USE OF PERFORMANCE ANALYSIS STATISTICS IN COMPUTER SYSTEM SIMULATION*

by

P. R. Katonak
Savannah River Laboratory
E. I. du Pont de Nemours & Company
Aiken, South Carolina 29801

ABSTRACT

A general purpose job stream simulation model has been developed for routine use in evaluating hardware configurations and performance tuning a large scale multiprocessor computing system operating in a multiprogramming mode. The model is designed to simulate a variable workload of jobs statistically generated from attributes of the actual workload for any time period. Major functional modules include job-step generation, job classing, scheduling and step initiation, core allocation, cpu operation, and I/O activity. Hardware attributes (such as, available core and number of DASD control units), operating considerations (such as, number and priority class structure of initiators), and workload characteristics can readily be varied at model initialization time.

INTRODUCTION

Simulation has provided a method for effectively using performance analysis statistics obtained from accounting, hardware, and trace monitors, as well as experience gained through the use of a benchmark test stream, to analyze and project computer system performance and hardware requirements. In this approach, detailed timing data derived from the hardware and trace monitors are used as the basis for cycle timings of the operating system and devices within the body of the computer system model. Both multiprogramming (MVT) and multiprocessing are simulated with this model. The workload processed by the model is derived from accounting monitor statistics. The benchmark test stream was used as the input workload in initial MVT job scheduling models and was valuable in validating their accuracy.

This paper will be concerned with the use of accounting monitor statistics as the basis for probabilistic generation of the model workload. A general description of the overall model with the logic used in various modules is included in the Appendix. Additional descriptions of initial applications involving the use of the benchmark test stream can be found in References 1 and 2.

* The information contained in this article was developed during the course of work under Contract AT(07-2)-1 with the U. S. Atomic Energy Commission.

PERFORMANCE ANALYSIS PROGRAM

A continuous performance analysis program has been established in conjunction with the System/360-65 Multiprocessor computer facility at Savannah River Laboratory. This program provides the information base and analytical methods required to:

- Measure efficiency of operating system, user programs, and operational policies.
- Locate problem areas in operating systems or hardware.
- Assist in determining operational strategies.
- Detect changes in overall workload or demand for services.
- Determine equipment changes that most economically fulfill increased demands.
- Provide data for general information, special request management or government reports, equipment feasibility and justification proposals, and post-installation reports.

Data for the performance analysis program are collected from a variety of sources. Systems activity information is continuously accumulated by an accounting monitor routine developed at SRL. This monitor accumulates approximately 40 fields of data relating to every aspect of system performance activity which can readily be monitored for every step of each job processed through the system. The accounting monitor provides a complete data base from which statistics on CPU, core, and I/O activity can be summarized for any time period. A second internal monitor is utilized to accumulate statistics regarding the utilization of individual direct access storage drives over selected time periods.

A hardware monitor is periodically used for direct measurement of component utilization. This device can simultaneously monitor a number of system points (channels, control units, devices, or the CPU) providing wait or busy status data for each point. Control panel wiring permits analysis of combinational conditions (i.e., CPU wait and channel busy). This unit provides supplemental timing and utilization information that is not available from the accounting monitor.

Trace monitors which are background programs designed to continually gather highly detailed operating statistics have been developed. Information obtained through the trace analysis covers areas such as data set utilization, usage of operating system routines, and time to service I/O requests and is useful in determining contention between and utilization of system resources (operating system routines, I/O and auxiliary storage devices, data channels, etc.), and inefficiencies in specific programs. This type of information is not readily available through use of either accounting or hardware monitors.

A benchmark test stream consisting of a series of actual jobs with overall attributes matching workload has also been developed. This test stream consists of 37 jobs which require approximately 60 minutes of S/360-65 time when run serially. The benchmark has been used for determining operational strategies for an MVT operation, and in evaluating effect of system and hardware changes on operational throughput.

METHODS OF SPECIFYING THE WORKLOAD IN COMPUTER SYSTEM MODELS

The use which can be made of a computer system simulation model is highly dependent upon the manner in which the workload being received and processed by this model is defined. A variety of approaches have been taken for designating this workload in computer system simulation and mathematical modeling studies. One, which minimizes presimulation analysis of the actual workload, specifies the workload in terms of job interarrival rates plus service times.[3] An alternate approach which probably involves the greatest amount of detailed workload data utilizes trace monitor output as the basis for the defined workload.[4] The packaged computer system simulation programs such as CASE and SCERT define the workload in terms of file descriptions and type of operations.[5]

None of the above approaches was suitable for use in the computer system model designed for use in analyzing hardware requirements and operational strategies at Savannah River Laboratory (SRL). The monthly workload at SRL consists of several thousand nonproduction scientific jobs written primarily in FORTRAN and several hundred repetitive scientific and business production jobs written in both FORTRAN and COBOL, and is overall CPU bound. Because of the preponderance of nonproduction jobs and the continual changes in both volume and characteristics of these jobs, it was highly desirable to develop a model whereby the defined workload could be stated in terms of the workload currently being processed or projected on the actual computer system. Specifications of the workload in terms of interarrival rates and service times, although easy to define, did not provide an adequate indication of the variance in CPU time, I/O requirements, and core size between the various jobs processed. Use of the trace monitor data to define the workload was not considered to be practical because of problems in selecting representative programs from the overall workload for tracing, and the vast amount of additional detail which would be required within the model to utilize the trace data. The packaged simulation workload definition method, whereby files and operations are specified for each job processed, was not considered because of the wide variety of jobs being processed and the temporary nature of the bulk of the jobs making up the workload.

In addition to the variation in SRL workload characteristics on a month-to-month basis, the manner in which jobs are received and processed indicates a considerable difference in job-mix between operating shifts. For example, the bulk of the FORTRAN compile and GO jobs are received on the day shift and processed on a 60 minute or less turnaround basis. These jobs have below average core and CPU time requirements and are essentially I/O bound. The 4 p.m. to 12 midnight shift job-mix is made up of overflow jobs from day shift and a few routine CPU bound production jobs and thus has a different overall profile from the day shift. The 12 midnight to 8 a.m. shift workload is primarily devoted to long running CPU bound production jobs and, therefore, has a third profile. The relative job characteristics for the three shifts are illustrated in Figure 1. As a result, each shift must be considered independently in making studies concerning operating strategies or hardware requirements. Therefore each shift workload must be defined by its own unique set of attributes in the simulation model, and thus a simplified method of workload definition is an absolute necessity. To meet this objective, the routine accounting monitor analysis programs were modified to provide the basic workload statistics required by the computer simulation model.

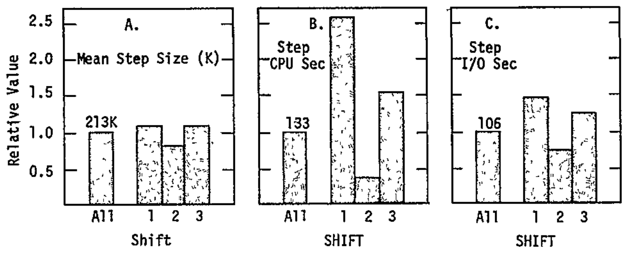


FIG. 1 RELATIVE DIFFERENCE IN MEAN STEP SIZE, CPU TIME, AND I/O TIME BETWEEN SHIFT 1 (12-8), SHIFT 2 (DAYS) AND SHIFT 3 (4-12) FOR TYPICAL MONTHS' WORKLOAD

JOB PROFILE STATISTICS

The primary attributes selected to define the workload include step core size, CPU time, and I/O time. This data provides the basis for simulation studies relating to system core requirements, CPU utilization, and I/O contention under varying operating conditions. These three areas have been of prime interest to date in the SRL S/360-65 configuration.

Figure 2 includes three simple frequency distributions of step characteristics for a typical day shift operation directly summarized from the accounting monitor analysis. Figure 2A, which illustrates the proportion of steps within each 60K step size interval, indicates that 75% of all steps on this shift are either 60-120K bytes or 180-240K bytes. Figure 2B, which indicates the percentage of total day shift steps which fall into various CPU time intervals, shows that 80% of the day shift steps utilize the CPU for less than 10 seconds. Figure 2C summarizes steps by % I/O interval and indicates that 60% of these steps are 80% or greater I/O. The term "% I/O," which is used in a number of charts, is the ratio of a steps estimated I/O time to the total estimated elapsed time (actual CPU time + estimated I/O wait time) for that step. (I/O time, unlike core size and CPU time, is not directly measured by the accounting monitor because of the problems involved in differentiating between a step's I/O time and its involuntary wait time in a multiprogramming environment. Step I/O time is therefore estimated from the I/O channel program counts (EXCP's) which are accumulated for each step by the accounting monitor. The relationship between I/O time and EXCP count was derived through correlation of step wait time and EXCP counts for several hundred serially processed jobs.

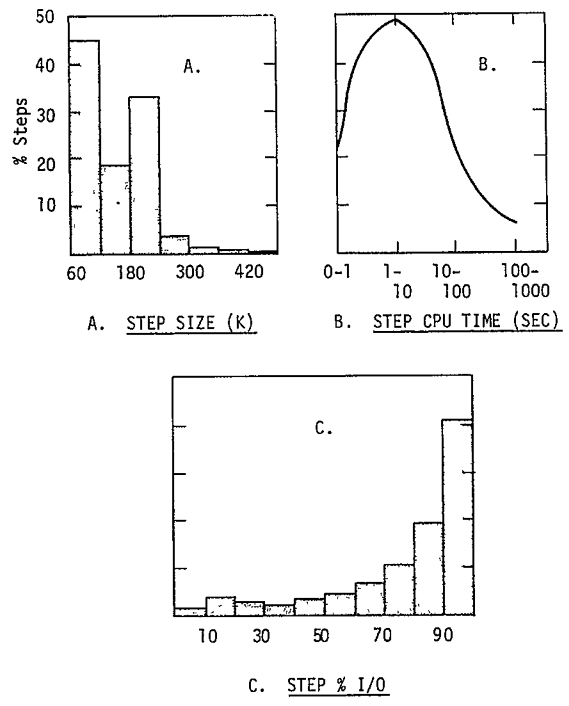


FIG. 2 SIMPLE FREQUENCY DISTRIBUTION SHOWING % STEPS WITH VARIOUS SIZE, CPU TIME AND % I/O FOR TYPICAL DAY SHIFT WORKLOAD

In addition to the simple frequency distributions of the percent of steps within various core sizes, CPU time, and % I/O ranges, it is also possible to relate these attributes to one another. Figure 3A shows the % of total CPU time for the day shift that is consumed by steps within various core size intervals. Figure 3B illustrates the % of elapsed time (I/O + CPU time) which is consumed by steps in each % I/O interval. These figures provide a significantly different impression from that obtained from the simple distributions in Figure 2. Although 45% of the steps fall within the 60-120K size interval (Figure 2A), these steps only account for 15% of the total CPU time (Figure 3A). Secondly, the 11% of the steps requiring 30% or less I/O (Figure 2C) account for 32% of the total elapsed time (Figure 3B).

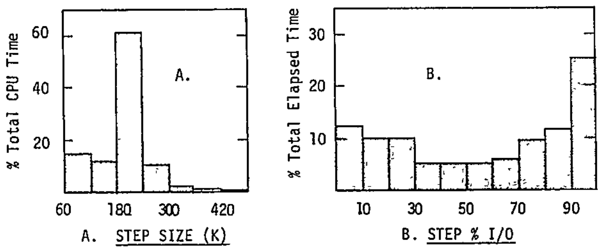


FIG. 3 DISTRIBUTIONS RELATING CPU TIME TO STEP SIZE AND ELAPSED TIME (SERIAL I/O + CPU TIME) TO % I/O FOR TYPICAL DAY SHIFT WORKLOAD

Unfortunately, random selection of step attributes based upon the simple frequency distributions as shown in Figure 2 will not provide attribute relationships matching those shown in Figure 3 for two reasons: CPU time is a function of selected step size, and I/O time is a function of selected CPU time in our job mix. The relationship between mean CPU time and step size for the basic data in Figures 2 and 3 is shown in Figure 4. It is obvious that independent assignment of step size and CPU time from Figures 2A and 2B would not result in the direct relationship between CPU time and step size as illustrated in the latter figure.

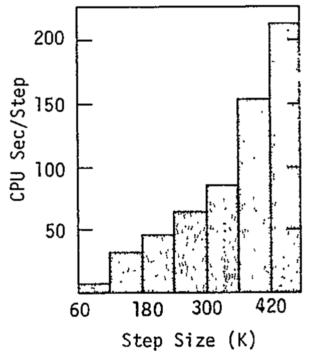


FIG. 4 MEAN CPU TIME FOR STEPS OF VARIOUS SIZES

The distribution of CPU times also has been found to vary within each core size interval. The cumulative distribution of steps by CPU time range for three core size intervals are shown in Figures 5A to 5C. Based upon these differences, it is apparent that separate frequency distributions relating % occurrences to CPU time must be used for each core size interval to properly relate these attributes. The method in which this is done within the model will be discussed in the following section.

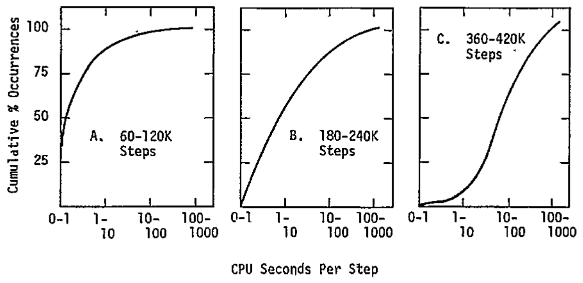


FIG. 5 CUMULATIVE DISTRIBUTION OF CPU TIME FOR VARIOUS STEP SIZE GROUPS

Although a relationship between I/O time and core size can also be found, a more significant correlation exists between I/O time and CPU time. For example, it can be seen in Figure 6 that step CPU time is inversely related to the step % I/O. Assignment of I/O time independent of CPU time would not result in a job-mix having this relationship.

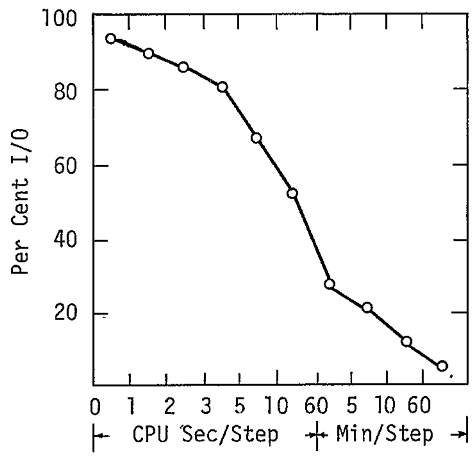


FIG. 6 AVERAGE % I/O FOR STEPS OF VARIOUS CPU TIME REQUIREMENTS FOR TYPICAL DAY SHIFT WORKLOAD

Similar to the variation in CPU time for various core size intervals, as shown in Figure 5, the distribution of steps with various I/O requirements fluctuates by CPU time group. Several examples illustrating distribution of steps in each % I/O interval are shown in Figure 7. The actual estimated I/O time for the step can be calculated from the assigned CPU time and % I/O.

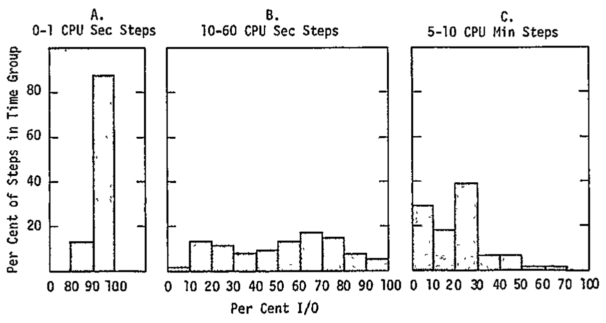


FIG. 7 PERCENT OF STEPS (WITHIN SAME CPU TIME RANGE) IN EACH PERCENT I/O INTERVAL

USE OF PROBABILITY DISTRIBUTIONS WITHIN WORKLOAD GENERATION MODULE

The various probability distributions pertaining to job and step characteristics are specified in function tables for use in the workload generation module of the computer simulation model. A complete workload description consists of 24 function tables relating to step size, CPU time, and % I/O, and two tables relating to job interarrival times and number of steps per job. A list of the types of function tables is given below:

Description of Functions	No. of Tables
Distribution of Job Interarrival Times	1
Distribution of Steps per Job	1
% of Steps in each Core Size Interval	1
Distribution of Step Sizes within each Interval	7
Distribution of CPU Time in each Core Size Interval	7
Distribution of % I/O for each CPU Time Interval	8

Figure 8 illustrates the points in which accounting monitor analysis function tables are employed in the workload generation module. A number of required control blocks within this module are not shown in order to simplify the flowchart. The source and use of each function entity is described as follows:

- GENERATE JOB TRANSACTION: The actual clock time each computer job enters the system is stored in that job's accounting monitor record. A probability distribution function relating the % of occurrences to the seconds between successive jobs can be defined for the time period to be simulated. This function is used to generate jobs at approximately the actual interarrival time (IAT) rate. Although the example indicates the use of only one IAT distribution function during the entire simulation period, multiple generate blocks with independent functions could be used if warranted by actual arrival rate fluctuations.

- SPLIT INTO STEPS: The distribution of steps per job is defined in a function table. The SPLIT block creates copies of the job transaction based upon the step function and these new transactions are identified as job-steps. The job transaction is then temporarily held in a USER CHAIN while the step attributes are selected and assigned to each job-step transaction.

- ASSIGN SIZE GROUP: Using the "% of steps in each core size interval" function, a core size group or interval number is assigned to each step (i.e., group number 1 for core size range between 60-120K to group number 7 for 420-480K steps). This number represents the core size interval which the step is in and serves as a pointer to appropriate function tables used in subsequent core size and CPU time assignment blocks.

- ASSIGN CORE SIZE: Seven functions define the actual distribution of expected core size for steps within each group. Using the core size group number as the argument, the step is assigned a core value obtained from the appropriate distribution of step size function as determined from the size group pointer.

- ASSIGN CPU SECONDS: Separate functions also define the probability of different CPU times for steps within each size group. Using the core size group number as a pointer, step CPU seconds are selected from the appropriate CPU time function table.

- ASSIGN % I/O: A separate function relating probability of occurrences for each % I/O is defined for each CPU time interval. Using the previously assigned CPU time as a pointer, step % I/O is selected from the appropriate % I/O function table. Step serial elapsed time is equal to CPU Sec/(1-%I/O/100), and I/O time is equal to elapsed time minus CPU time.

The remainder of the generate module summarizes CPU time, I/O time, and maximum core sizes for all steps within each job. This data is passed to the job transaction after all step transactions are created for the job.

Validation of the probability distribution method was performed by comparing a variety of statistics concerning the simulated workload with accounting monitor analysis data from the actual workload. Several of these comparisons are illustrated in Figure 9. Simulation data was generated for an average day's workload for one 8-hour-day shift operation. Actual data is taken from analysis of the computer workload for the day shift for an entire month. As indicated, while a minor variation exists in the distributions, the general structures are identical. Comparisons of significant averages for the same time period (Table I) indicates general agreement in overall statistics with difference being greatest for those averages with the smallest number of samples.

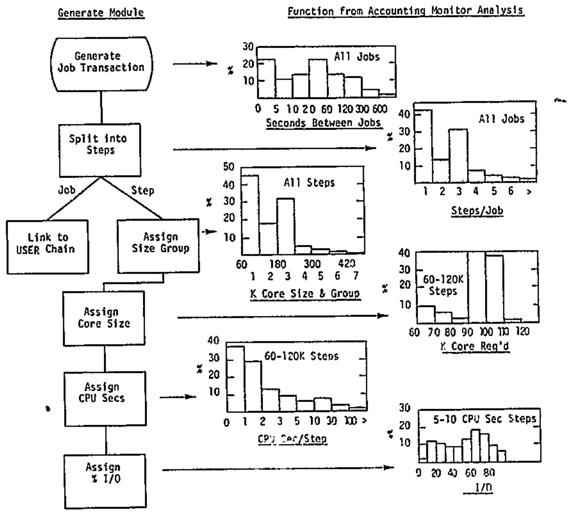


FIG. 8 USE OF ACTUAL WORKLOAD STATISTICS TO CREATE SIMULATED WORKLOAD

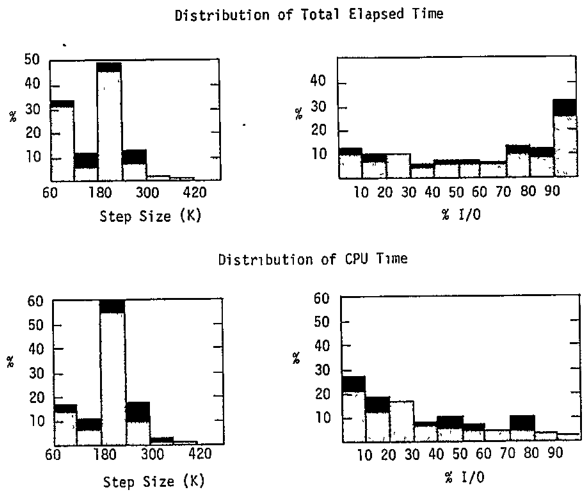


FIG. 9 COMPARISON OF ACTUAL VERSUS SIMULATED WORKLOAD CHARACTERISTICS (SHADED AREA REPRESENTS DIFFERENCE)

TABLE I COMPARISON OF SELECTED AVERAGES

Attribute	Mean Value		% Total
	Workload*	Simulation**	Steps***
Step Size	152K	151K	
Time/Job			
CPU sec	48.9	48.9	
I/O sec	125.0	125.0	
% I/O	61	61	
Elapsed Time, sec			
60-119K Steps	29.6	30.0	45
120-179K Steps	70.4	42.1	19
180-239K Steps	72.0	83.3	31
240-299K Steps	117.2	143.0	3
300-359K Steps	102.8	162.0	2

* Average for 1-month day shift operation.
** Average for 8-hour day shift operation; workload generated from 1-month day shift statistics.
***% total steps by size range identical for workload and simulation.

VALIDATION AND USE OF THE SIMULATION MODEL

Validity of the entire simulation model has been tested through the use of the basic test stream. Initially the model (see Appendix) was designed without consideration for I/O channels, control units, and devices. In this phase, I/O was handled either on a no contention (all I/O for every step was available when required) or a full contention (only one job step could seize the I/O facility at a time) basis.

Five configurations of S/360-65 single and multiprocessor systems were evaluated with the simulation model using test stream job characteristics for the model workload. The actual test stream was then actually run on each of the configurations. A comparison of observed and simulation results is shown in Figure 10. The I/O module was later expanded to include existing channel, DASD control unit and DASD device activity. Simulation results for this model ran within 5% of the observed results shown in Figure 10.

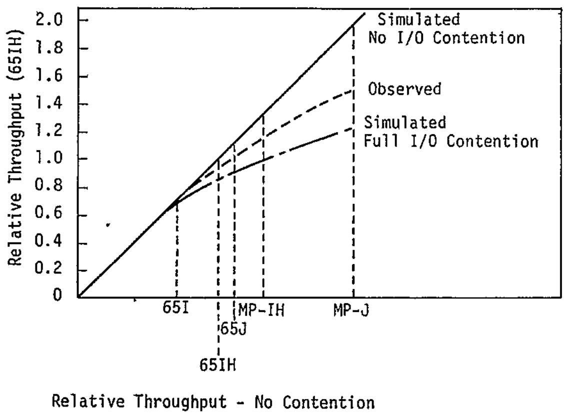


FIG. 10 RELATIVE THROUGHPUT OF VARIOUS S/360-65 COMPUTER SYSTEMS FOR CONSTANT WORKLOAD

The primary use of the model to date has been to study future hardware requirements necessary to meet projected workloads. Availability of the model has permitted estimation of the relative throughput of various size and speed systems for different workload projections. In addition, the model has permitted an effort to be made to analyze the effect of utilizing additional DASD or higher speed control units or storage devices in different combinations to determine the most reasonable hardware investment strategy. For example Figure 11 illustrates the simulated relative system power for two types of theoretical workloads and three combinations of I/O equipment. In both cases, I/O channel contention was assumed negligible. As illustrated, the configuration with a high speed drum appears to provide greater throughput over a system with only direct access storage (DASD) units when the workload is CPU bound than when the workload is 50% CPU time and 50% I/O time.

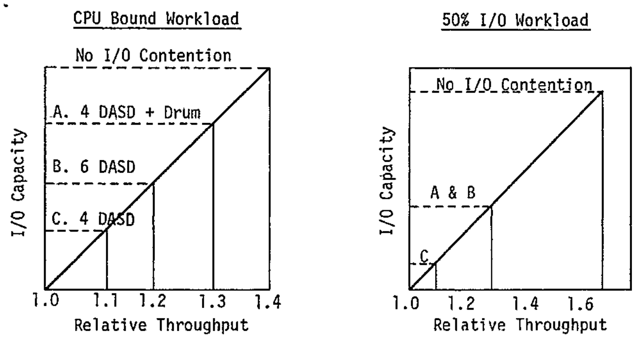


FIG. 11 RELATIVE SIMULATED THROUGHPUT FOR SYSTEMS WITH VARIOUS I/O CONFIGURATION, FULL CONTENTION CONSIDERED RELATIVE POWER 1.0

In the area of operational strategy, studies have been made concerning job classing structures, initiator priority settings, and core allocation methods.

SUMMARY

This model provides a very basic approach to simulating a computer system and its associated workload based upon information which is routinely accumulated as part of the standard computer system performance analysis program. A number of areas within the model require further refinement. These include incorporating more realistic methods of distributing Operating System CPU and I/O overhead, which is currently spread evenly between each CPU or I/O burst, and development of more detailed I/O modules. Being modular in design, the existing framework will permit such modifications to be made without requiring a complete revision of the entire program.

ACKNOWLEDGMENT

Work done by W. R. Hartshorn, L. F. Zimmerman, and J. L. Kilpatrick of the SRL Computer Operations Division in development of the accounting monitor and related analysis programs has provided the data base which made this simulation approach possible. The assistance of F. C. Fortune who developed the CMAP routine and participated in the model design stage was invaluable in providing the foundation for subsequent development. The helpful critique of this paper by T. E. Bell of The Rand Corporation was also appreciated.

REFERENCES

1. Katonak, P. R. and Fortune, F. C., Sr. "Role of System Simulation in a Continuous Computer Performance Analysis Program." *Ninth Annual Conference of Southeastern Region ACM, St. Petersburg Beach, Florida* (1970).
2. Katonak, P. R. "Performance Analysis of the Multiprocessor/65 through Simulation." *SHARE XXXV, Montreal, Canada* (1970).

3. Hoffman, E. G. "Studying Multiprogramming Systems with Queuing Theory." *Datamation, 13:6,* 47 (1967).
4. Cherg, P. S. "Trace-Driven System Modeling." *IBM Systems Journal, 8,* 280 (1969).
5. Pomerantz, A. G. "Predict Your Systems Fortune: Use Simulations Crystal Ball." *Computer Decisions, 2(6)* (1970).

APPENDIX I

DESCRIPTION OF BASIC JOB SCHEDULING MODEL

The basic job scheduling model simulates a workload consisting of a predefined combination of jobs and job-steps with varying core sizes, CPU, and I/O requirements through a S/360-65 computer system in a multiprocessing and/or multiprogramming mode of operation. The basic model consists of approximately 400 cards of which approximately half are used for function, savevalue, and table definitions. The CPU running time for simulating a given time period will vary proportionally with the I/O burst time defined by the model user. When the actual measured mean I/O cycle time (30 ms per burst) is used, the simulation CPU time on the S/360-65 will be approximately 20% of the actual CPU running time for the same set of jobs. If the simulation I/O cycle time is increased by a factor of four, CPU running time will drop to 5% of the actual CPU time for the period being simulated. The model runs in 140K bytes. Less than one man year has been spent in the simulation modeling effort by computer operating systems personnel assigned to this project.

As shown in Figure 1, the model consists of seven separate modules. The simulation is written in GPSS[1] but calls one FORTRAN subroutine via a HELP block. Individual modules can be expanded or modified without altering other modules, as long as the appropriate savevalues and transaction parameter values are always processed in a standard manner. The basic transaction, except for two cases, represents either a job or job-step. Both types of transaction carry the job number identification as one of the parameter values. In addition, a separate parameter is used to distinguish between each step in a multiple step job and the job master transaction. The two cases where a transaction represents something other than a job or step are for system initialization and simulation run timing purposes and are created by GENERATION blocks not shown on Figure 1.

The basic function of each of the modules shown in Figure 1 is as follows:

1. Generate

The generate module creates jobs and job-step transactions representing the workload being simulated. A description of parameter contents for these job and job-step transactions is shown in Figure 2. Three alternative methods have been used to date to represent the simulated workload. These include (1) predefining job and step attributes in function tables and using the ASSIGN block to transfer the attributes to appropriate job and step transaction parameters, (2) creating an input JOBTAPE of job and step transactions directly from the accounting monitor record for a given time period, and (3) generating a series of transactions from probability distribution tables created by accounting monitor analysis of an actual workload. The probability distribution technique is discussed in detail in the body of this paper. The job-step transaction is linked to a user chain at the end of the generation module.

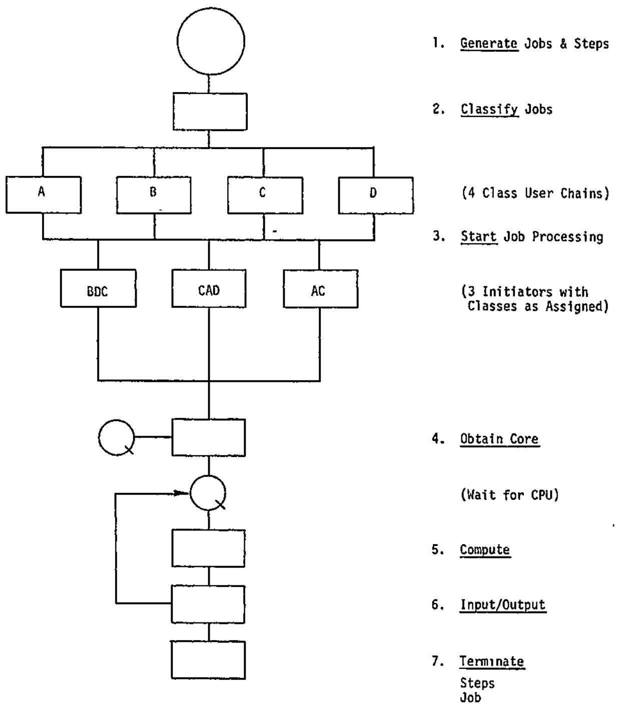


FIG. 1 BASIC JOB SCHEDULING MODEL (MULTIPROGRAMMING WITH A VARIABLE NUMBER OF TASKS (MVT))

Transaction No.^{1}	Job Type^{1}	% I/O^{1}	CPU-I/O Cycles^{1}	CPU Seconds^{1}
	I/O Seconds^{1}	CPU Burst MS^{1}	Class^{2}	Initiator^{3}
Priority^{3}	Core Req'd^{1}	Core Start^{4}	Core End^{4}	Core Return Code^{4}
	I/O Cont'l^{6}	I/O Cycle Time^{1}	Print Lines^{1}	
			Step No.^{1}	Job No.^{1}

FIG. 2 JOB AND JOB-STEP TRANSACTION DESCRIPTION (DIGIT INDICATES MODULE CREATING DATA)

2. Classify

Each job transaction is classified according to the algorithm being used or under consideration with the system. The purpose of the classification scheme is to control the mix of jobs being processed at any one time from a wide variety of available jobs and to optimize CPU, core, and I/O utilization. Classification schemes can be based upon job core requirements, % I/O, CPU time, elapsed time, amount of output or a combination of these or other factors. Using GPSS, classification is accomplished by using the TEST block to evaluate appropriate job transaction parameters.

An example of the simulation logic for a classification module based on core size, serial processing time, and output volume is shown in Figure 3. The assigned job class is stored in a transaction parameter for reference in later steps. Job transactions are linked to a USER CHAIN (user chain blocks for four classes A, B, C, and D are represented by respectively labeled blocks in Figure 1) associated with the assigned class.

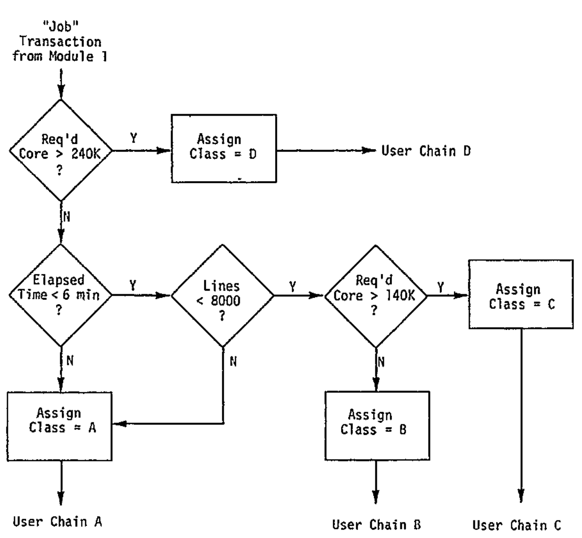
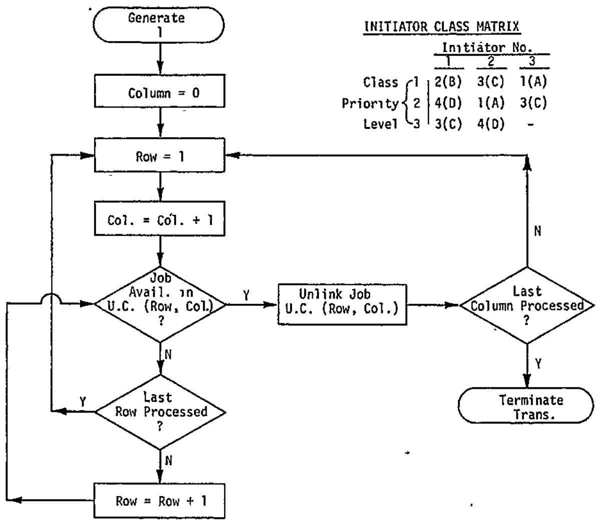


FIG. 3 JOB CLASSIFICATION LOGIC BASED ON TIME, LINES, AND CORE

3. Start

The START module is divided into two sections. The first, which operates only at the beginning of a simulation, activates each initiator with the highest predefined priority class job available in the various job class user chains. For example, the START module (Figure 1) would attempt to UNLINK a class B job for the first initiator. In the event there were no class B jobs available, START would attempt to find a "D" or as a last resort, a "C" class job. In the event that none of the classes were available, the initiator would stay idle until a new job of one of the matching classes entered the system. After an attempt is made to attach a job to each initiator, the transaction which operates Phase I of the START module terminates and subsequent job initiation is controlled in the TERMINATE module.

GPSS logic for the START module (Phase I) is shown in Figure 4. As illustrated by the initiator class matrix, a MATRIX SAVEVALUE is predefined with the classes as assigned to each initiator. (An equivalent numeric code must be used in the actual matrix.) This table can be changed to evaluate the effect of alternative classing or initiator setting schemes on system throughput for a given workload and configuration.



		Initiator No. 1	2	3
Class	1	2(B)	3(C)	1(A)
Priority	2	4(D)	1(A)	3(C)
Level	3	3(C)	4(D)	-

FIG. 4 START MODULE OPERATION (PHASE I) The Predefined Initiator Class Matrix Points to Appropriate Class User Chains of Available Jobs. The Unlinked Job Transaction Passes to Phase II of the Start Module.

The second phase of the start module controls the interaction between job and job-step transactions. After a job is unlinked from the class user chain and assigned to an initiator, the job transaction UNLINKS the next matching sequential job-step transaction from the step user chain. The assigned initiator number is placed in a parameter in both the job and job-step transactions. The job transaction is then relinked to the appropriate class user chain and the job-step transaction progresses to the next module.

4. Core

The core requirement for each step is specified in a parameter in terms of K (1024) bytes. This core must be provided in continguous blocks for each step, and the starting address for an assigned core block may vary for multiple steps within the same job. To simplify programming, a FORTRAN subroutine called CMAP was written for controlling core allocation within the simulation. The total amount of system core available for user programs is established in the simulation initialization phase through the use of a SAVEVALUE. This savevalue is passed to CMAP which establishes the upper limit of core available for user programs.

The job-step transaction parameter containing the amount of core required and a code indicating whether the core is being requested or returned is passed to CMAP. This routine, which keeps track of the absolute locations of the core being used, assigns required core if available, passes back a nonavailable parameter code when the core is requested but not available, or removes the core blocks from active use status when the step is completed and core is returned. Job-step transactions are temporarily placed on a USER CHAIN when adequate core is unavailable.

5. Compute

The compute module loops each job-step transaction through the CPU(s) for the number of cycles required to complete the CPU seconds specified. The basic model time increment is one millisecond. The number of CPU-I/O cycles is derived by dividing the computer I/O time for each step by the average I/O cycle time which is predefined in a SAVEVALUE. The average CPU burst time is subsequently derived by dividing predefined job-step CPU seconds by the number of calculated CPU-I/O cycles. The resulting CPU burst time is then adjusted to include Operating System CPU overhead which is obtained from the accounting monitor statistics. The variance between CPU burst times which has been observed through the use of trace analysis is recognized through the use of an exponential function modifier to the CPU ADVANCE time. Separate CPU modules have been written for single and multiprocessor systems.

6. Input/Output

The degree to which I/O is simulated can vary greatly. This, of course, is dependent upon the area of interest and importance for a particular study. The degree of detail included in various I/O modules ranges from none, where I/O requests are handled under either no I/O contention or full I/O contention conditions between requests, to the case where activity for I/O channels, DASD control units, and disk drives are included. A configuration of a multiprocessor model with expanded I/O facilities is given in Figure 5.

The simple approach (no contention/full contention) provides useful estimates of the effect of CPU power and core size on system capacity. The extended approach is useful in studying the effect of I/O capacity or data set distribution on throughput for a given workload and central processing system. The latter module requires additional simulation input data such as the distribution of data set usage among various control units and devices. This information can be derived through accounting monitor data analysis and incorporated into the model through the use of function tables.

Modules have also been developed which ignore device and channel activity. These modules, which reduce the simulation running time,

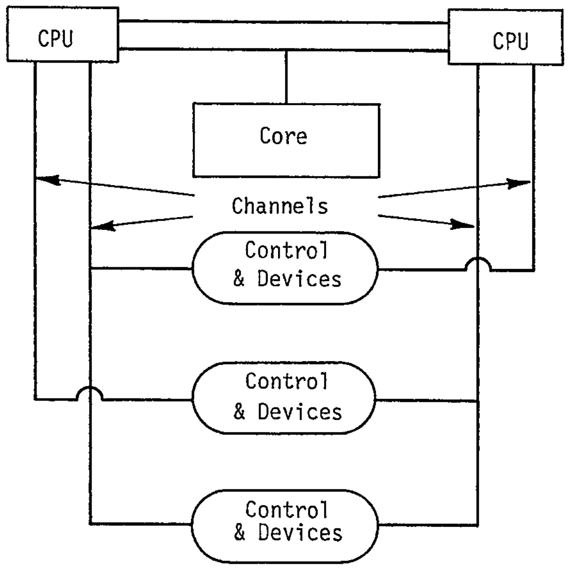


FIG. 5 COMPONENTS INCLUDED IN MULTIPROCESSOR MODEL WITH I/O

are valid for cases where device and channel contention is negligible. As indicated under compute, the I/O cycle time is predefined by the user. The average actual cycle time for our job-mix using 2314 Direct Access Storage Devices is approximating 30 ms per I/O burst, as measured through detailed trace analysis. It has been determined that artificially inflated cycle times can be used in the model without affecting comparative simulation results. This is of importance because the simulation running time is directly proportional to the specified I/O cycle time. The average I/O cycle time used within the model is internally adjusted to reflect relative access rates of devices other than IBM 2314's, when the I/O configuration includes such devices. The measured operating system I/O overload as measured by the accounting monitor is evenly distributed over each I/O burst.

7. Terminate

The function of the TERMINATE module, besides terminating steps which have completed required CPU and I/O cycles, is to maintain continuity of operation within the system. As previously discussed, the first phase of START activates each initiator after a few jobs enter the system. The terminate module is designed to keep initiators active, if possible, by using the terminating jobs to pull the next available job into the system. The initiator class matrix used in START is referenced by TERMINATE to unlink the proper class jobs. This logic is shown in Figure 6.

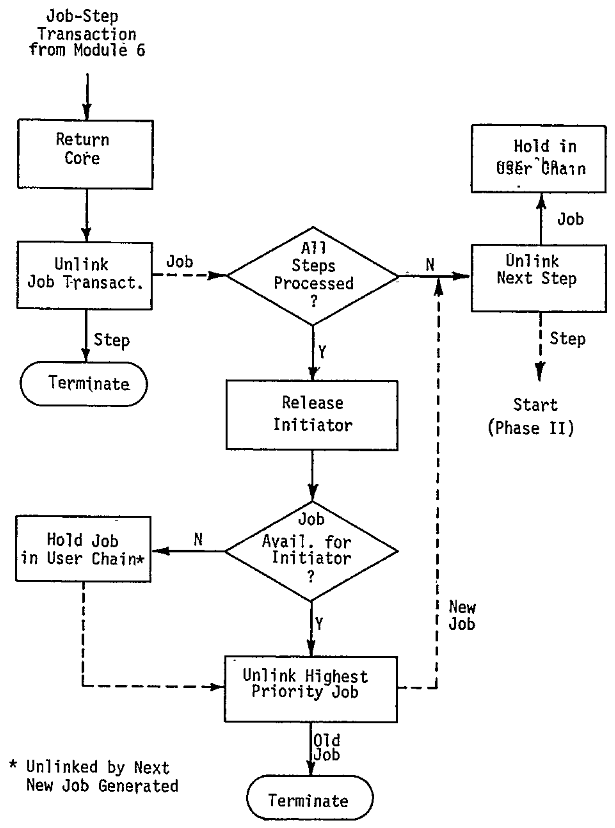


FIG. 6 TERMINATE MODULE

REFERENCE

1. IBM, "General Purpose Simulation System/360 Users Manual," Form H20-0326.