Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

GRID COMPUTING ALGORITHM USING PRIORITY SCHEDULER

Vipin Rai

IIMT College of Engineering, Greater Noida, U.P.

Abstract—Grid Computing has evolved as a vital field focusing on sharing of resources. Efficient scheduling of tasks is most challenging issues in Grid Computing. Load Balancing is a technique to improve utilization of resources increasing throughput managing, parallelism and to reduce the time taken for response through proper distribution of the jobs. Generally there are three type of phases related to Load balancing i.e. Information Collection, Decision Making, Data Migration. Through this paper, we are proposing a Load balancing algorithm for optimal scheduling. It schedules the task by reducing completion time and by rescheduling waiting time of each job to obtain load balance. This algorithm works on providing optimal solution so that it minimizes the execution time and expected price for the execution of all the jobs in the grid system. Load balancing algorithms are basically of two types, static and dynamic. Our algorithms in this paper based on dynamic nature load balancing.

Keywords—Execution Cost, Resource Monitoring, Computational Grid, Load balancing, and Priority scheduler. ELB: Enhanced Load Balancing

I. INTRODUCTION

The enhanced development in computing resources has increased the performance of computers and decreased their costs. This combination of low cost powerful computers along with the constantly increasing popularity of the Internet and high-speed networks has led to computing environment being mapped from distributed to Grid environments [1]. In fact, recent researches on computing architectures have accepted the emergence of a new computing paradigm known as Grid Computing. Grid is a kind of distributed system that supports the sharing and coordinated use of geographically distributed resources, independently from their physical type and location, In dynamic virtual organizations that share the same goal of solving large-scale applications. In order to meet the user expectations in terms of performance and efficiency, efficient load balancing algorithms are required in the Grid system for the distribution of tasks. A load balancing algorithm attempts to reduce the response time of applications submitted by user for ensuring maximal utilization of available resources. The main aim is to prevent, the condition where some processors are overloaded with a set of tasks while others are lightly loaded or even idle [2]. Although problem of load balancing in conventional distributed systems has been deeply studied, still new challenges in Grid computing make it an interesting topic and many research projects are under way to manage the same. This is due to the diverse characteristics of Grid computing and the complex ever changing nature of the problem itself. Load balancing algorithms that were used in classically distributed systems, usually run on homogeneous and dedicated resources, but it cannot work well in the Grid architectures. In Grid computing, individual users can retrieve computers and data available transparently, without taking into consideration the location, operating system, account administration, and other details. In Grid computing, the details are reckoned, and the resources are virtualized. Grid Computing has to enable the job in question to be run on an idle machine, than elsewhere on the network [6] The main job of grid computing is to allocate resources for a process; i.e., allocating of tasks to various resources. For example, mapping of 200 tasks into 20 resources produces 20^100 possible mappings.

This is because every job can be mapped to any of the resources. In this case the mapping of jobs is in terms of reallocation which means depending on the status of resources on either it is heavily loaded or not. Here resource means processors which are involved in the process of scheduling. We use resources and processors along side. The other complexity of resource allocation is the lack of exact details about the status of the resources.
Before the scheduling of jobs in the grid environment, the nature of the grid should be taken into account. Some of the characteristics of the grid include

- Geographical distribution : In this the resources of grid may be located at different places
- Heterogeneity: A grid consists of software as well as hardware resources that may be software components, display devices, sensor programs, scientific instruments, computers, files, supercomputers networks etc.
- Resource sharing: Various different organizations may own the resources of the grid Multiple administrations: Each and every organization may establish various security and administrative protocols to access their resources
- Resource coordination: to get connected computing capabilities, there must be coordination among Grid Resources [4]. Scheduling is highly perplexed by the heterogeneous ownership of the grid resources as Load balancing algorithm are basically of two types 1) static and 2) dynamic. In static scheduling, all the information regarding the jobs and resources such as execution time of the job, execution speed of the processor are available by the time the application is scheduled. There for in this Scheduling, it is easy form the scheduler's point of view to program.

But in the case of dynamic scheduling, the execution time of the tasks may not be known due to the number of iterations in the loop, direction of branches etc. So, the task has to be allocated on the runway as the application begins to execute. Both static and dynamic scheduling is majorly adopted in the grid. Here, system may not be aware of the run time behavior of the application before it begins to execute and dynamic load balancing algorithms distributes the tasks among workstations at run-time. Grid uses current or recent load information when making decisions regarding distribution of tasks. Multi computers with dynamic load balancing allocate and reallocate resources at runtime depending on a priori task information, which determines when and whose tasks can be migrated [7]. As a output, dynamic load balancing algorithms can provide a major improvement in performance over static algorithms. But, this compromises additional cost of collecting and maintaining load information, so it is important to keep these overheads within set limits that are reasonable [8]. There are three major parameters which usually define the strategy on how a specific load balancing algorithm will be employed [9].

These three parameters provide us with the answer of three important questions:

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

II. PROBLEM DEFINITION

In real time grid environments, the resources being shared are dynamic in nature, which in turn severely affects application performance. Resources and Workload management are two vital functions given at the base level of the Grid infrastructure of the Software. For the betterment of the global output of these environments, efficient load balancing algorithms are vital. The aim of our study is to take into consideration the factors which can be used as features for decision making to initiate Load Balancing. Load Balancing is one of the most vital factors which can initiate the

throughput of the grid application. This thesis work considers the existing Load Balancing modules and tries to locate performance hindrance in it. All Load Balancing algorithms depend upon implementation of five policies [7]. The efficient implementation of these policies decides overall performance of Load Balancing algorithm. The main objective of this thesis is to propose an efficient and improved Load Balancing Algorithm.

Algorithm for Grid environment:

- The Main difference among existing Load Balancing algorithm and proposed Load Balancing (ELB Load Balancing Algorithm) is in implementation of three policies: Information Policy, Triggering Policy and Selection Policy.
- In order to implement Information Policy all existing Load Balancing algorithm are using periodic approach, which is in turn time consuming. The proposed approach focused on using activity based approach for implementation of Information policy.
- In order to implement Triggering ELB Load Balancing algorithm is working on two parameters that calculates Load Index. On the basis of the Load Index, Load Balancer then decides on when to activate Load Balancing process.
- In order to Implement Selection Policy ELB algorithm uses Job length as a parameter, which is more trustworthy to make decision about selection of task in order to shift from heavily loaded node to lightly loaded node. As a outcome the following things have been found in this paper:
- Studying the existing Load Balancing algorithm for a Grid environment
- An ELB Load Balancing algorithm has been proposed and executed in the simulated Grid environment.

Motive of this thesis is to design and develop a performance efficient Load Balancing algorithm which overcomes the shortcomings of the current state of the art in the context.

III. PROPOSED ELB LOAD BALANCING ALGORITHM

Load balancing is defined as the allocation of the work of a single application to processors at run-time so that the execution time of the application is minimized. This chapter is going to discuss the design of proposed Load Balancing algorithm.

3.1 Background

Choosing a load balancing algorithm for a Grid environment is never an easy task.
Various algorithms have been proposed in the Literature and all of them differ based on certain specific application domain. Some load balancing strategies work well for short and quick tasks while other works well for applications with large parallel tasks, some strategies focuses on managing data-heavy tasks, while others are more suitable for parallel tasks that are computation heavy. Since various different load balancing algorithms have been proposed, there are four basic steps that nearly all of them have in common:

- Monitoring workstation performance (load
- monitoring)
- Exchanging information among workstations (synchronization)
- Calculation of new distributions (rebalancing
- criteria)
- Actual movement of data(job migration) Efficient Load Balancing algorithm makes Grid Middleware efficient which ultimately leads to fast execution of application. In this work, an

attempt has been made to create a decentralized, load balancing algorithm initiated by sender for Grid environments which depends upon different parameters. One of the most important features of this algorithm is to estimate system parameters such as queue length and CPU utilization of each node and to migrated job if required.

IV. DESIGN OF ELB LOAD BALANCING ALGORITHM

Load balancing needs to take place when the situation of load has changed. There are activities that change the load configuration in Grid environment. The activities can be categorized as following:
Incoming new job and its queuing

- Completion of job.
- Incoming new resource
- Withdrawal of existing resource.

Whenever any of these four activities takes place It is communicated to master node and then load information can collect to check load balancing condition. If load balancing condition meets the requirements then actual load balancing activity is performed.
Following is the proposed algorithm for ELB Load Balancing:

4.1 Proposed Algorithm For Resource Allocation In Grid Computing Priority Schedular (P S) Model:

Suppose A is the number of process in process queue 'Prq'and is indicated by- Pr1 Pr2 … Pra
These Processes are then allocated to B number of Resources queue 'Req' resources queue- Re1 Re2 …. Ren
Resource queue is then maintained as per the priority of the resources. The resources which are having a low loading factor have assigned a higher priority in importance and vice versa which have high loading factor have assigned a lower priority. Then we can find the overall process execution cost in terms of time-
Let t1, t2, t3……..ta are the time of execution for individual process.
Let T(Pri, Rej) be the total cost for ith process in jth resources can be calculated as :
n m n **Σ Σ T(Pri, Rej)=Σ Σ ti×PN+ CT**
I=0 J=0 I=0 J=0 Where ti is the execution time of process
PN= Priority Number
CT= Communication Time

Load factor of a given resource is calculated as:
Algorithm for finding load factor:
In this algorithm let us assume that all recourses are free and we start ELB algorithm –

4.2 Function Start-

Load Factor ()
{
// **Initialize load factor algorithm**
Step 1. Initialize pro_queue 'Prq'
And re_queue 'Req'
Step 2. Start LOOP
For every resources i=1 to b
Step 3. Initialize load factor of every resource
Rei =0
Step4. Initialize priority for every resource = 1

Now insert the resources in a pr_queue
End of LOOP
Step5. Do: every process in pr_queue
'Prq'
Step6. Assign process to the resources which is in the "Begining" of the priority queue
Step7. Update the priority in 'Req'
SET Rp=Rp+1
Step8. On Successful Completion of the process execution then the priority of resource decrease by 1
By Rp=Rp-1
End of the for loop
}

In this ELB load factor algorithm we set the load factor of each resource to zero and maintain priority of each resource to one because there is no process for execution at that time. As the process arrives to the load, load factor of resources is increased and priority of resources keep on decreasing i.e. the right arrow indicate increase of load factor and left arrow indicate the decrease of priority. This can be shown in the following figure 5.1

Pseudo Code Related To Load Balancing
Algorithm Priority Scheduler (PS) –
Function LB_Start
{
Start
Call Ld_Factor ()
Step1. If (Priority of resources is Max and CPU queue length is Max)
Then load factor is automatically max Heavily load_Res
Else if
Step2. If (Priority of resources is min and CPU queue length is min)
Then ld_factor() is min
Lightly load_Res
Else if
Step3. Shift the process from heavily load_res to lightly load_res
End of the algo
}

4.3 Functions Used In The Above Algorithm Are:-

Cond_occurs ():

Work of this function is to return Binary value 0 and 1. In case any of above stated condition is true it returns 1else it returns 0.

LdBal_Begin ():

The work of this function is also to return Binary Value depending on the basis of given parameters (Resource priority and queue length) will be required it will return 1 else it will return 0. This function also updates two lists:
HeavilyLoad_Res list and LightlyLoad_Res list.

In the proposed ELB algorithm information is collected only if configuration of grid is changed. This information is used to perform load balancing.
Above Mentioned is the flow diagram depicting the same? First of all it starts different parameters. Whenever any of four activities are required to start information Policy, it starts collecting load balancing information. Once information has been gathered then it is works out on if load balancing is required or not. For this purpose application uses queue length parameters and CPU utilization. With these parameters we decide on which resource is heavily loaded and which resource is lightly

loaded. Once selection of resource is complete, the application selects job out of n-jobs running on that resource. This selection is based upon on CPU consumption of different jobs. Jobs consuming least amount of CPU will be selected for migration. Once job is selected, application checks for available lightly loaded resource. In case lightly loaded resource is available then it migrates selected job from heavily loaded resource to lightly loaded resource. If no lightly loaded resource is available then it puts selected job to pending job list or pending queue. This job will going to be executed later when certain lightly loaded resource will be available. Finally all the data will be updated in database. Also, it supports check pointing, in where a disk image of an active process is stored in order to facilitate both process migration and fault tolerance.

V. CONCLUSION

We have hereby presented a new design of scheduling algorithm Priority Scheduler. Our proposed ELB Priority scheduler completed a task by applying highly utilized low in cost resources with minimum computational time. Our algorithm uses the priority queue to achieve a higher throughput. This algorithm performs better for task in real environment. However, in all situations, the proposed algorithms perform better then the some existing ones as it needs to be tested further. But performance in grid application still remains a challenge in dynamic grid environment. Resources from here can be submitted and can be withdrawn from Grid at any moment. This feature of Grid makes Load Balancing one of the most critical features in Grid infrastructure We here implemented the above algorithm by using Gridism toolkit. We can also further hybrid the Priority Scheduler with any evolutionary scheduling algorithm like Genetic algorithm technique to achieve a high throughput and high resource utilization.

In this thesis we have aimed 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. Every Load Balancing algorithm depends upon implementation of five policies. The efficient implementation of these policies decides performance of Load Balancing algorithm. In this work we analyzed existing Load Balancing algorithm and proposed an ELB enhanced algorithm which more efficiently implements three out of five policies mentioned above implemented in existing Load Balancing algorithm.

These three policies are: Information Policy, Triggering Policy and Selection Policy. To be noted that Proposed ELB algorithm is executed in simulated Grid environment.

VI. FUTURE DIRECTIONS

The work performed in this Project can be used as the basis for an improved load balancing module in Condor. A further extension to this work would be in making this Load balancing Module a middleware independent module.

REFERENCES

[1] Clovis Chapman1, Paul Wilson2 (2004) 'Condor services for the Global Grid: Interoperability between Condor and OGSA' Proceedings of the UK e-Science All Hands Meeting, ISBN 1-904425-21-6, pages 870-877, Nottingham, UK.

[2] Jean-Christophe Durand (2004) 'GridComputing a Conceptual and Practical Study'Proceeding of the Fifth IEEE Workshop on Grid Computing (GRID'04).

[3] I. Foster, C. Kesselman, and S. Tuecke. (2001) 'the anatomy of the Grid: Enabling scalable virtual organization' International Journal of Supercomputer Applications.

[4] Kai Lu, Riky Subrata and Albert Y. Zomaya(2002) 'An Efficient Load Balancing Algorithm for Heterogeneous Grid Systems considering Desirability of Grid Sites'Sydney, Australia.

[5] Rajkumar Buyya (1999) 'Grid Computing and distributed Systems (GRIDS) Lab', Australia.

[6] L. Xiao, Z. Xu. and X.Zhang (2003) 'Lowcost and reliable mutual anonymity protocols in peer-to-peer networks' IEEE Transactions on Parallel and Distributed Systems, 14(90): 829-840.
[7] Ratnesh Kumar Nath, Seema Bawa, Inderveer Chana (2007) 'Load Balancing Issues in Grid Environment' National Seminar on Recent Advancement in Information Technology (RAIT) organized by Indian School of Mines, Dhanbad .
[8] F. Azzedin and M. Maheswaran (2002)'Evolving and managing trust in grid computing systems' In *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pages 1424-1429.*
[9] Ratnesh Kumar Nath, Seema Bawa, Inderveer Chana (2007) '"Load Balancing in Grid Environment: Strategies, Algorithms and Policies' National Symposium on Security and Soft Computing (NSSC) organized by National Institute of Technology, Surat.
[10] I. Foster, et al (2002) 'The Physiology of the Grid: An Open GridServices Architecture for Distributed Systems Integration' GlobusResearch Work in Progress.
[11] Moreno Marzolla, Paolo andreetto, Valerio Venturi, Andrea Ferraro, and Shiraz Memo net al. (2007) 'Open standards based interoperability of job submission and management interfaces across the grid middleware platform' In Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, Pages 592-601.
[12] Heinz Stockinger (2007) 'Defining the grid: a snapshot on the current view' The Journal of Supercomputing,
[13] M. Dobber, R. Mei, and G. Koole, "Dynamic Load Balancing and Job Replication in a Global-Scale Grid Environment: A Comparison", IEEE Transaction on Parallel and Distributed Systems, Vol. 20, no. 2, pp. 207-218, February 2009.
[14] Junwei Cao "agent based using performance driven task scheduling" C&C Research Laboratories, NEC Europe Ltd., Sankt Augustine, Germany IEEE.
[15] Shah, Veeravalli, "On the Design of Adaptive and Decentralized Load-Balancing Algorithms with Load Estimation for Computational Grid Environments" IEEE Transactions on Parallel And Distributed Systems, 2007.
[16] "A Hybrid Load balancing Strategy of Sequential Tasks for Computational Grids" Yajun Li, Yuhang Yang Rongbo Zhu Electronic Engineering Department Shanghai Jiaotong University, China 2009.
[17] Rajkumar Rajavel, Kongu Engineering College, Perundurai, Erode, India. December, 2010 "Decentralized Load Balancing for the Computational Grid Environment".
[18] C. Mattmann, et al.(2006) 'A software Architecture-Based Framework for Highly Distributed and Data Intensive Scientific Applications' In Proceeding of the ICSE. 2006. Czalkowski. K., Fitzgerald S., Foster I. And Kesselamn C. (2001)'Grid Information Services for Distributed Resource Sharing', In10th IEEE International Symposium on High Performance Distributed Computing, IEEE Press, 181-184.
[19] Foster, I., Kesselman, C., Nick, J.M. and Tuecke, S. (2002) 'Grid Services for Distributed Systems Integration' IEEE Computer, 35(6). 2002

