Grid Environment Load Balancing

Dr.Belsam Jeba Ananth.M
Professor
DMI College of Engineering
Chennai

J.Anitha Thulasi
Asst professor
DMI college of Engineering
Chennai

ABSTRACT

Grid computing creates the illusion of a simple but large and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources which lead to the problem of load balance. The main goal of load balancing is to provide a distributed, low cost, scheme that balances the load across all the processors. To improve the global throughput of Grid resources, effective and efficient load balancing algorithms are fundamentally important. Focus of this paper is on analyzing Load Balancing requirements in a Grid environment and proposing an algorithm with machine learning concepts to find more efficient algorithm.

Key terms
Grid Computing, Load balancing, Machine learning, Job migration.

1. INTRODUCTION

Grid computing is a type of parallel and distributed system that enables the distribution, selection and aggregation of geologically resources dynamically at run time depending on their availability, capability, performance, cost, user quality of self-service requirement [1].Grid computing, individual users can retrieve computers and data, transparently, without taking into account the location, operating system, account administration, and other details. In Grid computing, the details are abstracted, and the resources are virtualized. Grid Computing should enable the job in question to be run on an idle machine elsewhere on the network [2]. Grids functionally bring together globally distributed computers and information systems for creating a universal source of computing power and information [3]. A key characteristic of Grids is that resources (e.g., CPU cycles and network capacities) are shared among various applications, and therefore, the amount of resources available to any given application highly fluctuates over time. Load balancing is a technique to enhance resources, utilizing parallelism, exploiting throughput improvisation, and to reduce response time through an appropriate distribution of the application [4].
Load balancing algorithm are two type static and dynamic,Static load balancing algorithms allocate the tasks of a parallel program to workstations based on either the load at the time nodes are allocated to some task, or based on an average load of our workstation cluster. The decisions related to load balance are made at compile time [5].

1.1. Static Load Balancing Algorithm

The decisions related to load balance are made at compile time when resource requirements are estimated. The advantage of algorithm is the simplicity according to both implementation and overhead, since there is no need to constantly monitor the nodes for performance statistics.

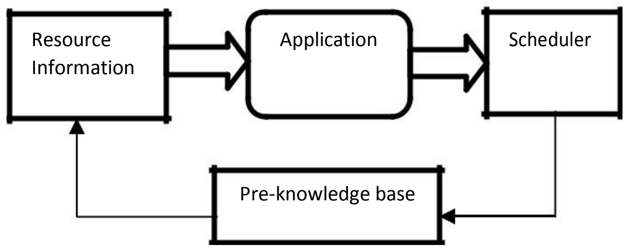

Figure. 1. Static Load Balancing [5].

1.2. Dynamic Load Balancing Algorithm

Dynamic load balancing algorithms make changes to the distribution of work among nodes at run-time; they use current or load information when making distribution decisions.

Dynamic load balancing algorithms are advantageous over static algorithms. But to gain this advantage, we need to consider the cost to collect and maintain the load information.

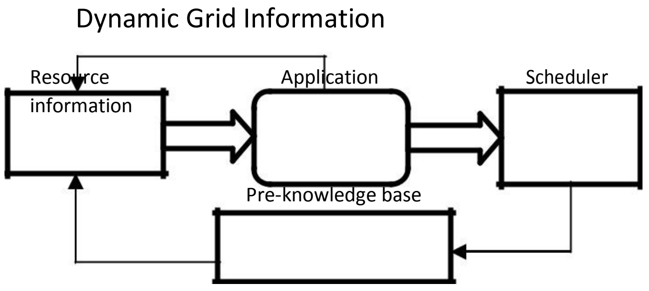

Figure. 2. Dynamic Load Balancing [5].

A few static load balancing techniques are Round robin algorithm, Randomized algorithm, simulated annealing orgenetic algorithms, and Dynamic load balancing algorithms make changes to the distribution of work among workstationsat run-time; they use current or recent load information when making distribution decisions [6].

Multicomputers with dynamic load balancing allocate/reallocate resources at runtime based on no a priori task information, which may determine when and whose tasks can be migrated [7].

As a result, dynamic load balancing algorithms can provide a major improvement in performance over static algorithms. However, this comes at the additional cost of collecting and maintaining load information, so it is important to keep these overheads within reasonable limits [8]. There are three major parameters which usually define the strategy a specific load balancing algorithm will employ [9].These three parameters answer three important questions:

- Who makes the load balancing decision
- What information is used to make the load balancing decision, and
- Where the load balancing decision is made.

Various Load Balancing Algorithms are available now days but they contain several drawbacks. Such type of problems can be eradicated by our proposed dynamic load balancing algorithm.

1.3 Load Balancing Policies

Load balancing algorithms can be defined by their implementation of the following policies [3]:

Information policy: specifies what workload information tobe collected, when it is to be collected and from where.

Triggering policy: determines the appropriate period to start a load balancing operation.

Resource type policy: classifies a resource as server or receiver of tasks according to its availability status.

Location policy: uses the results of the resource type policy to find a suitable partner for a server or receiver.
Selection policy: defines the tasks that should be migrated from overloaded resources (source) to most idle resources(receiver).

2. MACHINE LEARNING

Machine learning is the study of design and development of algorithms and techniques to make the computer learn stuff and integrate knowledge to continuously improve them to finish their tasks efficiently and effectively. In general, machine learning is about learning to do better in the future base on what was experienced in the past.
There are three main learning paradigms [6,7] according to the available information in the training examples. They are supervised learning, reinforcement learning and unsupervised learning.

2.1 Supervised learning

Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal) and the goal of the machine is to learn to produce the correct output given a new input.

2.2 Reinforcement learning

Reinforcement learning differs from the supervised learning problem in that correct input/output pairs are never presented. Further, there is a focus on on-line performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). Machine interact with its environment by producing actions a1, a2 ...These actions affect the state of the environment, which in turn results in the machine receiving some scalar rewards (or punishments) r1, r2,... The goal of the machine is to learn to act in a way that maximizes the future rewards it receives (or minimizes the punishments) over its lifetime.

2.3 Unsupervised learning

Unsupervised learning differs from other methods because in this method the machine receives the inputs only without outputs or rewards from its environment. The goal of unsupervised learning is to find the patterns in the inputs and build the representation of the input that can be used to make decision, predict future inputs and so on.

3. PROPOSED SYSTEM ARCHITECTURE

Based on the study of load balancing techniques in Grid environment, we proposed a technique which dynamically balances the load. The rules generated by data mining techniques are used for migrating jobs for load balancing. Load balancing should take place when some change occurs in the load situation. There are some particular activities which change the load configuration in Grid environment which can be categorized as following:

- Arrival of any new job
- Completion of execution of any job
- Arrival of any new resource
- Withdrawal of any existing resource

Whenever any of above mentioned activities happens we have to retrieve the current load information and the initial load information from the database. System will compare them and generate specific rules to migrate the jobs, so jobs will be distributed optimally on all nodes. After the job migration database will be updated by this current load information. This kind of approach is advent to balance the load in Grid environment.

Also with this kind of behavior using concept of unsupervised learning, we are able to decide a threshold value for each node, which will help to distribute the load at the initial level when this grid will be used in future.

Data analysis is a practice in which raw data is ordered and organized so that useful information can be extracted from it.The process of organizing and thinking about data is a key to understanding what the data does and does not contain. There area variety of ways in which people can approach data analysis and it is notoriously easy to manipulate data during the analysis phase to push certain conclusions or agendas. For this reason, it is important to pay attention when data analysis is presented, and to think critically about the data and the conclusions which were drawn. In this algorithm, Data analysis module will compare previously stored parameters with current parameters of different nodes and generates rules, used for job migration. Rules generated by data analysis module contain heavily loaded and lightly loaded node information. This information will be used to balance load accordingly. Job migration refers to the process of moving a job from one node to another. This facilitates load balancing by moving jobs from a heavily-loaded host to a lightly-loaded host.

Following is the proposed algorithm for Load Balancing:

Algorithm 1 Load balancing Algorithm

```
1: BEGIN
2: Previous = initial status of all nodes on grid.
3: while Tasks > 0 do
4: if Change state = true then
5: Current = Get current state();
6: Newrule = generate rule(Current, Previous);
7: Threshold = generate threshold(Threshold,Newrule);
8: Migrate jobs(Newrule);
9: Previous = Current;
10: end if
11: end while
12: END
```

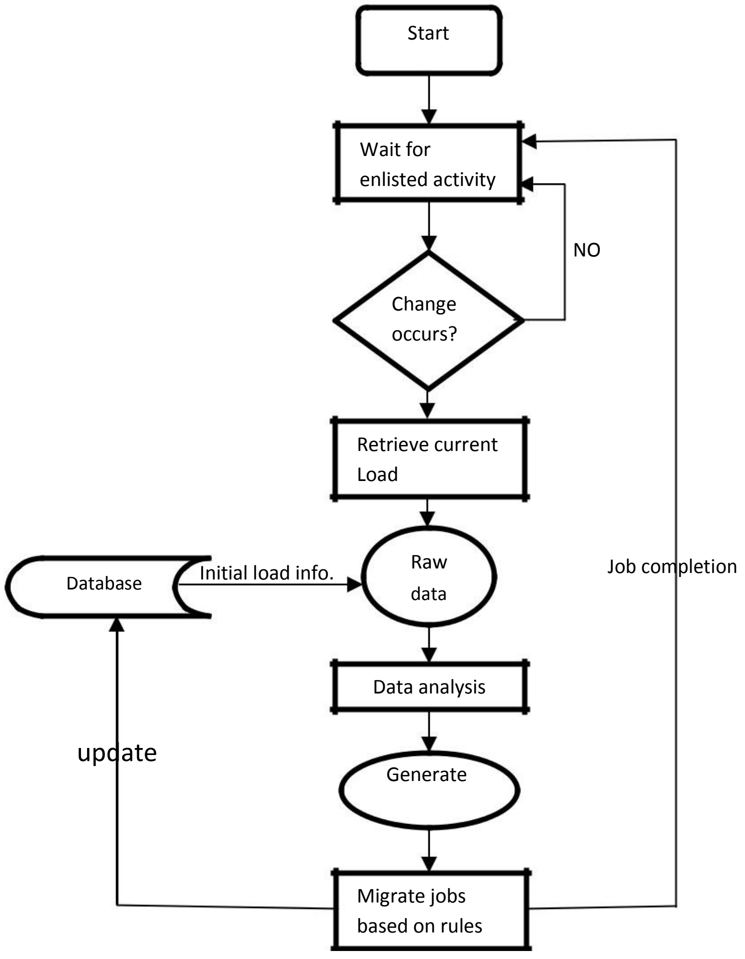

Figure 3. System Architecture

Variables Used:

Tasks : indicates no of jobs already running on grid.
Type - integer
Change state : indicates change in the state of load in grid.
Type - Boolean
Current : indicates current load status of all the nodes on grid.
Type - structure of different parameters of grid.
Previous : indicates load status of all nodes on grid before the change.
Type - structure of different parameters of grid.
Newrule : indicates the rules decided for load distribution on grid.
Type - structure of different parameters of node to transfer load.
Threshold : indicates maximum limit of handling load for each node on a grid.
Type - structure of arrays on different node and its threshold values.
Methods used:
Getcurrentstate() : to get the current status of load on each node of grid.
Generaterule (Current, Previous) : to generate rules to balance change occurred in the load as per the current and previous status of the nodes.
Generatethreshold (Threshold, Newrule) : to find out new threshold values of each nodes of grid for future using concept of data mining from old threshold and newly generated rules.
Migratejobs (Newrule) : to migrate jobs as per the newly generated rules.

4. CONCLUSSION

Our work focuses on design and implementations of innovative load balancing system for Grid environment using Machine learning principle proposed algorithm can use initial load information stored in the database at the initial level and current load information after load imbalance are first recorded as raw data. Then the raw data is processed and analyzed by different machine learning techniques. Finally, rules will be automatically generated. In the future work, we plan to extend the system to deal with the raw data and apply the data mining concepts to automatically generate rules, which can give better results than other existing methods.

5. FUTURE WORK

At the current stage, here proposed prototype system is able to collect data and generate the rules using learning method. To achieve load balancing in Grid environment, this system uses examples related to job distribution for providing initial knowledge to the system. Based on some predefined information and new generated rules, this system can take decisions to balance load.
Although, we can monitor many load balancing activities in our prototype system, but organizing that information and using them to take decisions is still a challenging task. A dynamic mechanism for adding, testing, and deleting rules can be done too. That means, we do not hard code the rules, but develop a system where rules can be easily added and deleted. Here proposed mechanism is able to add, test and delete rules. These rules are negotiable, based on need of the system.So that, when we perform experiments, we can simply add specific rules to the system. If the result is not satisfactory, we can modify rules for further experiments. In this way, we can efficiently perform experiments for efficient load balancing. Our future objective is to develop a system with the solution of the

challenges like defining templates of rules, categorizing and fining similar rules.

6. REFERENCES

[1] http://www.gridcafe.org/, Francois Grey, Matti Heikkurinen, Rosy Mondardini, Robindra Prabhu.
[4] B. Yagoubi, Y. Slimani, ”Task Load Balancing Strategy forGrid Computing”, Journal of Computer Science 3 (3): 186-194, 2007 ISSN 1546-9239 2007 Science Publications.
[5] Javier Bustos Jimenez, ”Robin Hood: An Active Objects Load Balancing Mechanism for Intranet”, Departamento de Ciencias de la Computacion, jbustos@dcc.uchile.cl, Universidad de Chile.
[2] Ratnesh Kumar Nath, ”Efficient Load Balancing Algorithm in Grid Environment”, Thapar University, Patiala, May2007.
[3] Belabbas Yagoubi and Yahya Slimani, ”Dynamic Load Balancing Strategy for Grid Computing”, World Academy ofScience, Engineering and Technology 19 2006.

[6] Zoubin Ghahramani, ”Unsupervised Learning”,GatsbyComputational Neuroscience Unit, zoubin@gatsby.ucl.ac.ukhttp://www.gatsby. ucl.ac.uk/ zoubin, University College London, UK, September 16, 2004.
[7] D. Zhang and J. J.P. Tsai,”Machine Learning Applicationsin Software Engineering”, World Scientific Inc., 2005.