Volume 2, Issue 1, January 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

INTER-INTRA FAIRNESS SCHEDULAR FOR RESOURCE MANAGEMENT IN GRID ENVIRONMENT

Navjeet Kaur[*], Harpal Kaur[2], Satinder Pal Ahuja[3]
[*]*M.Tech. (Computer Science & Engineering)*
IGCE, Punjab Technical University, Kapurthala – 144601 (Punjab)
[2]*Asstt.prof (Computer Science & Engineering), IGCE,*
[3]*Associate Professor & HOD (CSE&IT), Indo Global College of Engineering*

Abstract: ***Resource management is a vital task of grid computing environment. It is the responsibility of grid system to ensure that all applications/clients/tasks requesting for resources are getting resources in a timely manner. Various recourse allocation strategies are there which provide guidance for grid systems to make resource allocation decisions. The detail paper will describes various Proportional share schedulers with O(1) overhead for resource management in grid environment. The Proportional/fair share scheduler(s) ensures that resources are allocated to in an efficient manner and this ensures fairness in resource allocation. Through this paper, we are also proposing a new inter- intra fairness scheduler who integrates the features of fair share schedulers in an efficient way and will surely contribute well in managing critical issues of scheduling jobs and resources in a smooth way. The detail paper will be divided into four sections where the first section comprise a general introduction of the subject, second section describe various proportional scheduling scheduler, third section contain our proposed model detail theory and finally we summarise the paper with our future work.***

Keywords: Fairness, Grid Environment, Inter-Intra Fairness Scheduler, Proportional Share Schedulers, Resource Allocation, Resource Management.

1 Introduction

Resource management is one of the chaotic issues of grid environment. For effective utilization of the resources in grid systems, efficient application/task scheduling methods are required. Task scheduling algorithms are commonly applied by grid resource managers to optimally dispatch tasks to grid resources. Typically, grid users submit their own tasks to the grid manager to take full advantage of the grid facilities. The grid manager in a grid system tries to distribute the submitted tasks amongst the grid resources in such a way that the total response time is minimized. Similarly, there is an additional issue of providing fair share to each application of individual users according to their priority by the grid manager.

There are various fair scheduling algorithms which provided better proportional sharing accuracy. However, the time to select a client for execution using these algorithms grows with the number of clients. Most implementations require linear time to select a client for execution. In this paper, we discuss various Proportional/fair scheduling methods which provided better proportional sharing accuracy and O (1) scheduling overhead. Also we propose a new efficient Inter- Intra Fairness scheduler which integrates the features of discussed proportional/fair scheduling methods.

2 Background

This section discusses various fair scheduling methods with high fairness accuracy and O(1) overhead. These methods are basically developed for multiplexing time shared resources among a set of clients C. Section 3.1 describe briefly about Virtual time round robin scheduling method for efficient allocation of task to different resources, section 3.2 discuss about Group Ratio Round Robin, section 3.3 includes Dynamic error based fair scheduling method, section 3.4 includes Dynamic task scheduling method using service time error and virtual finish time.

2.1 Virtual Time Round Robin (VTRR)

The VTRR algorithm orders the client in the queue in the order from largest to smallest share and executes the first client for one time quantum. Once the first client completed its time quantum then its counter is decremented by one and its virtual finish time (VFT) is incremented as

$$VFT_i(t+Q) = VFT_i(t) + Q/S_i$$

Where Q is the time quantum and S_i is the share of the client. Then the scheduler moves to the next client in the run queue. At that time, we check for violation of the time counter invariant which means if the counter of current client is less than the next client then the next client will be chosen over the current one and schedule for one time quantum. If the time counter invariant is not violated then scheduling decision is made using VFT. VFT of the next client is compare with the Queue Virtual Time (QVT) which is a measure of what a client's VFT should be if it has received exactly equal to its share. This is called as VFT inequality which can be represented as

$$VFT_i(t) - QVT(t+Q) < Q/S_i$$

If this comparison is true, the scheduler selects and executes the next client in the queue for one time quantum and the process continues for the rest of the queue. If it is false, then the scheduler selects the first client to execute. One scheduling cycle ends when the time counter of all clients reaches zero.

2.2 Group Ratio Round Robin (GR^3)

This method is further divided into two different algorithms i.e. Intergroup GR^3 and Intragroup GR^3. It uses an efficient grouping strategy by grouping clients into one group with similar share values. This method deals with choosing group from a list of groups for scheduling. Groups are ordered in a from the largest to smallest group share which is the sum of shares of all clients in a group. It uses group ratio (GR) between the groups to determine which group to select. The GR of the last group is set to one and GR of rest of the groups are greater than or equal to one. The GR for group G_1 can be calculated as

$$GR_G = \text{group share of } G_1/\text{group share of } G_2$$

After selecting the group, the intragroup scheduling method selects the current client to execute within the group in a round robin manner that taken into accounts for the amount of service each client has already received. The method determines at the beginning of the round which clients still have at least as much remaining time to run in the scheduling cycle as their proportional share of service, then run those clients during the round. Given a client A with share S_A and counter C_A in a group G with group share S_G. the method runs a client if the following inequality hold when $C^{last}_G > 0$:

$$S_A/S_G <= C_A/ C^{last}_G$$

The scheduling cycle ends when all the counters become zero.

2.3 Dynamic error based fair scheduling method

This method also based on multiple queues time shared systems. The scheduler orders the tasks in the form of the queues. Queue size is obtained by dividing total no of task with total no of available processor. Every queue contains the tasks similar to its size. Every task is assigned with its respective share value and the queue is rearranged in the form of smallest to the largest share value. The scheduler selects the first task from the queue to run for one time quantum. When the first task completed its one quantum, then the STE is calculated for the current job. If

$$STE_{\text{current job}} > 0$$

is true then put the current job at the end of the queue. Then calculate and compare the STE for the first job and the second job in the queue. If

$$STE_{\text{first job}} > STE_{\text{second job}}$$

is true then the scheduler chose second job to execute otherwise first job. If the STE of the current job is false then the scheduler executes the current job again for one time quantum and the process continues till all the jobs have completed their execution.

2.4 Dynamic task scheduling method using service time error and virtual finish time.

This method is combines the benefit of GR^3 and VTRR. Tasks are maintained in the form of queues. Every client is associated with its unique identity no, counter value and weights/share. Weights are usually assigned upon the priority of the client or how much he is willing to pay for accessing resources. The first task is executed for one time quantum and then its STE is calculated and compared with the STE of the task at the head of the queue. If

$$STE_{\text{current job}} > STE_{\text{first job}}$$

is true, then execute the first job otherwise execute the current job. After completion of every scheduling cycle, VT and VFT are calculated for every task. Arrange them in largest to the smallest order and scheduler always choose the task to execute with the smallest VFT. Scheduling cycle continues with the comparison of STE and the process continues.

3 New Proposed "Inter- Intra Fairness Scheduler"

The method is an proportional share scheduler that schedules task with O(1) time complexity. The method is based upon multiple queues. We organise the clients having similar priority level with different or similar share/ weights into a single queue. Note that priority is directly related to the weight that assign to the job. For example, a job is given high share values only if it is attain a high priority level. The share value we assign to the different clients is based upon some important parameters like

1. Cost he is willing to pay
2. Deadline for the application.
3. Number of resources required by the application. (Resource Requirements)

Different combination of the above parameters can be given different priority level. This depends upon system to system requirement. The scheduling method is divided into two parts which are

- **Inter-queue scheduling**: Queues are ordered from the largest to smallest priority level with different or similar share value into single queue. Note the purpose of using different or similar share is that some defined share value range comes under same priority level like share value with 1, 2, 3 comes under priority level A, similarly share value with 4, 5, 6 comes under priority level B and so on. The method assign similar priority on the basis of cost the client is willing to pay for the application and different or similar share value on deadline or resource requirements. Queue size is obtained by dividing total no of task with total no of available processor. Every queue contains the tasks similar to its size. Every queue is associated with its queue ID, queue counter, priority level, queue share value, Virtual Finish Time (VFT) where queue ID is the unique identification of the queue, counter is like a timer of the queue which is decremented after one scheduling cycle is completed, and priority level is the level which differentiates one queue from the other one on the basis of their respective importance. Queue share value is the total number of client share within the queue which can be represented as

$$\text{Queue total share value} = [\text{cleint}^1 \text{ share value} + \text{cleint}^2 \text{ share value} + \text{cleint}^3 \text{ share value} + \ldots\ldots\ldots\ldots\ldots \text{ cleint}^n \text{ share value}]$$

Every queue is selected on the basis of their VFT. To explain VFT, we first explain the notion of VT i.e. Virtual time. The VT of a queue is a measure of the degree to which a queue has received its proportional allocation for the jobs relative to other. When a queue is executes, its VT advances at a rate inversely proportional to the queue total share value. In other words, the VT of a queue Q at time T is the ratio of $W_Q(T)$ to S_Q.

$$VT_Q(T) = W_Q(T)/S_Q$$

Where $W_Q(T)$ is the amount of service received by queue Q at time T and S_Q is the proportional share. Given a queue VT, the queue VFT is defined as the VT the group would have after executing the jobs within the queue for one time quantum each.

$$VFT_Q(T) = 1/VT_Q$$

The method then schedules queues by selecting the queue with the smallest VFT. Once the queue is selected for execution its counter is decremented and VFT is updated. If we denote the system time quantum as TQ_Q, the current queue share value as S_Q and current queue VFT as VFT_Q, then $VFT_Q(t)$ is updated as

$$VFT_Q(t+TQ) = VFT_Q(t) + TQ/S_Q$$

Then the scheduler moves to the queue from the list of the queues.

- **Intra-queue scheduling**: Once the group has been selected a job within the group is selected to run on the basis of their Service Time Error (STE). First client of the selected queue is executed for one time quantum. When the first task completed its one quantum, then the STE is calculated for the current job. The Service Time Error is the difference between the amount time allocated to the client during interval (t^1, t^2) under the given algorithm, and the amount of time that would have been allocated under an ideal scheme that maintains perfect fairness for all clients over all intervals. It is defined as $E_i(t^1, t^2)$, for any clients i out of the list of clients C where C= [i,j,…..] for time interval T where t^1 is the service starting time and t^2 is the service ending time as

$$E_i(t^1, t^2) = W_i(t^1,t^2) - (t^2-t^1) S_i/\sum_i S_i$$

Or

STE (Service Time Error) = ST given by algorithm – ST of an ideal system

A positive service time error indicates that a client has received more than its ideal share over an interval; a negative error indicates that a client has received less. If

$$STE_{\text{current job}} > 0$$

is true then put the current job at the end of the queue. Then calculate and compare the STE for the first job and the second job in the queue. If

$$STE_{\text{first job}} > STE_{\text{second job}}$$

is true then the scheduler chose second job to execute otherwise first job. If the STE of the current job is false then the scheduler executes the current job again for one time quantum and the process continues till all the jobs have completed their execution.

```
Input: A set of Queues Q where Q = [$Q_1$, $Q_2$, $Q_3$....Qn]. Every queue contains N number of jobs
from the set of total jobs T which can be allocated to M number of available processors.

Output: Schedule T onto M

    1.  Create multiple queues set Q.
    2.  While there are unexecuted queue do
    3.  For each queue
    4.  Calculate VFT, VFT=1/VT where VT(T) = W (T)/ S
    5.  Chose the queue with the smallest VFT called it as current_queue
    6.  For each current_queue, $Q_i$ in Q
    7.  Queue_Size = Remaining T/ Available M
    8.  Remove Queue_Size jobs from T and enqueue them to $Q_i$
    9.  While there are jobs in the queue do
    10. Execute the first job from $Q_i$ for one time quantum to the available resource(s) which is called as current_job.
    11. Calculate the STE for the current_job
    12. If $STE_{\text{current job}} > 0$ is true
    13. Then, put current_job at the end of the
```

Figure 1: Pseudo code for Inter-Intra fairness scheduler

4 Future Work

Our future work will be based upon implementing this inter-intra fairness scheduler on recently developed GridSim based Alea simulator on QoS parameters and tracing its behaviour with real workload traces. Also, we will try to compare it with one or more other scheduling method.

References

[1] Maruthanayagam and Uma Rani,” Grid Scheduling Algorithm- A Survey”, International Journal of Current ResearchVol. 11, pp.228-235, December, 2010.

[2]Erik Elmroth and Peter Gardfjall,”Design and Evaluation of a Decentralized System for Grid-wide Fairshare Scheduling”, Proceedings of the First International Conference on e-Science and Grid Computing (e-Science’05), IEEE 2005.

[3]Per-Olov Ostberg and Daniel Henriksson and Erik Elmroth, “Decentralized, Scalable, Grid Fairshare Scheduling (FSGrid)”, Future Generation Computer Systems, March 2011.

[4]Suja Cherukullapurath Mana “Recourse Management using a Fair Share Scheduler”,International Journal of Computer Science and Security (IJCSS), Volume (1): Issue (3), 2011.

[5]Fangpeng Dong and Selim G. Akl “Scheduling Algorithms for Grid Computing: State of the Art and Open Problems” Technical Report No. 2006-504, Queen’s University Kingston, Ontario, January 2006.

[6] Jason Nieh, Chris Vaill, Hua Zhong “Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler” Proceedings of the 2001 USENIX Annual Technical Conference Boston, Massachusetts, USA ,June 25–30, 2001.

[7]Wong chun chang, Jason Nieh “ Group Ratio Round Robin: An O(1)Proportional Share Scheduler” Technical report CUCS -012-03, Columbia university, April 2003.

[8]Daphne Lopez, Kasmir raja “A Dynamic error based fair scheduling algorithm for a computational grid”,

Journal of Theoretical and Applied Information Technology, 2005-2009.

[9]S. V. Kasmir Raja and Daphne Lopez "Dynamic task scheduling using service time error and virtual finish time", Journal of Engineering and Computer Innovations Vol. 2(5), pp. 90-97 May 2011.

[10] Saeed Parsa, Reza Entezari-Maleki ,"RASA: A New Grid Task Scheduling Algorithm", International Journal of Digital Content Technology and its Applications Volume 3, Number 4, December 2009.