ON OBTAINING TRACE LOAD AND PERFORMANCE DATA IN A UNIVERSITY ENVIRONMENT

Charles M. Shub

University of Wyoming

ABSTRACT

One of the problems in validating a simulation model of a computer system is that of obtaining detailed load and performance data from the computer system being modeled. The purpose of this paper is to describe the methods used to obtain appropriate load and performance data for a case study.

A brief summary of the simulation model, the actual system, and requirements for the load and performance data are given. Next an analysis of the tools available or procurable to perform the measurements is presented. Consideration is given to the problem of level of detail of measurement. Analysis of the effects of the compromises between the optimal measurements and actual measurements is presented. Then the measurement process is described. Next, the analysis of the data and its conversion into a form suitable for input to the simulation is described. This description of the data reduction phase describes and justifies the simplifying assumptions. The validation of the model using this method is presented.

Coupled with the description of this project is commentary on the task of educating computation center staff as to the need for modeling their system, the practicality of the task, and the results. A major theme of the paper is the effect which the constraints imposed by computation center policy had on the effectiveness of the project.

INTRODUCTION

Given a simulation model of a computer system each additional simulation experiment performed with the model can be thought of as a mechanism for increasing confidence in the validity of the model. This is despite the fact that a long line of validation tests have already been performed. Recent developments indicate that the technique of trace driven modeling (3, 7) can provide for further validation of such systems. Basically, the technique involves four phases. The first is obtaining a trace of the operation of the system being modeled while the system is running under a workload which may be considered typical or in some sense representative of the load which is usually found present on the system. The second phase involves reducing the trace data to a form suitable for input to the simulation model. The third phase involves running the simulation model with the reduced trace data as its stimulus. Finally, the performance characteristics of the simulation model are compared with the performance characteristics of the actual system. In this procedure, the last two steps are relatively straight forward. The problems arise in the first two steps. There are many problems involved, including such matters as what trace information is available, the effect of collecting the trace data, the method of trace data collection, the approximations made in the data reduction phase, and the type of trace data desired. Of course, if one is going to perform a simulation experiment involving trace driven modeling, the facilities and equipment to perform the tracing must be available.

This paper, then, is concerned with addressing these points by describing an effect toward trace driven modeling. The effort was undertaken at a Midwestern university as part of a project concerned with attempting to identify and isolate important parameters related to the effect of scheduling overhead in large scale computing systems. The effort was not an unqualified success. Therefore, this paper attempts to describe the process to indicate exactly where in the effort problems arose, the nature of the problems, and how they were coped with. This is done so that others contemplating a similar effort will have the knowledge to be able to avoid some of the delays and forced compromises and thus provide better results. Considerable attention is paid to the planning of the trace effort, because it is felt that the planning was sound. Had everything gone according to plan, the end results would have been better. The system will be briefly described. Then the type of trace data necessary for driving the model will be described. Next the question of obtaining the trace data will be addressed. After describing the choice of methods for trace data collection, the actual collecting process will be described. Then the data reduction phase of the project will be described. The actual experiment and some of the approximations in the experiment will be described. The validation of the model will be presented in the context of the experiment. Finally, conclusions are drawn and recommendations made.

THE SYSTEM

The system being modeled is a Honeywell 635 with 200K words of core, two card readers, two line printers, six tape drives with a single controller, two channels of type 167 discs, one channel of type 180 discs, and a Datanet 30 communications processor which would handle up to thirty remote terminals. The operating system was General Comprehensive Operating System (GCOS) III (Software Development Letter) 6.2. The machine was being used for the full gamut of a typical university's computing activities including administrative data processing, research, and student programming. The machine handled batch, remote batch, and time sharing at a load of approximately 1600 jobs per day. Batch turn around was in the neighborhood of six hours.

THE MODEL

The model being used is the author's Stable Time Independent Queue model which is concerned with measuring scheduling overhead in computing systems (2). The SIMSCRIPT I.5 implementation of this model is parameterized with respect to the hardware configuration of the system, device characteristics, system scheduler timing, queueing disciplines, user resource space and usage requirements, and user sequencing of requests for quanta of time on the various resources available. The output of the simulation reflects queue lengths, resource utilization statistics which include idle, busy, and overhead times, counts and timings of scheduling operations, timings of user flow through the system, and user waiting statistics. The model operates at the level of detail of the system scheduler, or dispatcher, and operates at a speed-up factor of about 3 to 1. The model can accept either trace data directly or approximate load information supplied by giving distributions of the various user parameters.

ALTERNATIVES FOR OBTAINING INPUT PARAMETERS

The hardware configuration parameters required for the simulation are rather coarse and consist of the number of devices of various type and the amount of memory available. In addition, information is needed as to seek and latency time, tape start-up time and similar properties of the hardware. These data can be obtained rather easily from equipment specification sheets and computer center propaganda. Similarly, the information as to scheduling disciplines is easily obtainable, usually through a conversation with the senior systems analyst or from a listing of the operating system code.

There remains, then, three tasks, namely obtaining scheduler operation timing data, obtaining user profiles, and obtaining performance statistics from the actual system to compare with the simulation output. There are several possible methods which could be used to obtain either this information, or an approximation to it. The coarsest level of information, which is computer center data on number of jobs processed per day and estimates of turn around time, is valueless for this study.

The first refinement involves the use of an available software monitor such as the General Electric System Resource Monitor (1) which provides two types of information. The first is an accounting of each activity or job step giving activity type, start time, memory usage, elapsed time, processor time, peripheral connect time, and number of data sets. This information is also averaged for each of nineteen types of activities. The individual statistics could be used as trace data, or the averages could be used to develop distributions of user profiles. The resource monitor also provides a modicum of performance data. The monitor also provides values for processor overhead percentage, idle percentage, and slave or program percentage. It also provides statistics on memory usage as well as connect and usage statistics for peripherals. It should be pointed out that the accuracy of the performance statistics are dependent upon the frequency with which the monitor samples the system. Experimentation has shown that sampling more often than every three seconds seriously degrades system performance. Thus the performance statistics are quite coarse in nature. However, the user profile information is accurate.

This system resource monitor operates in two phases. The first phase is the probe phase which is a "privileged slave" program and is supposedly independent of the operating system. The "privilege" is the ability of the program to access tables and data within the operating system monitor. This data is accessed every probe period and assembled in a buffer in the probe area of memory. When the buffer is full, the information is written out on the accounting tape. Thus the probe phase does nothing more than collect the performance statistics and write them on the accounting file. Concurrently, the normal accounting routine is collecting the load data which will be used later for billing purposes. This load information is written out on the accounting file as well. Since this accounting for charging purposes goes on continually, there can be a problem relative to obtaining just that load data relative to the period during which the probe is running.

The second phase of the software monitor is the analysis or data reduction phase. This proprietary program can be run at will and requires nothing more than an accounting file as an input. The operation of this program is to read the accounting file and produce meaningful trace information and summaries from the accounting data as well as to provide performance data from the information provided by the probe portion.

The next refinement considered was a locally designed software probe and/or data reduction package. Initially, the local software probe was rejected as being impractical because it was felt that the existing probe did a more than adequate job of obtaining what performance information was actually available. Attempts to obtain more performance data would have meant redesign of a major portion of the operating system.

The considerations involved in initially rejecting the development of a local monitor analysis package were more complex. The simple modification to the output phase of the analysis package to provide the accounting information or trace load data in a form suitable for direct input to the simulation was planned. The second modification considered was that of attempting to provide meaningful dynamic performance information rather than mere summaries. This was planned for at a later stage of the project. Concurrent with this proposed change in the analysis package was a modification to the simulation model to allow for more dynamic performance displays than were currently available. The major reason for not implementing these modifications at the earliest stage of the project was the strong possibility that a hardware probe would be available.

The final refinement, and this applies to the measurement of system performance and scheduler operation timing, is the use of a hardware monitor to collect the desired statistics without degrading the system performance. With such a device all desired parameters of interest could be measured.

The optimal solution to the problem of obtaining input and performance data for the simulation was therefore quite clearcut. The system resource monitor would provide user profile information, and a hardware monitor would provide performance and scheduler timing information. However, this was not the way things occurred. The first problem was that funding to obtain a hardware monitor was unavailable within the time constraints of the project. This meant that the coarser performance data provided by the system resource monitor would be the best data available. Also it meant that there was no way to measure the timing of scheduling operations.

OBTAINING SIMULATION INPUT PARAMETERS

The information as to the timing of scheduling operations was obtained by the rather straightforward but painful method of procuring the code listings of that portion of the operating system and counting instructions. Timings were then derived from the instruction frequency and the known instruction timings. The performance data as well as the user profiles was to be obtained from the system resource monitor. Due to a four-month delay for modification of the system resource monitor to make it compatible with the latest version of the operating system, serious time constraints were placed on the remainder of the project. Therefore, rather than develop software to prepare trace data from the monitor output tape, the approach of using the averages from the monitor analysis package and converting them to distributions of user profiles was necessary. The inaccuracy this introduced was that the load would average out to be the same as if the trace data had been used, but the characteristics of individual activities generated might not have exactly the same arrival pattern. However, there is literature that indicates this is a good approximation (5).

THE EXPERIMENT

The system resource monitor was run for a period of one hour with performance probes made every 15 seconds. The following method was used to insure that the accounting data would reflect the period of time that the probe was active. The system was stopped. A new accounting tape was mounted. The system was started. The probe was started. An hour later, the probe was stopped. Further input to the system was stopped, and the work in the system was allowed to run to completion. Then the system was stopped, and the accounting tape was removed. The monitor analysis software was used to provide the load and performance summaries. The load data summary was manually converted for input to the simulation. The simulation model was then run. The output of the simulation model was then compared with the performance data generated by the system resource monitor.

APPROXIMATIONS IN THE EXPERIMENT

There were several approximations made in this experiment which apparently did not affect the results. The first approximation was that of using virtual users rather than the actual users to drive the simulation. In addition to the justification for this in the literature (5), a detailed manual examination of the trace data provided by the analysis package, and the event trace provided by the simulation model indicated that the approximation was a good one.

A second approximation involved a possible inaccuracy in the summary data provided by the accounting routines. The accounting information would not be put on the accounting file until that activity was completed. It was felt that any effects of this were minimized by the procedure used in running the experiment. This assured that all activities which had started while the probe was in operation were recorded on the accounting file. It also meant that since there was a small amount of work done after the probe was turned off, the user profile input to the simulation indicated slightly more work done than was actually done.

A third approximation involved was that the simulation did not account for the possibility of memory compaction. The memory map output of the analysis package indicated that no compaction had occurred during the experiment.

An approximation to seek and latency delays on the discs was made. The disc channels on the actual system allow for overlapping seeks on one drive with reads on another drive. This was accounted for by adjusting the distribution of device delay parameters to reflect the effect of the overlap. Also, on the actual system, an activity would request a particular tape handler. The simulation, on the other hand, assigned the request for disc or tape transfer to any available device on the channel. With the low utilization rates for the peripherals, this was felt to be a valid approximation which did not affect the results of the experiment.

In addition, there was an approximation relative to input and output insofar as the card reader and line printer were not accounted for in the simulation. The justification for this was that the pooling on input and output is treated as slave programs and not accounted for separately within the software monitor. In addition, the load on the system of driving the card reader and printer was considered negligible.

VALIDATION

The Simulation Model has already been extensively tested and validated (2,8). Some of the techniques which were used in this earlier validation are described below. An examination of voluminous program trace information ensured that events were occurring in the proper sequence. Analysis of program output indicated that random variates were being properly generated. Several runs were made with the simulation parameterized to reflect M/M/1 and M/G/1 systems (3) and the results of the simulation were statistically identical to those predicted by theory. An analytical model reflecting the effect of introducting scheduling delays in the form of a non-zero time to perform scheduling operations was compared with simulation results when the simulation model was parameterized to reflect these delays. Again the results were statistically identical (8). Therefore, it has been concluded that the model itself is valid model of a computer system, and the simulation program is a valid implementation of the model. It remains, then, to either attempt to explain any discrepancies between the model and the monitor or to reject the conclusion that the model is indeed valid.

Figure 1 provides some comparison figures between the monitor and the simulation program. The first column indicates the parameter of interest. The second and third columns indicate the monitor and simulation observed values. For rows involving percentages, the fourth column gives the Z values for testing the hypothesis that the theoretical monitor percentages are equal to the theoretical simulation percentages (6). The final column gives the sample size for the simulation percentages. The sample sizes for the monitor percentages are not given in the figure since they are all of size 230. For the Z test, the difference would be significantly different at the = .01 level if the absolute value of the Z statistic is greater than 2.55.

For the remaining rows, which involve counts, the fourth column gives the relative difference between the two figures. The value in Figure 1 for scheduler overhead percentage in the simulation column was obtained by subtracting the sum of core allocation time percentage and peripheral allocation time percentage from the observed overhead value of 11.72%. This was done because in the simulation program, the scheduler overhead percentage represents the total time for both scheduling and performing peripheral and core allocations.

Considering the known error in the monitor values with respect to the data concerning the tape channel (it produced negative utilization), the only corresponding values which are significantly different are the number of connects for the disc 180 units and the slave processor percentages. Careful consideration suggests that there may be valid explanation for this in light of the simplifications made in the simulation. The disparity in slave processor percentages can perhaps be best explained by noting that the time sharing system provides a real load on the monitored system, in fact that 10% of the load, and the absence of the load of the time sharing system in the simulation may be partly responsible for this. It is also true that in the real system, the input and output media conversion service programs are run in slave mode and thus may account for additional disparity. The connects disparity can be explained in light of the simplifying assumptions made in the model, especially the assumption that in the simulation all the transfers involve one system standard 320 word block at a time, while this is not necessarily true on the system. Also, the simulation model does not account for slave service area I/O and 4/5 of the total disc utilization was operating system service utilization as opposed to user utilization. Thus the simulation value, which is slightly less than 1/5 the monitor value can be assumed correct. In fact, the difference on this basis is only 29%. Thus it is felt that in view of the approximations made in the model, the results of the simulation are not in conflict with those of the monitor.

CONCLUSIONS AND RECOMMENDATIONS

The experiment was a qualified success. The results indicated that the model performed acceptably close to the actual system. This, coupled with several other validation and verification techniques described elsewhere (2,6) increases the degree of confidence in both the model and the simulation program. It should be noted that a change is being planned for the simulation model to allow for a more detailed validation procedure which will include not only the comparison of percentages and means, but also to look at distributions. These techniques have been discussed recently (9).

The recommendations concern steps which must be taken if future experiments in modeling systems and collecting trace data in a university environment are to be successful. First, there must be a commitment to the project at the highest level. This should include not only the commitment and support of those sponsoring the experiment (typically an academic department), but also the guarantee of the fullest level of cooperation and support from those responsible for the equipment (typically a division of computer services). This commitment should be more than a vague "this sounds like a good idea, why not go ahead with it." The possibilities of requiring commitments in the form of work time by the computer center staff should be recognized. The authorization and support must be agreed upon at the highest levels and transmitted down to the actual people doing the job. There is a need for a greater awareness of the desirability of this sort of effort, not only from the viewpoint of it being interesting in a pure research sense, but also from the viewpoint that it can lead to improved performance, higher capacity, and increased user satisfaction, perhaps even at lower costs.

REFERENCES

1. ...., System Resource Monitor - GECOS III, General Electric Report No. ASL600J1.401, Phoenix, Arizona.

2. Bulgren, William G. and Shub, Charles M., "A Stable Time Independent Queue Model For Studying the Effects of Overhead in Computing Systems," Proceedings of 1975 Summer Computer Simulation Conference, San Francisco, California, July, 1975, pp. 639-642.

3. Chang, P. S., "Trace Driven System Modeling," I.B.M. Systems Journal 8, 4 (April, 1968), pp. 280-289.

4. Coffman, Edward G., Jr., and Denning, Peter J., Operating Systems Theory, Prentice Hall, Englewood Cliffs, New Jersey, 1973.

5. Denning, Peter J., "A Statistical Model for Console Behavior in Multiuser Computers," Communications of the A.C.M. 11, 9 (September, 1968), p. 605.

6. Gordon, Geoffrey, System Simulation, prentice Hall, Englewood Cliffs, N.J., 1969.

7. Sherman, Stephen, Baskett III, Forest, and Browne, J. C., "Trace Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System," Communications of the A.C.M. 15, 12 (December, 1972), pp.1063-1069.

8. Shub, Charles M., "Simulation Studies on the Effect of Overhead In Computing Systems," Proceedings of 9th Annual Simulation Symposium, Tampa, Florida, March, 1976.