An Extenics-based Load Balancing Mechanism in Distributed Computing Systems

Der-Fu Tao[†‡]*, and Liang-Teh Lee*[‡]

[†]Department of Electronic Engineering, Northern Taiwan Institute of Science and Technology
[‡]Department of Computer Science and Engineering, Tatung University

Summary

In distributed computing systems, load balancing is one of the most important factors that affect the system performance. This paper presents a sender-initiated dynamic load balancing policies in a distributed computing system. There are four policies consisted in the proposed Extenics-based Load Balancing Mechanism (ELBM). For the transfer policy, instead of defining a fixed threshold value in the traditional load balancing policies, we are applying the extension set theory and using the average response time of jobs as a factor to create an adaptive threshold value which is calculated by relational function of average response time, for determining that a new arriving job should be migrated or not. In addition to the transfer policy, bypass-transfer method is used in the location policy. Jobs that need to be migrated do not need to be transferred to central node first, but are transferred to destination node directly, so as to reduce the communication costs. In order to make the system more reliable, a template queue is added at the master node of the system to reduce the job arrival failure. A simulation model has been built for evaluating the performance of the system. Comparing with other load balancing mechanisms, such as RT and ALBCII algorithms, the simulation results show that a better performance can be achieved by the proposed mechanism.

Keywords: distributed computing system, extenics-based load balancing mechanism, extension theory, response time, bypass-transfer.

1. Introduction

A distributed computing system is considered as a collection of autonomous computers (nodes) located at possibly different sites and connected by a communication network [3]. The availability of low-cost microprocessors and advances in communication technologies has spurred considerable interest in distributed computing systems. The primary advantages of these systems are high performance, availability, and extensibility at low cost. To realize these benefits, system designers must overcome the problem of allocating jobs evenly in a distributed computing system so that the system resources can be fully utilized [2, 9].

Through the communication network, in a distributed computing system, users at different locations can share resources of the system. To improving the load-balancing mechanism so as to enhance the system performance is one of the most important issues in distributed systems. The proposed Extenics-based Load Balancing Mechanism (ELBM) consists of four policies, transfer policy, location policy, selection policy, and information policy. The transfer policy determines a job should be performed locally or not. In generally, most algorithms used the CPU waiting queue length to define the threshold of every node in distributed computing system. Once the number of jobs exceeds the threshold, the new arriving job should be transferred to another node for processing. The location policy [4] is used to determine the destination, which a selected job should be transferred. There are three basic methods of location policy: random, threshold, and shortest queue methods. The random location method is simply to select a destination node in random, it may result in a useless job transfer if the random selected node is heavy-loaded. Similar to random location method, the threshold_method also selects destination node randomly but it migrates the job when it finds the destination node with the CPU queue length less than the threshold value is found before the poll limit value has been reached. The shortest queue location method selects destination node with the shortest CPU queue length. The selection policy is used to select a job for migrating if necessary. Most algorithms proposed previously select the newest arriving job to transfer. The information policy [5] gathers and maintains the system information for the transfer decisions. There are three classifications of information policy: demand-driven policy, periodic policy and state-change-driven policy. Demand-driven policy is used in decentralized topology to collect the state of other nodes only when it becomes either a sender or a receiver. Periodic policy collects information periodically. In state-change-driven policy, nodes disseminate information about their states whenever their states changed by a certain degree.

The transfer policy of our proposed ELBM, we define a relational function by using the average job response time to be a factor for determining whether a job should be transferred or not. In order to avoid extra job transfer and reduce the time of decision making in location policy, instead of random, threshold, and shortest

queue location policies, we use a master node to provide the information for senders to dispatch the jobs that need to be transferred to destination nodes. In information policy, state-change-driven policy is applied. Once a job is completed in a node, information policy is triggered. Similar to most previous proposed algorithms, a new arrival job is selected for migration in the selection policy of our proposed load balancing mechanism.

2. Extenics-based Load Balancing Mechanism

Many proposed load-distributing algorithms previously, such as sender-initiated algorithm and receiver-initiated algorithm, are using static thresholds for their transfer policies to determine whether the jobs should be migrated or not. However, their static thresholds may not be suitable for the system with system-state changed frequently. To overcome this drawback, a dynamic threshold generation function using extension theory [1, 8] has been proposed in this paper. The threshold is generated with examining the system condition at all time and will be more adaptive than fixed ones. For algorithms using star topology, such as RT and ALBCII [5], when a job should be migrated, the job must be passed through the central node to destination node. The path of job migration will cause the central node to become a bottleneck of the system. Therefore, the bypass-transfer method is used in the location policy of proposed algorithm. In this method, jobs are migrated to destination node directly and will not pass through the master node. Consequently, using bypass-transfer method will reduce the time wasting in migrating jobs significantly. The detail of the proposed ELBM will be illustrated below.

2.1 System Overview

The system model of distributed computer system in this study is shown in Fig. 1. The model is a distributed computer system that consists of many slave nodes and one master node connected through a communication network in star-like topology. The slave nodes are receiving jobs, processing jobs, and transferring jobs. In addition to the operations that slave node processes, the master node collects the state information of slave nodes and locates the selected jobs if they should be migrated. When a newly job arrives at a node, for balancing the system loads, the transfer policy of proposed mechanism is applied first. In the transfer policy, an adaptive threshold generating function is used to generate threshold for determining whether the newly arriving job should be migrated or not. To compose of the adaptive threshold generating function, the concept of extension theory has been used. If the transfer policy decides to transfer the selected job, the location policy is triggered. The location policy of proposed mechanism is used to locate the destination node for job migration. Applying the proposed bypass-transfer method for migrating jobs, communication costs will be reduced significantly. The information for determining where a job should be transferred is collected through the information policy. The state-change-driven method is adopted as the information policy in the proposed mechanism. Once a node finishes a job, the necessary information should be transferred to master node.

2.2 Extension set and Extended relational function

In the real word, many problems appear to be unsolvable but can be made solvable through some methods of transformation. For example, it is impossible for utilizing a steelyard to weigh an elephant. However, the well-known story of “Tsao Chung weighs an elephant,” happened in ancient China, revealed a possible way of solving this weighing problem. The extension theory, first proposed by Cai in 1983, discusses how to systemically transfer or solve the contradiction or incompatible problems into solvable issue [1]. In our proposed mechanism the extended relational function and extension set are used to define the threshold dynamically in transfer policy. The extension set and extended relational function will be illustrated below.

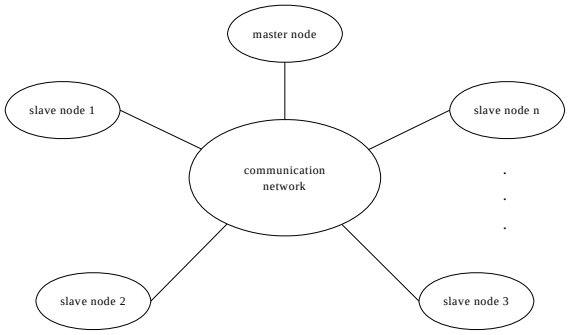

Fig. 1 System Model.

2.2.1 Extension set

In the Cantor set, a datum or element either belongs to a set or not. But there is something different in innate characteristics which element does not belong to the sets. For example, as shown in Fig. 2, the qualified work-pieces, which have been processed by a lathe, are specified with a diameter of $50 \pm 0.1mm$. Processed work-pieces may be divided into two portions, i.e., qualified and unqualified. For the unqualified work-pieces, some have diameters $d > 50.1mm$, and some have $d < 49.9mm$. The diameters $d > 50.1mm$ are called wasted products. But the latter can be transformed into qualified ones if reprocessing the work-pieces is allowed. Clearly, both the wasted and re-workable products are not qualified. But they are essentially different.

2.2.2 Extended relational function

The extended relational function is defined to quantify the relationship between an element and a set. The range of extended relational function is $(-\infty,+\infty)$ which means that an element belongs to any set with a certain degree. In the universe of discourse *U* shown in Fig. 2, we define an extended relational function by $k(x) \in (-\infty,+\infty)$ for any $x \in U$.

(1) If $k(x) \geq 0$, such as the qualified products, then it belongs to the positive set,
$X = \{x \mid k(x) \geq 0, x \in U\}$.

(2) If $k(x) < -1$, such as the unqualified products, then it belongs to the negative set,
$\overline{X} = \{x \mid k(x) < -1, x \in U\}$.

(3) If $k(x) = 0$, such as the critical products, then it is called the zero point.

(4) If $-1 \leq k(x) < 0$, such as the re-workable products, then it belongs to the extension set,
$\underline{X} = \{x \mid -1 \leq k(x) < 0, x \in U\}$.

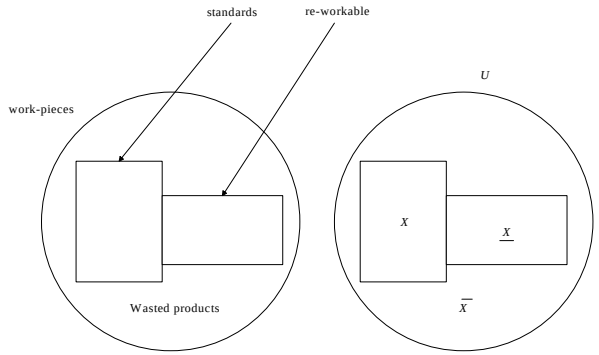

Fig. 2 The concept of extension sets.

The distance between a datum *x* (wasted or re-workable product) and the real interval *Xo*=[a_0,b_0] can be defined as $d = |x - \frac{a_0 + b_0}{2}| - \frac{1}{2}(b_0 - a_0)$ and denoted by P(*x, Xo*). Note that the distance is different from the distance in classic mathematics. If *x* is outside the interval, the distance between *x* and the interval is the closest distance of the endpoint to *x*. Besides, in classic mathematics, the distance is 0 while *x* is in the interval. But in extension sets, the distance is negative while *x* is in the interval. Besides the relationship between a point and an interval, we need to consider the relationship among intervals and point to two intervals. For example, there is a rated voltage but still has a tolerant voltage in any electrical machine. Certainly, if the voltage is over or below the tolerance, the electrical machine will break down or cannot work properly. Therefore, we name the relationship between inside or outside a datum *x* and two intervals, *X* and *Xo*, as positional distance and denoted it by D (*x, Xo, X*). The following states the basic properties. Assume

(1) *Xo* = [a_0,b_0] and *X* = [c_0,d_0] are two intervals in the real domain,

(2) $X \supset Xo$, and

(3) They have no common end point.

Then, the positional distance can be defined as:

$$D(x, Xo, X) = \begin{cases} p(x, X) - p(x, Xo), & x \notin Xo \\ -1, & x \in Xo \end{cases}$$

In our mechanism, the transfer policy is using extended relational function to determine whether a new arrival job should be transferred. In other words, instead of using fixed threshold, the proposed mechanism uses extended relational functions to generate an adaptive threshold.

2.3 Load balancing mechanism

The proposed load balancing mechanism is based on a sender-initiated extenics-based load balancing system. As mentioned before, the mechanism applies an adaptive threshold generating function and bypass-transfer method in transfer policy and location policy. The notations used in our load balancing mechanism are listed as follows.

Job_i: current jobs at node *i*

$Threshold_i$: threshold of node *i*

$AveExe_i$: average job execution time at node *i*

JTCC: communication cost of job transfer

2.3.1 Adaptive threshold generating function

The transfer policy in the proposed mechanism, we build an adaptive threshold generated by extended relational function for determining whether the job should be transferred or not. Three parameters *M, Y,* and *G* are defined for generating the threshold value.

M is the smallest *AveExe* Job* of the cluster,

Y is the first fifty percentage *AveExe* Job* of the cluster, and

G is the largest *AveExe* Job* of the cluster.

The relationship among *M, Y,* and *G* is shown in Fig. 3. Once a job arrives at a node, the transfer policy applies the extended relational function to generate a threshold for determining whether the job should be transferred or not. The extended relational function used to generate the adaptive threshold is defined below.

Xo is the interval (*Y,G*); the region that jobs have to be transferred,

X is the interval (*M,Y*); the region that jobs need not be transferred,

P (x,Xo) is *|x–(Y+G)/2|-(G-Y)/2*

D (x,Xo,X) is defined as

$$\begin{cases} p(x, X) - p(x, Xo) \text{ if } x \text{ is not included by } Xo \\ -1 \text{ if } x \text{ is included by } Xo \end{cases}, \text{ and}$$

then we can calculate the $k(x) = \frac{P(x, Xo)}{D(x, Xo, X)}$ as our adaptive threshold value.

In the above function, x is the value of $AveExe_i * Job_i$ in node i for evaluating $k(x)$ in the transfer policy**.** If the value of $k(x)$ is positive, the new arrival job should be transferred. If the value of $k(x)$ is negative, the new arrival job should not be transferred. $k(x)$ is the dynamic threshold of a node, and it can be adjusted according to the system state dynamically. Since the need transfer region (*Xo*) and need not transfer region (*X*) may be extended or shrunk while the system state is changed, by applying the extended relational function to generate the dynamic threshold of all nodes adaptively can avoid the situation that migrates jobs to heavy-loaded nodes while loads of some other nodes are light in the fixed threshold mechanism.

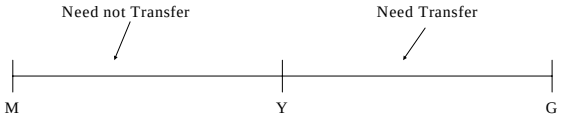

Fig. 3 Relationships of M, Y, and G.

2.3.2 Bypass-transfer method

In the proposed location policy, once the transfer policy decides to migrate a job, the slave node sends a message to the master node for making determining the location. If the number of jobs at node *i* exceeds the threshold (of node *i*), the master node should find out minimum average job execution time ($AveExe_j * job_j + JTCC$) according to the collected information from slave nodes to migrate the selected job. Then master node compares ideal job execution time at node *i* ($AveExe_i * job_i$) with job execution times at other nodes ($AveExe_j * job_j + JTCC$). If master node can find minimum job execution time of other nodes shorter than ideal job execution time at node *i*, master node will send message to node *i* to tell node *i* to perform the job transfer. Otherwise, node *i* should process the job by itself (if the number of current jobs at node *i* does not exceed threshold of node *i*). Fig. 4 shows the location policy of our mechanism. In the proposed mechanism the bypass-transfer method is used for migrating jobs. Since jobs transfer need not pass through the master node, the communication cost of job transfer can be reduced significantly.

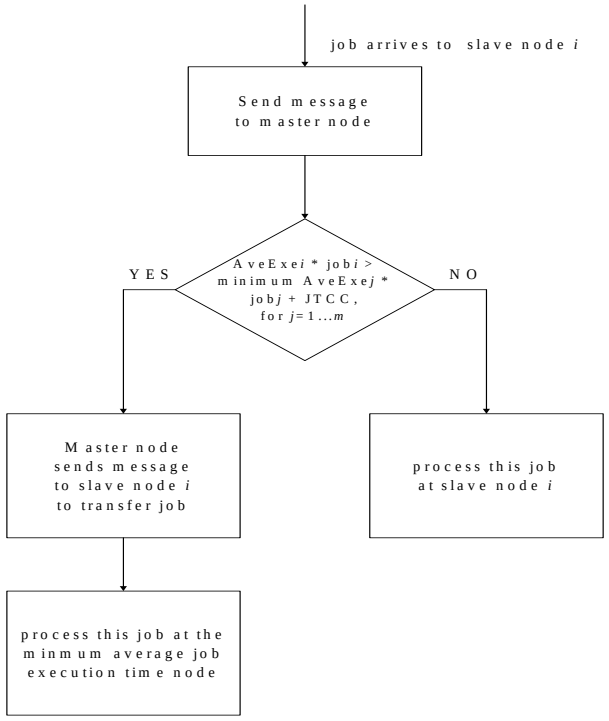

Fig. 4 Location policy.

2.3.3 Information policy and selection policy

State-change-driven information policy is used in proposed information policy and it is shown in Fig. 5. After a job is completed in the slave node, it must send the response time of this finished job to master node for calculating the mean job response time of this slave node. According to the above operations, the master node could know the state of every slave node and could make a best decision for determining the location, which the job should be transferred to. In the selection policy of our mechanism, the new arriving jobs are selected for migrating if it is necessary.

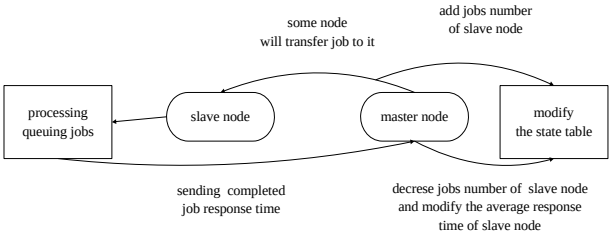

Fig. 5 Information policy.

2.3.4 System extension

The proposed mechanism can be extended easily if it is necessary to connect with many clusters. When connecting more than one cluster, the communication between clusters just need to send message between master nodes of clusters because they hold all slave nodes

information of clusters. For system extension, the location policy is also according to two factors: average job execution time and communication cost. The master node of each cluster holds the information of the smallest value of *AveExe* * *Job* of its own cluster and also holds the values of the smallest *AveExe* * *Job* of other clusters. Therefore, when master nodes make decision for determining the location which the job should be transferred to, they can ponder all clusters in the system. The configuration of multiple clusters is shown in Fig. 6.

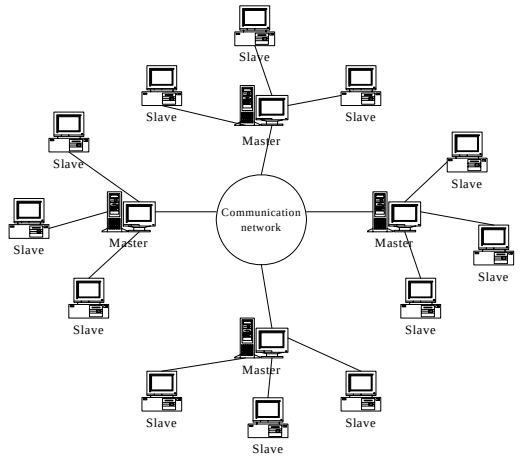

Fig. 6 System extension.

3. Experiments and result analysis

The target system is modeled and simulated on a desktop PC with SMPL (**S**i**M**ulation **P**rogram **L**anguage) [6]. In this simulation, the number of node is chosen as 10 including the master node, the polling limit *Lp* as 2, and threshold value T_i as 3. The processing power and the mean job arrival rate of each node are chosen according to the purpose of experiments. The inter-arrival time of jobs and the service time are assumed to have an exponential distribution. The job transmission time between the master node and the slave node is assumed to have exponential distribution with mean of 5. The communication cost due to polling or information gathering is assumed to be negligible.

3.1 The simulation model

By using *SMPL*, we construct a simulation model to evaluate the performance of the proposed mechanism [7]. This model comprises three routines, as shown in Fig. 7. Arrival event reveals the arriving of a job to nodes on the cluster. When arrival event occurs, the master node should determine the destination node for this job and then make the destination node request event occur. Reserve event represents that a job reserves a node to perform the job. Release event represents that a job service is completed and the server is released for other request use.

3.2 Results analysis

Four mechanisms are compared: NOLB, RT, ALBCII, and ELBM. Fig. 8 and Table 1 show that a better performance can be achieved by applying the propose algorithm.

The performance comparison of the four algorithms is shown in Fig. 8. With the light system load, we can see that the mean job response times of all four algorithms are with little difference, even in the NOLB system that is not applying the load balancing mechanism. However, when the system load is going heavy, the mean job response times of all four algorithms are quite different. Obviously, the performance of a heavy loaded system is affected by the load balancing mechanism significantly. In a heavy loaded system, without applying load balancing mechanism will slow down the system performance. The system performance of applying RT mechanism is better than that applying NOLB. However, there is a drawback in RT, that is, if a node cannot poll light node at all time will slow down the system performance. In other words, when the system load is light, it is easy to poll the light-loaded nodes to transfer jobs by using RT algorithm; however, when the system load is becoming heavy, it is very difficult to find light-loaded nodes to transfer jobs, thus the polling operation will result in a congested system and degrade the system performance.

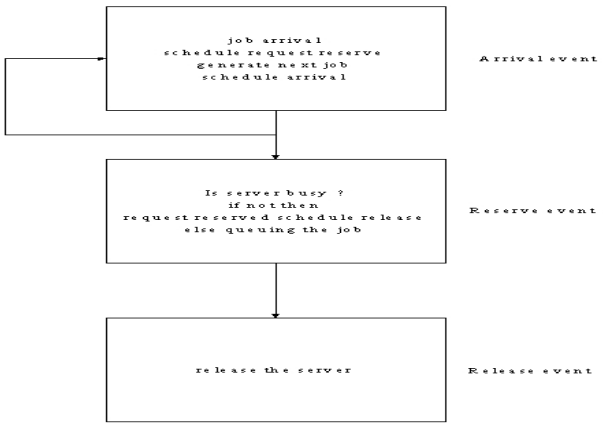

Fig. 7 SMPL simulation mode.

Fig. 9 and Table 2 show the simulation result of the proposed mechanism with 0.5 mean job arrival time. From the simulation result, we can see that the best performance is achieved when the mean job response time is evaluated in the first fifty percentages. Thus, in the proposed mechanism the transfer policy decides to transfer the job if the average job response time of the job arriving node is the first fifty percentages of all nodes in the cluster.

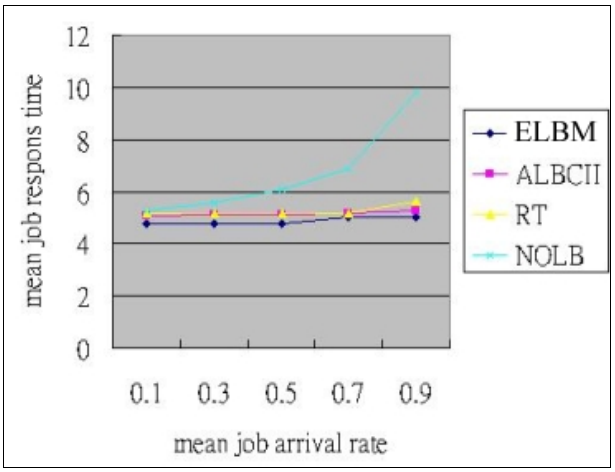

Fig. 8 Performance comparison.

Table 1: Mean job response time of algorithms

Algorithms	Mean job response time				
	Mjar=0.1	Mjar=0.3	Mjar=0.5	Mjar=0.7	Mjar=0.9
ELBM	4.7563	4.7628	4.7832	5.002	5.0283
ALBCII	5.0617	5.1039	5.1593	5.1872	5.2734
RT	5.1592	5.1684	5.2361	5.4738	5.6475
NOLB	5.2504	5.5632	6.0743	6.8946	9.7835

Mjar: Mean job arrival rate

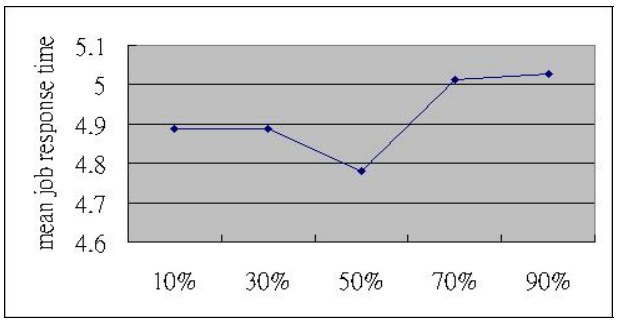

Fig. 9 The first percentage of job response time in a cluster.

Table 2: Mean job response time at different percentage

The first percentage of job response time in a cluster	10%	30%	50%	70%	90%
Mean job response time	4.889	4.889	4.78	5.013	5.028

The performance of the ELBM with and without relational function (RF) is also compared. The simulation result is shown in Fig. 10 and Table 3. We can know that the performance of the ELBM with relational function is better than without using relational function. Since the bypass-transfer methods is applied in the ELBM, the performance of both mechanisms, with and without RF, are better than ALBCII.

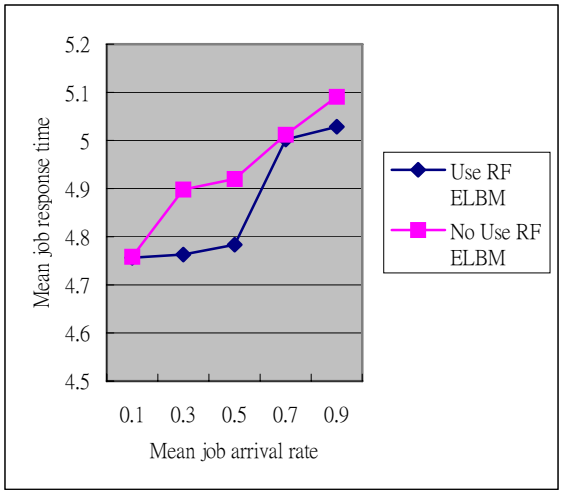

Fig. 10 Using or not using RF in the ELBM.

Table 3: Comparison of the ELBM with and without RF

Algorithms	Mean job response time				
	Mjar=0.1	Mjar=0.3	Mjar=0.5	Mjar=0.7	Mjar=0.9
Use RF ELBM	4.7563	4.7628	4.7832	5.002	5.0283
No use RF ELBM	4.7586	4.898	4.92	5.012	5.0909

Mjar: Mean job arrival rate

4. Conclusion

In this paper we proposed an ELBM for distributed computing systems, this mechanism is using a threshold generating function to generate an adaptive threshold dynamically instead of traditional fixed threshold in the transfer policy. Through this adaptive threshold, a better performance of the system can be achieved than using the fixed threshold, since the system using fixed threshold does not ponder the state-change of the system. In the star topology, all job migrations must pass through the central node before jobs can be transferred to the destination node, and will cause the central node to be a bottleneck. To overcome this problem, in the proposed location policy, we apply the star-like topology and the bypass-method for job migrating. Through the bypass-method, the job migrations do not pass through the master node but pass to the destination node directly. This kind of job migration will reduce the communication cost of transferring time significantly.

References

[1] W. Cai, "The extension set and incompatible problem," Journal of Scientific Exploration, vol. 1, pp. 81-93, 1983.

[2] T. L. Casavant and J. G. Kuhl, "Effective of Response and Stability on Scheduling in Distributed Computing Systems," IEEE Transactions on Software

Engineering, Vol. 14, No. 11, pp. 1578-1588, Nov. 1988.
[3] H. Ashihara, N. Mizuguchi, and M. Maekawa, "Reduction of Information Exchange for Adaptive Load Sharing in Distributed Systems," ICPADS, pp. 92-96, 1991.
[4] N.G. Shivaratri, P. Krueger, and M. Singhal, "Load Distributing for Locally Distributed Systems," IEEE Computer, Vol.25, No. 12, pp. 33-44, Dec, 1992.
[5] Kyung-Soo Lim, Chong-Gun Kim, and DGun Kim, "Dynamic Load Balancing in Distributed Computer Systems with Star Topology," Proceedings of IEEE 15th International Conference on Distributed Computing Systems, pp. 514-517, Aug, 1995.
[6] M. H. MacDougall, "Simulating Computer Systems: Techniques and Tools," http://www.amazon.com/gp/product/026213229X/002-6367517-9268065?v=glance &n =283155.
[7] Kameda, H., Fathy, El-Z. S., Ryu, I. and Jie Li, "A performance comparison of dynamic vs. static load balancing policies in a mainframe-personal computer network model," IEEE Conference on Decision and Control, pp. 1415 - 1420, Dec. 2000.
[8] Zeng Zeng and Veeravalli, B., "Rate-based and queue-based dynamic load balancing algorithms in distributed systems," ICPADS, pp. 349 - 356, July 2004.
[9] Liudong Xing and Shrestha, A., "Distributed Computer Systems Reliability Considering Imperfect Coverage and Common-Cause Failures," ICPADS, pp. 453 - 457, July 2005.

Der-Fu Tao received the B.S. degree in Electronic Engineering from National Taiwan University of Science and Technology in 1990, and his M.S. degree in Computer Science and Engineering from Tatung University in 1996. Currently, he is the Senior Lecturer of the Department of Electronic Engineering, Northern Taiwan Institute of Science and Technology and he is the Ph.D. candidate in Tatung University. His research interests include architecture design of ATM switch, parallel and distributed systems, embedded systems, and high speed switching architectures.

Liang-Teh Lee received the B.S. degree in Electrical Engineering from Tatung University in 1971 and his M.S. and Ph.D. degree in Computer Engineering from University of Southern California in 1978 and 1989, respectively. He was the director of the Computer Center and the chairman of the Department of Computer Science and Engineering of Tatung University. Currently, he is the Professor of the Department of Computer Science and Engineering, Tatung University. His research interests include computer architectures, parallel and distributed computing, embedded systems, and high speed switching architectures.