1977

Computer Systems Modeling, Measurement and Evaluation: Project Report 1976-1977

Peter J. Denning

Saul Rosen

Herbert D. Schwetman

Report Number:
77-245

COMPUTER SYSTEMS MODELING, MEASUREMENT AND EVALUATION:

Project Report 1976-77

Peter J. Denning
Computer Science Department
Purdue University
West Lafayette, Indiana 47907

Saul Rosen
Computing Center
Purdue University
West Lafayette, Indiana 47907

Herbert D. Schwetman
Computer Science Department
Purdue University
West Lafayette, Indiana 47907

CSD-TR 245

October 1977

This report covers work done under National Science Foundation Grant GJ-41289 for the period September 1976 through August 1977.

TABLE OF CONTENTS

SUMMARY 1

NOTABLE ACHIEVEMENTS 2

PERSONNEL 3

MODELING PROJECTS 5

- Operational analysis of queueing networks 5
- Computational algorithms for queueing networks 7
- Operational analysis of renewal processes 8
- Approximations for load and response time distributions in time sharing systems 9
- Approximations for heavily loaded systems 11
- Hybrid simulations 11
- Program behavior and optimal multiprogramming 13
- Generalized working sets 15
- Linear Paging Phenomena 16
- MIN distance strings 17

EVALUATION PROJECTS 17

- Certifying queueing network evaluators 18
- System variability and model validation 18
- Workload compression 19
- Implementations of capability machines 20

MEASUREMENT PROJECTS 22

- New Approaches to Performance Data Reduction 23
- Measurement package upgrade 24

PUBLICATIONS & REPORTS 25

SUMMARY

This project, continuing research into computer system modeling, measurement, and evaluation, began at Purdue on 1st January 1974 and was renewed on 1st July 1976. This report summarizes our progress since summer 1976, the time of our last report.

The Projects

Models of computer systems continue to be the central theme of the supported research. We are moving toward our long-term goal of practical analysis techniques that guide the design, acquisition, and control of multiple-process computer systems. Queueing network models have been the framework. The modeling work is complemented by several evaluation projects that either apply models in practice or validate models against a real system. Our measurement work supports the other projects by providing a rich source of data from a large system (the Dual MACE operating system on the CDC 6500 computer in the Purdue University Computer Center, and its time sharing subsystem MESA).

Miscellaneous

a) In collaboration with J. Browne and J. Peterson of the University of Texas, Denning wrote an overview of the impact of operating systems research. It will appear as a book chapter in Research Directions in Software Technology [4].

b) Some of the grant's travel funds were used to send the three investigators and six graduate students to the International Symposium on Computer Performance Modeling, Measurement and Evaluation at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, on August 16-18, 1977. This proved to be of significant value to the students, who met and interacted with leading researchers.

NOTABLE ACHIEVEMENTS

In collaboration with J. Buzen, Denning has made significant progress in developing the operational theory of queueing networks; this theory derives the same basic results under much weaker, and more intuitive assumptions than the Markovian (stochastic) queueing network theory. Since practical validations substitute operational quantities for stochastic parameters, and compare stochastic steady-state probabilities against proportions of time, this theory helps explain why queueing network models have succeeded so well in validations. (See publications [6,7,8].)

Using a combination of modeling and experiment, Graham and Denning have shown that the multiprogramming memory management problem is solved in the following sense. A working set memory policy with a single, global value of window size can be tuned to cause system throughput to be within 5% to 10% of the best possible over all nonlookahead memory policies. (See publications [5,8,12,13].)

In collaboration with D. R. Slutz, Denning has extended the principles of working set memory management. The generalized working sets (GWS) include all one-parameter, demand-fetching policies whose resident sets satisfy an inclusion property for increasing values of the parameter. The time-window working set, the stack paging algorithms, and the VMIN optimal policy, are special cases. The procedures for measuring memory demands of GWS policies are very fast. (See publication [11].)

Balbo, Bruell, and Schwetman have extended the exact computational algorithms for evaluating queueing-network statistics to include open/closed networks, job classes, load-dependent servers, and different queueing disciplines. Their program permits jobs to change classes, not permitted by previous evaluator programs. (See publications [2,3].)

Schwetman has successfully speeded up simulations by replacing the central subsystem with a queueing network model. The hybrid simulators are 50 to 200 times faster than the original simulators, but give almost the same answers. (See publications [16,17].)

PERSONNEL

a. Co-principal Investigators

Peter Denning

Saul Rosen

Herb Schwetman

b. Graduate Research Assistants (September 1976 to September 1977)

Name	Status	Area of Research
Gianfranco Balbo	Working on Ph.D. Thesis	Approximate solutions to networks-of-queues; operational analysis of renewal processes
Steven Bruell	Completed second part of qualifying exam, working on Ph.D. Thesis	Solution techniques for closed networks-of-queues containing load dependent servers and multiple job classes, with applications
Ed Gehringer (not supported after August, 1977)	Working on Ph.D. Thesis	Evaluating the performance impacts of capability mechanisms in secure operating systems
G. Scott Graham (in absentia)	Completed Ph.D. Thesis, Dec. 1976	Thesis Title: "Program Behavior and Memory Management"
Kevin Kahn	Completed Ph.D. Thesis, Aug. 1976	Thesis Title: "Program Behavior and Load Dependent System Performance"
Robert Mead	Completed second part of qualifying exam, Working on Ph.D. Thesis	Workload Characterization for the Batch Portion of Multi-Programmed Systems: Techniques, Accuracy, Tradeoffs

Name	Status	Area of Research
Richard Simon	Working on Ph.D. Thesis	Models of Program Behavior in Paged Memories
Stephen Tolopka	Completed Qualifying Exam, Part 1	Approximate Solutions for Networks-of-queues under heavy load

c. Other Collaborators, not Supported by the Grant

i. With P. Denning:

J. P. Buzen, BGS Systems, Inc.
J. Leroudier, IRIA, France
R. R. Muntz, UCLA
D. Potier, IRIA, France
D. R. Slutz, IBM, San Jose
R. Suri, Harvard

ii. With S. Rosen

V. Abell, Purdue
J. Eykholt, Purdue

MODELING PROJECTS

Several modeling projects have been completed, notably the ones dealing with memory management problems such as multiprogramming level load control and generalized working sets. We have made significant progress in the operational theory of queueing networks, and will continue much of our effort in this direction. We have achieved some promising results with new approximations. A summary of the major projects is given below.

Operational Analysis of Queueing Networks

P. J. Denning

This work dates back to the summer of 1975, when P. Denning and J. Buzen began collaborating on "simple explanations of queueing networks." This work was motivated by our curiosity: why do the results of Markovian queueing network theory, whose many assumptions seem to be violated in practice, agree so well with actual measurements? We observed that analysts frequently substitute operational (directly measured) quantities for the stochastic parameters, and then compare the model's estimates of a steady-state probability with the proportion of time that a state is occupied. This led to the hypothesis that the steady-state balance equations of Markovian queueing network theory are in fact identities among operational quantities. Our investigations have confirmed this hypothesis.

Buzen's first public presentation of this hypothesis showed that familiar equations such as

Utilization = (throughput)*(service time)

Mean Queue Length = (throughput)*(Response Time)

are identities among operational definitions of the performance quantities.[1]

---

[1] J. P. Buzen, "Fundamental operational laws of computer system performance," Acta Informatica 7, 2 (1976).

Denning and Buzen [7] subsequently extended the concepts to queueing networks in the following way. Let $\underline{n}$ denote a state of the network, that is, a distribution of jobs among the various devices. The Markovian steady-state balance equations, which hold only in the limit of an infinitely long observation period, are relations among these stochastic quantities:

$p(\underline{n})$ steady state probability the system is in state $\underline{n}$,

S_i mean of the exponential service distribution of device i,

q_{ij} Markov chain transition probability, that a job selects device j next after device i.

Suppose that we replace these with operational quantities for a finite observation period,

$$p(\underline{n}) = \frac{\text{Total time during which state is } \underline{n}}{\text{Total observation time}}$$

$$S_i = \frac{\text{Total busy time of device i}}{\text{No. jobs processed by device i}},$$

$$q_{ij} = \frac{\text{No. transitions i to j}}{\text{No. jobs processed by device i}}.$$

Moreover, suppose that we assume

- Flow Balance: The number of entries to a given device (system state) is the same as the number of exits from that device (state) during the observation period.
- Homogeneity: The mean service time per request at a device, when the device is on line, is the same as would be measured by taking the device off line and subjecting it to a constant load.

Under these assumptions, the equations relating the operational $p(\underline{n})$ to the operational S_i and q_{ij} are identical in form to those relating the stochastic $p(\underline{n})$ to the stochastic S_i and q_{ij}. Therefore, all the results derived for Markovian queueing networks from these equations actually apply

under the weaker operational interpretation. Since the operational assumptions are closer to practice than the Markovian, the robustness of the models is easier to understand.

As part of the work we showed that most of the error between model predictions and real measurements arises from the homogeneity assumption, which is equivalent to an assumption of perfect decomposability. We also showed that the operational assumptions lead to a theory of queueing networks which is weaker than under the Markovian assumptions; for example, within the present operational assumptions we can study neither the transient behavior of a system, nor correlations among its states.

Denning and Buzen have written two tutorial papers on queueing networks developed in the operational approach; the one for Infotech [6] treats networks whose parameters are independent of load, while the one for Computing Surveys [8] treats the general case and gives more examples. Because the material requires no background in stochastic queueing theory, we have used it successfully in courses and professional seminars; indeed, students find stochastic queueing network theory much less difficult if they have studied operational queueing network theory first.

We plan to continue the research: extending the principles to other mathematical systems for modeling, developing operational approximations for difficult measures such as queue and response-time distributions, and studying the sensitivity of the results to the changes of parameters. Some of these projects have already been started; they are discussed below. To qualify as an extension of operational analysis, a model must for any given observation period, have measurable parameters, make testable predictions, and use assumptions which can be proved or disproved by measurement.

Computational Algorithms for Queueing Networks

H. D. Schwetman
G. Balbo
S. Bruell

Efficient algorithms for computing the performance measures of closed queueing networks were published by Buzen in 1973. Shortly thereafter Muntz and Wong and, independently, Chandy, Kobayshi, and Reiser extended these algorithms to networks with customer classes. While implementing our own versions of these algorithms, we found that the existing algorithms assumed

implicitly that customers do not change class. We extended the algorithms to permit customers to change class [2,3]. In so doing we discovered, to our surprise, that allowing class-switching reduces the computational complexity of the problem! We have also discovered that the algorithm can be sped up if we take account explicitly of service centers which are independent of load or which contain multiple independent servers.

We are currently applying our queueing network evaluator to the Purdue time sharing system (MESA). When it is validated, we will use the model to study the system under varying loads and different equipment configurations.

In attempting to use our queueing network evaluator on a problem with a large state space -- 150 customers, two load-dependent servers, 4 service centers, and 3 customer classes -- we encountered evidence of numerical instability [negative "probabilities"]. Reiser's program [IBM, Yorktown Heights] exhibited similar behavior on the same problem. The reason seems to be that, at certain steps, the algorithm computes the differences between very large numbers. We are experimenting with alternate algorithms that avoid subtractions altogether.

We believe that these computational algorithms cannot be used for systems with very large state spaces. They all compute quantities representing sums over very large numbers of states; when the number of terms in a sum approaches the reciprocal of the arithmetic precision of the machine, overflows and roundoffs will become severe. To solve such systems, we must employ approximation algorithms that group states into relatively small numbers of aggregates.

Operational Analysis of Renewal Processes

P. J. Denning
G. Balbo

Many phenomena encountered in performance studies are, intuitively, renewal processes. But the assumption of renewal theory, that we are observing the prefix of an infinite sequence of i.i.d. inter-event times, is not testable. Our goal is to reformulate the theory within the principles of operational analysis. To do this, we express the measure of the process as a recursion employing the empirical distribution of inter-event times. This recursion is identical in form to the renewal equation, but it is interpreted

relative to the operational assumptions. It would define a method of computing the measure of interest. We believe that the analog of the renewal theorem is a bound on the error between the true measure of the process and an approximation obtained by ignoring end effects in the interval between the final event and the end of the observation period.

The rudiments of these ideas have been used in computing working set measures for segment reference strings. In 1968 Denning[2] showed how to compute mean working set size and paging rate by assuming that each page is referenced by the program according to a renewal process; Opderbeck and Chu rediscovered the idea in 1975.[3] In 1974, Slutz and Traiger[4] showed a simple case: the results are computable by relaxing the renewal assumption and simply using the empirical interreference distributions for the given, finite observation period. Denning and Slutz have extended these ideas to the generalized working sets for segment reference strings [11]. This experience suggests strongly that the analysis technique can be extended to general renewal processes.

Approximations for Load and Response Time Distributions in Time Sharing Systems

P. J. Denning
G. Balbo

In applying the principle of decomposition to a time sharing system, an analyst begins by treating the central computing facility in isolation. He determines its job completion rate X(N) as if it were subjected to a constant load of N active jobs. If the central computer is connected to M time-sharing terminals of mean think-times Z, the job submission rate is (M-N)/Z jobs per second when there are N jobs already active. The "balance point" of the system is the load $N=N_0$ which satisfies the equation of perfect balance between job

---

[2] P. J. Denning, "Resource allocation in multiprocess computer systems," MIT Project MAC, TR-50 (May 1968). See also P. J. Denning & S. C. Schwartz, "properties of the working set model," Comm. ACM 15, 3 (March 1972).

[3] H. Opderbeck and W. Chu, "The renewal model for program behavior," SIAM J. Computing 4, 3 (1975).

[4] D. R. Slutz, I. L. Traiger, "A note on calculating mean working set size," Comm. ACM 17, 10 (October 1974).

submissions and completions:

$$(M-N)/Z = X(N) .$$

Following Courtois's suggestions, we can obtain the solution of this equation graphically as the intersection between the graph of X(N) and the load-line (M-N)/Z.

A deviation of N from N_0 produces a counterbalancing tendency for N to return to N_0. If N_0 is not too small, the restoring tendency is symmetric and the load distribution p(N) will be approximately normal. Using simple central-limit and diffusion arguments, we can approximate the mean and standard deviation for this distribution as, respectively,

$$m = N_0$$

$$\sigma = 2N_0/X_0(N_0)$$

Our preliminary work with this shows that the normal is a poor approximation for small N_0 (as expected), but is good when $N_0 > \sigma$; however, when N_0 gets very large, this σ begins to be too large. We are studying how to improve the estimate of σ for large loads.

Besides giving a simple estimate of the load distribution, this approximation is efficient for computing a load distribution for moderate and large loads. The exact solution is obtained from an iteration,

$$p(N) = p(N-1)(M-N+1)/ZX(N), \quad N = 1,\ldots,M,$$

which must be normalized [6,8]. Although the distribution can be calculated in time proportional to M, a separate computation is needed for each value of M. Except for small M, the approximation is much faster to compute.

The response-time distribution should also be approximately normal when N_0 is not small, or when the number of tasks in each job is not too small [8]. We are checking this hypothesis against simulations, and we seek simple methods of finding the mean and variance when it applies. This result would enable an analyst to estimate percentiles of response-time distributions without simulation, thereby answering one of the most difficult questions in performance analysis.

Approximation for Heavily Loaded Systems

H. D. Schwetman
S. Tolopka

The goal of this project is solving queueing networks with large numbers of customers, and applying the results to the Purdue Time Sharing System (MESA). The numerical difficulties of algorithms for computing exact solutions for systems with large state spaces has been noted earlier. These algorithms are unreliable for loads in excess of 100 users, the conditions of most interest in MESA.

To avoid these difficulties, we are extending the decomposition techniques of Chandy, Herzog, and Woo.[5] To study the queue distribution of a given device, we replace the remainder of the network with an equivalent device (whose service rate is the throughput at the given device when its service time is imagined to be zero). The state space of the resulting two-device system is quite simple and can easily be solved by iterative methods. We are extending the analysis by permitting stage-type service at each of the two devices,[6] thereby approximating the effect of nonexponential distributions.

An initial algorithm for this problem was numerically unstable; we are implementing another (which is stable). When it is working, we will compare its predictions with actual measurements from MESA.

Hybrid Simulations

H. D. Schwetman

Hybrid simulation is a modeling technique which combines discrete event simulation with analytic modeling to achieve faster simulations without loss of accuracy [16,17]. In the models investigated, the resources of the system were partitioned into two sets: those which jobs use for a long period (e.g., main memory) and those which jobs use for a short period (e.g., the CPU

---

[5] K. M. Chandy, U. Herzog, and L. Woo, "Approximate analysis of general queueing networks," IBM J R & D 19, 1 (January 1975), 43-49.

[6] C. H. Sauer, "Configuration of computing systems: an approach using queueing network models," Ph.D. Thesis, University of Texas, Austin, Texas (1975).

and I/O devices). The hybrid model uses a queueing network to model the use of the short-term resources, and a simulation to model the arrivals of jobs and allocation of long term resources.

The essence of the hybrid scheme is as follows. When the simulator is initialized, the parameters are used to compute the completion rate X(N) of the central-server part of the entire system. (This is an application of decomposition since, for each N, X(N) is computed as if the central server were subjected to a constant load of N jobs.) If the simulated scheduler decides to activate a new job at time t, it selects for that job a number n of CPU-bursts from the empirical distribution of such bursts for the given jobs. Now, without queueing, these n bursts require a mean time to complete

$$T(n) = nS_1 + \frac{n-1}{1-q_{10}}(q_{12}S_2+\ldots+q_{1K}S_K),$$

where S_1 is the mean CPU burst-time, S_i is the I/O service time at device i, q_{1i} is the probability of selecting device i on completion of a CPU burst, and q_{10} is the probability that the job completes on finishing a CPU burst. If jobs $j = 1,\ldots,N$ are active at time t, then

$$m = \min\{n_1,\ldots,n_N\}$$

is the number of cycles until the first job completes; the expected time for this is

$$T = \frac{X(N)}{X(1)}\,T(m) \quad .$$

(The factor X(N)/X(1) represents the speed loss in the central server due to queueing.) The simulator sets an event for the next completion at time t+T, and associates this with a job J for which n_J=m. At time t+T, all the n_j are reduced by m.

We compared the hybrid simulator with a simulation of the entire system. Replacing the simulation of the central server by a model considerably improved the speed of the simulator: depending on parameters, speedup factors from 20 to 200 were observed. The hybrid simulator predicted CPU utilization within 2% of the full simulator's calculation, and total memory-residence-time of given jobs within 5%.

Program Behavior and Optimal Multiprogramming

P. J. Denning
G. S. Graham

G. Scott Graham completed (in absentia) his dissertation in the Fall of 1976 [12]. In the first part of his thesis, he classified memory policies for multiprogramming from the least efficient (fixed partitions) to the most efficient (variable partitions determined by working set measures). He also classified program behavior models from the least realistic (independent references, independent LRU stack distances) to the most realistic (phases and transitions). Much of this work was published in 1975.[7]

In the second part of his work, Graham conducted extensive experiments on program behavior using 8 address trace tapes. For various memory policies, he measured each program's lifetime function, which specifies the mean virtual time between page faults for given memory resident set size under a given memory policy. A lifetime function is a sufficient representation, for use in simple queueing network models, of a program's memory demand. He compared the policies WS (working set), PFF (page fault frequency), LRU, and VMIN (the optimal), and used lifetime curves to study phase-transition properties of programs. In all the comparisons WS averaged at least as well as the other policies, and, in some, was significantly better.

In Spring 1976, Denning et al. [9] combined efforts on a paper about multiprogrammed load control. The objective was a dynamic control on the level of multiprogramming so that thrashing could not occur and system throughput was near maximum. The simplest control, the 50% rule, adjusts load to keep paging-device utilization at about 50%; higher utilization would imply a mean queue size there greater than 1, a sign of thrashing. An intermediate control, the L=S rule, adjusts load so that the mean system lifetime (L) between page faults balances the mean service time (S) of a page swap; smaller L (higher load) imply that programs are requesting new pages faster than the system can respond to such requests, a sign of thrashing. The best control, the knee rule, grants each active program a resident set

[7] P. J. Denning and G. S. Graham, "Multiprogrammed memory management," IEEE Proc. 63, 6 (June 1975).

whose corresponding lifetime value is at the primary knee of the program's lifetime curve; this limits load implicitly to the number of jobs whose resident sets fill memory. At the primary knee, the component of memory spacetime due to page faults is minimum; operating all programs at their knees tends to maximize throughput and avoid thrashing. Queueing network studies, using measured lifetime functions and compared with data from actual systems, corroborated that these criteria do work and demonstrated that the knee rule is indeed the most robust.

Graham and Denning extended these results further still. Graham's data [12] included measurements of lifetime and spacetime curves for each program. Under the WS policy, the primary knee resident set size generated spacetime within 1% of the minimum; this affirmed the hypothesis of the knee criterion and explained more clearly why it works. These data further showed that, of all known nonlookahead memory policies, no policy's minimum spacetime is smaller than WS's knee spacetime; nor is any such nonlookahead policy likely to be discovered. This showed that WS is the best candidate for optimal memory management. We then asked, how far from optimum need be a well-tuned WS policy with one single global value of the parameter? For Graham's sample of programs, a window of 73,000 references caused all programs to operate within 10% of their WS spacetime minima, the average deviation over the sample being about 5%. This puts the system throughput for the tuned WS within 5% of optimum. The same question for the PFF policy showed that the best global parameter setting could only guarantee that each program operates with 50% of its minimal spacetime, suggesting that a tuned PFF would not be practical. These results, which are part of Graham's thesis, were presented at the IFIP symposium in August 1977 [13].

All these arguments and data lead to the conclusion that a tuned WS policy can easily be built; this policy has almost no overhead if given proper hardware support (which is also inexpensive), and causes the system to operate within a few per cent of its optimum capacity. Denning has taken the position that, in this sense, the multiprogramming problem is now solved [5].

Generalized Working Sets

P. J. Denning

In Spring 1975, Denning and Slutz [of IBM Research, San Jose] independently discovered the same results; their subsequent collaboration extended the theory of working sets. We completed a third revision of the paper during summer 1977 [11]. The result of the research is the generalized working set (GWS), which is a model for the entire class of one-parameter, demand-fetching memory policies whose resident sets satisfy an inclusion property as the parameter is increased. The program's reference string $r(1)r(2)...r(T)$ is a sequence in which $r(t)=i$ is a reference to a segment of size z_i bytes. (For paging, all z_i are the same.) Associated with segment i at time t is a "retention cost" function $R(i,t)$, which specifies the cost of keeping segment i in the resident set since its prior reference. A valid retention cost is infinite before the first reference to a segment, is nondecreasing in t, and is reset to 0 just after a reference to segment i. A segment fault generates a delay proportional to $1+Az_i$, where A is the ratio of byte-transfer time to swapping device access time.

The GWS is based on the idea of balancing retention cost against retrieval cost. Using a "threshold" θ to represent the relative cost of memory and swapping, the GWS $W(t,\theta)$ contains r(t) plus each segment i whose retention cost has not yet exceeded the retrieval cost:

$$R(i,t) \leq \theta(1+Az_i) \ .$$

If $R(i,t)$ measures time elapsed since prior reference, and A=0, the GWS reduces to the time-window WS for paging with window size $\theta+1$. If $R(i,t)+1$ is the stack-distance function of a given stack paging algorithm, GWS simulates that algorithm for memory of size $\theta+1$ pages.

Corresponding to each GWS is an optimal policy GOPT that minimizes the total actual retention and retrieval costs. It deletes segment $i=r(t)$ from the resident set just after time t, if the retention cost will exceed $\theta(1+Az_i)$ by the time of next reference.

We devised efficient procedures for computing such measures as mean retention cost, mean resident set size, fault rate, and byte-flow rate of missing-segments. These procedures calculate the measures at selected value of the threshold (θ), using data collected on a single pass of the reference string, for both GWS and GOPT. (Graham used the GOPT procedure to calculate the VMIN lifetime curves in his experiments [12].)

Linear Paging Phenomena

P. J. Denning
R. Simon

We began this project out of curiosity about the causes of the "linear headway function" Saltzer reported for Multics.[8] The headway function, h(M), gives the mean time between system page faults when M main memory page frames are available in the system. A measurement showed h(M) = aM for some constant a in Multics. Denning [9] showed that linearity could not be a manifestation of program behavior; rather it must be a property of the way the user community loads down the system as a whole.

Simon undertook a study of this. Some simple approximations showed that h(M) should be approximately linear as long as M is small compared to the total amount of user-data stored in the system. This conclusion was not particularly startling, since any continuous curve will be approximately linear near the origin. In search of a relation between the approximately linear range of h(M) and the other parameters of the system, we set up a simple queueing network model. We quickly discovered that the load control and swapping policies have a significant effect on the headway function actually observed. There seems to be a relatively narrow memory-size range (M_1, M_2) such that M_1 is too small to contain the working sets of active users and M_2 is just sufficient. The load on the swapping device is high when $M=M_1$ but not when $M=M_2$. The rapid change in headway across this range was not consistent with a linear approximation for M larger than M_1.

As noted earlier in this report, we have not yet perfected the analysis techniques for systems with large numbers of users. Thus we have been unable to explore the headway function in systems with large amounts of user-data stored in them. This project has, thus far, led to no firm conclusions.

Recently, Hamilton [10] published data showing that, in Michigan's system MTS, h(M) is of the form aM^b for some $b > 1$. No one seems able to explain

---

[8] J. H. Saltzer, "A simple linear model of demand paging performance," Comm. ACM 17, 4 (April 1974), 181-186.

[9] P. J. Denning, "Comments on a linear paging model," Proc. SIGMETRICS Symposium (October 1974).

[10] J. A. Hamilton, "Congestion in multilevel paging hierarchies," Computer Performance (M. Chandy and M. Reiser, Eds.), North Holland (1977), 397-410.

why this data differs from the Multics data. Until we have a better understanding, we will avoid analyses that depend on a particular assumption about the form of h(M).

MIN Distance Strings

P. J. Denning

The MIN paging algorithm produces the smallest possible paging rate for each fixed-size of resident set. In their 1970 study of stack algorithms, Mattson et al showed how to compute MIN distance strings (and therefore page fault curves) in two passes of the reference string. In 1974 Belady and Palermo showed a method whereby, at each time t, the correct MIN distance d(t) could be computed knowing only the references r(1)...r(t) -- in other words, MIN distances can be generated in real time without lookahead even though MIN resident sets cannot be. Shortly after this Lewis and Nelson [IBM, Yorktown Heights] presented an alternative proof that MIN distances are real-time generable. Inspired by this, Denning found a short proof, which Muntz then helped shorten further [10].

The proof begins with a lemma that the MIN stack distance of x after a reference strong of the form α x β (where x does not appear in β) depends only on β but not α. This lemma does not lead directly to an efficient algorithm for computing the stack distance of x. An efficient algorithm results if we maintain the MIN stack at each time t on the hypothesis that the ordering of references in the future is the reverse of the LRU stack at time t; on seeing r(t+1)=x, we can correct this stack if r(t+1)≠x is inconsistent with the hypothesis about the future. The distance of x in the MIN stack before the correction is d(t). The proof uses the lemma noted above.

EVALUATION PROJECTS

Our evaluation projects are directed both toward validating specific modeling methods, and toward applying models to real systems. They include certifying queueing network evaluator programs, characterizing when a model's results are "close" to actual measurements, compressing workload data, and implementing capability machines efficiently.

Certifying Queueing Network Evaluators

H. D. Schwetman

Several research groups have implemented versions of algorithms for computing performance quantities from queueing network models -- for example, University of Texas [Chandy et al], IBM Yorktown Heights [Reiser et al], BGS Systems [Buzen et al], and our own group. These programs are not simple; certifying that they work correctly is difficult.

We have undertaken a project that will help authors of future queueing network evaluators to certify their programs, and users to know if an evaluator meets minimum standards. The ultimate objective is a set of solved sample problems. The problems must span a wide range of system types, customer class assumptions, service disciplines, and load dependencies -- to exercise all features of evaluator programs. An evaluator that correctly solves all the sample problems would have a high probability of being correct.

We have the enthusiastic support of other research groups, who are anxious to cooperate in developing the sample problems and to see if their programs all produce the same answers on the same problems. We are hoping that each group will be willing to report the space and time required in their evaluators for each problem, and the dollar cost of obtaining each solution on their host system.

System Variability and Model Validation

H. D. Schwetman
R. Mead

One purpose of a model is defining an efficient algorithm for computing performance quantities from parameters of the system and workload. A good model will estimate performance quantities significantly faster than they could be measured, and the estimates will be consistently close to the values that would be measured. Defining what "close" means is more difficult than one might expect.

The problem is that a real system's performance may vary considerably for the same workload. We applied the same workload to our system three times and found that over these runs

- CPU utilization varied just 0.5%,
- Mean CPU queue length varied 7%, and
- The variation in memory residence time of some jobs was 53%.

These results suggest that expecting a queueing network model to estimate CPU utilization within 0.5%, or mean queue length within 7%, or memory residence time within 53% -- might be futile. What is needed is a distribution of observations over several runs of the same workload; the model's estimates must fall within tolerable confidence intervals.

We plan further experiments of this type. Our goal is quantifying the confidence intervals that should be expected in various performance measures. An assessment of a model's accuracy must be relative to the confidence intervals of each type of performance measure.

Workload Compression

H. D. Schwetman
R. Mead

We are undertaking a study of the effect of workload data-compression on the accuracy of simulation results. The approach is, first, to drive a simulator with an event trace measured directly from the real system then, second, to drive the same simulator with event traces generated from summaries of the workload. If the two simulation outputs thus obtained agree consistently, then the method of "compressing the workload" -- i.e., obtaining the summary and generating artificial event traces from it -- is considered effective.

We have experimented preliminarily with these compression methods:

- Use a job's files in random order rather than the specific sequence in the original event trace;
- Replace all service times at a given device with the mean service time; and
- Select service times by sampling from the empirical distribution.

These simple ideas have produced compressions of up to 95% with only small changes in the results of the simulation. (That is, the workload summary and associated event trace generator required as little as 1/20 of the storage as the full event trace of the system.)

In one of our experiments looking for regular patterns which could be exploited to compress data, we constructed an "activity plot" of file usage of a job (see the figure). This plot shows very definitely that file usage exhibits phase and transition behavior, much like page or segment referencing in programs [12]. At present we do not have enough data on logical file use -- the figure shows only the actual file references occurring when buffers are full or empty -- so we are not yet able to apply the idea.

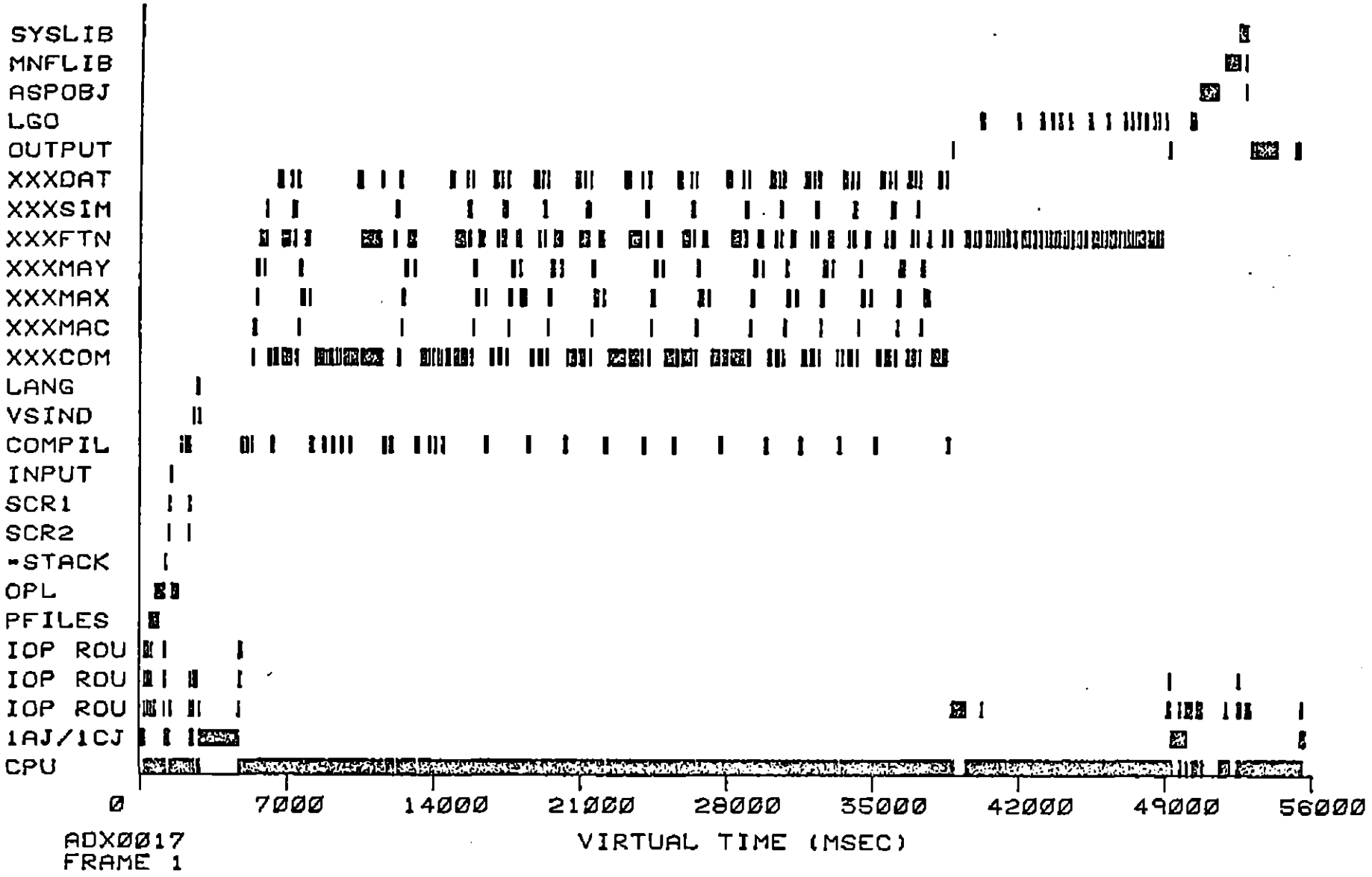
Implementations of Capability Machines

H. D. Schwetman
E. Gehringer

Much current research in secure operating systems uses the "capability"[11] as a means of authorizing access to segments, files, or other objects in the computer system. Among the more important experimental designs are the Hydra system [Carnegie-Mellon University], the CAP system [University of Cambridge], and the Provably Secure Operating System (PSOS) [SRI International]. The only commercial machine is the British Plessey Corporation's System 250. The goal of our research project is a performance comparison among the various design approaches employed in capability systems. Examples of this are discussed below.

Addressing efficiency is of basic importance: Can a capability machine translate addresses as fast as a machine which uses an ordinary segmented memory? For most address calculations, yes. The additional checks to validate a capability-based address can be performed in parallel with other addressing steps. A performance analysis, however, is complicated by the small associative store used to speed up address calculation. Computing an entry for the

[11] P. J. Denning, "Fault tolerant operating systems," Computing Surveys 8, 4 (December 1976), 359-389.



Activity Plot Showing Phase Behavior

associative store is more difficult in a capability machine. With an ordinary segmented virtual memory, associative store loads can be tolerated as often as every 100 instructions. But in a capability machine, these loads can be tolerated much less often, and thus a substantially larger associative store may be needed.

Capability based operating systems use a data structure called the "map" to record the binding between each capability and an object. Most implementations use a hash table; but the CAP system uses a tree-structure, in which a capability is located by following a series of links. Our investigation has shown that a linked map occupies less primary store, but requires more overhead in manipulating capabilities. A linked map leads to a simpler microprogram for address translation, whereas a hashed map seems to allow a more straightforward implementation of shared segments.

Two schemes have been proposed to protect capabilities from alteration: tagged memory and partitioned memory. We will use computer simulation to compare them. The results should quantify the advantages of building the hardware support for tagged memory.

Various schemes have been proposed for implementing structures known as "protected objects" or "extended-type objects".[12] These schemes have never been compared in performance. There is no systematic treatment of the overhead in switching from one protection environment to another while using these objects. We plan to study this too.

MEASUREMENT PROJECTS

Our measurement work supports the modeling and validation work by providing data from a large system. These data are bases for checking model predictions and supporting model assumptions. The projects include a study of the performance parameters of our time sharing subsystem, job flow analysis in our main operating system, and upgrading our measurement package.

---

[12] T. A. Linden, "Operating system structures to support security and reliable software," Computing Surveys 8, 4 (December 1976), 409-445.

New Approaches to Performance Data Reduction

S. Rosen
V. Abell
J. Eykholt

Scherr's famous study of CTSS has accustomed us to think of time-sharing jobs as alternating between two states - "active" and "thinking". In the early systems, the length of the active period characterized accurately the response time perceived by the user; it also characterized accurately the duration of a job's demand on the resources. In many systems today, this view is no longer accurate. In MESA, for example, considerable system resources are used by jobs in active states during which they are delivering output to the user. In these cases the user can perceive a response well before his job leaves the active state. The "initial response" time differs significantly from the total response time that would be predicted by the two-state model of a user interaction.

Our objective is a more careful analysis of the data. It should be stratified to report statistics on the three states - running/outputting, running/no-outputting, and thinking/no-outputting. Our preliminary work shows that this gives a useful and accurate presentation of the data [1].

The three-state model provides a more realistic presentation of system responsiveness. Data has been collected and will continue to be collected under various system loads. Our objective here is to be able to calculate the proportion of time a job will spend in each state on the average, as a function of the load on the system. A special effort is being made to collect data under very heavy system loads to provide us with a better understanding of the phenomena associated with system saturation.

We are also starting to study related techniques for jobs in the batch-stream of the operating system. For this purpose, the Dayfile Analyzer of the MACE system has been extended; it now supplies data on the flow of jobs into and out of the system, and into and out of major queues. We are planning further changes that will report job flows by jobsize categories.

In summary, our goals under this part of the proposed renewal of the grant are:

1. Continued study of active substates of time sharing jobs, particularly as it relates to characterizing the responsiveness of systems and their behavior under near-capacity loads.

2. Extension of the techniques to the analysis of data on job flows in the main operating system.

Measurement Package Upgrade

H. D. Schwetman

The MACE operating system, and its time sharing subsystem MESA, are under constant modification. To provide the data necessary for our modeling projects, we occasionally need to modify the system's data gathering routines. For example, we recently installed new queues for the disk subsystem; since these potential points of congestion must be reflected in models, we must modify the measurement routines to report their statistics.

PUBLICATIONS & REPORTS

1. V. A. Abell and S. Rosen
Performance of an ECS-based time sharing subsystem
Computer Performance, (K. M. Chandy and M. Reiser, Eds.), North-Holland (1977).

2. G. Balbo, S. Bruell, H. D. Schwetman
Computational aspects of closed queueing networks with different customer classes, TR-210 (Spring 1977).

3. G. Balbo, S. Bruell, H. D. Schwetman
Customer classes and closed network models -- a solution technique, Proc. IFIP Congress, North-Holland (1977), 559-564.

4. J. C. Browne, P. J. Denning, and J. L. Peterson
The impact of operating systems research on software technology
For Research Directions in Computer Technology (P. Wegner & J. Dennis, Eds.) MIT Press (1978).

5. P. J. Denning
Program Behavior, working sets, and multiprogramming
For: Current Trends in Software Methodology (M. Chandy & R. Yeh, Eds.) Prentice-Hall (1978). Available as TR-194. Written 6-76, revised 2-77.

6. P. J. Denning and J. P. Buzen
An operational overview of queueing networks
For: Infotech State of the Art Report on Performance Modeling and Prediction (1977).

7. P. J. Denning and J. P. Buzen
Operational analysis of queueing networks
Proc. 3rd Int'l Symposium on Modeling & Performance Evaluation of Computer Systems (October 1977), North-Holland.

8. P. J. Denning and J. P. Buzen
Queueing network models
A tutorial for: Computing Surveys special issue (G. S. Graham, Ed.) to appear 1978.

9. P. J. Denning, K. C. Kahn, J. Leroudier, D. Potier, R. Suri
Optimal multiprogramming
Acta Informatica 7, 2 (1976), 197-216.

10. P. J. Denning and R. R. Muntz
A short proof of Real time generation of MIN distance strings
TR-211 (November 1976; Revised September 1977); to J.ACM.

11. P. J. Denning and D. R. Slutz
Generalized working sets for segment reference strings
Accepted for Comm. ACM. (1978) - 3rd Revision.

12. G. S. Graham
Program Behaviour and Dynamic Memory Management
Ph.D. Thesis, CS Dept., Purdue University (Dec 1976).

13. G. S. Graham and P. J. Denning
On the relative controllability of memory policies.
In Computer Performance (K. M. Chandy and M. Reiser, Eds.)
North-Holland (1977), 411-428.

14. H. D. Schwetman
Job scheduling in multiprogrammed computer systems
Submitted to Software Practice and Experience.

15. H. D. Schwetman
Hybrid simulation models of computer systems
Accepted for publication in Comm. ACM.

16. H. D. Schwetman
Hybrid simulation models: a speed-up technique combining analytic and discrete-event modeling
Modelle für Rechensystemme Workshop 1977 (P. P. Spies, Ed.) Lecture Notes in Computer Science (9), Springer-Verlag, 226-235.

17. H. D. Schwetman
Computer systems: overview, performance evaluation, and performance prediction
TR-218 (June 1976)
[Based on lecture presented 4th April 1976 at the Institute of Information Sciences, University of Turin, Italy.]