Load Balancing Strategy for Computational Grid System using an Improved Algorithmic Approach

Prakash Kumar
MTU, Noida
Knowledge Park-2
Greater Noida, India

Pradeep Kumar
MTU, Noida
Knowledge Park-2
Greater Noida, India

Vikas Kumar
Eurus Internetworks
Safdarjung Enclave
New Delhi, India

ABSTRACT
Grid system is interconnected computer systems where the machines utilize the same resources collectively. Grid computing usually consists of a main computer that distributes information and tasks to a group of networked computers to accomplish a common goal.The goal of Grid computing is to create the delusion of a simple but large and authoritative self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources. A load balancing strategy chooses the resources that should be used to run a job in order to improve a given performance measure. Backfill is a scheduling optimization technique which allows a scheduler to make better use of available resources by running jobs out of order. Enabling backfill will increase system utilization and improve turnaround time by an even greater amount. Objective of the grid environment is to achieve high performance computing by optimal usage of geographically distributed and heterogeneous resources.

General Terms
Grid computing, Load balancing, Distributed system, Grid model, Workloads, Resources, Throughput.

Keywords
Grid, Load balancing, Scheduling, Cluster, Random search, Backfilling algorithm, Gap search.

1. INTRODUCTION
A computational Grid is a hardware and software communications that provides consistent pervasive and inexpensive access to high end computational capacity. An ideal grid environment should provide access to all the available resources seamlessly and fairly [9]. In Grid environment, various workstations or computers are linked through a communication network to form a large loosely coupled distributed computing system. The problem of task scheduling and load balancing in distributed system are most important and challenging area of research in computer engineering field. Task Scheduling and load balancing in distributed system has an essential role in overall system performance [6]. When load on the computers in a distributed environment has important variance of workloads, high performance can be achieved by redistributing loads. The task of redistributing the loads on the computers is called load balancing. Load Balancing in a distributed system is an important process to reduce delays and to improve response times in order to speed up applications and results. In Dynamic Load Balancing, it compares the various approaches to Load Balancing in distributed system, with focus on dynamic approaches in general; some processes are discussed in detail to simplify issues concerning factors contributing to load and the mechanisms used in any balancing process [8]. Grid resources are distributed geographically in a large-scale way and resource presentation changes quickly over time. Scheduling process directs the job to appropriate resource and monitoring process monitors the resources. The resources which will be heavily loaded will act as server of task and the resources which are Lightly Loaded will act as receiver of task. Task will be migrated from heavily loaded node to lightly loaded node. Resources are dynamic in nature so the load of resources varies with change in configuration of Grid so the Load Balancing of the tasks in a Grid environment can significantly manipulate Grid's performance. A variety of strategies, algorithms and policies have been proposed, implemented and classified for implementing Load balancing in Grid environment.

2. PREVIOUS WORK
Implemented various scenarios showing the intra site, inter-site and inter cluster load balancing in grid environment. For intra-site load balancing, have queue length of computing elements from 12 to 45 and for inter-site load balancing queue length of computing element is increased from 46 to 85 and then inter site load balancing is done [10]. The changes of computation nodes in a non-dedicated grid are often volatile: some changes might lead to an obvious performance decline while others might be insignificant [13]. A tasks placement strategy is promoted; it has the advantage of being able to divide the input task graph into set of connected component in order to reduce the response time of system application [9]. Real-time load balancing operations are implemented by sending a 'change command' from one power system to the other power system(s) once the load exceeds the specific limit [12]. The scheduling strategy that incorporates all the techniques: gang scheduling, backfilling, and migration consistently outperforms the others for average job slow down, job wait time, and loss of capacity. It also achieves the highest system utilization, allowing the system to achieve up to 95 percent utilization [14]. Parallel job scheduling is proposed to solve the problem of the parallel jobs being scheduled inefficiently and in effectively by existing methods when there are multiple jobs that can be combined [1].

3. LOAD BALANCING
Load balancing is a technique to improve resources, utilizing parallelism, exploiting throughput creativeness, and to reduce response time through an appropriate distribution of the application.

3.1 Types of Load Balancing Algorithm
A load balancing has basically two strategies as Static Load Balancing (SLB) and Dynamic Load Balancing (DLB) depending on the originality of the input parameters used for determining the load distribution [6, 7].

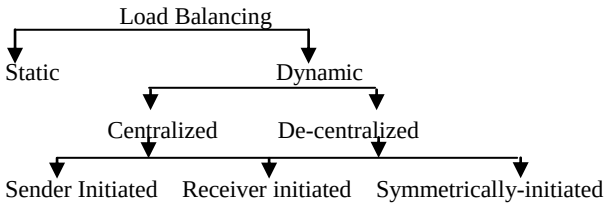

Figure 1: Types of Load Balancing

In a centralized scheme, the load balancer is located on one master workstation node and all decisions are made there only. In de-centralized approach the load balancer is replicated on all workstations and there are different algorithms used in de-centralized scheme for job selection. In sender-initiated policies, jam-packed nodes attempt to move work to lightly-loaded nodes. In receiver-initiated policies, lightly-loaded nodes look for heavily-loaded nodes from which work may be received.

3.2 Issues in Load Balancing:

a) Load balancing is critical because processes may migrate from one node to another even in the middle of execution to make sure equal workload.

b) Load balancing and task scheduling in distributed operating systems is a critical factor in overall system performance because the distributed system is non-preemptive and non-uniform, that is, the processors may be different.

c) One of the crucial aspects of the scheduling problem is load balancing. The challenge for a scheduling algorithm is that the requirements of equality and data locality often clash.

d) An important problem is to decide how to get stability in the load distribution between processors so that the computation is completed in the shortest possible time.

e) A good load balancing scheme needs to be general, scalable, stable, and to add a small overhead to the system. These requirements are mutually dependent.

f) Algorithms for load balancing have to rely on the supposition that the on hand information at each node is exact to prevent processes from being continuously circulated about the system without any progress [6].

3.3 Metrics for load measurement

Design takes into explanation the following parameters to measure the load at each node in the system; based on which the decision for further distribution is done.

g) maximum CPU utilization

h) maximum CPU queue length

i) maximum memory queue length

j) maximum memory utilization

k) maximum I/O utilization

l) Maximum I/O queue length [7].

4. SCHEDULINNG APPROACH

Scheduling is important for grid resource utilization. For many scheduling algorithms, they often assume that the degree of computing stability is known in advance, and then they can generate a job allocation pattern based on certain predicted machine conditions [13]. Dynamic Load Balancing (DLB) is used to provide application level load balancing for individual parallel jobs which ensures that all loads submitted through the DLB environment are distributed in such a way that the overall load in the system is balanced and application programs get highest benefit from available resources [11]. Backfilling attempts to assign unutilized nodes to jobs that are behind in the priority queue (which are waiting jobs), rather than keep them idle. To prevent starvation for larger jobs, (conservative) backfilling requires that a job selected out of order completes before the jobs that are in front of it for the priority queue are scheduled to start [14].

4.1 Dynamic Load Distribution:

Load distribution seeks to get better the performance of a distributed system, usually in terms of resource availability or response time, by allocating workload amongst a set of cooperating hosts. Dynamic load distribution systems typically examine the workload and hosts for any factors that may affect the choice of the most suitable assignment and distribute jobs consequently [8].

4.2 Cluster Creation:

i. Initialize cnumber, site, ce, qlength, nos.
ii. For cnumber=1 to 10.
a) Generate number of sites between 1 to 5 randomly.
b) Generate computing element ce of each site between 1 to 5 randomly.
c) Generate queue length of each computing element ce between 1 to 50 randomly.
End For.

4.3 Node Selection:

Any number of nodes can make up a group, but limiting the size to only two or three gives us the advantages of low overhead, fall of job thrashing as well as other advantages; like scalability, robustness, stability and efficiency.The selection of nodes, for three nodes per cluster, or for two nodes per cluster can be done using the code:

```
Cycle =1
While (true)
For (i=0;i< groups ; i++)
For (j=0;j< clustersize ; j++)
X=i*clustersize+j+cycle;
If (x>nodes)
C[i+1][j+1] = x % Nodes
Else
C[i+1][j+1] = x;
The Load Balancing Code is placed in this area
If (++cycle>Nodes) cycle=1;
```

In this configuration, the load over the groups consisting of nodes I;j;k :

NetLoad = int ((load I+ load j)/2) , for two node groups
NetLoad = int ((load I+ load j+ load k)/3) , for three node groups [8].

4.4 Algorithmic Approach:

Proposed algorithm attempts to find the best possible configuration for the multiple queues. It tries to maximize the utilization at every scheduling step, thereby dropping the mean response time. The jobs are divided into two parts:

running and waiting. The jobs that are waiting may be either in the waiting queue or in the selected queue (for execution). All jobs have two attributes: estimated computing time remaining and size (number of processors or computing nodes required). The system free capacity is defined as the number of idle processors currently not assigned to any jobs. The goal of parallel system scheduler design is to maximize both system utilization and user occurrence while preserving scalability of the algorithm. The main task of this algorithm is to select jobs from the waiting queue and assign available processors to them to maximize utilization. Allowing small jobs from the back of the queue to execute before larger jobs that arrived earlier. Gap search algorithm selects the first gap in the scheduler to be fills by a new job have been introduced and 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 repeatedly. When a job is submitted, it is given a reservation to start at the earliest time that does not violate any earlier existing reservations [4]. Backfill is a technique in which lower priority jobs requiring fewer resources are initiated before one or more currently waiting higher priority jobs requiring as thus far unavailable resources. Processors are frequently the resource involved and the purpose of backfilling is to reduce average wait time and increase system utilization [3]. Backfilling improves resource utilization by allowing the first job of the queue to reserve resources that are not available and by evaluating the possible execution of succeeding jobs in the submission queue [5]. Here, backfilling technique is used to recover the past waiting jobs which were in the waiting queue for getting its running time.

4.5 System Architecture:

There are various types of grid resource allocation architectures can be classified as: centralized, decentralized, and a hierarchy one. A centralized scheme is likely to lose scalability while a decentralized enough state information to get an optimal result, research is based upon a hierarchy structure which shows its advantage in terms of flexibility and adaptive capability. Size of distribution system required, network bandwidth required number of servers and memory required with disk and computational capacity [2].

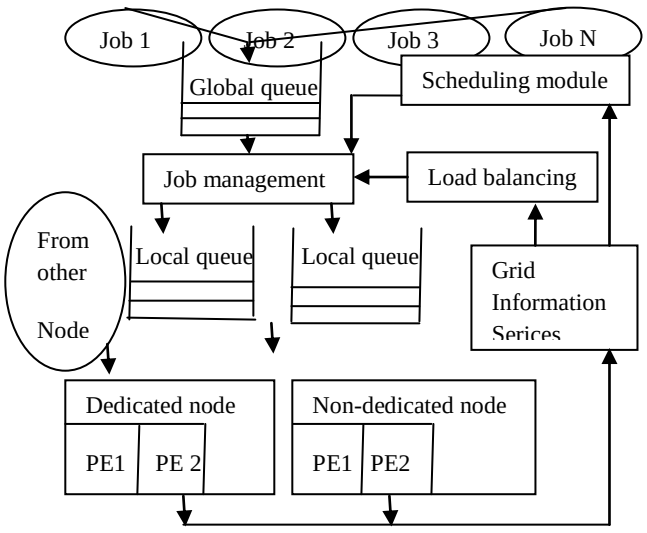

Figure 2: System architecture

In view of figure 2, all the job buffers are assumed infinite so the situation of overloads in terms of storage space is discussed. For the non-dedicated nodes, jobs from other grids may be submitted at times so its capability of dealing with current grid's jobs will be affected [13].

5. PROBLEM STATEMENT

Grid computing enables sharing, selection and aggregation of large collections of geographically and organizationally distributed heterogeneous resources for solving large-scale data and compute intensive problems. Resources are dynamic in nature so the Load of resources varies with change in configuration of Grid and it makes Load Balancing more important in case of Grid environment. The focus of the study is to consider factors which can be used as characteristics for decision making to initiate Load Balancing. Load Balancing is one of the most important factors which can affect the performance of the grid application. Main difference between existing Load Balancing algorithm and proposed Load Balancing is in implementation of Scheduling policies: selected by Selection Policy. To improve the global throughput of these environments, effective and efficient load balancing algorithms are fundamentally important. The focus of the study is to consider factors which can be used as characteristics for decision making to initiate Load Balancing.

6. PROPOSED METHODOLOGY

Some load balancing strategies work well for applications with large parallel jobs, while others work well for short, quick jobs. Some strategies are focused towards handling data-heavy tasks, while others are more suited to parallel tasks that are computation heavy. While many different load balancing algorithms have been proposed, there are four basic steps that nearly all algorithms have in common:

a) Load monitoring (Monitoring workstation performance)

b) Exchanging this information between workstations (synchronization)

c) Calculating new distributions and making the work movement decision (rebalancing criteria)

d) Actual data movement (job migration)

6.1 Proposed Grid Model Components:

Cluster-Level manager (CM) can fully control the computing nodes within it, but cannot manage the computing nodes of other clusters directly. The computing nodes within the cluster are referred as friends. CM maintains the load information along with registration information of its computing nodes. In Cluster-Level, each friend runs a CM and role is to balance the intra-cluster workload. A designated friend with highest CPU speed in each cluster is treated as the cluster server or master. Cluster System Monitor (CS) determines the load index of computing nodes and provides this information to Cluster-level manager. Based on the load information Sender Initiated Load Balancing (SI-LB) chooses most suitable node for each job, thereby minimizing job execution time and maximizing system throughput. Load information, generally defined in term of load index is necessary condition of SI-LB algorithm. The load at each computing node contributes to the overall load of the cluster and can be determined as: CPU utilization, CPU speed and queue length. The load index is determined dynamically and the weighted sum of squares method is used to calculate the load at each computing node.

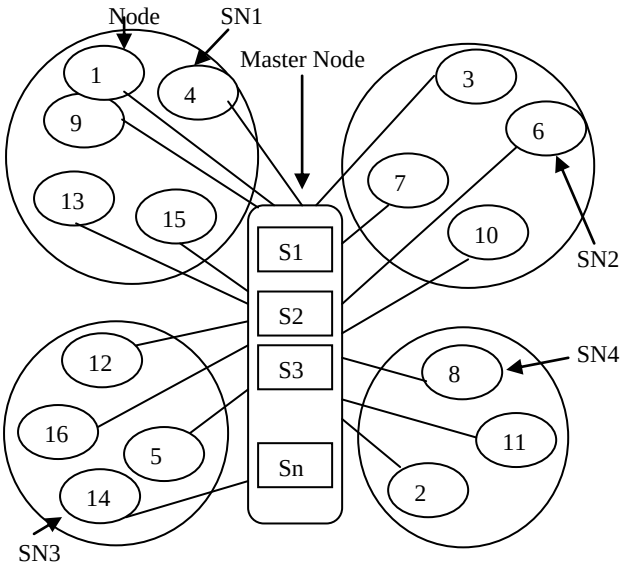

Figure 3: View of groups in simplified distributed system with supporting nodes

The Master Node MSN are replaced by various supporting nodes as *S1, S2, S3,...., Sp*. Each node in the system is allowed to access any supporting node (shown in Figure 3). When a node becomes overloaded, it first queries the member nodes of its group and collects the load information in a decentralized manner. There is a high probability that the Overloaded node will find a lightly loaded node in the same group Here, when a highly loaded node does not find a lightly loaded node in the same group, it randomly selects one supporting node and now supporting node follows the same procedure as pursued by MSN [7].

6.2 Java and ALA3 Simulator:

Java language offers several features that assist with easiness the development and deployment of a software environment for Grid computing. Java's network-centric approach and its built-in support for mobile code enable the distribution of computational tasks to different computer platforms. Java's platform-independent byte code can be executed securely on many platforms, making Java an attractive basis for portable Grid computing. Because of the heterogeneity of the hardware and of operating systems working by Internet users, it is essential that a platform for large-scale Grid computing be available for a large variety of different computer systems. Accordingly, a Java- based platform potentially allows every computer in the Internet to be exploited for distributed, large-scale computations, while at the same time the maintenance costs for the platform are minimal. For result, Alea3 is a tool which contains workload parsers that can read popular trace formats such as the Grid Workloads Format (GWF) and the Standard Workloads Format (SWF). Same as the GridSim, the Alea3 is an event-based modular simulator which consists of the centralized scheduler, the grid resource(s) with the local job allocation policy. The advanced Alea3 now also provides complex visualization tool which supports an export of simulation results into several bitmap formats particularly.

7. EXPERIMENTAL RESULTS

Simulation result shows that the proposed algorithm provides less response time, improvement in resource utilization and load balancing is done in effective manner.

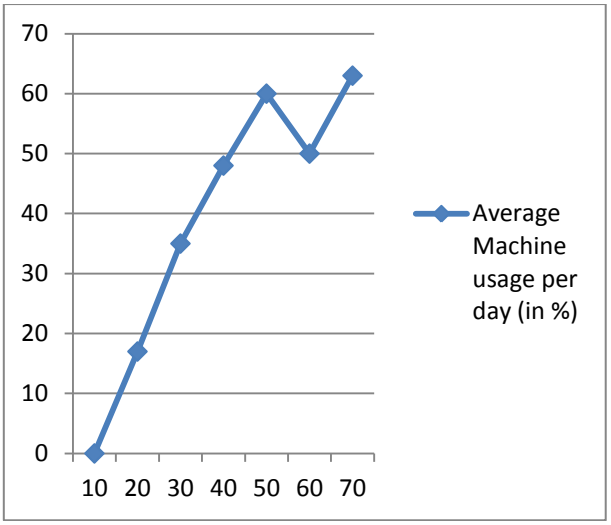

Figure 4: Average machine usage per day (%) in previous algorithm

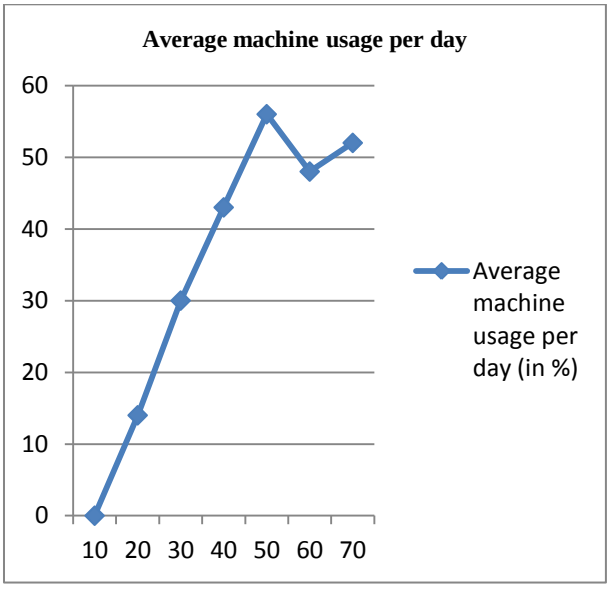

Figure 5: Average machine usage per day (%) in proposed algorithm

The final experimental result shows the comparison with respect to the average machine usage per day in previous algorithm and proposed algorithm and it is seen that proposed algorithm is better as shown in the Figure 4 and Figure 5.

8. CONCLUSION & FUTURE SCOPE

Focusing on Load Balancing and tried to present the impacts of Load Balancing on grid application performance and finally proposed an efficient Load Balancing algorithm for Grid environment. Aanalysed existing Load Balancing algorithm and proposed an enhanced algorithm which more efficiently implements three out of five policies implemented in existing Load Balancing algorithm. The goal of a truthful mechanism with verification is to give incentives to agents such that it is beneficial for them to report their true values and execute the assigned jobs using their full processing capacity. In future, would like to study this problem and

devise a fault tolerant load balancing mechanisms that exhibits good properties such as truthfulness and voluntary participation. Another idea is to use cryptographic protocols for secure cooperative Function Evaluation for computing the allocations and payments.

9. ACKNOWLEDGMENTS

Would like to express the deepest appreciation to my guide **Mr. Pradeep Kumar** and HOD **Dr. Prashant Singh** for their guidance and great support.

10. REFERENCES

[1] Shengwei Yi, Zhichao Wang, Shilong ma, Zhanbin Che,Yonggang Huang, Xin Chen, "An Effective Algorithm of Jobs Scheduling in Clusters", Journal of Computational Information Systems, Volume 6, Issue 10, October 2010.

[2] Ye Xia, Shigang Chen, Chunglae Cho, Vivekanand Korgaonkar, "Algorithms and Performance of Load Balancing with Multiple Hash Functions in Massive Content Distribution", Elsevier, Volume 53, Issue 1, January 2009, Pages 110-125.

[3] William A. Ward, Jr., Carrie L. Mahood, and John E. West, "Scheduling Jobs on Parallel Systems Using a Relaxed Backfill Strategy", Springer, Volume 2537, Pages 88-102.

[4] Srividya Srinivasan, Rajkumar Kettimuthu, Vijay Subramani, P. Sadayappan, "Selective Reservation Strategies for Backfill job Scheduling", Springer, Volume 2537, 2002, Pages 55-71.

[5] A.D.Techiouba, G.Capannini, Ranieri Baraglia, D. Puppin, M. Pasquali,"Backfilling Strategies For Scheduling Streams Of Jobs On Computational Farms", Springer, 2008, Pages 103-115.

[6] Abhijit A. Rajguru, S.S. Apte, "A Comparative Performance Analysis of Load Balancing Algorithms in Distributed System using Qualitative Parameters", International Journal of Recent Technology and Engineering (IJRTE), Volume-1, Issue-3, August 2012.

[7] Urjashree Patil, Rajashree Shedge, "Improved Hybrid Dynamic Load Balancing Algorithm for Distributed Environment", International Journal of Scientific and Research Publications, Volume 3, Issue 3, March 2013.

[8] Rose Suleiman Rotating Load balancing Algorithm in Distributed System, "Dynamic Rotating Load Balancing Algorithm In Distributed Systems", ICITNS, 2003.

[9] Meddeber Meriem, Yagoubi Belabbas, "Tasks assignment for grid computing", ACM, Volume 7, Issue 4, January 2011.

[10] Sahil Verma, Sandip Kumar Goyal, Kavita, "Tree based Apporach for Load Balancing in Grid Environment", International Journal of Engineering Research & Technology, Vol. 1 Issue 9, November 2012.

[11] Jagdish Chandra Patni, M. S. Aswal, Om Prakash pal, Ashish Gupta, "Load balancing Strategies for Grid Computing", Electronics Computer Technology (ICECT), IEEE, Volume 3, 2011, Pages 239-243.

[12] R. Al-Khannak, B. Bitzer, "Load Balancing for Distributed and Integrated Power Systems using Grid Computing", ICCEP, IEEE, May 2007.

[13] Yixiong Chen, "Load Balancing in Non-dedicated Grids Using Ant Colony Optimization", 4th International Conference on Semantics, Knowledge and Grid, IEEE, December 2008.

[14] Yanyong Zhang, Hubertus Franke, Jose Moreira, Anand Sivasubramaniam, "An Integrated Approach to Parallel Scheduling Using Gang-Scheduling, Backfilling, and Migration", IEEE Transactions On Parallel And Distributed Systems, Volume 14, Issue 3, MARCH 2003.

[15] Igor Grudenic, "Scheduling Algorithms and Support Tools for Parallel Systems", IEEE, July 2012.