Title: Computer Job Scheduling: A Simulation Project

Course:
Year: 1993
Author(s): J. Munz

Report No: P93055

COMPUTER JOB SCHEDULING
A SIMULATION PROJECT

Jim Munz

EMP -- 9355

Computer Job Scheduling
A Simulation Project

Jim Munz
Instructor: Dick Deckro
EMGT 510/610
Manufacturing System Simulation
December 2, 1993

Abstract: This is a final report of a project investigating the use of ProModel Manufacturing Simulation Software as a tool for capacity planning and performance evaluation of a large scale multi-processing computer system. One significant finding of this project is the ease with which ProModel was able to simulate the computer environment. This software may have provided too many alternative ways to model the various features and characteristics of the system. It also proved to be easy to modify, which allowed me to examine several different approaches to simulating the computer environment and testing different approaches to solving the scheduling problem.

Introduction

This is the final report of a project investigating the use of ProModel Manufacturing Simulation Software as a tool for capacity planning and performance evaluation of a large scale multi-processing computer system. This project benefited from the help of the Technical Services Group at Multnomah County. They served as the "experts" by providing a conceptual understanding of the process of computer job scheduling, gathering data to determine the parameters of the system and critiqueing the model at every stage.

The findings of this project add very little to our current knowledge of the performance of the County's mainframe computer. This is the result of attempting to squeeze a reasonably sized project into a one quarter class schedule. A significant finding of this project is the ease with which ProModel was able to simulate the computer environment. This software may have provided too many alternative ways to model the various features and characteristics of the system. It also proved to be easy to modify which allowed me to examine several different approaches to simulating the computer environment and testing different approachs to solving the scheduling problem.

Problem Description

Multnomah County is a $440 million corporation with over 3500 employees and offices in 107 locations. While the County has a number of turnkey, departmental computers and over 40 local area networks (LANs) supporting nearly 2000 personal computers, most of the data processing necessary to support the major service areas is done on a large IBM compatible mainframe computer. This central computer center operates 24 hours a day, seven days a week and provides support to the criminal justice system, for health care delivery, real and personal property assessment and the collection and distribution of nearly a billion dollars of tax revenue.

There are a number of different kinds of work which are processed through the center. About 40% of the work of the center is in support of the County's wide area network and the processing of online transactions necessary to conduct the day-to-day business of the different departments and divisions. The remaining 60% of the work is done in the form of batch processing of computer jobs. Most of these batch jobs are scheduled by a computer operator and are processed after the close of work each day. This scheduled workload may begin as early as 4:00 p.m. In all cases, it must begin early enough to allow the results of the work processed each day to be available either online or in the form of printed computer listings before the

start of work the following day.

The focus of this project is the 500 plus jobs which are submitted by user agencies and applications programmers. These jobs represent about 15% to 20% of the work which goes through the center. All of these jobs are processed between 8:00 a.m. and 4:00 p.m. each weekday. These jobs present a special problem to the data center. Unlike the nightly production jobs which are scheduled months or sometimes even years in advance, the daily work is not predictable. The number of jobs and the composition of those jobs varies from hour to hour and day to day. There are a number of different classes or types of jobs submitted. The rapid processing of several of these classes is essential if the work of the County is to be done in a timely fashion. Failure to provide rapid turnaround for most of these job classes will reduce the efficiency and effectiveness of many County programs and will result in reduced quality and decreased customer satisfaction. The goal of this project was to identify ways the system could be modified to provide better system throughput and improved customer satisfaction.

Historical Approaches to Computer Performance

Until the introduction of the multi-processing operating system in the mid 1960s, the modeling of computer performance and evaluation was not a problem. From their introduction in the late 1940s until the mid 1960s, computers were a single queue with finite capacity and a single server with known parameters. If you wanted to investigate the capacity of an existing computer or estimate the performance of a different computer, the analysis was easily done analytically. The introduction of the multi-processing operating system expanded the complexity of the problem by allowing for multiple servers with different performance characteristics.

During the 1970s the issues of computer capacity planning and performance evaluation became important topics. Large mainframe computers were multi-million dollar investments and most corporations were interested in getting the most out of their investment. Lucas[1] describes a number of these techniques such as timings, instruction mixes, benchmarks, synthetic programs, software/hardware monitors, etc., none of which proved very satisfactory. Most of these approaches depended on developing a standard job or package of jobs which was tested on each computer being evaluated.

---

[1] H. Lucas, "Performance Evaluation and Monitoring." Computing Surveys, Volume 3, Number 3 (Sept. 1971), pages 79-91

A primary failing of most of these approaches was their inability to model the actual work which would be performed by the computer.

A number of individuals began to experiment with the use of queuing network models for capacity and performance analysis. Willy Chiu[2] developed a cyclic queue model which allowed various resources in the system to be treated as independent service facilities. Boyse and Warn[3] modeled the central processing system using what they describe as a machine repair model. This was a finite population model with multiple tasks (machines) to process (repair) and a single server or repairman (central processing unit). This model was widely known as the "straightforward model" of Boyse and Warn. It required a number of simplifying assumptions about how computer processing actually took place. The most significant of these had to do with their failure to include in the model a number of queues which existed in the actual computer systems they were modeling. The model was expanded to allow up to three servers (a degree of multi-programming of 3) but this required a number of other assumptions to

---

[2] Willy Wai-Yee Chiu, Analysis and Application of Probabilistic Models of Multiprogrammed Computer Systems, unpublished doctoral dissertation, University of California, Santa Barbara, December 1973

[3] J. W. Boyse and D. R. Warn, "A straightforward Model for Computer Performance Prediction," ACM Computing Surveys, Volume 7, Number 2, June 1957, pages 73-79

keep the calculations as simple as possible.

The capacity planner and performance evaluation specialist had a limited number of options available to obtain the optimum performance out of the existing computer system and, at the same time, justify the need for a newer, larger computer. He/she could: 1) adopt the simplifying assumptions required of the "straightforward model" of Boyse and Warn; 2) write a computer simulation model using FORTRAN or one of the scientific languages available or; 3) learn to use the General Purpose Simulation System (GPSS) which had been developed by IBM in the early 1960s. GPSS is a discrete-event simulation language which allows you to model a task or customer which must pass through one or more queues to complete a process.

During the late 1980s and 1990s, the interest in simulation for computer capacity and performance evaluation has declined. My impression is that the rapid decreases in cost per MIP (million instruction per-second) have reduced the need to get the maximum performance out of the existing computer system. Buying the next larger computer system may be less costly to the organization than using the technical services staff to squeeze the last bit of performance out of the existing configuration.

The Simulation Model

The simulation model which was developed as a part of this project had to address several key issues.

1) The project was focused on the 500 batch jobs which are submitted between 8:00 a.m. and 4:00 p.m on weekdays. There are a number of scheduled production jobs which are submitted by the operator beginning at 4:00 P.M each day. These jobs are placed in the queue and are handscheduled into initiators. It did not seem appropriate to attempt to model this partially controlled environment. It was also the consensus of the technical services staff that batch throughput was not a performance problem after 4:00 P.M.

2) The batch jobs which are submitted during the modeled hours fall into a number of categories or classes. Each of these classes of jobs has a unique set of characteristics and processing requirements. These classes include:

CLASS A: These jobs are submitted by applications programmers who are designing and testing new computer applications or are maintaining and enhancing existing applications.

A one week sample indicated that the average number of class A jobs which arrived between 8:00 a.m. and 4:00 p.m. was 205; the range was from 168 to 210.

CLASS H: These jobs are submitted by application programs. There are a number of computer applications which allow a user at a online terminal to request a complex process to be executed as a batch job and the results printed on a remote printer at the user's location. The listings which are produced are business critical in that the normal work process at that location depends on having the listing available within minutes. The average number of class H jobs during the period was 143; the range was from 98 to 178.

CLASS Q: The County operates three data base management systems, each supporting a variety of applications areas. Class Q jobs are ad hoc reports which are submitted by users of a particular data base management system (DBMS). The average number of class Q jobs during the period was 105; the range was from 96 to 127.

CLASS R: Class R jobs are ad hoc reports which are submitted by users of a particular DBMS. By identifying the jobs which are being processed against a particular DBMS by a job class, we can control the number of jobs accessing a particular DBMS at any point in time and can stop all processing against that DBMS if necessary. The average number of class R jobs during the period was 63; the range was from 45 to 76.

These four job classes make up 99% of the jobs which are processed during the 8:00 a.m. to 4:00 p.m time period. Each of these classes of jobs arrives in the system in a different time pattern and has different characteristics. The input of the arrival side of the simulation model was set up as four queues, one for each of the four classes of jobs being processed and each of which had different arrival characteristics.

The actual system being simulated uses ten servers or initiators. Each initiator is capable of processing one job at a time and a single initiator can be set to service one or more queues. The order in which the queues are searched for available jobs is a variable parameter which can be changed dynamically by the computer operator. For instance, the first initiator could

be set to classes "HAQR." Each time the initiator was free, it would search the H queue for available jobs. If it found one, it would accept the job. At the completion, it would again search the H queue. If the H queue was empty, the initiator would then move to the A queue to look for jobs to process. It would continue through the queues until it found a job or finished its search.

To avoid resource contention in a large scale computer system, only one job can be in initiation or termination at a time. To account for this, a single resource identified as "INITERM" was defined in the system. This resource is responsible for the initiation and termination of all tasks in the system.

Finally, a single output queue was established to receive all jobs when processing is completed. In most large mainframe computers this would be a number of printer queues which would receive and route jobs to the proper destination. A diagram of the final simulation model is included on the following page.

There are a number of key features of the computer system which were not included in the simulation model. There are three other classes of jobs which could be processed. They accounted for less than 1% of

the total jobs and several days during the period none were submitted. There didn't seem to be any discernable pattern to when these jobs were submitted. Since these jobs averaged nearly 5 minutes of CPU time when they ran, a more complete simulation should include these other jobs.

It should be understood that we are modeling a subsystem of a much larger system. The batch job processing which we are modeling is competing for resources against several thousand online terminals which support normal business processes. In some instances, the batch jobs will be accessing the same data being used by online terminal users. This creates a resource contention which may increase the processing time of a particular job or class of jobs. To the extent that the data which was used to construct the model included these effects, they are accounted for in the model. Since these factors vary independently of any of the factors which are modeled, a more complete model would identify the nature of the relationship between online processing and batch throughput and include this as a parameter in the model. This would allowe the online workload to vary to determine the total impact on batch throughput.

A final factor not included in the model is the actual detail of the job processing. In the model used for the paper, job processing was modeled as

three separate steps; job initiation, job computation and job termination. Each of these parts of the process was assigned a parameter based on the sample data gathered. In a multi-processing computer, job computation is a complex process which requires access to direct access storage devices (DASD) which involves a separate set of queues, contention for memory and page data sets, and a number of other factors which could be included in a more complete simulation of computer job processing.

The parameters which were selected for the model were developed from a number of sources. Most large computers provide some sort of facility for collecting data about the processing which takes place in the computer. In a large IBM computer this data is collected in a number of SMF and RMF records which are generated at various points in the processing of work through the computer. These parameters were changed several times during the construction and validation of the model. Since the model building process was iterative, the descriptions of the probability distributions selected will also identify some of the decisions which were made during the verification process.

Arrival Times - Each of the four job classes was set up as a separate input queue and each class had its own arrival parameters. The initial attempt to

identify a distribution to describe the arrival time for each class used a uniform distribution which allowed for the correct number of arrivals in each class spread evenly over the eight hour period. After examining a larger data sample (see appendix B) which included a number of days, it became obvious that the distribution of arrival times was bimodal with large peaks around 10:00 a.m. and 3:00 p.m. and a low during the lunch hour. The final model used the ProModel Cycle Table to develop hourly arrival numbers based on five day hourly averages; the arrivals were assumed to be normally distributed during the hour.

Service Times - Each class of jobs has three separate processing parameters; job initiation time, job processing time and job termination. Job initiation and termination require the use of the single resource, "INITERM," and are captured in the SMF5 job termination record as the CPUSRB time. The time is not separated into initiation and termination data. After some amount of discussion with technical services staff, job termination was modeled as the same value for all jobs processed through the system. The differences in CPUSRB times for the different classes of jobs are captured in the job initiation parameter. Within the ProModel system, these two times are modeled using the time it takes the INITERM resource to move a job from the queue to the service location and the time it

takes to move the completed job from the service location to the output queue. It is assumed to be a constant for all jobs in the same job class.

The job processing times for each class are represented in the system as Weibull distributions. The initial model was constructed using a Lognormal distribution. Lognormal has the property of a sharp peak at the mode, the $\lim f(x) = 0$ as x approaches 0 and, it will distribute a large number of observations between zero and one. The initial attempts to fit the data to a lognormal curve using the software package I had available provided unsatisfactory results. During verification of the model, other distributions were tried but without much success.

Visual inspection of the data indicated that each of the job classes had between 5% and 10% of its execution time values which were outliers. These were values that were 10 to 100 times the value of the mean of the distribution. During verification, I decided to break three of the four job classes into two separate groups of data. For classes A, Q and R, there are two different distributions. The largest number of the jobs processed, 90% to 95% depending on job class, are represented by a Weibull distribution. The outliers are represented by a Triangular distribution. The Triangular distribution was a simplifying assumption. It assumes an arbitrary cutoff

point of 10 minutes. Any value below 10 minutes was included in the Weibull; any value equal to or greater than 10 minutes was included in the Triangular distribution. The Triangular distribution uses 10 seconds as the lower limit, the largest value observed during the period is the high value.

The parameters of the Weibull distribution were estimated by hand and do not produce satisfactory Chi-square values. This is far and away the weakest part of the simulation model. Given more time, I would have dumped the execution data to disk and used the Unifit program to develop better parameter values for the distributions.

Using two different distributions to model the jobs being processed through the model required probability logic in the arrival queues to identify each job as small or large. It also required the use of If-Then-Else logic to choose the appropriate distribution for the processing time of each job.

How well does the model represent the environment being simulated? During verification of the model, I benefited from the advice and counsel of the technical services staff. It was generally agreed that the model captures most of the essential features of the environment being simulated. While more detail could be added, the model seems to operate conceptually in the

same way that most of the staff feel the computer works.

It was during this time that I learned that experts don't always tell you everything you need to know on the first pass. The original concept for the model was sketched out on a white board and everyone seemed satisfied with the result. As the model was constructed and people begin to watch the visual model in operation, new and often contradictory information would come to light. This turned the model building process into a kind of trial and error process where I would build or modify the model as people told me things they assumed that I should have known or that they were sure they had told me earlier.

One of the things that was brought up late in the validation process had to do with the number of job classes and how the initiators were defined. I was originally informed that I should create as many initiators for job class A and H as necessary to process all jobs in a timely fashion. Most if not all initiators would process class H jobs and search the class H queue first. I was also informed that Q and R jobs were single threaded through the system so that multiple batch jobs would not contend with online teleprocessing using the same DBMS.

The student model allows only 10 locations. My original model was constructed using 4 input queues (one for each job class) one output queue and five initiators. Each initiator was defined to have multiple job classes, and would search for class H jobs first. One initiator would then move to class Q and another was assigned to class R. All initiators would process class A jobs if there were nother classese available. The model produced less than satisfactory results. The wait times for A, Q and R jobs was much longer than the actual data indicated.

After some amount of discussion with the "experts," the actual configuration of the initiators was obtained. The system initiators were defined in the following way:

Initiator	Classes
1	H
2	HR
3	QH
4	RH
5	AR
6	AQ
7	QA
8	HA
9	HA
10	HA

In fact, there were multiple initiators for both job classes Q and R. I rebuilt

the model using 10 initiators defined in the same way that they were in the real system.

Experimental Design

Using the 10 initiator model, a number of simulation runs were processed using different seed values for each of the runs. The average values of the simulation runs were evaluated against the observed values for the system being modeled. The average values for **arrival times** of each of the job classes fell within the observed values for the real system. At no time did the simulated values fall outside the range of the observed value and the average of the simulation runs for all classes was within one standard deviation of the observed value, I could have performed a Chi-square test but the values were good enough that I didn't that it was necessary.

The results of the simulation provided **better throughput and lower wait times** for three of the job classes than the actual system. Intuitively, the model seems to capture most of the major elements of the real system. I suspect the ability of the model to produce "better" results than the real system is a function of the distributions being used to model the job processing times. It was difficult to compare the actual processing time for each job class with the observed values. Since each initiator in the model was processing multiple job classes, I could not find a way to compare the simulation results with the observed values. The exception to this was the first initiator which handled only class H jobs. A comparison of the model

runs and the observed values for this one job class provides the following:

	Queue Time	Processing Time
Model	.023	2.248
Actual	.027	0.71

A second truth came to light during the validation of the model, that is, model building and testing is an iterative process. Each time you test your model, you not only vary the parameters, you also re-examine all of the major components and processing logic of the model.

Simulation Results

While I believe that the model which was developed as part of this project has the potential to improve capacity and performance evaluation of the system being modeled, it will need a lot more work before I would be willing to modify the current system based on the results of the simulation. The most obvious refinement would be to increase the observed data from the system being modeled and develop more accurate distributions for execution time of the four job classes.

The one interesting factor which the simulation project did produce came from working with the system experts. After the original simulation models failed to produce the desired results, it became obvious that some of the information provided by the experts was incorrect. Their white board conceptual model was an accurate picture of how they believed the actual system was configured. When the simulation results did not compare with the observed data, the following investigation discovered that some of their knowledge of the system was based on out of date information.

All of the experts believed that class Q and R jobs were being single threaded through the system to minimize the contention for data with online terminal access. That decision had been implemented years ago to

insure that we would be able to meet the standards which were set for minimum acceptable online response time. All of the simulation models which included a single class Q and a single class R initiator produced queue times of anywhere from 10 to 80 minutes on average depending on the seed chosen. Only by moving to at least six initiators with at least two class Q initiators in the system did the simulation begin to produce anything that even remotely resembled the actual system performance.

It was after one of these discussions of the inconsistency of the model results and the observed values that one of the technical services people produced the actual configuration of the initiators which indicated three class Q initiators and three class R initiators in the system.

If I was really pressed to identify results which were provided by the simulation model, I would offer the following:

1) Based on the results of the simulation runs, if we were forced to restrict the class Q and R jobs to a single initiator to minimize contention with online teleprocessing, we would not be able to provide adequate turnaround for these classes of jobs between 8:00 a.m. and 4:00 p.m during weekdays. All of the single threaded simulation runs

produced queue wait times of between 10 and 80 minutes.

2) Based on the results of the simulation runs, six initiators properly configured will provide nearly the same level of throughput as the ten initiators which are currently defined in the system. This conclusion may be important in light of a statement made by one of the experts during a white board session. He suggested that the more initiators that were defined in the system, the greater the contention for resources. In essence, the efficiency of the system decreases as the number of jobs which are being processed simultaneously increase. In an exterme situation, if you could put enough jobs in execution at once, the system would spend so much time switching from job to job that it would get no useful work done. The concept is called thrashing.

There is nothing in the simulation model that accounts for this decrease in system efficiency as the number of jobs being processed increases. There is also no obvious way to identify the loss in efficiency using the SMF/RMF data available and I'm not sure how I would include it in the model if it were available. If the decline in efficiency is really a feature of the actual system, being able to

accomplish the same work with fewer initiators might result in a measurable increase in system throughput.

Project History

Like most computer projects, the process of building and testing the model takes more time than the project schedule allows for. Initially, I depended on the technical services group for data about the actual system. I was not high on their priority list and I built a lot of the model before I had reasonable data. By the end of the project, I had obtained copies of the programs that processed the SMF/RMF data. This made it much easier for me to analyze the distributions and parameters for those distributions. Unfortunately, it did not give me enough time to process the data through distribution fitting software.

My original project plan (Appendix C) called for all the modeling to be done before November 25, 1993. Thanksgiving weekend would be the time to document the program and write the project report. Unfortunately, the actual configuration of the system was not discovered until the Friday after Thanksgiving. This put me back into a modify, verify and validate cycle before the final runs could be made.

I used the simulation software with the OS/2-Windows operating system. It allowed me to work on the report while the simulation models were running. Unfortunately, the system dumps data to the printer faster than

the Laser 4 can handle it. This tends to over-run the print buffer and prints garbage in the middle of graphic output. I retouched the only graphic in the paper, but the lack of figures in the paper is the result of the hardware problem with my computer.

Finally, I found the ProModel software had a different conceptual framework than any other software I have used before. It took me a while to begin to think and conceptualize problems using the software. The context sensitive help function is not very context sensitive.

Conclusion

I believe that the use of simulation modeling can make a significant contribution to computer capacity and performance analysis. ProModel seems to be a tool with more than enough flexibility to handle the complexity of large mainframe computer systems. I think that the declining costs of mainframe and Personal Computer hardware and software have moved many people in the direction of adding capacity rather than analyzing and improving the performance of the existing configuration. The real value of tools like simulation modeling will be in identifying and removing bottlenecks before they are reached. It will also be an aid in identifying and resolving performance problems that are not capacity related.

Biblography

1. Allen, Arnold O., Probability, Statistics and Queueing Theory With Computer Science Applications. New Your: Academic Press, 1978

2) Capacity Planning; Basic Models - Independent Study Program. Mew York: International Business Machines Corporation, 1980

3. Harrell, Charles R., Robert E. Bateman, Thomas J. Gogg, Jack R. A. Mott, System Improvement Using Simulation. Orem, Utah: ProModel Corporation, 1992

4. Merrill, H. W. "Barry," Merrill's Expanded Guide to Computer Performance Evaluation Using the SAS System. Cary, North Carolina: SAS Institute Inc., 1984