cs 1994

Simulation of a MSIMD System with Resequencing

Helen D. Karatza
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, GREECE, 54006

Abstract

In this work we study the performance of a multiprocessor system model which executes multiple SIMD jobs. Such a parallel computer organization contains multiple Control units (CUs) which share a resource pool of a finite number of Processing Elements (PEs) and operates multiple single-instruction-multiple-data streams (MSIMD). We assume that after the CUs service, resequencing of SIMD jobs takes place which ensures that jobs leave the processing unit on a first-in-first-out basis.

A closed queueing network model of a MSIMD computer system is simulated. The performance of two different queueing disciplines in conjunction with the effect of the resequencing delay is investigated for various degrees of multiprogramming and coefficients of variation of the CUs service times.

1 Introduction

This paper studies the performance of a MSIMD computer system network.

A MSIMD machine is composed of two or more CUs sharing a finite number of PEs. Each CU needs to be allocated with a subset of PEs for the execution of a single vector job (SIMD process). The only way vector jobs can interact with each other is through their independent needs for the same resources.

Other studies on MSIMD computer organizations include Hwang and Ni [1], [2], Ni and Hwang [6] and Karatza [4]. All these works study exponential computing demands without any resequencing (synchronization) of SIMD jobs.

The resequencing problem can be found in several system concepts. In those systems, customers which arrive at a system should depart in the same order as their arrivals. Therefore 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. The delay due to resequencing is called resequencing delay.

The resequencing delay in multiserver queues is studied by Iliadis and Lien [3], Sasase and Mori [7], Takine and Hasegawa [9] and Varma [10]. They study jobs without any inherent parallelism (SISD jobs).

In this work we study the performance of a MSIMD system model where resequencing of SIMD jobs after the CUs service ensures that jobs leave the CUs on a first-in-first-out basis.

A closed queueing network model is considered the results being obtained using simulation techniques. Queueing systems and simulation are powerful tools for performance analysis and prediction and have been extensively used for modelling computer systems. When effective, queueing theoretic techniques can quickly provide mathematical insight into the behavior of systems over a broad range of parameter values. Their major limitation is the number of restrictive assumptions that must be satisfied to ensure accuracy. Conversely, simulation models can mimic a realworld system as closely as understanding permits and needs require.

We examine the performance of two different queueing disciplines in conjunction with the effect of the resequencing delay for various degrees of multiprogramming and coefficients of variation of the CUs service times. Our purpose is to investigate cases for better performance of this model.

2 Model description

The configuration of the queueing network model is shown in Fig. 1. The network consists of two units: the processing unit (CUs and PEs) and the I/O channel.

The model is closed as the degree of multiprogramming N is constant during the simulation experiment. A fixed number of vector jobs N is circulated alternately within the system between the processing unit

0-8186-6322-7/94 $04.00 © 1994 IEEE

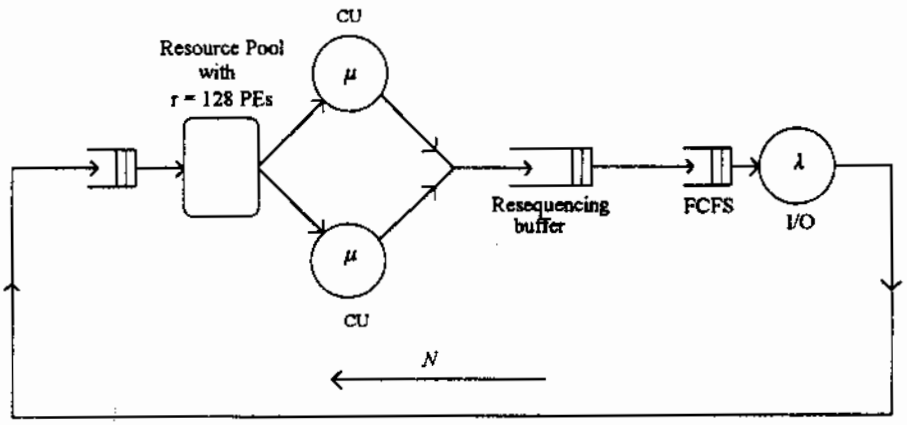

Fig. 1. The queueing network model

and the I/O channel. In our model the departure of jobs from the processing unit should be the same order as their arrivals. Therefore, after the completion of their services jobs who go out of order are forced to wait in the resequencing buffer until the overtaken job completes its service.

The processing unit has $p = 2$ identical servers called CUs and $r = 128$ identical PEs. There is one queue for all CUs and one queue for the I/O channel. The first available CU is always used. The subset of PEs allocated to a given CU may vary in size according to the job requirement.

Each CU can supervise the execution of one independent SIMD job at a time. At most p SIMD jobs can be executed simultaneously in the processing unit. The PEs are shared by the CUs. Each PE can be allocated to one CU at a time.

A CU is in the wait state when it is available for a job assignement. An availacle CU enters the busy state if there is a vector job and the PEs demanded by that job are also available from the resource pool. Otherwise it remains in the wait state. When a PE is allocated to a CU that PE enters the busy state; otherwise it is in the idle state. The time required to allocate PEs to their designated CU is considered part of the CU service time. Since the PEs pool size is r, the number of PEs required by each job is always within the range $[1, r]$.

We assume that the number of PEs required by a vector job is a random variable X, called the PE-demand. The PE-demand of a vector job is assumed to remain constant throughout the vector process. Successive vector jobs have independent and identical PE-demand distribution F.

In practice the PE-demand distribution F does not follow a specific pattern. However, in order to avoid large variability of PE-demands we assumed a "bounded" Normal Distribution for F with mean PE-demand $\eta = r/p = 64$ and with standard deviation $\sigma = \eta/4 = 16$.

We used two different queueing disciplines at the processing unit. The First-Come-First-Served (FCFS) and the Least-PE-Demand- First-Served (LDFS).

The FCFS discipline decides the order of service strictly by the order of job arrivals. Clearly it has the disadvantage of having a SIMD job with a large PE-demand blocking in the front of the queue other smaller vector jobs.

In the LDFS case the insertion of successive jobs must maintain a decreasing PE-demand order in the queue so that less-PE-demanding jobs are always placed ahead of more-PE-demanding jobs.

We would probably expect faster service of jobs from the processing unit with the LDFS discipline than with the FCFS. However, we cannot predict the performance because the subsequent delay of jobs in the resequencing buffer may affect seriously the results.

At the I/O channel we used the FCFS queueing discipline.

Let C be the coefficient of variation of the CUs service times. We assumed three cases of CUs service times distributions [8]:

1. Erlang-k distribution with $k = 2$ $(C = 1/\sqrt{k} = 0.707)$.
2. Exponential distribution $(C = 1)$.
3. Branching Erlang distribution with two stages for $C = 2, 4$.

In all these cases the mean is μ. The I/O service times are assumed to be exponential with mean λ.

3 Performance parameters

We define:

Response time of a random job the interval of time measured from an arrival of a job at the processing unit until the service completion of this job (waiting plus service time).

Cycle time of a random job the time between two service requests of a job from the processing unit.

We denote the following:

RT : mean response time
RTD : mean response time plus resequencing delay
K : mean cycle time
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). Our purpose is to investigate cases which yield smaller RTD values which in turn result in better performance. By definition RTD depends on both the response times of jobs and the resequencing delays.

When the model works first with the FCFS queueing discipline and then with the LDFS we define the relative performance parameters calculated on a percentage basis as follows:

D_{RT} : relative decrease in mean response time
D_{RTD} : relative decrease in RTD
D_K : relative decrease in mean cycle time
D_R : relative increase in system throughput rate

4 Simulation results

We considered a system with balanced program flow with $\mu = 1$ and $\lambda = 0.5$. Degree of multiprogramming N was taken as 2, 4, 6, 8, 10.

We simulated the queueing network model with discrete event simulation models we wrote in FORTRAN.

Every simulation model was repeated 30 times for 10,000 CPU service time requests each time, with the same starting conditions but with different streams of random numbers, so as to ensure that the replications are independent [5]. The mean output values of the performance parameters were given by the overall means of the 30 replications.

We run the simulation programs in an Apollo HP9000 / S715-33 Workstation.

Tables 1-4 represent the relative performance of the two queueing disciplines for all N in the cases of C= 0.707, 1, 2, 4. From the results we observe the following:

N	D_{RT}	D_{RTD}	D_K	D_R
2	0.00	0.00	0.00	0.00
4	8.33	1.83	-0.91	-0.90
6	18.82	6.87	-0.83	-0.79
8	25.39	9.49	-0.53	-0.46
10	27.78	11.33	-0.38	-0.31

Table 1. $C = 0.707$

N	D_{RT}	D_{RTD}	D_K	D_R
2	0.00	0.00	0.00	0.00
4	8.36	2.84	0.14	0.15
6	18.86	5.95	0.24	0.25
8	25.52	9.06	0.47	0.48
10	27.98	10.92	1.24	1.25

Table 2. $C = 1$

For $N = 2$ the FCFS and LDFS disciplines gave the same results as in this case at any instance there is at most one job in the processing unit queue.

N	D_{RT}	D_{RTD}	D_K	D_R
2	0.00	0.00	0.00	0.00
4	10.76	3.12	0.68	0.86
6	21.65	7.12	1.65	2.68
8	26.33	10.70	3.59	4.27
10	28.57	11.41	3.67	4.41

Table 3. $C = 2$

N	D_{RT}	D_{RTD}	D_K	D_R
2	0.00	0.00	0.00	0.00
4	10.10	1.45	0.49	0.65
6	18.06	3.20	0.68	0.90
8	25.31	4.49	1.33	1.47
10	28.25	6.86	2.74	1.62

Table 4. $C = 4$

For $N > 2$ and for all C, jobs are served faster by the processing unit in the LDFS case than in the FCFS. This superiority of the LDFS discipline increases with increasing N. However, because of the resequencing delay this superiority is not completely exploited. This is the reason the values of D_{RTD} are much smaller than the respective D_{RT}.

If we examined an open model we would be satisfied using the LDFS discipline. However, the model under study is a closed one and its overall performance depends on the alternate circulation of programs between the processing unit and I/O unit. So, except of the RTD, queueing delays at the I/O channel make a

contribution to the mean cycle time as well. This is the reason that the overall performance (K and R) is much less influenced by the application of different queueing disciplines than the RT and RTD values.

Especially for low C (cases $C = 0.707$ and $C = 1$) the advantages of the LDFS discipline almost do not affect the total performance. These advantages yield some improvement for $C = 2$ at hight N and some lighter for $C = 4$ at high N.

This occurs as the lower the C is the more the service times are closer to the mean service time μ, whilst the higher the C is the more they vary from the mean. Furthermore, for higher C more service times are produced that are much shorter than μ and fewer ones that are much longer than μ. So when a CU is busy for a long time processing a job with a long service time, the allocated PEs to this CU cannot be used for this long time by other jobs. Therefore the LDFS discipline's abilities can be exploited better in cases of long service times. However, since at high C the long service times tend to be very rare, from some value of C the LDFS abilities are rarely exploited.

5 Conclusions

In this work we studied the performance of a MSIMD system model. This model contains 2 Control Units (CUs) which share a resource pool of 128 Processing Elements (PEs) and executes multiple SIMD jobs. We assumed that after the CUs service resequencing of jobs takes place which ensures that jobs leave the processing unit on a first-in-first-out basis.

A closed queueing network model of a MSIMD computer system was simulated. The performance of two different queueing disciplines in conjunction with the effect of the resequencing delay was investigated for various degrees of multiprogramming N and coefficients of variation C of the CUs service times.

From the simulation results it is shown that although the LDFS discipline reduced significantly the mean response time in all cases examined, its advantages over the FCFS discipline were not completely exploited. This was mainly due to the resequencing delays of jobs and partly to subsequent queueing delays at the I/O unit. The performance improvement due to the LDFS discipline was not generally significant. From all cases examined the best results were obtained for $C = 2$ at high N. However, since even in this case the gain in performance is small in relation to the complexity of the LDFS queueing discipline, we could conclude that the FCFS discipline is preferable.

This work is a case study. However, further research needs to be carried out to investigate cases with different numbers of CUs and PEs.

References

[1] K. Hwang and L. M. Ni, "Performance evaluation and resource optimization of multiple SIMD computer organizations", in *Proc. Int. Conf. Parallel Processing*, pp. 86-94, Aug. 1979.

[2] K. Hwang and L. M. Ni, "Resource optimization of a parallel computer for multiple vector processing", *IEEE Trans. Comput.*, Vol. C-29, pp. 831-836, Sept. 1980.

[3] I. Iliadis and Y. C. Lien, "Resequencing delay distribution for a queueing system with two heterogeneous servers under threshold scheduling", in *Data Communication Systems and Their Performance*, L. F. M. de Moraes et al. (eds.), Elsevier Science Publishers B.V., pp. 359-373, IFIP 1988.

[4] H. D. Karatza, "Simulation study of a MSIMD Computer System Network", *Int. Journal of Modelling and Simulation*, Vol.11, pp. 75-82, 1991.

[5] A. Law and D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, New York, 1991.

[6] L. M. Ni and K. Hwang, "Performance modeling of Shared - Resource Array Processors", *IEEE Trans. Software Eng.*, Vol. SE-7, pp. 386-394, July 1981.

[7] I. Sasase and S. Mori, "Resequencing delay for a queueing system with multiple servers under threshold-type scheduling", in *Proc. Tenth Annual Joint Conf. of the IEEE Computer and Commun. Societies*, IEEE, Vol.1, pp. 391-399, 1991.

[8] C. H. Sauer and K.M. Chandy, *Computer Systems Performance Modelling*, Prentice-Hall, Englewood Cliffs, N.J., 1981.

[9] T. Takine and T. Hasegawa, "Resequencing delay in Preemptive Prioprity M/M/2 Queues" in *Performance 90*, P. J. B. King et al. (eds.), Elsevier Science Publishers B.V., pp. 109-121, 1990.

[10] S. Varma, "Optimal Allocation of Customers in a Two Server Queue with Resequencing", *IEEE Trans. on Autom. Control*, Vol.36, pp. 1288-1293, Nov. 1991.