The Critique on Load Balancing for Computational Grid

Khuman Hina
Computer Engineering
Parul Institute Of Engineering and Technology
Vadodara, India
Khuman.hina@gmail.com

Astha Baxi
Computer Science & Engineering
Parul Institute Of Engineering and Technology
Vadodara, India
adbaxi@gmail.com

***Abstract*— Computational grids provide a massive source of processing power. The potential of grid architectures provides ability to solve large scale advanced scientific and engineering applications whose computational requirements exceed the local resources as well as the job turnaround time through workload balancing across multiple computing facilities. 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 leads to the problem of load balance. Load Balancing strategies try to ensure that every processor in the system performs almost the same amount of work at any point of time. Efficient and effective algorithms are fundamentally important in order to improve the global throughput of this environment. The main aim of load balancing is to provide distributed, low-cost, schemas that balance the load across all the processors. While this reassessment, we found that hierarchical load balancing is quite feasible considering the computational grid environment.**

Keywords*— Computational grid, Load Balancing, Hierarchical load blancing.

I. Introduction

Grid Computing provide computational utility environment, where a collection of potentially heterogeneous and geographically remote computer systems are connected together in a distributed fashion to share their perspective resources and present the appearance of a single large and powerful virtual computer system as illustrate in Figure 1. Grid computing is often more effective in terms of system utilization efficiency than traditional computing environment given that many computers often significantly underutilized much of the time. In addition, the Grid has attracted researchers as an alternative to supercomputers for high performance computing. Hence, Grid computing taken as the next generation IT infrastructure that promises to transform the way organizations and individuals compute, communicate and collaborate. In a grid based distributed computing environment, where several autonomous systems are interconnected, equal load distribution is main concern. The traffic on particular machine could be large at a moment while on the other end, it could be negligible. A high degree of parallelism can be achieved if different tasks run on different machines simultaneously [1]. The problem of Load balancing came into the limelight as soon as the concept of multiprocessor as well as multi computation system architecture was proposed. Since there is multitude of resources in a Grid environment, convenient utilization of resources in a Grid provides improved overall system performance and decreased turn-around times for user jobs [2]. A major drawback in the search for load balancing algorithms across a Grid is the lack of scalability and the need to acquire system-wide knowledge by the nodes of such a system to perform load-balancing decisions [3].

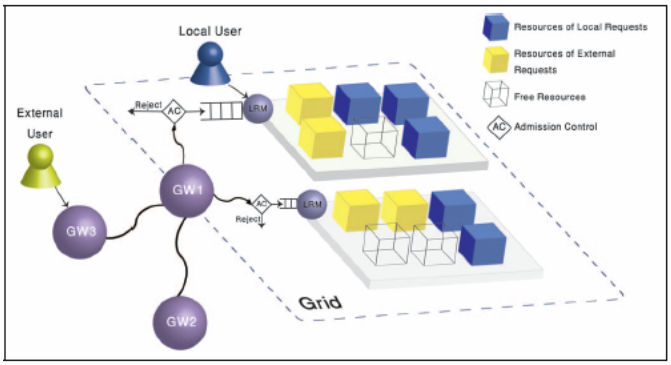

Figure 1. Grid Environment

The organization of the paper is as follows: In Section 2, an introduction to various methods of performing load balancing for a grid is given. In section 3, the dynamic load balancing policies for heterogeneous resources are considered. In Section 4, the different metrics for comparing the load balancing algorithms are identified and compared through tabular form. Finally, a conclusion arrived.

II. Load-Balancing Strategies

Load balancing algorithms are based on the intuition that, for better resource utilization, it is desirable for the load in a

grid system to be balanced evenly. Thus, the basic Load balancing Algorithms on basis of strategies, briefly categorized as:

I. Static Algorithms

Static algorithms use only information about the average behavior of the system, ignoring the current state of the system, i.e., scheduling possibly carried out according to a predefined approach only. Load balancing decisions are made deterministically or probabilistically at compile time and remain constant during runtime. Because the static approach cannot respond to a dynamic runtime environment, it may lead to load imbalance on some nodes and significantly increase the load balancing time [4].

II. Dynamic Algorithms

Dynamic Algorithms react to the system state that changes dynamically. Thus, a dynamic approach can be made adaptive to changes in system parameters such as job arrival rate, CPU processing rate, loads and communication bandwidth between computers. However, as computers may have stale, incorrect information about the state of other computers, it is predestined that poor scheduling decisions maybe made.

In terms of static and dynamic load balancing, a hybrid load balancer attempts to combine the merits of static and dynamic load-balancing algorithms and, by doing so, minimizes their relative inherent disadvantages [5]. But Decentralized algorithms are more scalable and have better fault tolerance.

I. Centralized dynamic Algorithms

In centralized load balancing algorithms, the parameters necessary for making the load balancing decision are collected at, and use by, a single resource (decision maker) i.e., only one resource acts as the central controller and the remaining act as slaves [6]. A global view of the load information in the system allows decision making on how to allocate jobs to the resources participating. Nevertheless, it may result into communication overhead, also being single-armed controller consequences into bottleneck and single point of failure. Hence, whole system may crumple due to single failure. Therefore, this approach is not scalable for vast environment like Grid.

II. Decentralized dynamic Algorithms

In decentralized algorithm, there is not usually a specific node as controller. Instead, all nodes have knowledge about some or all other nodes, thereby, resulting onto huge communication overhead. Furthermore, this information is not very reliable because of the drastic load variation in the Grid and the need for frequent updating [7].

I. Hierarchical Algorithm

In case of Hierarchical algorithm, load is balanced starting from leaf node to root node. All the computing nodes are distributed in tree from where leaf node compute the user resource requirement and in case if imbalance in the grid system, load is balance by the superior node of the leaves and transfer to neighborhood if required. Hierarchical algorithms are always scalable as any computing node can be added as leaf any time and also fault tolerance.

III. Load balancing Policies

The four policies that govern the action of a load-balancing algorithm when a load imbalance is detected deal with information, transfer, location and selection.

I. Information Policy

The focus is mainly to specify what workload information to be collected, when it is to be collected and from where. While balancing the load, certain type of information such as the number of jobs waiting in queue, job arrival rate, CPU processing rate, and so forth at each processor, as well as at neighboring processors, may be exchanged among the processors for improving the overall performance. Information unruffled eventually as:

I. On-Demand Information Collection

Decentralized approach, collects host information only when host becomes either sender or receiver.

II. Periodic Information Collection

Either Centralized or Decentralized approach, information collected at fix time interval.

III. State-Variation driven Information Collection

Either Centralized or decentralized approach, information collected whenever there is change in states of the resource nodes.

II. Transfer Policy

Transfer policy determines the condition under which a task should be transferred. A typical transfer policy includes task migration and/or task re-scheduling. Migration is suspending an executing task, transferring it to another processor and resumes its execution from the state of suspension. On the other hand, re-scheduling involves only the transfer of tasks which have not started their execution yet. So migration is preemptive where re-scheduling is non-preemptive in nature.

Most of the proposed transfer policies are based on “threshold”. Defining a suitable threshold value for a particular computing environment is challenging task. Thresholds are generally expressed in terms of units of loads. Load of a host can be expressed in various ways, for example

the present CPU queue length of a host can be used as a load index for the associated host. Whenever a new task is submitted at any host, the transfer policy updates the local load information and tries to define the host as a 'sender' or a 'receiver' on the basis of the predefined threshold value(s).

III. Selection Policy

Once the transfer policy decides that a host is a sender, a selection policy selects a task for transfer. The simplest and popular approach is to select the newly arrived task for transfer that just transforms the host into a sender. Transferring such a task is relatively cheap, since the transfer is effectively becomes non-preemptive.

IV. Location Policy

Responsibility of the location policy is to find a suitable "transfer partner" (sender or receiver) for those hosts, which are already marked as sender or receiver by the transfer policy. One of the major tasks of location policy is to check the availability of the service(s) required for proper execution of the migrated and/or re-scheduled task(s) within the selected transfer partner. Hence the broad classifications of location policies are: sender initiated, receiver initiated, symmetrically initiated.

IV. Metrics

The traditional objective, when balancing sets of computational tasks, is to minimize the overall execution time called makespan.The communication cost, induced by load redistribution, is also a critical issue.

TABLE I. COMPARTIVE STUDY OF LOAD BALANCING ALGORITHMS CONSIDERING VARIOUS METRICS

Metrics	**Static**	**Dynamic**	**Hirerarchical**
Nature	Static	Dynamic	Bottom-Up
Communication Overhead Associated	Less overhead	Overhead associated is high and increases if dynamic decentralized considered	Hierarchical overhead(increases as going from bottom to up)
Utilization of resources	Not good as deterministic or probabilistic allocation at compile time	More utilization	Good utilization
Preemptiveness	Non-preemptive	Preemptive and non-preemptive	Preemptive and non-preemptive
Predictability	More predictable	Less predictable	Less Predictable
Adaptability	Less Adaptive	More adaptable	Adaptable
Scalability	No possible as Static Grid Environment	Possible	Possible
Response Time	Less	More	More
Reliability	Less	More	More
Stability	More	Less	Less
Robustness and flexibility	No	High	High

IV. CONCLUSION

Through comparative study of each load balancing algorithm, considering almost every metric, authors came to the conclusion that using hierarchical load balancing will be quite feasible looking metric criteria like communication overhead, scalability, adaptability, etc. Therefore, Hierarchical approach is very suitable for grid resource and operational manager.

References

[1] Sachin Kumar and Niraj Singhal, "A Priorty based Dynamic Load Balancing Approach in a grid based Distributed Computing Network",*International Journal of computer Applications*(0975-8887),Volume 59-No.5,July2012.

[2] Arora, M., Das, S., K., "A De-centralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments.", Proc. of Int. Conf. Parallel Processing Workshops, 499, IEEE, Washington,2002.

[3] Resat Umit Payli, Kayhan Enricyes and Orhan Dagdeviren, "Cluster-Based Load Balancing Algorithms for Grids", International Journel of Computer Networks & Communication(IJCNC) Vol.3,No.5,Sep 2011.

[4] K. Lu, R. Subrata, A. Zomaya, "An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems considering Desirability of Grid Sites", 25th IEEE International Conference on performance, Computing and Communication, April 2006.

[5] Riky Subrata, Albert Y. Zomaya, and Bjorn Landfeldt, "Game-Theoretic Approach for Load Balancing in Computational Grids," IEEE Transactions on Parallel and Distributed Systems, vol. 19, no. 1, 2008.

[6] Said Fathy El-Zoghdy, "A Hierarchical Load Balancing Policy for Grid Computing Environment", I.J.Computer Network and Information Security,June 2012.

[7] Belabbas Yagoubi and Meriem Meddeber, "Distributed Load Balancing Model for Grid Computing", Revue ARIMA,vol. 12,pp 43-60,Sept 2010.