Second International Conference on Computer and Network Technology

Grid Load Balancing Using Ant Colony Optimization

Husna Jamal Abdul Nasir
College of Arts and Sciences,
Universiti Utara Malaysia,
06010 Sintok, Kedah, Malaysia.
E-mail: oxalic2131@gmail.com

Ku Ruhana Ku-Mahamud
College of Arts and Sciences,
Universiti Utara Malaysia,
06010 Sintok, Kedah, Malaysia.
E-mail: ruhana@uum.edu.my

***Abstract* –An enhanced ant colony optimization technique for jobs and resources scheduling in grid computing is proposed in this paper. The proposed technique combines the techniques from Ant Colony System and Max – Min Ant System and focused on local pheromone trail update and trail limit. The agent concept is also integrated in this proposed technique for the purpose of updating the grid resource table. This facilitates in scheduling jobs to available resources efficiently which will enable jobs to be processed in minimum time and also balance all the resource in grid system.**

Keywords - ***Enhanced Ant Colony Optimization, Grid Resource Management, Load Balancing, Stagnation, System Architecture***

I. INTRODUCTION

Cluster and grid computing are several ways for establishing distributed system [5]. In a cluster computing environment, several personal computers or workstation are combined through local networks in order to develop distributed applications. Cluster computing give rise to the application being inflexible in variation because they are limited to a fixed area. Grid computing is developed through a combination of various resources from different geographic locations.

Grid computing is based on large-scale resources sharing in a widely connected network such as the Internet [6]. This makes grid computing different from conventional distributed computing and cluster computing. However, computational grid has different constraints and requirements to those of traditional high performance computing systems. In grid system, resource management and scheduling are key grid services, where issues of task allocation and load balancing represent a common problem for most grid systems. In grid computing environment, load balancing algorithm should be 'fair' in distributing jobs across the resources. The objectives of the load balancing algorithm are to equally spread the job on each resource, minimizing the total task execution time of each job and maximizing the utilization of each resource. In order to achieve these objectives, the difference between the heaviest-loaded node and the lightest node should be minimized.

Load balancing algorithm can be classified into static or dynamic, and centralized and decentralized. In the static load balancing algorithm, all information about jobs, resources and communication network are known in advance and jobs are assigned to suitable resources before execution begin. Once started, they keep running on the same resource without interruption. However, static load balancing has one major disadvantage which is all information about jobs and resources are remaining constants during the process. In contrast, dynamic load balancing attempt to use the runtime state information to make a load balancing decision more informative. Reevaluation is allowed of already taken assignment decisions during job execution in dynamic load balancing algorithm [8]. In these comparisons, static load balancing algorithm is easier to be implemented and has minimal runtime overhead compare to dynamic load balancing algorithm. However, dynamic load balancing may result in better performance.

In the centralized approach, one node in the grid system acts as a scheduler and makes load balancing decisions for all resources. All information from other nodes will be sent to this node. However, in the decentralized approach, all nodes in the grid system are involved in the load balancing decision. This is very costly and difficult to obtain and maintain the dynamic state information of the whole system. In decentralized approach, only partial local information is determined to make sub-optimal decisions.

This paper proposes an enhance ACO technique that can balance the entire resources in grid computing environment and at the same time minimize the computational time of jobs. Section 2 describes the grid load balancing algorithms while ACO algorithms in grid environment will be discussed in section 3. The proposed technique will be discussed in section 4 and the experimental result is presented in Section 5. Lastly, concluding remarks are highlighted in Section 6.

II. GRID LOAD BALANCING

Grid load balancing is one of the most difficult problems that must be handled in grid computing system. Load balancing aims to distribute workload evenly across two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, maximize throughput, minimize response time, and avoid overload. The problem of balancing resources is defined as Nondeterministic Polynomial (NP)-complete problem [12]. Grid resource management is a very challenging task since the resources that need to be shared are distributed and heterogeneous [2]. Resource management involves the

978-0-7695-4042-9/10 $26.00 © 2010 IEEE
DOI 10.1109/ICCNT.2010.10

process of scheduling jobs to suitable resources as illustrated in Fig. 1. During the scheduling process, jobs are needed to be matched with suitable resources that can fulfill their requirements as well as balancing loads on resources.

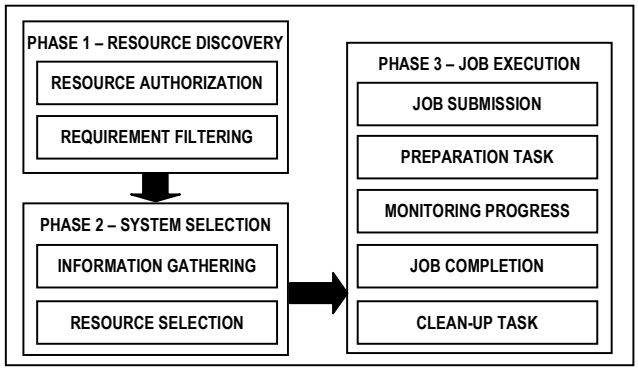

Figure 1: Grid Scheduling Process

There are many types of algorithms that have been used in resource balancing in grid computing system. The research by [1] addresses the use of Genetic Algorithm (GA) and Tabu Search (TS) to solve the grid load balancing problem in the dynamic environment. In the study, GA and TS performed better than the Best-Fit, Random, Min-min, Max-min and Sufferage algorithms in terms of time taken to schedule submitted jobs and job completion time. GA and TS can balance the extra overhead by considering the ever-decreasing costs of storage and processing power. However, these algorithms require extra storage and processing requirement at the scheduling nodes.

A hybrid load balancing policy that can be integrated in static and dynamic load balancing technique is studied in [3] with the objectives to allocate effective nodes, identify system imbalance immediately when a node become ineffective and fill in with new nodes. Result proved that this technique is more effective than the FCFS and GA in term of selecting nodes, reducing task completion time and avoiding the occurrence of task re-distribution and re-execution. Thus it can maintain load balancing on each resource and improve the performance of the system. Suitable resources will be allocated according to task properties which can reduce wrong selection of ineffective nodes decision.

The study by [12] uses a combination of intelligent agents and multi-agent approaches that work in grid load balancing area. In static grid load balancing, the iterative heuristic algorithm is better than the FCFS algorithm. The study highlights that a peer to peer service advertisement and discovery technique are more effective in dynamic grid load balancing environment. Instead of using a centralized control, distributed agent can reduce the network overhead significantly and allow the system to operate well in distributed environment which can help the user to achieve good resource utilization and minimizing the processing time of each job.

A hybrid load balancing strategy of sequential tasks that uses a combination of static and dynamic load balancing strategies which combines a FCFS algorithm with a special designed GA was proposed in [2]. The FCFS algorithm can make decision instantly which can reduce the system response time, resulting in a shorter makespan. GA was used to control the overall performance over a list of tasks and targets the balance of the resources in grid computing area. A sliding-window technique is used to trigger the switch between the two algorithms and to make a rapid task assignment as well. From the experiment conducted, hybrid GA provides better performance than dynamic GA and FCFS in different conditions such as makespan and the current work load. Other technique such as Ant Colony Optimization has also been used in solving the load balancing problem [14].

III. ACO Algorithms In Grid Environment

Ant Colony Optimization (ACO) is inspired by a colony of ants that work together to find the shortest path between their nest and food source. Every ant will deposit a chemical substance called pheromone on the ground after they move from the nest to food sources and vice versa. Optimal path will be chosen based on the pheromone value. The path with high pheromone value is shorter than the path with low pheromone value. This behavior is the basis for a cooperative communication. 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) [9].

ACO has been applied in solving many problems in scheduling such as Job Shop Problem, Open Shop Problem, Permutation Flow Shop Problem, Single Machine Total Tardiness Problem, Single Machine Total Weighted Tardiness Problem, Resource Constraints Project Scheduling Problem, Group Shop Problem and Single Machine Total Tardiness Problem with Sequence Dependent Setup Times [9]. A recent approach of ACO researches in the use of ACO for scheduling job in grid computing [17]. ACO algorithm is used in grid computing because it is easily adapted to solve both static and dynamic combinatorial optimization problems. In a research by [21], ACO has been used as an effective algorithm in solving the scheduling problem in grid computing.

Balanced job assignment based on ant algorithm for computing grids called BACO was later proposed by [14]. This research aims 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 chooses optimal resources to process the submitted jobs. The local and global pheromone update technique is used to balance the system load. Local pheromone update function updates the status of the selected resource after job has been assigned and the job scheduler depends on the newest information of the selected resource for the next job submission. Global pheromone update function updates the status of each resource for all jobs after the completion of the jobs. By using these two update techniques, the job scheduler will get the newest information of all resources for the next job submission.

From the experimental result, BACO is capable of balancing the entire system load regardless of the size of the jobs.

The study to improved ant algorithm for job scheduling in grid computing which is based on the basic idea of ACO was proposed in [4]. The pheromone update function in this research is performed by adding encouragement, punishment coefficient and load balancing factor. The initial pheromone value of each resource is based on its status where job is assigned to the resource with the maximum pheromone value. The strength of pheromone of each resource will be updated after completion of the job. The encouragement and punishment and local balancing factor coefficient are defined by users and are used to update pheromone values of resources. If a resource completed a job successfully, more pheromone will be added by the encouragement coefficient in order to be selected for the next job execution. If a resource failed to complete a job, it will be punished by adding less pheromone value. The load of each resource is taken into account and the balancing factor is also applied to change the pheromone value of each resource.

An ant colony optimization for dynamic job scheduling in grid environment was proposed by [17] which aimed to minimize the total job tardiness time. The process to update the pheromone value on each resource is based on local update and global update rules as in ACS. In the study, ACO algorithm performed the best when compared to First Come First Serve, Minimal Tardiness Earliest Due Date and Minimal Tardiness Earliest Release Date techniques.

The study by [21] proposed a bio-inspired adaptive job scheduling mechanism in grid computing with the aim to minimize the execution time of the computational jobs by effectively taking advantage of the large amount of distributed resource. Various software ant agents were designed with simple functionalities. Comparison was also performed between the bio inspired adaptive scheduling with the random mechanism and heuristic mechanism. Experimental results showed that bio-inspired adaptive job scheduling has good adaptability and robustness in a dynamic computational grid.

Simple grid simulation architecture for resource management and task scheduling was proposed in [22]. The study validated the scalability of ant algorithm where the ant algorithm for grid task scheduling is integrated into the simulation architecture and good results were obtained in terms of resource average utilization, response time and task fulfill proportion.

From the above research, ACS is the most popular variant of ACO that has been successfully used in grid computing to solve the scheduling and load balancing problems. However, the algorithm in this application domain can be enhanced to provide better performance for load balancing.

IV. PROPOSED ACO FOR GRID LOAD BALANCING

This proposed technique will focused on reducing the computational time of each job and at the same time to balance the entire resources available in grid environment. The technique will select the resources based on the pheromone value on each resource. A matrix that contains the pheromone value on each resource will be used in this technique to facilitate the selection of suitable resources to process submitted jobs. The technique utilized four main components namely the grid information server, grid resource broker, jobs and resources, and works as follow:

1) User will send request to process a job. Details about the job such as the total number of jobs, size of each job, and CPU time needed by jobs will be included in the request.
2) Grid resource broker starts to calculate the relevant parameter to schedule the job after receiving the message from the user. The information server also provides the resource information to grid resource broker.
3) The largest entry in the pheromone value (PV) matrix will be selected by proposed technique as the resource to process the submitted job. A local pheromone update is performed after a job is assigned to a resource.
4) A global pheromone update is performed after a resource completed processing a job.
5) The execution results will be sent to the user.

In this proposed technique, an ant represents a job in the grid system. The grid resource broker will find available resources from grid information server. Ant will move randomly in grid system and check the status of each resource. Pheromone value on a resource indicates the capacity of each resource in grid system. Pheromone value will be determined by two types of pheromone update technique which are local pheromone update in ACS [7] and global pheromone update in MMAS [18].

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:

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

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 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:

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

The largest entry from PV matrix 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 only 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,$0< \xi<1$ 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 make an already chosen resource less desirable for a following ant [9]. So, the exploration of not yet visited resource is increased.

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:

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

where $\Delta\ \tau_{jr}^{best} = 1/L^{best}$. 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 $0<p_{min}\leq p_{ij}\leq p_{max}\leq 1$. 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.

V. EXPERIMENTAL RESULT

In the experiment, there are three jobs (j_1, j_2, and j_3) that need to be processed and also three resources (r_1, r_2, and r_3) are available in 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. INITIAL STATUS OF EACH RESOURCE

Status	r_1	r_2	r_3
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 are shown in the following PV matrix:

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

The maximum pheromone value (PV_{23}) in the PV matrix will be selected by grid resource broker. So j_3 will be processed by r_2. After assigning j_3 to r_2, the local pheromone update is performed to the second row of r_2. Column 3 is no longer needed because j_3 has been assigned. The new PV matrix is as follows:

Local update

$$PV = \begin{bmatrix} PV_{11}=2.01 & PV_{12}=3.35 \\ PV_{21}=4.17 & PV_{22}=6.94 \\ PV_{31}=2.34 & PV_{32}=3.89 \end{bmatrix}$$

After r_2 finished processing j_3, 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 j_3 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 ρ value is used in evaporation process.

TABLE II. NEWEST STATUS OF EACH RESOURCE

Status	r_1	r_2	r_3
Processor Speed (MIPS)	217	464	195
Load	15%	25%	20%
Bandwidth (Megabits/s)	8.67	15.87	10.26
ρ	0.00	0.05	0.00

The ρ value of r_2 is 0.05 and ρ values for r_1 and r_3 are zero since they have not been assigned any job for execution. The new PV matrix is as follows:

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

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.

VI. CONCLUSION

The load balancing process in this proposed technique is based on the combination of local pheromone update and trail limits. The local pheromone trail update will reduce the amount of pheromone in assigned resource, to ensure the resource is less desirable for other ants while the trail limit, which is the allowed range of the pheromone strength, is limited to maximum and minimum trail strength. This is a technique to control the value of pheromone updated on each resource. The proposed technique is simple to be implementing due to the existing of information of each resources and jobs. By implementing this technique, the load on each resource can be balanced and the execution time of each job can be minimized.

REFERENCES

[1] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," presented at Proceedings of the First European Conference on Artificial Life, Paris, France, Amsterdam: Elsevier Publishing, pp. 134-142, 1991.

[2] G. Pavani and H. Waldman, "Grid resource management by means of ant colony optimization," 3rd International Conference on Broadband Communications, Networks and Systems (BROADNETS), 2006.

[3] H. Singh and A. Youssef, "Mapping and scheduling heterogeneous task graphs using genetic algorithms," presented at 5th IEEE Heterogeneous Computing Workshop (HCW'96), 1996.

[4] H. Yan, X. Shen, X. Li, and M. Wu, "An improved ant algorithm for job scheduling in grid computing," presented at Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, vol. 5, pp. 2957-2961, 2005.

[5] I. Foster and C. Kesselman, *The grid: blueprint for a new computing infrastructure*. San Francisco: Morgan Kaufmann, 2004.

[6] K. Yang, X. Guo, A. Galis, B. Yang, and D. Liu, "Towards efficient resource on-demand in grid computing," *ACM SIGOPS Operating Systems Review*, vol. 37(2), pp. 37-43, 2003.

[7] L. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," presented at Proceedings of the IEEE Conference on Evolutionary Computation, Nagoya, Japan (ICEC96), pp. 622-627, 1996.

[8] M. Chtepen, "Dynamic scheduling in grids system," Sixth Firw PhD Symposium, Faculty of Engineering, Ghent University, pp. 110, 2005.

[9] M. Dorigo and T. Stützle, *Ant colony optimization, Cambridge, Massachusetts, London, England*: MIT Press, 2004.

[10] M. Dorigo, "Optimization, learning and natural algorithms," *Ph. D. dissertation, Politecnico di Milano, Milan, Italy, 1992*.

[11] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: An autocatalytic optimizing process," *Thechnical Report 91-016 Revised, Dipartimento di Electronica, Politecno di Milano, 1991*.

[12] O. Ibarra and C. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *Journal of the ACM (JACM)*, vol. 24(2), pp. 280-289, 1977.

[13] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," presented at 7th IEEE Heterogeneous Computing Workshop (HCW'98), 1998.

[14] R. Chang, J. Chang, and P. Lin, "Balanced Job Assignment Based on Ant Algorithm for Grid Computing," presented at Proceedings of the 2nd IEEE Asia-Pacific Service Computing Conference, pp. 291-295, 2007.

[15] R. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, and J. Lima, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet," presented at 7th IEEE Heterogeneous Computing Workshop (HCW'98), 1998.

[16] S. Fidanova and M. Durchova, "Ant algorithm for grid scheduling problem," *Lecture Notes in Computer Science*, vol. 3743, pp. 405-412, 2006.

[17] S. Lorpunmanee, M. Sap, A. Abdullah, and C. Chompoo-inwai, "An ant colony optimization for dynamic job scheduling in grid environment," *International Journal of Computer and Information Science and Engineering*, vol. 1(4), pp. 207-214, 2007.

[18] T. Stutzle and H. Hoos, "MAX-MIN ant system," *Future Generation Computer Systems*, vol. 16, pp. 889-914, 2000.

[19] V. Naik, P. Garbacki, K. Kummamuru, and Y. Zhao, "On-line evolutionary resource matching for job scheduling in heterogeneous grid environments," *Proceedings of the 12th International Conference on Parallel and Distributed Systems(ICPADS'06), 2006*.

[20] X. Bai, H. Yu, Y. Ji, and D. Marinescu, "Resource matching and a matchmaking service for an intelligent grid," *International Journal of Computational Intelligence*, vol. 1(3), pp. 197-205, 2004.

[21] Y. Li, "A Bio-inspired Adaptive Job Scheduling Mechanism on a Computational Grid," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 6(3), pp. 1-7, 2006.

[22] Z. Xu, X. Hou, and J. Sun, "Ant algorithm-based task scheduling in grid computing," presented at Electrical and Computer Engineering IEEE CCECE, Canadian Conference, pp. 1107-1110, 2003.