cooperating machines), which of several systems a job should be executed in. From this viewpoint, our methodology is an alternative for load balancing. The gap that needs to be closed is that of relating the

resource requirements of an arbitrary task to those of a predefined set of benchmarks for which measurements have been made.

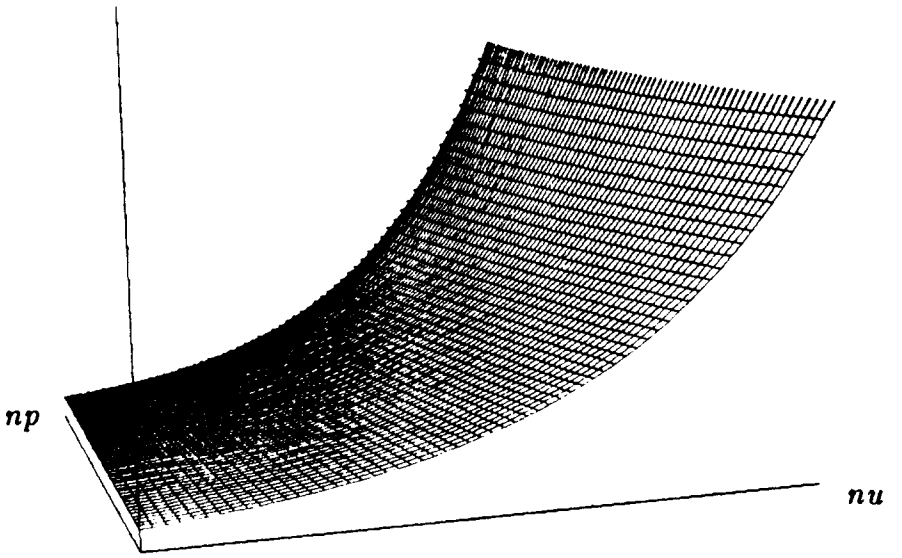


Figure 6.3: Task: C compilation. Performance Index: Response Time.
System: P7A. Estimator: <*nu*,*np*>.

## 7. Conclusions

We have found that each of the three performance indices we observed as functions of our workload estimators presented a statistical behavior such that the utilization of generalized linear models, GLMs, was appropriate and called for. GLMs were used to model the observed distributions of our performance indices and to predict their values in regions where we had no data. We call these models the characteristic surfaces of an installation. The behavior of our indices, in turn, strongly suggests that the utilization of rather imprecise workload estimators, such as the number of processes (*np*), the number of users (*nu*), and the number of active users (*nau*), is adequate as a basis for estimating selected performance indices in a computer installation. All installations were monitored while processing their natural workloads. The experiments did not require the systems to be shut down.

When measured against our three workload estimators (*np*, *nu*, and *nau*), response time and system time appear to have gamma distributed values. User time has normally distributed values. User time values are independent of load. This is not so for system time and response time. The variances of user time distributions appear to be constant. Those of system time distributions grew slightly as a function of load. The ones of response time grew ostensibly as a function of load. What is more, we have observed that the means and variances of these distributions are linearly related as functions of each of our workload estimators. In the case of response time, they are highly correlated as well. For response time, the ratio between the first and second moments of the distributions is almost directly proportional to the values of the workload estimator. This behavior is essential when using a GLM with Gamma error. All of these observations give us a better understanding of these indices, in particular of response time, in installations running their natural workload.

The models obtained using GLIM proved to be very robust. We have seen that 300 measurement points per installation appear to be necessary for obtaining minimum accuracy. This bound is hardware configuration dependent, because in UNIX the range of *np* depends, for example, on the number of terminals. Doubling the number of measurements reduced by 25% the standard error of the parameters estimated by the model. With these rules we may assess the cost of achieving a predetermined error in a performance study. The robustness of the models allows us to compare different computer installations using the terminal probe method as a data gathering technique. While the systems are executing their natural workloads, we may find their characteristic surfaces and use them for comparisons. The observed stability of user behavior between hardware changes justifies this approach. These results also lead us to believe that our methodology is an alternative for dynamic task assignment and load balancing in a distributed computing environment.

When analyzing in pairs the relationships that existed between our workload estimators, we observed very few outliers. We also saw high correlation values among them. This points to some degree of redundancy. The pair <*nu*,*np*> was the least correlated. When modelling our three performance indices (user time, system time, and response time), we also observed that, for most systems and tasks, *nau* adds no significant additional information to that given by *np* and *nu*. These facts can be used to reduce the data collection cost and the overhead of the terminal probe method by not gathering superfluous information. This can be done with essentially no sacrifice in modelling accuracy.

In synthesis, we have presented results which show that easy to measure portable workload estimators can serve as a basis for performance comparison studies among different computer installations, and for load balancing strategies. These studies can be done without stand-alone experimentation. We have also given bounds on the length of the data gathering period. Our procedures are based on the terminal probe method implemented in UNIX through scripts.

**Acknowledgements**

We would like to thank the user communities for their patience and assistance gotten both at Berkeley and at Santiago while gathering data and processing it. In particular, Luis Disset, Javier Pinto, and Stefano Zatti helped us with some of the data analyses.

## 8. Bibliography

[1] Babaoglu, O., Joy, W., and Porcar, J., "Design and Implementation of the Berkeley Virtual Memory Extension to the UNIX Operating System," Department of EECS--Computer Science Division, University of California, Berkeley, 1979.

[2] Baker, R. J., and Nelder, J. A., "The GLIM System Manual, Release 3," Numerical Algorithms Group, NAG Central Office, Mayfield House, 256 Banbury Road, Oxford OX2 7DE, 1978.

[3] Bourne, S. R., "The UNIX Shell," The Bell System Technical J. 57, 6 Part 2 (July-August 1978), 1971-1990.

[4] Cabrera, L. F., "A Performance Analysis Study of UNIX," Proceedings of the Computer Performance Evaluation Users Group 16th Meeting, CPEUG 80, NBS Special Publication 500-65, Orlando, Florida, October 1980, pp. 233-243.

[5] Cabrera, L. F., "Benchmarking UNIX: A Comparative Study," in Experimental Computer Performance Evaluation (D. Ferrari and M. Spadoni eds.) North-Holland, Amsterdam, Netherlands, 1981, pp 205-215.

[6] Cabrera, L. F., "The Terminal Probe Method Revisited. Some Statistical Considerations", Proceedings of the Computer Performance Evaluation Users Group 19th Meeting, CPEUG 83, NBS Special Publication 500-103, San Francisco, California, October 1983, pp. 197-214.

[7] Digital Equipment Corporation, VAX 11/780 Technical Summary. Maynard, Mass., 1983.

[8] Fateman, R. J., "Addendum to the Mathlab/MIT MACSYMA Reference Manual for VAX/UNIX VAXIMA," Department of EECS–Computer Science Division, University of California, Berkeley (Dec. 1979).

[9] Ferrari, D. "Computer Systems Performance Evaluation," Prentice-Hall, Englewood Cliffs, NJ, 1978.

[10] Ferrari, D., Serazzi, G., and Zeigner, A., "Measurement and Tuning of Computer Systems," Prentice-Hall, Englewood Cliffs, NJ, 1983.

[11] Karush, A. D., "The Benchmarking Method Applied to Time-Sharing Systems," Rept. SP-3347, System's Development Corporation, Santa Monica, CA, August 1969.

[12] Kernighan, B. W. and Mashey, J. R., "The UNIX Programming Environment," Computer 14, 4 (Apr. 1981), 12-24.

[13] Nelder, J. A. and Wedderburn R.W.M., "Generalized Linear Models," Journal of the Royal Statistical Society, A, 135, pp 370-384, 1972.

[14] Nie, N. H. et al, "SPSS, Statistical Package for the Social Sciences," Second Edition, McGraw-Hill, 1975.

[15] Ritchie, D. M. and Thompson, K. L., "The UNIX Time-Sharing System," Comm. ACM, Volume 17, 7 (July 1974), 365-375. A revised version appeared in The Bell System Technical J. 57, 6 Part 2 (July-Aug. 1978), 1295-1990.

[16] Ritchie, D. M., Johnson, S. C., Lesk, M. E., and Kernighan B. W., "The C Programming Language," The Bell System Technical J. 57, 6 Part 2 (July-Aug. 1978), 1991-2019.

[17] Ryan, T.A., Joiner, B. L. and Ryan B. F., "MINITAB Reference Manual," Minitab Project, Statistics Department, 215 Pond Laboratory, Pennsylvania State University, University Park, PA 16802, November 1982.

[18] "UNIX Programmer's Manual," 4.2 Berkeley Software Distribution, Virtual VAX-11 Version, August 1983. Computer Science Division, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

[19] Sequin, C.H., Segal, M. and Wensley, P., "UNIGRAFIX 2.0 User's Manual and Tutorial", Report No. UCB/CSD 83/161, December 1983, University of California, Berkeley, Berkeley CA 94720.