Balanced Ant Colony Optmization Algorithm for Job Scheduling in Grid Computing

R. Tharani
Assistant Professor
Dept. of CSE
JCT College of Engineering and Technology
Coimbatore, Tamil Nadu, India

.*Abstract*—**Grid computing is a novel technology for building high speed computing environment in which heterogeneous, distributed and dynamically available resources are integrated. However, in Grid environment, it is a big challenge to design an efficient scheduler and its implementation. It aims to find a suitable allocation of resources for each job. In the natural environment, the ants have a tremendous ability to team up to find an optimal path to food resources. An ant algorithm simulates the behavior of ants. In this paper, we propose a Balanced Ant Colony Optimization (BACO) algorithm for job scheduling in the Grid environment. The main contributions of our work are to balance the entire system load while trying to minimize the makespan of a given set of jobs. Compared with the other job scheduling algorithms, BACO can outperform them according to the experimental results.**

Keywords—Grid computing, BACO, Ant colony optimization algorithm, Grid scheduling.

I. INTRODUCTION

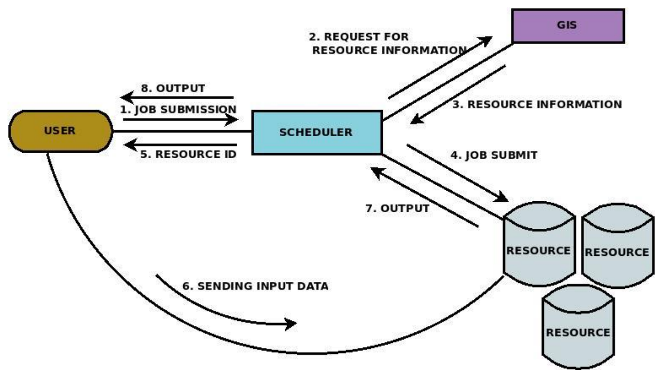
In grid computing environment, applications are submitted for use of grid resources by users from their terminals. The resources include computing power, communication power and storage. An application consists of number of jobs; users want to execute these jobs in an efficient manner. There are two possibilities of submission of jobs/data on resources; in one of them, job is submitted on the resources where the input data is available and in the other, on the basis of specific criteria, resource is selected on which both job and input data are transferred. This paper uses second approach, wherein the job is submitted on a scheduler and data on a resource identified by the scheduler. A resource in existing algorithms is selected randomly, sequentially or according to its Grid Computing enables sharing, selection, aggregation of geographically distributed resources dynamically at run time depending on their accessibility, ability and users Quality of Service requirements. The main objective of the grid technology is to maximize the utilization of the organization's computing resources by making them as shareable entities, and provide computing on demand to the users. Balancing the load of all available resources is another important issue in the grid.

The demand for scheduling is to achieve high performance computing. Typically, it is difficult to find an optimal resource allocation for specific job that minimizes the schedule length of jobs. The scheduling problem is defined NP-hard problem and it is not trivial. The motivation of this paper is to develop a grid scheduling algorithm that can perform efficiently and effectively in terms of minimizing total tardiness time. Not only does it improve the overall performance of the system but

it also adapts to the dynamic grid system. First of all, this paper proposes an Ant Colony Optimization (ACO) algorithm

to find the optimal resource allocation of each job within the dynamic grid system. Secondly, the simulation of the experiment is presented. This simulation is an extension of GridSim toolkit version 4.0, which is a popular discrete-event simulator and grid scheduling algorithm. The simulator defines the different workload of resources, the arrival time of independent jobs, the length of each job, the criteria of a scheduler, etc. Finally, this paper compares the performance of various job schedulers and dispatching rules for grid environment within fully controlled conditions. The experiment considers:

- First Come First Served (FCFS)
- Minimum Time Earliest Due Date (MTEDD)
- Minimum Time Earliest Release Date (MTERD)
- Ant Colony Optimization (ACO)



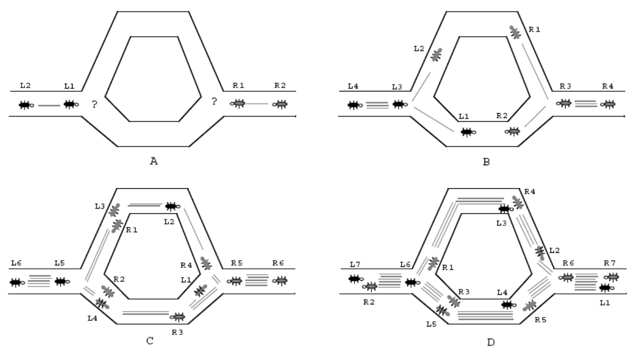
. *Fig.1. Steps for job submission on a resource*

II. LITREATURE REVIEW

In the past few years, researchers have proposed scheduling algorithms for parallel system [2 - 6]. However, the problem of grid scheduling is still more complex than the proposed solutions. Therefore, this issue attracts the interests of the large number of researchers [7- 11]. Current systems [17] of grid resource management was surveyed and analyzed based on classification of scheduler organization, system status, scheduling and rescheduling policies. However, the characteristics and various techniques of the existing grid scheduling algorithms are still complex particularly with extra added components. At the present time, job scheduling on grid computing is not only aims to find an optimal resource to improve the overall system performance but also to utilize the existing resources more efficiently. Recently, many researchers have been studied several works on job scheduling on grid environment. Some of those are the popular heuristic algorithms, which have been developed, are min-min [12], the fast greedy [12], tabu search [12] and an Ant System [13]. The heuristic algorithms proposed for job scheduling in [12] and [13] rely on static environment and the expected value of execution times. H. Casanova et al. [18] and R. Baraglia et al. [19] proposed the heuristic algorithms to solve the scheduling problem based on the different static data, for example, the execution time and system load. Unfortunately, all of information such as execution time and workload cannot be determined in advance of dynamic grid environments. In 1999, the Ant Colony Optimization (ACO) metaheuristic was proposed by Dorigo, Di Caro and Gambardella which has been successfully used to solve many NP-problem, such as TSP, job shop scheduling, etc. In the past few years, several researchers proposed solutions to solve grid scheduling problem by using ACO [20]. Several studies have been trying to apply ACO for solving grid scheduling problem. Z. Xu et al. [21] proposed a simple ACO within grid simulation architecture environment and used evaluation index in response time and resource average utilization. E. Lu et al. [22] and H. Yan et al. [23] also proposed an improved Ant Colony algorithm, which could improve the performance such as job finishing ratio.

III. THE PROPOSED SYSTEM

Ant Colony Optimization (ACO) is one of the metaheuristics. It can be applied not only to solve discrete optimization problems but also to solve both static and dynamic combinational optimization problems. ACO is inspired by a colony of artificial ants cooperate in foraging behavior.This behavior enables ants to find the shortest paths between food sources and their nest. In fact, they deposit a chemical pheromone tail on the ground after they walk from the nest to food sources and vice versa. Hence, they choose the way with higher probability paths, which are marked by stronger pheromone concentrations. This behavior is the basis for a cooperative interaction.



. *Fig.2. illustrate of how real ants can lead to identify the shortest path around an obstacle.*

A. *The balanced ant colony optimization (BACO) algorithm*

BACO inherits the basic ideas from ACO algorithm to decrease the computation time of jobs executing in Taiwan UniGrid environment and it also considers about the loading of each resource. BACO changes the pheromone density according to the resource status by applying the local pheromone update and the global pheromone update functions. The purpose is to try to minimize the complete time for each job while balancing the system load.

B. *System architecture*

The system architecture is shown in Fig. 3. There are four main components: Portal, Information Server, Jobs Scheduler and grid resources. The Portal provides an interface to users for job execution. The Information Server collects resource information by using the NWS (Network Weather Service). The NWS demon reports system information back to Information Server periodically. The Jobs Scheduler selects the most appropriate resources to execute the request according to the proposed BACO algorithm. Finally, the execution results would be sent back to the user.

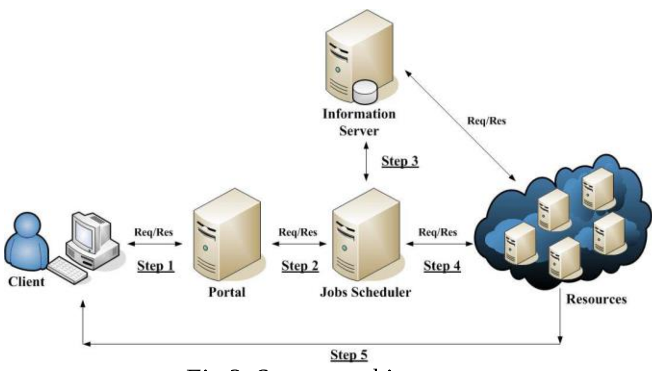


Fig.3. System architecture

C. *The proposed BACO algorithm*

In order to map the ant system to the grid system, we explain
their relationships below:

a. An ant

An ant in the ant system is a job in the grid system.

b. Pheromone

Pheromone value on a path in the ant system is a weight for a resource in the grid system. A resource with a larger weight value means that the resource has a better computing power. The scheduler collects data from Information Server and uses the data to calculate a weight value of a resource. The pheromone (weight) of each resource is stored in the scheduler and the scheduler uses it as the parameters for BACO algorithm. At last, the scheduler selects a resource by a scheduling algorithm and it sends jobs to the selected resource by the APIs of the Globus Toolkit.

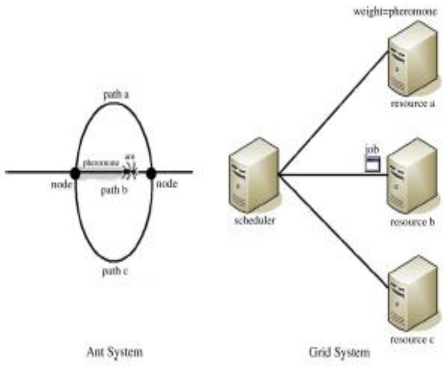


Fig.4. shows the mapping between the ant system and the grid system.

The initial pheromone value of each resource for each job is calculated based on the estimated transmission time and execution time of a given job when assigned to this resource. The estimated transmission time can be determined by $\frac{S_j}{bandwidth_r}$ where S_j is the size of a given job j and $bandwidth_r$ is the bandwidth available between the grid resource broker and the resource. The initial pheromone value is defined by:

$$PV_{ij} = \frac{S_j}{bandwidth_r} + \frac{C_j}{MIPS_r * (1 - load_r)} \qquad (1)$$

where PV_{ij} is the pheromone value for job j assigned to resource r, C_j is the CPU time needed of job j, $MIPS_r$ is the processer speed of resource r and $1 - load$ is the current load of resource r. The load, processor speed and bandwidth can be obtained from grid information server.

Assume there are n jobs and m resources in the PV matrix:

$$PV = \begin{array}{c|cccc} & j_1 & j_2 & .. & j_n \\ \hline r_1 & PV_{11} & PV_{12} & .. & PV_{1n} \\ r_2 & .. & .. & .. & .. \\ .. & .. & .. & .. & .. \\ r_m & PV_{m1} & PV_{m2} & .. & PV_{mn} \end{array}$$

The largest entry from PV matrix which reflects the best resource, will be selected in each iteration. Assuming PV_{ij} is selected then job j will be processed by resource r. The local pheromone update is performed after job j has been assigned to resource r. This formula can only be applied to unassigned jobs in the PV matrix. The local pheromone update is formulated as follow:

$$\tau_{jr} = (1-\xi).\tau_{jr} + \xi.\tau_0 \qquad (2)$$

where ξ,0< ξ<1 and τ_0 are two parameters. The value of τ_0 is set to be the same as the initial value for the pheromone trails. A good value for ξ was found to be 0.1, while a good value for τ_0 was found to be $1/nC_{nn}$, where n is the number of resources and C_{nn} is the resource with high pheromone value. The effect of the local pheromone update is to ensure an already chosen resource less desirable for the following ant [10] which will increase the exploration of not yet visited resource.

When a job is completely processed, global pheromone update is performed to recalculate the entire *PV* matrix. After all ants have constructed a solution, the pheromone trails are updated according to the following formula:

$$\tau_{jr}(t+1) = (1-\rho).\tau_{jr} + \rho \quad \tau_{jr}^{bs} \qquad (3)$$

where $\Delta\ \tau_{jr}^{best} = 1/L^{best}$. This global pheromone update is limited to a specific upper and lower trail limit. The ant which is allowed to add pheromone may be the *iteration-best solution* or *global best solution.* If a specific resource is often used in the best solution, it will receive a larger amount of pheromone and stagnation will occur. So, lower and upper limits on the possible pheromone strengths on any resource are imposed to avoid stagnation. The imposed trails limits have the effects of limiting the probability ρ_{iu} of selecting resource u when ants is in resource i to an interval$[p_{min},p_{max}]$, with $0< p_{min} \le p_{ij} \le p_{max} \le 1$. With this minimum trail limit, the resource is less desire to be selected by the jobs.since it will select the resource that has the upper trail limit. For example, there are three jobs (j_1, j_2, and j_3) that need to be processed by three resources (r_1, r_2, and r_3) in the grid system. The initial status of each resource is shown in Table I and size of each job is 5MB, 3MB and 1MB. The CPU cycles needed for each job are 5M, 3M and 1M respectively.

TABLE I.STATUS OF EACH RESOURCE

Status	r1	r2	r3
Processor Speed (MIPS)	217	464	195
Load	15%	10%	20%
Bandwidth (Megabits/s)	10.62	24.50	12.62

The initial pheromone values of each entry obtained from equation (1) are shown in the following PV matrix:

$$PV = \begin{array}{c|ccc} & j_1 & j_2 & j_3 \\ \hline r_1 & PV_{11} = 2.01 & PV_{12} = 3.35 & PV_{13} = 10.04 \\ r_2 & PV_{21} = 4.63 & PV_{22} = 7.71 & PV_{23} = 23.14 \\ r_3 & PV_{31} = 2.34 & PV_{32} = 3.89 & PV_{33} = 11.68 \end{array}$$

The resource with high pheromone value will be selected by grid resource broker. So j3 will be processed by r2. After assigning j3 to r2, the local pheromone update is performed to the second row of r2. Column 3 is no longer needed because j3 has been assigned. The new PV matrix is as follows:

Local update

$$PV = \begin{array}{cc} PV_{11} = 2.01 & PV_{12} = 3.35 \\ \boxed{PV_{21} = 4.17 \quad PV_{22} = 6.95} & \\ PV_{31} = 2.34 & PV_{32} = 3.89 \end{array}$$

After r2 finished processing j3, the global pheromone update is performed to get the newest pheromone value for the next job submission. The newest status of each resource after the execution of j3 is as shown in Table II. The load status of each resource will be changed according to the size of the current load. On the other hand, the value is used in evaporation process.

TABLE II.UPDATE STATUS OF EACH RESOURCE

Status	r1	r2	r3
Processor Speed (MIPS)	217	464	195
Load	15%	25%	20%
Bandwidth (Megabits/s)	8.67	15.87	10.26
	0.00	0.05	0.00

The value of r2 is 0.05 and values for r1 and r3 are zero since they have not been assigned any job for execution. The new PV matrix is as follows:

$$PV = \begin{array}{cc} PV_{11} = 1.64 & PV_{12} = 2.73 \\ PV_{21} = 2.88 & PV_{22} = 4.81 \\ PV_{31} = 1.93 & PV_{32} = 3.22 \end{array}$$

The remaining job will be assigned in the same way. The local pheromone update will be performed after a grid resource broker assigned a job to a resource. After a resource finished processing a job, all entries of the PV matrix will be updated by the global pheromone update rules.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

The performance of the proposed algorithm has been compared to other algorithms such as particle swarm, space shared and time shared. The average completion time has been used to compare the performance of the algorithms.

Two experiments have been conducted to evaluate the performance of the proposed algorithm in terms of their processing time. First experiment processed 10 jobs with 100 resources while the second experiment processed 100 jobs with 1000 resources. Details of the scheduling parameters are shown in the Table 3.

TABLE III. SCHEDULING PARAMETERS FOR THE EXPERIMENTS

Experiment	1	2
No. of machine per resource	1	1
Number of PEs per machine	1 – 5	1 – 5
PE ratings	10 or 50 MIPS	10 or 50 MIPS
Bandwidth	1000 or 5000	1000 or 5000
	B/S	B/S
Number of Gridlet	100	1000
Number of Resource	10	100

The purpose of simulating the mixed size is that there may be many different jobs in the grid and we want to know how the BACO algorithm performs in such dynamic situation. We choose 1000 jobs for execution and set the number of each size to one third of the total number of jobs. So the number of jobs is 333, 333, and 334, respectively for each size whether in matrix multiplication or in linear programming. We compare the makespan (total execution time) and the standard deviation of load in each method.

Fig.5 show the makespan of each method with mixed sizes. We can find out that BACO uses less time to complete all jobs. In the case of mixed jobs, the pheromone update functions of BACO still work well. For iACO, it applies the encouragement and punishment methods. It changes each pheromone by variables defined by users and it ignores the real status of resources. If the resources with bad computing power never fail to execute jobs, they will always be encouraged and get more pheromone. Then iACO will assign more jobs to the bad resources which have higher pheromone and this will increase the total execution time of the given jobs. That is the reason why iACO has larger makespan than BACO.

The prediction means that it uses user-defined variables to encourage or punish resources after the assignment of jobs or the completion of jobs. It may not work sometimes when the pheromone values do not match the real status of resources. So BACO can work better than iACO.

DFPLTF and Sufferage have similar performances and are also comparable to that of BACO. However, they do not consider the bandwidth issue and assume the data is readily available in the resources, which is sometimes not true.

FPLTF may have bad performance if the number of hard tasks is too many and it always assigns jobs to the fastest resources which may already have a heavy load. It will cause the system load to be unbalanced and take much time for job executions. Random has the true random performance. It does not consider about the status of resources and the size of jobs and assign jobs to resources randomly.

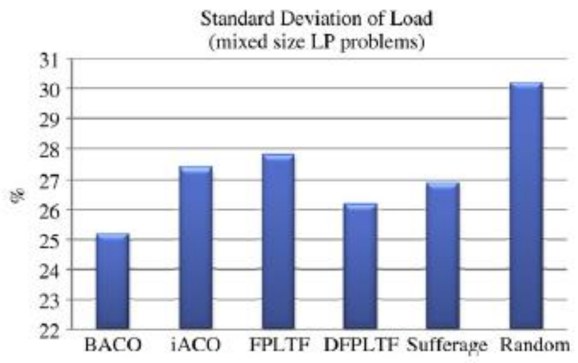


Fig.1. Makespan of each method with mixed sizes for matrix multiplication.

V. CONCLUSION

In this paper, we propose a BACO algorithm to choose suitable resources to execute jobs according to resources status and the size of given job in the Grid environment. The local and global pheromone update functions do balance the system load. Local pheromone update function updates the status of the selected resource after jobs assignment. Global pheromone update function updates the status of each resource for all jobs after the completion of a job. It offers the Job Scheduler the newest information of all resource for the next jobs assignment. The experimental result shows that BACO is capable of balance the entire system load.

VI. FUTURE WORK

In future, we will study whether there are any other situations which we do not take into account on our definitions of the pheromone indicator or the pheromone update functions. We will also try to apply BACO algorithm to various grid computing applications. For example, instead of independent jobs, assume now we are scheduling workflows. That is, there are precedence relations among jobs.

Then BACO has to be modified to include a synchronization scheme among resources. When a job is to be assigned to a resource for execution, we must be certain that all its precedent jobs running on other resources have been completed. Finally, this paper focuses on the computing grid. We may redefine the pheromone indicator and pheromone update formulations for the data grid to consider the replica strategy to select or predict which resources have more storage or are suitable for file replications by their newest status in future.

REFRENCES

[1] D.A. Reed, Grids, the TeraGrid, and Beyond, IEEE Computer 36 (1) (2003)62–68.
[2] BOINC website, http://boinc.berkeley.edu/.
[3] D. Kondo, D.P. Anderson, J. McLeod, Performance evaluation of scheduling policies for volunteer computing, in: Proc. IEEE International Conference on e-Science and Grid Computing, Dec. 2007, pp. 415–422.
[4] Ruay-Shiung Chang, Jih-Sheng Chang, Shin-Yi Lin, Job scheduling and datareplication on data grids, Future Generation Computer Systems 23 (7) (2007)846–860.
[5] Yang Gao, Hongqiang Rong, Joshua Zhexue Huang, Adaptive grid job scheduling with genetic algorithms, Future Generation Computer Systems 21(1) (2005) 151–161.
[6] EunJoung Byun, SungJin Choi, MaengSoon Baik, JoonMin Gil, ChanYeol Park, ChongSun Hwang, MJSA: Markov job scheduler based on availability in desktop grid computing environment, Future Generation Computer Systems 23 (4) (2007) 616–622.
[7] F. Dong, S.K. Akl, Scheduling algorithms for grid computing: State of the art and open problems, Technical Report No. 2006-504, School of Computing, Queen’s University, Kingston, Ontario, Canada, January 2006.
[8] M. Dorigo, C. Blum, Ant colony optimization theory: A survey, Theoretical Computer Science 344 (2–3) (2005) 243–278.
[9] M. Dorigo, Ant colony optimization, http://www.aco-metaheuristic.org.
[10] M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.
[11] E. Salari, K. Eshghi, An ACO algorithm for graph coloring problem, in: Congress on Computational Intelligence Methods and Applications, 15–17 Dec. 2005, p. 5.
[12] Xiaoxia Zhang, Lixin Tang, CT-ACO—hybridizing ant colony optimization with cycle transfer search for the vehicle routing problem, in: Congress on Computational Intelligence Methods and Applications, 15–17 Dec. 2005, p. 6.
[13] H. Yan, X. Qin, X. Li, M.-H. Wu, An improved ant algorithm for job scheduling in grid computing, in: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, vol. 5, 18–21 Aug. 2005, pp. 2957–2961.
[14] D. Saha, D. Menasce, S. Porto, Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures, Journal of Parallel and Distributed Computing 28 (1) (1995) 1–18.
[15] D. Paranhos, W. Cirne, F. Brasileiro, Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids, in: International Conference on Parallel and Distributed Computing (Euro-Par), in: Lecture Notes in Computer Science, vol. 2790, 2003, pp. 169–180.
[16] T. Stutzle, MAX-MIN Ant System for Quadratic Assignment Problems Technical Report AIDA-97-04, Intellectics Group, Department of Compute Science, Darmstadt University of Technology, Germany, July 1997.
[17] B. Bullnheimer, R.F. Hartl, C. Strauss, A new rank-based version of the ant system: A computational study, Central European Journal for Operations Research and Economics 7 (1) (1999) 25–38.
[18] E.D. Taillard, L.M. Gambardella, Adaptive memories for the quadratic assignment problem, Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.
[19] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B 26 (1) (1996) 29–41.
[20] Kwang Mong Sim, Weng Hong, Sun, Multiple ant-colony optimization for network routing, in: Proceedings of the First International Symposium on Cyber Worlds, 6–8 Nov. 2002, pp. 277–281.
[21] J. Heinonen, F. Pettersson, Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem, Applied Mathematics and Computation 187 (2) (2007) 989–998.
[22] Jianfu Li, Wei Zhang, Solution to multi-objective optimization of flow shop problem based on ACO algorithm, in: Proceeding of 2006 International Conference omputational Intelligence and Security, vol. 1, Nov. 2006, pp.417–420.
[23] E. Burke, G. Kendall, Silva Landa, R. O’Brien, E. Soubeiga, An ant algorithm hyperheuristic for the project presentation scheduling problem, IEEE Congress on Evolutionary Computing 3 (2005) 2263–2270.

[24] Walter, J. Gutjahr, M.S. Rauner, An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria, Computers & Operations Research 41 (3) (2007) 645–666.
[25] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Operating System Concepts, 7th edition, John Wiley & Sons, 2005.
[26] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R. Freund, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing system, Journal of Parallel and Distributed Computing 59 (1999) 107–131.
[27] Taiwan unigrid project portal site.http://www.unigrid.org.tw.
[28] Network Weather Service (NWS), http://nws.cs.ucsb.edu/ewiki/.
[29] Globus Toolkit v4, http://www.globus.org/toolkit/downloads/4.0.4/.
[30] V. Sarkar, Determining average program execution times and their variance, in: Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation, 1989, pp. 298–312.
[31] C.Y. Park, Predicting program execution times by analyzing static and dynamic program paths, Real-Time Systems 5 (1) (1993) 31–62.
[32] J. Engblom, A. Ermedahl, Modeling complex flows for worst-case execution time analysis, in: Proceedings of the 21st IEEE Real-Time Systems Symposium, 2000, pp. 163–174.
[33] F. Stappert, P. Altenbernd, Complete worst-case execution time analysis of straight-line hard real-time programs, Journal of Systems Architecture 46 (4) (2000) 339–355.
[34] Yuanyuan Zhang, Wei Sun, Yasushi Inoguchi, Predict task running time in grid environments based on CPU load predictions, Future Generation Computer Systems 24 (6) (2008) 489–497.
[35] The globus alliance, http://www.globus.org/.
[36] Academia sinica, http://www.sinica.edu.tw/.
[37] National Center for High Performance Computing, http://www.nchc.org.tw/.
[38] Hsing Kuo University of Management (HKU), http://www.hku.edu.tw.
[39] National Dong Hwa University (NDHU), http://www.ndhu.edu.tw.
[40] Daniel P. Bovet, Marco Cesati, Understanding the Linux Kernel, O'reilly Media, Oct. 2000.
[41] R.L. Henderson, Job scheduling under the portable batch system, in: Lecture Notes in Computer Science, vol. 949, 1995, pp. 279–294.
[42] Load sharing facility http://www.platform.com/Products/platform-lsf.
[43] GLPK, http://www.gnu.org/software/glpk/.
[44] Robert G. Bland, Donald Goldfarb, Michael J. Todd, Ellipsoid method, a survey, Operations Research 29 (6) (1981) 1039–1091.