On Load Balancing Model for Cluster Computers

Huajie Zhang

School of Math, Physics and Information Engineering College of Zhejiang Normal University, Jinhua 321004, China

Summary
Recently, cluster computers become a viable and less expensive alternative to multiprocessor systems. However, a serious difficulty in concurrent programming of a cluster computer system is how to deal with scheduling and load balancing of such a system which may consist of heterogeneous computers. Based on the four-tuple *DLBM* (dynamic load balancing model) which is proposed by other scholars, this paper introduces a five-tuple *DLBM* to better solve the above problems. The five-tuple involves balancing environment, task allotment, load estimate, scheduling strategy and scheduling evaluation. The five-tuple *DLBM* interprets the full aspects of load balancing and makes the logical relationship between the five tuples more clear by using formalized language. According to the five-tuple model and the corresponding solution of each tuple, this paper realizes the model under certain experimental conditions in the Linux cluster system. Experiments are given testing the five-tuple *DLBM* based on tasks that are independent, having a little or no relativity, uncertain executing time. The result shows that the model proposed in this paper can make the load balancing better. What's more, scheduling evaluation shows that the strategies are reasonable.

Key words:
Parallel, Cluster computers, Load balancing model, Load estimate

1. Introduction

Today with the scale of scientific computing areas going up, an increasing requirement of application's performance is needed. Thus, cluster computing becomes a feasible and cheap way to achieve high performance. However the approaches to handling scheduling and load balancing in homogeneous and heterogeneous cluster system are not adequate yet. Usually homogeneous workstations may have processors, memory and I/O with identical specifications. But it's also not cheap for some researchers to buy large numbers of homogeneous computers. Also Moore's law says that CPU clock speeds double every 18 months. Thus, constructing heterogeneous workstations is necessary. For improving the power of clustered computers, load balancing is the key technology of cluster systems [2], [3].

The distributed system can conveniently realize lots of parallel processes, and it has many advantages such as the usability of system's resource, expansibility of the scale and parallelism. How to implement the management and scheduling of the distributed system is the critical factor in mining system's potential parallelism sufficiently. In general, a distributed system needs to process many tasks synchronously and every computing node needs to handle many processes. So it is necessary to use scheduling and assignment algorithm to optimize the distribution of task, which will shorten the response time availably and reduce extra overhead in the execution. Thus load balancing is an important way to improve the distributed system's performance, and one of its aims is improve the system's performance, cut down the average response time of customer's task, the other aim is to utilize the entire system's resource equally and adequately. These two aims are coherent as utilization of the resource gets more balanceable, the response time of tasks gets shorter [3–5].

In general, there are two scheduling approaches in load balancing, static scheduling and dynamic scheduling. Static scheduling includes lots of schemes such as simulated annealing algorithm [6], genetic algorithm [7], heuristic algorithm[8], algorithm based on chart theory [9], etc. Dynamic scheduling includes some schemes like algorithm based node's probability of choosing task randomly [10], method based on gradient model [11], an adaptive nearest neighbor contract algorithm [12], etc.

There exists many ingredients to affect load balancing, and the implement of scheduling schemes are different. Many load balancing techniques designed to support distributed systems have been proposed and reviewed in the literature [13–18].

After considering some of load-balancing mechanism, this paper proposed a five-tuple load-balancing model, on the base of four-tuple load balancing model proposed by literature [1]. Load-balancing ranges over many aspects that are physical environment, software environment and all kinds of strategies. They are influenced by each other. The five-tuple load balancing model involves hardware environment, scheduling environment, task allotment, load estimate, scheduling strategy and scheduling evaluation. They are all expressed through mathematical definition. The five-tuple load balancing model interprets the full aspects of load balancing and makes the logical relationship between the five tuples more clearly by using formalized language. We use the five-tuple $DLBM = (BE, T, LE, SS, EC)$ to describe the load balancing.

According to the five-tuple *DLBM* and the corresponding solution of each tuple, the paper realized the

Manuscript received October 5, 2008
Manuscript revised October 20, 2008

model under a certain experimental conditions based on the Linux cluster system. Task allotment is completed by transferring array's index. It's simple and is no need to access kernel files. This paper adapts the max-min method which is better than before in the aspect of estimate accuracy. Furthermore, the machine states are reduced into two states, one is free and the other is overloaded. First of all, the former sends a request to the control node. On receiving the request from the idle node, control node calculates the most overloaded node that is to send the first index of tasks to idle node. In this way, all the nodes spend less time on scheduling the tasks. Experiments are made according to the five-tuple load balancing model based on tasks that are independent, having a little or no relativity, uncertain executing time. The result shows that the strategies proposed in this paper can make the load balancing better. What's more, scheduling evaluation shows that the strategies are reasonable.

The remainder of this paper is organized as follows. All five elements are discussed respectively in the section 2. In section 3, experimental results are presented, and some conclusions and future work are given in section 4.

2. Theoretical Consideration

2.1 Dynamic load balancing model

Definition 1($DLBM$): Dynamic load balancing model is a five-tuple like (BE, T, LE, SS, EC) and every element indicates its own meaning as follows:

BE : Balancing Environment;

$T = \{T_1, T_2, T_3, ..., T_P\}, T \neq \phi, p \in R^+$: Customer's task, $p \in R^+$, p denotes computing node;

$LE = \{LE_1, LE_2, LE_3, ..., LE_p\}$: Load Estimate, p denotes computing node;

SS : Scheduling Strategy;

EC : Evaluate Criterion.

After interpreting every element, the figure which describes $DLBM$ is given as follows:

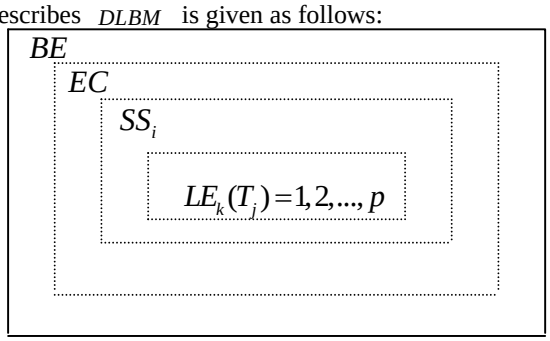

Fig. 1 Dynamic load balancing model.

2.2 Balancing environment

Define 2: The cluster's balancing environment BE is a three-tuple, $BE = (CS, SE, SU)$ and every element respectively denotes:

CS (Cluster System): the cluster's physical network arrangement. Cluster system is a mutual cooperation and resource-sharing system composed by a group of distributed workstations under the communication protocol of physical connection. The dynamic load balancing strategy presented in this paper is to use large number of idle computer resources of lab or office LAN, also for the purpose of fully mining the existing computers' idle period. Such environment is connected by ordinary LAN hub or switch and is composed of a variety of computer resources, such as PC etc. Similarly to COW (Cluster of Workstation), it is composed of one main controlling node and other computing nodes, called radial topological structure.

SE (Scheduling Environment): the node's logical topology. In the scheduling process of load balancing, the distributed system makes up of logical map topology, namely scheduling environment, in accordance with the cooperation agreement between the nodes, nature of tasks and different load balance scheduling strategy. The paper is mainly for relative and independent tasks which communicate with each other only at the beginning or in the end.

SU (Scheduling Unit): the nodes participated in the computing. $SU = \{SU_1, SU_2, SU_3, ..., SU_q\}, q = 2^p - 1$

p denotes computing node.

Scheduling implementation of load balancing needs nodes involved in. Each node is a unit that implementing tasks and transfer or receive tasks following scheduling strategy. The paper is designed as that the main control computer generates child process which sees to corresponding node to collect and send messages.

2.3 Task Set

Definition 3 (Task Set): The task set is a three-tuple like $T = (TT, TP, TR)$, and every element is interpreted as follows:

$TT = \{TT_{cpu}, TT_{IO}, TT_{cpu,IO}\}$: The task's type of the customer submitted. TT_{cpu} is a CPU-dependent task that requires prolonged occupation of CPU, TT_{IO} is an I/O-dependent task that needs a large number of I/O operations, $TT_{cpu,IO}$ is also needs CPU and I/O resources.

$TP = \{TP_{task}, TP_{data}\}$: Task divided pattern by customer. The task is decomposed in two dimensions. Task decomposition dimension sees the problem as an

instruction stream, and the instructions are decomposed in many tasks' sequence, and the tasks can be carried out synchronously. Data decomposition TP_{data} focuses on analyzing the data that task requires, analyzing how to split the data into many different blocks.

$TR = \{TR_1, TR_2, TR_3\}$: The relativity among every task. Time correlation TR_1, that is, restraint of tasks' executing order. One task's implementation is dependent on another task's executing result. When another task is completed, the task could be carried out. For example, $T_1 : A(i) = B(i) + X, T_2 : C(i) = A(i) + Y$ while one node is dealing with T_1, the other processes T_2, apparently T_2 is dependent on T_1. When a task set is required to run at the same time, like T_1 and T_2 is executed on the same node, then another type of sequential restraint happens, namely T_2. In many parallel programs, the initial problem is split into many blocks. These blocks should be updated synchronously; otherwise, the parallel problem may halt or be in deadlock. In some cases, tasks in a group are completely independent, namely T_3. These tasks are not bound by the order, so they can be implemented in any order, including parallel implementation. This article is designed base on TR_3.

2.4 Load Estimate

Definition 4(Load Estimate): Load Estimate is a five-tuple like $L = (LP, LF, LT, LS, \beta)$, here $LP = \left(LP_1, LP_2, LP_3, ..., LP_p, T_{LP}\right)$, p denotes the number of computing nodes, LP denotes the load parameter of p computing nodes at the time T_{LP}. There are many common used load parameters, such as the length of CPU queue, the average CPU queue in a period time, the size of available memory, the frequency of system calls, the utilization of CPU and memory, frequency of interruption etc. It is found that using different parameters for load balancing algorithm has a significant impact on performance, and simple parameters are often more effective. LP_i is defined as follows:

$LP_i = (CPU, I/O, MEM, Q, t), i = 1, 2, ..., p;$

CPU : CPU utilization in period time

I/O : Utilization of I/O resources in period time

MEM : Memory utilization in period time

Q : Length of CPU queue in period time

t : Length of time for collecting load parameters' information.

Elements in LP , some have percent value and some have specific integral value, so we use weight denoting load. The weight is greater, the load is busier.

LF : Load Function. If the node i has the parameter LP_i , LF is defined as follows:

$LF(LP_i) = \mid L_i \mid$, L_i is the current actual load of node i .

From the above definition, LP only provides a weight value and is embodied in a measure that the busy level of nodes. However, LP_i can be transformed into quantitative load through load function.

LT : Load Threshold. LT is an ordered pair $< \gamma_1, \gamma_2 >, \gamma_1, \gamma_2 \in R, \gamma_1, \gamma_2 > 0$ and $\gamma_1 \le \gamma_2$.

LS : The set of Load State. $LS = (NL, LL, SL, HL)$, NL denotes empty load, LL denotes light load, SL denotes suitable load and HL denotes high load or overload. According to γ_1, γ_2 , load state of workstation can be partitioned NL, LL, SL, HL .

β : Load state modifying factor. β is used to revise LT , showing autonomy of working node. It can adjust the value of load threshold and denotes a self-adaptive load balance scheduling strategy, $\beta \in R$, $\beta \neq 0$. If $LF < \gamma_1, \gamma_2 >$ is used, β is used to modify as follows:

$$\Omega = \{< x, y > \mid x \in \{r_1 - \beta, r_1, r_1 + \beta\} \wedge y \in \{r_2 - \beta, r_2, r_2 + \beta\} \wedge 0 < x \le y\}$$

LT should be chosen in this range but β is unnecessarily the same to modify up threshold and down threshold. It needs be revised according to the fact.

2.5 Scheduling Strategy

Definition 5(Scheduling Strategy): Scheduling strategy is a four-tuple like $SS = (SD, ST, SF, SC)$, here

SD : Scheduling Domain. In the distributed network environment, workstation that participates in load balancing work and provides users network computing services is transparent. In general, only some workstation resources provide services for users through load balancing strategy, for the purpose of balancing resources and responding users in the shortest time. These workstations are called scheduling domain.

ST : Scheduling Type. Form different point of view, job scheduling has different classification method. This paper uses the structure of scheduling program or the scope of scheduling information to partition, namely centralized

scheduling shown in figure2 and distributed scheduling given in figure3. The former is used in this paper.

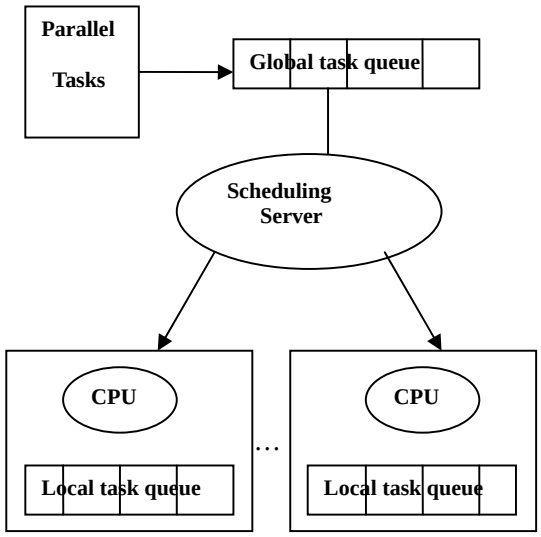

Fig.2 Centralized scheduling

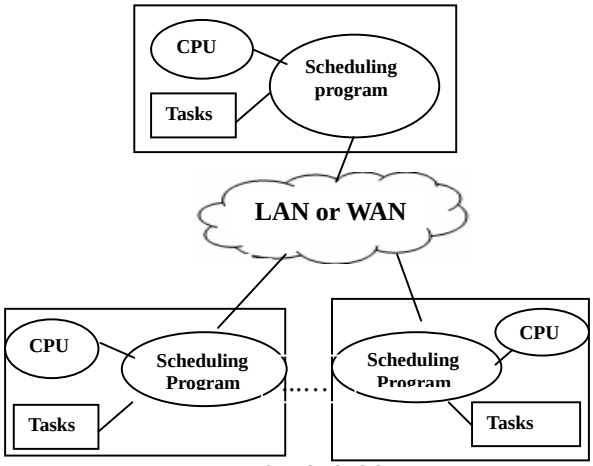

Fig.3 Distributed scheduling

SF : Scheduling Framework. Scheduling framework contains four components: transfer strategy, selection strategy, positioning strategy and information strategy. They form the entire life cycle of load balancing scheduling activities. In the phase of transferring, the master node and slave nodes for task migration are determined. The main control processor is notified of the quantity and slave node of tasks for load balancing. In the phase of selection, main control processors select the most suitable tasks for efficient and effective load balancing and send them to the appropriate slave nodes. In the third phase, tasks' receivers are determined. In static scheduling algorithm, the server is responsible for positioning and collecting system information and the sender gets information from the server to determine which node to receive. In dynamic scheduling, polling algorithm is widely used [19], namely the sender asks each of the other nodes in order to determine whether load can be shared. The last phase decides how to collect information, where to collect and which kind of nodes' status messages to collect, etc. There are three familiar information strategies such as demand driven strategy, cycle strategy and status change driven strategy.

SC : Scheduling Constraint. Different scheduling framework has different scheduling constraint, and different SC determines different scheduling algorithms. Because the parallel scheduling is a typical NP-complete problem, different scheduling algorithms are adopted according to problems' characteristic such as task divide pattern.

The paper is mainly for relative and independent tasks which communicate with each other only at the beginning or in the end.

2.6 Scheduling Evaluate Criterion

As is known to everyone, most experiments always use some criteria to evaluate, and then this paper is no exception.

Definition6 (Scheduling Evaluate Criterion): The evaluate criterion is a three-tuple like $SE = (LE, AL, SDL)$.

LE (Load Efficiency): T_1 denotes the time of completing all the tasks without DLB, T_2 denotes the time of completing all the tasks with DLB.

$$\eta = \frac{T_1 - T_2}{T_1} \tag{1}$$

AL (Average Load): The average load at every moment, and its computing formula is as follows:

$$Load_{avg} = \frac{1}{N_t} \sum_{t=1}^{N_t} Load_{avg,t} \tag{2}$$

$Load_{i,t}$ denotes the load of node i at time t, N_p denotes the number of the nodes executing tasks, N_t denotes the time interval of collecting info in the execution, and its value is always an average one. Generally speaking, high efficiency load balancing makes the average load $Load_{avg}$ monotone increasing in fixed percentage along with the increase of task.

SDL (Standard Deviation of Load): deviation of node's real load and average load at every moment. It is defined as:

$$Load_{std} = \frac{1}{N_t}\sum_{t=1}^{N_t}\sqrt{\frac{1}{N_p - 1}\sum_{i=1}^{N_p}(Load_{i,t} - Load_{avg,t})^2} \quad (3)$$

Variables in formula (3) are the same as in formula $Load_{avg}$. In general, high efficiency load balancing keeps $Load_{std}$ in a smaller domain along with the increase of task.

3. Experimental Consideration

3.1 Experiment Environment

This paper adopted centralized dynamic load balancing using Master/Slave mode. The figure is shown as fig.4.

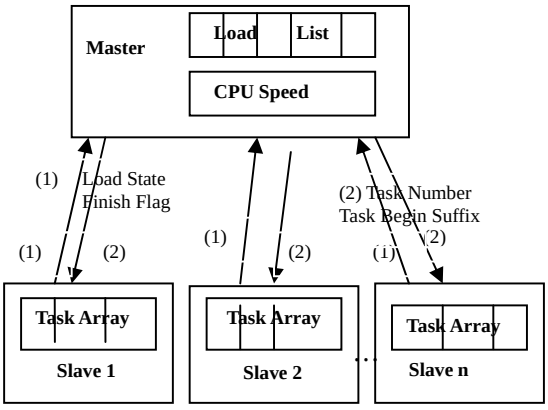

Fig.4 Parallel program structure

The experimental environment in the multiple-cluster environment implementation had four computing nodes numbered from 1 to 4, forming 2 clusters. Computing nodes 1, 2, and 3 formed the relatively homogeneous cluster A, with 4 CPUs, as shown in table 1. Computing nodes 1, 2, and 4 formed relatively heterogeneous cluster B, with 4 CPUs, as shown in table 2. Each cluster system had two slave nodes and one master node, computing node 1, called HPC.

3.2 Experiment Result

This paper constructed a basic testing case characterized by independence of task, little communication between tasks and uncertain execution time of task. We use LE (load efficiency), AL (average load) and SDL (standard deviation of load) to estimate performance. According to the uncertain execution time of task, this paper suppose the task as multidimensional matrix, dimension is generated randomly and ranges from 50 to 100. Due to uncertain dimension, tasks are independent. The testing case experimented in three respects. First, the comparative result of executing tasks with no-DLB and DLB based on homogeneous system. Second, the comparative result of executing tasks with no-DLB and DLB based on heterogeneous system. Third, according to the proposed model, this paper estimates performance based on heterogeneous system. Experimental results are shown in table 3 and table 4.

Table 1: Hardware configuration of Cluster A

Cluster A	Node name	CPU	Memory
1	HPC	3.06GHZ	521M
2	Snode1	1.8 GHZ	128M
3	Snode2	1.8 GHZ	128M

Table2. Hardware configuration of Cluster B

Cluster B	Node name	CPU	Memory
1	HPC	3.06GHZ	521M
2	Snode1	1.8 GHZ	128M
4	Snode3	1.7 GHZ	512M

Table3.The results of executing tasks with no-DLB and DLB based on the homogeneous system

Total Task	Time with no-DLB (s)	Time with DLB (s)	Load Efficiency
30	60.22	65.06	-0.080
40	67.50	75.7	-0.121
50	81.31	98.04	-0.206
60	100.21	120.70	-0.204
70	116.86	110.22	0.057
80	130.32	102.28	0.215
90	162.53	116.85	0.281
100	201.54	152.30	0.244
200	375.22	251.20	0.331

From what is shown, when the number of the task decreases, like total task 30, 40 in table 3, the cost time margins narrow because task doesn't migrate in load balancing. However, it makes load computing and information transfer overhead and hasn't shown dominance. When the number is 50 or 60, the time with DLB is still larger than time with no-DLB, that's because

deviation of remain load makes the redistribution unreasonable and the time of beginning task number 20 or 30 is so short that it leads to no rebalancing. Meanwhile, load Efficiency shown in table 3 is upwardly mobile along with the number of task increasing, although arising jolty phenomenon.

Table 4. The results of executing tasks with no-DLB and DLB based on heterogeneous system

Total Task	Time with no-DLB (s)	Time with DLB (s)	Load Efficiency
30	56.12	60.64	-0.081
40	70.24	80.30	-0.143
50	83.55	90.46	-0.082
60	110.01	90.73	0.175
70	125.82	145.86	-0.159
80	146.26	167.01	-0.142
90	180.13	165.15	0.083
100	226.22	173.67	0.232
200	512.45	451.56	0.113

The results based on heterogeneous system is shown in table 4. It seems more complicated and costs more time than homogeneous system. One of the reasons is the performance of node itself is poorer than the one based homogeneous system. The other reason is division in term of executing speed is not very precise. The last reason is task's reallocation doesn't consider nodes' performance again leading to the number of transferring task inaccuracy. Therefore, it's suitable for homogeneous system.

3.3 Performance Evaluation

(1) Load Efficiency
Load efficiency is also shown in table 3 and table 4.When Load efficiency is above and beyond 0, Dynamic load balancing is efficient, for details see section 3.2.
(2) Average Load
According to the formula (2) in section 2.6, we can obtain the following line chart as shown in figure 5.

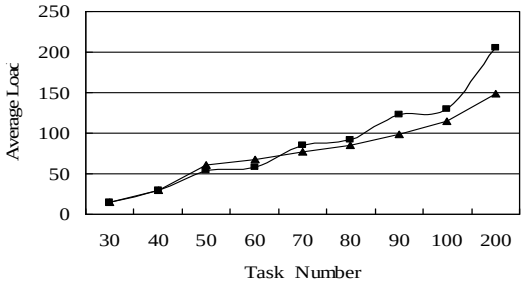

Fig.5 Line chart of average load

(3) Standard Deviation of Load
We have seen the formula (3) in section 2.6 and figure 6 shows its line chart.

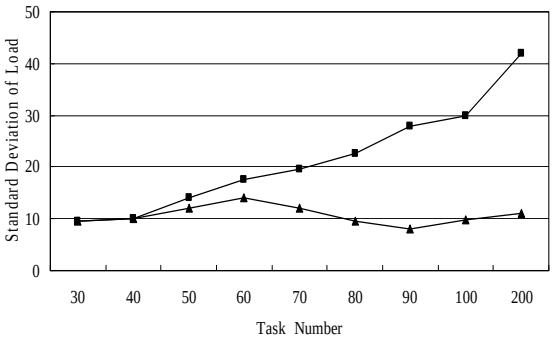

Fig.6 Line chart of standard deviation of load

From table 3, we can see that when the number of task climbs to 70, average load with DLB is lower than that with no-DLB. When the number is 30 or 40, average load is similar. That's because the number is too small and no need to transfer task. We see the same state arising in table 4. All in all, DLBM proposed by this paper really has efficiency on load balancing. Although it can't apply into all environments, it is suitable for independent tasks.

4. Conclusion

On the base of four-tuple DLBM (dynamic load balancing model) proposed by literature [1], this paper introduces a five-tuple DLBM which involves balancing environment, task allotment, load estimate, scheduling strategy and scheduling evaluation. They are all expressed through mathematic definition. The five-tuple DLBM interprets the full aspects of load balancing and makes the logical and makes the logical relationship between the five tuples more clear by using formalized language. In the near future, we will solve these problems that how to estimate surplus load more accurately and how to make the load transfer more self-adaptively.

Acknowledgment

The authors would like to express their cordial thanks to my supervisor Dr. Jiuzhen Liang for his valuable advice.

References

[1] Donghai Li,Haihu Shi Study on general load balancing scheduling scheduling model. *Computer Engineering and Applications*, 43(8):121-125, 2007(in Chinese).

[2] Christopher A. Bohn, Gary B. Lamont. Load balancing for heterogeneous clusters of PCs. *Future Generation Computer Systems*, 18:389-400, 2002.
[3] Hui Chi-Chung,Chanson S T. Hydrodynamic load balancing. *IEEE Transactions on Parallel and Distributed System*, 10(11): 1118-1137, November 1999.
[4] Marc H,Willebeek-L M. Strategies for dynamic load balancing on highly parallel computers. *IEEE Transactions on Parallel and Distributed System*, 4(9): 979-993, September 1993.
[5] Chao-Tung Yang and Shun-Chyi Chang. A parallel loop self-scheduling on extremely heterogeneous PC clusters. *Journal of Information Science and Engineering*, 20(2): 263C273, 2004.
[6] Kyung-Geun Lee, Soo-Young Lee. Efficient parallelization of simulated annealing using multiple Markov chains:an application to graph partition. *Proceedings of the 1992 International Conference on Parallel Processing*, 177-180, 1992.
[7] Kwok Y-K,Ahmad I. Efficient scheduling of arbitrary task graphs to multiprocessors using a parallel genetic algorithm. *Journal of Parallel and Distributed Computing*,47(1): 58-77, 1997.
[8] Efe E. Heuristic models of task assignment scheduling in distributed systems. *IEEE Computer*, 15(6): 50-56, June 1982.
[9] Hendrickson B and Leland R. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2): 452-469, 1995.
[10] S. Chakrabarti, A. Ranade, and K. Yelick. Randomized load balancing for tree-structured computation.*IEEE Scalable High Performance Computing Conference*, 666 -673,May 1994.
[11] Lin F C H,Keller R M. The gradient model load balancing method. *IEEE Transactions on Software Engineering*, 13(1): 32-38, January 1987.
[12] Derek L. Eager, Edward D. Lazowska, John Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, 12(5): 662-675, May 1986.
[13] T. L. Casavant and J. G. Kuhl. A taxonomy of scheduling in general purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14(2): 141-154, Feb 1988.
[14] Y.-T. Wang and R. J. T. Morris, Load sharing in distributed systems. *IEEE Transactions on Computers*, 34(3): 204-217,March 1985.
[15] M. J. Berger and S. H. Bokhari, A partitioning strategy for no uniform problems on multiprocessors. *IEEE Transactions on Computers*, 36(5): 570-580, May 1987.
[16] Fox, Geoffrey C. A review of automatic load balancing and decomposition methods for the hypercube. *Institute for Mathematics and Its Applications*, 13: 385, Nov 1988.
[17] K. Ramamritham, J. A. Stankovic, and W. Zhao. Distributed scheduling of tasks with deadlines and resource requirements. *IEEE Transactions on Computers*, 38(8): 1110-1123, August 1989.
[18] K. M. Baumgartner, R. M. JSling, and B. W. Wah. Implementation of GAMMON: An efficient load balancing strategy for a local computer system. *International Conference on Parallel Processing*,2:7- 80,August 1989
[19] Zhou Songnian, A trace-driven simulation study of dynamic load balancing. *IEEE Tranctions on Software Engineering*, 14(9): 1327-1340, September

Huajie Zhang received the Bachelor's degree of Computer Science and Technology from HeNan Institute of Finance and Economics in 2005. Then she worked in e-commercial company hc360 for one year. During 2006-2008, she stayed in Zhejiang Normal University of China to study Computer Software and Theory and will get B.S. degree in 2009.