GROUPING BASED JOB SCHEDULING ALGORITHM USING PRIORITY QUEUE AND HYBRID ALGORITHM IN GRID COMPUTING

Pinky Rosemarry[1], Ravinder Singh[2], Payal Singhal[3] and Dilip Sisodia[4]

Department of Computer Engineering,
Govt. Engineering College, Ajmer, India
[1]Pinkyrose001@gmail.com
[2]ravikaviya@gmail.com
[3]payalsinghal8@gmail.com
[4]sisodia.dilip@gmail.com

ABSTRACT

Grid computing enlarge with computing platform which is collection of heterogeneous computing resources connected by a network across dynamic and geographically dispersed organization to form a distributed high performance computing infrastructure. Grid computing solves the complex computing problems amongst multiple machines. Grid computing solves the large scale computational demands in a high performance computing environment. The main emphasis in the grid computing is given to the resource management and the job scheduler .The goal of the job scheduler is to maximize the resource utilization and minimize the processing time of the jobs. Existing approaches of Grid scheduling doesn't give much emphasis on the performance of a Grid scheduler in processing time parameter. Schedulers allocate resources to the jobs to be executed using the First come First serve algorithm. In this paper, we have provided an optimize algorithm to queue of the scheduler using various scheduling methods like Shortest Job First, First in First out, Round robin. The job scheduling system is responsible to select best suitable machines in a grid for user jobs. The management and scheduling system generates job schedules for each machine in the grid by taking static restrictions and dynamic parameters of jobs and machines into consideration. The main purpose of this paper is to develop an efficient job scheduling algorithm to maximize the resource utilization and minimize processing time of the jobs. Queues can be optimized by using various scheduling algorithms depending upon the performance criteria to be improved e.g. response time, throughput. The work has been done in MATLAB using the parallel computing toolbox.

KEYWORDS

Grid Computing, Job Scheduling, Job Grouping.

1. INTRODUCTION

Grid computing, originally motivated by wide-area sharing of computational resources [1], has evolved to be mainstream technologies for enabling large-scale virtual organization [2]. During the beginning to mid 1990's distributed computing was holding a big account on the research projects. One of the researches on it was that to develop a tool that will allow it to act like a single big computer. Ian Foster of the US department of energy's Argonne National labs and university of Chicago has given a demonstration to create one super "meta computer" called I way.The term "Grid" refers to systems and application that integrate resources and services distributed across multiple control domains [3].Computational grids provide large-scale resource sharing, such as personal computer, clusters, MPPs, Data Base, and online instructions, which may be cross-domain, dynamic and heterogeneous[4]. Grid computing needs to support various services:

DOI: 10.5121/ijgca.2012.3405

security, uniform access, resource management, job scheduling, application composition, computational economy, and accounting. In a Grid computing environment, scheduler is responsible for selecting the suitable machines or computing resources in Grid for processing jobs to achieve high system throughput, but there exist several applications with a large number of lightweight jobs [5].Job scheduling with light weight gives low performance in terms of processing time and communication time. So to achieve high performance, jobs are to be scheduled in group instead of light weight jobs. The main purpose of this paper is to develop an efficient job scheduling algorithm to maximize the resource utilization and minimize processing time of the jobs, how they are grouped and allocated to resources in dynamic environment. This paper is organized as follows section 2, discusses related work, section 3 the scheduling activity, section 4 the proposed Algorithm, section 5 describes our experiments and the results and section 6 gives the conclusion of the paper.

2. RELATED WORK

The paper proposes [5] an algorithm which is designed for minimize overhead time and computational time. It is a big challenge to design an efficient scheduler. There exists some grouping based scheduling and non grouping based scheduling. To minimize total processing time by reducing overhead time and computational time grouping based job scheduling is used. On the other hand maximizing resource utilization, without grouping based scheduling is used. Overall processing time is reduced with the help of Modified grouping based job scheduling in computational grid.To achieve better performance the concept of grouping based job scheduling is extended.

An agent based dynamic resource scheduling strategy [6] focuses on maximize the processing time of jobs. The process of selecting a job is based on maximum heap tree. The processed outsource model is a hierarchical two layer approach in which top layer is called grid level an d other is called cluster level. In this model with FCFS-Job Grouping Strategy has of two major parts, first part provides resource management, which selects highest computational power cluster at the grid level and the second part provides FCFS-job grouping strategy that makes efficient utilization of the selected cluster by submitting a matching group of jobs from a FCFS job queue. But this paper does not consider bandwidth and file size constraint except computational power of the resource.

In paper [7] adaptive fine grained job scheduling algorithm (AFJS) is described which focuses on scheduling lightweight jobs. Before this algorithm, many other algorithms were developed which considered either the application characteristics or resource characteristics but not both. AFJS algorithm is the algorithm that considers both the characteristics. This algorithm starts with obtaining information about the resources and the resource monitoring mechanism is based on GRIM prototype and GRIR protocol. The algorithm has a constraint which says that the processing time of the coarse-grained job should not exceed the expected time and to ensure that the execution of a parallel program is faster than sequential execution, the calculation time should exceed communication time.

In paper author proposed a grouping based fine grained job scheduling algorithm [8] starts with obtaining information about the resources. In this light weight jobs are grouped as coarse grained jobs. The grouping based algorithm utilized algorithm utilized resourced efficiently of integrates greedy algorithm of FCFS algorithm. This model reduces the total processing time of jobs, maximizes the utilization of the resources and reduces the network latency. The time complexity scheduling algorithm is very high. It does not pay any attention to memory size constraint and preprocessing time of job grouping is high.

In bandwidth aware job grouping based scheduling algorithm the job grouping concept [9] is explored coupled with bandwidth aware scheduling. This algorithm focuses on grouping independent jobs having small processing requirement into jobs with larger processing requirements and then schedules them according to network conditions. The concept of bandwidth was used for performing load balancing at stream control transmission protocol (SCTP) layer. Its main objective was to provide the in-order delivery over multiple paths. This approach reduces total job processing time as compared to job scheduling without grouping.

A Dynamic job grouping based scheduling algorithm [10] group the jobs according to MIPS of the resource. It selects resource in first come first serve order. It selects jobs and group the jobs and assigns the group jobs in FCFS order and compare to resource if group job MI is less that to resource MIPS of this process continues until the resource MIPS is less to group job.

Scheduling framework for Bandwidth-aware strategy [11] schedules jobs in grid systems by taking of their computational capabilities and the communication capabilities of the resource's into consideration. It uses network bandwidth of resources to determine the priority of each resource. The job grouping approach is used in the framework where the scheduler retrieves information of the resources processing capability. The scheduler selects the first resource and groups independent fine-grained jobs together based on chosen resources processing capability. These jobs are grouped in such a way that maximizes the utilization of the resources and reduces the total processing time. After grouping, all the jobs are sent to the corresponding resource's whose connection can be finished earlier which implies that the smallest request is issued through the fastest connection giving best transmission rate or bandwidth. However, this strategy does not take dynamic characteristics of the resources into account, and preprocessing time of job grouping and resource selection are also high.

Hierarchical job scheduling approach used two levels Scheduling [12]global scheduling & local scheduling .The global scheduler uses separate queues for different type of the jobs for scheduling with the FCFS,SJF and first fit (FF) and the local scheduler uses same queue for different type of the jobs.

Optimal Resource Constraint (ORC) scheduling algorithm [13] includes the combination of both the Best fit allocation and Round Robin scheduling to allocate the jobs in queue pool. This algorithm improved the efficiency of load balancing and dynamicity capability of the grid resources.

3. SCHEDULING ACTIVITY

- Find available parallel computing resources from the defined parallel configuration (job manger).
- Create job object in the scheduler and in the client.
- Create new task in job.
- Calculating the minimum MI among the jobs.
- Sort the jobs in array with minimum MI and the Shortest time job is executed first
- The time slot is defined for all the jobs to be executed and the Job with higher MI above time slot allocated to it executed using round robin algorithm.
- The time required for each job using all three algorithms is plotted on the graph
- The execution time required for all jobs using all three algorithms(Shortest Job First, First Come First Serve, Round Robin) is plotted which is less than the execution time required for all jobs using only one algorithm (Shortest Job First).

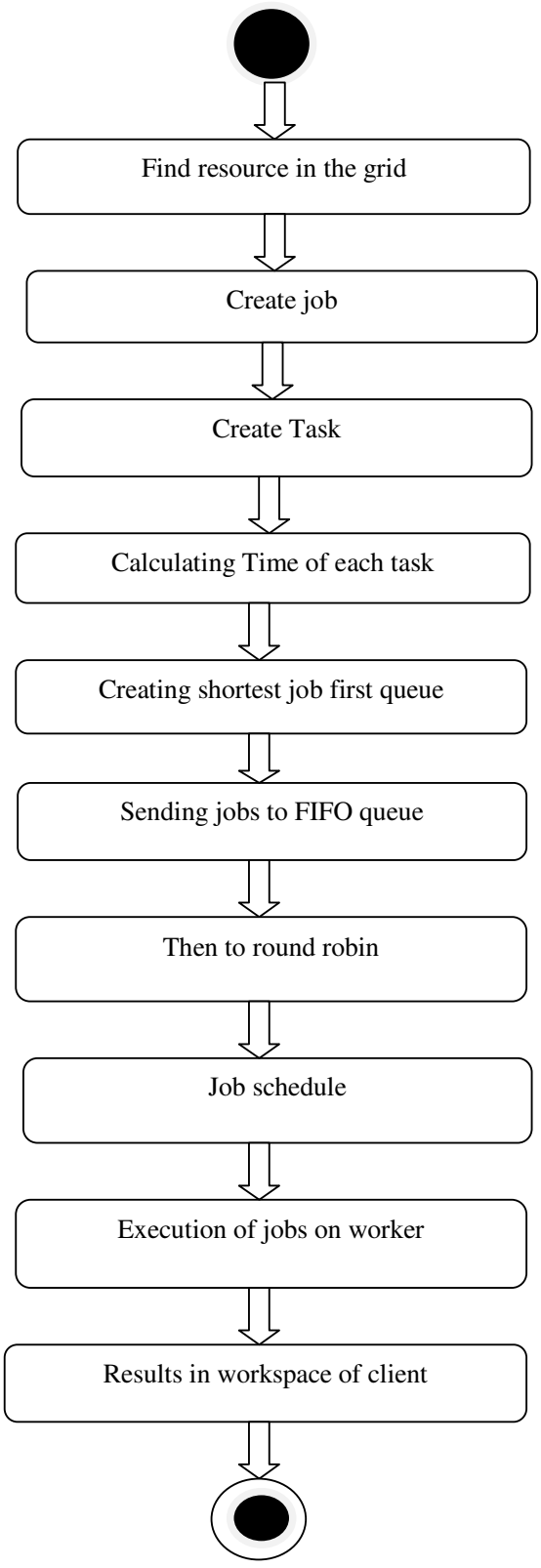

Figure 1. Scheduling Activity

4. PROPOSED ALGORITHM

4.1. Algorithm Steps

1. find Resource: Find available parallel computing resources
2. Create Job: Create parallel job object as many as your requirement
3. Create Task: create task for job to be performed for each job.
4. tic; %calculating execution time for each job start time
5. create Task(job1, @rand, 1, {{3,3}}); %for example
6. time_1=toc; %end time
7. time_array=[execution times of all jobs in array]
8. job_array=[parallel job objects in array];
9. g=[1 2 3 4 5];
10. shortest_job=[];
11. ----------Shortest job first----------
12. for k=1:No. of jobs
13. shortest_job_time=min(time_array); %min execution time job
14. for i=1: No. of jobs
15. if(execution time of jobs in array is min);
16. Then creating array of jobs with minimum MI jobs
17. l(k)=g(i); %for displaying the job no. that had been executed
18. And making the minimum MI job maximum to get next min value
19. end;
20. end;
21. end;
22. ----------FIFO and Round Robin----------
23. timeout=0.02; %time slot for each job
24. round_robin=[];
25. for i=1: No. of jobs
26. if(time slot allocated to each job is greater than execution time of respective job)
27. tic;
28. send the job to the job scheduler for computing
29. execution_time(i)=toc;
30. w(i)=shortest_job(i); %array of jobs executed within allocated time slot
31. end;
32. end;
33. for i=1: No. of jobs
34. if(time slot allocated to each job less than execution time of respective job from original job array)
35. tic;
36. send the job to the job scheduler for computing
37. execution_time(i)=toc;
38. x(i)=m(i); % array of jobs executed out of allocated time slot
39. end;
40. end;
41. round_robin=[array of combination of jobs executed within and out of allocated time slot];

4.2 Description of the Algorithm

The find Resource function performs two functions firstly it identifies the available parallel computing resources and job managers. Job managers create on object representing a job manager in your local MATLAB session. To find a specific job manager use parameter –value pairs for matching. For an example MyJobManager is the name of the job manager while MyJMhost is the host name of the machine running the job manager lookup service. The create job function creates parallel job as many as required using job manager configuration. The job is actually created on job manager but it executes in client session.

The Create task function defines the task to be executed by the job for each job. Tasks are assigned to the jobs, once the job is created by the job manager. Tasks define the function to be evaluated by the workers during the running of the job. The tic variable calculates the execution time for each job in the task from the start time. Create task has four parameters the first parameter defined the state of the job the second parameters defined its function the third parameters gives start time of the jobs and the fourth parameter gives the running time. In this example, each task will generate a 3 by 3 matrix of random numbers.

Execution time of each task in each job is calculated before it is send to queue of the job scheduler. The execution time of these jobs is stored in the array called time array. The variable time array is used for shortest job first using the time array, The variable 'g' is used for the sequence of jobs that will execute using all the three algorithms. The jobs are arranged in ascending order using the loop the minimum value is first stored in array in first place and that value is made maximum in order to get next minimum value in the array .The sorted jobs are stored in the variable called shortest job. Then these jobs are sent to the FIFO and Round_ robin queue. All the sorted jobs in shortest job array are passed through the queue as first in first out and are further sent to the next algorithm implementation. The time slot for each job to be executed is defined initially. In the first for loop, all the sorted jobs who's execution time is more than the time slot assigned is send to queue using the summit function. Now, in second for loop the job with time more than timeout is detected and is sorted in. This jobs whose time is more than the defined time slot is interrupted and execute finally. To plot time on the graph the final execution time required for all the jobs is stored in the variable 'final execution'. All the jobs executed within the timeout and interrupted and executed out off timeout are stored in the variable 'round robin' for plotting.

5. EXPERIMENTAL EVALUATION

5.1 Experimental setup and comparison

Simulation experiments have been carried out using MATLAB v7.4. The client should have the parallel computing toolbox for grid computing. If not should install the toolbox for parallel computing. The scheduler or job manger manages the job sent from the client and sends to the worker. The worker should have MDCE (Matlab Distributed Computing Server) install as administer. The MDCE server and its installation are explained further.

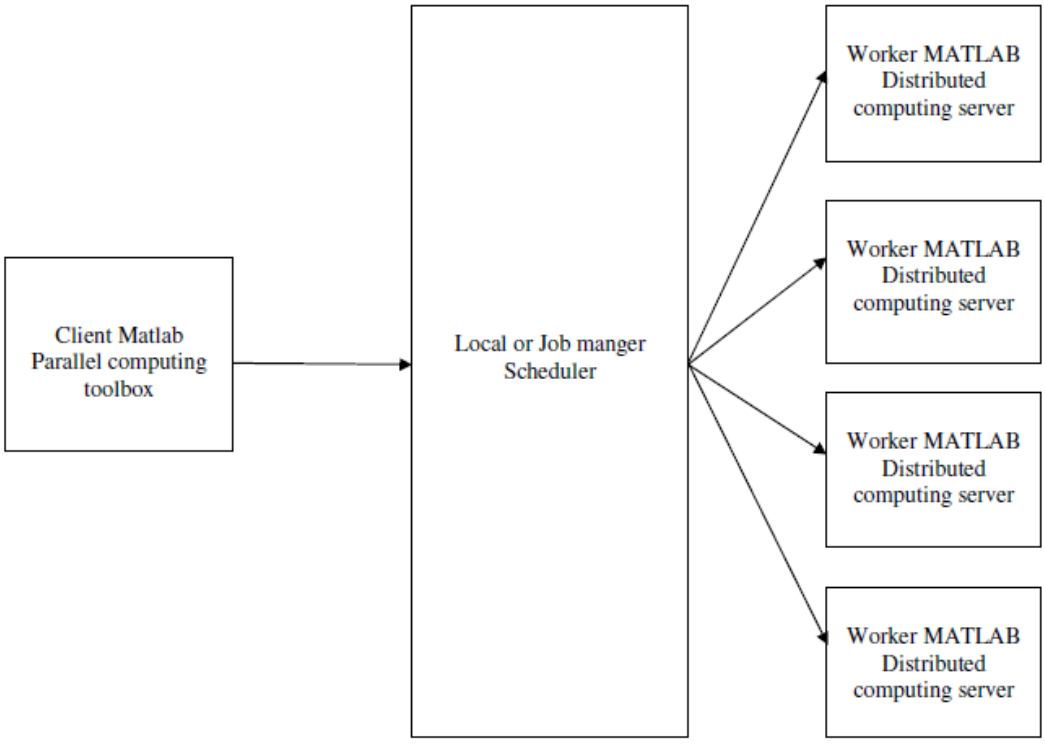

Figure 2. Parallel Computing Configurations

Table 1: Execution time Comparison

NO. OF JOBS	**PROCESSING TIME**	
	Hybrid algorithm	**FIFO algorithm**
Job 1	1.9405	2.0539
Job 2	1.7848	1.7968
Job 3	2.9623	3.3426
Job 4	2.8408	3.3287
Job 5	2.7098	2.9536

The above table shows the execution time comparison between the FIFO algorithm and all the three algorithms (Shortest time first, FIFO and Round robin).

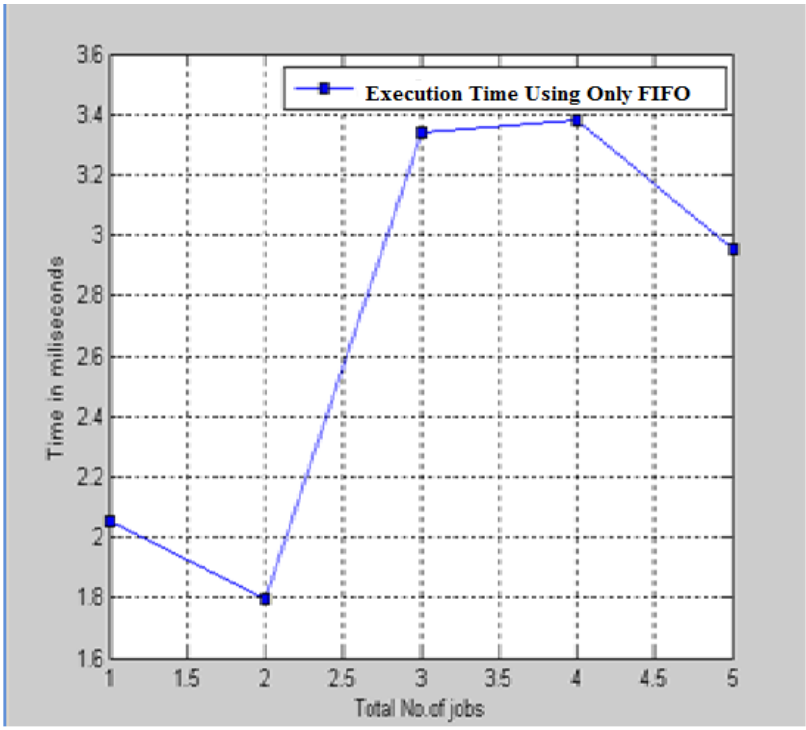

Figure 3.Execution time using FIFO algorithms

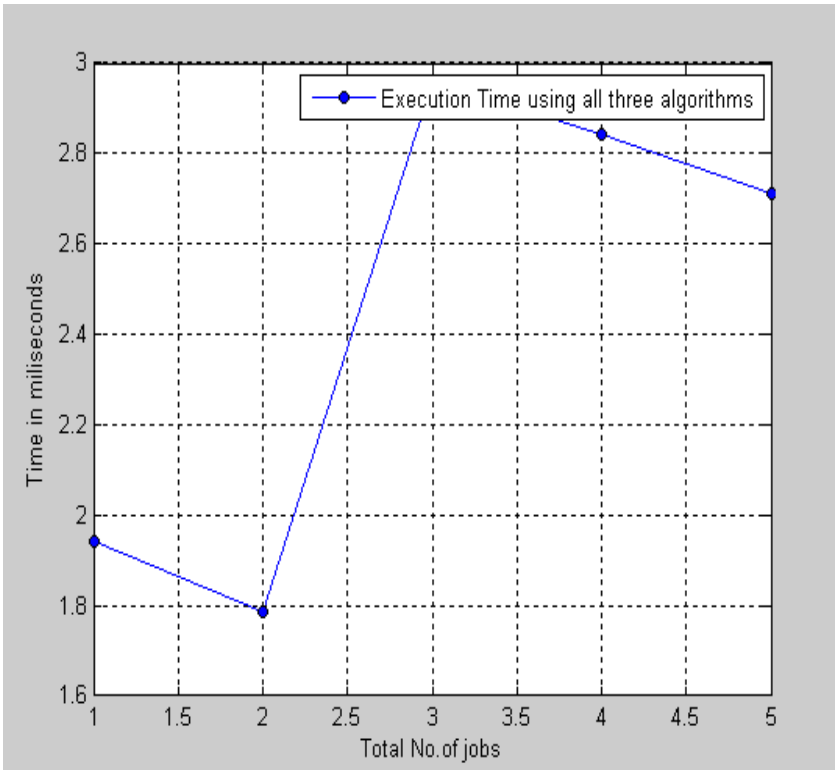

Figure 4. Execution time using all three algorithms (SJF, FIFO, RR)

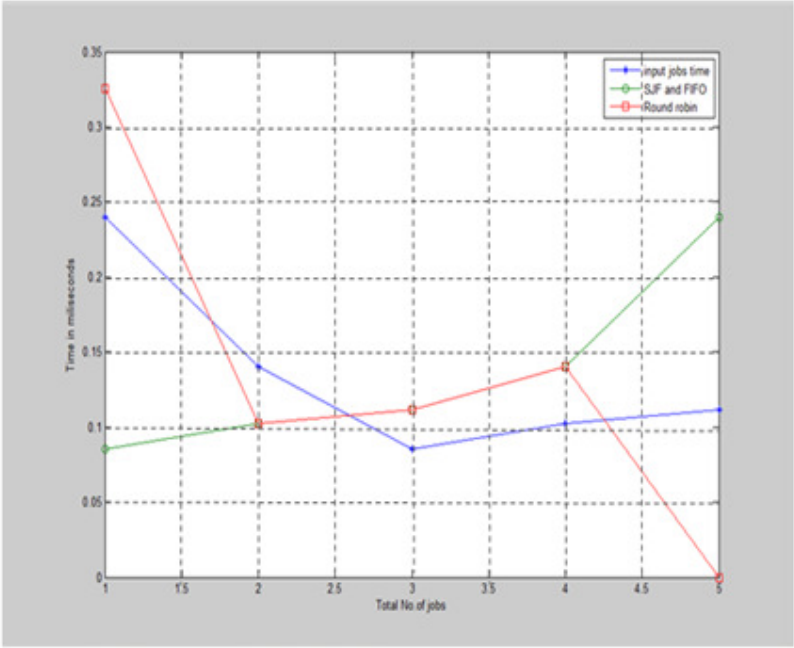

Figure 5. Implementation of all three algorithms with time constraints

Figure 3 shows the execution time required using only one algorithm (FIFO), the y-axis range is from 1.6 to 3.6. The time required is more than using all three algorithms as shown figure and the table also. Figure 4 show the execution time required using all three algorithms (Shortest Job First, First in First Out, Round Robin), the y axis range is from 1.6 to 3. The time required is less as shown in figure. From figure 2 and 3 we can conclude that time required for FIRST IN FIRST OUT ALGO is more than our hybrid algorithms, which is shown in figure 1 with their execution timings.

6. CONCLUSIONS

An algorithm is designed for an efficient job scheduling algorithm to maximize the resource utilization and minimize processing time of the jobs. We have proposed an efficient three scheduling algorithm (Shortest Job First, First Come First Serve, Round Robin) for jobs and send to the queue. We have got some better performance in terms of processing time than job scheduling on the FIFO algorithm. Also we have implemented Shortest Job First algorithm along with all three algorithms for performance analysis. The simulation results have shown that the proposed model is able to achieve the mentioned objectives in grid environment. However, allocating large number of jobs to one resource will increase the processing time. Therefore to avoid this situation during job grouping activity, the total number of jobs group should be created such that the processing loads among the selected resource are balanced. The comparative study also shows that the proposed hybrid algorithm gives better performance than shortest job first algorithm alone in terms of processing time. The proposed approach has been critically analyzed. Additionally, the simulation environment is semi-dynamic and it can't reflect in the real computational grid environment sufficiently that promotes further research in the proposed area. In future research , the resource can be managed by considering some more factors like current load of resource, network delay, QoS(Quality of Service) requirements' will be taken into account

ACKNOWLEDGEMENTS

The authors are thankful to everyone who supported or motivated us

REFERENCES

[1] I. Foster and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan-Kaufmann, 1998.

[2] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications, Vol. 15, No. 3, 2001.

[3] Foster, I. and Kesselman, C. Computation Grids. Foster, I. and Kesselman, C. eds. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 2-48.

[4] Ian Foster and Carl Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Elsevier Inc., Singapore, Second Edition, 2004.

[5] N. Muthuvelu, Junyan Liu, N.L.Soe, S.venugopal, A.Sulistio, and R.Buyya, "A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids," in Proc of Australasian workshop on grid computing, vol. 4, pp. 41–48, 2005.

[6] Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra,"An Agent Based Dynamic Resource Scheduling Model With FCFS-Job Grouping Strategy In Grid Computing", Waset, ICCGCS 2010. In Press.

[7] Quan Liu, Yeqing Liao, "Grouping-Based Fine-grained Job Scheduling in Grid Computing",Vol.1, pp.556- 559, IEEE First International Workshop on Education Technology and Computer Science, 2009.

[8] T.F Ang, W.K.Ng, T.C Ling, "A Bandwidth-Aware Job Grouping-Based Scheduling on Grid Environment",Information Technology Journal, vol .8, No.3, pp. 372-377, 2009.

[9] Nithiapidary Muthuvelu, Junyang Liu, "A Dynamic Job Grouping- Based Scheduling for Deploying Application with Fine-Grained tasks on Global Grids, Australasian Workshop on Grid Computing and e- Research, vol. 44, AusGrid -2005

[10] Ng Wai Keat,Ang Tan Fong,Ling Teck Chaw,Liew Chee Sun,"Scheduling Framework For Bandhwidth-Aware Job Grouping-Based Scheduling In Grid Computing",Vol.19(2), Malaysian Journal Of Computer Science, 2006.

[11] J. Santoso; G.D. van Albada; B.A.A. Nazief and P.M.A. Sloot:"Hierarchical Job Scheduling for Clusters of Workstations", ASCI 2000, pp. 99-105. ASCI, Delft, June 2000.

[12] K.Somasundaram, S.Radhakrishnan, "Node Allocation In Grid Computing Using Optimal Resource Constraint (ORC) Scheduling",VOL.8 No.6, IJCSNS International Journal of Computer Science and Network Security, June 2008

Authors

Pinky Rosemarry has received her B.Tech degree in CSE (Computer Science & Engineering) in 2010 from Ajmer Institute of Technology, Ajmer affiliated to Rajasthan Technical University, Kota and M.tech from Govt. Engineering College, Ajmer in 2012 (RTU-Kota). She has presented several papers in international & national conferences. Her research interest includes Grid Computing.

Mr.Ravinder Singh (Assistant Professor) received his B.E (Computer Engineering)from Sobhasaria engineering college, Sikar in 2006.M.E (CSE) from, Motilal Nehru National Institute of Technology, 2009 India. He has three years teaching experience. He is presently working as Assistant Professor (CS & IT) in Govt. engineering College, Ajmer. He has several papers published in international and national journals/conferences. He is the guide of various projects of B.Tech. students and also the guide of more than 5 students of M.Tech. thesis.

Payal Singhal has received her B.Tech degree in CSE (Computer Science & Engineering) in 2010 from Ajmer Institute of Technology, Ajmer affiliated to Rajasthan Technical University, Kota and M.tech from Govt. Engineering College, Ajmer in 2012 (RTU-Kota). She has presented several papers in international & national conferences. Her research interest includes Grid Computing.

Mr. Dilip Sisodia (Assistant Professor) received his B.E (Computer Engineering) from Government Engineering College, Ajmer in 2006. He has five years teaching experience. He is presently working as Assistant Professor (CS & IT) in Govt. engineering College, Ajmer.He is the guide of various projects of B.Tech. students.