M. Schatzoff
C. C. Tillman

Design of Experiments in Simulator Validation

Abstract: A common problem encountered in computer system simulation is that of validating that the simulator can produce, with a reasonable degree of accuracy, the same information that can be obtained from the modeled system. This is basically a statistical problem because there are usually limitations with respect to the number of controlled tests that can be carried out, and assessment of the fidelity of the model is a function of the signal to noise ratio. That is, the magnitude of error which can be tolerated depends upon the size of the effect to be predicted. In this paper we describe, by example, how techniques of statistical design and analysis of experiments have been used to validate the modeling of the dispatching algorithm of a time sharing system. The examples are based on a detailed, trace-driven simulator of CP-67. They show that identical factorial experiments involving parameters of this algorithm, when carried out on both the simulator and on the actual system, produced statistically comparable effects.

Introduction

A common problem encountered in computer system simulation is that of validating that the simulator can predict, to some required degree of accuracy, the behavior of the modeled system. Typically, this problem is approached in a direct and obvious manner, namely, by observing the behavior of the modeled system under a set of controlled or measurable loading and other conditions (e.g., configuration variations), then comparing the observations to corresponding predictions of the simulator. When the observations and predictions agree within required limits for all conditions treated, the simulator is considered to be validated.

The procedure described above has several recognized difficulties, not the least of which is the uncertainty associated with drawing a general conclusion from a finite (and typically small) number of experiments. This uncertainty is of particular concern in the validation of detailed simulators, where one may be attempting to predict effects which are of the same order of magnitude as the random fluctuations or "noise" inherent in real system measurements. In such cases, it is advantageous to use techniques for statistical design and analysis of experiments, both to reduce the number of experiments needed for a given level of confidence and to indicate the statistical significance of measured and simulated effects. In this paper, we illustrate this point by describing the use of statistical techniques in validating the modeling of the dispatching algorithm of a paged time-sharing system. Our discussion is based, in particular, on a detailed, trace-driven simulator [1] of the CP-67 system [2]. Because this simulator was developed specifically for investigating new system decision algorithms such as those used in dispatching and paging, it was important to determine whether changes to such algorithms would produce comparable effects in the simulator and in the real system. Such a determination is significant with regard not only to the CP-67 system, but also to the very similar and more current VM/370 system [3]. We feel, in fact, that while modification of the present model to simulate VM/370 would be straightforward, conclusions about alternative decision algorithms reached using the present CP-67 model are generally applicable to VM/370 as well.

Our basic approach in this work has been to view both the real system and the simulator as "black boxes" with certain inputs and outputs. Our inputs in this case, aside from the job streams themselves, were essentially switches for disabling or modifying different parts of the dispatching mechanism. For example, a provision for preempting users after a given amount of paging activity could be turned on or off. Outputs of interest were typical performance indicators such as paging rate and CPU utilization in various states based on a fixed interval of time. In all experiments, user job streams and hardware configuration were held fixed. Thus our procedure was to run identical experiments on the real system and on the simulator, selecting various combinations of switch settings in accordance with the techniques of statistical design of experiments. The outputs from these experiments were then analyzed to estimate the statistically significant effects for each "black box," i.e., to extract the signals from the noise, and to verify that the same signals were present in both cases. In addition, a set of "sensitivity" experiments was carried out on the

simulator alone to indicate its robustness with respect to several timing parameters representing CP-67 overhead for various functions and disk hardware characteristics.

In the next section we present an overview of the simulator per se. This is followed with a review of some of the important concepts underlying the statistical design and analysis of experiments. Then, in the following section we describe and present the results of the validation and sensitivity experiments outlined above. Brief descriptions of the CP-67 dispatching algorithm and of the job streams used in the experiments are given in Appendices 1 and 2, respectively.

Simulator overview

The purpose of this section is to provide a general description of the CP-67 simulator on which this work was based, avoiding the specifics of implementation where possible. The reader interested in a more complete treatment of this material is directed to Ref. [1].

In the Introduction, the simulator was characterized as being "detailed" and "trace-driven." "Detailed" is, of course, a relative term, and is used here merely to suggest a greater level of detail than is perhaps typical of full-system simulators. For example, the basic unit of time in our simulator is one microsecond, which permits explicit accounting of the overheads for elemental but frequently occurring operations associated with paging and other I/O, dispatching, etc. By "trace-driven" we refer to the fact that job streams are represented by data derived by tracing actual jobs in execution. Furthermore, this data defines explicit sequences of resource demands that are interpreted by the simulator in a strictly deterministic manner, i.e., trace data are not reduced to produce statistical distributions or other summary characterizations to be used as input for a stochastic driving mechanism. As we shall see later, the level of detail in this trace-derived data is, again, quite high. An inevitable disadvantage of our approach is, of course, significant simulator execution time; in particular, depending on the job stream and other factors, from one to ten seconds of execution time are required to simulate one second of real time. On the other hand, there is the advantage of the potential ability to simulate accurately the effects of relatively minor changes in system software and hardware. In fact, the simulator has already been used to investigate the effects of page sharing [4], various CMS/APL performance enhancements [5], response surface techniques for system tuning [6], and an alternative dispatching strategy [7].

Before elaborating further on the CP-67 simulator, we should recall that CP-67 is a hypervisor for the System/360 Model 67, which provides functional simulations of multiple System/360s to users at remote terminals. These functional simulations, called virtual machines, may potentially all have different configurations and run different virtual operating systems. In our simulation work, however, we have generally concentrated on virtual machines running the interactive CMS operating system [2], and in the experiments reported here, in particular, all virtual machines were running CMS.

Key to the operation of CP-67, and hence to the simulation of CP-67, is the System/360 distinction between privileged and non-privileged instructions, the former being executable only when the CPU is in supervisor as opposed to problem state. This distinction enables CP-67 to execute a virtual machine's non-privileged instructions directly on the CPU in problem state, whereas privileged instructions are executed interpretively by software. In essence, then, CP-67 consists of a set of procedures for simulating privileged instructions, plus procedures for virtual memory management, user (i.e., virtual machine) dispatching, real I/O management, and accounting. To facilitate performance monitoring, these procedures maintain various time and event counters, as described in [8]. It is, of course, unnecessary for CP-67 to provide high-level support for such functions as language translation, file maintenance, link-editing, etc., because this support is provided by the conventional operating systems which run on virtual machines. Accordingly, it is unnecessary to deal explicitly with high-level operating system functions in simulating CP-67. The simulation of such functions is provided, in effect, by tracing all activities of a virtual machine, i.e., operating system as well as problem program activities.

Our approach, specifically, is to perform full-instruction traces of a virtual machine's activities while running jobs of interest. Subsequently, the trace data are compacted to form files (called SIMLOAD files) whose records (called load macros) represent the resource demands of the virtual machine as seen by CP-67. Combinations of these files, representing the job streams of interest, then form the input for the simulator. The manner in which the original trace data are compacted to form load macros is, of course, crucial to the operation and accuracy of the simulator and can be summarized as follows. Privileged instructions are mapped to load macros in a one-to-one fashion, along with various data necessary for simulation. For example, load macros used to represent start-I/O instructions provide virtual I/O device addresses, channel program descriptions, referenced page lists, etc. Non-privileged instructions, on the other hand, are aggregated to varying degrees, depending on the requirements of the simulation experiment. In general, a sequence of n non-privileged instructions is represented by a load macro specifying the value n and a list of the set of distinct pages which the instructions caused to be referenced (change-bits, as well as page numbers, are

recorded for both privileged and non-privileged load macros). Pages are listed in order of first reference, but information as to the exact sequence of page usage is lost. This loss of page sequence information, clearly a source of simulation error, is controlled by prescribing limits on the maximum number of instructions and/or pages to be represented by a single load macro. During compaction, if one of these limits is reached before a privileged instruction is encountered, the current macro is recorded and a new one started. Encountering a privileged instruction, of course, terminates the current macro perforce. In experiments reported in [1] it was found that errors due to the loss of sequence information are, in fact, tolerable for most purposes (e.g., page exception rates accurate within five percent) if the load macro page limit is less than the average number of pages available to a virtual machine while in execution. In the experiments reported here, this condition was satisfied by setting the page limit to ten (the instruction limit was sufficiently large as to be ineffective).

While we will not here describe the internal structure of the simulator explicitly, some indication of this structure is provided by considering some of the available controls and output capabilities. The available controls include system parameters, load parameters, and operational controls, as follows:

1. *System parameters*—hardware configuration and characteristics, software options (e.g., several alternative, parameterized paging and dispatching algorithms are built-in), and software overhead times (see, e.g., Table 9);
2. *Load parameters*—number of virtual machines, SIMLOAD file sequence for each virtual machine, average instruction execution time to be associated with each file, and virtual-to-read (mini-) device mappings;
3. *Operational controls*—initial memory state, shared/fixed page specifications, initial virtual machine states, delays before virtual machine start-ups, and conditions for simulator interruption (e.g., for periodic output) and termination.

The available outputs include tables summarizing hardware utilization, various event counts and rates, individual virtual machine activities, current system and virtual machine state variables, and individual virtual page usage patterns. Included, in particular, are outputs corresponding to the various time and event counters maintained by CP-67 itself. Outputs can be triggered not only when specified points in time have been reached, but also when various system and virtual machine events are recognized. The latter events may include high-level events such as SIMLOAD end-of-files (for response-time recording) and low-level events such as page exceptions and queue transitions (see Appendix 1).

Statistical design of experiments

In this section, we review some of the basic concepts underlying the statistical design and analysis of experiments, in order to facilitate understanding of the analyses described in the next section. A thorough treatment of the subject is provided by Cochran and Cox [9]. Although these ideas have been widely and profitably applied in a number of different areas of experimental research over a period of many years, they seem not to have been employed very extensively in the study of computer systems. A few examples where such techniques have been successfully applied in computer studies are given in Refs. [10-12]. It is our belief that this methodology can be extremely valuable in studying a wide range of computer system design and evaluation problems, but that there is a large communication gap which must be bridged between computer systems people and statisticians. The reader who is already familiar with this subject matter may wish to proceed directly to the following section.

Statistical design of experiments embodies principles to maximize the information per observation, and to permit valid inferences about the effects, on responses of interest, of variations in factors under the experimenter's control. One of the key ideas underlying this discipline is that of varying several factors simultaneously, rather than one at a time, as had been the usual practice in experimental science prior to the introduction and formalization of the fundamental concepts of experimental design by Fisher [13]. By varying factors simultaneously, the experimenter can capture information pertaining to their joint effects, or interactions, as well as their individual or main effects. Additionally, different effects can be estimated from the same set of observations.

A simple example of this notion is that of a balanced factorial experiment with two factors, each to be varied at two different levels (high vs low, or on vs off). We will label the factors A and B, and the factor levels 1 and 2. The total number of observations, or experimental runs, will be denoted by $N = 4n$.

The one-factor-at-a-time approach is to do n runs at each level of each factor, as shown in Table 1. The effect of varying factor A, say, can be measured by the difference in the means of the observations at each level of A. The appropriate t-test for assessing the statistical significance of this difference uses a sample standard deviation based on $2n - 2$ degrees of freedom, computed by pooling the sample standard deviations at each level. In essence, the standard deviation among observations, recorded under identical experimental conditions, provides an estimate of the variation in model errors.

The main effect of varying factor B can be measured in exactly the same way, but because each factor is varied separately, the experiment does not provide any in-

formation about possible interactions between the two factors. That is, we have no way of knowing whether the effect of varying factor A is different at different levels of factor B.

The factorial approach is shown in Table 2. For the same expenditure of observations, we now have $2n$ observations at each level of each factor, and can estimate the interaction between the factors. We could, using the factorial design, cut the size of the experiment in half, leaving us with n observations at each level of each factor, as in the one-at-a-time approach. We would then have approximately the same precision in estimating main effects as in the one-at-a-time experiment, plus the ability to estimate the interaction of the two factors.

For experiments involving larger numbers of factors, the number of factor-level combinations increases geometrically, but higher order interactions can be estimated. In many applications, certain high order interactions may be assumed to be negligible, and further economies can then be realized by using designs which are fractions of full factorials. They are designed in such a way as to sacrifice information on specified interactions by eliminating particular factor-level combinations. The examples discussed in the following section employ these fractional factorial designs exclusively.

The main idea behind fractional factorials is illustrated by Fig. 1, which depicts a $\frac{1}{2}$ fraction of a 2^3 design (3 factors, each at 2 levels). The coordinates of each vertex of the cube represent the levels of factors A, B and C. Note that the design corresponding to the four vertices denoted by the circles provides estimates of all three main effects. For example, the B main effect is measured by the difference between the means of the pairs of points on the top and bottom faces of the cube. The other main effects are measured by similar comparisons of observations on the other pairs of faces of the cube. In higher dimensions, smaller fractions of balanced points on the hypercube can be used.

The analysis of data from a designed experiment is based on the assumption of a linear model with normally distributed error terms. For a 3-factor design, the model would be

$$Y_{ijkl} = \mu + \alpha_i + \beta_j + \gamma_k + (\alpha\beta)_{ij} + (\alpha\gamma)_{ik} + (\beta\gamma)_{jk} + e_{ijkl};$$
$$i = 1, 2,$$
$$j = 1, 2,$$
$$k = 1, 2; \qquad (1)$$

where, for example, α_i is the effect due to factor A being at level i, β_j is the effect due to factor B being at level j, and $(\alpha\beta)_{ij}$ is the interaction effect due to factor A being at level i and factor B being at level j. The e_{ijkl} are assumed to be independent errors from a normal distribution with mean zero and unknown variance σ^2.

This model states that each observation is composed of an overall mean, μ, plus main effects due to the three factors, plus interaction terms and a random error term. The index l is used to represent possible replications of observations made under identical experimental conditions. Usually, an analysis of variance is carried out on the data from a designed experiment. Essentially, the total sum of squares of all observations is partitioned into component sums of squares corresponding to the

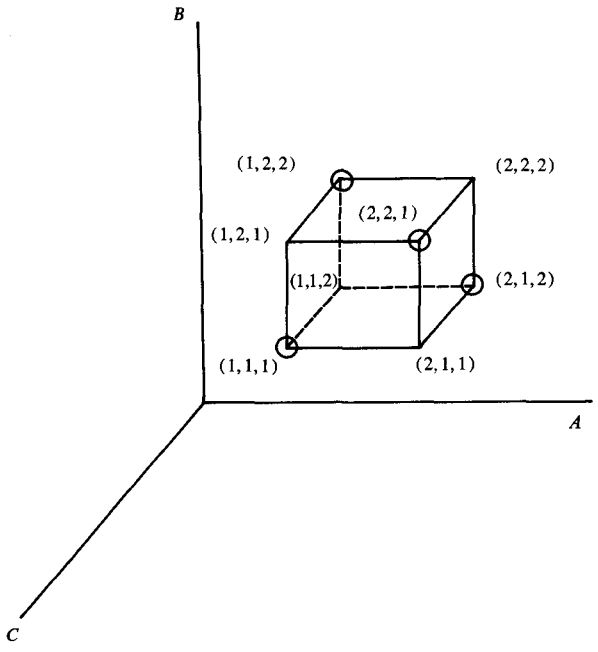

Figure 1 One-half replicate of a 2^3 fractional factorial design.

Table 1 One-factor-at-a-time arrangement.

A		B	
1	2	1	2
n	n	n	n

Table 2 Factorial arrangement.

		A	
		1	2
B	1	n	n
	2	n	n

Table 3 Experimental design for validation study.

Experimental run	*Paging penalty*	*CPU use penalty*	*Max page I/O check*	*Variable multi-prog. control*	*Two-level Q1*
1	−1	−1	−1	1	1
2	1	−1	−1	1	−1
3	−1	1	−1	1	−1
4	1	1	−1	1	1
5	−1	−1	1	1	−1
6	1	−1	1	1	1
7	−1	1	1	1	1
8	1	1	1	1	−1
9	−1	−1	−1	−1	−1
10	1	−1	−1	−1	1
11	−1	1	−1	−1	1
12	1	1	−1	−1	−1
13	−1	−1	1	−1	1
14	1	−1	1	−1	−1
15	−1	1	1	−1	−1
16	1	1	1	−1	1

Table 4 Raw data.

	Proportion problem state		*Proportion supervisor state*		*Page reads per second*	
Run	*Act.*	*Sim.*	*Act.*	*Sim.*	*Act.*	*Sim.*
1	0.286	0.330	0.360	0.313	98.007	89.530
2	0.285	0.286	0.355	0.356	92.675	102.300
3	0.291	0.307	0.335	0.334	87.638	97.700
4	0.283	0.319	0.344	0.334	92.416	95.990
5	0.377	0.427	0.320	0.292	73.844	80.130
6	0.379	0.384	0.299	0.291	66.675	75.250
7	0.365	0.438	0.289	0.292	62.218	76.340
8	0.394	0.415	0.322	0.303	74.176	77.450
9	0.402	0.461	0.250	0.249	43.052	50.000
10	0.398	0.458	0.220	0.216	32.351	37.690
11	0.397	0.469	0.222	0.227	33.607	39.490
12	0.391	0.461	0.249	0.248	42.612	49.670
13	0.417	0.449	0.219	0.212	29.953	35.860
14	0.389	0.475	0.253	0.228	42.775	50.270
15	0.391	0.447	0.247	0.258	40.237	49.750
16	0.449	0.451	0.218	0.215	30.377	37.630

terms in the linear model. Each of these is then divided by its appropriate degrees of freedom to obtain sample variances. The sample variances corresponding to the main effects and interactions are then compared to the error variance by means of F-tests in order to assess the statistical significance of each of the observed effects.

Because the validity of inferences drawn from such analyses is dependent on the adequacy of the assumed model, various analyses and plots of residuals (differences between the observations and fitted values obtained from the model) may be employed to indicate whether there appear to be serious discrepancies.

Validation experiments

As noted in the Introduction, one of the objectives of the simulation project was to construct a tool that could be reliably used to evaluate design changes in scheduling and dispatching algorithms. Such design changes could encompass a spectrum of possibilities ranging from that of changing values of fixed parameters in the existing algorithm, to that of replacing the algorithm with a completely new one. It was envisioned from the start of the project that factorial experiments would be used to specify patterns of parameter changes and to assess the effects of such changes on various aspects of system performance. Therefore, a necessary objective of the validation was to demonstrate that experiments of this type would produce similar effects both in the actual system and in the simulator.

The specific dispatching experiment chosen as the basis for the validation involved the following five parameters (factors), each of which could be viewed as an on-off switch.

x_1 Paging penalty
x_2 CPU usage penalty
x_3 Maximum page I/O preemption
x_4 Maximum multiprogramming level
x_5 Two-level Q1

A summary description of the algorithm is provided in Appendix 1; a more detailed account is given in a technical report by Schatzoff and Wheeler [14].

The experiment consisted of a one-half replicate of a 2^5 factorial. It used a workload of four different CMS job streams, each replicated different numbers of times, to provide a total of 40 virtual machines. The workloads are described briefly in Appendix 2. The design of the experiment is given in Table 3, which uses plus ones and minus ones to represent on and off conditions, respectively.

The results, for three variables of interest—CPU problem state time, CPU supervisor state time, and page read rate—are given in Table 4. Ten minutes of real time were simulated for each run. A number of other variables were also measured. However, the key aspects of the system's performance can be characterized adequately for our purposes by these parameters, which represent the CPU throughput, overhead, and paging activity levels of the system. The same methodology could be applied to any other variables of interest.

Initial examination of the data is not very revealing. It shows that our estimates of problem state time and page read rate are more than ten percent too high on average, while supervisor state time is much more closely ap-

proximated, being only about three percent too low. Individual estimates are sometimes in error by over twenty percent. It should be recalled, however, that the objective is to provide valid information concerning the effects, on the average, of changes in parameter values, rather than to estimate individual results with great precision.

A more incisive view of the data, then, is provided by Table 5, which shows the main effects and interactions for each of the variables, for both the simulator and the actual system. These statistics show the deviations from average attributable to the various factors, singly and pairwise.

To understand how these statistics are computed, we generalize Eq. (1) to five factors and re-parameterize it in the form

$$Y_{ijkl} = \sum_{i=0}^{5} \theta_i x_i + \sum_{i \neq j=1}^{5} \theta_{ij} x_i x_j + e_{ijkl}, \qquad (2)$$

where x_i has the value +1 or −1 depending on whether the ith switch (factor) is on or off. Thus, in terms of Eq. (1), we have imposed the constraints $\alpha_1 + \alpha_2 = 0$, $\beta_1 + \beta_2 = 0, \cdots$, etc. in order to be able to estimate the effects θ_i, $\theta_{ij}, \cdots$, etc. It should be noted that the re-parameterization reduces the model to the form of a regression equation in which the x_i are the values of the independent variables, and the θ_i, θ_{ij} are the regression coefficients.

The main effect for a given factor is computed by subtracting the mean of all observations taken at the low (−1) level of the factor from the mean of the observations at the high (+1) level and dividing the resulting difference by 2 [15]. All of the main effects can be readily computed in one step by means of a simple matrix multiplication. Let us denote by **X** the design matrix of Table 3, and by **Y** the data matrix of Table 4. Then, the main effects given by rows 2-6 of Table 5 are computed by $\mathbf{X}'\mathbf{Y}/16$, where $\mathbf{X}'$ is the transpose of **X**. Interaction effects are computed in a similar way. First, columns are adjoined to **X** by computing the element by element products of each pair of columns of the original **X** matrix. A new column, formed as the product of columns corresponding to factors i and j, represents the (i, j) interaction effect. Then, the computation $\mathbf{X}'\mathbf{Y}/16$, where **X** represents the extended matrix, provides not only the main effects, but the interactions as well. Adjoining a column of 1s to the **X** matrix will yield the overall means (row 1 of Table 5) in the same computation.

A striking aspect of the analysis provided by Table 5 is that for each variable, only three or four effects are large in magnitude relative to all the others. Furthermore, they are the same set of effects for all variables, the directions of all such effects for both the simulator and the real system are always identical, and the magnitudes are approximately equal as well. For example, imposition of a fixed maximum multiprogramming level results in a substantially lower paging rate, which is accompanied by a sharp reduction in overhead and a corresponding increase in problem state. The preemption of tasks which exceed the maximum allowed paging rate produces similar but somewhat smaller effects on these three variables. Unfortunately, the gains reflected by the main effects of these two factors are not strictly additive, because the factors interact negatively with one another on all three variables of interest, as can be seen from the (3, 4) row of Table 5. Another effect that seems to be marginally important is that of two-level Q1, which results in reduced paging rates and overhead, but does not seem to affect problem state.

Table 5 Factorial effects.

	Proportion problem state		*Proportion supervisor state*		*Page reads per second*	
	Act.	*Sim.*	*Act.*	*Sim.*	*Act.*	*Sim.*
θ_0	0.3684	0.4111	0.2814	0.2730	58.9133	65.3156
θ_1	0.0026	−0.0049	0.0011	0.0009	0.3438	0.4656
θ_2	0.0018	0.0023	−0.0031	0.0034	−1.0032	0.1869
θ_3	0.0268	0.0247	−0.0105	−0.0116	−6.3814	−4.9806
θ_4	−0.0359	−0.0478	0.0466	0.0414	22.0428	21.5206
θ_5	0.0034	0.0012	−0.0100	−0.0105	−3.2128	−4.3431
$\theta_{1,2}$	0.0065	0.0031	0.0039	−0.0022	1.6413	−0.7831
$\theta_{1,3}$	0.0050	0.0004	0.0010	−0.0030	0.6251	−0.6506
$\theta_{1,4}$	0.0001	−0.0073	0.0009	0.0058	0.1856	0.4456
$\theta_{1,5}$	0.0029	−0.0043	−0.0022	0.0006	−0.5896	0.2019
$\theta_{2,3}$	0.0029	−0.0003	0.0013	0.0023	0.2233	−0.2294
$\theta_{2,4}$	−0.0010	0.0042	−0.0024	−0.0020	−0.8409	−0.1531
$\theta_{2,5}$	0.0000	0.0047	0.0000	0.0011	−0.0428	1.2031
$\theta_{3,4}$	0.0195	0.0281	−0.0100	−0.0082	−5.3464	−4.5631
$\theta_{3,5}$	0.0040	−0.0064	−0.0046	0.0016	−2.0133	0.2781
$\theta_{4,5}$	−0.0076	0.0033	0.0050	0.0036	2.0857	1.7844

Thus far, we have commented in a qualitative way on two important questions raised by the simulation experiment, namely, how well do the simulator results approximate those of the real system, and how significant are the various effects that have been calculated. A partial answer to the first question is provided by comparing differences in effects measured on the simulator and the real system with independent estimates of errors from one run to another on the real system. The latter measurements were obtained from initial replicated trials carried out for the purpose of checking out the testing procedures to be employed on the real system. The data from these tests, which were run under three different combinations of dispatch parameter settings, are shown in Table 6, together with the pooled-within-sample stan-

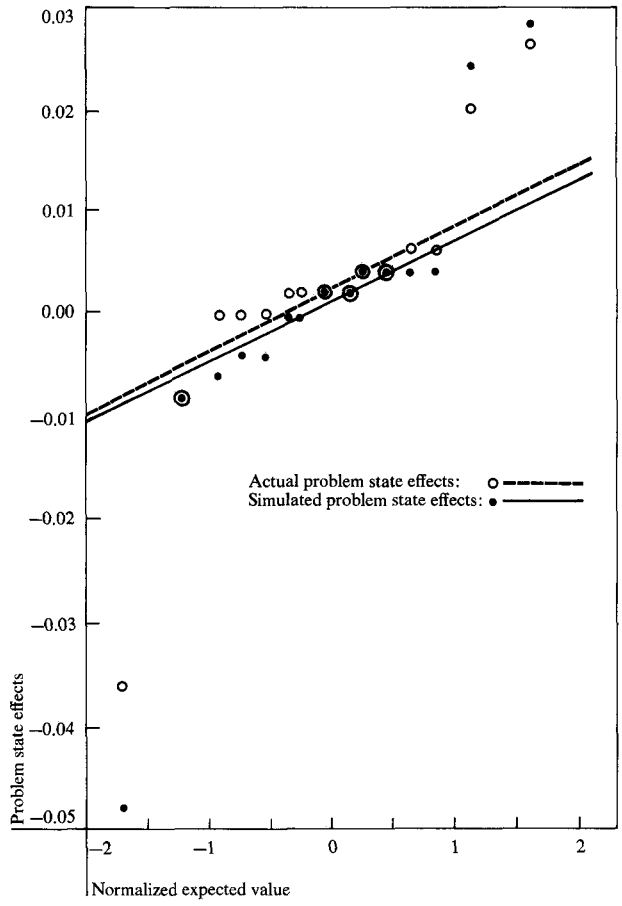

Figure 2 Normal probability plot of actual and simulated problem state effects.

Table 6 Repeatability of real system measurements.

Parameter settings	*Proportion problem state*	*Proportion supervisor state*	*Page reads per second*
1	0.335	0.316	77.10
	0.291	0.333	85.41
	0.275	0.342	92.65
	0.273	0.346	92.21
2	0.460	0.237	18.41
	0.456	0.237	17.79
	0.441	0.224	17.64
	0.458	0.231	19.42
3	0.421	0.285	51.79
	0.424	0.280	49.78
	0.410	0.290	55.02
Pooled-within-sample standard deviations	0.0188	0.0095	4.64

dard deviations. The corresponding standard deviations of differences between two individual effects, each of which is a mean of 16 observations (with alternating signs), are therefore estimated by dividing these within-sample standard deviations by $\sqrt{8}$ (since the variance of the difference between the means of two samples of size n is given by $2\sigma^2/n$). These are compared in Table 7 with the standard deviations of measured differences in effects between the real system and simulator, which may be readily calculated from Table 5. We may conclude from these comparisons that differences in factorial effects measured on the simulator and the real system are not significantly different from those that might be expected between one machine run and another.

An interesting and revealing graphical representation of the close agreement in simulator and real system effects is provided in Figs. 2 and 3, which are normal probability plots of the effects and their differences. In a normal probability plot, the observations are arranged in ascending order, and plotted against the standardized expected values of the corresponding order statistics of a sample from a normal distribution. The plot has the property that a random sample drawn from a normal distribution should scatter about a straight line. Based on the linear model of Eq. (1), if the underlying effects are all non-existent (i.e., $Y_{ijkl} = \mu + e_{ijkl}$), then the computed effects should plot as a straight line. If most effects are non-existent (or negligible), but a few are large in magnitude, then the plot should show most points lying close to the straight line, but those corresponding to the non-negligible effects should be relatively far away from the line.

Figure 2 shows the probability plots for the 15 problem state effects calculated from the experiments on the real and simulated systems. These plots are almost identical. Each clearly reveals the presence of two large positive effects and one large negative effect. As noted from Table 5, these correspond to the main effect for factor x_3 (maximum page I/O preemption), the (3, 4) interaction, and the main effect for factor x_4 (maximum multiprogramming control), respectively. Figure 3 is a normal probability plot of the differences in the effects between the actual and the simulated system. These plot very close to a straight line, indicating that simulator errors are random in nature and independent of parameter changes.

These graphs help to answer our second question, which deals with assessment of significance of effects. A more formal analysis is provided in Table 8, which presents t-values for the various effects, based on the standard errors of effects estimated on the actual system (last row of Table 6 divided by 4). If we refer these to tables of the t-distribution on eight degrees of freedom, we find

Table 7 Standard deviations of differences between factorial effects.

	Proportion problem state	*Proportion supervisor state*	*Page reads per second*
Real system (replicated runs)	0.0066	0.0034	1.64
Simulator-real system	0.0068	0.0038	1.22

Table 8 t-Values of effects.

	Proportion problem state		*Proportion supervisor state*		*Page reads per second*	
	Act.	*Sim.*	*Act.*	*Sim.*	*Act.*	*Sim.*
θ_1	0.56	−1.05	0.47	0.36	0.30	0.40
θ_2	0.37	0.49	−1.30	1.41	−0.86	0.16
θ_3	5.69	5.25	−4.37	−4.84	−5.50	−4.29
θ_4	−7.63	−10.17	19.43	17.24	19.00	18.55
θ_5	0.72	0.25	−4.17	−4.37	−2.77	−3.74
$\theta_{1,2}$	1.38	0.65	1.61	−0.94	1.41	−0.68
$\theta_{1,3}$	1.06	0.09	0.42	−1.25	0.54	−0.56
$\theta_{1,4}$	0.03	−1.56	0.36	2.40	0.16	0.38
$\theta_{1,5}$	0.61	−0.92	−0.94	0.26	−0.51	0.17
$\theta_{2,3}$	0.61	−0.07	0.52	0.94	0.19	−0.20
$\theta_{2,4}$	−0.21	0.89	−0.99	−0.83	−0.72	−0.13
$\theta_{2,5}$	0.00	1.00	0.00	0.47	−0.04	1.04
$\theta_{3,4}$	4.15	5.97	−4.17	−3.44	−4.61	−3.93
$\theta_{3,5}$	0.85	−1.37	−1.93	0.68	−1.74	0.24
$\theta_{4,5}$	−1.62	0.70	2.08	1.51	1.80	1.54

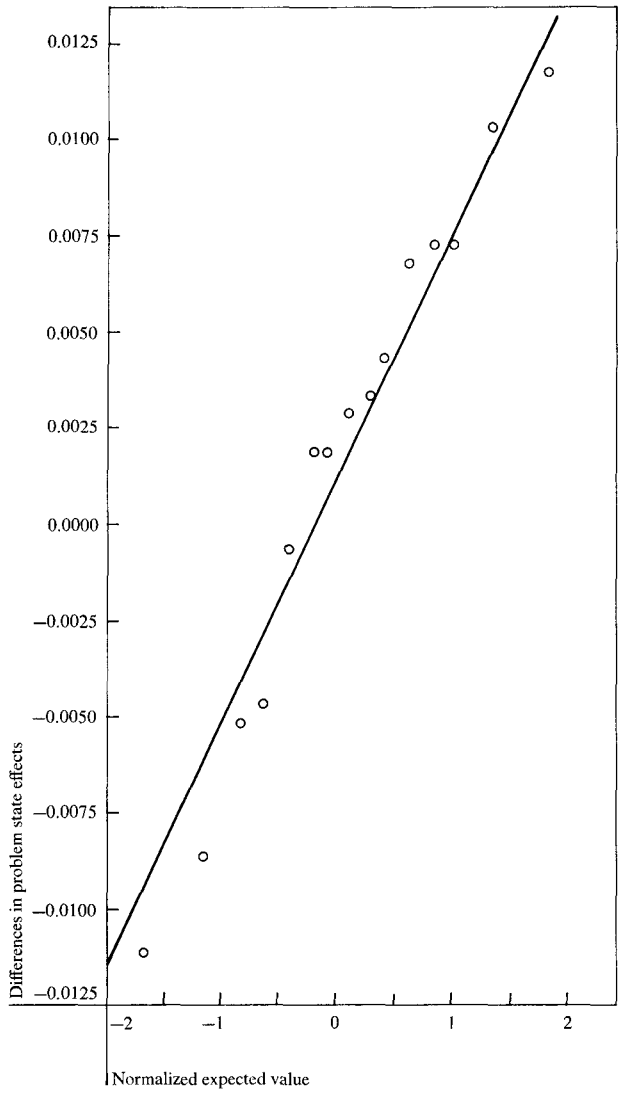

Figure 3 Normal probability plot—actual minus simulated effects.

that the three effects that revealed themselves so emphatically on the probability plot are all significant at the one percent level of significance or beyond. The two-level Q1 main effects are also significant on supervisor time and paging rate, although the t-value for the actual system on paging rate falls just short of the two percent level. No other effects are significant at the one percent level.

All of the analyses described thus far have been concerned with validation. An associated problem of interest is that of calibration, or adjusting parameters of the simulator in such a way as to provide better agreement in the outputs with those of the system being modeled. Validation and calibration may be viewed as iterative and complementary processes. During the course of our validation work, we were continually trying to locate and correct sources of discrepancies, and indeed were led to the discovery of a number of programming bugs and logical errors in our modeling of the system. Although we have been very pleased with the high degree of fidelity that has been achieved in estimating the effects of certain parameter changes, as already described, we have been somewhat perplexed by the apparent biases in estimating mean levels of problem state time and paging rates. Because there are a large number of timing parameters in the simulator model, we designed an experiment aimed at indicating the main effects of mild changes in the values of these parameters. Thirty-eight parameters were identified and classified into eight groups, as shown in Table 9. Within each group, the parameters were varied simultaneously at two levels, 15 percent above and below the nominal values for seven of the groups, and plus or minus 2.5 percent for the eighth, which represented disk characteristics. A 1/16 fraction of a 2^8 factorial was carried out, using the same workloads as before. However, only 400 s of real

Table 9 Factors for simulator sensitivity experiment.

Factor	Function	Parameter(s)	Values used (μs) Low (−1)	High (+1)
1	Dispatch – Task (re) start	T_READY	100	130
		T_DISP	50	65
		T_DISP_LOOP	10	13
		T_DISP_OVHD	85	111
		T_NODISP_OVHD	15	20
		T_REDISP	100	130
		T_REDISP_FAST	35	46
2	Dispatch – Qdrop	T_QDROP	400	520
3	Dispatch – Qadd test	T_QCHECK	500	650
4	Paging	T_PRALG	150	195
		T_PG_QUEUE	200	260
		T_PG_SIO	100	130
		T_PG_INT	600	780
5	Virtual selector I/O initiation all virtual I/O termination	T_CCWTRANS	1500	1950
		T_DIAGTRANS	500	650
		T_IO_QUEUE	100	130
		T_DISK_SIO	150	195
		T_START_SEEK	85	111
		T_RIO_INT	300	390
		T_VIO_END	125	163
		T_VIO_INT	175	228
6	Console I/O	T_CONS_DISC	1100	1430
7	Disk characteristics	T_DISK_SEEK0	23300	24500
		T_DISK_SEEK1	475	500
		T_DISK_SRCH	11870	12500
		T_DISK_XFER	3.05	3.21
8	Privileged instruction simulation	T_SIM_SSK	282	367
		T_SIM_ISK	282	367
		T_SIM_SVC	51	66
		T_SIM_SSM	205	257
		T_SIM_LPSW	205	257
		T_SIM_DIAG	196	255
		T_SIM_WRD	118	153
		T_SIM_RDD	118	153
		T_SIM_SIO	368	478
		T_SIM_TIO	368	478
		T_SIM_HIO	368	478
		T_SIM_TCH	325	423

time were simulated in this experiment, as compared with 600 s in the previous one. In this design, all main effects are estimable, but the two-factor interactions are confounded with one another in groups of four.

The results of the experiment are analyzed in Table 10, which shows that problem state time is relatively insensitive to variations in the design factors. None of the main effects is significant at the five percent level, and the largest main effect is only two percent of nominal in magnitude. There are significant effects in some of the other variables, for example, paging rate and supervisor state time, but the magnitudes of these effects are much smaller, proportionately, than the variations in the individual factors. Thus, it appears that the simulator is fairly robust with respect to moderate variations in timing parameters.

Table 10 Factorial effects for calibration experiment.

	Proportion problem state	Proportion supervisor state	Page reads per second
θ_0	0.3533	0.3127	74.139
θ_1	−0.0021	0.0063	0.444
θ_2	−0.0054	−0.0030	−1.008
θ_3	0.0041	−0.0001	−0.594
θ_4	0.0017	0.0245	2.903
θ_5	−0.0063	0.0114	1.427
θ_6	0.0005	0.0028	1.329
θ_7	−0.0074	−0.0054	−1.671
θ_8	−0.0017	0.0019	−0.437
$\theta_{1,5}$	0.0008	−0.0016	−0.463
$\theta_{1,6}$	−0.0042	−0.0007	0.134
$\theta_{1,7}$	0.0047	0.0001	−0.303
$\theta_{1,8}$	−0.0002	0.0006	0.135
$\theta_{1,3}$	0.0052	0.0014	0.372
$\theta_{1,4}$	−0.0028	−0.0035	−1.332
$\theta_{1,2}$	−0.0018	0.0010	0.923

We have not uncovered the source of the observed biases, but remain confident in the ability of the simulator to accurately predict the effects of changes in the basic resource allocation algorithms.

Summary and conclusions

We have described a detailed trace-driven simulation model of a time sharing system (CP-67), and have shown how techniques of experimental design can be used to validate and calibrate such a model and to study the effects of changes in system design. Identical experiments involving five controllable factors of a dispatching algorithm have been carried out both on the simulator and the real system. The results indicate that the errors in estimating factorial effects are almost identical to those expected from one actual machine run to another.

Appendix 1: Dispatching algorithm

CP-67 employs two CPU service queues, Q1, which is used for servicing interactive requests (those which can be satisfied within a specified short interval of time) and Q2, which services requests that cannot be completed in the allotted Q1 quantum. Scheduling priorities for admission to Q2, the background queue of this so-called foreground-background dispatcher, are calculated for each user as functions of current time of day, initial assigned priorities, and calculated penalties for excessive paging or CPU activity. Whenever a user is dropped

Table 11 Jobstream summary.

Virtual machine numbers	*CMS functions performed*	*Sleep count*	*DASD I/O count*	*Virtual CPU time* (s)
1-16	7 text editor invocations	71	84	1.1
17-31	3 F-level assemblies 2 G-level FORTRAN compilations	5	1027	9.8
32-36	7 H-level assemblies	0	1308	26.2
37-40	2 F-level assemblies, 3 F-level PL/I compilations, 2 G-level FORTRAN compilations	0	2671	25.3

from Q2, another eligible user is admitted in order of scheduling priority, as long as the total of the estimated working sets of all Q2 residents does not exceed the available main memory.

The algorithm employs two feedback mechanisms to insure that mis-estimation of working sets does not result either in page thrashing or in under-utilization of the CPU. The first mechanism calculates an estimate of page thrashing which is used to scale individual working set estimates. The second is a check on individual paging activities, used for preemptive dropping from Q2 of offending users.

The combined effect of the Q2 admission policy and feedback mechanisms is to produce a variable multiprogramming level which attempts to adjust dynamically to changing workload conditions. A fixed maximum multiprogramming level can be imposed to prevent temporary large excursions.

When a task has not completed during its allotted Q1 quantum, it becomes eligible for Q2 admission. Under certain conditions (frequent terminal interactions, indicative of interactive work), the task will be dropped to a second level of Q1 and re-dispatched so that it can have a chance to complete before losing its resident pages.

Appendix 2: Job streams

The experiments described in this paper involved running forty virtual machines, each executing jobs under the CMS (version 3.1) operating system. In the case of the real system, the virtual machines were stagger-started at roughly one-second intervals using a modified version of CP-67 with an automatic log-on mechanism. This mechanism, in effect, simulated the logging on of a terminal user, the execution of a CMS initial program load (IPL), and the invocation of a CMS EXEC file specifying a sequence of commands to be executed. In the case of the simulator, the virtual machines were stagger-started at exactly one-second intervals using the delay mechanism mentioned in the simulator section. SIMLOAD file sequences were, of course, defined to represent the loads imposed by log-ons, IPLS, and the commands specified by the various EXEC files. In both the real system and the simulator, appropriate performance counters were recorded at the time of the last log-on, i.e., nominally forty seconds after start up, then again ten (real or simulated) minutes later. The data discussed under validation experiments were obtained by differencing the recorded counter values.

The forty virtual machines, in fact, ran only four distinct job streams, as summarized in Table 11. In this table, the virtual machine numbers represent log-on order, i.e., virtual machine n was the nth virtual machine to be logged-on. The "sleep counts" shown in the third column refer to delays interspersed between some of the commands to imitate the effects of terminal interactions. In the real system these delays were effected by modifying the CP SLEEP function to accept an argument representing "time before wake-up," while in the simulator special SIMLOAD control macros were employed. The information in Table 11 can be further summarized by saying that virtual machines 1-16 ran a highly interactive job stream imitating a sequence of edit sessions; virtual machines 17-31 ran a moderately interactive job stream involving assembly language and FORTRAN compilations; virtual machines 32-36 ran a non-interactive job stream involving repeated assembly language compilations; and virtual machines 37-40 ran a non-interactive job stream involving assembly language, PL/I, and FORTRAN compilations. Each of the four job streams was so designed that, in the multi-user context of the experiments, it would not reach completion before the end of the ten-minute measurement period.

References and notes

1. C. Boksenbaum, S. Greenberg, and C. Tillman, "Simulation of CP-67," *Scientific Center Report G320-2093*, IBM Data Processing Division, Cambridge, MA 02139 (June, 1973).
2. *CP-67/CMS System Description Manual Form No. GH20-0802-2*. IBM Data Processing Division, White Plains, NY 10601 (1971).
3. *IBM Virtual Machine Facility/370: Introduction, Form No. GC20-1800*, IBM Data Processing Division, White Plains, NY 10601 (1972).

4. P. N. Wahi, "On Sharing of Pages in CP-67," *Proc. ACM SIGARCH-SIGOPS Workshop on Virtual Computing Systems*, Harvard University, pp. 127-149 (March 26-27, 1973).
5. W. M. Buco, A. J. Cristoforo, D. J. Hatfield, and C. C. Tillman, "A methodology for paging performance enhancements," unpublished manuscript.
6. M. Schatzoff, and P. Bryant, "Regression Methods in Performance Evaluation: Some Comments on the State of the Art," *Proc. Computer Science and Statistics, 7th Annual Symposium on the Interface*, Iowa State University, pp. 48-57 (October 1973).
7. Y. Bard, "Application of the Page Survival Index (PSI) to Virtual-Memory System Performance," *IBM J. Res. Develop.* **19,** 212 (1975), this issue.
8. Y. Bard, "Performance Criteria and Measurement for a Time-Sharing System," *IBM Syst. J.* **10,** 193 (1971).
9. W. G. Cochran and G. M. Cox, *Experimental Designs*, John Wiley & Sons, Inc., New York, 1957 (2nd edition).
10. M. Schatzoff, R. Tsao, and R. Wiig, "An Experimental Comparison of Time Sharing and Batch Processing," *Comm. ACM*, **10,** 261 (1967).
11. R. L. Tsao, W. Comeau, and B. H. Margolin, "A Multifactor Paging Experiment: I. The Experiment and the Conclusions," in *Statistical Computer Performance Evaluation*, edited by W. Freiberger, Academic Press, Inc., New York, 1972, pp. 103-134.
12. Y. Bard, "Experimental Evaluation of System Performance," *IBM Syst. J.* **12,** 302 (1973).
13. R. A. Fisher, *The Design of Experiments*, Oliver and Boyd, Edinburgh, 1935.
14. M. Schatzoff, and L. H. Wheeler, "CP-67 Paging Priority Dispatcher," *Scientific Center Report G320-2088*, IBM Data Processing Division, Cambridge, MA 02139 (March 1973).
15. By convention in the special case of 2^n experiments, effects are frequently defined by taking only the difference.

Received June 20, 1974; revised December 20, 1974

The authors are located at the IBM Data Processing Division, Scientific Center, 545 Technology Square, Cambridge, MA 02139.