SIMULATION OF COMPUTER SYSTEMS: LOOKING BACK

Organizer and Moderator:

Lawrence L. Rose
Battelle Columbus Laboratories
Columbus, Ohio 43201

Panelists:

E.B. Zucker
Bell Laboratories
Denver, Colorado 80234

Udo Pooch
Industrial Engineering Department
Texas A&M
College Station, Texas 77843

Peter L. Haigh
NCR
Dayton, Ohio 45479

Edward R. Comer/Ronald M. Huhn
Harris Corporation
Melbourne, Florida 32901

INTRODUCTION

The past two decades have seen a phenomenal growth in the area of computer systems simulation. As hardware and software systems increase in size, cost and complexity, accurate and timely models become even more vital to the decision maker. Significant research contributions have been made in simulation languages and models in an effort to meet the needs of society. This panel discussion will focus upon the area of computer systems simulation to assess the current state of the art, to learn from past successes and failures, and to predict how we can better model computer systems in the future.

The panelists respond herein to four questions, in an effort to provide a framework for discussion at the panel session and to give the prospective attendees an idea of the substance for the panel discussion. The first question tries to delineate the differences between simulating computer systems and, for example, transportation systems. Given that differences exist, we go on to consider if this application area deserves specialized language constructs.

Statistical analysis forms an integral part of simulation modeling; its role in computer systems simulation is discussed. Lastly, the generality of computer simulation software is considered: how can the modules of a particular model or methodology be applied to other computer systems simulations?

The panelists have participated in significant research, both academic and industrial, over the past years in computer systems simulation. Their views on this subject, as delineated herein, are diverse. Yet there is a consensus that more sophisticated tools and methodologies are requisite to the successful modeling of computer systems.

Fruitful areas of research in computer systems simulation that will be considered include: (1) model software construction that enables multiple levels of detail, (2) model execution heuristics to enable nanosecond and hourly events to be modeled without undue computer cost, and (3) model analysis methodologies to ensure sound conclusions.

Proceedings of the 1982
Winter Simulation Conference
Highland * Chao * Madrigal, Editors

82CH1844-0/82/0000-0698 $00.75 © 1982 IEEE

WHAT IS DIFFERENT ABOUT COMPUTER SYSTEMS SIMULATION?

HAIGH: Simulation models of computer systems are not fundamentally different from simulation models of other systems. A special characteristic of computer systems, however, which differentiates them from certain other systems is that decisions made by computer hardware and software will always be the same, given the same conditions. This cannot be said of systems involving people, for example, where judgment is less predictable, or of the economy, where the importance of certain factors on spending behavior is not known, or of biological systems, where such things as growth, reproduction, and survival rates depend upon complex probabilities.

In theory, it should be possible to determine (and thus, to model) exactly how a computer system will behave under a specific set of circumstances. In fact, for a properly validated computer system simulation model, inaccuracies in model input data may be more significant than inaccuracies in the model.

Another aspect of computer systems is that they are inherently self regulating. Only so much work can be done in a given amount of time. This establishes some convenient boundary conditions on a model. The temperature in a nuclear reactor, on the other hand, may rise and cause unpredictable instability modes which cannot be verified for obvious reasons.

POOCH: Simulation is a popular approach to the solution of computer system models because of its generality and because system details can be represented very accurately. There are however, a number of significant problems with simulation: (1) Constructing a simulation program and verifying that the program is logically correct and represents the simulated system; (2) It may be very difficult to characterize the system workloads and system features which have the greatest impact on performance.

The workload for the simulation model has most often been described by means of probability distributions and thus may not include interdependencies of the workload characteristics. This problem can be overcome by using trace driven modeling. When simulating a virtual memory system a complete page trace is impractical, and other representations of the paging activity is required. This representation can include distance strings in stack replacement algorithms, lifetime functions, semi-Markov characterization, macro-instructions of Boksenbaum, the page survival index, paging index, and the global LRU analysis.

ZUCKER: The typical computer system is a complex queuing network which can be modeled at many levels of detail. Modeled at a very high level, the complexity of the system is washed out. Here we are concerned only with requests for the cpu or I/O devices, and are interested in deriving utilizations, determining bottlenecks, and perhaps calculating average delays. A variety of analytical packages and techniques, as well as simulation, are applicable here.

A low level, detailed simulation is required when we are interested in response time distributions, especially when they are compounded by such things as dependent queuing disciplines, priority scheduling, feedback loops, intricate itineraries through the network, and mixes of diverse job types. Computer models will also have interdependent resources, causing problems such as an idle server which is blocked from accepting the next job in its queue, and limited resources which can be seized or preempted from many points in the network. Resources can also be nested, so that one transaction can hold many resources simultaneously.

The clock of a detailed computer system simulation will be incremented in units of milliseconds or even microseconds, and the total number of events can be a million or more.

A response time is the time it takes to complete some subset of a total task. The capacity of the computer system is determined by the response time distributions of selected criteria as opposed to the absolute capacity of the system. It is usually the tail of the distribution which is of most interest.

Simulation is also used to engineer queue sizes. Here the queue length distribution is required, and again, the tail of the distribution is the most important.

COMER: The proliferation of small but powerful computers has resulted in a widespread need for computer systems analysis. Even relatively simple systems are plagued with complex performance problems. Sophisticated analysis techniques such as simulation are required to correctly deal with these complexities. The same system complexity that requires this analysis results in complex simulations which are often too costly or not timely enough.

(continued)

WHAT LANGUAGE FEATURES ARE REQUISITE FOR COMPUTER SYSTEMS SIMULATION?

HAIGH: Computer system simulation models have been implemented using general purpose languages, such as FORTRAN and PL/1. Indeed, any computer language may be used. Obvious advantages are gained by using a simulation language. Even more may be gained by using languages with certain features when modeling computer systems. Several such features readily come to mind.

Process pre-emption. In computer systems, interrupts occur, causing suspension of processing. Furthermore, tasks in execution may become pre-empted by higher priority tasks. A language feature which permits this process preemption while freeing the programmer from the related housekeeping is extremely useful in computer system models. GPSS and SLAM, for example, provide PREEMPT statements which need no other attention by the programmer.

Priority. Processes in computer systems contend for resources and are granted service on a priority basis. A language feature which permits processes or events to be automatically ordered on the current and future events chains according to a priority which is dynamically assigned during model execution is essential for modeling computer systems.

Indirect addressing. In computer systems there are likely to be many similar processes. A feature which permits the programmer to address model entities by developing a pointer or by indexing into a list is a valuable language feature. Without this capability, similar processes must be coded separately, greatly increasing the size of the source program.

ZUCKER: A single job can make hundreds of visits to many nodes in some predetermined order, so some method for specifying this itinerary through the network is required. Many subsequences of events are used frequently or repeatedly, so nesting of itineraries should also be possible. Probabilistic branching is necessary for the high level simulation models.

This also implies that the flow of a transaction through the network should be managed by the language in a process interaction orientation (as in SLAM or GPSS), rather than modeled by the analyst in next-event style.

The trace capability of the language should be highly flexible or user defined. The capability to turn a trace on or off at selected times for selected transactions as they pass through selected nodes is necessary. One should be able to easily verify that a transactions is executing its itinerary properly.

To prevent the inadvertent loss of the tail end of a distribution, the histogram function should display some "intelligence" in how it stores and reports values. It is usually the case that these tail values are critical to the system analysis.

Nested servers, or the ability to seize multiple resources, are necessary and should be available from any point in the network. Another requirement is the ability to preempt an entity that is holding multiple resources.

For the high level model, the Processor Sharing queuing discipline should be available, and again, preemptive queuing disciplines are a must.

COMER: The objectives of computer system modeling and simulation efforts change as the development progresses. There is the need to develop the simulation model with increasing level of detail as the system evolves topdown. Depending on the stage in the development lifecycle where simulation is applied, the objectives could include:

- Requirements definition
- Design feasibility
- Make/buy decisions
- Vendor selection
- System architecture studies
- Hardware/software tradeoffs
- Detailed design decisions
- Operations research
- Expansion/enhancment planning
- Marketing

Often the simulation analyst must support a number of these objectives over time. To do this efficiently, the model must be structured to allow progressive elaboration of detail. This technique, termed hierarchical modeling, is essential to the cost effective application of simulation to the design of computer systems. Currently no simulation language readily supports the use of hierarchical modeling. It is possible to implement hierarchical simulations using a carefully planned simulation architecture; this however requires a level of sophistication for the analyst.

WHAT IS THE ROLE OF STATISTICAL ANALYSIS (E.G., VARIANCE REDUCTION AND REGRESSION) IN LARGE MODELS?

HAIGH: Statistics are important in all simulation studies, for it is through the statistics collected during simulation runs that knowledge about the behavior of the simuland is gained. Statistical analysis of model data, however, is more in the realm of the model builder and system designer than that of the decision maker. Analysis of variance, regression analysis, and curve fitting are techniques which can be appropriately applied by knowledgeable analysts for model verification and validation and for system design analysis.

There is often a problem, however, in presenting raw statistics, or even a thorough statistical analysis, to decision makers. The problem is that, to those not familiar with statistics terminology, the significance of certain information may be overlooked. This can lead to unjustified or, worse, incorrect conclusions. This is not meant to infer that decision makers are not intelligent, but rather that they are usually not mathematicians. Model results usually will require interpretation and explanation.

POOCH: Running a simulation program should be considered a statistical experiment, and thus the obtained performance measures should be analyzed using standard statistical methods. Anytime a system is characterized by probability distributions it is necessary to have generators to produce samples from these distributions. Nearly all of these generators for general distributions require a uniform generator which must be designed by careful use of number theory and then rigorously statistically testing its behavior for the desired properties.

The results of the probabilistic simulation may not be accurate estimates of model performance measures. To careful analyze this output such techniques as the Regenerative Method for confidence intervals, stopping rules, extensions to response time distributions, variance reduction and improved estimators have been developed.

A reduction in the number of simulations required to cover a parameter space can be obtained by an appropriate design of experiments. This approach is also useful for validating the model itself.

ZUCKER: The time to reach equilibrium can vary greatly with changes made only to the arrival rate of jobs to the system. The language should be devised so that the start of statistics accumulation can be keyed to various, dynamic system parameters, such as the mean occupancy of a particular server.

Two types of curve fitting would be useful. One method of analysis is to plot some parameter (e.g., mean response time) versus the offered load to the system. This requires simple regression analysis and would be useful since each point requires a separate run of the model.

Another method would be to fit an analytical distribution to a histogram. If the parameters of the distribution could be determined as a function of the inputs to the model, then the simulation could be put to rest (provided that distribution is all we wanted from the simulation.) This would also require a goodness of fit test.

These features would also be desirable in a general purpose, discrete simulation language. In this case the needs of a computer simulation language are not significantly different.

COMER: Once a conceptual model is formulated, determination of parameter values often remains to be a sizeable task. The analyst may be faced with requiring information which the design team has not yet generated. Here his stopwatch and clipboard do not assist in parameterizing the hardware and software. Often the desired delay can only be accurately extracted by using specialized (and expensive) hardware monitors. Since computers are actually deterministic machines, the simplification of variable parameters which are in fact deterministic into probabilistic distributions may require extensive analysis.

CAN COMMON MODULES BE IDENTIFIED--IS THERE SOME GENERALITY TO COMPUTER SYSTEMS MODELS?

HAIGH: The answer to this question is a resounding YES! This assumes, of course, that models of several different computer systems are to be developed and, furthermore, that the models are to require similar levels of detail. Given this, then one can define common processes, such as transfers between the central processor/main memory and peripheral devices (i.e., I/O processes), communication processes, link protocols, file management, and database access methods. Each of these processes can be broken down into event strings containing processing, logic, queue handling, access characteristics, and resource contention. I/O devices, processor, operating systems, protocols, files and other system attributes may be parameterized and stored as library modules for building models of specific system configurations. The modeler need only program the applications to be simulated and the model stimuli to complete the system model. Re-invention of the parameter driven library modules is thus avoided.

POOCH: Detailed simulation programs require large run times to provide accurate performance estimates. One factor in the computational expense of simulation is the disparity in event rates in different parts of the system. This results in very long simulations so as to accommodate the low event rates. Another factor is the time spent inserting/deleting events to/from the event set.

ZUCKER: Memory allocation and swapping decisions in real computer systems are based on the size of a process, the amount of processing received so far, how long it has been waiting, and the amount of memory available. Common modules to handle this aspect of computer system simulation are needed. Standard output should include statistics on swapping frequency and memory utilization.

The modeling of disk I/O is another area which would benefit from common modules. The frequency at which particular data is accessed and the size of the buffer cache affect the probability of a cache hit. The type and size of the file system or database determines the number of disk accesses necessary. In a detailed study it is not enough to know that the disk is a bottleneck. We need to know how it is spending its time, and at whose request.

Interprocessor communication in a multiprocessor requires seizing and/or preempting at least two processors and a bus simultaneously. This should be attainable in a model in a single statement.

Interprocessor communication in a network is much more complex. In fact, a special simulation language just for computer networks may be appropriate. This area of simulation study will grow as network technology grows.

COMER: Simulation of computer systems is indeed a tricky business. The modeler must deal with numerous complexities and complications in a manner which provides timely input to a constantly moving lifecycle. Because the simulations are highly application oriented the analyst is hard pressed to develop any sizeable repertoire of common routines or model segments. There has been success in the development of highly parameterized hierarchical simulations which provide some level of flexibility. More work however is needed to provide the computer systems analyst with better tools, techniques and languages.

WHAT IS DIFFERENT ABOUT COMPUTER SYSTEMS SIMULATION?

COMER (cont'd): Computer system simulations must deal with complexities rarely matched in other applications. Necessarily the simulation must model not only the computer system itself, but also the application. Since the essence of the application may exist as thousands of lines of software, modeling the application is often the most difficult of these tasks. Simulations of this kind range in complexity from simple queuing models to detailed representations approaching emulation. The analyst is faced with the difficult decision as to the appropriate level of detail for the model.

Computer system models must address the actions of the hardware, software and human operators. Often it is difficult to separate these elements in the model. Consider for example the queuing in a multitasking system. The queuing rules of a model for this would represent the combined interactions of both hardware and software. Task priorities are established in software. Time slices are triggered by a hardware clock according to rules set in software. Interrupts which preempt processing are initiated with hardware logic. Consider also the case where model decision logic reflects the combined interaction of the human user and the interactive software. To make this situation worse, there are often few who can guide the modeler in these hardware/software-operator issues.

The flexibility provided to the systems designer in software creates a nightmare for the modeler. Decision logic is sometimes so complex that the model can do little more than duplicate it. Network routing algorithms and communication protocols are an example of this point. Software often creates new and unusual queuing rules which are not supported by the simulation language and therefore must be specially handled.