Validation of a trace-driven CDC 6400 simulation

by J. D. NOE and G. J. NUTT

University of Washington
Seattle, Washington

INTRODUCTION

A computer center typically faces questions of how to deal with a growing load in the face of tight financial constraints and with the need for lead time for planning ways to meet the demand. The needs may be met by altering the equipment configuration, changing priority algorithms and other features of the operating system, controlling the time scheduling of various classes of job load, or perhaps shifting load from one machine to another if the center is large enough to have such a capability. It is often difficult to gather enough data and insight to show the direction these changes should go and to support the decision to do so. One useful set of tools is provided by simulation backed up by measurements required for validation. However, it is all too common to find that fear of interruptions of the computing center’s service to user, combined with an overworked systems programming staff, prevents insertion of the desired measurement probes into the operating system. Then one is restricted to measures that can be derived from the normal accounting log, or “Dayfile”, and to software probes that can be injected in the guise of user’s programs. In spite of these limitations, useful results can be obtained, and this paper describes validation of a simulation model of a multiprogramming system, the Control Data 6400, making use of these restricted measurements.

The level of detail included in the model affects the degree of confidence one can place on the results of the simulation. It is necessary to represent the system accurately at the level of detail needed to support the predictions of interest. This implies not only enough detail to include the system resources that may be altered, but also enough detail to faithfully represent the interaction of these resources. In a complex system this becomes an important factor since the change in one area may have surprising effects in another part of the system. This argues for inclusion of finer detail.

A counter argument relates to high cost, which is a frequently-discussed difficulty with simulation. However, cost is a relative matter. There is no point in spending money to simulate or measure a system unless the resulting changes lead to savings well in excess of expense; but, when one is dealing with systems costing on the order of fifty to one hundred thousand dollars per month, plus supporting expenses equal or greater to that amount, a relatively small increase in efficiency can more than pay for a useful amount of simulation and measurement effort. It is still important to hold down the cost of simulation and one way of doing so is to avoid more detail in the model than is necessary.

These points, of course, lead to a compromise in the choice of what to include in such a model, and as will be explained later in the paper, we found that the process of validating the model and examining the sensitivity to various parameters gave useful insight on the level of detail that was necessary.

THE CONTROL DATA 6400

Many descriptions of the Control Data 6000 Series exist and there is no point in repeating a lengthy discussion (see Thornton, Control Data, MacDougall, Noe). Only the salient features of the system will be described here in order to remind readers of its general structure.

The Control Data 6400 is a multiprogrammed, file-oriented system, using a central processor, ten peripheral processors and twelve channels, a central memory of up to 131K 60-bit words is available to the system and to the user programs. Each peripheral processor has 4K memory (12 bits per word) for its local use. The peripheral processors communicate with the central processor through central memory and they take over many system tasks, such as communicating with the I/O equipment. Concurrent users’ programs within the
A variety of operating systems for the CDC 6400 exists. The one on which this simulation is based is SCOPE 3.2 with some modifications that are local to the University of Washington Computer Center. One of these modifications is the disk image (DI) option. This option, when used on a tape request control card, causes the system to first search the disk directory to see if that tape has been copied onto disk. If so, the disk file is used. If not, the operator is requested to mount a tape, which is then copied to disk and retained there for some period of time, (e.g., eight hours). This modification is advantageous in an environment in which many calls may be made for the same tape during any given day, providing disk saturation is not a factor. Another modification is the addition of a staging queue. Jobs that require magnetic tape are held on disk in this queue until the operator has mounted the reel on the tape drive; then they are allowed to proceed into the input queue where they wait for central memory and for a control point.

A feature of the operating system that has been necessary to this simulation and validation study is the system Dayfile. This file accumulates data on individual jobs, such as time into the card reader, time of exit from the input queue, amount of central processor time used, time into the output queue and time off the system. This file, which was originally designed for accounting purposes, provides a significant amount of information that can be used for describing the job load in the trace driven simulator and it will be discussed more fully in subsequent paragraphs.

THE APPROACH

The general approach taken in this study was to develop a simulation at the level of detail that shows the interaction between tasks and system resources. The Dayfile was analyzed to obtain trace data, i.e., to show how the actual job load on any given day made use of the system resources. Validation runs were then made by driving the simulation model with trace data from various days to provide a range of load conditions.

Provisions are included in the simulation model to represent the input job load either as individual jobs (to allow driving the model with the trace data) or as job classes (to drive the model with distributions representative of groups of jobs). Only the former method is reported on in this paper, because it is restricted to a discussion of the model and its validation. The use of job classes will be discussed in a separate paper reporting on use of the model to explore effects of load variations.

This work was done under a number of constraints that limited some of the things we would like to have done. It was important not to interfere with the service being provided by the Computer Center. In particular, it was not possible to insert probes in the system monitor, or to dedicate a peripheral processor to the measurement role, as has been done in work by others (see Stevens, Control Data, and Sherman). As a result, we were unable to include disk statistics among the parameters used to describe jobs, except in a very general way. Disk accesses were not modeled in detail although their overall effect on job flow was accounted for in the simulation through the time interval a job stayed at a control point.

In spite of these constraints, a considerable amount of information was available through the Dayfile and inasmuch as the constraints under which we were operating are not uncommon, it may be of interest to others to see how much can be done without the freedom to alter the system programs for measurement purposes.

DESIGN OF THE SIMULATION

In general structure, the simulation parallels the operating system and allows parameter choices representing a range of available operating system and hardware options. The level of detail of the model is restricted to focus on system resources that may be altered, omitting wherever possible the inclusion of resources for which no decisions may be made. Execution speed of the model is approximately fifty times faster than the real system, i.e., one hour of real system operation is represented by 70 seconds of operation of the model (see Nutt for a detailed description).

The model is written in ANSI FORTRAN and
its general structure is similar to BASYS (see MacDougal¹⁰). ANSI FORTRAN was used rather than a specialized simulation language for two reasons: first it made the program available on both the CDC 6400 and on the XDS Sigma 5 (for which we had no discrete system simulation language compiler); second, since this was done in a learning environment in a university, the use of FORTRAN gave an opportunity to learn more about the details of queue handling and sequence timing that are obscured by the automatic features of simulation languages.

The model is trace-driven, i.e., jobs are represented by the resources they require. It then provides data about jobs that are dependent on system efficiency, such as turnaround, job dwell-time at control points and dwell-time in input and output queues. It also provides data on system performance such as queue lengths, the number of control points in use, CPU utilization, and the amount of central memory occupied by all jobs at sample intervals.

The system actually simulated is shown in Figure 1. The notation is that of the modified Petri net, as introduced in Reference 5. To generate a simulation, it is essential to have a clear representation of the system being modeled. The modified Petri net notation is used because it shows the computer structure under the influence of the operating system.* For the purposes of the present description, it should be adequate to note that the vertical lines are transitions. When the necessary input conditions (denoted by incoming arrows) exist, the transition “fires” causing the output conditions to come into being (denoted by outgoing arrows). Arrows marked at either end with the symbol “O” denote EXCLUSIVE OR; arrows with no symbol attached denote AND ... i.e., two or more are required. Note that different symbols may exist at the two ends (input or output) of an arrow.

As an example of the interpretation of Figure 1, “JOB IN CARD READER” represents the existence of one or more jobs in the card reader. Whenever the “CARD READER AVAILABLE” and “JOB IN

* At the state of development of the notation used here, the system representation by Petri nets is qualitative. Further work is in progress to add quantitative features to the modified Petri nets.
TABLE I—Characteristics in Simulator Job Array

Characteristics that are inputs to the model (Columns in Job Array)

1. Central Memory (during compilation)
2. Central Memory (during execution)
3. Central Processor Time Limit
4. Central Processor Time Actually Used
5. Peripheral Processor Time Used
6. Job Priority
7. Magnetic Tapes Used
8. Disk Image Files Used
9. Number of Cards Read
10. Number of Cards Punched
11. Number of Lines Printed
12. Total Rollout Time
13. Number of Times Job Rolled Out
14. Time Job Entered Card Reader

Characteristics that may be calculated outputs or (Columns in Job Array)

15. Time of Entry into Staging Queue
16. Time of Entry into Input Queue
17. Time of Assignment to Control Point
18. Time of Entry into Output Queue
19. Time Job Vacated Machine

CARD READER conditions are true, the associated transition is fired, causing a job to be removed from the card reader and to be placed in the staging queue ("JOB IN STAGING QUEUE"). Note that the staging queue can accept jobs from the card reader or the remote input, but not both simultaneously. The use of EXCLUSIVE OR when leaving the staging queue is interpreted as follows: When "JOB IN STAGING QUEUE" is true and no tape is required, the next transition fires, altering job status to "JOB IN INITIAL QUEUE." Or (exclusive) when a "tape job" is in the staging queue and a tape is assigned to it, the transition fires.

In making use of the model, it is important to be aware of system resources that are not included. Most important are channels, peripheral processors, and the details of disk access (disk file activity is accounted for only in length of dwell-time of jobs at a control point*). A most important reason for not including channels and peripheral processors is that the number of each available is fixed, i.e., addition of channels and peripheral processors is not among the decisions that might be implemented by the Computer Center. Therefore, it was pointless to examine simulation results dependent on changes in the number of these units. However, it is important to bear in mind that before deciding upon some system changes suggested by the simulator, one should make measurements to see if the proposed change is likely to cause saturation in channel or PPU usage, thereby shifting the bottleneck out of the "range of view" of the simulation.

A principal feature of the simulation model is the method used to describe the job load driving the system. Because this method provided flexibility, it aided validation of the model as well as extrapolation into future situations of interest. Visualize an array in which each row describes a job during the period it is active in the system. As jobs exit from the system, they are removed

TABLE II—Outputs from Simulator

Job Statistics:

Mean and standard deviation for the group of jobs streaming through the simulator are provided for all the characteristics listed in Table I. In addition, histograms of the distributions are printed for

- Central Memory (average of compilation phase and execution phase)
- Processor Times (central and peripheral)
- Length of time in Rollout status
- Time on Control Point
- Time in Output Queue
- Time in combined Staging and Input queues
- Turnaround Time (from card entry to vacating machine)

Complete trace on each job as it progressed through the model was provided.

Queue Statistics:

- Card Reader Queue
- Staging Queue
- Input Queue
- Central Processor Queue
- Output Queue
- Rollout Queue (i.e., jobs awaiting Rollin)

Resource Utilization: Accumulated busy-time for

- Card Readers and Punch
- Line Printers
- Magnetic Tapes
- Central Processor
- Control Points

* Specifically, an average disk access time was used, as measured by a software probe entered as a user's job. The number of accesses per job was approximated by dividing the job's peripheral processor time by the average access time. The measurement program was written, and the data gathered, by Geoffrey C. Leach, University of Washington.
from the array to provide room for new jobs to be added. Therefore, the array size need be large enough only to handle the currently active jobs rather than all the jobs entered within a given simulation run. The columns in this array represent the trace characteristics of each job, and their contents are listed in Table I. Some of these columns (1 through 14) must be pre-specified to describe the job; they may contain positive numbers representing the actual value of the parameter, or may contain negative integers indicating that a value is to be obtained by the simulator system from a distribution identified by the negative integer. This feature allows driving the model with either actual trace data extracted from the Dayfile of the real system, or from data approximated by distributions that represent a job mix, or possibly some combination of the two.

Columns 15 through 19 in the job array show characteristics that may be pre-specified (again from actual values or from distributions), or may be calculated by the simulator. This flexibility is helpful during validation of the model since it allows one initially to drive the model with predetermined data, thus force-fitting it to behave like the real system; and then one can back off as confidence is gained, and let the simulator provide more and more of the parameters. This has proven to be a very useful tool during model validation. It is also useful for taking care of “peculiar” jobs, such as control programs that handle terminals and other I/O and which appear to the system as long-term “user” jobs rather than as part of the operating system.

Based upon these input statistics for individual jobs and on the simulator’s calculation of the time each job requires to go from point to point in the system, data are gathered on the overall operation of the multi-programmed system and the nature of the aggregate results provided are listed in Table II.

VALIDATION OF MODEL

General

The procedure followed to validate the model is shown schematically in Figure 2, which starts from the tape containing the Dayfile, i.e., the CDC 6400 normal record of the day’s operation, originally conceived as an accounting tool. The Dayfile does not exist in a format suitable to drive the simulator, since entries are ordered according to clock time and information on a unit job is scattered throughout the file. The program DFFPREP extracts the proper parameters for each job, creating a new file of properly formatted trace data selected over the time period being examined. As DFFPREP reads the Dayfile and creates input for the simulator, it analyzes the data to provide statistics on system performance and on the jobs entered into the system. The trace data are used to drive the simulator, entering only the input descriptions of the job into the simulator and asking it to provide simulated system performance.

The process just described sounds deceptively simple. In actual fact, getting the simulated model to agree with measured performance over a range of parameters and job mixes proved to be a complex and time-consuming process. The authors suspect that this is a typical situation and accounts for the fact that the extensive literature describing simulation of computer systems is rather sparsely populated by papers on validation.

On the other hand, this validation phase proved to be very valuable, not only in terms of the essential point of building confidence in the realism of the simulator’s performance, but also in the better understanding of the system and of the model that was gained during the validation process. For example, during validation it became clear that it was necessary to extend the model to include jobs that bypassed part of the system, e.g., a remote batch capability that bypasses the card reader queue and the output queue had

![Diagram](image-url)
grown in importance in actual system use so that it was necessary to incorporate it. Also, the sensitivity of the system to the manner in which magnetic tapes were handled became apparent as did the effect of the disk image (DI) option. Some errors in the model, in the control point area and in central memory field length variation, were detected and corrected as a result of validation. It was found necessary to approximate the number and duration of disk accesses in order to properly fragment each job's use of the CPU. It also became apparent that it is important to be able to precondition the state of the simulated system when starting it in the middle of the day's operation. In other words, the start-up transient in the simulation, which seems to be on the order of fifteen minutes to one-half hour of simulated time, can obscure the results if one attempts to simulate two or three hours of mid-day performance without preloading the system.

In retrospect, it is tempting to say that some of the points that needed changing should have been included in the model as originally constructed. Yet, we would argue against going overboard in this direction. The question is one of performance sensitivity to a given system feature. If one initially attempts to include so much detail that all significant factors are likely to be in the model as originally constructed, one is certain to end up with a number of unnecessary details that only serve to lengthen the program writing, debugging, validation, and execution time. The authors believe it makes far more sense to construct a model along reasonably simple lines, containing flexibility for injecting needed changes, then adding features when the validation process shows them to be necessary.

Interestingly enough, some of the most difficult things to validate in the model involved human actions and human judgment, particularly the time required to fetch and mount magnetic tapes, and the criteria used by the operators in deciding when to place a program in rollout status (i.e., removing it from contention for core memory and for the CPU) were subject to wide variations. On the other hand, inclusion of these factors was necessary for realism and will provide ways to test some operational procedures and policies that might be applied.

Validation data

Validation and alteration of the model finally converged to a point where reasonable confidence in the model was established. To illustrate this, data are presented that cover four different job mixes and job densities chosen to represent two extremes in computer usage and an intermediate case. One extreme was near the end of an academic quarter, during which the system was used heavily by students trying to finish problems and term projects. The other extreme was between academic quarters when the student load had decreased significantly, leaving a predominance of research jobs and system development jobs running on the machine. The mixes ranged from a low of 37.5 percent student jobs up to a high of 75.7 percent student jobs. The job densities covered a range greater than 3.5
to one in jobs per hour, and included densities near the current record day for this installation.

Figure 3 summarizes the mean values of delay times as the jobs progressed through the system. It shows comparison values for the real and simulated system for each of the four load conditions. The lower curve (A) in Figure 3 shows average delay times through the combined staging and input queues. The next higher curve (B) adds control point dwell times to the staging and input delays. The top curve (C) adds output queue times to the cumulative delays and thus represents turnaround times, i.e., time between entering the card reader and ejection from the line printer.

A word of warning about Figure 3: Do not interpret it as "how delay time varies as a function of rate of job flow"; instead, interpret it as "for the specific cases (job mix and density), how do the real and simulated systems compare?" The reason for this is that the higher densities shown in this figure correspond to cases in which a higher proportion of student jobs were run. These jobs on student account numbers are predominantly short jobs with three or four pages of listing and less than ten seconds central processing time (see Hunt10).

The set of points corresponding to the highest job density (and highest student job proportion) proved initially to be very troublesome. Upon closer investigation the difficulty proved to stem from tape jobs. For some reason the standard deviation in tape fetch and mount time for that day was unusually high (greater than three times the mean value). For the other days, the standard deviation ranged from 1.63 to 2.25 times the mean value. Therefore, for that troublesome day, the tape jobs were pre-specified in the model, to allow validation of its performance on the remaining, more normal job load.

In Figure 3 the middle curves (B) are critical, since control point occupancy time directly influences memory and the number of control points available. During validation overall results were found to be most sensitive to agreement in this area. Figure 3 shows only the mean values. The variation in these values is quite great and comparison of the distributions of results is important to the validation. Figures 4 through 7 com-
pare the simulated and actual distributions for the various queues and for overall turnaround time for the job load corresponding to 169.4 jobs per hour and 64.5 percent student jobs. The agreement between simulated and actual mean values for this case, as shown in Figure 3, is within 9 percent through the staging and input queue, 11 percent when the control point dwell time is added, and 4 percent for overall turnaround time. The agreement in the distributions, as shown in Figures 4 through 7, shows that the model is operating quite well over a wide range in values. Figures 8 and 9 show similar comparisons between the number of control points and the amount of memory in use in the simulation and in the actual system. These are sample data with the samples not synchronized between the real and simulated systems. A sample was taken approximately every ten seconds and the total run during which comparison was made lasted for two hours.

The agreement of the distribution for this particular load case was representative for the other cases and they are not repeated here. The goal was to verify the model to within 10 to 15 percent. This was generally achieved over the range of loads and over a range of parameters within the model. There are some exceptions, but they are generally understood. For example, the simulation of the 169.4 jobs/hour and 64.5 percent student jobs assumed that there was a constant availability of two line printers. Analysis showed this not to be the case. In the actual system only one line printer was operating initially. Twenty minutes later, another line printer was turned on, and 25 minutes after this event, 3 line printers were operational for a short period of time. Also there were cases where some very unusual system development jobs appeared in the real system and they were not accounted for in the simulation; however, these cases are understood to the point where they do not detract from the confidence in the model to predict performance for the major types of jobs that are important in this environment.

SUMMARY

At this point, the model is ready for use in experiments to examine the effects of changes in load and system configuration, and these are proceeding.

Perhaps the most important point established so far is that it is possible to gather enough data to validate a useful model, even under the stringent conditions imposed by a non-interruptible computing service center. There is no doubt, though, that special software and hardware measurements would be useful. For one thing, this would provide validation data for a more detailed simulation of the disk system.

The model as it now stands can be used to predict the effects of changes in job arrival density or in job nature. Effects of changes in job CP use and memory use (estimated and actual), number of cards, lines printed, tapes and disk image usage can be observed. Some configuration changes that can be studied include those in central memory, central processor speed or numbers, line printers, card readers, tapes, and number of control points. Scheduling algorithms for these resources can be varied, and the effects of a variety of operational policies can be observed, such as tape-handling, rollout, and control of types and schedules of jobs submitted to the system.

REFERENCES

1 J E THORNTON
Design of a computer: The Control Data 6600
Scott Foresman & Co 1970
2 CDC 6400/6500/6600 reference manual
    Control Data Publication number 60100000 1965
3 CDC 6400/6500/6600 SCOPE 3.2 reference manual
    Control Data Publication number 60189400 1968
4 M H MacDOUGALL
    Simulation of an ECS-based operating system
    SJCC Proceedings 30 pp 735-741 1967
5 J D NOE
    A Petri net description of the CDC 6400
    Proceedings ACM Workshop on System Performance Evaluation Harvard University pp 362-378 1971
6 D F STEVENS
    System evaluation on the Control Data 6600
    IFIP Proceedings C34-38 1968
7 CDC 6400/6500/6600 partner reference manual
    Control Data Publication number 60222500
8 SHERMAN ET AL
    Trace driven modeling and analysis of CPU scheduling in a multiprogramming system
    Proceedings ACM Workshop on System Performance Evaluation Harvard University pp 173-199 1971
9 G J NUTT
    SIM6000 program description and use
    University of Washington Computer Science Group Technical Report number 71-07-05
10 M H MacDOUGALL
    Computer system simulation: An introduction
    Computing Surveys 2 No 3 pp 191-209 1970
11 E HUNT G DIEHR D GARNATZ
    Who are the users?—An analysis of computer use in a university computer center
    SJCC Proceedings 38 pp 231-238 1971