SIMULATION AND MATHEMATICAL MODELING OF AN ON-LINE ACCOUNTING SYSTEM

Jack E. Shemer
Xerox Data Systems
El Segundo, California

Summary

Two approaches are presented for analyzing and predicting performance of an on-line business data accounting system. The computer system's responsiveness to typical (high priority) file updating demands is examined by two methods of a priori study: simulation and mathematical modeling. A discrete event simulation model, written in FORTRAN, is developed to comprehensively describe system performance. This simulation model is based upon the isolation and sequential ordering of logical "phases" which are seen to occur during the servicing of a user's request. To supplement this model, two mathematical models are developed from assumptions of exponential inter-arrival and service time distributions. The first of the mathematical models presents "worst case" estimates in the sense that no simultaneity of Drum and Disc Pack operations is considered possible; whereas the second model provides "best case" estimates since it assumes that a maximum degree of file I/O simultaneity is achievable. As expected, the simulation results (in all cases examined) were between the upper and lower bound response time estimates provided by the mathematical models. Hence the mathematical models serve to substantiate the validity of the results obtained from the simulation, and in concert, the three models provide performance estimates which range from "worst case" to "expected real world" to "best case." Thus it is shown how mathematical modeling can be coupled with simulation to increase the fidelity of system analysis.

I Introduction

In general, there are three basic methods for analyzing computer systems; 1) mathematical modeling, 2) simulation and 3) empirical investigation. With mathematical analysis the system is typically examined in a macroscopic manner. Commonly the methods of queueing theory are employed,[1-4] and the system is studied under equilibrium conditions. However, at best, mathematical models provide only general indications of system performance, since probability analysis is amenable only to a limited number of variables. In contrast, simulation models[5-6] strive to comprehensively inter-relate all variables of the system, thereby providing a microscopic view of the system. However, detailed simulation is frequently prohibitive due to the effort involved. Moreover, the fidelity of a simulation is often questionable because values for many of its internal parameters are unknown (especially prior to actual system operation). At the other extreme, empirical investigations[4-7] provide a degree of accuracy that is not experienced with mathematical models or simulation since such studies obtain results while the system is in operation. But, due to considerations of practicality and timeliness, posterior studies (requiring operation of the system itself) may prove to be infeasible or exorbitantly expensive. Therefore, only the techniques of mathematical modeling and simulation can be employed for a priori study when a system is in its early stages of design. Hence, ideally, these approaches should supplement each other since their results provide a basis of comparison.

With a priori studies it is typical that considerable doubt arises as to the validity and accuracy of the modeling results. Thus a basic analysis problem which must be confronted is that of establishing the credibility of system performance predictions. Here it is shown how the severity of this problem can be mitigated by employing several different a priori models and comparing their results. This paper employs simulation and mathematical modeling to obtain an a priori analysis of a proposed on-line business accounting system. The system's responsiveness to typical (highest priority) updating demands is examined and graphically analyzed for a variety of usage conditions in order to ascertain the capabilities of the proposed system.

II Brief System Description

Consider a proposed on-line business accounting system (OBAS) which would time-share its services among a number of remote users and tailor its functions to: a) file updating, b) information inquiry, and c) report generation. Broadly speaking, assume the computer system configuration consists of: a) a XDS Sigma 7 CPU, b) core memory sufficient for the operating system and up to an additional 64K words for on-line users (4K words per user "slot"), c) up to four medium speed Rapid Access Disc (RAD) controllers (7231s) and up to two RAD (head per track) storage units (7232s) per controller, all of which are dedicated to OBAS servicing, d) up to four Removable Disc Controllers (7240s), with each servicing a maximum of two dual spindle disk pack mechanisms (7242s) where, again, the entire Disc Pack complex is dedicated to OBAS, also, e) whatever additional hardware is deemed necessary to support report generation, system initialization and restart, file back-up and batch operations (e.g., high speed printers, magnetic tapes, paper tapes, card readers, card punches, etc.).

The basic mode of on-line user input and system output response would be via teletypes. When an on-line user is allocated core space, he would be given a fixed size, 4K area of main memory which is selected from those "slots" available for on-line allocation. It is assumed that user programs would run interpretively and that the system employs the Sigma 7 hardware memory map for relocation. Therefore, the assignment of a user to an area of core would not be binding, and a given user may be assigned to different areas of absolute memory during successive computations.

The OBAS executive would operate as a resident real-time program which is interfaced to a standard operating system that supports foreground operation. Here, the OBAS executive would control the scheduling and memory allocation of on-line users within its dedicated area of main memory and perform I/O via the File System of the host monitor.* All CPU activity for OBAS would be in the foreground mode.*

* Such features are common in a number of operating systems provided by XDS (e.g. BPM, RBM, etc.)

As previously stated, the system would be structured to provide preferential service to requests which fall in the updating class (e.g., posting activity, payroll inputs, accounts receivable, etc.) since these requests demand rapid response and, moreover, are considered to be the dominant mode of system demand. It is assumed that all such activities, on a per request basis, require a minimal amount of CPU time (in the majority of cases, a total of less than 16 ms. per request); whereas, due to the involvement of multiple files and indices to these files, a sizable amount of I/O time is demanded (typically six RAD access-es per request). However, problems of storage allocation are negligible since files are not dynamically expanding but, rather, simply being modified.

When a user inputs an update request to the system, the system must perform the following I/O operations to satisfy each such transaction:

1. Fetch from the RAD the user's identification (ID) block (≈ 1K words which locate the user's program, describe operational history, specify access privileges, etc.)
2. Read user program from RAD (≈1K words)
3. Employ a user supplied "key" and access the user's file directory from RAD (≈512 words)
4. Obtain appropriate record from Disc Pack (≈512 words)
5. Update record and write modified record back on the Disc Pack (step 4 is a "read with hold cylinder position" so that step 5 can be accomplished with reduced access time - i.e. the cylinder positioning on the write operation will be instantaneous).
6. Update user accounting data in the ID block and restore it upon the RAD

Steps (3), (4), and (5) are generally repeated two or three times per user transaction since the updating of a record can affect more than one file (e.g., time card data affects payroll files, labor files, and, perhaps, the general ledger).

In any system of this sort, additional I/O activity is required per transaction to insure operational fidelity and error recovery. There are many ways to achieve this. A simple approach, which requires minimal additional I/O activity per transaction, would be to maintain a log of all transactions.* This log, together with periodic duplication of all files on tape, would provide a basis for system integrity. At the other extreme, a separate Disc Pack could be dedicated to duplicating all modified files. But this approach would necessitate additional I/O activity per transaction (typically 3 or 4 write operations) and thereby tie up user core "slots" for extended periods of time. For the purposes of this paper, let us assume that the former approach is employed, whereupon little (if any**) additional I/O is required.

* This log could also be used as a basis for performance tuning or as a tool to drive simulations.

** Multiple transactions would generally be buffered before an update of the transaction log was required.

III Assumptions and Analytical Approaches

Since the proposed system provides preferential treatment to transactions of the updating variety and since such requests are assumed to constitute the majority of demands imposed upon the system, the analysis examines the system's ability to respond to these demands. It is assumed that the response time R to typical updating requests is, singly, the most relevant measure of performance, because "satisfaction" of such transactions is only achieved when "adequate" response is received. Here, response time R is defined as that amount of time which elapses from the instant at which a user "keys-in" an activation character (i.e., demands some service from the OBAS executive) until the system responds with the prompt character (and/or associated output data) which acknowledges the servicing of the transaction and informs the user that the system is ready to receive another request.

Within the above framework it is apparent that so far as updating activity is concerned, the system's capacity to accommodate I/O requests is the limiting aspect of performance. Therefore the models developed in the Appendix principally examine I/O queueing* (i.e., RAD and Disc Pack servicing). To facilitate the study, the I/O system is considered to be modularized. The "basic I/O module" consists of a single RAD controller (7231) with two storage units (7232's) and one Removable Disc Controller (7240) with two dual spindle mechanisms (7242's). Now, since the system is I/O limited (and also due to storage capacity considerations), only a limited number of users can be accommodated by a single "I/O module." However, since the CPU time/transaction is minimal, it is reasoned that performance estimates derived for a "one module" system with S core "slots" for N concurrent users are applicable to a "k module" system which operates with N·k concurrent users and S·k user core "slots" (i.e., providing the user population is equally distributed over the "k modules" and also that the system is capable of operating as many as "k modules" concurrently, whenever they are demanded).

A detailed investigation of system performance is achieved by means of a discrete event simulation model. This model is described in the Appendix and comprehensively accounts for the queueing phenomena which user transactions experience during various phases of service.

Also, two mathematical models are formulated in the Appendix to supplement the simulation study. The first of these presents "worst case" estimates for the expected response time E[R] since it assumes one user core "slot" per "I/O module." Even though more than one user is queued for an "I/O module," only one is assumed to be active at any given time. Hence, the servicing of I/O activities for that "module" is strictly sequential with no time overlap (i.e., it does not permit simultaneous operation of a RAD and Disc Pack which comprise the same "module"). The second mathematical model provides "best case" performance estimates since it assumes that a maximum degree of I/O simultaneity is achievable. Given that a request for RAD service is queued, the second model considers that it is always possible to commence servicing such a request (i.e., there is always an available user "slot"). The only conservative aspect of this "best case" model is its inclusion of device interference conflicts which can exist between individual Disc Pack service

* However, the analysis does not account for any contention problems which may arise when files are shared.

requests. This is achieved by relating the description of Disc Pack servicing to a) the number M of independent mechanisms and the maximum number S of user core "slots" available per "module", and b) the number n_2 of requests queued for the removable disks.

With the above models, it is generally assumed that:

1. Each on-line user interacts with the system at an average rate λ(requests/sec.) which, depending upon the environment, varies from 1 request every 3 seconds to 1 request every 7 seconds. The interaction interval corresponds to the time period during which user is preparing a request for the system (i.e. his thinking plus typing time). For each user the duration of this interval is described by a negative exponential distribution function whose mean* is $1/\lambda$.

2. A total of N users per "I/O module" are concurrently "on-line" and vying for service.

3. The average total RAD service time per request is $1/\mu_1$ = 160 ms. (including six random access and all associated data transfers).

4. The average total Disc Pack service time per request is $1/\mu_2 = 440$ ms. (including all accesses and associated data transfers, excluding file back-up).

Further details of the user transaction cycle and associated services are included in the simulation model as described in the Appendix.

If the simulation model and mathematical models are accurate, then it is expected that the simulation will yield average response times somewhere between the two estimates provided by the mathematical models.

IV Results

Employing the three models given in the Appendix, estimates of expected response time** E [R] are derived and graphically displayed in Figures 1, 2, and 3 where λ, the interaction rate of each user, equals 1 request/3 seconds, 1 request/5 seconds and 1 request/7 seconds, respectively.

Figures 4, 5, and 6 dipict the 80% point of the response time distribution as obtained from the simulation model for corresponding values of λ. The ordinate axis represents that value of response time R(80%) such that 80% of the user transactions experience response times less than or equal to R(80%).

Here it is assumed in the simulation and "best case" model that the number of user core "slots" per "I/O module" equals the number of independent Disc Pack mechanisms which are available per "module" (i.e., M = S = 4). Thus if the total CPU time required per user transaction is negligible; if problems of secondary storage allocation don't exist; and if the system is capable of operating k "I/O modules" concurrently whenever they are demanded, then it is reasoned that the estimates for N users per "I/O module" can be extended to N·k users when k "I/O modules" make up the system, providing that core space is expanded to provide 4·k user "slots" and the user population is evenly distributed over the "k modules". Given these assumptions, when k = 4 it is reasonable to conclude that the proposed system is capable of supporting 80 users (and still afford response times which are typically less than 3 sec.). However, it should be noted that this linear extrapolation is only an approximation, and therefore should be viewed cautiously (e.g. for some value of k, CPU time would become significant due to functions of I/O initiation and I/O interrupt handling performed by the File System).

V Comments

As previously noted, the simulation results should be somewhere between the two estimates provided by the mathematical models. Upon examining Figures 1-3, it is evident that all simulation results are consistent with this conjecture. Therefore, given that the assumptions of the investigation are correct, it is reasonable to attribute considerable accuracy to the simulation results. To this extent, it has been shown that mathematical modeling can be coupled with simulation to increase the fidelity of the analysis.

However, it must be again emphasized that throughout the analysis the total CPU time per request was assumed to be negligible (typically, < 16 ms per transaction). This total included user program execution (interpretively) together with all the processing required by the File System of the host operating system. Thus, the OBAS Interpreter should be highly efficient and the number of interpretive commands which users are allowed to execute per transaction should be constrained by some mechanism in the OBAS monitor (perhaps adjustable depending upon the environment). Moreover, all I/O activity performed by the File System must be accomplished with minimum delay. If this is not accomplished, then CPU queueing must be considered in the models, and the predictions displayed in the figures are grossly inaccurate.

None of the models presented here accounted for a priority structure in the OBAS scheduler or considered requests other than those of the update variety. A more general and flexible simulation model could be readily developed to allow for flexible priority assignment, arbitrary definition and sequencing of service phases for each distinguishable request type (e.g. update, inquiry, report generation, etc.), and comprehensive treatment of multiplexing control for k "I/O modules". Then providing the user environment could be accurately characterized, this latter model could be used to obtain more detailed performance estimates.

In summary, it must be emphasized that this is a preliminary analysis. Many of the assumptions employed here which characterize system usage are subject to closer review. For example, as the system evolves, it is reasonable to assume that more and more diverse demands would be imposed upon the hardware/software complement, thus negating the assumption of a homogeneous user environment. Yet even though the models are extremely simplified approximations to complex situations, their results are credible and, in this respect, demonstrate the potentials of the proposed OBAS system.

* Here $1/\lambda$ is average duration between system generated response (user receipt of a system generated character) and the generation of the next user request (the "keying-in" of an activation character).

** Recall that E [R] is the average response time afforded to requests which are "typical" update transactions and which are therefore characterized by a relatively large amount of I/O but minimal CPU time.

Acknowledgements

The author is indebted to P. F. King for his programming work to implement the simulation model.

Bibliography

1. L. Kleinrock, "Time-Shared Systems: A Theoretical Treatment", Journal of the ACM, Vol. 14 (April, 1967), pp. 242-261.

2. E. G. Coffman, Jr., "Stochastic Models of Multiple and Time-Shared Computer Operations" (Report No. 66-38; Department of Engineering, University of California at Los Angeles, June, 1966).

3. J. E. Shemer, "Some Mathematical Considerations of Time-Sharing Scheduling Algorithms", Journal of the ACM, Vol. 14 (April, 1967), pp. 262-272.

4. J. E. Shemer and D. W. Heying, "Performance Modeling and Empirical Measurements in a System Designed for Batch and Time-Sharing Users", Proceedings 1969 FJCC, November, 1969, pp. 17-26.

5. A. L. Scherr, "An Analysis of Time-Shared Computer Systems," M.I.T. Project MAC Report MAC-TR-18, (Cambridge, June, 1965).

6. N. R. Nielson, "The Simulation of Time-Sharing Systems", Comm. of ACM, Vol. 10, pp. 397-412.

7. G. E. Bryan, "JOSS:20,000 Hours at the Console - A Statistical Summary", Proceedings 1967 FJCC, pp. 769-777.

Appendix

A. Model 1 - Simulation Model
The discrete simulation modeling technique is based upon the isolation of logical "phases" which are seen to occur during the servicing of a user's request. Phases here are distinguished by the required action of entering a queue for their initialization. Such actions will be required for users and their respective programs to obtain access to core, CPU, and I/O processors. The instants marking entry of a user's request into a queue and completion of service for that user's request are simulated in order of occurrence and cause the simulation clock to advance. These event epochs are respectively equivalent to initiation of a phase of service and completion of a phase of service. In short, a "phase" of service is a single distinct task performed by one of the servicing units* (RAD, CPU, DISC, or CORE) which must be completed before a user's request is completed. When a user enters a phase of service, a service element is specified, a service requirement is simulated, and the request is entered into the respective queue according to a first-come-first-served rule. For each distinguishable phase, a service time calling parameter is used together with a service distribution function to simulate the service time (or amount of service resource) required.

* Also during each transaction cycle (see Figure 7), a user passes through a read state (phase 0) which corresponds to the request generation interval (i.e. a user's typing plus thinking time).

In general, each user request will spawn several phases of service which must be sequentially completed before the user can again impose another request upon the system. During this sequence, a given user is in one, and only one phase of service at any arbitrary instant of time. The migration of a user from one phase of service to another is simulated in the manner depicted in Figure 7. For each of the tasks associated with the phases of the model a service time must be generated. The method used to obtain these times is the commonly employed Monte-Carlo technique. First a pseudo-random number is generated, then an inverse mapping under an appropriate probability distribution function is performed to yield the corresponding service time. The same method is also used to determine i) the total number of files to be modified during any single transaction, and ii) also which of the four disc queues (within the module) shall service the user.

As diagrammed in Figure 7, each user transaction is comprised of the set of all service phases invoked to complete a request (i.e., the response cycle is equivalent to that period elapsing between request initiation and request completion). Thus, the response times to any user request depends on the individual times required for each of the set of predefined phases as encountered during the simulation of their sequential execution where, of course, the time required for phase completion results from time spent queued in addition to the actual service time. The model accumulates a complete history of simulated transactions and presents this in the output summary which contains a probability distribution for response time, the mean response (and its standard deviation), the mean utilization of each I/O device, the average amount of I/O and CPU service per request, and a number of other salient performance indexes.

The program employs eight queues to model each "I/O module". Queues are assigned for core, the RAD, and the CPU. The Disc Pack controller is assigned four queues to correspond to the possible maximum of two dual spindle mechanisms. These Disc Pack spindle queues are multiplexed to another queue representing a data channel in order that a shortest-access-time-first strategy can be simulated for channel servicing of the four independent disc mechanisms. Elements of most queues have user and phase identification associated with them, thereby providing a means of determining and monitoring the sequence of discrete events that occur within the system since each user is characterized by a state vector whose elements depict his current phase, next phase, core residence condition, arrival time, accumulated service time, etc.

The model itself, was implemented in FORTRAN in order to maximize the ratio of simulated time to job run-time and also to conserve memory requirements.

B. Model 2 - "Worst Case" Estimates
Consider that the generation of on-line transaction requests on each communication line is an exponential process with parameter λ denoting the mean request rate. Thus, the time interval t elapsing between completion of a request (i.e., the sending of a prompt character to a user) and the generation of a new request by that user is described by the distribution function

$$A(t) = \begin{cases} 1-e^{-\lambda t} & \text{for } t \geq 0 \\ 0 & \text{for } t < 0 \end{cases} \qquad (1)$$

where $A(t)$ denotes the probability of a new request arrival.

Similarly, assume that the total I/O service time t required by each on-line request is exponentially distributed with mean $1/\mu$ and described by the distribution function

$$B(t) = \begin{cases} 1-e^{-\mu t} & \text{for } t \geq 0 \\ 0 & \text{for } t < 0 \end{cases} \qquad (2)$$

Here it is assumed that $1/\mu$ is the expected sum of the RAD and Disc Pack service times. Thus, the model doesn't consider that the RAD and Disc Pack can operate concurrently (i.e., it does not allow for simultaneous servicing of different requests).

Assume that the system can be modeled as a single server queue with state dependent arrival rate. Then, given that there are N unique communication sources, let $p_n(t)$ denote the probability that n on-line requests are queued at an arbitrary instant of time t for $n = 0, 1 \ldots N$, whereby

$$\frac{dp_n(t)}{dt} = \begin{cases} -N\lambda p_o(t) + \mu p_1(t) & \text{for } n=0 \\ -[(N-n)\lambda + \mu]p_n(t) + (N-n+1)\lambda p_{n-1}(t) + \mu p_{n+1}(t) & \text{for } 0<n<N \\ -\mu p_N(t) + \lambda p_{N-1}(t) & \text{for } n=N \end{cases} \qquad (3)$$

From these equations, the stationary probability that n on-line requests are queued is

$$P_n = \frac{N!}{(N-n)!} (\lambda/\mu)^n p_o \qquad (4)$$

where

$$p_o = \left[1 + \sum_{n=1}^{N} \frac{N!}{(N-n)!} (\lambda/\mu)^n\right]^{-1}$$

Note that in the above equations, the input rate is (N-n) when n requests are queued. Thus, the model accounts for the natural variations in demand intensity which result because there are a finite number N of input sources.

Now employing the above results, the expected response time $E[R]$ is expressed

$$E[R] = 1/\mu[E[n] + 1] \qquad (5)$$

where

$$E[n] = \sum_{n=1}^{N} np_n$$

C. Model 3 - "Best Case" Estimates

Again, consider the input per communication source to be an exponential process described by equation 1 above.

Assume that the RAD and Disc Pack service times required by each on-line request are individually exponentially distributed with means respectively equal to $1/\mu_1$ and $1/\mu_2(n_2)$ when n_2 requests are queued for Disc Pack service.

Consider that the system can be examined as two queues in tandem. Let $p(n_1, n_2, t)$ denote the probability that n_1 requests are queued for RAD service and that n_2 requests are queues for Disc Pack service at time t where $n_1 + n_2 = 0, 1 \ldots N$. Then, the state differential equations become

$$\frac{dp(0,n_2,t)}{dt} = -\left[(N-n_2)\lambda + \mu_2(n_2)\right] p(0,n_2,t) + \mu_2(n_2+1)\, p(0,n_2+1,t) + \mu_1 p(1,n_2-1,t) \qquad \text{for } 0 \leq n_2 < N$$

$$\frac{dp(n_1,0,t)}{dt} = -\left[(N-n_1)\lambda + \mu_1\right] p(n_1,0,t) + (N-n_1+1)\lambda p(n_1-1,0,t) + \mu_2(1)\, p(n_1,1,t) \qquad \text{for } 0 \leq n_1 < N$$

$$\frac{dp(N,0,t)}{dt} = -\mu_1 p(N,0,t) + \lambda p(N-1,0,t) \qquad (6)$$

$$\frac{dp(0,N,t)}{dt} = -\mu_2(N)\, p(0,N,t) + \mu_1 p(1, N-1, t)$$

$$\frac{dp(n_1,n_2,t)}{dt} = -\left[(N-n_1-n_2) + \mu_1 + \mu_2(n_2)\right] p(n_1,n_2,t) + \mu_1 p(n_1+1, n_2-1, t) + \mu_2(n_2+1)\, p(n_1, n_2+1, t) + (N-n_1-n_2+1)\lambda p(n_1-1, n_2, t) \qquad \text{for } 0 < n_1, n_2, < N$$

Hence the stationary probability that $n_1 + n_2$ on-line requests are queued is

$$p(n_1,n_2) = \frac{N!\ \lambda^{n_1+n_2}\ p(0,0)}{(N-(n_1+n_2))!\,(\mu_1)^{n_1}\,(\mu_2(1)\ldots\mu_2(n_2))} \qquad (7)$$

where

$$p(0,0) = \left[\sum_{i=0}^{N} \sum_{j=0}^{N-i} \frac{N! \quad \lambda^{i+j}}{(N-i-j)!(\mu_1)^i(\mu_2(1)\ldots\mu_2(j))} \right]^{-1}$$

In the above equations, it is again assumed that the arrival rate is a function of the number queued n_1+n_2. Also, it is assumed that Disc Pack servicing is a function of the number queued n_2 since multiple independent mechanisms (up to four in a single "I/O module") can function concurrently when more than one request is queued. In general, if there are M independent device mechanisms and if there are a total of S user core "slots", then when $S \geq M$ the function $\mu_2(n_2)$ can be expressed

$$\mu_2(n_2)=\mu_2 \Bigg[M(1/M)^{n_2} + 2 \sum_{\substack{\text{FOR ALL} \\ k_1+k_2=n_2}} \frac{n_2!}{k_1!k_2!} (1/M)^{n_2} \frac{M!}{2!(M-2)!}$$

$$+3 \sum_{\substack{\text{FOR ALL} \\ k_1+k_2+k_3=n_2}} \frac{n_2!}{k_1!k_2!k_3!} (1/M)^{n_2} \frac{M!}{(M-3)!(3!)} + \ldots + \tag{8}$$

$$\left\{ \begin{array}{l} M \displaystyle\sum_{\substack{\text{FOR ALL} \\ k_1+..k_M=a_2}} \frac{a_2}{k_1!k_2!..k_M!} (1/M)^{a_2} \Bigg] \text{ for } n_2>M \\ \qquad\qquad \text{or} \\ \dfrac{n_2}{(M-n_2)!} (M!/n_2!) \dfrac{n_2!}{k_1!..k_{n_2}!} (1/M)^{n_2} \Bigg] \text{ for } 2 \leq n_2 \leq M \end{array} \right.$$

where $k_i \geq 1$ and

$$a_2 = \left\{ \begin{array}{l} n_2 \text{ if } M \leq n_2 \leq S \\ S \text{ for } n_2 \geq S \end{array} \right.$$

Thus with $M = 4 = S$

$$\mu_2(1) = \mu_2$$

$$\mu_2(2) = 2\mu_2 - \mu_2/M$$

$$\mu_2(3) = 3\mu_2(1-1/M) + \mu_2/M^2$$

$$\mu_2(4) = 4\mu_2 - 6\mu_2/M + 4\mu_2/M^2 - \mu_2/M^3$$

$$\mu_2(j) = \mu_2(4) \text{ for } 4 \leq j \leq N$$

and for cases in which $4 \leq n_2 \leq S$, a good approximation is

$$\mu_2(n_2) \cong \mu_2 + (n_2-1)(1-1/M)\,\mu_2$$

Now, employing the above results, the expected response $E[R]$ is approximated by

$$E[R] = E[n_1]\,(1/\mu_1) + E[n_2]\,(1/\mu_2)\,(1/M) + (1/\mu_1 + 1/\mu_2) + \frac{Pr[n_1>S]\,(1/\mu_2)}{M} \tag{9}$$

where

$$E[n_1] = \sum_{n_1=1}^{N} \sum_{n_2=0}^{N-n_1} n_1 p(n_1,n_2)$$

and

$$E[n_2] = \sum_{n_2=1}^{N} \sum_{n_1=0}^{N-n_2} n_2 p(n_1,n_2).$$

However, it should be noted that there is an inherrent inaccuracy in this model. Each I/O activity (and also CPU activity) invoked by a given request is conditionally related to some previous activity for that request; therefore, it is not possible to batch the RAD and Disc Pack services associated with that request. The only situations which allow for concurrent operation of the RAD and Disc Pack are those in which two or more users reside in core and simultaneously demand different I/O devices. That is to say, the "best case" estimates become more plausible as more user "slots" are available (providing there is no appreciable delay in initiating I/O action).

FIGURE 1

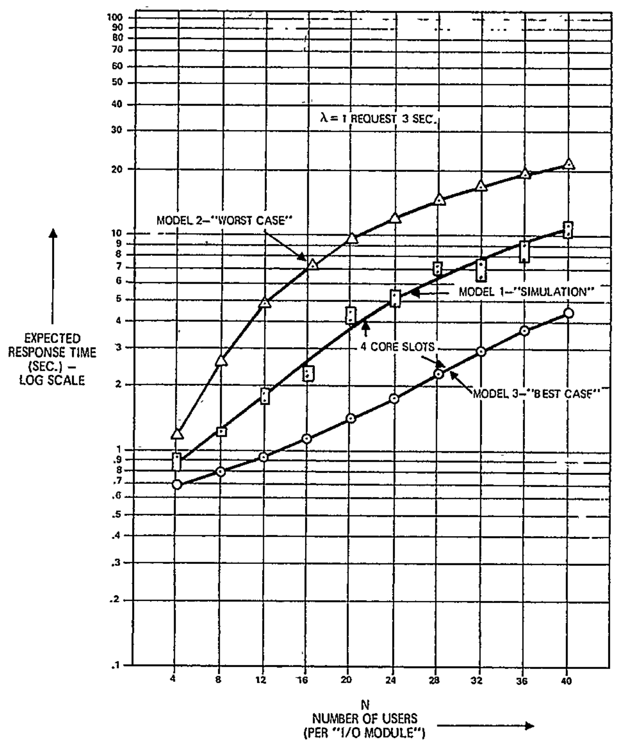


FIGURE 3

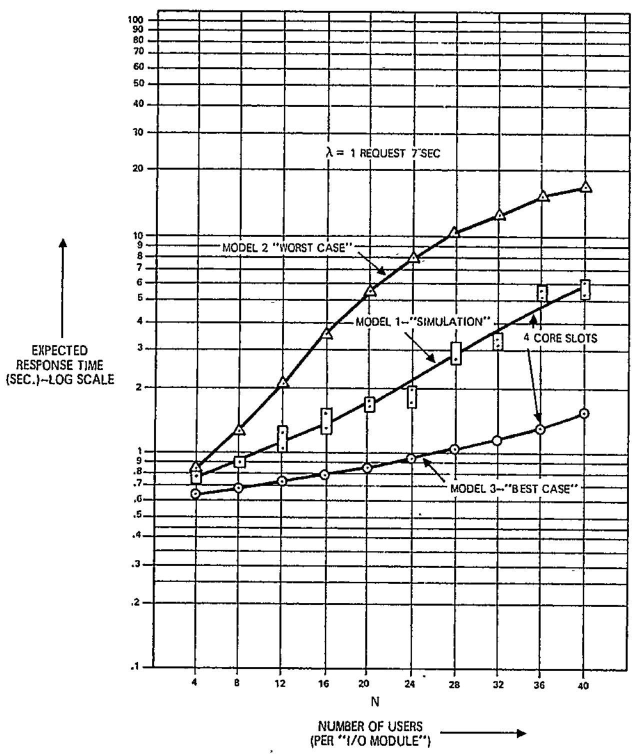


FIGURE 2

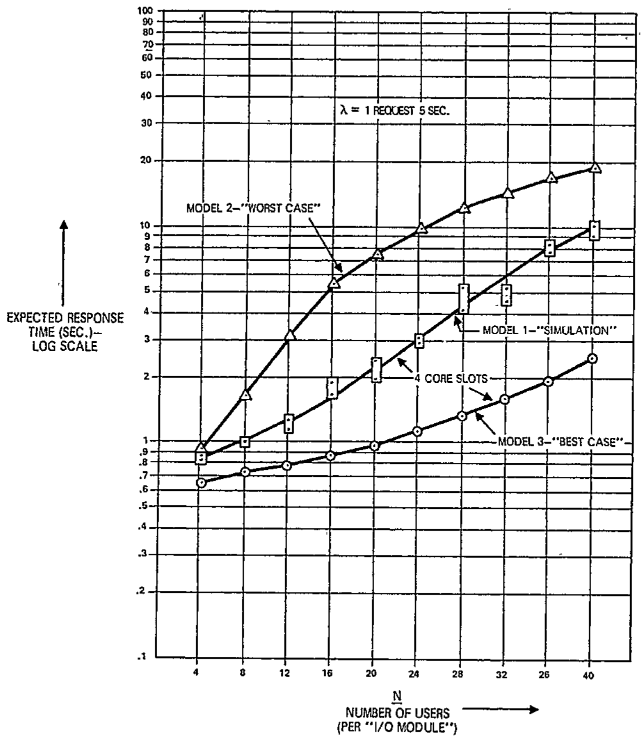


FIGURE 4

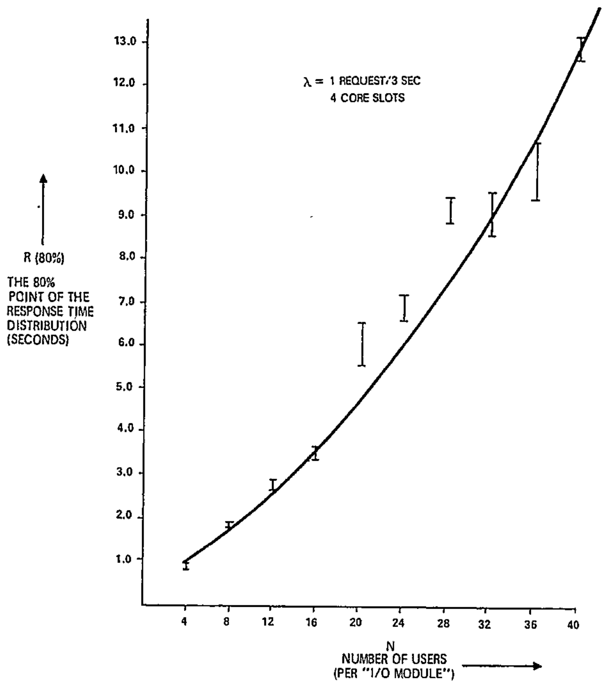


FIGURE 5

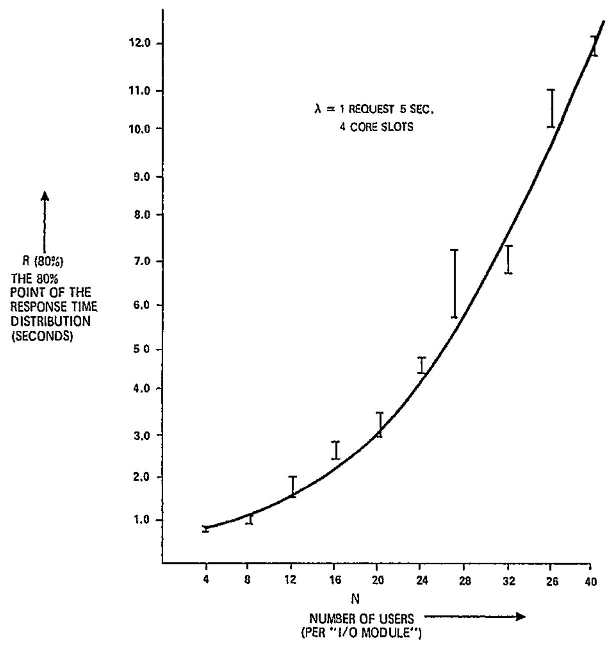
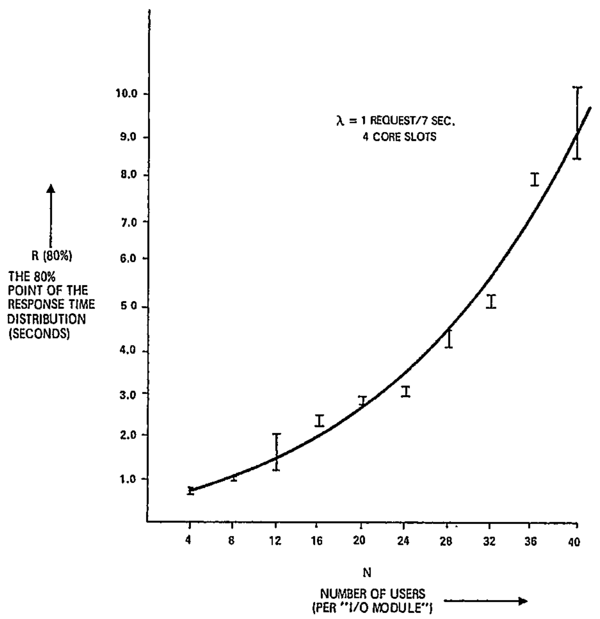
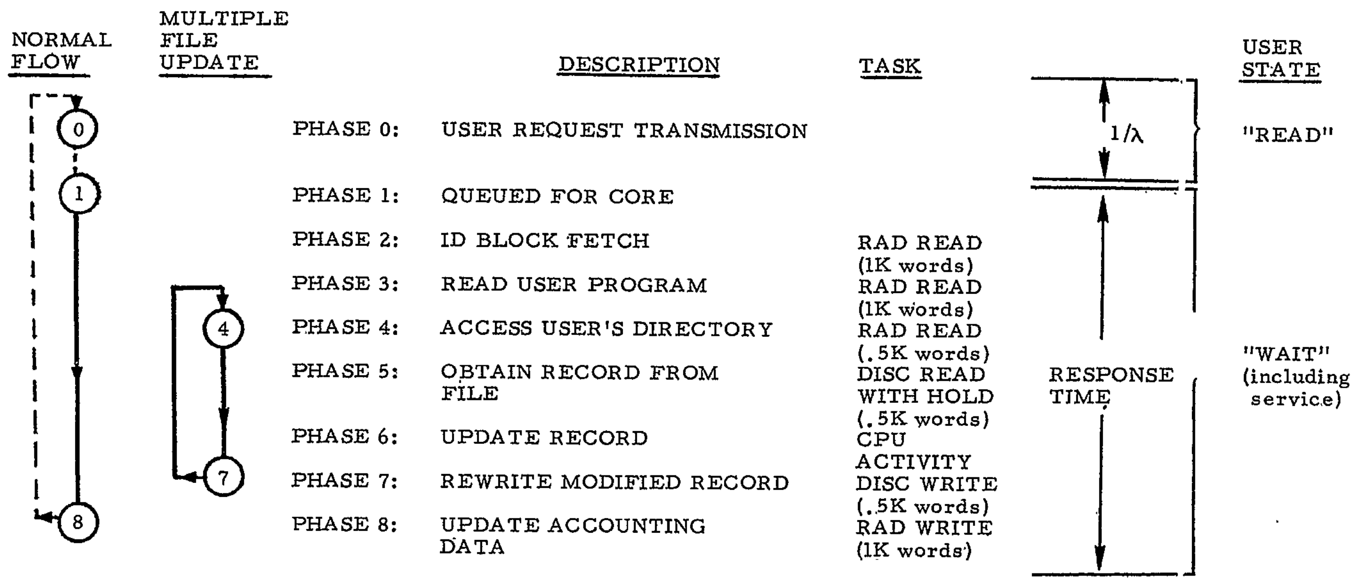


FIGURE 6





	DESCRIPTION	TASK
PHASE 0:	USER REQUEST TRANSMISSION	
PHASE 1:	QUEUED FOR CORE	
PHASE 2:	ID BLOCK FETCH	RAD READ (1K words)
PHASE 3:	READ USER PROGRAM	RAD READ (1K words)
PHASE 4:	ACCESS USER'S DIRECTORY	RAD READ (.5K words)
PHASE 5:	OBTAIN RECORD FROM FILE	DISC READ WITH HOLD (.5K words)
PHASE 6:	UPDATE RECORD	CPU ACTIVITY
PHASE 7:	REWRITE MODIFIED RECORD	DISC WRITE (.5K words)
PHASE 8:	UPDATE ACCOUNTING DATA	RAD WRITE (1K words)

FIGURE 7 -- PHASES OF USER SERVICE DURING A TRANSACTION