Computational Grid System Load Balancing Using an Efficient Scheduling Technique

Prakash Kumar
CSE Department, MTU

Pradeep Kumar
CSE Department, MTU

Vikas Kumar
Eurus Internetworks

Abstract
Grid computing is the collection of computer resources from various locations to achieve a common goal. Grid computing is a type of parallel system which enables the dynamically selection, aggregation and distribution of geologically resources at run time depending on their user quality of self service requirement, availability, performance, cost, capability. Load balancing is very effective technique to reduce response time and to improve resources utilization, exploiting through proper distribution of the application. Assuming homogeneous set of nodes linked with homogeneous and fast networks, various load balancing algorithms were developed. Analyzing the past results and to improve the performance and throughput, proposed efficient algorithms with better scheduling policies.

Keywords
Grid computing, Load balancing, Scheduling, Cluster, Throughput, Conservative Backfilling algorithm, GAP Search

1. Introduction

A distributed computer system is considered to be a collection of autonomous computers or nodes, connected by a communication network and located at possibly different sites. Grids enable the selection, aggregation, sharing of a wide variety of resources including storage systems, supercomputers, data sources, and specialized devices that are geographically distributed and owned by different organizations for solving data intensive problems and large-scale computational problems in engineering, science. Processing power, resources are much larger than the traditional node distributed computing environment computing nodes, where each computing resources according to the system's scheduling policy of the tasks assigned to their scheduler and implementation. Resource management and scheduling is the strongest key grid services, but to achieve scheduling and efficient grid resource management, load balancing and task scheduling is one of the key issues that must be addressed. The technique of load balancing is to distribute workload across two or more computing nodes, in order to get maximum throughput, optimal resource utilization, minimize response time, and avoid overload. Two main aspects that have to be considered in implementing any load balancing algorithms are scalability and adaptability. Resource or task assignment is an important issue in grid computing systems, which provides a better exploitation of the system parallelism and improves its performance. Many applications can benefit from the Grid infrastructure, including data exploration, collaborative engineering, high-throughput computing, and in fact distributed supercomputing. Resources in grid environment are geographically distributed in a large scale way and time to time changes resource performance.Workload and resource management are two essential functions provided at the service level of the Grid software. A grid based distributed system can utilize the computational resources efficiently by allowing multiple independent jobs to run over a network of heterogeneous clusters. In this paper, we propose load balancing algorithm that can handle heterogeneous grid sites. The proposed algorithm will balance the load in the grid based on the queue length of resource and transfer the job to resource having minimum queue length. The proposed algorithm will be implemented using Grid simulator toolkit using net beans IDE [2].

2. Previous Work

The structure of the Grid comprises characteristics of loosely coupled as well as tightly coupled systems; homogeneous as well as heterogeneous systems. Load balancing strategies aim to adapt the load optimally to the environment and they mainly consider the application running on a parallel, homogeneous system [3]. An efficient load balancing algorithm namely Sender Initiated load balancing (SI-LB) for grid has been presented where the availability of resources and jobs are dynamic. Through simulation experiments, it is seen that the SI-LB algorithm provides shorter response time, enhances the resource utilization and balances the load in an effective manner [1]. Some work addresses the problem of scheduling and load balancing in a grid architecture where computational resources are dispersed in different administrative domains or clusters which are connected to the grid scheduler by means of heterogeneous communication bandwidths is considered [11]. There are many researchers worked with load balancing approaches

Manuscript received August 5, 2015
Manuscript revised August 20, 2015

and uses different scheduling policies to improve the throughput and minimize the response time.

3. Grid Model

Grid model basically describes three different associative levels as: Level 0, which is acting as a Grid manager; Level 1, which is acting as cluster manager; Level 2, which is acting as processing elements. Grid manager duty is to scheduling tasks to balance the workloads [7].Particular Gird model is shown in the figure 1 and describing its Levels as follows:

Level 0: At this level, we find the clusters are associated within a particular Grid. Grid level is basically responsible for: maintaining the clusters and feedback from its associated processing elements.

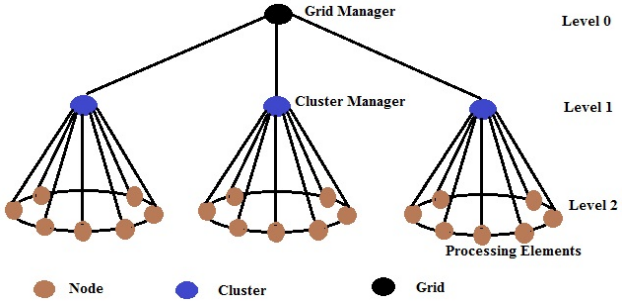


Figure 1: Grid computational model

Level 1: Each cluster manager of this level is associated with a physical cluster of the Grid. In our load balancing strategy, this manager is responsible for: sending the load balancing decisions to the worker nodes which they manage for execution; initiating a global load balancing, which we will call inter-clusters load balancing; deciding to start a local load balancing, which we will call intra-cluster load balancing; estimating the workload of associated cluster and diffusing this information to other cluster managers; maintaining the workload information related to each one of its worker nodes.

Level 2: At this level, we find the worker nodes of a Grid linked to their respective clusters. Each node at this level is responsible for: sending this information to its cluster manager; maintaining its workload information; performing the load balancing decided by its cluster manager [10, 14].

A grid manager which is present in the grid calculates the available capacity of each supernode. The schematic structure of cluster formation is shown in figure 2. Considering figure 2, the arrival of nodes and the application of decentralized load balancing algorithm with some cluster formation constraints lead to the formation of clusters [5].

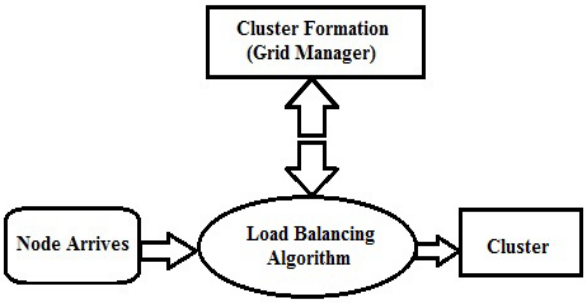


Figure 2: Cluster formation structure

According to cluster formation structure, when a task arrives, it is assigned to exactly one cluster for processing. A timer is assigned to each task, if the task is not processed within the assigned time, then the task is given highest priority for execution and when a fault occurs, each cluster has a master which takes the decision of load balancing [5]. To improve the performance of clusters in a grid, workload is balanced among the clusters. A workload is defined to be a job which can be an independent program or a partitioned module of a parallel program.

4. Load Balancing Approaches

Algorithms on basis of condition of system can be categorized as static, dynamic, adaptive algorithms. In static algorithms, scheduling is based on a policy that takes place as default. Condition of the system in every scheduling will be specified and based on this, scheduling takes place and no more scheduling will take place until the work is done. A dynamic algorithm adapts its decision with the system and in processing duties with changing system condition with tuned system allocated to grid may change. Such load balancing on basis of the present condition system makes decision and quickly adapts with workload fluctuations. Adaptive algorithm is a special type of dynamic algorithms which in spite of normal dynamic algorithms polices and scheduling where its parameters are constant. This algorithm change its parameters and scheduling policy on basis of general [6]. Dynamic algorithm is classified into two load balancing algorithm that is centralized load balancing and decentralized algorithm. The centralized approach is a simple approach and is beneficial when the communication cost is less significant and it is mainly used for a small size grid. The decentralized algorithms are scalable and have better fault tolerance. Although the decentralized approach is suitable for dynamic heterogeneous resources it increases the communication overhead to a large extent. The decentralized approach is preferred because elements of the network may vary in capacity or number during run time. The centralized and

decentralized load balancing algorithms can be further classified into sender-initiated algorithms, receiver-initiated algorithms and symmetrically initiated algorithms according to their location policies. Various kinds of location methods are sender initiated, receiver initiated and symmetrically initiated algorithm. Sender initiated algorithms let the heavily loaded nodes take the initiative to request the lightly loaded nodes to receive the jobs; while receiver initiated algorithms let the lightly loaded nodes invite heavily loaded nodes to send their jobs. Symmetrically initiated algorithms combine the advantages of both sender and receiver initiated algorithms [6, 7].

5. Problem Statement

A typical distributed system will have a number of interconnected resources which can work independently or in cooperation with each other. Each resource has owner workload, which represents an amount of work to be performed and every one may have a different processing capability. To minimize the time needed to perform all tasks, the workload has to be evenly distributed over all resources based on their processing speed. The essential objective of a load balancing consists primarily in optimizing the average response time of applications, which often means maintaining the workload proportionally equivalent on the whole resources of a system. Although the centralized approach is used currently, it limits the scalability of the grid by becoming a bottle neck and also failure of central controller can cause the entire system to fail [1]. If jobs distributed in girds unevenly, then potential would force jobs to be moved from overloaded nodes to lightly loaded nodes and grids would converge to the balanced state eventually [8]. Task assignment problem is a combinatorial optimization problem which consists of assigning a given computer program formed by a number of tasks to a number of processors/machine and this is subject to a set of constraints, and in such a way a given cost function to be minimized [12]. An experimental result shows the fewer throughputs, higher waiting time and less cluster utilization.

6. Proposed Methodology

Grid computing allows a cluster of loosely coupled computers to perform large tasks or tasks that generally consume more resources and time than is feasible for a single system [9]. Grid architecture identifies the fundamental system components, specifies purpose and function of these components, and indicates how these components interact with each other. Grid architecture is protocol architecture, with protocols defining the basic mechanisms by users and resources negotiate, establish, manage and exploit sharing relationships. Grid architecture is also a services standards based open architecture that facilitates extensibility, interoperability, portability and code sharing. A heterogeneous grid environment by using various resource specifications was built, which proposes the method of creating a user job and different types of heterogeneous resources. The resources differ in their operating system type, CPU speed, RAM memory [11].

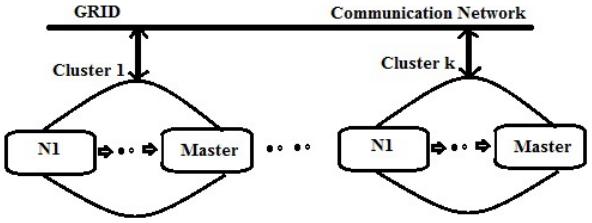
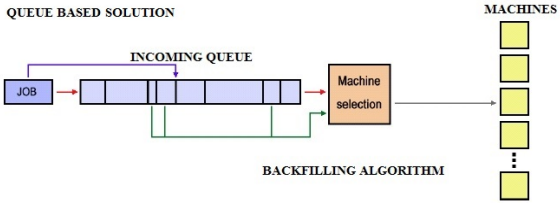


Figure 3: Topology of the Grid

Grid manager calculates the minimum communication cost of sending or receiving jobs to/from remote clusters based on the information collected in the last exchange interval. Master of each cluster also runs the Grid manager as shown in figure 3 [1]. We have two techniques namely queue-based (shown in figure 4) and schedule-based techniques (shown in figure 5). Basically we have applied two efficient algorithms with respect to getting better throughput and less response time. The algorithms are namely Consecutive Backfilling algorithm and Gap search algorithm. Conservative algorithm tried to maximize the utilization at every scheduling step, thereby reducing the mean response time. The main task of this algorithm is to select jobs from the waiting queue and assign available processors to them to maximize utilization



.

Figure 4: Queue based solution technique

Gap search select the first gap in the scheduler to be filling by a new job have been introduced. Fill the first gap with possible jobs that can fix in, biggest gap search for the biggest holes in the queue and match them with any jobs that can fix in. If the hole or gap does not match the jobs or too small for the job, the system has to search again [15, 16].

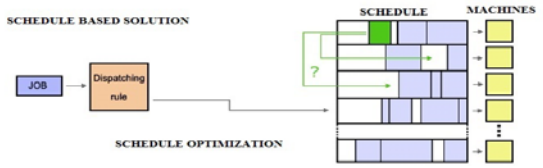


Figure 5: Schedule based solution technique

The scheduler is concerned mainly with: Throughput, which is the total number of processes that complete their execution per time unit; Turnaround time , which is the total time between submission of a process and its completion; Response, which is the amount of time it takes from when a request was submitted until the first response is produced. A Computational Grid is a software and hardware infrastructure that provides consistent, dependable, inexpensive and pervasive access to high-end computational capabilities [13].

7. Experimental Results

For analysing experimental results, we used GridSim toolkit, which is java-based discrete-event Grid simulation toolkit. The toolkit focuses on modelling and simulation of heterogeneous grid resources, application models and users. It can also be used for the modelling and simulation of application scheduling on various classes of parallel and distributed computing systems such as grids, clusters [4, 11].

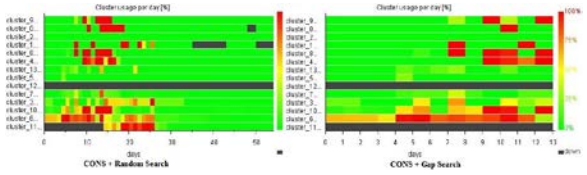


Figure 6: Comparison between previous algorithm and proposed algorithm with respect to cluster usage per day

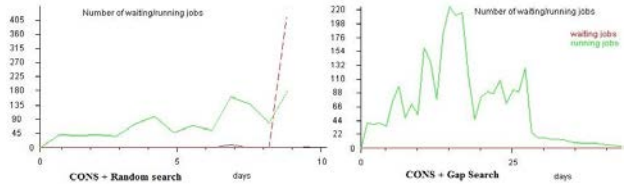


Figure 7: Comparison between previous algorithm and proposed algorithm with respect to running jobs per day

Experimental result shows the comparison between previous algorithm and proposed algorithm with respect to cluster uses per day and number of waiting/running jobs per day. Figure 6 show the result of proposed algorithm is better than the past. Figure 7 shows the waiting jobs using the proposed algorithm are improved.

Conclusions and Future Scope

Testing and evaluating the performance of our model, we developed our strategy under the GridSim simulator written in Java. We have randomly generated clusters with different characteristics and a set of dependent tasks. The experimental results are encouraging since we can significantly reduce the average response and waiting time. The simulation results show that with increase in the number of super nodes and the execution time is less as compared to existing algorithm. We also assume that cluster utilization is effective, throughput is better and response time is better than past algorithm. In the future we want to improve the proposed strategy by a better scheduling approaches using networking concepts or integrating the multi-agent systems.

References

[1] Jasma Balasangameshwara, Nedunchezhian Raju," A Symmetric-Initiated Load Balancing Model for Computational Grid Systems" International Conference on Network Communication and Computer, IEEE, 2011.
[2] Manpreet Singh, Sandeep Kumar Goyal, Vishal Gupta,"An Adaptive Load Balancing Algorithm for Computational Grid", JET, Vol. 1, Issue 2, June 2013.
[3] Jagdish Chandra Patni, Dr. M.S. Aswal, Om Prakash Pal, Ashish Gupta,"Load balancing strategies for Grid Computing",IEEE,2011.
[4] Jasma Balasangameshwara, Nedunchezhian Raju, "A hybrid policy for fault tolerant load balancing in grid computing environments", Elsevier, Journal of Network and Computer Applications, 35, Page no. , 412-422, 2012.
[5] Itishree Behera, Chita Ranjan Tripathy, Satya Prakash Sahoo," An Efficient Method of Load Balancing With Fault Tolerance for Mobile Grid", IJCST ,Vol. 3, Issue 3, July - Sept 2012.
[6] Mohsen Moradi, Mashaala Abbasi Dezfuli, Mohammad Hasan Safavi, "A New Time Optimizing Probabilistic Load Balancing Algorithm in Grid Computing",IEEE,2010.
[7] Mehdi Nikkhah, Raheleh Safaeipour, Mohsen Moradi ,"Investigating of Probabilistic Load Balancing Algorithms in Grid Computing", International Conference on Education Technology and Computer,IEEE,2010
[8] Jie Hu, Raymond Klefstad, "Decentralized Load Balancing on Unstructured Peer-2-Peer Computing Grids", Network Computing and Applications, IEEE, 2006.
[9] V. Prasanna Venkatesh,V. Sugavanan,"High Performance Grid Computing and Security through Load Balancing", International Conference on Computer Science and technology,IEEE,2009.
[10] N. Malarvizhi, Dr. V. Rhymend Uthariaraj, "Hierarchical Load Balancing Scheme for Computational Intensive Jobs in Grid Computing Environment", ICAC, IEEE, 2009.

[11] Dinesh S. Gawande, Rajesh C. Dharmik, Chanda Panse, "A Load Balancing in Grid Environment", IJERA, Vol. 2, Issue 2, March 2012.
[12] Meddeber Meriem, Yagoubi Belabbas, "Tasks Assignment for Grid Computing", International Journal of Web and Grid Services, ACM, Vol. 7 Issue 4, Pages 427-443, January 2011.
[13] Dj Tayeb Lilia, Halima Si Moussa, "Load Balancing in Grid Computing", Asian Journal of Information Technology, Medwell, Vol. 5, Issue 10, Pages 2095-1103, October 2006.
[14] Belabbus Yagoubi, Meriem Meddeber, "Distributed Load Balancing Model for Grid Computing", ARIMA Journal, Vol. 12, Pages 43-60, September 2010.
[15] Zafril Rizal M Azi, Kamalrulnizam Abu Bakar, Mohd Shahir Shamsir, "Scheduling Grid Jobs Using Priority Rule Algorithms and Gap Filling Techniques", International Journal of Advanced Science and technology, Vol. 37, December 2011.
[16] Hasasn Rajaei, Mohammad Dadfar, "Comparison of backfilling Algorithms for Job Scheduling in Distributed Memory Parallel System", American Society for Engineering Education, 2006.