Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

SCHEDULING POLICIES FOR A COMPUTING SYSTEM

Mary A. Johnson
Udatta S. Palekar
Yi Zhang

Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801, U.S.A.

ABSTRACT

We describe simulation experiments conducted to analyze scheduling procedures for batch jobs on a Cray computer. Two prominent themes are use of information and avoidance of interference of small jobs by large jobs.

1 INTRODUCTION

The scheduling policies used on time-sharing computing systems have a critical effect on system performance. We present simulation experiments aimed at evaluating and improving the scheduling rules of a heavily used Cray-YMP supercomputer at the University of Illinois. We observe the performance of a simple system and consider various modifications, some of which are elements of the current system and others potential changes. Two themes recur throughout our study: use of information about resource requirements and avoidance of interference of small jobs by large jobs. Resources considered are processing-time and memory.

Studies of queueing disciplines for computing systems include Coffman and Denning (1973), Conway, Maxwell, and Miller (1967), and Kleinrock (1976). Prominent ideas include pre-emption, use of information, and time splicing. Lavenberg (1988) provides an overview of queueing-network models of computing systems.

2 CRAY-YMP SYSTEM

When a batch job is submitted to the computer, it is queued in one of a matrix of queues where the job waits to be admitted into the kernel. Each queue corresponds to a specific range of processing-time and memory requirements, which are estimated by the user. Processing-time ranges are (0 min., 5 min.], (5 min., 15 min.], (15 min., 1 hr.], (1 hr., 5 hr.], and (5 hr., 20 hr.]. Penalties for under-estimation of resources typically lead to over-estimation of required resources. Each loading queue is assigned a priority which determines the order in which the queues are polled; within a queue jobs are selected FCFS. For certain classifications of jobs, there are limits on the number of such jobs that can reside together in the kernel.

Once a job is admitted into the kernel, it waits in a second queue for processing. We consider three disciplines which vary in the amount of information used: processor sharing (PS), shortest-elapsed-time first (SET), and shortest-processing-time first (SPT).

3 SIMULATION EXPERIMENTS

The input for our simulation runs consists of actual data for a 23-day period. Table 1 below shows the number of jobs in each processing-time classification, based on the user estimates and on actual processing time. Notice the changes in the first and last classifications are consistent with our assertion that users tend to overestimate resource requirements. For a given set of jobs, the performance measure presented here is the total response time of all jobs in the set, normalized by the total processing times of those same jobs.

Table 1: Counts of Jobs by Processing-Time Range

Upper bound on range	User-Estimates	Actual Time
5 min	1942	3904
15 min	2104	681
1 hour	1156	854
4 hour	503	321
20 hour	108	54

Table 2 shows performance for a system that imposes a limit of 18 jobs in the kernel at one time. The priorities assigned to the loading queues are such that priority decreases with increasing (estimated) resource requirements. Relative to their processing times, the response times of small jobs is much larger than for large jobs. Use of SET or SPT is a means of using processing-time information to prevent large jobs from interfering with small jobs. This mitigates the large response times of small jobs somewhat and improves overall performance, at a slight cost to the larger jobs.

SPT is better than SET, because it uses more information.

Table 2: Normalized Mean Response Times for Simple Loading Policy

Class	PS	SET	SPT
5 min	42.62	26.29	21.31
15 min	33.25	13.73	9.95
1 hour	11.51	7.10	5.57
4 hour	7.11	4.37	3.49
20 hour	3.36	4.29	4.55
Overall	9.94	6.15	5.12

Table 3 shows the effect of using actual processing times (instead of user estimates) to classify jobs into the loading queues. Though perfect information is not feasible, improvements could be attained by lessening the penalty for underestimation of resource requirements. Also, sampling of large jobs to detect jobs with bugs would avoid unecessarily long waits for such jobs. The most noticeable effect of the use of perfect processing-time information is that performance under SPT improves for every class and is now much better than performance under SET. Thus, availability of accurate information makes SPT much more effective.

Table 3: Normalized Mean Response Times using Actual Processing Times

Class	PS	SET	SPT
5 min	51.05	28.86	13.20
15 min	39.98	9.71	4.49
1 hour	9.19	5.67	3.32
4 hour	5.48	3.61	2.53
20 hour	3.08	4.25	3.72
Overall	10.10	5.92	3.71

Table 4: Normalized Mean Response Times after Imposing Job-Mix Constraints

Class	PS	SET	SPT
5 min	5.88	2.81	2.96
15 min	4.57	2.21	1.59
1 hour	9.54	5.51	5.11
4 hour	6.22	3.65	3.10
20 hour	2.85	4.62	4.34
Overall	5.99	4.26	3.85

Finally, Table 4 shows the effect of imposing limits on the job mix allowed in the kernel. The key idea here is that large jobs are prevented from dominating the kernel. As in Table 3, exact processing times are assumed to be known. Performance under the PS policy is improved dramatically for small jobs and overall. Performance of small jobs also improves under SET and SPT. In fact, the SET and SPT are now biased in favor of small jobs.

Another issue not apparent from the performance measures shown here is that lack of information and an appropriate priority/limitation structure, leads to large tails in the distributions of response-time-to-processing-time ratios. Use of exact processing times dramatically improves the tail behavior for jobs classified as large, since it eliminates the problem of misclassified small jobs receiving low priority because of their misclassification. Addition of limitations on the job mix provides a similarly dramatic improvement in the tails for the small jobs.

REFERENCES

Coffman, Jr., E.G., and P.J. Denning, *Operating Systems Theory*, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

Conway, R.W., W.L. Maxwell, and L.W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, Massachusetts, 1967.

Kleinrock, L., *Queueing Systems Volume 2: Computer Applications*, Wiley, New York, 1976.

Lavenberg, S.S., A Perspective on Queueing Models of Computer Performance, *Queueing Theory and its Applications--Liber Amicorum for J.W. Cohen*, North-Holland, Amsterdam, 1988.

AUTHOR BIOGRAPHIES

MARY A. JOHNSON is an assistant professor in the Department of Mechanical and Industrial Engineering at the University of Illinois at Urbana-Champaign. Her research interests include algorithmic probability and queueing approximations.

UDATTA S. PALEKAR is an associate professor in the Department of Mechanical and Industrial Engineering at the University of Illinois at Urbana-Champaign. His research interests include scheduling, discrete optimization, and production planning.

Yi Zhang is a PhD student in the Department of Mechanical and Industrial Engineering at the University of Illinois at Urbana-Champaign. His research interest include scheduling and queueing applications.