AGENT-BASED GRID COMPUTING LOAD BALANCING AT APPLICATION LEVEL

Husam Ali Abdul Mohsin Alaa Hassan Harif
University of Baghdad - College of Science
Computer Science Dept.

Abstract

Grids functionally combine globally distributed computers and information systems for creating a universal source of computing power and information. Grid is a very large scale, generalized distributed **network computing** system that can scale to Internet-size environments with machines distributed across multiple organizations and administrative domains. The emergence of new variety applications demand that Grids support compatible efficient data and resource management mechanisms, This paper provides an **application level load balancing** for individual parallel jobs for grids data, implemented by using Borland JBuilder 7 Enterprise - WebLogic Edition.

Key words: Grids, Network Computing, Application Level, Load balancing.

المستخلص

شبكة الحواسيب تجمع عمل حاسبات موزعه في جميع أنحاء العالم مع أنظمة المعلومات لخلق مصدر يشمل القوة الحسابية مع المعلوماتية. شبكة الحواسيب واسعة الحجم، يكون شبكة نظام حسابية موزعه، قادرة على التوسع بكبر حجم بيئة الانترنت مع مكائن موزعة في عدة منظمات و في عدة مجالات حكومية. نشوء تطبيقات متنوعة جديدة يتطلب من شبكة الحواسيب أن تحتمل معلومات فعالة و لديها ميكانيكية إدارة المصادر، هذا البحث يعطي مستوى تطبيقي يوزع بها حمولة الأعمال التي تأتي فرادا و في نفس الوقت، و التي تطلب معلومات من شبكة الحواسيب، و هذا البحث طبق باستخدام "
Borland JBuilder 7 Enterprise - WebLogic Edition "

1. Introduction

The computational grid is a promising platform that provides large resources for distributed algorithmic processing. Such platforms are much more cost-effective than traditional high performance computing systems. However, computational grid has different constraints and requirements to those of traditional high performance computing systems. To fully exploit such grid systems, resource management and scheduling are key grid services, where issues of task allocation and load balancing represent a common problem for most grid systems.

The load balancing mechanism aims to approach an equal spread of the load, on each computing node, maximizing their utilization and minimizing the total task execution time. In order to achieve these goals, the load balancing mechanism should be 'fair' in distributing the load across the computing nodes. This implies that the difference between the heaviest-loaded node and the lightest-loaded node should be minimized [4]. In this paper, we propose an applicable algorithm that uses agent technology to implement scalable load balancing on Grid environments; so workload and resource management are two essential functions provided at the service level of the Grid software infrastructure. To improve the global throughput of these environments, effective and efficient load balancing algorithms are fundamentally important. Most strategies were developed in mind, assuming homogeneous set of resources linked with homogeneous and fast networks. However for computational Grids we must address main new challenges, like heterogeneity, scalability and adaptability.

This paper addresses the issue of load balancing grid resources in a reliable applicable algorithm. Sections 2, describe the related works in this approach. Sections 3, describe the background and the details of the proposed effective algorithm. In sections 4 and 5 we describe the future works and conclusion.

2. Related Work

Availability of grid resources is dynamic, which raises the need to develop robust and effective applications against changing circumstances. This paper proposes a robust load balancing applicable algorithm and the control task scheduler is discussed. Based on improving fault tolerant service on request chain processing, the paper enhances ORBUS1.1 software with a load balancing service. Simulation results show that the implementation is able to obtain high system performance with control task scheduling and MAX_PERFORM policy [1]. Load balancing is a key concern when developing parallel and distributed computing applications. The emergence of computational grids extends this problem, where issues of cross-domain and large-scale scheduling must also be considered. In this work an agent-based grid management infrastructure is coupled with a performance-driven task scheduler that has been developed for local grid load balancing. Each grid scheduler utilizes predictive application performance data and an iterative heuristic algorithm to engineer local load balancing across multiple processing nodes. At a higher level, a hierarchy of homogeneous agents is used to represent multiple grid resources. Agents cooperate with each other to balance workload in the global grid environment using service advertisement and discovery mechanisms. A case study is included with corresponding experimental results to demonstrate that both local schedulers and agents contribute to overall grid load balancing, which significantly improves grid application execution performance and resource utilization [2]. To be mentioned that the applicable algorithm proposed in this paper works in the local nodes in the same grid rather than the global grid environment.

3. Grid Load Balancing

This section describes issues related to our applicable algorithm. This algorithm mainly consists of a GBalancer in each node that manages system related tasks and nodes that join the Grid and provide resources. A GBalancer role in the system is to manage tasks, including maintenance of load balancing, schedule the status of each node in the Grid, selecting nodes for task execution and to participate in the load balancing operation. To make a GBalancer accomplish its mission in the most efficient way, a GBalancer has the mechanisms of calculating the most effective node. Each GBalancer in each node in grid environment sends messages describing its status to each GBalancer at other nodes in the grid in a fixed period of time, so each node in the grid will have a status table describing the status of all the nodes in the grid. Now, if any node in the grid is over loaded in that moment and has exceeded the CPU load threshold of the algorithm(in our case, the threshold is 75% of the CPU usage), that node can chose the best node in the grid that can load balance its load with, using the status table. When the loaded node chooses the idle or low loaded state node, a node that can provide its resource, a "join" message will be sent from the loaded node to the unloaded one, after the join is accepted, related hardware information will be transmitted to the GBalancer of the loaded node and the process of the load balancing will be launched by using the COBRA approach in the JAVA language, else, when the unloaded node can no longer provide resource, it has to transmit an "exit" message to the GBalancer of the loaded node, that in turn will search for other nodes to load balance with. The process of calculating the nodes load, is measured by simple static load methods by calculating CPU LOAD, MEMORY UTLIZATION and other factors specified by the administrator [3], but before that, we have to perform the isreachable function, that returns the true value when the node is

online, and a false value if the node is offline. The process of static load balancing stage is described as:

I. When a request for executing a task is proposed, GBalancer dispatches an agent to collect related information of each node participating in this Grid by receiving a broad casting message from each node.
II. Agents collect related information of nodes used in the load balancing method [3].
III. Nodes sent calculated information to agent.
IV. Agent provides all node related information to GBalancer as a table generated by MS-Access connected with Java agent code using ODBC Driver.
V. GBalancer builds a table of effective nodes by mechanism of calculation of effective nodes (lowest load).
VI. GBalancer selects effective nodes set from a table of effective nodes by mechanism of node selection (by SQL selection method using java compiler).
VII. GBalancer assigns subtasks to selected nodes, using the COBRA mechanism.

In a Grid environment, the effectiveness of nodes may vary with time. Thus, the assignment of tasks has to be dynamically adjusted in accordance with the fluctuation of node status. The variation of the node status can be identified in condition by checking the status of each node at threshold time (in our system at each 3 minutes), when the GBalancer receives the message that a certain node can no longer provide resources "exit". So the process of dynamic load balancing stage, as follows:

I. When state of node changes, GBalancer dispatches agent to collect related data of each node in the table of effective nodes, for time consuming.
II. Agent collects related information of node in the table of effective nodes, by sending a message to that node requiring its status.
III. Nodes sent their own information to agent.
IV. Agent provides collected data to GBalancer. GBalancer compares the collected data with historic data in order to confirm if the node is still effective and to see if there is a dramatically change in the load of the overall grid.
V. If the node is confirmed ineffective, a node with the highest value will be selected from the wait aggregate, and the subtask is re-executed by new effective node.

Therefore, by using the previous stages in our system these will provide the ability to distribute the load of jobs through the grid environment with high performance.

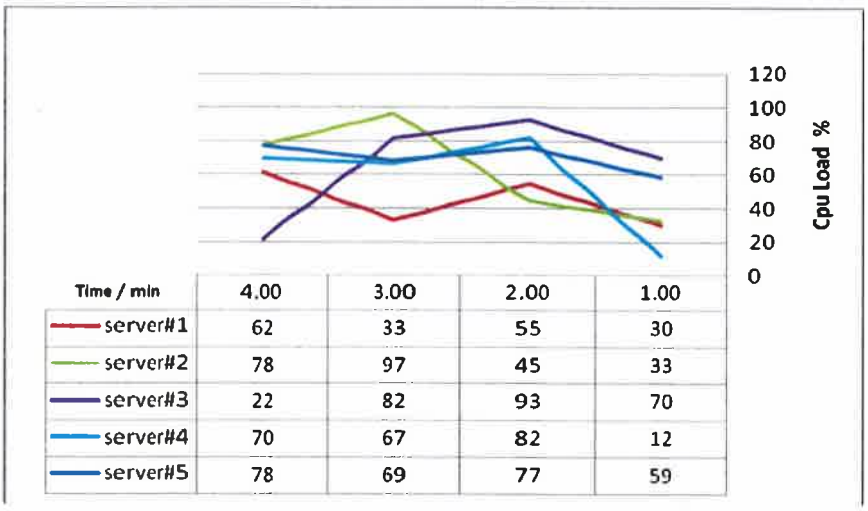
4. Experimental Results

This experiment is achieved according to the specifications of our simulated environment. This simulated GBalancer environment includes heterogeneous servers that are loaded with jobs. The number of servers tested in this simulated environment are 5 servers, each with a different operating system, and hardware specifications as shown in table1.

Server name	Memory size	CPU speed	Operating system
Server#1	2G	2.3Gz	Windows xp
Server#2	4G	2.6Gz	Server 2003
Server#3	2G	2.6Gz	Windows7
Server#4	4G	2.3Gz	Server 2000
Server#5	4G	2.3Gz	Windows xp

Table 4.1: Shows The Simulated Heterogeneous Environment

Figure 4.1 shows the status of the experimental environment before performing the load balancing procedure. The figure below shows that the load of the 5 servers are limited between 10% and 96% of the CPU usage. At that point, the system needs to be balanced, to distribute the load on the overall system, and to reach for a stable status and limiting the load of the servers in the Grid between 45% and 75% of the CPU usage. Figure 4.2, show the status of the Grid after performing the load balancing. It is obvious, that the load average of the overall system CPU usage reached to the status this research is seeking for.



Time / min	4.00	3.00	2.00	1.00
server#1	62	33	55	30
server#2	78	97	45	33
server#3	22	82	93	70
server#4	70	67	82	12
server#5	78	69	77	59

Figure 4.1 below shows a line chart that sketches the information of the table below, those information represent the CPU load of each node in our heterogeneous environment in each time interval.

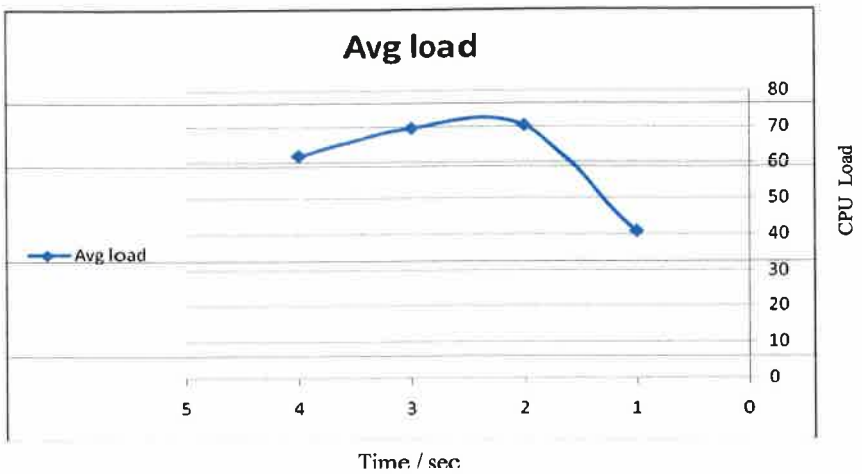


Figure 4.2 below shows a line chart that sketches the average CPU load of our heterogeneous environment in each time interval.

5. Conclusions

We presented our architecture to design Grid Computing Services. We also proposed a two-level load balancing algorithm, which minimizes the overall tasks response time and maximizes the grid system utilization and throughput at steady-state, by combined between the effective usage load balancing on each node for it self and load balancing the overall grid system this increase the scalability, and availability of load distribution, also the heterogeneousty of our system supported by JAVA2 compiler these increase the platform Independents in our system.

5. Future works

1- There are, of course, several open issues that need to be addressed in load balancing for Data Grids. First, the performance of our load-balancing strategy in Data Grids (global usage of storage resources) depends to some extent on data movements. Reducing

overhead incurred by data movements can ultimately improve performance of Data Grids.

2- This research proposes an Gbalancer that one of its duties is to collect information from each node in the grid and send it to all nodes, and then each loaded node will use this information to decide where to distribute its load, in this case more than one node can distribute its load to the same unloaded node, that can cause some temporary saturation to that node, not mentioning that this strategy will not disserve the system as shown in the results, nevertheless , we can add a new strategy to enhance this algorithm, that is, to change the time schedule of each node so that all the nodes will send their information via the GBalancer in different times, in order for each to distribute their load to different nodes.

References

[1]. A Robust Load Balancing Pattern for Grid Computing, Junling Wang, Yun Wang Department of Computer Science, Southeast University, Nanjing, 210096, China {wjl007, yunwang}@seu.edu.cn

[2]. Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling Junwei Caol, Daniel P. Spooner†, Stephen A. Jarvis†, Subhash Saini and Graham R. Nudd C&C Research Laboratories, NEC Europe Ltd., Sankt Augustin, Germany Department of Computer Science, University of Warwick, Coventry, UK,NASA Ames Research Center, Moffett Field, California, USAEmail address for correspondence: cao@ccrl-nece.de

[3]. Web Load Balancing By Routing Agents, Samer Sami Hassan,University of Baghdad , College of Science, Computer Science Dept. ssami@scbaghdad.com

[4]. Artificial life techniques for load balancing in computational grids, Riky Subrata, Albert Y. Zomaya, Bjorn Landfeldt Advanced Networks Research Group, School of Information Technologies, University of Sydney, NSW 2006.