Algorithm is Based on Ant Colony Optimization using Grid Simulaor

R. Tharani
Assistant Professor
Department of Computer Science and Engineering
JCT college of Engineering and Technology, Coimbatore, Tamil Nadu, India

Abstract

In today's competitive environment the objectives and goals of the producers (also called resource owners) and consumers (also called end users) are different. Computational grid has been considered as the best paradigm for handling large scale distributed system having geographically allocated resources. Load balancing algorithms are important in the research of network applications. In this paper we present an algorithm which reduces the average execution time and cost of the tasks. This method considers both cost and time constraints. The proposed algorithm is implemented with Gridsim toolkit which can simulate a decentralized module. The GridSim toolkit abstracts the features and behaviour of complex fundamental grid elements such as grid tasks, grid resources and grid users. This algorithm provides services like resource discovery. For evaluation purpose a comparison of execution times and cost of proposed algorithm and the other similar algorithm is also provided in this paper. Results support the proposed approach.

Keyword- Grid Computing, Load Balancing, Resource management, Execution time, Execution Cost, Ant Colony Algorithm, Random Algorithm

I. INTRODUCTION

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 [1]. 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 [2]. Unlike scheduling problems in conventional distributed systems, scheduling problem in grid system is much more complex as new features of grid systems such as its dynamic nature and the high degree of heterogeneity of jobs and resources must be undertaken [3]. Scheduling is mainly classified into two types, static and dynamic scheduling. Static scheduling allocates the task to suitable resource before starting the execution. In case of static scheduler all the details of tasks should be known well before starting the process. Dynamic scheduling allocates the resource during the execution time i.e. the scheduler can take decision during job execution [4]. Online mode and batch modes are available in dynamic scheduling.

In online mode the scheduler will be always in ready state, so when a job arrives, it allocates the resource immediately. In batch mode the jobs are grouped as set of tasks, known as Meta task and the mapping is done in a prescheduled time. Scheduling can aim to provide- two objectives namely high performance computing and high throughput computing. High performance computing decreases the execution time whereas high throughput computing increases the Processing capacity of the system.

Scheduler has the following three main phases [5]. In phase one all the available resource will be collected. This is known as resource discovery. Second phase collects the information about the resources and choose the best suitable resource to the task. Third phase executes the job in the selected machine.

By harmonizing and distributing the grid resources efficiently, an advanced resource allocation strategy can reduce total run time and total expenses greatly and bring an optimal performance [6] [7].

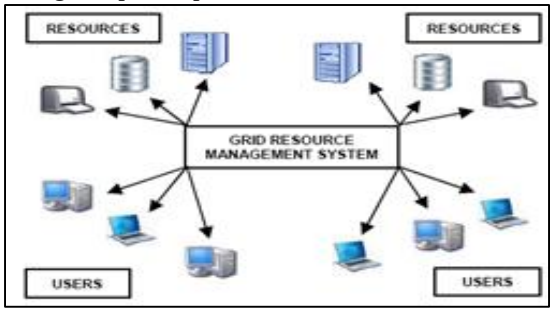

Fig. 1: Grid Computing Environment

All rights reserved by *www.grdjournals.com*

Resource scheduling is an NP-Hard problem [8]. Heuristic approaches help to solve such type of problems effectively. Opportunistic Load Balancing (OLB), Min-Min, Fast Greedy, Tabu-Search, Max-Min, Minimum Execution Time (MET) and Minimum Completion Time (MCT) Ant Colony Optimization are some of the heuristic approaches which help to solve the grid scheduling problem. The proposed algorithm is based on Ant Colony Optimization (ACO) algorithm which uses batch mode heuristic mapping. In this ant colony algorithm each job is considered as an ant and optimal solution is provided with the help of pheromone detail. Heuristic based ant colony algorithm is used in the second phase of the scheduler.

In the TIME_SHARED when jobs arrive, the systems start their execution immediately and share resources among all jobs. Whenever a new Gridlet job arrives, we update the processing time of existing Gridlet and then add this newly arrived job to the execution set. We schedule an internal event to be delivered at the earliest completion time of smallest job in the execution set. It then waits for the arrival of events. It can be considered as Round-Robin allocation policy.

The organization of the paper further is as follows. The motivation works are discussed in Section II, Literature review is analyzed in Section III, Section IV describe the problem ,section V deals with proposed algorithm, Implementation and experimental results are discussed in Section VI and conclusion with future work is presented in Section VII.

II. MOTIVATION

The Last decade has seen a substantial change in the way we perceive and use computing resources. Even through the computational capability and network performance has gone to a great extent but these computing networks are still not fully capable to deal with current complex and resource intensive problems [9]. With the evolution of grid computing many issues has been raised viz. designing, building, deploying and scheduling of various heterogeneous resources in grid environment [10]. To achieve the promising potentials of tremendous distributed resources in grid environment effective and efficient resource scheduling and load balancing algorithms are fundamentally important. Unfortunately scheduling algorithms in traditional parallel and distributed systems, which usually run on homogeneous and dedicated resources can't work well in grid environment. The motivation of the paper can be summarized as:

1) To aggregate the power of widely distributed resources and provide the non-trivial services to users.
2) Use of the largely unused computing resources.
3) Proper utilization of greatly increased in the CPU speed in the recent years.
4) To improve the efficiency and usability of networks in grid computing environment with high Communication demands.
5) To spread the load equally among the available resources, so as to reduce the waiting time of the tasks and increase the throughput of the resources.
6) To utilize widespread availability of fast, universal network connection.

III. LITERATURE REVIEW

Thorough research has been done in the area of resource scheduling and load balancing in grid computing and it is found that scheduling problems are NP-hard in nature, which can be better solved by the heuristic technique. Many heuristic algorithms have been designed for scheduling of task in grid environment. But all these algorithms have some drawbacks and still there is a scope to optimize the scheduling process, so as to maximum utilize the resources and increasing the throughput of the resources and the waiting time of the tasks.

Some of the popular heuristic resources scheduling algorithms are [11]:

A. Opportunistic load balancing (OLB)

Without considering the job's execution time, it assigns a job to the earliest free machine. If more than one machine is free then it assigns the job in arbitrary order to the processor. This scheduling mechanism runs faster. The advantage of this method is that it keeps almost all machines busy. Yet it does not assure load balance.

B. Minimum Execution time (MET)

The minimum execution time or MET assigns each job to the machine that has the minimum expected execution time. It does not consider the availability of the machine and the current load of the machine. The resources in grid system have different computing power. Allocating all the smallest tasks to the same fastest resource redundantly creates an imbalance condition among machines. Hence solution is static.

C. Minimum Completion Time (MCT)

The algorithm calculates the completion time for a job on all machines by adding the machine's availability time and the expected execution time of the job on the machine. The machine with the minimum completion time for the job is selected. The MCT considers only one job at a time. This causes that particular machine may have the best expected execution time for any other job. The drawback of MCT is that it takes long time to calculate minimum completion time for a job

All rights reserved by *www.grdjournals.com*

D. Max-min

Max-min begins with a set of all unmapped tasks. The completion time for each job on each machine is calculated. The machine that has the minimum completion time for each job is selected. From the set, the algorithm maps the job with the overall maximum completion time to the machine. Again the above process is repeated with the remaining unmapped tasks. Similar to Min-min, Max min also considers all unmapped tasks at a time.

E. Min-Min

Min-min algorithm starts with a set of all unmapped tasks. The completion time for each job on each machine is calculated. The machine that has the minimum completion time for each job is selected. Then the job with the overall minimum completion time is selected and mapped to the machine. Again, this process is repeated with the remaining unmapped tasks. Compared to MCT, Min-Min considers all unmapped tasks at a time. The drawback of Min-Min is that, too many jobs are assigned to a single node .This leads to overloading and response time of the job is not assured.

F. Ant Colony Optimization Algorithm

Ant colony optimization (ACO)[12] was first introduced by Marco Dorigo as his Ph.D. thesis and was used to solve the TSP problem .ACO was inspired by ants behavior in finding the shortest path between their nests to food source. Many varieties of ants deposit a chemical pheromone trail as they move about their environment, and they are also able to detect and follow pheromone trails that they may encounter. With time, as the amount of pheromone in the shortest path between the nest and food source increases, the number of ants attracted to the shortest path also increases. This cycle continues until most of the ants choose the shortest path. Various types of ACO algorithm are presented. Each of them has some special properties, i.e., Ant Colony System (ACS), Max-Min Ant System (MMAS). Fast Ant System (FANTS). In paper [13], they have proposed a simple grid simulation architecture using ACO. They have used response time and average utilization of resources as the evaluation index. In the paper [14] and [15], they have proposed ACO algorithms, which could improve the performance like job finishing ratio. In paper [16], the job is moved from one machine to another machine, so that the traffic in the grid system will be automatically increased. In paper [17], six different ant agents are used. To solve the grid scheduling problem, ACO is one of the best algorithms.

IV. PROBLEM DESCRIPTION

Scheduling a task to a particular resource in such a way so as to improve the execution time and cost is the foremost problem in the grid computing environment. Heterogeneous and dynamic nature of the resources makes it a challenging assignment. For mapping a task to a resource there is a grid scheduler that receives the applications from the grid users, selects the feasible resources for the applications according to acquired information from the grid information service module and submits the resource to the selected resource. The grid scheduler does not have control over the resources and also on the submitted jobs. Any machine can execute the any job, but the execution time differs. As compared with the expected execution time, the actual time may be varied at the time of running the jobs to allocated resource. The grid scheduler's aim is to allocate the jobs to the available nodes. The best match must be found from the list of available jobs to the list of available resources. The selection is based on the predictions of the computing power of the resource. The grid scheduler must allocate the job to the resources efficiently. The efficiency depends on two parameters one is execution time of the job and second is cost of using the resource. To minimize these parameters the workload has to be evenly distributed over all nodes which are based on their processing capabilities. This arises the new issue called load balancing. The main objective of a load balancing consists primarily to optimize the average response time of applications by equal distribution of the load among available resources. So we can the state the problem as "Map the best resource to a task so that overall execution time and cost of the resource can be minimized."

V. PROPOSED ALGORITHM

Proposed algorithm is based on Ant colony optimization. Ant algorithm is inspired on an analogy with real life behavior of a colony of ants when looking for food, and is effective algorithm for the solution of many combinatorial optimization problems.
In proposed algorithm the pheromone is associated with resources rather than path. The increase or decrease of pheromone depends on task status at resources. The main objective of algorithm is reduction in total cost and execution time

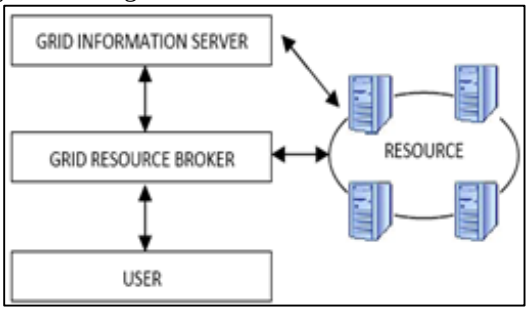

Fig. 2: System Architecture

All rights reserved by *www.grdjournals.com*

A. Algorithm_ACO_Scheduling

Let the number of tasks (ants) in task set T maintained by task agent is A and the number of registered resources (pheromone) is Q.

Step 1: [Initialization]

For every new resource R_i registered to Grid Information Server, the pheromone value is initialized as:

Ip (0) = (N + M) * Communication speed Where Ip (0) is the initial pheromone value.

N is the number of processing elements.

M is the MIPS rating of processing element.

Step 2:

Repeat 3 to 6 While ($T \neq \Phi$)

Step 3:

Select task t from task set T.

Step 4:

Determine the resource R_j for task t having higher transitions probability P (high pheromone intensity) Such as:

$$P_j(t) = \text{Max} \frac{\overline{[Cp_j(t)]^a} \; \overline{+ [Ip_j]^b}}{(\sum_r [Cp_j(t)]^a + [Ip_j]^b)/rc}$$

Where j, r are available resources. rc is cost of using resource.

Cp (t) denotes the current pheromone of resource Rj.

Ip_j represents the initial pheromone of R_j

i.e. $Ip_j = Ip_j(0)$.

a is the parameter on relative performance of current pheromone trail intensity.

b is the parameter on relative importance of initial performance attributes.

Step 5:

Schedule task t to Rj and remove it from T

i.e. T=T-{t}

Step 6: [Update Pheromone]

For every resource Rj assigned a task pheromone is $Cp^{new} = pd * Cp^{old} + \Delta$,

$\Delta j = -C$

If (Task_Status= Succesful_Complete)

$\Delta = \Phi * C$ [pheromone increases]

Where Φ is the encouragement argument. If (Task_Status= Failure)

[Pheromone decreases]

$\Delta = \Theta * C$, where Θ is the punishment argument.

Where pd = Pheromone decay parameter $0 < pd < 1$

1- pd pheromone permanence

C= Computational complexity of task

Δ_j= Pheromone variance

Step 7: Exit.

VI. IMPLEMENTATION WITH GRIDSIM

GridSim [18] toolkit is used to conduct the simulations based on the developed scheduling algorithm. Grid resources information and Grid user's information will be given as input to start the simulation. Then the simulation starts by resource creation followed by the creation of user tasks. The user tasks are created based on user specified parameters (total number of jobs, average MI of each job. Then, the system starts the GridSim simulation.

It first gathers the characteristics of the available Grid resources created in the resource creation section in the system. The scheduler maintains the pheromone value of each resource (from schedule center) and will compute the possibilities of the current resources available in the Grid every time when a job is to be scheduled. Then, the scheduler selects a resource based on the developed algorithm and schedules the job.

The simulation will be completed when all the user tasks get executed on the grid resources and get back the results.

A. Simulation Parameters

Following simulation parameters are used to conduct the simulation

Parameter	*Value*
Number of User	*1*
Number of Resources	*10-50*

All rights reserved by *www.grdjournals.com*

Number of PE per Resource	*4-16*
MIPS of PE	*300-900*
Resource Cost	*5-55 G$*
Total Number of Tasks(Ants)	*25*
Length of Task	*100-2500 MI*
Communication Speed	*25-150 IPS*
Allocation Policy	*TIME_SHARED*
Relative performance of current pheromone trail intensity (a)	*0.5*
Relative importance of initial performance attributes (b)	*0.5*
Pheromone decay parameter (pd)	*0.8*
Pheromone encouragement argument (Φ)	*1.1*
Pheromone punishment argument (Θ)	*0.8*

Table 1: Parameters for Simulation

VII. EXPERIMENTAL RESULTS

The proposed algorithm is compared with the random resource allocation algorithm already used in GridSim. This algorithm selects the next resource for task assignment in a random fashion.

Experiment 1

Execution cost of the proposed algorithm and random resource allocation algorithm is compared with varying number of resources (10-50) and constant number of the task (25).

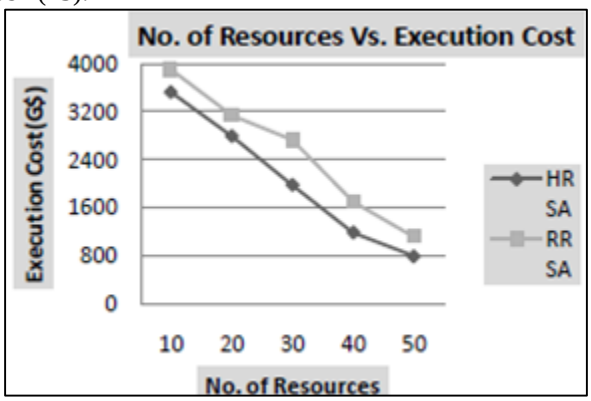

Fig. 3: No. of resources Vs. Execution Cost

Execution time of the proposed algorithm and random resource allocation algorithm is compared with varying number of resources (25) and constant number of the tasks (25)

Fig. 4: No. Of resources vs. Execution Cost

For selected set of parameters the proposed algorithm shows the improvement in comparisons of random scheduling of a task to a resource. The growth of both the curves in both graphs is approximately same, which shows that there is uniform improvement in the total execution time and cost. The following table shows percentage of improvement in the Average execution time and cost.

Performance Comparison	*Average Execution Time (Sec.)*	*Average Execution Cost (G$)*
Proposed Algorithm	*2498.31*	*131.4*
Random Resource Selection Algorithm	*4299.48*	*186.49*
Improvement %	*72.09*	*41.92*

Table 2: Performance Comparison

All rights reserved by *www.grdjournals.com*

VIII. CONCLUSION AND FUTURE WORK

The proposed scheduling strategy results in increased performance in terms of low processing time and low processing cost if it is applied to a Grid application with a large number of coarse granularity tasks. This works effectively in minimizing both the processing time of the tasks as well as the processing cost of the tasks depending upon the value chosen by the grid user who submits the task. Future work would involve developing a more comprehensive distributive scheduling system that takes into account the hierarchy of clusters of resources. And also to reduce the transmission costs and delays in the applications with large number of light weight jobs, job grouping can be done before load balancing the jobs on to the resources.

REFERENCES

[1] L.M. Nithya, A.Shanmugam "Scheduling in Computational Grid with a new hybrid Ant Colony Optimization Algorithm", European Journal of Scientific Research Vol.62 No. 2 (2011) pp.273- 281.
[2] Salehi, M.A., Deldari, H., Dorri, B.M., 2008. "Balancing Load in a Computational Grid Applying Adaptive, Intelligent Colonies of Ants", Informatics, Vol. 32, pp.327-335.
[3] Foster, I., Kesselman, C., Tuecke, S., 2001. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of High Performance Computing Applications, Vol. 15(3), pp. 200-222.
[4] Chtepen, M., 2005. "Dynamic Scheduling in grid systems", Sixth Firw. PhD Symposium. Faculty of Engineering, Ghent University, 110, pp.1-2.
[5] Schopf, J.M., 2002. "A General Architecture for Scheduling on the Grid", Special issue of JPDC on Grid Computing.
[6] Chapman, C., Musolesi, M., Emmerich, W., Mascolo, C., 2007. "Predictive Resource Scheduling in Computational Grids", IEEE Parallel and Distributed Processing Symposium, 2007 (IPDPS 2007), pp. 1-10.
[7] Krauter, K., Buyya, R., Maheswaran, M. (2002), "A Taxonomy and survey of Grid Resource Management Systems for Distributed Computing", Software: Practice and Experience (SPE) Journal, Wiley Press, USA, Vol.32 (2), pp. 135-164.
[8] Baca, D.F., 1989. "Allocating Modules to rocessors in a Distributed System", IEEE Transactions on Software Engineering, pp.1427–1436.
[9] Kuppani Sathish, A Rama Mohan Reddy, "enhanced ant algorithm based load balanced task scheduling in grid computing." IJCSNS VOL.8 No.10, October 2008.
[10] David De Roure, Mark A. Baker, Nicholas R. Jennings and Nigel R. Shadbolt, Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK " The Evolution of the Grid".
[11] Kousalya.K and Balasubramanie.P," An Enhanced Ant Algorithm for Grid Scheduling Problem" IJCSNS VOL.8 No.4, April 2008.
[12] M. Dorigo and T. Stützle, Ant colony optimization, Cambridge, Massachusetts, London, England: MIT Press, 2004.
[13] Z. Xu, X. Hou and J. Sun, "Ant Algorithm-Based Task Scheduling in Grid Computing", Electrical and Computer Engineering, IEEE CCECE 2003, Canadian Conference, 2003.
[14] E. Lu, Z. Xu and J. Sun, "An Extendable Grid Simulation Environment Based on GridSim", Second International Workshop, GCC 2003, volume LNCS 3032, pages 205–208, 2004.
[15] .H. Yan, X. Shen, X. Li and M. Wu, "An Improved Ant Algorithm for Job Scheduling in Grid Computing", In Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, 18-21 August 2005.
[16] Li Liu, Yi Yang, Lian Li and Wanbin Shi, " Using Ant Optimization for super scheduling in Computational Grid, IEEE proceedings of the 2006 IEEE Asia-pacific Conference on Services Computing (APSCC' 06)
[17] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," in 7th IEEE Heterogeneous Computing Workshop, pp. 79–87, Mar. 1998.
[18] Rajkumar Buyya, and Manzur Murshed, GridSim: A Toolkit for the Modeling, and Simulation of Distributed Resource Management, and Scheduling for Grid Computing, The Journal of Concurrency, and Computation: Practice, and Experience (CCPE), Volume 14, Issue 13-15, Pages: 1175-1220, Wiley Press, USA, November- December2002rocessors in a Distributed System", IEEETransactions on Software Engineering, pp.1427–1436.

All rights reserved by *www.grdjournals.com*