An Efficient Job Scheduling Algorithm for Grid Computing

Sarpreet Singh
Gian Sagar Medical College, Banur
Patiala, India
+91-9876062515
ersarpreetvirk@gmail.com

RK Bawa
Department of Computer Science
Punjabi University, Patiala, India
+91-9815653271
rajesh_k_bawa@yahoo.com

ABSTRACT

The emerging computational grid infrastructure consists of heterogeneous resource in widely distributed autonomous domain, which makes resources scheduling even more challenging. The most common objective function of task scheduling problem is makespan. In this paper, I have proposed an efficient Job scheduling algorithm for the grid environment so that the jobs are executed in minimum time and also all nodes of grid execute equal load relative to their executing power. This technique fetches the jobs from the job queue that is ready to execute and assign these jobs to the best nodes of the grid.

Categories and Subject Descriptors

I.1.2 Algorithms (F.2.1, F.2.2)

General Terms

Algorithms, Performance

Keywords

Algorithm

1. INTRODUCTION

In today's complex world of high speed computing, computers have become extremely powerful, even home-based desktops are powerful enough to run complex applications. But still we have numerous complex scientific experiments, advanced modeling scenarios, astronomical research, and a wide variety of simulations, complex scientific & business modeling scenarios which require huge amount of computational resources. To satisfy some of these aforementioned requirements, we have many advance computing environment, each having their own computing features like Parallel computing and Distributed computing, Cluster computing, Grid computing.

Grid is type of Parallel & Distributed computing. Grid computing in the simplest case refers to cooperation of multiple processors on multiple machines which are geographical distributed and its objective is to boost the computational power in the scientific problem and other field which require high capacity of the CPU or other resources. The theory behind "Grid Computing." is not to buy more resources but to borrow the power of the computational resources you need from where it's not being used". Grid Resources [1] fall into the categories of computation (i.e. a machine sharing its CPU) storage (i.e. a machine sharing its RAM or disk space), communication (i.e. sharing of bandwidth or a communication path), software and licenses and special equipment (i.e. sharing of devices). According to Foster and Keselman in [3] grid computing is hardware and software infrastructure which offer a cheep, distributable, coordinated and reliable access to powerful computational capabilities.

The resource management system (RMS) which is central component of a grid computing environment accept requests for resources from machines within the grid and assign specific machine resources to a request from the overall pool of grid resources for which the user has access permission. Now the performance of the RMS depends upon, how the resources in the distributed computing are using i.e. how efficiently jobs are schedules and assign to the best resources to minimize the makespan (the time between first input file is sent to computational server and the last file returned to the user or which is defined as the duration between the start time of the first job and the finish time of the last executed job). This work is done by the scheduler. In the field of grid, resource management and job scheduling is vital area.

A scheduler is a component of the resource management system on a grid [6]. A scheduler system provides the interface between a user and the grid resources. Scheduling of jobs on a grid or a cluster is the task of mapping jobs to the available compute-nodes. The Complexity of scheduling problem increases with the size of the grid and becomes highly difficult to solve effectively [4]. On a grid, scheduling is a NP-complete problem. [7]

According to Wright [1], a scheduler is designed to satisfy one or more of the following common objectives:

(a) Maximizing system throughput. (b) Maximizing resource utilizations. (c) Maximizing economic gains (d) Minimizing the turn-around time for an application.

This paper is organized as follow. A brief scheduling process is given in section 2. In section 3, I am presenting my proposed computational Grid Model, terminology & basic assumption related with algorithm. In section 4, I am proposing the scheduling algorithm. Section 5 presents the results that are taken by using proposed algorithm & other techniques. Section 6 concludes the result of proposed algorithm.

2. SCHEDULING PROCESS

A complete grid scheduling framework comprises application model, resource model, performance model, and scheduling policy. The scheduling policy can further decomposed in to three phases [5]: Phase one is resource discovery, which generates a list of potential resources. Phase two involves gathering information about those resources and choosing the best set to match the application requirements. In the phase three the job is executed. To achieve the promising potentials by using tremendous distributed resources, effective and efficient scheduling algorithms are fundamentally important. Scheduling such applications is challenging because target resources are heterogeneous, because their load and availability varies dynamically [2]. Previous parallel system was assumed to be organized with homogeneous platform and connected via memory, bus, or LAN. But today its platform is heterogeneous and connected via Internet so each platform has different ability of computation performance and different network bandwidth. So, traditional list scheduling algorithms which consider just makespans are inefficient to current parallel system. For that reason, I have developed a new algorithm for grid environment so that the jobs are executed in minimum time and also all nodes of grid execute equal load relative to their executing power.

3. PROPOSED COMPUTATIONAL GRID MODEL

Grid is basically a collection of nodes. Each node can handle any number of child nodes. Here we are proposing the binary grid model. This model acts as a binary tree, having each node handling two nodes only. In real grid environment these nodes are basically real computational resources that execute the real jobs.

Figure 1 shows the structure of binary grid model having six nodes. Each node is handling two nodes for example Node 0 has two children - Node 1 & Node 2. The numerical value beside each node denotes the computation power which is in Hz. The numeric value in each node is the ID of each node.

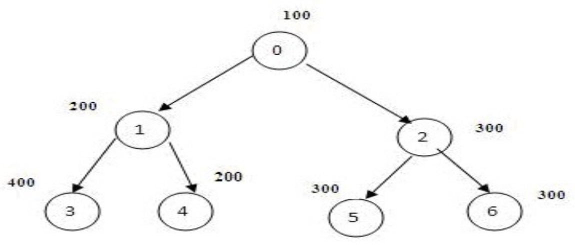

Figure 1. Binary Tree as Grid.

3.1 Basic Assumption

1. This algorithm is applied to the binary grid structure i.e. each computer or node send instructions to their left or right child only.
2. All the jobs are submitted to the main node i.e. Node 0.
3. The Power of each node is in Hz. i.e. the capacity to execute the instruction per second.
4. This technique simulates the node of binary tree as real computer and each jobs as real jobs with fix number of instruction.
5. Arrival time is considered in seconds.

3.2 Terminology

1. This algorithm use the array for storing the jobs i.e. JOBQUEUE [arrival time] [# instruction] having two parameter. First parameter stores the arrival time of job and second parameter for number of instructions.
2. The core parameter of this algorithm is TAT(Turn Around Time) of jobs ,which is the interval from the time of submission of a job to the time of completion .TAT is the sum of the period spent waiting to get into the memory, executing on the CPU, and doing the I/O. When the TAT of each node is zero, it means the node executes the job completely and now it is free for executing more jobs
3. Old TAT is the previous TAT of a node. Upon the arrival of a new job old TAT is added with (instruction/power of node), this gives the new TAT of that node.

4. PROPOSED ALGORITHM

When we submit the jobs in this binary grid, jobs are submitted by their arrival time and the number of instructions to the JOBQUEUE [arrival time] [# instruction]. This technique also works efficiently in the case when multiple jobs are simultaneously submit to JOBQUEUE.

At the starting when whole grid is free, TAT of the all nodes will be zero, which means that all nodes are free. This technique using one thread called “scanning thread”, which continuously scans the JOBQUEUE at each second, to check whether the user has submitted any job to JOBQUEUE or not. If any job is submit with their arrival time and number of instructions then this technique search the best node from the whole grid, which executes that job in minimum time. This is done by checking the TAT value on all the grid nodes with the help of following formula:

Turn Around Time (TAT) = old TAT + (number of instruction/ Power of the node (Hz))

TAT is calculated at j^{th} node by adding the old TAT of j^{th} Node to the number of instruction of i^{th} job / Power of the j^{th} node (Hz)

Then we assign that i^{th} job to j^{th} node whose TAT is minimum so that the job takes minimum time to complete the execution. We also update TAT for j^{th} Node by adding the new calculated TAT to its previous TAT.

After assigning the node, another thread called “decrement thread” executed which decrement the TAT value of all the nodes by one till TAT of all the nodes becomes zero, which would mean that every node has executed its job and is now free for executing more jobs.

4.1 Algorithm

1. Receive all the jobs in the job queue.
2. At starting assign zero to Turn Around Time (TAT) of all the nodes in the grid i.e. all the nodes are free.
3. Fetch the jobs from JOBQUEUE.
4. Read the number of instruction for ith job and assign the jth node whose TAT for execution the ith job is minimum. // assign the node

a) Check the value of TAT on all the nodes for ith job by:

Turn Around Time (TAT) = old TAT + (number of instruction/ Power of the node (Hz)

b) Assign the jth node whose TAT is minimum

5. Update the grid by adding the calculated TAT to the Previous TAT of jth node which is assigned to ith job in above step.
6. At every second decrement the TAT value of all the nodes whose TAT is more than Zero.
7. Scan the job queue. If any job is arrived, go to step 3
8. Else decrement the TAT value of all the nodes by one until TAT value of all the nodes is zero i.e. execution of all the jobs have completed

5. EXPERIMENT RESULT

For implementing an efficient job scheduling algorithm in grid, Binary grid model is implemented on single machine using C language with binary tree data structure. The nodes of tree simulate the real machine in the real grid. The proposed algorithm is applied on different set of inputs.

Following is the first set of input and output Table 1 shows the input of ten jobs with their arrival time and number of instructions & Table 2 shows the power capacity of each node.

Table 3 is resulting table, shows which job was assigned to which node and in how much time it has executed.

Table 1. Input to JOBQUEUE.

Job id	Arrival time	Number of instruction
j0	0	2500
j1	0	7000
j2	1	2000
j3	3	3000
j4	4	5000
j5	8	1500
j6	13	1000
j7	22	8000
j8	26	9000
j9	33	12000

Table 2. Node's Power.

Node	Power
0	100
1	200
2	300
3	400
4	200
5	100
6	300

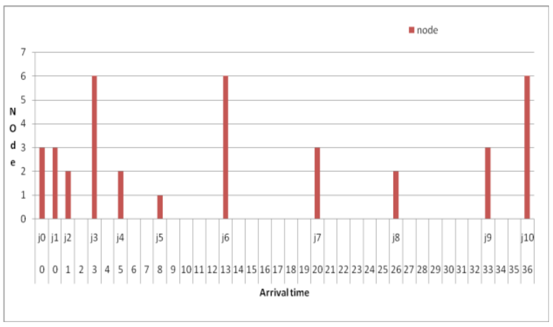

Figure 2. Scanning thread of Proposed algorithm Scan the JOBQUEUE.

Fig 2 shows working of scanning thread, which scan the JOBQUEUE at every second and check whether user submitted any job or not. Like job j0 and j1 both were arrived at 0th second and j3 was submitted at 3rd second. But at 9th, 10th, 11th, 12th second there was no job. This fig gives one more information that which job is assigned to which node.

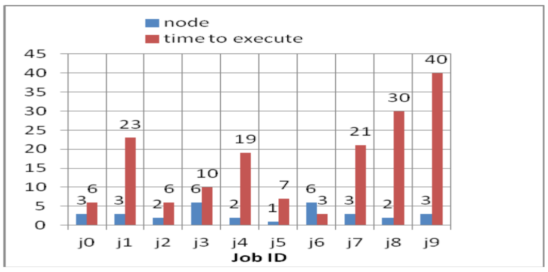

Figure 3. Result after using the proposed technique for assignment of jobs.

Fig 3 shows the result of the proposed technique. A blue bar shows the jth node and the red bars shows in how much time, ith job will be executed. Like job j0 is assigned to the node 3 and will be executed in 6 seconds.

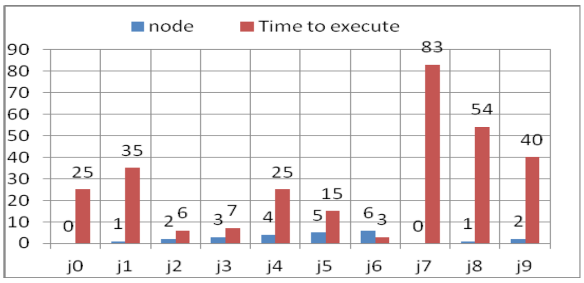

Figure 4. Result after using the sequentially assignment of job.

Fig 4 shows the result of second method of assignment i.e. sequential assignment of job**.** A blue bar shows the jth node and the red bars shows in how much time, ith job will be executed.

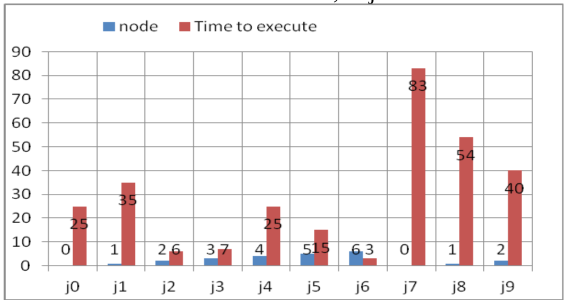

Figure 5. Result after using random assignment of jobs.

Fig 5 shows the result of third method of assignment i.e. random assignment of job**.** A blue bar shows the jth node and the red bars shows in how much time, ith job will be executed. In this any ith job is assign to any jth node.

6. CONCLUSION

After taking the result of I have given input of Table 1 to our proposed algorithm and taking the results. These results are compared with the other two methods of assignment i.e. with sequential assignment method and with the random assignment method. In each case it is observed that our technique gives better results than others. Our technique schedules the jobs on the grid nodes in such a way that it takes minimum time to execute the jobs.

Table 3. Comparison Table.

JOB ID	Proposed technique	Sequentially Assignment	Randomly Assignment
	Time to Execute(sec)	Time to Execute(sec)	Time to Execute(sec)
j0	6	25	25
j1	23	35	35

j2	6	6	6
j3	10	7	15
j4	19	25	18
j5	7	15	15
j6	3	3	3
j7	21	83	20
j8	30	54	38
j9	40	40	40

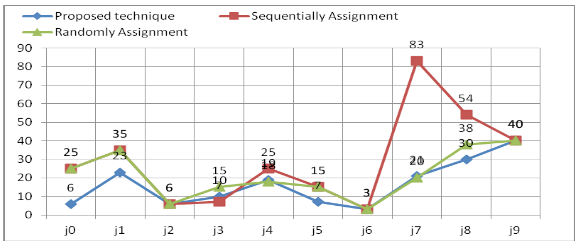

Figure 6. Comparison Chart.

7. REFERENCES

[1] D Wright, Cheap Cycles from the Desktop to the Dedicated Cluster: Combining Opportunistic and Dedicated Scheduling with Condor, Proceeding of HPC Revolution 01, Illinois, 2001.

[2] H. Casanova, A. Legrand, D. Zagorodnov and F. Berman, Heuristics for Scheduling Parameter Sweep Applications in Grid Environments, in Proc. of the 9th hetero-geneous Computing Workshop (HCW'00), pp. 349-363, Cancun, Mexico, May 2000.

[3] I. Foster, "What is the Grid?" Daily News and Information for The Global Grid Community, Vol.1, No.6, July 22, 2002.

[4] Jamshid Bagherzadeh , Mojtaba MadadyarAdeh:' An Improved Ant Algorithm for Grid Scheduling Problem" Proceedings of the 14th International CSI Computer Conference , 2009.

[5] Jarek Nabrzyski, Jennifer M. Schopf & Jan Welglarz, Grid Resource Management– State of the art and Future trends, Kluwer Academic Publisher.

[6] M. Aggarwal, R.D. Kent and A. Ngom, Genetic Algorithm Based Scheduler for Computational Grids, in Proc. of the 19th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'05), pp.209-215, Guelph, Ontario Canada, May 2005.

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