Alex. Eng. J., Alex. Univ., Volume 28 No.3 , pp. 209-222 (1989)

SIMULATION, ANALYSIS AND COMPARISON OF TWO JOB SCHEDULING ADAPTIVE STRATEGIES

Khalil M. Ahmed, Mohamed N. El Derini and Hesham H. Ali
Department of Computer Science,
Faculty of Engineering, Alexandria University

Abstract

In this paper, we introduce and analyse two new adaptive job scheduling strategies whose goals are to improve the average response time using the load balancing techniques. Each strategy uses the concept of decentralized control.

A simulation technique is used to evaluate the effect of these strategies on various performance indices in four different case studies. A simulator is built to simulate our distributed system and study the effect of using the job scheduling strategies.

The results of the simulation runs including the comparison of the two algorithms are given in this paper.

1. Introduction

An interesting research problem arises in the design of a distributed computer system, namely how to improve its overall performance by balancing the workload over the entire system. This can be done through one or both mechanisms; task sheduling [2,3,5,9,10,11] and job scheduling [1,6,7,8].

Our attention in this paper is focussed on job scheduling mechanism. Load balancing can be done statically at system configuration time by preassigning certain jobs or classes of jobs to specific processors, or dynamically as the load and state of the system changes by automatically transfering jobs from one processor to another.

In general, the analytical solutions of the load balancing problem can take two different approaches. One approach is to formulate the problem of assigning jobs to hosts as a combinatorial optimization problem [6,7].

The other approach is to develop queueing models to analyze the performance of systems incorporating simple job routing policies that automatically shift jobs from heavily loaded hosts to lightly loaded hosts. Various techniques are available for obtaining the exact or the approximate solutions of queueing models. Notable among them are the power iteration method [8], generating function approach [4], product form solution [1] and recursive solution technique [6,7].

In this paper, we introduce and analyze two new job scheduling strategies which use the concept of decentralized control. The decentralized controlled job scheduling algorithm utilizes N physically distributed entities, each entity is considered as a local controller which runs asynchronously and concurrently with the others. Each entity makes decision based on a system-wide objective function, rather than on a local one. Each entity makes decisions on an equal basis with the other entities. It is intended that the job scheduling algorithm adapts to the changing busyness of the various hosts in the system.

In this paper, it is assumed that each entity transmits periodically its busyness to all other entities and combines receiving updates in some manner so as to obtain its new view of the system. The updating scheme used in our simulation is described in the next section.

The execution costs involved in running a decentralized controlled job scheduling algorithm include the cost of running the algorithm itself, the cost of transmitting update information and the cost of moving jobs. The primary goal of the algorithm is to minimize the response time with minimum job movement. The secondary goal is to balance the load. All these costs and goals are addressed in the simulation.

2. Model Description And Assumptions

We consider a distributed system which consists of n nodes. These nodes represent the host computers which are connected on an arbitrary fashion by a communication network. All hosts are identical but may be hetorogeneous i.e. they may have different speed characteristics. However, they have the same processing capabilities, to process a job from start to finish at any node.

Jobs arrive at any node i according to a time-invariant Poisson process with rate λ_i. The service time of a job i, scheduled for excution, is chosen from an exponential distribution with rate μ_i. Each arriving job may be either processed at the same node i, or transferred to another node j according to an adaptive strategy as shown in figures (1) and (2).

The delay in the communication subnet is modeled simply as a constant delay which is the maximum possible delay between any two hosts. Both job and state information are passed into the subnet, thereby modeling two of the major costs involved. The effect of changing that delay is discussed in section 4.

The third major cost is that of the running algorithm which is modelled as a small fixed cost of 50 milliseconds which is a reasonable figure because of the simplicity of the strategies involved.

Each node sends periodically its expected status information to all other nodes. This estimation, at the original node, will be easier and more accurate than the estimation at the receiving node. So, the receiving node may use this information without any modification. Each node decides at a scheduling interval whether it will send jobs to other nodes or not, depending on the adaptive strategy. The effect of changing the scheduling interval is discussed in section 4.

Each node is modelled as a simple FCFS process scheduler that does not allow multiprogramming. A statistics gathering capability sufficient to determine the average response time perjob, the percentage of moved jobs to the total number of jobs, the percentage of processing time at each node to the total processing time, and the average queue length at each node are included in the model. The model is designed so that

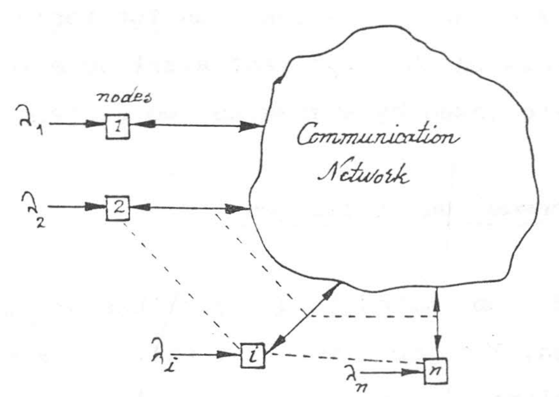


Figure (1): Distributed Computer System

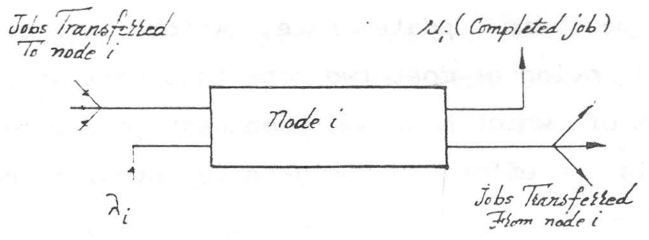


Figure (2): Job Flow in node i

it is easy to test the system under any type of workload, and under different job scheduling algorithms.

All simulations are run for five minutes of simulation time, statistics are cleared, and then run for forty five more minutes. This is done to minimize the transient start up effects of an empty system. The results are taken by averaging two different runs.

3. The Job Scheduling Strategies

Two proposed job scheduling algorithms are implemented and compared via simulation. The job scheduling entities are activated periodically every scheduling interval in each algorithm. Furthermore, the state information at each node is sent every scheduling interval to other nodes.

For both strategies, if the delay in the communication subnet is large compared to the scheduling interval, then the scheduling entities may continue to send jobs to a host not realizing that there are many jobs on the way to that host. This problem can be solved by choosing a fairly slow periodic update rate, which is acceptable for job scheduling and moving at most two jobs to a host at one time. Also, we can keep track of which host has been sent jobs recently and use this information in an effort to mitigate the problem and to minimize job movement.

Another practical consideration taken into account is to avoid moving jobs by a host if it observes that each host has less than certain lower margin or more than certain upper margin.

3.1 The First Stategy

In this algorithm, each entity compares its own busyness with the estimated busyness of the lightly loaded host. The difference between the busyness is compared with two biases. If the difference is greater than the second bias, two jobs are moved to the least busy host. If that difference is only greater than the first bias, only one job is moved. Otherwise no jobs are moved.

The term busyness is measured by the function:

$$F_i = L_i + alpha * \lambda_i / \mu_i$$

where L_i is the estimated number of jobs at node i, λ_i is the arrival rate at node i, μ_i is the service rate at node i, and alpha is a tunning factor.

3.2 The Second Strategy

In this strategy a similar algorithm is used except that each entity compares its own busyness with the estimated busyness of every other node using the same busyness function, F_i. If all the differences are less than or equal to the first bias, then no jobs are moved to other nodes. If any difference is greater than the first bias, one job is moved, and if that difference is greater than the second bias, two jobs are moved.

4. Results

The effect of using our adaptive strategies on four different case studies is examined. In all four case studies the network delay is taken to be 4 seconds and the scheduling interval is equal to 2 seconds. There are three main differences between the four chosen cases: the number of nodes in the system, its total traffic intensity, and the variations among the traffic intensities in its nodes. The four case studies are summurized in table I.

The mean arrival rates are generated randomly satistying the characteristics (3) and (4). The case study-4 is a reasonably balanced load, thus the tunning factor; alpha; is chosen to be one and serves only as a tie breaker.

Four performance indices are taken into account: the average response time per job (ART), the percentage of moved jobs to the total number of jobs, the percentage of the processing time at each node to the total processing time, and the average queue length at each node in the system.

The simulation results of the two strategies for the four case studies are summarized in table II after using jobs cutoffs. The effects of changing the network delay and the scheduling interval over the average response time are shown in figures (3) and (4) for two of the four case studies.

5. Conclusions

As shown in table II, both strategies give good effect on the system performance in each case study. The average response time is significally decreased, and the total processing time is distributed in an equal way over the system nodes. The sum of the average queue

Table I

Characteristic of the four case studies

Characteristics \ Case study number	1	2	3	4
1. number of nodes	3	5	5	5
2. Average service time (seconds)	5,6,7	5,5,6,7,7	5,5,6,7,7	5,5,6,7,7
3. Total traffic intensity of the system	0.7	0.7	0.8	0.7
4. traffic intensity of the nodes	0.662,0.658 0.855	0.453,0.873 0.798,0.596 0.793	0.81,0.85 0.88,0.82 0.56	0.689,0.72 0.694,0.668 0.717

TABLE II

Results of the four case studies

Case studies	Strategies	ART(Sec)	% Job Mov.	Average of percentage process time	Coefficient of variation of % process time	Sum* of Average queue length at the nodes	Coefficient** of variation of average queue length	Optimum margin	
								Lower	upper
Case study No. 1	No strategy	23.916	0.0	33.34	0.3296	8.65	0.4081	---	----
	strategy 1	12.579	23.58	33.33	0.0573	4.18	0.0431	2	12
	strategy 2	12.126	25.61	33.33	-0.0246	3.87	0.0233	2	12
Case study No. 2	No strategy	22.934	0.0	19.998	0.5881	12.86	0.2877	-	-
	strategy 1	12.199	24.55	20.002	0.1865	6.7	0.1104	3	20
	strategy 2	12.433	26.63	20.000	0.1320	7.19	0.0876	2	18
Case Study No. 3	No strategy	33.568	0.0	20.008	0.4493	23.35	0.3936	-	-
	strategy 1	15.624	21.58	20.000	0.2215	9.86	0.0791	3	18
	strategy 2	14.525	60.34	20.002	0.1145	8.72	0.0275	2	14
Case study No. 4	No strategy	17.696	0.0	19.998	0.2565	9.84	0.1098	-	-
	strategy 1	11.481	26.44	20.000	0.0965	6.29	0.0302	2	12
	strategy 2	11.932	35.92	20.000	0.1010	6.2	0.0306	2	16

* Sum of average queue length = $\sum_{\text{nodes}}$ Average queue length

** coefficient of variation = Range/sum of average queue length.

Fig. 3 The effect of NETWORK delay on ART

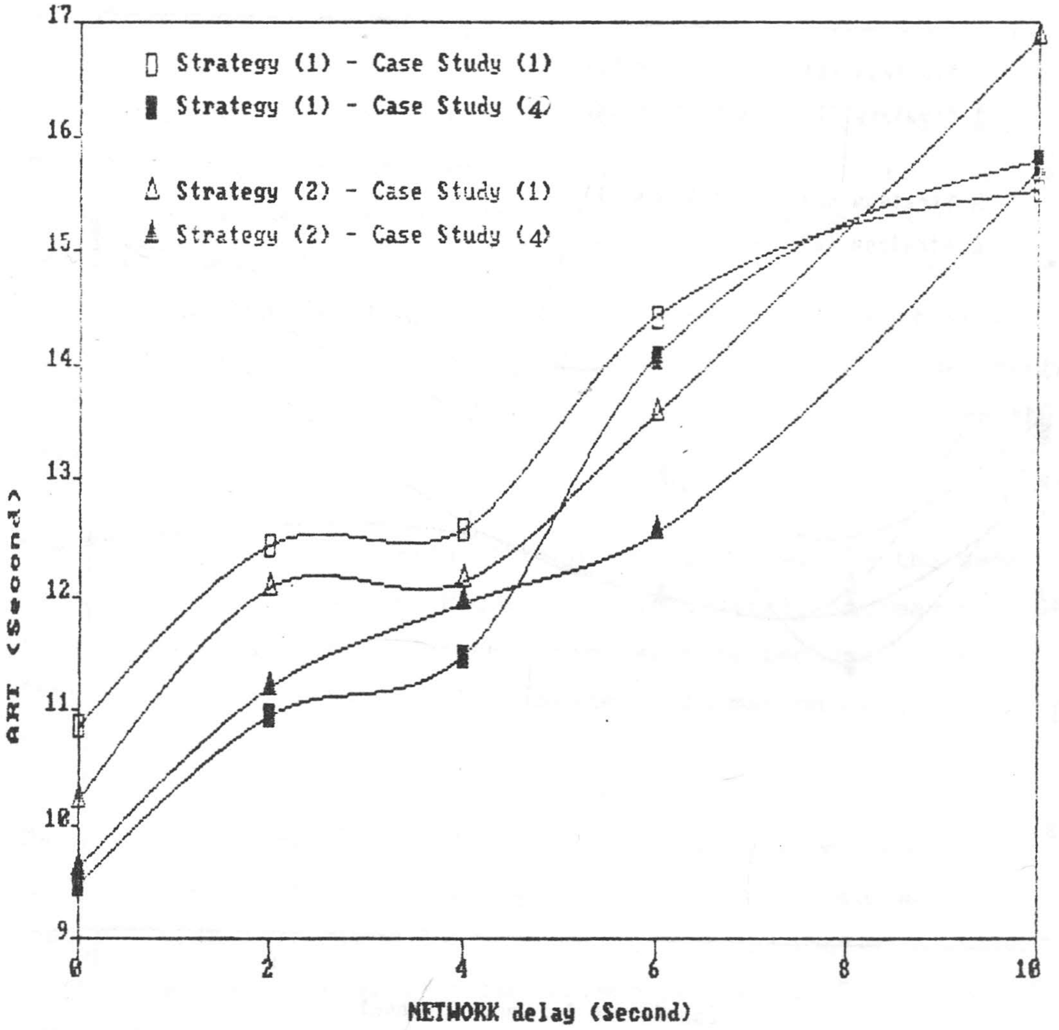
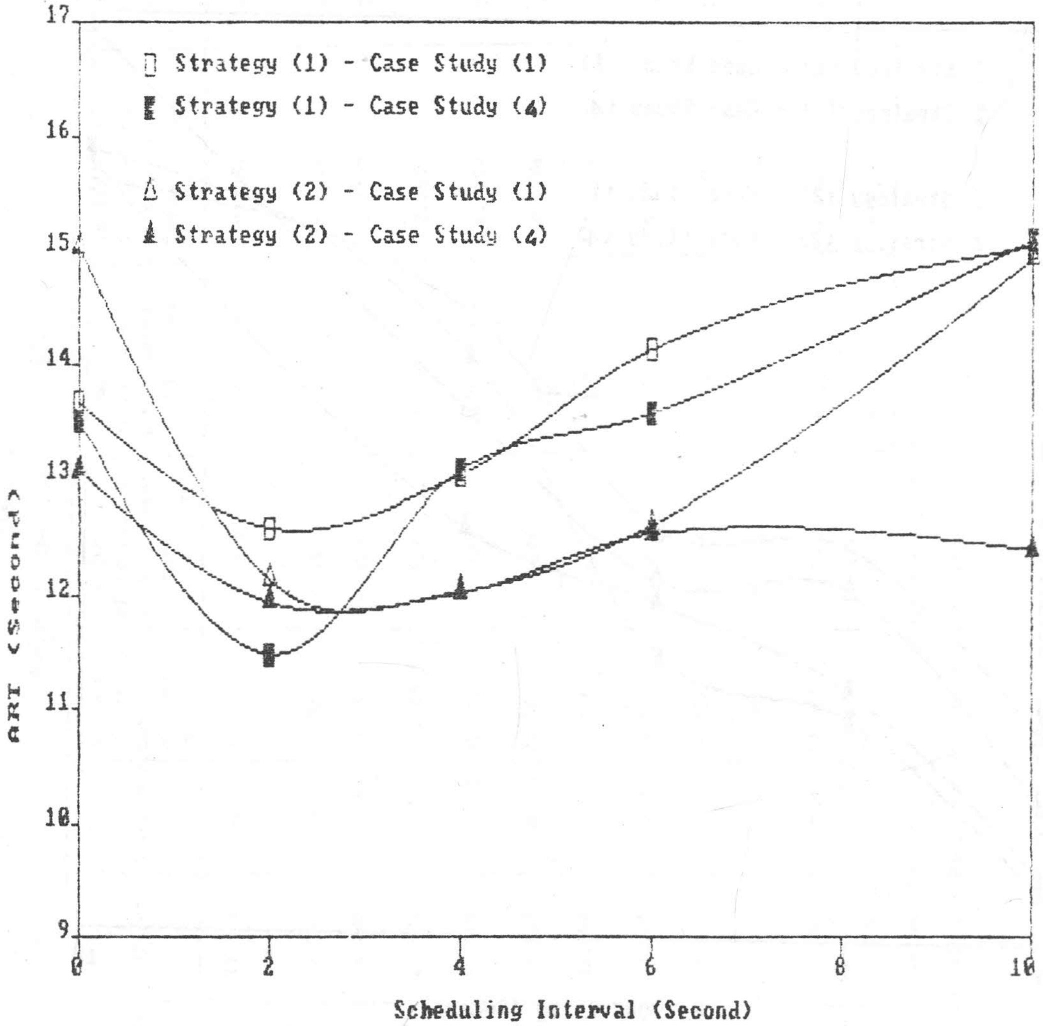


Fig. 4 The effect of Scheduling Interval on ART



length at all nodes is reduced when using both strategies besides the reduction of its coefficient of variation.

The good effect of the job scheduling algorithms seems to be very clear when the system load is unbalanced, as illustrated in the first and the second case studies.

By Comparing the two adaptive strategies, we found that each one has better performance in certain cases. The second strategy, performs good results because it distributes the moved jobs to different receivers, but it has a tendency to move too many jobs if not tuned well. While the first strategy sends fewer jobs at each interval with the tendency to direct most of the jobs to a single node, specially, when the system has only one light loaded node as shown in the third case.

The effect of increasing the networks delay is the same in both algorithms as shown in Figure (3). In general, increasing delays in the subnet increases system response time because increased delays degrade the quality of the state information of nodes about each other.

The scheduling interval must be chosen carefully because a large value will result poor balancing while a small value may produce instabilities. It is clear from Figure (4) that the bad effect of increasing the interval is more serious in the first strategy. This is expected because in the second strategy, many decisions are taken each interval, and so, increasing the scheduling interval will have a less effect.

It is worth mentioned that if the distributed system is geographically scattered, differences among the communication delays between the system hosts might be taken also into account.

References

[1] F. Baskett, K. M. Chandy, R.R. Muntz, and F.G. Palacios. Open, Closed, and mixed networks of queues with different classes of customers. JACM Vol. 22, No, 248-260, April 1975.

[2] W.W. Chu, L. J. Holloway, M.Lan, and K. Efe. Task allocation in distributed data processing. IEEE transactions on computers, Vol. C-13, 57-69, November 1980.

[3] Kemal Efe. Heuristic models of task assignment scheduling in distributed systems. IEEE computer, Vol. 15, 50-56, June 1982.

[4] L. Kleinrock. Queueing systems. Vol. I: Theory, Wiley-Interscience, New York, (1975).

[5] P.Y. R. Ma, E. Y. S. Lee, and M. Tsuchiva. A task allocation model for distributed computing systems. IEEE transactions on computers, Vol. C-31, 41-47, January 1982.

[6] A.N. Tantawi and Don Towsley. Optimal load balancing in distributed systems. Research Report, IBM Thomas J. Watson Research Center, January 1984.

[7] A.N. Tantawi and Don Towsley, A general model for optimal load balancing in star network configurations. Research Report, IBM Thomas J. Watson R.C., May 1984.

[8] V. L. Wallace and R.S. Rosenburg. Markovian models and numerical analysis of computer system behavior. SJCC, Vol. 28, 141-148, (1966).

[9] M.A. Iqbal, J. Saltz, and S. Bokhari, "A comparative analysis of static and dynamic load balancing strategies, "In Proc. 1986 Int. Conf. Parallel processing, 1040-1047, (1986).

[10] H. Lu and M.J. Carey, "Load-balanced task allocation in locally distributed computer systems," in Proc. 1986 Int. Conf. Parallel processing 1037-1039, (1986).

[11] Shahid H. Bokhari, "Partitioning problems in parallel, pipelined, and distributed computing," IEEE Transactions on Computers vol. 37, number 1, 48-57, January 1988.