Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

MULTITASKING AND RESEQUENCING IN A TWO-STAGE MULTIPROCESSING SYSTEM

Helen D. Karatza

Department of Informatics
Aristotle University of Thessaloniki
54006 Thessaloniki, GREECE

ABSTRACT

The effects of multitasking and resequencing on system and program performance of a two-stage multiprocessing system are studied using simulation techniques. A closed queueing network model is considered. It consists of the CPU and the I/O channel. The CPU consists of a FCFS single-server sequential processing center and a multiprocessing center with two FCFS independent processors each serving its own queue. Resequencing of jobs after CPU service ensures that jobs leave the CPU on a first-in-first-out basis. We examine how the synchronisation of tasks in the multiprocessing center in conjunction with the subsequent resequencing of jobs affects the overall performance for various coefficients of variation of the CPU service times (exponential and Branching Erlang distributions) and for different degrees of multiprogramming.

1 INTRODUCTION

In this paper we study the effects of multitasking and resequencing on system and program performance of a two-stage multiprocessing system. The results are obtained using simulation techniques.

A closed queueing network model is considered which consists of the CPU and the I/O unit. The CPU consists of a FCFS single-server sequential processing center and a multiprocessing center with two FCFS independent processors each serving its own queue. Therefore the CPU system consists of two stages, Stage 1 and Stage 2, called the "job stage" and the "task stage", respectively.

In center Stage 2, each service time of a job (program) consists of two tasks. In half of the jobs the tasks are independent and can be processed in parallel. Synchronisation between tasks is required. In the other half of jobs both tasks of the same job must be executed sequentially on the same processor. These jobs join the shorter queue.

An alternative approach we use to avoid the synchronisation of tasks is to run sequentially both tasks of all jobs. These two approaches are studied and compared.

Resequencing of jobs after CPU service ensures that jobs leave the CPU on a first-in first-out basis.

The resequencing problem can be found in several system contexts such as distributing computing systems and computer communication systems. In those systems customers who arrive at the system should depart from the system in the same order as their arrivals. Therefore, after the completion of their services, customers who go out of order are forced to wait in a special buffer—the so-called resequencing buffer—in order to rearrange their sequences, i.e. until the overtaken job of the same class completes its service. The delay due to resequencing is called resequencing delay.

The resequencing delay in multiserver queues has been studied by many authors as Iliadis, and Lien (1988); Lien (1986); Sasase, and Mori (1991); Takine, and Hasegawa (1990); Varma (1991). Load balancing in a system of two queues with resequencing has been studied by Jean-Marie (1988). All these works study open queueing network models and exponential CPU service times.

Multitasking and resequencing in a homogeneous distributed system is studied in Karatza (1995a). This work studies closed queueing network models.

Two-stage parallel processing systems with parallel servers at the second CPU center are studied in Persone, and Iazeolla (1994) and in Karatza (1995b).

In this work we study a two-stage multiprocessing system where there are two independent processors each serving its own queue at the second CPU center. Our intention is to examine if the synchronisation of tasks in the multitasking case, in conjunction with the subsequent resequencing of jobs is rewarded with increased parallelism for various coefficients of

variation of the CPU service times (exponential and Branching Erlang distributions) and for different degrees of multiprogramming.

2 MODEL DESCRIPTION

We consider a closed queueing network model which consists of the CPU and the I/O channel. The CPU consists of a FCFS single-server sequential processing center and a multiprocessing center with two FCFS independent processors each serving its own queue. Therefore the CPU system consists of two stages, Stage 1 and Stage 2, called the "job stage" and the "task stage" respectively.

In center Stage 2, each service time of a job consists of two tasks. In half of the jobs the tasks are independent and can be processed in parallel. In the other half of jobs both tasks of the same job must be executed sequentially on the same processor.

In the case of jobs with parallel tasks, upon arrival at the center Stage 2 a job forks into two tasks. Task i, $i = 1, 2$ is assigned to the ith queue. Tasks corresponding to same job are joined before departing the CPU system, i.e. the task that completes its service first waits for its sibling to finish execution. Therefore in our model synchronisation between tasks is required. The price that one pays for the increased parallelism is the synchronisation delay that occurs when tasks wait for their siblings to finish execution.

In the case of jobs with sequential tasks they join the shorter queue. If both queues have an equal number of jobs (including empty), the arriving job at center Stage 2 is dispatched randomly to one of the two queues with equal probability.

Therefore in this paper we consider multiprocessing systems in which programs are composed of two stages which must be processed in sequence. For each program there is some preparatory sequential computation, processed by center Stage 1, followed by a second computation which is parallel for half of the jobs and is processed by processors in center Stage 2.

The important requirement of the model is that jobs must leave the CPU system in the order of their arrival. For this, each job enters the resequencing buffer after completion of its service, where it eventually waits until all jobs that entered the system before it have been served. We furthermore assume that the time taken by its resequencing process is negligible: if this job has no one to wait for, it leaves instantaneously the system. On the other hand, it leaves as soon as the last customer it has to wait for enters the resequencing buffer.

Both tasks of a given job are to be completed at center Stage 2 before the job enters the resequencing buffer.

An alternative approach we use in this paper to avoid the synchronisation of tasks is to run sequentially both tasks of all jobs. In this case the only delay that incurs is due to resequencing of jobs.

In both cases, the FCFS queueing discipline is assumed.

The model is closed as the degree of multiprogramming N is constant during the simulation experiment. Neither arrivals nor departures are permitted while the system is under observation. The configuration of the model is shown in Figure 1. A fixed number of jobs N is circulating within the system alternatively between the CPU and the I/O channel.

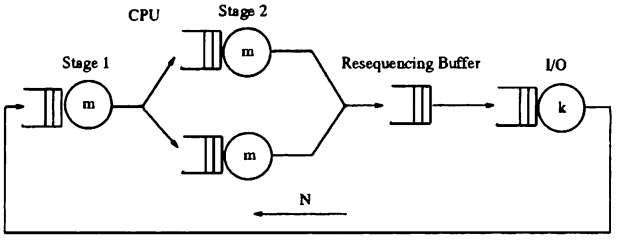

Figure 1: The Queueing Network Model

We examine two cases of CPU service time distributions:

(1) CPU service times are independent and identically distributed (IID) exponential random variables with mean m at each processor.

(2) CPU service times have a Branching Erlang distribution with two Stages (Sauer and Chandy 1981) and are IID. The coefficient of variation is C, where $C > 1$ and the mean is m at each processor.

A job after leaving the CPU requests service from the I/O unit. The I/O queueing discipline is FCFS. The I/O service times are exponentially distributed with mean k and are IID.

3 PERFORMANCE PARAMETERS

We define:

Stage i response time of a random job as the interval of time measured from an arrival of this job at center Stage i, i=1,2 to service completion of this job at center Stage i.

Cycle time of a random job as the time between two successive CPU service requests of this job.

We also denote the following:

$RT1$: mean Stage 1 response time
$RT2$: mean Stage 2 response time
D : mean resequencing delay
$RT2D$: mean Stage 2 response time plus resequencing delay
K : mean cycle time
N : degree of multiprogramming
R : system throughput rate

The performance of our model is indicated by the mean cycle time (program performance) and the system throughput rate (system performance).

When the model works first with tasks of all jobs running sequentially, and then with multitasking at center Stage 2, we define the relative performance parameters calculated on a percentage basis as follows:

D_{RT2} : relative decrease in $RT2$
D_{RT2D} : relative decrease in $RT2D$
D_K : relative decrease in K
D_R : relative increase in R

4 SIMULATION RESULTS

We considered a balanced system with $m = 1.0$ and $k = 1.0$. The system was examined in cases of exponential CPU service times (C=1) and Branching Erlang for $C = 2, 4$. We have not examined cases with $C > 4$ because in real systems jobs service times tend to have low variations. Degree of multiprogramming N was taken as 2, 4, 6, 8, 10.

We simulated the queueing network model with discrete event simulation models using the method of independent replications (Law, and Kelton, 1991). For every mean value a 95% confidence interval was evaluated. All confidence intervals were less than 5% of the mean values.

In Tables 1–6 performance parameters of all cases are presented. Tables 7–9 represent relative performance parameters for $C = 1, 2, 4$.

In Figures 2 and 3 relative performance parameters are plotted versus N in all cases examined.

Table 1: Sequential Tasks Case, C=1

N	$RT1$	$RT2$	D	$RT2D$	K	R
2	1.259	2.000	0.240	2.240	4.854	0.412
4	1.792	2.473	0.334	2.807	6.583	0.608
6	2.356	2.993	0.399	3.392	8.364	0.717
8	2.944	3.518	0.432	3.950	10.221	0.783
10	3.588	4.105	0.427	4.532	12.091	0.827

Table 2: Multitasking Case, C=1

N	$RT1$	$RT2$	D	$RT2D$	K	R
2	1.255	2.121	0.042	2.162	4.707	0.425
4	1.807	2.645	0.193	2.838	6.561	0.610
6	2.349	3.256	0.315	3.572	8.493	0.706
8	2.934	3.877	0.399	4.276	10.414	0.768
10	3.492	4.458	0.529	4.987	12.380	0.808

Table 3: Sequential Tasks Case, C=2

N	$RT1$	$RT2$	D	$RT2D$	K	R
2	1.282	2.004	0.609	2.613	5.280	0.379
4	1.868	2.567	1.061	3.628	7.653	0.523
6	2.447	3.094	1.298	4.392	9.851	0.609
8	2.960	3.574	1.533	5.108	11.956	0.669
10	3.414	4.127	1.616	5.743	13.933	0.718

Table 4: Multitasking Case, C=2

N	$RT1$	$RT2$	D	$RT2D$	K	R
2	1.280	2.302	0.187	2.489	5.096	0.392
4	1.875	3.013	0.528	3.541	7.470	0.535
6	2.455	3.658	0.792	4.450	9.711	0.618
8	3.006	4.286	0.992	5.278	11.872	0.674
10	3.384	5.089	1.135	6.224	13.982	0.715

Table 5: Sequential Tasks Case, C=4

N	$RT1$	$RT2$	D	$RT2D$	K	R
2	1.313	1.934	0.857	2.791	5.497	0.364
4	1.999	2.489	1.872	4.361	8.560	0.467
6	2.658	3.101	2.440	5.541	11.327	0.530
8	3.264	3.443	3.234	6.677	14.137	0.566
10	3.841	3.738	3.983	7.721	16.804	0.595

Table 6: Multitasking Case, C=4

N	$RT1$	$RT2$	D	$RT2D$	K	R
2	1.317	2.390	0.321	2.712	5.381	0.372
4	1.998	3.253	0.990	4.243	8.408	0.476
6	2.668	4.056	1.551	5.607	11.339	0.529
8	3.256	4.938	1.867	6.805	14.069	0.569
10	3.756	5.657	2.299	7.956	16.744	0.597

Table 7: Relative Performance, C=1

N	D_{RT2}	D_{RT2D}	D_K	D_R
2	-6.04	3.46	3.02	3.11
4	-6.95	-1.10	0.33	0.33
6	-8.81	-5.31	-1.54	-1.52
8	-10.21	-8.27	-1.89	-1.86
10	-8.61	-10.04	-2.39	-2.34

Table 8: Relative Performance, C=2

N	D_{RT2}	D_{RT2D}	D_K	D_R
2	-14.87	4.74	3.48	3.60
4	-17.38	2.42	2.39	2.45
6	-18.24	-1.32	1.41	1.44
8	-19.92	-3.33	0.70	0.70
10	-23.31	-8.37	-0.35	-0.35

From the results we observe the following.

In all systems, both cases of multitasking and of sequential tasks perform almost the same as D_K and D_R are less than 4%.

In all cases $RT2$ is lower with the sequential case. This is due to the following.

(a) In the multitasking case, half of the jobs after arriving at the center Stage 2 split into two tasks where each one is assigned to a different processor queue independently of the system state. However, in the sequential tasks case there is an optimal job allocation to two queues as all jobs join the shortest queue.

(b) In the multitasking case there is the advantage of parallel processing of tasks of the same job at center Stage 2, but this incurs in synchronisation delay of tasks.

For all N, the relative difference in $RT2$ increases with increasing C. This is due to the increase in the variability of task sizes with increasing C which results in unbalanced processor queues at Stage 2 and therefore in better exploitation of the "join the shortest queue" policy. However, because of the subsequent resequencing delay of jobs, for all C, the relative difference in $RT2D$ is much smaller than the relative difference in $RT2$.

For $C = 1$ and for all N the mean resequencing delay is relatively small compared to $RT2$. However, for $C > 1$ and for all N, D increases with increasing C and it becomes a considerable part of K especially in the case of jobs with sequential tasks. For example for $C = 4$ and $N = 10$, D is close to $K/4$ and it is bigger than $RT2$ (Table 5).

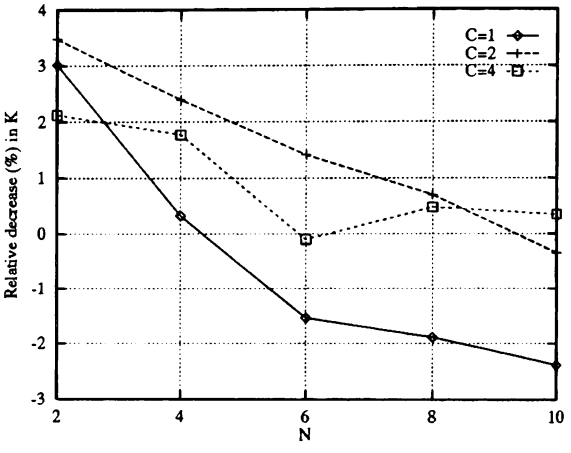

Figure 2: D_K versus N

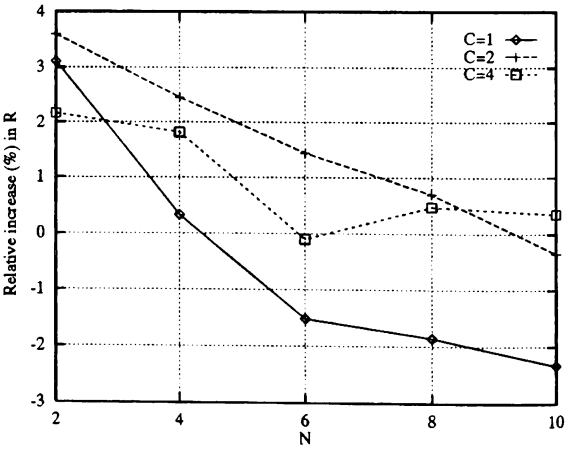

Figure 3: D_R versus N

Table 9: Relative Performance, C=4

N	D_{RT2}	D_{RT2}	D_K	D_R
2	-23.62	2.84	2.12	2.16
4	-30.71	2.70	1.77	1.81
6	-30.80	-1.20	-0.10	-0.10
8	-43.43	-1.92	0.48	0.48
10	-51.36	-3.05	0.35	0.35

This is due to the fact that the higher C is, the more the CPU service times vary from the mean service time m. Furthermore, more service times are produced that are much shorter than m and fewer ones that are much longer than m. So, when a processor at Stage 2 is busy for a long time processing a task with a long service time, the other processor

can serve both tasks of other subsequent jobs which, however, must wait in the resequencing buffer until the big task is finished.

The resequencing delay is more serious for high degrees of multiprogramming because it is more probable at high N more subsequent jobs to go out of order.

In all systems for all N the performance deteriorates with increasing C (K increases and R decreases with increasing C).

5 CONCLUSIONS AND FURTHER RESEARCH

The effects of multitasking and resequencing on system and program performance of a two-stage multiprocessing system were studied using simulation techniques. The simulation results reveal the following.

- In the multitasking case the synchronisation of tasks in the multiprocessing center results in higher response times in this center compared with the case of sequential tasks. However, because the resequencing delay is higher with the case of sequential tasks, the two cases do not differ significantly.
- The resequencing delay of jobs increases with increasing C and also with increasing N.

In this research one class jobs have been considered. It should be extended to the following direction.

- Jobs of different classes to be considered depending on whether they consist of parallel or sequential tasks, and preemption.

ACKNOWLEDGMENTS

This work was carried out while the author was on sabbatical in the Department of Computer Science, at York University, Ontario, Canada.

REFERENCES

Iliadis, I., and Y. C. Lien. 1988. Resequencing Delay Distribution for a Queueing System with Two Heterogeneous Servers under Threshold Scheduling. In *Data Communication Systems and Their Performance*, ed. L. F. M. de Moraes, E. de Souza e Silva, and L. F. G. Soares, 359–373. Elsevier Science Publishers B.V., North-Holland, IFIP.

Jean-Marie, A. 1988. Load Balancing in a System of Two Queues with Resequencing. In *Performance '87*, ed. P.-J. Courtois, and G. Latouch, 75–88. Elsevier Science Publishers B.V., North-Holland.

Karatza, H. D. 1995a. Simulation Study of Multitasking and Resequencing in a Homogeneous Distributed System. In *Proceedings of the Eurosim Congress '95*, ed. F. Breitenecker, I. Husinsky, 541–546. Elsevier Science Publishers B.V., North-Holland.

Karatza, H. D. 1995b. Simulation Study of Two-Stage Parallel Processing Systems with Resequencing. In *Proceedings of the 7th European Simulation Symposium, ESS 95*, ed. M. DalCin, U. Herzog, G. Bolch, A. Riza Kaylan, 464–468. Society for Computer Simulation.

Law, A., and D. Kelton 1991. *Simulation Modeling and Analysis*. 2d ed. New York: McGraw-Hill.

Lien, Y. C. 1986. Evaluation of the Resequencing Delay in a Poisson Queueing System with Two Heterogeneous Servers. In *Computer Networking and Performance Evaluation*, ed. T. Hasegawa, H. Takagi, and Y. Takahashi, 189–197. Elsevier Science Publishers B.V., North-Holland, IFIP.

Persone, V. de Nitto, and G. Iazeolla. 1994. Performance Analysis of the Parallel Cyclic Two-Stage Queueing Model. *Performance Evaluation* 19: 167–193.

Sasase, I., and S. Mori. 1991. Resequencing Delay for a Queueing System with Multiple Servers under Threshold-Type Scheduling. In *Proceedings of the Tenth Annual Joint Conference of the IEEE Computer and Communications Societies*, 391-399, Vol.1, Institute of Electrical and Electronics Engineers.

Sauer, C. H., and K. M. Chandy. 1981. *Computer Systems Performance Modelling*, Englewood Cliffs, New Jersey: Prentice-Hall.

Takine, F., and T. Hasegawa. 1990. Resequencing Delay in Preemptive Priority M/M/2 Queues. In *Performance '90*, ed. P. J. B. King, I. Mitrani, and R. J. Pooley, 109–121. Elsevier Science Publishers B.V., North-Holland.

Varma, S. 1991. Optimal Allocation of Customers in a Two Server Queue with Resequencing, *IEEE Transactions on Automatic Control* 36 (11) :1288–1293.

AUTHOR BIOGRAPHY

HELEN D. KARATZA is an Assistant Professor in the Department of Informatics at Aristotle University of Thessaloniki, Greece. She has a B.S. degree in Mathematics and a Ph.D. degree in Computer Science from Aristotle University. Her research interests mainly include Performance Evaluation of Parallel and Distributed Systems and Simulation.