DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4546

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

LOAD BALANCING NON-PREEMPTIVE JOBS IN GRID ENVIRONMENT

P.Neelakantan
Professor, Department of CSE,
VNR VJIET,
Hyderabad,India

C.Kiranmai
Professor, Department of CSE,
VNR VJIET
Hyderabad,India

Abstract: The computing resources are effectively utilized in Grid computing by using the concept of scheduling and load balancing. The non-preemptive jobs arrive at random intervalsunder varying load conditions consists of multiple interdependent tasks and independent tasks which will be executed in multiple nodes and have to be assigned to the most appropriate node during the initial placement itself. Algorithms existing in the literature considered only the processing capacity of nodes in the Grid for assigning the incoming jobs. However, the performance of the Grid system can be further improved in terms of response time by considering the length or the execution time of the jobs. In this paper, the load balancing algorithm which considers both the processing capability of the nodes and job length has been proposed for scheduling jobs to the appropriate node.

Keywords: Grid; Load balancing; Non preemptive Jobs

1. INTRODUCTION

The Grid Computing is becoming a platform for the execution of high performance applications. Organizations, individuals can communicate and collaborate by using Grid computing technologies. The goal of Grid computing is to create an illusion of single but large and powerful self – managing node out of a large collection of multiple nodes with different capabilities by using a high communication network. The Resource manager in the Grid must identify the requirements of applications, allocate, schedule and monitor those resources in an efficient manner. The Scheduling process directs the job to appropriate resource and monitoring process monitors the resource [1].

Various load balancing algorithms, such as round robin, weighted round robin, dynamic load balancing, Equally Spread Current Execution (ESCE) Algorithm, First Come First Serve, Ant Colony algorithm, and Throttled algorithm are available. The most frequently used scheduling techniques for a non-preemptive system are first in first out (FIFO) and weighted round robin (WRR) [2]. The above algorithms support grid systems consisting of multiple heterogeneous nodes.

With the aid of Grid computing and network facility, the application can use node resources in the grid and is constrained by the total processing power. The nodes in Grid computing supports resource provisioning policies which leverage virtualized services. The higher performance and utilization of nodes are improved by dynamic load scheduling implemented as improved weighted round robin, under varying load conditions. This produces the faster response time to the application when it is submitted to the grid.

The objective of the Grid computing is to assign jobs to the most suitable nodes by considering the requirements of each job and the load on the nodes present in the grid. The arrivals of jobs are directed to any of the nodes in the grid, based on the grid management policies depending.

The non-preemptive scheduling uses round robin and the weighted round robin polices. The round robin method does not consider the resource capabilities, priority and the length of the job. This causes higher priority and long jobs to take higher execution times. However the resource capabilities of the nodes are considered by the weighted round robin method and it assigns more number of jobs to the high end nodes based on the weightage. However weighted round robin algorithm will not consider the length of the job to select the appropriate node, The proposed load balancing algorithm, Modified Weighted Round Robin Load Balancing algorithm (MWRRLB) considers the length and priority of the job and selects the appropriate node to execute the jobs for lower execution times.

Performances of nodes have been improved by identifying the length of the jobs, resource capabilities, task dependency and effectively finding the lightly loaded nodes. The consideration of additional parameter" job length" helps to schedule the jobs to the appropriate nodes at any instant and is able to deliver the minimum completion time for any job. The proposed algorithm will also minimize job migrations in the grid system. The performance of the Modified Weighted Round Robin Load Balancing algorithm is compared with the existing round robin and weighted round robin algorithm and here it is assumed that the job contains multiple tasks and tasks have dependency between them. A job can use multiple nodes for its various tasks to complete its entire processing.

2. RELATED WORK

The important characteristic of job scheduling in a grid environment is balancing non preemptive dependent tasks on nodes. The load is shared among the lightly loaded nodes to achieve optimal resource utilization by the tasks to complete within less span of time. The node uses two types of job execution mechanisms such as space and time shared. In space shared mechanism the jobs will be executed one after the other and only one job/task is assigned to the CPU. The remaining jobs must wait at the job queue of the node and it makes easy to migrate a job from the overloaded node to the under loaded node. However in a time shared

mechanism, jobs share CPU slice time for execution and it gives an illusion that all jobs are executing at the same instant. In time shared policy, job migration becomes difficult due to the time sliced execution of all the jobs. The cost of moving a job from highly loaded node to a lightly loaded node will be high as a job may lose previously completed portion of the instructions in highly loaded nodes. The load balancing algorithm should also consider the unpredictable nature of job arrivals and assigning them to the appropriate nodes by considering jobs with multiple tasks and dependencies between tasks. The proposed algorithm is designed by considering that it must be suitable for both homogeneous and heterogeneous environments for varying job execution times [12].

The load balancing algorithm uses Honeybee behavior to balance the load across nodes to maximize the throughput. The amount of time the job has to wait in a queue will be reduced. This algorithm works for heterogeneous nodes and balances non preemptive independent jobs. Paper [3][10] discusses the importance of performance optimization and queuing model for a group of heterogeneous nodes with varying speeds and sizes. This algorithm addresses the optimal load allocation and distribution of jobs over multiple heterogeneous nodes across the grid.

The metric skewness used to measure the unevenness of a node can be minimized by combing different types of workloads that use different node resources. The load prediction algorithm [4][11] will minimize the number of nodes by estimating the resource usages of jobs in the future and nodes are allocated appropriately. The weighted round robin enhanced scheduling, algorithm [5][9], considers nodes processing power, job length and multiple dependent tasks in the job for balancing the load among the nodes.

The scheduling algorithm for dependent tasks in the grid environment proposed by [6][8] uses direct acyclic graph based applications and Earliest–Finish scheduling algorithm for heterogeneous computing proposed by [7] uses a stochastic hill climbing approach for load distribution in a grid environment. The workflow execution time and scheduling overhead is reduced in the grid by dynamic workflow scheduling which uses genetic algorithm that produces better results within a shorter time. The job scheduling algorithm considers priority of jobs as a main Quality of Service parameter for scheduling jobs in the grid environment. This algorithm also considers issues like complexity, makespan and consistency for improving the performance of the system..

3. SYSTEM MODEL

Let VS= $(vs_1, vs_2, \ldots vs_n)$ be the set of n nodes which executes m number of jobs represented by the set J=$\{j_1, j_2, \ldots j_m\}$. The nodes in the grid environment run in parallel to execute the incoming jobs. There is no sharing of resources owned by nodes to other nodes. Non- preemptive dependent jobs are scheduled to these nodes, where 'm' jobs are assigned to 'n' nodes represented as a linear programming model.

Execution time: Let C_{ij} be the computing time of a assigned job "i" to Node 'j" and define

$$M_{ij}=\begin{cases} 1 & \text{if job i is assigned} \\ 0 & \text{Otherwise} \end{cases}$$

The model is represented as

Minimize $\sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}M_{ij}$

Subject to $M_{ij} = 1,\ j = 1,2, \ldots m$

Average Node Utilization: The important objective in the grid computing is maximizing node utilization which is derived as

$$\text{Average Node utilization} = \frac{\sum_{x\in VS} P_x}{\text{Makespan} * \text{Number of Nodes}}$$

Where makespan can be expressed as

$$\text{Makespan} = Max\left\{\frac{P_x}{x\in VS}\right\}$$

4. PROPOSED METHOD

The figure1 shows the design of load balancing and scheduling that schedules jobs during runtime to appropriate node. Jobs may arrive at different intervals and the load status of the nodes may also change at different intervals. The under loaded node at one interval may become overloaded at another interval. The current status information of all nodes in the grid is collected by Resource Manager Module and that information is used by the Load Balancer for migrating jobs from a heavily loaded node to an idle or lightly loaded node.

4.1. Scheduling and Load Balancing Design

The user submits a job to a grid and job manager checks whether the job contains multiple independent tasks or dependent tasks. The scheduler will notify parent tasks only after child tasks are completed. Independent tasks are assigned to appropriate nodes by the proposed load balancing algorithm based Weighted Round Robin method. Each node maintains a job execution list, job pause list and job waiting list.

The user submits a job to a grid and job manager checks whether the job contains multiple independent tasks or dependent tasks. The scheduler will notify parent tasks only after child tasks are completed. Independent tasks are assigned to appropriate nodes by the proposed load balancing algorithm based weighted round robin method. Each nodeconsists ajob execution list, job pause lit and job waiting list information where a job execution list contains the current executing job list and the JobpauseList contains temporarily paused jobs and waiting jobs information is found on JobWaitingList queue. For every job arrival , the scheduler finds the least utilized node in the grid system and job is assigned to it.

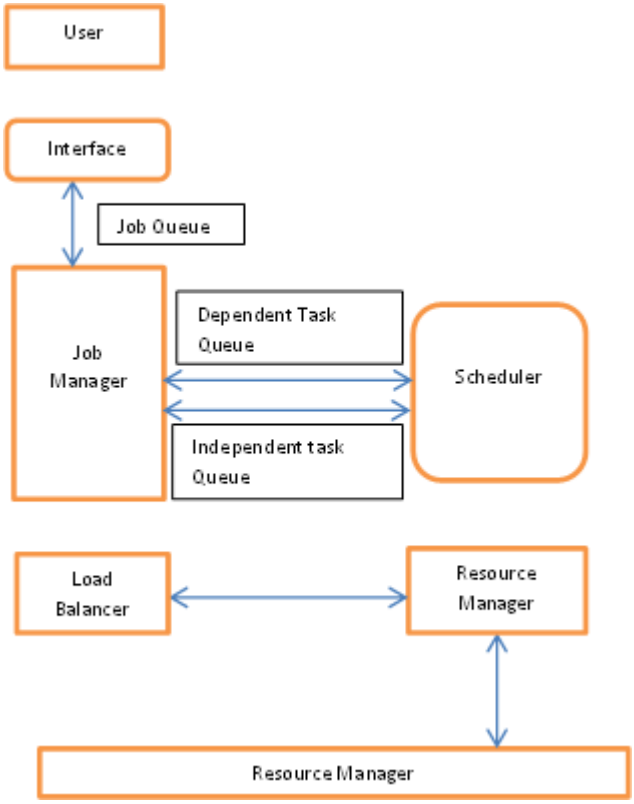

Figure 1: Scheduling and Load balancing design

The grid resource manager calculates the weightage of node based on the processing capacity and also identifies the amount memory available in each node. The ratio of the number of jobs running to the number of nodes are calculated by the load balancing algorithm and if the ratio is less than 1, then it communicates the scheduler to identify a Node for allocating the job. If the utilization is less than 0.2, then the lightly utilized node will be allotted or else the communication will be sent to the scheduler to identify the most suitable node for the job.

4.2. Computation of Load Imbalance Factor

Let VS= $(vs_1, vs_2, \ldots vs_n)$ be the set of n nodes which executes m number of jobs represented by the set J={$j_1, j_2, \ldots j_m$}. The nodes in the grid environment run in parallel to execute the incoming jobs. There is no sharing of resources owned by nodes to other nodes. Non- preemptive dependent jobs are scheduled to these nodes, where 'm' jobs are assigned to 'n' nodes represented as a linear programming model.

Execution time: Let C_{ij} be the computing time of a assigned job "i" to Node 'j" and define

$$M_{ij}=\begin{cases} 1 & \text{if job i is assigned} \\ 0 & \text{Otherwise} \end{cases}$$

The model is represented as

Minimize $\sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}M_{ij}$

Subject to $M_{ij} = 1,\ j = 1,2, \ldots m$

Average Node Utilization: The important objective in the grid computing is maximizing node utilization which is derived as

$$\text{Average Node utilization} = \frac{\sum_{x\in VS} P_x}{\text{Makespan} * \text{Number of Nodes}}$$

Where makespan can be expressed as

$$\text{Makespan} = Max\left\{\frac{P_x}{x\in VS}\right\}$$

4.3. Algorithms

The most frequently used scheduling policies in a non-preemptive system are round robin and weighted round robin policies. The proposed algorithm is modified weighted round robin algorithm which allocates job to the next node in the queue irrespective of the load of that node. The resource capabilities and job lengths are not considered by the round robin policy; hence jobs with more execution times and lower priority will end up with higher response times. The resource capabilities of the nodes is considered in the weighted round robin algorithm and based on the computing capacity the weightage has been assigned to the nodes. However, this algorithm does not consider the length of the job and hence there are possibilities of assigning the job to the wrong node.

4.4 Modified Weighted Round Robin Algorithm

The proposed algorithm considers the processing capacity, load on the node, job length for assigning a job to the appropriate node. The load at each node has been identified by the dynamic scheduling of aproposed algorithm for allocating the job to the appropriate node. Sometimes there is a probability that job execution time becomes longer than initial calculations due to the execution of more number of cycles (loop). In such situations, the load balancer rearranges jobs according to the idle slot available in the other lightly loaded nodes by moving a waiting job from heavily loaded node to the lightly loaded node. There will be no job migrations, if there are no lightly loaded nodes in the grid system. The load balancer calculates the resource load only after the completion of any of the jobs on any of the nodes to remove the overhead on the nodes. This will in turn reduces the number of job migrations between nodes and the number of probe messages requesting the lightly loaded nodes in a grid environment.

4.4.1. Implementation of the Algorithm

The implementation of the algorithm consists of four modules (a) Scheduler (b) Load balancer (c) resource manager. The scheduler has to find suitable nodes for incoming jobs. The scheduler can schedule the jobs during run time or before runtime. The job migration froma heavily loaded node to lightly loaded node has been decided by the load balancer by utilizing the resource monitor information. The resource manager module communicates with all Nodes and collects information such as their processing capacity, current load and number of jobs queued at each node. This information serves as a basis for the load balancer to migrate the jobs to the suitable nodes to complete the jobs in minimum span of time.

The system architecture is shown below, in which the grid system consist of grid broker and multiple nodes. The node can host any number of jobs based on its processing

capacity. The grid broker has important components such as scheduler and load balancer to schedule and balances the jobs effectively.

Load balancing using MWRR is done by collecting the pending execution time of all jobs from all nodes and arranged in ascending order of pending time followed by runtime of the arrived jobs in the queue based on the priority. The job mapping involves selecting the job from the queue and calculation of job completion time in each node and job has been assigned to the most appropriate node based completion time of the job and pending execution time of all the jobs residing at that node. The corresponding jobs and their execution time is added to the pending time of node by the load balancer to rearrange the nodes based on their utilization.

1) Calculate unfinished execution time in each of the node by collecting the unfinished execution length from JobExecution ,JobWaitingQueue and JobPauseQueue list.
    a) Set LengthofUnfinishedJobs=LengthofJobsRemainLengthinJobExecutionlist+JobsRemainginglength in Jobwaitlist+JobsRemaining in JobPauselist
    b) Pi is Processing capacity of each node
    c) Set EPTime = TotLengthofunfinishedJobs/Pi
2) Sort Virtualnodes based on the leastunfinishedexecutiontime
    a) Nodes of the same unfinishedjob execution length is assigned with execution time as key and its associated node as a value.
    b) Sort the VirtualNodes by the unfinishedExecution Time of each node
3) sort the incoming Jobs based on the length & priority of the Jobs.
    a) Sort the JobSubmittedList based on length & priority.
4) Initialize the Indexvs, indexjob&totalJobs
    a) Set indexvx = 0
    b) Set totalJobs = length of JobSubmittedList
    c) Set VScount= size of nodes in the grid
    d) Set indexjob= 0
    e) Set jobToVSratio = totalJobs/VScount
5) Assign the incoming jobs to the VS based on the least unfinished Execution Time in the Nodes and their processing capacity.
6) Remove all the assigned Jobs from the JobSubmittedList

5. EXPERIMENTS

The performance of MWRR algorithm is compared with the existing algorithm round robin and aweighted round robin in the grid sim. The simulator is used to find response time of the jobs, the number of job migrations under the heterogeneous grid environment. The load balancer in the MWRR identifies the highly loaded node from the grid and calculates the possible completion of those jobs present in the overloaded node and the lightly loaded node. If the lightly loaded node can finish any of the jobs present in the overloaded node in the shortest possible time, then that job will be migrated to a lightly loaded node. The proposed algorithm considers the ratio of processing capacity of the node to the total processing capacity of the nodes and assigns the appropriation number of arrived jobs to the nodes. The simple RR will not consider node capabilities and job lengths. It simply assigns jobs to nodes one after the other in an ordered manner. So, the completion time of the jobs is higher than the other 2 algorithms.

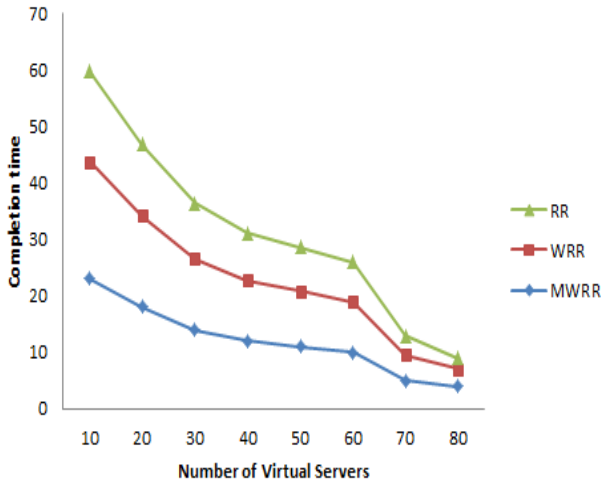

Figure2: Number of virtual servers vs Completion time

Comparison of Job Migration

The modified WRR algorithm identifies the most appropriate nodes and hence the job migrations are minimal when compared to the round robin and weighted round robin algorithm The weighted round robin considers only the processing capacity of the nodes and hence there will be a few job migrations as it won't consider the job length. In contrast, the MWRR considers both processing capacity and job length and hence it is possible for the algorithm to identify the appropriate nodes for the newly arriving jobs and hence there will be minimal job migrations or none.

6. CONCLUSIONS AND FUTURE WORK

The objective of the grid computing is to achieve high system utilization of geographically dispersed distributed and heterogeneous resources. However the application performance in the grid remains a challenging task in a dynamic grid environment. Resources can be submitted to the grid at any moment and similarly they can also be withdrawn at any moment. Every load balancing algorithm consists of five policies. The effective implementation of these polices by the load balancing algorithm drives the overall performance of systems. The most important feature of Grid Middleware will be the execution of intensive compute applications. Load balancing algorithms that use Genetic algorithms may be considered for the future implementation of scheduling jobs to nodes.

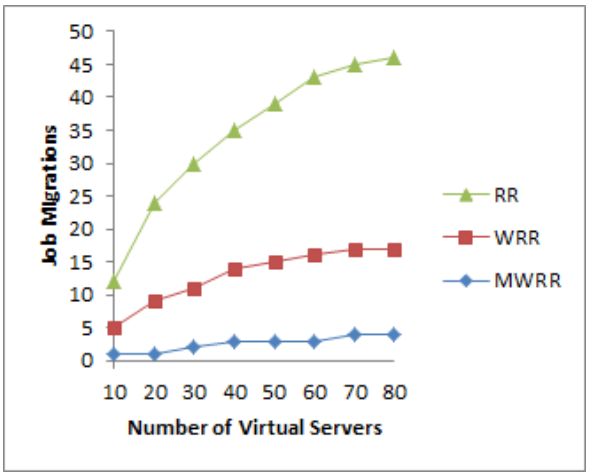

Figure 3: Number of virtual servers vs Job migration

REFERENCES

1. Buyya, R., D. Abramson, J. Giddy and H.Stockinger, 2002. Economic models for resource management and scheduling in grid computing. J. Concurrency and Computation: Practice and Experience, 14: 1507-1542.
2. Sachin Kumar, NeerajSinghal, “A Priority based Dynamic Load Balancing Approach in a Grid based Distributed Computing Network”, International Journal of Computer Applications (0975-8887), Vo. 49, No.5, July 2012, pp. 11-13
3. Sharma D, Sharma K, Dalal S. Optimized load balancing in grid computing using tentative ant colony algorithm. International Journal of Recent Research Aspects. 2014 Jun; 1(1):35–9.
4. Nadimi-Shahraki MH, Fard ES, Safi F. Efficient load balancing using ant colony optimization. Journal of Theoretical and Applied Information Technology. 2015 Jul; 77(2):253–8.
5. Buyya R, Murshed M. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and computation: Practice and experience. 2002 Nov; 14(13-15):1175–20.
6. Srivastava PK, Gupta S, Yadav DS. Improving performance in load balancing problem on the grid computing system. International Journal of Computer Applications. 2011 Feb; 16(1):6–10.

a. Dorigo M, Blum C. Ant colony optimization theory: A survey. Theoretical Computer Science. 2005 May; 344(23):243–78.

7. Ludwig SA, Moallem A. Swarm intelligence approaches for grid load balancing. Journal of Grid Computing. 2011 Sep; 9(3):279–301.
8. AlexandruCârstea, Georgiana Macariu, Towards a grid enabled symbolic computation architecture, Pollack Periodica Aug 2008, Vol. 3, Issue 2, pp. 15-26.
9. BalázsTukora, Tibor Szalay, High performance computing on graphics processing units, Pollack Periodica Aug 2008, Vol. 3, Issue 2, pp. 27-34
10. PuligundlaNeelakantan, Ambati Reddy," Decentralized load balancing in distributed systems," Pollack Periodic, Aug 2014, Vol. 9, Issue 2, pp. 15-28
11. Shanmugasundaram Suresh, JeevaPoornaselvan, Chidambaram Divyapreya,”Optimal path planning approach to Grid environment”,PollackPeriodica Apr 2011, Vol. 6, Issue 1, pp. 131-140.