International Journal of Research in Advent Technology (IJRAT) (E-ISSN: 2321-9637)
Special Issue
National Conference "CONVERGENCE 2016", 06th-07th April 2016

Design and Implementation of load balancing in grid using min-min algorithm

Ashutosh V Kale, Gopal U Tangade, Yogesh J Jadhao, Prof.V.P.Narkhede, Prof. S. M. Dandage
Computer science and engineering.
Ashutoshkale.kale@gmail.com,yjadhao493@gmail.com,jayeshtangade@gmail.com

Abstract-

Grid computing is become as real alternative to traditional supercomputing environments for developing parallel applications that harness massive computational resources. However, the complexity incurred in building such parallel Grid-aware applications is higher than the traditional parallel computing environments. It addresses issues such as resource discovery, heterogeneity, fault tolerance and task scheduling. Load balanced task scheduling is very important problem in complex grid environment. So task scheduling which is one of the NP-Complete problems becomes a focus of research scholars in grid computing area. The traditional Min-Min algorithm is a simple algorithm that produces a schedule that minimizes the make-span than the other traditional algorithms in the literature. But it fails to produce a load balanced schedule. In this paper a Load Balanced Min-Min (LBMM) algorithm is proposed that reduces the make span and increases the resource utilization. The proposed method has two-phases. In the first phase the traditional Min-Min algorithm is executed and in the second phase the tasks are rescheduled to use the unutilized resources effectively. Grid computing is a kind of distributed computing that involve the integrated and collaborative use of distributed resources. It involves huge amounts of computational task which require reliable resource sharing across computing domains. Load balancing in grid is a technique which distributes the workloads across multiple computing nodes to get optimal resource utilization, minimum time delay, maximize throughput and avoid overload. It is a challenging problem that has been studied extensively is the past several years. The design of load balancing algorithm requires completed understanding of the grid system and the scheduling strategies used, the heterogeneity between available nodes, the challenges and issues, and their limitations within the grid Architecture.

Index Terms- Grid Computing, Load Balancing, Min-Min Algorithm

1. INTRODUCTION

Load balancing is deviding the amount of work that a computer has to do between two or more computer so that more work get done in same amount of time and,in ganeral, all user get served faster ,load balancing can be implementing with hardware, software or communication of both.

Mixed-machine heterogeneous computing environments [2] are a group of heterogeneous high - performance machines interconnected with high - speed links. They are used to solve a variety of computationally intensive applications that require different computing environments.[1] The Min-Min algorithm first finds the minimum execution time of all tasks. Then it chooses the task with the least execution time among all the tasks. The algorithm proceeds by assigning the task to the resource that produces the minimum completion time. The same procedure is repeated by Min-Min until all tasks are scheduled[1]. The algorithm proposed in this paper outperforms all those algorithms both in terms of make span and load balancing. Thus a better load balancing is achieved and the total response time of the grid system is improved. The proposed algorithm applies the Min-Min strategy in the first phase and then reschedules by considering the maximum execution time that is less than the makespan obtained from the first phase[1].

2. LOAD BALANCING：

Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a *server farm* or *server pool*.

Modern high-traffic websites must serve hundreds of thousands, if not millions, of concurrent requests from users or clients and return the correct text, images, video, or application data, all in a fast and reliable manner. To cost-effectively scale to meet these high volumes, modern computing best practice generally requires adding more servers.

A load balancer acts as the "traffic cop" sitting in front of your servers and routing client requests across all servers capable of fulfilling those requests in a manner that maximizes speed and capacity utilization and ensures that no one server is overworked, which could degrade performance. If a single server goes down, the load balancer redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancer automatically starts to send requests to it.

In this manner, a load balancer performs the following functions:

- Distributes client requests or network load efficiently across multiple servers
- Ensures high availability and reliability by sending requests only to servers that are online
- Provides the flexibility to add or subtract servers as demand dictates

3 .RELATED WORK :

A load balancing algorithm aims to increase the utilization of resources with light load or idle resources thereby freeing the resources with heavy load. The algorithm tries to distribute the load among all the available resources. At the same time, it aims to minimize the makespan with the effective utilization of resources.[1] Braun et al [2] have studied the relative performance of eleven heuristic algorithms for task scheduling in grid computing. They have also provided a simulation basis for researchers to test the algorithms. Their results show that Genetic Algorithm (GA) performs well in most of the scenarios and the relatively simple Min-Min algorithm performs next to GA and the rate of improvement is also very small. The simple algorithms proposed by Braun are Opportunistic Load Balancing (OLB), Minimum Execution Time(MET), Minimum Completion Time(MCT), Min-Min, Max-min.

Opportunistic Load Balancing (OLB) assigns the jobs in a random order in the next available resource without considering the execution time of the jobs on those resources. Thus it provides a load balanced schedule but it produces a very poor makespan.
Minimum Execution Time (MET) assigns jobs to the resources based on their minimum expected execution time without considering the availability of the resource and its current load.
This algorithm improves the makespan to some extent but it causes a severe load imbalance.
Minimum Completion Time (MCT) assigns jobs to the resources based on their minimum completion time. The completion time is calculated by adding the expected execution time of a job on that resource with the resource's ready time. The machine with the minimum completion time for that particular job is selected. But this algorithm considers the job only one at a time. Min-Min algorithm starts with a set of all unmapped tasks. The machine that has the minimum completion time for all jobs is selected. Then the job with the overall minimum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped tasks are assigned. Compared to MCT this algorithm considers all jobs at a time. So it produces a better makespan. Max-Min is similar to Min-Min algorithm. The machine that has the minimum completion time for all jobs is selected. Then the job with the overall maximum completion time is selected and mapped to that resource. The ready time of the resource is updated. This process is repeated until all the unmapped tasks are assigned. The idea of this algorithm is to reduce the wait time of the large jobs.[1]

LOAD BALANCED MIN-MIN ALGORITHM :

.

the load balanced min-min algorithm as shown in fig 1. The algorithm starts by executing the steps in Min-Min strategy first. It first identifies the task having minimum execution time and the resource producing it. Thus the task with minimum execution time is scheduled first in Min-Min. After that it considers the minimum completion time since some resources are scheduled with some tasks. Since Min-Min chooses the smallest tasks first it loads the fast executing resource more which leaves the other resources idle. But it is simple and produces a good make-span compared to other algorithms.[1]

for all tasks *Ti*
for all resources
Cij=Eij+rj
do until all tasks are mapped
for each task find the earliest completion time and the
resource that obtains it
find the task *Tk* with the *minimum* earliest completion time
assign task *Tk* to the resource *Rl* that gives the earliest
completion time
delete task *Tk* from list
update ready time of resource *Rl*
update *Cil* for all *i*
end do
// rescheduling to balance the load
sort the resources in the order of completion time
for all resources R
Compute makespan = max(CT(R))
End for
for all resources
for all tasks
find the task Ti that has minimum ET in Rj
find the MCT of task Ti
if MCT < makespan
Reschedule the task Ti to the resource that produces it
Update the ready time of both resources
End if
End for
End for
//Where MCT represents Maximum Completion Time

Fig 1.lbmm heuristic

So LBMM executes Min-Min in the first round. In the second round it chooses the resources with heavy load and reassigns them to the resources with light load. LBMM identifies the resources with heavy load by choosing the resource with high makespan in the schedule produced by Min-Min. It then considers the tasks assigned in that resource and chooses the task with minimum execution time on that resource. The completion time for that task is calculated for all resources in the current schedule. Then the maximum completion time of that task is compared with the makespan produced by Min-Min. if it is less than makespan then the task is rescheduled in the resource that produces it, and the ready time of both resources are updated. Otherwise the next maximum completion time of that task is selected and the steps are repeated again. The process stops if all resources and all tasks assigned in them have been considered for rescheduling. Thus the possible resources are rescheduled in the resources which are idle or have minimum load.[1]

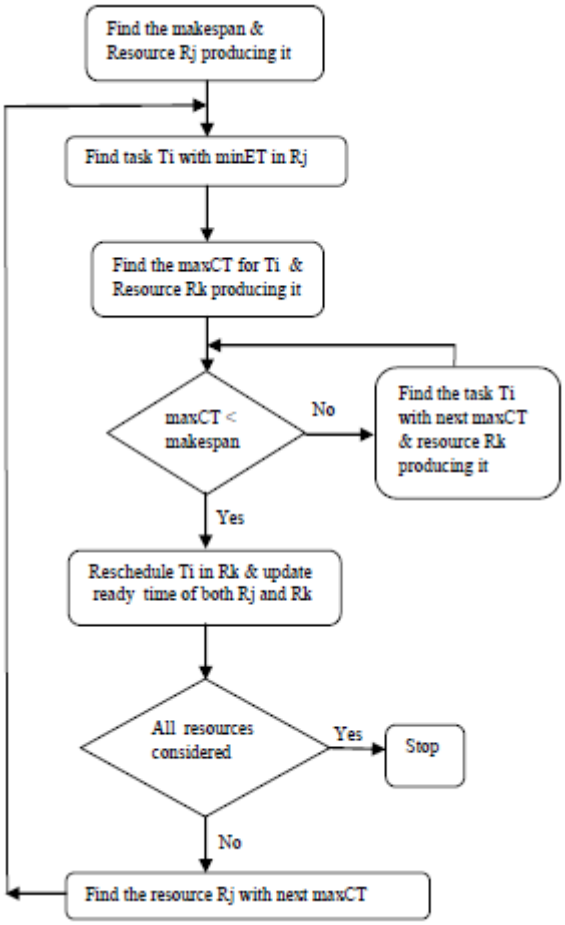

Fig.2 Rescheduling phase of LBMM

EXAMPLE:

Consider a grid environment with two resources R1 and R2 and a meta-task group *Mv* with four tasks *T1*, *T2*, *T3* and *T4*. The grid scheduler is supposed to schedule all the tasks within *Mv* on the available resources *R1* and *R2*. Since Min-Min algorithm is simple and produces a better make-span than the other algorithms discussed in the literature, the proposed algorithm executes the Min-Min algorithm in the first phase to schedule the jobs. But to remove the limitation of unbalanced load in Min-Min the jobs are rescheduled in the second phase. In this problem the execution time of all tasks are known prior. They can also be calculated if the number of instructions in each job and the computation rate of each resource is known. They are represented (in sec) in Expected Time to Compute (ETC) table. Table 1 represents the execution time of the tasks on each resources[1]

Table 1. Expected Execution Time of Tasks

Tasks	Resources	
	R1	R2
T1	7	2
T2	11	3
T3	12	3
T4	6	2

Static mapping of tasks to machines based on Min-Min is shown in Figure 3. Min-Min choose the minimum completion time and so all tasks are scheduled to resource R2 and resource R1 remains idle. The makespan produced by Min-Min is 10 sec

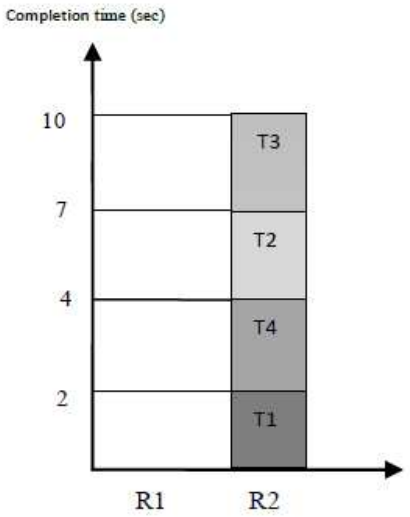

Figure 3. Gantt chart of Min-Min Algorithm

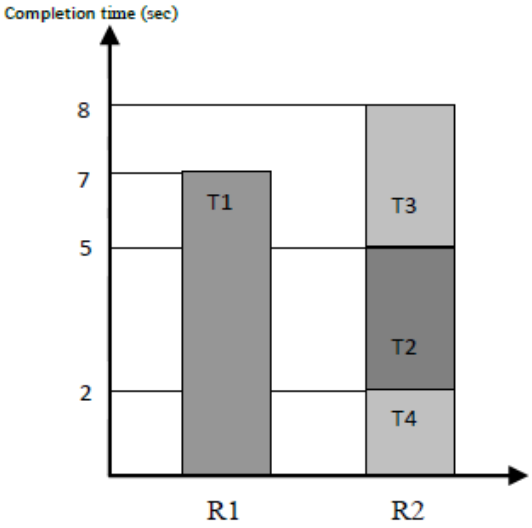

Figure 4. Gantt chart of LBMM Algorithm

According to the proposed LBMM task T1's maximum completion time is less than makespan produced by Min-Min. Other task's maximum completion time is not less than makespan. So task T1 is rescheduled in resource R1 and the remaining tasks are scheduled in the same resource R2. The result of LBMM is shown in Figure 2. Thus the rescheduling of Min-Min algorithm utilizes the idle resource R1 as well as reduces the makespan to 8 sec. Mapping of tasks based on LBMM is shown in figure 4.

CONCLUSIONS :

Min-Min and Max-Min algorithms are applicable in small scale distributed systems. When the number of the small tasks is more than the number of the large tasks in a meta-task, the Min-Min algorithm cannot schedule tasks, appropriately, and the makespan of the system gets relatively large. Furthermore it does provide a load balanced schedule

REFERENCES

[1]T.kokilavani, Dr.D.I.George Amalarethinam,"Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing," International Journal of Computer Applications (0975 – 8887) Volume 20– No.2, April 2011

[2] Braun, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., et al. "A comparison of eleven static heuristics for mapping a class
of independent tasks onto heterogeneous distributed computing systems", Journal of Parallel and Distributed
Computing, Vol. 61, No. 6, pp.810–837, 2001

[3] Chapman. C, Musolesi. M, Emmerich. W, Mascolo. C, "Predictive Resource Scheduling in Computational Grids" in Parallel and Distributed Processing Symposium, IEEE International Vol. 26, pp.1 – 10, 2007

[4] Ian Foster, Carl Kesselman, Steven Tuecke, "The Anatomy
of the Grid Enabling Scalable Virtual Organizations" International Journal of Supercomputer Applications, 2001.

[5] Mr. V.P.Narkhede,Prof S.T.Khandare," Dynamic Load Balancing In Grid Computing Using Fair Scheduling", Research Article Issn: 2319-507x V P Narkhede, Ijpret, 2014; Volume 2 (8):623-635

[6] Kokilavani.T and George Amalarethinam.D.I, Applying Non -Traditional Optimization Technique s to Task Scheduling in Grid Computing, International Journal of Research and Reviews in Computer Science, Vol. 1, No. 4, Dec 2010, pp. 34 - 38

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

[8] Geoffrey Falzon, Maozhen Li, "Enhancing list scheduling heuristics for dependent job scheduling in grid computing environments", Journal of Supercomputing, Springer, March 2010.

[9] Do reen Hephzibah Miriam. D and Easwarakumar. K.S, A Double Min Min Algorithm for Task Metascheduler on Hypercubic P2P Grid Systems, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 5, July 2010.

[10] He. X, X- He Sun, and Laszewski. G.V, "QoS Guided Min- min Heuristic for Grid Task Scheduling," Journal of Computer Science and Technology, Vol. 18, pp. 442-451, 2003.

[11] Kamalam.G.K and Muralibhaskaran.V, , A New Heuristic Approach:Min - Mean Algorithm For Scheduling Meta - Tasks On Hetero genous Computing Systems , IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.1, January 2010.

[12] Sameer Singh Chauhan,R. Joshi. C, QoS Guided Heuristic Algorithms for Grid Task Scheduling, International Journal of Computer Applications (0975 –8887), pp 24-31, Volume 2,No.9, June 2010.