CALIBRATING THE SIMULATION MODEL OF THE IBM SYSTEM/360 TIME SHARING SYSTEM

P. E. Barker
IBM World Trade Corporation
Sindelfingen, Germany

and
H. K. Watson
International Business Machines Corporation
Systems Development Division
Kingston, New York

Abstract

A GPSS/360 model was developed to support the IBM System/360 Time Sharing System (TSS/360) design and evaluation effort. The model was validated by conventional desk-checking procedures and by an extensive measurement effort involving a hardware monitor (SPAR), a software monitor (SIPE), and miscellaneous data-reduction programs. Calibration results obtained with TSS/360 are described, as are some of the benefits derived from the model.

1. INTRODUCTION

Around the middle of 1965, the decision was made to write a GPSS/360 simulation model of the IBM System/360 Time Sharing System (TSS/360), which was then in the development stage. An earlier model had been written to simulate the hardware in order to evaluate such things as cable delays, memory interference, and dynamic-address-translation-feature degradation. The new model was to include these hardware considerations as gross factors, but would concentrate on the programming system with a view to evaluating system performance under different operating environments.

The objectives of the model were:

(1) To supply performance predictions to customers and prospective customers prior to the availability of the system.

(2) To assist the development effort by detecting logical errors and by suggesting design improvements.

(3) To evaluate the effect on performance of various user-specified system parameters.

In order to satisfy these objectives, it was thought essential to calibrate the model against live system measurements and thus establish the model's credibility. This has proved to be a continuing task because, as the system is developed, the model must be changed to reflect modifications. This report describes the calibration process and results obtained from calibrating the model.

2. THE MODEL

In order to make this report more meaningful, a brief description of the model is given here.

A TSS/360 user may invoke various functions by using commands, such as LOGON, LOGOFF, RUN FTN, and DATA. For modeling purposes, each function was broken down into its consistituent parts, which are termed subactions; typically, a function may be represented by 10 to 20 subactions. The characteristics of a subaction are the shared and non-shared pages referenced, the working storage pages obtained and freed, the number of pages moved between main storage and external storage, the number of IOCAL macro instructions issued (which provide for the initiation and execution of I/O operations), the virtual memory time (time during which the PSW is in the problem state), and the supervisor time (time during which the PSW is in the supervisor state). Early in 1967, before IBM System/360 Model 67 was available, these parameters were measured with an instruction-trace program on a Model 50 running interpretively as a Model 67. Later they were measured on a Model 67 using ITM (Instruction Trace Monitor).[2]

The subactions relate only to virtual memory activity-they do not consider the resident supervisor. The supervisor was modeled using logic descriptions in the development workbook and by keeping in close touch with the TSS/360 designers. Interrupt-handling times were either estimated from the flowcharts or measured with ITM.

The model contains about 2000 GPSS/360 blocks and runs one-to-three times slower than real time on a Model 65. The basic clock time is one tenth of a millisecond.

2.1 MODEL INPUT

The user-controlled variables can be classified into three main types: hardware configuration, parameter settings, and load. As the model is supplied with built-in values for each of the variables, it is only necessary for the user to respecify the particular variables he is experimenting with. This keeps the model input to a minimum.

Hardware configuration includes such data as number of CPU's, memory size, disk-storage drive type (IBM 2311 or IBM 2314), number of paging disks, number of public disks, and number of on-line terminals. The system-parameter settings that may be varied include the quantum length, number of quanta per time slice, maximum number of pages per task, main storage and drum overflow thresholds, and delta values; in fact, most of the items in the system table are conveniently repsented in a GPSS/360 function. The variables involved with specifying the load include the job type per terminal, the number of statements to be handled, the interterminal start time, the "think" time, the number and type of third-level tasks, and the bulk I/O load.

2.2 MODEL OUTPUT

The model contains a special report section, which provides data on the simulation run in an easy-to-read, English-language-like form. Broad external-performance figures, such as number of jobs of each type processed, response times per job, and hardware utilization, are listed. In addition, there is a wealth of information on the internals of the system; this is invaluable for checking out both the model and the system. Included is such information as the number and type of time-slice ends, the average number of tasks in main storage, the

average amount of main-storage space occupied, and the distribution of the average program size per time-slice.

3. STAGES OF CALIBRATION

There are two distinct stages in the checking of any model. The first falls under the heading of desk checking. The second, which is usually much more lengthy and expensive, can be called measurement; this includes measurements of the system modeled or, where this is not possible, measurements of some similar system and development of analytic approximations.

3.1 DESK CHECKING

Desk Checking the TSS/360 model was performed over a period of about two months by two people who had not been concerned with its implementation. This is thought to be a prerequisite of any good auditing procedure, since it is difficult to check one's own work. The basic approach during this period was to compare the model logic with the system specifications by a static study of the model listing. This revealed such errors as wrong pages referenced by a particular subaction and incorrect specification of profile parameters. It also provided an opportunity to learn how the model worked. The next stage was to perform simple model runs on the computer to check that the various modules were interfacing correctly. The first runs were single-thread. For these, the model was modified slightly to provide output at meaningful checkpoints, such as end of LOGON and end of RUN. Comparisons of statistics for these functions were then made against the specifications. Any discrepancies were resolved by obtaining model output at finer details of resolution.

These procedures provided a good check of the task profiles and flow but not of the supervisor logic with its complex dispatching algorithm, I/O-device handling, and storage management. A first attempt at checking this was made with multiple-thread runs of varying degrees of complexity, but little was proved except that the model ran to completion. The determination that the completion was successful and that the results were meaningful was accomplished by actual system measurements, which were compared to the results from the model runs.

3.2 MEASUREMENT EFFORT

Measuring a system's performance involves defining the environment, specifying the values to be measured, and developing the tools to provide the measurements. Since the measurements were oriented toward understanding why the system was behaving in a certain way, rather than just determining the system throughput, something more sophisticated than a stopwatch was needed. To capture the hardware-device utilizations and number of I/O events, a hardware monitor (SPAR) was used. Details of the internals of the operating system, such as main-storage utilization and distribution of time-slice ends, were obtained from a software tool (SIPE).

3.3 MEASUREMENT TOOLS

A conversational system, such as TSS/360, is inherently undisciplined and nonreproducible from the point of view of the user at the terminal: his keying rates, length of responses, and "think" times vary; his keying accuracy varies; his activity relative to some other terminal varies between different sessions of the same run. This makes it very difficult to perform a repeatable experiment.

Hence, conversational runs were made with TEST/360, a program that runs in a Model 40 connected to a Model 67. The Model 40 has a prestored dialog of the commands required at each terminal, and transmits them in a controlled manner and at specified time intervals to the Model 67. As far as TSS/360 is concerned, the situation is identical to having terminals on-line. Messages sent and received are time-stamped by the Model 40, and a data-reduction program produces a response-time analysis.

A gross measure of how the system is performing is provided by SPAR (System Performance Activity Recorder).[3] This is a hardware monitor which records system activity in mechanical and electrical counters under the direction of a wired control panel. By writing the contents of these counters on tape and by using an appropriate data-reduction program, a continuous picture of the system activity during the run can be constructed. SPAR is connected to the Model 67 through a special interface. The main use of this tool is for hardware measurements, such as channel and CPU utilization; however, in single-thread work, a limited amount of software information can be provided, such as length of a FORTRAN phase or of a time-slice.

The most suitable tool available for making internal measurements was SIPE (System Internal Performance Evaluation), a program integrated into the resident supervisor.[4] When SIPE is in use, it records in a main-storage buffer the time at which key modules (such as page posting, time-slice end, and move wall) are entered, together with other pertinent information. When the buffer is filled, the information is written on magnetic tape. The tape is processed later by various data-reduction programs. These have been designed so that the SIPE user may have reports at various levels of detail, ranging from overall run statistics down to the microsecond level, depending upon the problem being studied.

To avoid any misunderstanding, it should be stated that these tools were not developed solely for the simulation group, although it was one of the main users; other interested groups were design-evaluation and product-test. All of these groups were active in providing specifications for the tools and in monitoring their development.

3.4 THE ENVIRONMENT

To make maximum use of machine time, a measurement plan was drawn up by the simulation, design, and product-test groups such that, where possible, the same runs could be used for different purposes. Table 1 shows a typical set of loads which were used for calibrating the simulation model against an early release of TSS/360. They were of three types:

(1) One terminal, conversational only (runs 1-8)

(2) Two tasks, batch only (runs 9-10)

(3) Twelve terminals (run 11-14; three of these were conversational only, one included a FORTRAN batch compile).

The emphasis was on the 1024K, or maximum, system, but a few runs were made on the 512K, or minimum, system. A single IBM 2311 Disk Storage Drive was used as the auxiliary paging device, and one IBM 2311 was used as the public volume. Both were on the same channel.

A typical set of parameters used for early calibration runs is shown in Table 2. Although the settings used for the measurements were arbitrary to some extent, they were determined by simple analysis wherever possible. The simulation model provides an efficient method of determining optimum values of many of these parameters. It is also helpful in providing understanding of the interrelationships between parameters.

In general, the requirements for calibration are (1) that the environment can be reproduced very closely in the simulation model and (2) that it provides a representative application of the model rather than a limiting case.

4. CALIBRATION PROCESS

The actual method of performing a model calibration, once one is armed with both model and system measurements, is somewhat like programming-more of an art than a science. As such, it is difficult to lay down a set of rules on how to proceed. However, there were some broad principles that were followed each time.

ONE: The first measure of the model's performance is how it compares on the external characteristics of average elapsed time per job and average response time. These are the measurements presented in this paper. Initially these quantities did not compare well but, even if they had, it would not have guaranteed that the model was working correctly. Compensating errors could have produced the right answers for the wrong reasons.

TWO: A reasonable second step is to examine the gross internal characteristics, such as the number of drum reads and writes, the number and type of time-slices, and the average amount of unassigned main storage. This will usually give a strong indication of the area in which the trouble lies. For example, if a large number of AWAIT time-slice ends show up when the systems does not have a minimal amount of unassigned main storage, it suggests that the algorithm for dealing with tasks in AWAIT status is not working properly. Or if the system shows a higher space utilization of main storage than does the model, it suggests an error in the purging area, incorrect profiles, an error in model main-storage accounting, or one of several other possibilities.

THREE: The identification of the reasons for the discrepancy is one of the most challenging and interesting parts of the process. It involves forming a hypothesis and then checking it by studying more detailed data reductions, perhaps to the millisecond level, and by making additional model runs to examine the effect of certain changes. Most of the problems encountered lay in the area of system bugs, which were eliminated as they were identified. The main model weakness uncovered was in the area of reclaiming pages from the pending list. Due to accounting difficulties, the unclaimed pages for a task were discarded, and this resulted in a much lighter main-storage load than occurred in the real system. However, this was only significant under heavy load conditions, as shown in the following results.

4.1 CALIBRATION RESULTS

The results from one of the calibrations is shown in Table 3. This table shows the difference in elapsed times between the system and the simulator expressed as a percentage of system time. The

difference in response times are also shown for multi-terminal runs.

Figure 1 graphically displays the results shown in Table 3. The unshaded bars show the difference before corrections were made to the model while the shaded bars show results after corrections.

The elapsed time of the multiple-terminal runs includes "think" time, which was held at approximately 20 seconds for both system and simulator. For the single-terminal runs, "think" time was subtracted out of both system and simulator elapsed times to provide a comparison of actual operating time.

4.2 ANALYSIS OF RESULTS

The results shown in Table 3 include external comparisons only. This is not to say that internal comparisons were not made-they were. On all runs, SPAR and SIPE recordings were taken and checks were made on a myriad of internal parameters. However, none of the checks showed a serious discrepancy between the model and the system, and the detailed results have therefore been excluded in the interest of clarity and brevity.

As shown in Figure 1, using a four-core box (1024K), single-terminal system, the elapsed-time deviation for the basic model was approximately 5 percent. The exception to this were the FORTRAN loads (runs 1, 2, and 7).

On runs 1, 2, and 7, the simulator had the FORTRAN compiler in initial virtual memory (IVM). This feature was not incorporated into TSS/360 Release 1.1. Run 2A was made with a prerelease version of system 1.2, which did have the FORTRAN compiler in IVM. The deviation here was 5 percent, which was in keeping with the elapsed-time deviation for runs, 3, 4, 5, and 6.

Multiple-terminal deviation was approximately 24 percent.

The updated model gave improved results for multiple-terminal runs of about 8 percent. Single terminal runs were effectively unchanged at 6 percent. This indicates that the updated model calibrates to the same accuracy for single-or multiple-terminal loads.

The purpose of the runs on the two-core box (512K) system was to provide a checkpoint on the minimum system; as such, they are of more qualitative than quantitave significance. As shown in Figure 2, the deviation for the basic model was about 17 percent for single-terminal runs and 24 percent for multiple-terminal runs. The updated model gave corresponding results of 14 percent in each case.

A certain amount of instability exists with results that calibrate to better than 10 percent in that, as patches are incorporated, the calibration fails to behave in a smooth manner. This is because as the model is improved in one direction a discontinuity is crossed in another, and this now influences the performance. At this point, one is at the accuracy threshold, or noise level, of the model.

Response-time deviations averaged 51 percent for the basic model and 22.3 percent for the updated model. These figures are misleading in that they include comparison of very short response times, where a small absolute difference may be a large percentage difference. A more meaningful statistic is the weighted deviation. Here the figures are 45 percent and 7 percent respectively, for the basic and updated models.

5. SOME BENEFITS OF THE TSS/360 MODEL

Some of the advantages of having a model are rather subtle and it is appropriate to set them forth, although the following is not necessarily their order of importance:

The process of measuring the system and examining data reductions was a fruitful way of finding system bugs. Although the model itself did not find the bug, the process of calibration did. Without the model, it would have been more difficult to know what to expect from the output.

In many cases, TSS/360 design changes could be evaluated more readily and economically with the model than by patching the real system. This was in spite of the fact that the model ran one-to-three times more slowly than real time, because the system also was slower than real time! Besides running the test case, the system had to go through initialization procedures that could take hours, especially during the early stages of the system development when system reliability was fairly low. On top of this, one had the cost of a Model 40 to drive TEST/360, a Model 67 to run the test case, and a Model 65 to run the data reduction. This cost is high when compared with that of only a Model 65 to run the model.

System Performance could be evaluated under a wider range of load conditions and configurations than was possible on the real system for the following reasons:

(1) In the early stages of TSS/360 development, the system reliability was so low as to preclude runs of any length or complexity.

(2) Similarly, the functional capability was lacking initially; for example, at one point the system could support no more than 12 on-line terminals.

(3) The resources to generate diverse test cases were not available, whereas load specification was a trivial task for the modeler.

(4) The physical configuration was not always available, as when a proposed new I/O device was being evaluated.

(5) Model 67 time was the scarcest resource available during the early development days.

For the same reasons, it was easier to evaluate the interrelationship between the system parameters with a model than by a series of runs on the system.

6. FUTURE OF THE MODEL

The model of TSS/360 has by no means outgrown its usefulness; in fact, it is perhaps now that it can be of greatest value. It has been rewritten and extended to include current system improvements. By being ahead of the implementation effort, it is possible for the model to detect design weaknesses and indicate improvements early enough to affect the real product. Besides this, there is the ability to provide fast, in depth studies of system performance under a wide range of loads and operating environments, even with new hardware that may not exist beyond a set of functional objectives.

7. CONCLUSION

This study has shown that it is possible to simulate a complex operating system with high accuracy and yet retain the traditional modeling advantages of shorter lead time to a finished product and increased system insight through simplification.

8. CITED REFERENCES

1. W. J. Kahn and R. E. Ross, Hardware Configuration Analysis of the IBM System/360 Model 67, IBM TR 53.0010, IBM Systems Development Division, Yorktown Heights, New York, 1969.

2. C. E. Seabold, An Instruction-Trace Technique for Time Sharing System/360, IBM TR 53.0012, IBM Systems Development Division, Yorktown Heights, New York, 1969.

3. F. D. Schulman, "Hardware Measurement Device for IBM System/360 Time Sharing Evaluation," Proceedings of the 22 National Conference, Association for Computing Machinery, P-67, pp. 103-109, Thompson Book Co. Washington, D. C., 1967.

4. W. R. Denniston, SIPE: A TSS/360 Software Measurement Technique, IBM TR 53.0013, IBM Systems Development Division, Yorktown Heights, New York, 1969.

9. BIOGRAPHIES

H. K. Watson

H. K. Watson received his Bachelor of Science Degree in Electrical Engineering from the University of Colorado in 1949. He was an instructor for the U.S. Air Force from 1950-1951 and joined International Business Machines Corporation in 1951 as a customer engineer in the Denver, Colo., facility. He later worked in a similar capacity in San Francisco, Calif., Seattle, Wash., Oakland, Calif., and Poughkeepsie, N.Y. He has worked since 1961 in teleprocessing and time sharing system design in Poughkeepsie and Kingston, N.Y. His current assignment includes simulation of large system hardware and software in conjunction with system design.

The author is a member of IEEE and Eta Kappa Nu.

P. E. Barker

P. E. Barker received his B.S. in Electrical Engineering from Queens University in Belfast and his Diploma of Imperial College from Imperial College in London where he studied telecommunications. He next spent two years with the Royal Radar Establishment in England where he was engaged in microwave and missile guidance systems.

The author joined IBM United Kingdom, Ltd., in 1958 as an applied science representative. He was later assigned to the advanced market development area where he was involved in real time system design. His next IBM assignment was with Systems Development Division in Poughkeepsie, N.Y. where he was involved with OS/360 and time sharing systems performance evaluation. He is currently with IBM's World Trade Corporation in Germany where he is assigned to the complex systems department.

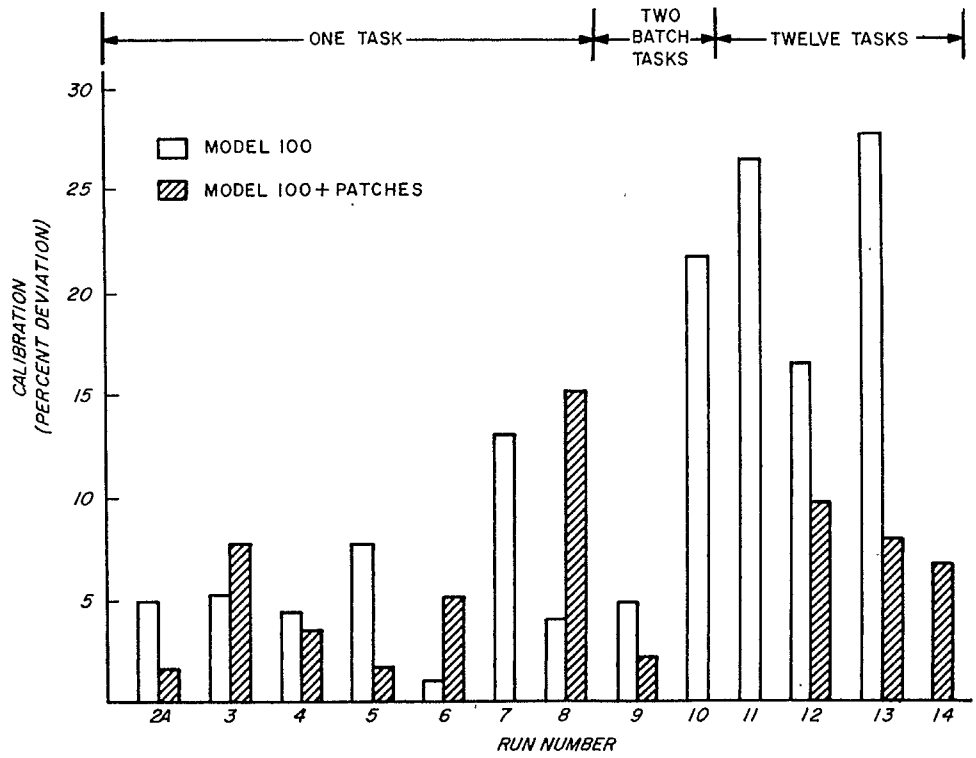

Figure 1. Calibration results for the 1024K (four-core-box), or maximum, system. The values for the seventh and tenth runs are distorted, since the model included the FORTRAN compiler in IVM, which was not true for Release 1.1 of TSS/360. Table 3 contains the data on which this graph is based.

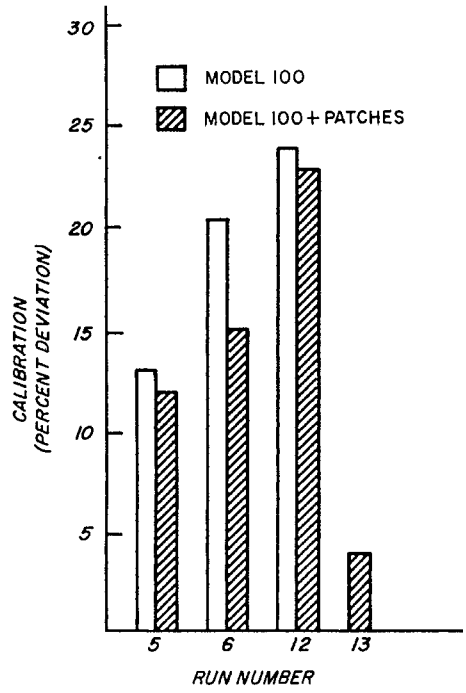

Figure 2. Calibration results for the 512K (four-core-box), or minimum, system. The data on which this graph is based are contained in Table 4.

Table 1. List of measurement runs

Run number	*Description of load*	*System size*	
		512K	*1024K*
1	FORTRAN, prestored, single terminal, F01 (89 cards)		•
2	FORTRAN, prestored, single terminal, F05 (145 cards)	•	•
2A	Same as 2, run on prerelease TSS/360 1.2		•
3	Assembler, prestored, single terminal, K53 (101 cards)		•
4	Assembler, prestored, single terminal, K55 (401 cards)		•
5	Assembler, prestored, single terminal, K56 (802 cards)	•	•
6	DATA command, single terminal, 11 statements	•	•
7	FORTRAN syntax edit, single terminal, 20 statements		•
8	Assembler syntax edit, single terminal, 11 statements		•
9	Assembler, batch, 2 tasks plus BULKIO, K53		•
10	FORTRAN, batch, 2 tasks plus BULKIO, F01		•
11	FORTRAN syntax edit, 12 terminals, 20 statements		•
12	DATA command, 12 terminals, 42 statements	•	•
13	FORTRAN syntax edit; Assembler, syntax edit; DATA command, 42 statements; 4 terminals each	•	•
13A	Same as 13, run on prerelease TSS/360 1.2		•
14	Same as 13, except for addition of one batch FORTRAN task		•

Table 2. Parameter settings for measurements

Parameter	*Setting*
Operational cycle time	2 seconds
Quantum time	250 milliseconds
Maximum quanta per time-slice	1
Third-level inclusion delta	250 milliseconds
Third-level delta time	25 milliseconds
Third-level inclusion ratio	100 percent
Number of time-slices to purge shared pages	5
Low-core high	10 pages
Low-core low	2 pages
Maximum task size	50 pages
Initial page estimate	15 pages
Drum threshold	15 pages

Table 3. Calibration results for 1024K (four-core-box) system

Run number	Interval measured	Model 100 Difference (percent)	Model 100 + patches Difference (percent)
1	LOGON to LOGOFF	+22.0	
2	LOGON to LOGOFF	+18.2	
2A	LOGON to LOGOFF	+ 5.0	+ 1.8
3	LOGON to LOGOFF	+ 5.2	+ 7.7
4	LOGON to LOGOFF	+ 4.5	+ 3.6
5	LOGON to LOGOFF	+ 7.8	+ 1.8
6	LOGON to LOGOFF response	+ 1.1 –64.9	+ 5.3 –83.3
7	LOGON to LOGOFF response	+13.0 –59.0	
8	LOGON to LOGOFF response	+ 4.1 –26.0	+15.1 –18.0
9	first LOGON to last LOGOFF	– 4.9	+ 2.3
10	first LOGON to last LOGOFF	+21.7	
11	first LOGON to last LOGOFF average, LOGON to LOGOFF response	+20.1 +26.5 + 5.4	
12	first LOGON to last LOGOFF average, LOGON to LOGOFF response	+ 9.0 +16.5 +71.1	+ 4.7 + 9.7 +28.9
13	first LOGON to last LOGOFF average, LOGON to LOGOFF average FORTRAN, LOGON to LOGOFF average DATA, LOGON to LOGOFF average ASSEMBLER, LOGON to LOGOFF response, FORTRAN response, DATA response, ASSEMBLER	+16.7 +27.7 +28.3 +24.0 +33.0 +60.8 +64.4 +58.9	
13A	first LOGON to last LOGOFF average, LOGON to LOGOFF average FORTRAN, LOGON to LOGOFF average DATA, LOGON to LOGOFF average ASSEMBLER, LOGON to LOGOFF response, FORTRAN response, DATA response, ASSEMBLER		+ 4.7 + 8.0 + 6.3 + 6.9 +12.1 – 9.8 + 7.4 0.0
14	first LOGON to last LOGOFF average, LOGON to LOGOFF average FORTRAN, LOGON to LOGOFF average DATA, LOGON to LOGOFF average ASSEMBLER, LOGON to LOGOFF response, FORTRAN response, DATA response, ASSEMBLER batch		+ 2.6 + 6.7 + 4.7 + 9.1 + 5.5 –22.9 + 9.1 –21.7 +47.1

Table 4. Calibration results for 512K (two-core box) system

Run number	*Interval Measured*	*Model 100* *Difference (percent)*	*Model 100 + patches* *Difference (percent)*
2	LOGON to LOGOFF	+24.0	
5	LOGON to LOGOFF	+13.1	–12.1
6	LOGON to LOGOFF response	+20.6 –57.0	+15.2 –65.3
12	first LOGON to last LOGOFF average, LOGON to LOGOFF response	+16.8 +24.0 +54.9	+17.4 +23.0 +40.8
13	first LOGON to last LOGOFF average, LOGON to LOGOFF average FORTRAN, LOGON to LOGOFF average DATA, LOGON to LOGOFF average ASSEMBLER, LOGON to LOGOFF response, FORTRAN response, DATA response, ASSEMBLER		– 8.9 – 4.2 + 0.9 – 8.9 – 3.6 –82.5 –82.0 –57.7