



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

www.iasir.net

A Resource Serviceability Based Heuristic-Aided Load Balancing Scheme

Sanjna Kalyan^{*1}, Chander Shekhar^{#2}

^{*} Department of Computer Engineering,

Guru Brahma Nand Ji Govt. Polytechnic, Nilokheri, India

[#] CSIR-CEERI, Pilani, Rajasthan, India

INDIA

Abstract: The computational grid has been considered as the best paradigm for handling large scale distributed system having geographically allocated resources; suiting the varied objectives and goals of producers and the consumers. Load balancing algorithms are important for efficient performance of network applications. In this paper a heuristic based load balancing algorithm has been presented which reduces the average execution time and cost of the tasks. The method considers both cost and time constraints. The proposed algorithm is implemented with GridSim toolkit which can simulate a decentralized module. The proposed algorithm provides resource discovery service too. For evaluation purpose a comparison of execution times and cost of proposed algorithm with a non-heuristic based algorithm is also provided. Simulation results support the proposed algorithm over non-heuristics based algorithm.

Keywords: Grid, Load Balancing, Heuristic Load Balancing Algorithm, GridSim, Computation Time.

I. Introduction

The Grid computing was introduced by Foster and Kesselman in late 90s. The duo explained it a way to utilize the geographically distributed available computation power to remote grid users for the execution of their computation requiring jobs or tasks or processes [15]. It involves the runtime aggregation of these resources in the form of virtual organization [3, 8] with lower latency constraint, according to the need of the jobs submitted by the grid users. Earlier user has to interact with a resource broker middleware that abstracts the complexities of Grid computing [4, 5]. The broker middleware discover & manage resources that a user can access, negotiates access cost and maps task to resources (scheduling). The middleware also associates the staging the application and data for processing (deployment), starts job execution, and finally gathers the results. The monitoring and tracking of application execution progress along with adapting to the changes in Grid runtime environment conditions and resource failures are also managed by grid middleware.

The under-utilization or over-utilization of resources in grid environment has been seen very often. So the balanced & optimized usage of grid resources is an area of concern. The load-balancing incorporates the inter-processor communication overhead regarding quantification of waiting jobs, task arrival rate, computing rate of processor etc. for improving the overall performance of the grid. Based on the information that a grid can use to balance the load, the algorithms are classified as 1) static; 2) dynamic or adaptive [7, 10]. Prior information about the computing devices and the communication network is at hand in static algorithms, and hence the systems take deterministic or probabilistic decisive outlook for balancing the load during compile time. Alternatively the dynamic load balancing algorithm uses runtime state information to make informative decision in sharing the system load. The runtime decision making attribute makes it robust and flexible for modern grids.

Further, dynamic algorithms have been classified on the basis of functional behavioral aspects such as *Centralized algorithm* is centric where the concerned parameters for making load balancing decision are collected, and used by a single device; *decentralized* is scalable and fault tolerant approach where all the devices are involved in decision making for load balancing; *Adaptive or Non-adaptive* algorithm that can respond to varying parameters and can change load-balancing decision while being on the run; otherwise, it is non-adaptive.

In this paper a new load balancing algorithm has been proposed which is heuristic in nature. Unlike previous algorithms which considered the system load of grid nodes or the completion time for a job but do not take into account job personal resource requirements (such as cost, QOS of a node). The presented algorithm considers the job size and mainly focuses on formulating a decentralized heuristic based load balancing for resource scheduling. The rest of the paper is organized as follows: Section II gives a detailed Literature review. Section III explains the system model of the Grid. Section IV will describe the proposed algorithm concept and design. Section V will show the simulation experiment and results, and finally Section VI will conclude the whole paper.

II. Related Work

Numerous researchers have proposed scheduling algorithms for parallel and distributed systems as well as Grid computing environment. The resource scheduling in grid is a NP complete problem [20]. Various algorithms have been designed to schedule the tasks in computational grids. The dynamic load balancing algorithms has to

bear the high communication overhead for exchanging state information frequently during its execution. In order to bring-down the communication overhead Martin et al. [18] studied the effect of communication latency, overhead and bandwidth in cluster architecture for application performance. Distributed network computing environments have become a cost effective alternative to achieve high performance computation to solve large scale problems. Martino and Mililotti addressed the use of Genetic Algorithm (GA) and Tabu Search (TS) to solve the grid load balancing problem in the dynamic environment [5]. The suggested algorithms had their own limitations like those algorithms require extra storage and processing requirement at the scheduling nodes. Grid computing involves coupled and coordinated use of geographically distributed resources for purposes such as large scale computation and distributed data analysis [17].

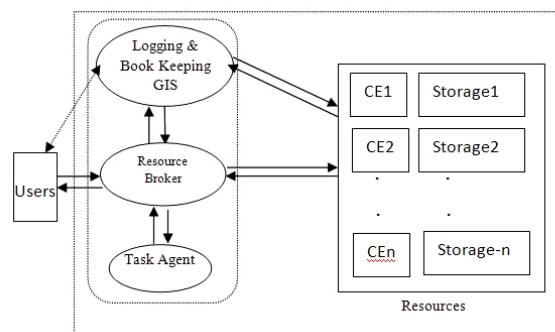
In order to place a new hybrid load balancing policy to work, it has to be integrated in static or dynamic load balancing techniques with the objectives to allocate effective computing resources, and identify the system imbalance immediately whenever a node become unable to participate in task processing. In [15] authors proposed a combination of intelligent agents and multi-agent approaches is applied for both local grid resource scheduling and global grid load balancing. In static grid load balancing, the iterative heuristic algorithm is better than the FCFS algorithm. Grid infrastructures are dynamic in nature in the sense of resource availability and hence a varying or dynamic network topology. Resource heterogeneity and network heterogeneity is also common in Grid computing environment. So, an efficient algorithm is needed that consider optimization in terms of time and cost for efficient scheduling and execution. Current grid resource scheduling algorithms are mainly based on User-Directed Assignment (UDA), Minimum Completion Time (MCT), Minimum Execution Time (MET), Min-Min [5], Max-Min, heuristic algorithm, genetic algorithm (GA)[5], Ant Algorithm (AA) [6], multi-Agent, computational economy model [7]. In [19] authors proposed a dynamic load balancing algorithm which considers CPU length, CPU and memory utilization and network traffic as load metric.

Although [6][14] presented scheduling algorithms based on ant algorithm, but the authors just analyzed ant algorithm based classified task scheduling method and ACA based scheduling without considering about the price attribute. In this paper, price factor and time factor are taken into consideration, and the objective is to save total execution time and cost.

III. System Model of the Grid

Grid system consists of coupling of heterogeneous resources (like different capacity processors, memory, networks, protocols, Operating Systems etc) [2]. In order to maximize the performance of computational Grid environment, it is foremost to distribute the load on different computational nodes in the grid. In grid computing environment, there exists more than one resource to process jobs in cooperation. One of the main challenges is to find the best or optimal resources to allocate to process a particular job so as to minimize the task computational time. Computational time is a measure of how long that resource takes to complete the job. In order to maximize the performance of computational Grid environment, it is essential to distribute the load on different grid nodes.

Figure 1 Grid Portal



Grid Resource Broker is connected to an instance of the user. Each grid job (composed of gridlets or tasks) is first submitted to the broker, which then schedule the grid job according to the user scheduling policy. *GIS* (Grid information Service) registers all the available resources. GIS contains all information about all available resources with their computing capacity and cost at which they offer their services to grid users. All Grid resources that join and leave the grid are monitored by GIS. Whenever a Grid broker has jobs to execute, it consults GIS to give information about available grid resources. Tasks register themselves to the GIS by specifying the QOS requirement such as cost of computation, deadline to complete the execution, the number of processors, speed of processing of internal scheduling policy and time zone. *Task Agent* or TA selects a resource as per the deployed algorithm (say Ant) for next task assignment and dispatch the task to selected resource through RMA. After task execution, results are received from resources and are returned to user by TA. Interaction among the entities is shown in the Fig.1.

Since the network topology in a Grid environment, is constantly varying, the proposed model in this paper captures this constraint as well by considering an arbitrary topology. The data transfer rate is not fixed and varies from link to link. The aim is to present load-balancing algorithm adapted to the heterogeneous Grid computing environment. An attempt has been made to formulate an adaptive decentralized sender-initiated load-balancing algorithm for computational Grid environments.

IV. Heuristic Load Balancing Algorithm

An algorithm for load balancing is successful, if the task executes within the deadline. All the tasks in this algorithm are distributed according to Ant Colony Algorithm. Heuristic Load Balancing Algorithm or HLBA 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. Investigations show that Ant has the ability of finding an optimal path from nest to food. On the way of ants moving, they lay some pheromone on the ground. While an isolated ant moves essentially at random, an ant encountering a previously laid trail can detect it and decide with high probability to follow it, thus reinforcing the trail with its own pheromone. The probability of ant chooses a way is proportion to the concentration of a way's pheromone. In HLBA, 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. The process of Heuristic based Load Balancing Algorithm is shown below:

Let the number of tasks (ants) in task set T maintained by task agent is P and the number of registered resources is Q . When a new resource R_i is registered to GIS, then it will initialize its pheromone based on:

$$\tau_i(0) = N \times M$$

Where N represents the number of processing elements and M corresponds to MIPS rating of processing element. Whenever a new task is assigned to or some task is returned from R_i , then the pheromone of R_i is changed as:

$$\tau_i^{new} = \rho \times \tau_i^{old} + \Delta\tau_i$$

Where $\Delta\tau_i$ is pheromone variance and ρ : $0 < \rho < 1$ is a pheromone decay parameter. When a task is assigned to R_i , its pheromone is reduced i.e. $\Delta\tau_i = -C$, where C represents the computational complexity of assigned task. When a task is successfully returned from R_i , $\Delta\tau_i = \Phi * C$, where Φ is the encouragement argument. Pheromone increases when task execution at a resource is successful. The possibility of next task assignment to resource R_j is computed as:

$$P_j(t) = \frac{[\tau_j(t)]^\alpha \times [\eta_j]^\beta}{\sum_r [\tau_j(t)]^\alpha \times [\eta_j]^\beta}$$

Where $j, r \in \text{available resources}$; $\tau_j(t)$ denotes the current pheromone of resource R_j ; and η_j represents the initial pheromone of R_j i.e. $\eta_j = \tau_j(0)$. α is the parameter on relative performance of current pheromone trail intensity, β is the parameter on relative importance of initial performance attributes. So, the algorithm is designed in order to influence the performance of the system which depends upon several factors and these factors could be simplified for reducing the complexities of the method to be used.

Heuristic Load Balancing Algorithm:

Begin

Phase 1 (Initialization Phase)

Initialise all parameters including resources (processing elements, MIPS rating), pheromone intensity, Task set etc.

Phase 2 (Operational Phase)

While (Task set $T \neq \Phi$)

Begin

Select the next task ' t ' from Task set T .

Determine next resource R_i for task assignment having high transition probability (or pheromone intensity).

$$P_i(t) = \max P_l(t); l \in q$$

Phase 3 (Result Phase)

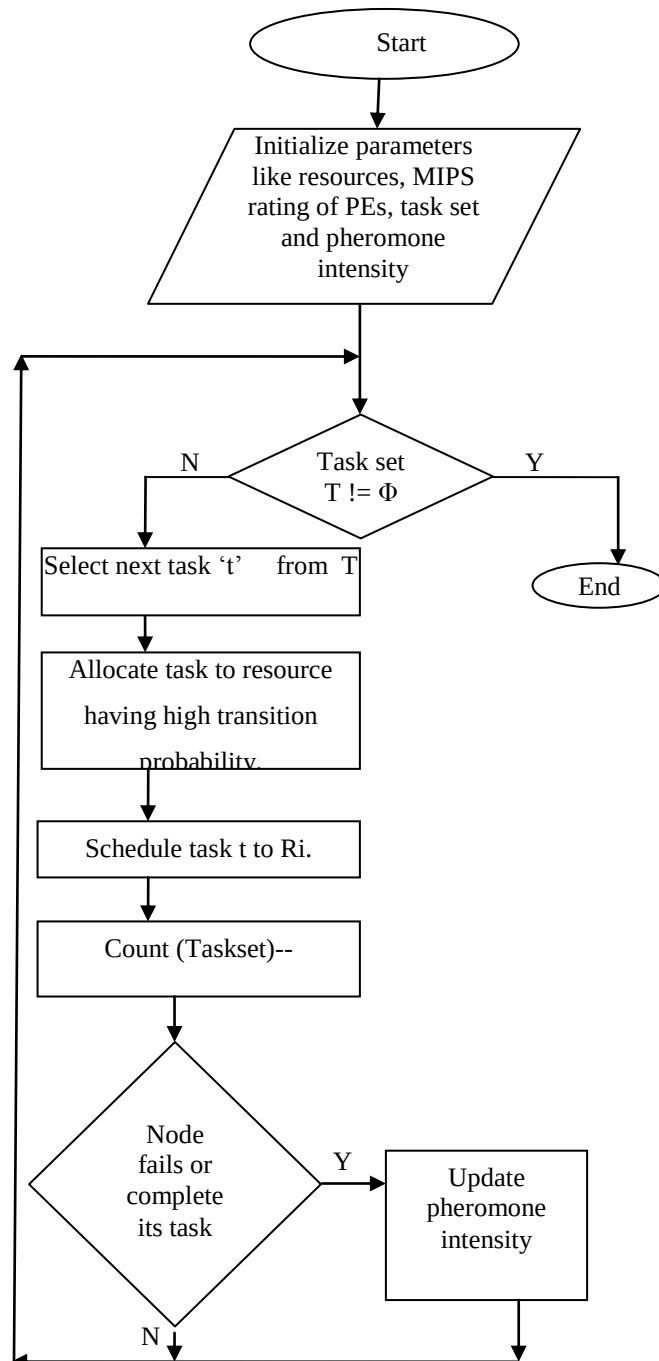
Schedule task ' t ' to R_i and update Task set by $T = T - \{t\}$.

If any node fails or complete its execution part update its pheromone intensity of that corresponding resource.

END While

END

Figure 2 Flowchart for the Heuristic Load Balancing Algorithm



All tasks are allocated to the resources depending on the value of pheromone or transition probability, which varies according to the status of the task at a particular resource. Resource having highest pheromone intensity would get the task before any other resource having lower than that intensity value. The algorithm is further elaborated by flowchart as given in Fig.2.

V. Simulation Results

To compare and establish the performance of the proposed approach over earlier developed approach simulation results are used. For simulation purpose GridSim simulator is used, which is a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing [11]. In order to test the

efficiency of the improved load balancing approach a comparison is done for the execution time (time utilized by PE for processing a task) of heuristic load balancing algorithm and non-heuristic load balancing algorithm.

Simulations for two different scenarios have been conducted to evaluate the performance of the proposed algorithm in terms of its processing time. First experiment processed 50-200 jobs with fixed number of resources taken as 10 while the second experiment processed 10 jobs with varying number of resources from 5-20. Details of the scheduling parameters are shown in the Table 1.

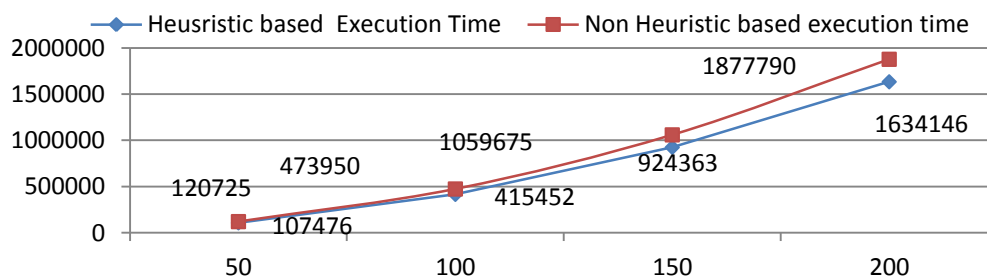
Table 1. Scheduling parameters for the simulations

Experiment	1st	2nd
No. of machines per resource	1	1
No. of Processing Elements(PEs) per machine	1-5	1-5
MIPS rating for PEs	10 or 50 MIPS	10 or 50 MIPS
Bandwidth	1000 or 5000	1000 or 5000

Average computation time (in seconds) has been used to compare the performance of algorithms. Fig. 3 depicts the comparison between the average computation times of the algorithms for the first experiment. In this number of resources remain fixed as 10 and the number of tasks varied from 50 to 200. It can be seen that the proposed algorithm labeled as Heuristic based load balancing algorithm has the lower execution time as compared to Non-Heuristic based load balancing algorithm at different instants.

A comparison between the computation times for the second experiment is shown in Fig. 4. Heuristic load balancing algorithm still performs better than non-heuristic load balancing algorithm. The performance of Heuristic approach is better because the communication that occur when each ant taking a step for searching the best resource is less when processing a large amount of jobs. Each ant will carry the history of visited node and will refer to it to find the best resource to process the job.

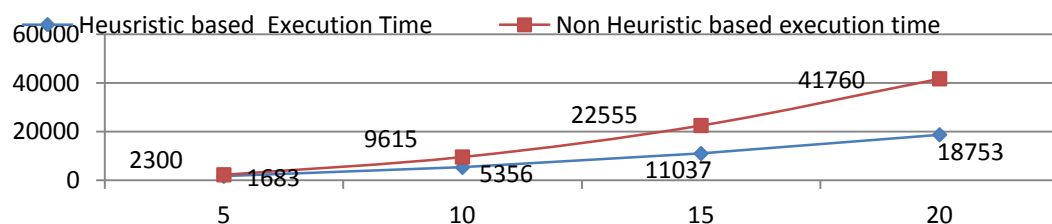
Figure 3. Comparison among the execution times of Heuristic and Non-Heuristic Algorithm.



In the second experiment the number of task remain fixed as 10 and number of resources varied from 5 to 20 and again compute the total execution time(in seconds) for different number of resources. Result supports the use of HLBA for time efficient execution of tasks then NHLBA.

Fig. 3 and Fig. 4 depict the result of the total execution time for two different scenarios. Results support the proposed approach by reducing the total execution time of jobs.

Figure 4. Comparison among the execution times of Heuristic and Non-heuristic Load Balancing Algorithms for Experiment 2.



VI. Conclusions

Multitasking and multiprocessing systems allow concurrently running tasks to share system resources such as processors, memory, storage, I/O, and network by scheduling their use for very short time intervals. The presented algorithm is adaptive, decentralized and distributed for load balancing among the aggregated

resources in the grid. Through a series of simulations by varying the number of tasks and resources the results has been obtained. It is found out that computation decreases until it reaches the optimal level i.e. resources with maximum computation capability. When the number of users competing for the same set of resource increases, there will be proportional impact on others depending on each user's strategies and constraints. Apart from complementing and strengthening the results of the proposed method, the simulations demonstrate the capability of GridSim and the ease with which it can be used for evaluating performance of new scheduling algorithms. Result of the simulation shows that HLBA enhances the performance of the system by reducing the execution time of the jobs.

References

- [1] I. Foster, and C. Kesselman, "*Globus: a metacomputing infrastructure toolkit*", International Journal of High Performance Computing Applications, 1997, vol. 11, no. 2, pp. 115-128.
- [2] I. Foster, and C. Kesselman (editors), "*The Grid 2: Blueprint for a New Computing Infrastructure*", The Elsevier Series in Grid Computing, 2003, Morgan Kaufmann, USA.
- [3] I. Foster, C. Kesselman, and S. Tuecke, "*The anatomy of the grid: enabling scalable virtual organizations*", International Journal of High Performance Computing Applications, 2001, vol. 15, no. 3, pp. 200-222.
- [4] X.S. He, X.H. Sun, and G.V. Laszewski, "*QoS guided min-min heuristic for grid task scheduling*", Journal of Computer Science & Technology, 2003, vol.18, no. 4, pp. 442-451.
- [5] V.D. Martino, and M. Mililotti, "*Scheduling in a grid computing environment using genetic algorithms*", proceedings of the 16th International Symposium on Parallel and Distributed Processing, , 2002, pp. 235-239.
- [6] Z.H. Xu, X.D. Hou, and J.Z. Sun, "*Ant algorithm-based task scheduling in grid computing*", proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, 2003, vol. 2, no. 3, pp. 1107-1110.
- [7] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "*Economic models for resource management and scheduling in Grid computing*", Concurrency and Computation: Practice and Experience, 2002, vol. 14, no. 2, pp. 1507-1542.
- [8] R. Buyya, D. Abramson, and S. Venugopal, "*The Grid Economy*", proceedings of the IEEE, 2005, vol. 93, no. 3, pp. 698-714.
- [9] O.O. Sonmez and A. Gursay, "*A novel economic-based scheduling heuristic for computational grids*", International Journal of High Performance Computing Applications, 2007, vol.21, no. 1, pp. 21-29.
- [10] R. Buyya, and Manzur Murshed, "*GridSim: A Toolkit for the Modeling, and Simulation of Distributed Resource Management, and Scheduling for Grid Computing*", Concurrency and Computation: Practice and Experience, 2002, vol. 14, no. 1, pp. 1175-1220.
- [11] Li Hao, "*Implementation of Computational Grid Application Scheduling Simulation with GridSim Toolkit*", Journal of Jilin Normal University (Nature Science Edition), vol. 3, no. 1, pp. 63-64, 2003.
- [12] Wang yue, and Tao Hongjiu, "*Application in TSP Based on Ant Colony Optimization*", Journal of Wuhan University of Technology (Information and Managing Engineering Edition), 2006, vol. 28, no. 11, pp. 24-26.
- [13] Marco Dorigo and Luca Maria Gambardella, "*Ant colonies for the travelling salesman problem*", BioSystems, 1997, vol. 43, no. 2, pp. 73-81.
- [14] Chen Ye, "*Ant Colony System for Continuous Function Optimization*", Journal of Sichuan University (Engineering Science Edition), 2004, vol. 36, no. 4, pp. 117-120.
- [15] Junwei Cao and Daniel P. Spooner and Stephen A. Jarvis and Graham R. Nudd, "*Grid Load Balancing Using Intelligent Agents*", Future Generation Computer Systems, 2005, vol 21, pp 135-149.
- [16] C. Sosa, and A.S. Grimshaw, "*Bringing the Grid Home*", proceedings of 9th IEEE/ACM International Conference on Grid Computing, 2008, pp. 152-159.
- [17] Casey, L.M., "*Decentralized Scheduling*", The Australian Computer Journal, 1981, vol. 13, no. 2, pp. 58- 63.
- [18] R. Martin, A.Vahdat, D. Culler, and T. Anderson, "*Effects of Communication Latency, Overhead, and Bandwidth in a Cluster Architecture*" proceedings of 24th Annual Symposium Computer Architecture (ISCA '97), pp. 85-97.
- [19] Xu Zhihong, and Gu Junhua, "*Research on Ant Algorithm Based Classified Task Scheduling in Grid Computing*", Journal of Hebei University of Technology, 2006, vol. 35, no. 3, pp. 68-71.
- [20] Parisa Rahmani, Mehdi Dadbakhsh and Soulmaz Gheisari, "*Improved MACO approach for grid scheduling*", IPCSIT, Singapore, 2012, vol.3, pp 135-142.