GRID LOAD BALANCING USING ENHANCED ANT COLONY OPTIMIZATION

Ku Ruhana Ku-Mahamud[1], Husna Jamal Abdul Nasir[2] and Aniza Mohamed Din[3]

[1]*Universiti Utara Malaysia, Malaysia,*
[2]*Universiti Utara Malaysia, Malaysia,*
[3]*Universiti Utara Malaysia, Malaysia,*

ABSTRACT. This study presents a new algorithm based on ant colony optimization for load balancing management in grid computing. The concentration is on improving the way ants search the best resources in terms of minimizing the processing time of each job and at the same time balancing the workload on available resources. An enhanced technique is proposed for the pheromone update activities. Single colony of ants is used for searching the best resources to process jobs. The credibility of the proposed algorithm was tested with other load balancing algorithm and results showed that the proposed algorithm was able to balance the load on the resources.

Keywords: grid load balancing, ant colony optimization, resource utilization

INTRODUCTION

Load balancing is one of the main issues in grid resource management besides resource discovery, resource scheduling, resource monitoring, resources inventories and resource provisioning (Sharma and Bawa, 2008). Scheduling algorithm is a main part of the load balancing algorithm. A good scheduling algorithm influences the balancing of each resource in grid computing system. Scheduling algorithms are classified into static and dynamic algorithms. In static scheduling, decisions are made at compile time while in dynamic scheduling, jobs are allocated at runtime and decisions are made by using the system-state information.

The job scheduling problem is defined as Nondeterministic Polynomial (NP)-complete problem (Ibarra and Kim, 1977; Yagoubi and Slimani, 2007) which means that there is no exact algorithm that can solve them in a polynomial time (Blum and Roli, 2003). The only way to solve these problems is to use approximate (heuristic) algorithms such as Genetic Algorithm (GA), Simulated Annealing (SA), Tabu Search and recently Ant Colony Optimization (ACO). ACO algorithm is used in grid computing because it is easily adapted to solve both static and dynamic combinatorial optimization problems. There are various types of ACO algorithm such as Ant Colony System (ACS), Max-Min Ant System (MMAS), Rank-Based Ant System (RAS) and Elitist Ant System (EAS) (Dorigo and Stützle, 2004).

The study to improve ant algorithm for dynamic job scheduling in grid computing which is based on the basic idea of ACS was proposed by Yan et al. (2005). The pheromone update function in this research was performed by adding encouragement, punishment coefficient and load balancing factor. The initial pheromone value of each resource was based on its status where job was assigned to the resource with the maximum pheromone value. The strength of pheromone of each resource was updated after completion of the job. The encouragement and punishment and local balancing factor coefficient were defined by users and were used to update pheromone values of resources. If a resource completed a job successfully, more pheromone was added by the encouragement coefficient in order to be selected for the next job execution. If a resource failed to complete a job, it was punished by

adding less pheromone value. The load of each resource was taken into account and the balancing factor was also applied to change the pheromone value of each resource.

Balanced job assignment based on ant algorithm for computing grids called BACO was proposed by Chang et al. (2007). The research aimed to minimize the computation time of job executing in Taiwan UniGrid environment which also focused on load balancing factors of each resource. By considering the resource status and the size of the given job, BACO algorithm chose optimal resources to process the submitted jobs. From the experimental results, BACO was capable of balancing the entire system load regardless of the size of the jobs in the static scheduling environment.

Load balancing in non-dedicated grids using ACO was proposed by Chen (2008). The proposed static algorithm was based on ACO algorithm in solving the load balancing problem in grid computing system. In the algorithm, the efficiency of the resources was maintained by immigrating jobs from overloaded resources to under loaded resources. Experimental results showed that the proposed algorithm performed better than the other algorithms in terms of makespan and resource usage. However, the proposed algorithm did not consider the requirement of each submitted jobs and the capacity of resources which might effect its performance.

In Moallem and Ludwig (2009), two distributed artificial life-inspired algorithms were introduced, which are ACO and PSO in solving the static grid load balancing problem. Distributed load balancing are categorized as a robust algorithm that can adapt to any topology changes in a network. In the proposed algorithm, an ant acted as a broker to find the best node in terms of the pheromone value stored in the pheromone table. The node with the lightest load was selected as the best node. The position of each node in the flock could be determined by its load in PSO. The particle compared the load of nodes with its neighbors and moved towards the best neighbor by sending assigned jobs to it. The proposed algorithm performed better than ACO for job scheduling where jobs were submitted from different sources and different time intervals. PSO showed better results than ACO in terms of the makespan. However, PSO used more bandwidth and communication compared to ACO.

ACO algorithm for dynamic load balancing in distributed systems through the use of multiple ant colonies was proposed by Ali et al. (2010). In this study, information on resources was dynamically updated for each ant movement. Multiple ant colonies were adopted such that each node sent a coloured colony throughout the network. Colored ant colonies were used to prevent ants of the same nest from following the same route and also force them to be distributed all over the nodes in the system. Each ant acted like a mobile agent which carried newly updated load balancing information to the next node. This approach did not consider resources capacity and jobs characteristics, thus matching of jobs with best resources a difficult task. Ant based algorithm was also introduced by Ku-Mahamud and Nasir (2010) for dynamic scheduling of jobs in grid system. The performance of the proposed ant based algorithm in terms of job computational time was found to be better than the ant based algorithm proposed by Moallem and Ludwiq (2009). In this study an enhanced ant based algorithm called EACO is proposed in balancing the load in the grid system. The enhancement will focus on the current status of the resources when scheduling new submitted jobs. This was not considered in the work by Ku-Mahamud and Nasir (2010).

THE PROPOSED FRAMEWORK

The proposed framework which is an enhancement of Ku-Mahamud and Husna (2010) consists of three mechanisms which are initial pheromone value mechanism, resource selection mechanism and pheromone updating mechanism that are used to organize the work of an ant colony (refer Figure 1).

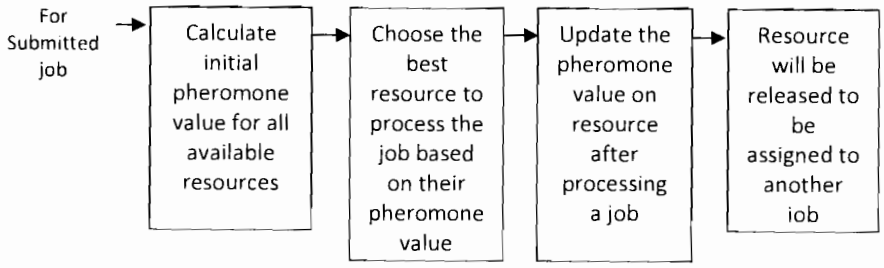

Figure 1. EACO framework

The initial pheromone value is calculated by considering the jobs characteristics and the capacity of resources. The resource selection mechanism is performed when ants try to search a new solution. However, this should be done under certain control to avoid exploring a very wide area of search space that might be far from the optimal solution. On the other hand, an exploitation of the search history is necessary to search the previously good solution. However, very strong exploitation is not required in order to prevent the stagnation problem of certain resources. The pheromone updating mechanism is also considered in this proposed framework. The global pheromone update mechanism is done after finishing processing a job. The best ant will deposit an amount of pheromone on its own nodes during the global pheromone update mechanism.

PROPOSED LOAD BALANCING ALGORITHM

The proposed EACO algorithm consists of 5 steps as below:

i. Obtain job requirements.
ii. Create an ant for the job.
iii. Calculate the initial pheromone value for all the resources.
iv. Assign resource with highest pheromone to the job.
v. Perform global pheromone update after complete processing the job.

In the first step, the scheduler will record details of the job size and CPU time needed by the job. Then an ant to represent a job in the grid system is created for every job that is submitted to the system. The task for the ant is to move from one resource to another with the aim to evaluate the best resource to be assigned to the job. Pheromone value on a resource indicates the capacity of each resource in grid system. In the third step, the initial pheromone value for 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:

$$PV_{rj} = \left[\frac{S_j}{bandwidth_r} + \frac{C_j}{MIPS_r * (1 - load_r)} \right]^{-1} \quad (1)$$

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 processor speed of resource r and $1 - load$ is the current load of resource r. It is assumed that the load, processor speed and bandwidth can be obtained from grid information server. Pheromone value will be stored in a table called pheromone Value

Table (PVT) as a reference to the other ants. The ant decides which resource to choose in its next step by looking at the PVT. The largest entry will be selected as the best resource to process the corresponding job.

In the final step of the algorithm, global pheromone update is performed to recalculate the all the values in PVT when a job is completely processed. After all ants have constructed a solution, the pheromone value is updated according to the following formula:

$$PV_{rj}(t+1) = (1-\rho).\tau_{rj} + 1/(\rho\Delta\tau_{rj}^{bs}) \quad (2)$$

where $\Delta\ \tau_{rj}^{best} = 1/L^{best}$ and ρ is the evaporation rate value that adaptively change with grid condition. The evaporation rate value is defined by:

$$\rho = \left[(R/J)*0.45^{n}*(J/R)\right] + 0.05 \quad (3)$$

where R is the number of resource, J is the number of job and n is defined by:

$$n = \sqrt[3]{(J/R)-1} \quad (4)$$

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. The effect of the global pheromone update is to make an already chosen resource less desirable for the following ant (Dorigo et al., 1991). So, the exploration of not yet visited resource is increased. Once the job is finished, the resource will be released to be used by other jobs.

EXPERIMENTAL RESULTS

The performance of EACO algorithm was compared with Antz algorithm (Moallem and Ludwiq, 2009) in terms of resource utilization. Standard deviation in workload distribution is often taken as the performance measure of a load balancing algorithm. A good load balancing scheme is indicated by a smaller standard deviation value. The utilization of each resource in grid system is dependent on the number of jobs which are assigned to the machine by the grid scheduler and the power of its processing elements. The total utilization percentages of all resources are supposed to be 100% for each experiment. The utilization of each resource can be calculated by

$$Utilization = \frac{total\ \ busy\ \ time}{total\ \ time\ \ process\sin g\ \ all\ \ jobs}$$

In the experiments, each resource is assumed to consist of one machine and each machine may have one or several processors known as processing elements (PEs) ranging between 1 and 5. The speed of processor or computational power is defined by the number of Cycles per Unit Time. As the processors in each machine can be heterogeneous, so, they may have different processing power and in this study, the processing power of each PE is 10 or 50 MIPS. The bandwidth is set to either 1000 or 5000 bits/sec. The length of each job is presented in MIPS and each job has different input and output size requirements as depicted in Table 1.

Table 1. Jobs Characteristics.

Length	0 – 50000 MI
File Size	100 + (10% to 40%)
Output Size	250 + (10% to 50%)

Both algorithms were tested with exactly the same scheduling parameters such as the number of jobs, number of resources, number of machines per resource, number of PEs per machine, PE ratings, bandwidth, size of jobs, and CPU time needed by each job.

The first experiment was conducted with 10 resources to process 500 jobs. Result of the experiment is shown in Figure 1. The standard deviation for EACO algorithm is 0.24781421 while the standard deviation for Antz algorithm is 3.6131422. The utilization of resources when EACO algorithm is applied does not varies as much as when Antz algorithm is used.

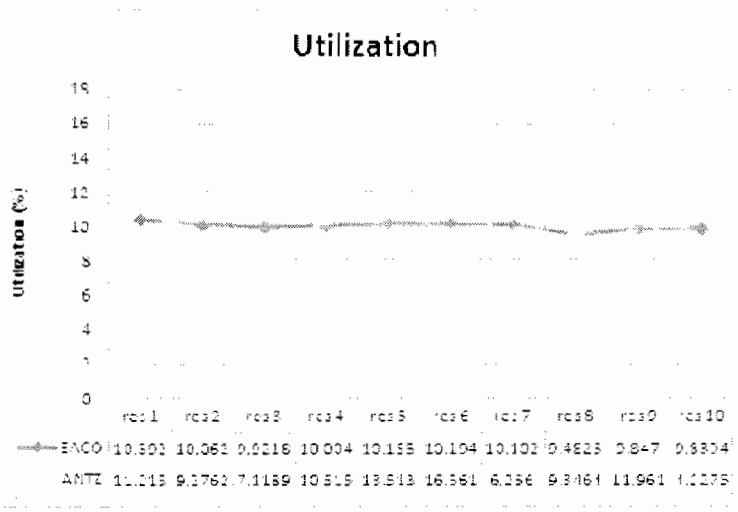

Figure 1. Utilization of 10 resources in processing 500 jobs by using EACO algorithm and Antz algorithm.

The second experiment was conducted to determine the utilization of each resource in processing 1000 jobs with 10 resources. Experimental results also showed that EACO algorithm was better than Antz algorithm as the standard deviation for EACO algorithm is 0.28084636 and the standard deviation for Antz algorithm is 2.77428296 (refer Figure 2). These results showed that the EACO algorithm successfully balanced the load among large number of resources.

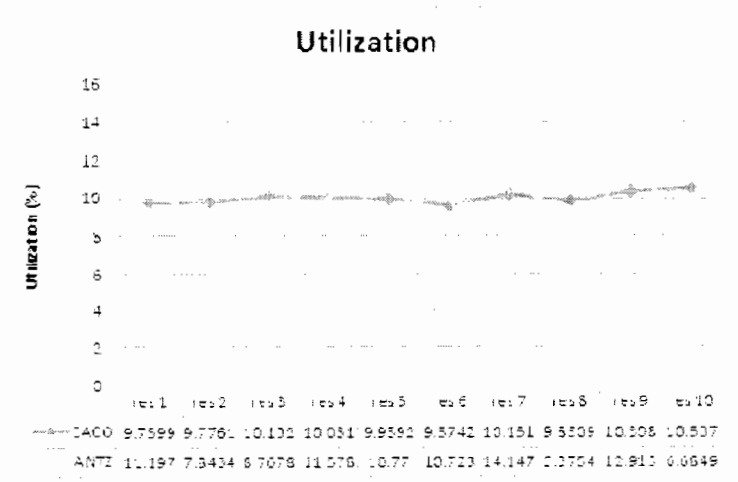

Figure 2. Utilization of 10 resources in processing 1000 jobs by using EACO algorithm and Antz algorithm.

The above experimental results showed that EACO performed better than Antz algorithm in terms of resource utilization. EACO algorithm has successfully scheduled the jobs among resource in all conditions which leads to a balanced load network.

CONCLUSION

The research output is a new member of the ACO family that offers the chance to enhance the performance in terms of utilization of available ACO algorithms on load balancing algorithms. The new proposed algorithm enhances the classical approach of ACO algorithm by minimizing the computational time of each job and dynamically schedule submitted jobs to suitable resources. At the same time, it balanced the entire resources. This is expected as the EACO algorithm keeps track of the state of all resources at each point in time which makes it able to make more optimal decisions each time when new jobs are submitted.

ACKNOWLEDGMENTS

The authors wish to express their gratitude to the Ministry of Higher Education for the financial support under the Fundamental Research Grant Scheme and to Universiti Utara Malaysia for facilitating the management of the research.

REFERENCES

Ali, A., Belal, M., A., & Al-Zoubi, M., B. (2010). Load balancing of distributed systems based on multiple ant colonies optimization. *American Journal of Applied Sciences, 7*(3), 433-438.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. Journal of *ACM Computing Surveys (CSUR), 35*(3), 268-308.

Chang, R., Chang, J., & Lin, P. (2007). Balanced job assignment based on ant algorithm for computing grids. *Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference*, 291-295.

Chen, Y. (2008). Load balancing in non-dedicated grids using ant colony optimization. *Proceedings of the 4th International Conference on Semantics, Knowledge and Grid,* 279-286.

Dorigo M., V. Maniezzo & A. Colorni (1991). The Ant System: An Autocatalytic Optimizing Process. *Technical Report No. 91-016 Revised*, Politecnico di Milano, Italy.

Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization.* Cambridge, Massachusetts, London, England: MIT Press.

Ibarra, O., & Kim, C. (1977). Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the Association for Computing Machinery, 24*(2), 280-289.

Ku-Mahamud, K.R. & Nasir, H.J.A. (2010). Ant colony algorithm for job scheduling in grid computing. *Proceeding of The Asia Modelling Symposium,* 26-28 May 2010, Kota Kinabalu, Malaysia, 40-45.

Moallem, A., & Ludwig, S. (2009). Using artificial life techniques for distributed grid job scheduling. *Proceedings of the 2009 ACM Symposium on Applied Computing,* 1091-1097.

Sharma, A., & Bawa, S. (2008). Comparative analysis of resource discovery approaches in grid computing. *Journal of Computers, 3*(5), 60-64.

Yagoubi, B., & Slimani, Y. (2007). Task load balancing strategy for grid computing. *Journal of Computer Science, 3*(3), 186-194.

Yan, H., Shen, X., Li, X., & Wu, M. (2005). An improved ant algorithm for job scheduling in grid computing. *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, 5,* 2957-2961.