Process Resource Allocation in Grid Computing using Priority Scheduler

Mayank Kumar Maheshwari
Amity University Noida, India

Abhay Bansal,PHD
Amity University Noida, India

ABSTRACT

Grid Computing has emerged as an important new field focusing on resource sharing. One of the most challenging issues in Grid Computing is efficient scheduling of tasks. Load Balancing is a technique to improve parallelism, utilization of resources increasing throughput managing and to reduce response time through proper distribution of the tasks. Generally there are three type of phases related to Load balancing i.e. Information Collection, Decision Making, Data Migration. In this paper, we propose a Load balancing algorithm for optimal scheduling. It scheduled the task by minimum completion time and rescheduled by waiting time of each task to obtain load balance. This algorithm scheme tries to provide optimal solution so that it reduces the execution time and expected price for the execution of all the jobs in the grid system is minimized. Load balancing algorithms is of two types, static and dynamic. Our algorithms in this paper based on dynamic nature.

General Terms

Grid computing, scheduling

Keywords

Computational Grid,Load balancing, Priority scheduler, Execution Cost, Resource Monitoring.

1. INTRODUCTION

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 [6] The main task of grid computing is the allocation of resources for a process; i.e., mapping of tasks to various resources. For example, mapping of 100 tasks into 20 resources produces 20^50 possible mappings. This is because every job can be mapped to any of the resources. In our case the allocation of jobs is in terms of reallocation which means depending on the status of resources either it is heavily loaded or not. Here resource means processors which are involved in the scheduling process. We used resources and processors simultaneously. The other complexity of resource allocation is the lack of accurate information about the status of the resources.

Before scheduling the tasks in the grid environment, the characteristics of the grid should be taken into account. Some of the characteristics of the grid include

- ➢ Geographical distribution where the resources of grid may be located at distant places
- ➢ Heterogeneity, a grid consists of hardware as well as software resources that may be files, software components, sensor programs, scientific instruments, display devices, computers, supercomputers networks etc.
- ➢ Resource sharing, different organizations may own the resources of the grid
- ➢ Multiple administrations, each organization may establish different security and administrative policies to access their resources
- ➢ Resource coordination, to get combined computing capabilities, grid resources must be coordinated [4].Scheduling is highly complicated by the distributed ownership of the grid resources as Load balancing algorithm are two type static and dynamic, In the case of static scheduling, all the information regarding the tasks and resources such as execution time of the tasks, speed of the processor are available by the time the application is scheduled.

In this type of Scheduling, it is easy to program from the scheduler’s point of view. But in the case of dynamic scheduling, the execution time of the tasks may not be known due to the direction of branches, number of iterations in the loop etc. So, the task has to be allocated on the fly as the application executes. Both static and dynamic scheduling are widely adopted in the grid. Here, system need not be aware of the run time behavior of the application before execution and dynamic load balancing algorithms distributes the tasks among workstations at run-time; they use current or recent load information when making distribution decisions of tasks. 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.

To date several grids scheduling algorithms have been proposed to optimize the overall grid system performance. The study of managing resources in the Grid environment started from 1960s. The economic problem [10] results from having different ways for using the available resource, so how to decide what is the best way to use them. Job scheduling in parallel system has been extensively researched in the past. Various Load Balancing Algorithms are available now days but they contain several drawbacks likes; the use of many of these scheduling algorithms has been limited due to restriction in application designs, runtime system, or the job management system itself.

Proposed scheduling algorithm is one of the algorithms which follow the economic strategy. Aim of this algorithm, to decrease the number of jobs that doesn't meet their deadlines. The resources are priced according to their performance. This algorithm also has a facility of fallback mechanism; which can inform the grid user to resubmit the jobs again, the jobs which are not met the deadline of the available resources.

2. PROPOSED ALGORITHM FOR RESOURCE ALLOCATION IN GRID COMPUTING

2.1 PRIORITY SCHEDULAR (P S) MODEL :

Let M is the number of process in process queue 'Pq' is indicated by-

P1	P2	…	Pm

Process are allocated to N number of Resources queue 'Rq' resources queue-

R1	R2	….	Rn

Resource queue is maintained according to the priority of the resources as given below:

The resources which have low loading factor have assigned a higher priority and which have high loading factor have assigned a lower priority.

Then we can find the overall cost process execution in terms of time-

Let t1, t2, t3……….tn are the time of execution of individual process.

Let T(Pi, Rj) be the total cost for ith process in jth resources can be calculated as-

$$\sum_{I=0}^{m}\sum_{J=0}^{n} T(Pi, Rj)=\sum_{I=0}^{m}\sum_{J=0}^{n} ti\times PR+ CT$$

Where ti is the execution time of process
PR= Priority Number
CT= Communication Time

2.2 Load factor of a given resource is calculated as:

2.2.1Algorithm for finding load factor:

In this algorithm we are assuming that all recourses are free i.e. there are currently no process is running when we start Load _Factor algorithm –

2.2.1.1 Function Start-

```
Load Factor ()
{
// Initialize load factor algorithm

Step1. Initialize the process_queue 'Pq'
        And resources_queue 'Rq'

Step2. Loop Begin
        For every resources i=1 to n

Step3. Set the load factor of every resource
         is Ri =0

Step4. Set the priority of every resource is 1
        And insert the resources in a priority
       Queue
       Loop End

Step5. For every process in process queue
       'Pq'

Step6. Assign the process to the resources
        which is in the FRONT of the priority
         queue

Step7. Update the priority of resources in
        'Rq'
        SET Rp=Rp+1

Step8. When the process execute
        successfully then the priority of
       resource decrease by 1
       By SETTING Rp=Rp-1
End of the for loop
}
```

In the load factor algorithm we are setting the load factor of each resources is zero and priority of each resources is one because there is no process for execution. As the process arrives the load factor of resources increased and priority of resources decrease 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 1.

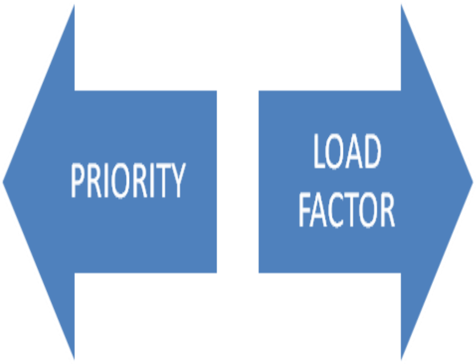

Fig.1 Shows priority VS. Load factor

2.2.1.2 Pseudo Code Related To Load Balancing Algorithm Priority Scheduler (PS) –

```
Function LB_Start
{
Start
Call Load_ Factor ()
Step1. If (Priority of resources is Maximum
         and CPU queue length is Maximum)
        Then load factor is maximum
        Heavily loaded_Resources
End if

Step2. If ( Priority of resources is minimum
         and CPU queue length is minimum)
        Then load factor is minimum
        Lightly loaded_Resources
End if

Step3. Migrate the process from Heavily
          loades_resources to lightly
         loaded_resources
        End of the algorithm
}
```

2.3 Functions used in the above algorithm are:-

2.3.1Condition_ happens (): This function return Binary value 0 and 1. If any of above defined condition is true it returns 1 otherwise it returns 0.

2.3.2Load Balancing_Start (): This function also return Binary Value on the basis of given parameters (Resource priority and queue length) load balancing will be required it will return 1 else it will return 0. This function also updates two lists: HeavilyLoaded_Resource list and LightlyLoaded_Resource list.

3. COMPARISON OF EXISTING ALGORITHMS AND PROPOSED ALGORITHMS

Table-1 Comparison Based Table

	Information Gathering policy	Firing Triggering Policy	Hitting Selection Policy
Existing Load Balancing	Load Balancing information is composed using periodic approach	Load Balancing is Triggered based on Queue Length	Task is selected for Migration using Job Length as criteria.
Proposd Load Balancing	Load Balancing is calculated based on regular monitoring of resources.	Load balancingis triggered based on queue length and priority of resources.	Task is selected for migration using CPU load.

In Condor (Existing) based algorithm, Information Policy may be fired by periodic approach while in Proposed algorithm it may be regular monitoring of resources. Triggering Policy in Existing Algorithm is based on Queue Length while in Proposed Algorithm it is based on Queue Length and priority of resources. Selection Policy in Existing Algorithm is done by Selected Task may be migrated using Job Length while in Proposed Algorithm Selection Policy may be fired by Selected task which is migrated based upon CPU load.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the design the new scheduling algorithm Priority Scheduler. Our proposed Priority scheduler completed a task by using highly utilized low cost resources with minimum computational time. Our scheduling algorithm uses the priority queue of resources to achieve a higher throughput. This algorithm is performing better for task in real environment. However, in all situations, the proposed algorithms perform better then the some existing ones.

But grid application performance remains a challenge in dynamic grid environment. Resources can be submitted to Grid and can be withdrawn from Grid at any moment. This characteristic of Grid makes Load Balancing one of the critical features of Grid infrastructure

We will implement the above algorithm by using Gridsim toolkit. we can hybrid the Priority Scheduler with any evolutionary scheduling algorithm like Genetic algorithm, Particle Swarm Optimization technique to achieve a high throughput and high resource utilization.

5. REFERENCES

[1] Sashi Tarun, Neeti Sharma Grid Computing: A Collaborative Approach in Distributed Environment for Achieving Parallel Performance and Better Resource Utilization,2011

[2]Dr.K.Vivekanandan, D.Ramyachitra, Professor, BSMED, Bharathiar University, Coimbatore, Tamil Nadu, IndiaA Study on Scheduling in Grid Environment ,feb 2011

[3] Prabhat Kr.Srivastava Improving Performance in Load Balancing Problem on the Grid Computing System, feb 2011

[4]Gregor von laszewaski, Ian Foster, Argonne National Laboratory, Designing Grid Based Problem solving Environments.

[5] Junwei Cao1, Daniel P. Spooner, Stephen A. Jarvis, and Graham R. Nudd, Grid Load Balancing Using Intelligent Agents.

[6] B. Yagoubi , Department of Computer Science, Faculty of Sciences, University of Oran and Y. Slimani , Department of Computer Science, Faculty of Sciences

Tunis, Task Load Balancing Strategy for Grid Computing .

[7] Rajkumar Buyya and S Venugopal, "A Gentle Introduction to Grid Computing andTechnologies",http://www.buyya.com/papers/GridIntroCSI2005.pdf

[8] Gregor von laszewaski, Ian Foster, Argonne National Laboratory, Designing Grid Based Problem solving Environments.

[9] Junwei Cao1, Daniel P. Spooner, Stephen A. Jarvis, and Graham R. Nudd, Grid Load Balancing Using Intelligent Agents.

[10] Nakai J. Pricing computing resources: Reading between the lines and beyond. Technical report, National Aeronautics and Space Administration, 2002.

[11] K. Somasundaram, S. Radhakrishnan, Task Resource Allocation in Grid using Swift Scheduler, Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844, Vol. IV, 2009.

[12] Hai Zhuge, Xiaoping Sun, Jie Liu, Erlin Yao, and XueChen, A Scalable P2P Platform for the Knowledge Grid

[13] Klaus Krauter, Rajkumar Buyya, and Muthucumaru Maheswaran, a Taxonomy and Survey of Grid ResourceManagement Systems

[14] Akinwunmi A.O., Aderounmu G.A., Onifade O.F.W. Enhanced Accounting Scheme for Grid Computing Architecture International Journal of Computer Applications (0975 – 8887) Volume 37– No.9

[15] Manoj Kumar Mishra, Prithviraj Mohanty, G. B. Mund A Time-minimization Dynamic Job Grouping-based Scheduling in Grid Computing International Journal of Computer Applications (0975 – 8887) Volume 40– No.16