Cost-based Job Grouping and Scheduling Algorithm for Grid Computing Environments

Sonal Yadav
M.Tech (CSE)
Sharda University
Greater Noida, India

Amit Agarwal, Ph.D
Associate Professor
University of Petroleum & Energy Studies, India

Ravi Rastogi, Ph.D
Associate Professor
Sharda University
Greater Noida, India

ABSTRACT
The integration of remote and diverse resources and the increasing computational needs of Grand challenges problems combined with faster growth of the internet and communication technologies leads to the development of global computational grids. Grid computing is a prevailing technology, which unites underutilized resources in order to support sharing of resources and services distributed across numerous administrative region. An efficient and effective scheduling system is essentially required in order to achieve the promising capacity of grids. The main goal of scheduling is to maximize the resource utilization and minimize processing time and cost of the jobs. In this research, the objective is to prioritize the jobs based on execution cost and then allocate over the resources with minimum cost by merging it with conventional job grouping strategy to provide solution for better and more efficient job scheduling which is beneficial to both user and resource broker. The proposed scheduling approach in grid computing employs a dynamic cost-based job scheduling algorithm for making efficient mapping of job to available resources in grid. It also improves communication to computation ratio (CCR) and utilization of available resources by grouping the user jobs before resource allocation.

General Terms
Algorithm, GridSim.

Keywords
Grid computing, Job Scheduling, Job Grouping.

1. Introduction
The widely growing popularity of the Internet/ Web and the availability of the powerful computers and high speed networks as the low cost commodity components are improving the way we perform computing and use computers [1]. Regardless of these improvements, there are many situations where computational resources fail to keep up with the demands placed on them. A solution for many of these emerging problems is Grid computing **[1]**. *Grid computing* has emerged as an evolutionary computing from the existing distributed computing systems for delivering information, resources and services to the user on demand. Today many computing resources distributed geographically are idle and underutilized. The objective of the Grid computing is to group these resources into a single system. It helps to solve problems that are too complex in the field of science, engineering and research. Grids are very large-scale virtualized, distributed computing systems. They bind multiple administrative domains and form virtual organizations. Such organizations can share their resources collectively to create an even larger grid. The name 'Grid' is similar with the electricity grid. Users can obtain a resource as utility such as electricity (on pay per use basis), or in this case computer processing from a variety of resources to fulfil their needs [2]. The goal is to provide users with access to the resources according to their need [2]. In order to achieve the assured benefits of distributed resources, effective and efficient scheduling algorithms are required. Task scheduling is the NP-complete problem in grid computing environment **[11]**. In this research work, the user's jobs are prioritized on the basis of job profit (job execution cost) in descending order. In order to gain maximum profit, jobs with higher profit can be executed on minimum cost based resource. The resource with minimum cost is selected and the higher priority jobs are scheduled on it. After prioritization of jobs, the job grouping strategy is applied. In job grouping strategy, the scheduling of application with a large number of small processing requirement (fine grained) job is done and they are converted to coarse grained jobs. The fine grained jobs are also known as light weight job that has few lines of code or very basic arithmetic expressions [5]. The transmission and processing overhead increased drastically due to sending of individual jobs. To overcome these scheduling issues, a cost based dynamic job prioritization and grouping scheduling strategy is introduced in order to minimize the processing time and cost and to achieve full utilization of the resources.

This paper is organized as follows. Section 2 is an overview of related work about job scheduling in Grid environment. Section 3 presented scheduling architecture and scheduling activity. Our proposed model and job scheduling algorithm are presented in Section 4. Section 5 contains simulation environment, experimental set up and results. Finally, Section 6 and 7 gives conclusion and references.

2. Related Work
There are several algorithms and approaches in the area of job scheduling in Grid. In [2], Selvarani and Sadhasivam presented an improved cost based algorithm for task scheduling in cloud computing. They focused on grouping the task while scheduling in cloud computing platform, where resources have different execution costs and computation performance. Due to job grouping, the communication to computation ratio (CCR) can be improved by grouping the user tasks according to a particular cloud resource's processing capability. In [3], N.Muthuvelu et. al proposed a grid job scheduling algorithm based on parameterized job-grouping strategy, which is adaptive to runtime environment. An average analysis is implemented by grid resource broker in order to determine the current status of the grid before performing the job grouping method.

In [4], Amudha and Dhivyaprabha presents a paper in which Grid scheduler first allocates the high priority jobs to the resources and then it allocates the low prioritize job so as to achieve the maximum resource utilization rate, minimize the makespan and avoid the load balancing level problem.

In [5], Q. Liu and Y.Liao feels that there is a need to reduce the communication time, processing time and enhance resource utilization in case of scheduling the light-weight or small jobs. The light weight jobs are grouped and then allocated to resource. This grouping algorithm integrated Greedy algorithm and FCFS algorithm to improve the processing of fine-grained jobs. The algorithm considers the dynamic characteristic of the grid environment. It does not pay any attention to memory size constraint and pre-processing time of job grouping is high.

In [6], Keat et al. proposed a scheduling framework for Bandwidth-Aware Job Grouping Based strategy that groups the jobs according to MIPS and Bandwidth of the resource, but the algorithm deals with two deficiencies. First is grouping strategies does not utilize resource sufficiently, and second, consideration of bandwidth strategy is not efficient to transfer the job. T.F.Ang et al. presents Bandwidth-Aware Job Grouping-Based scheduling strategy, that groups the jobs according to the MIPS and bandwidth of resources, but shortcomings of the algorithm is first, the model sends group jobs to the resource whose network bandwidth has highest communication or transmission rate, but the algorithm does not ensure that resource having a sufficient bandwidth will be able to transfer the group jobs within required time [7].

In [8], Muthuvelu et al. proposed a dynamic job grouping-based scheduling algorithm that groups the jobs according to MIPS of the available resources. This model reduces the processing time and communication time of jobs, but the algorithm doesn't take the dynamic resource characteristics into account. This strategy dynamically assembles the individual Fine-grained jobs of an application into a group, and sends these coarse-grained jobs to the Grid resources. This dynamic grouping strategy based on the processing requirements of each application, Grid resources availability and their processing capability and granularity size. In [9] Rosemarry et al. states that, allocating large number of jobs to one resource will increase the processing time. So 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 balance.

A described model presented by Soni et al. in [10] obtains the information about resources and job. This information is used for job grouping and for resource selection. When the jobs are put into a group according to the selected resources, the grouped job is dispatched to resources for computation. GBJS gives better performance than AFJS and DJGBSDA in terms of processing time. In [13], Soni et al. proposed a Constraint-Based Job and Resource scheduling in Grid Computing. This paper focuses on small jobs scheduling in Grid Computing. Constraint-Based Job and Resource scheduling (CBJRS) algorithm is proposed which will reduce the processing time, processing cost and enhance the resource utilization in comparison to other algorithms.

In [14], Zheng, G. and Liu proposed A selective algorithm based on multiple QoS constraints for grid task scheduling they propose an adaptive selective scheduling algorithm that based on multiple QoS constraints, and name as M-QoS selective algorithm. This new algorithm considers multiple QoS-requiring for the service resources and tasks, and divides different QoS constraint tasks into multiple groups.

In our proposed algorithm we combine the priority and grouping strategy in such a way that the jobs are prioritize according to the cost, then grouped and send to the resource with minimum cost in order to provide benefit to the user as well as resource broker. Using job grouping algorithm for scheduling after prioritization, the processing time and in turn the cost is reduced over the algorithm without job grouping and with job grouping (without prioritization) [8].

3. Scheduling Architecture Model

3.1 Scheduling architecture

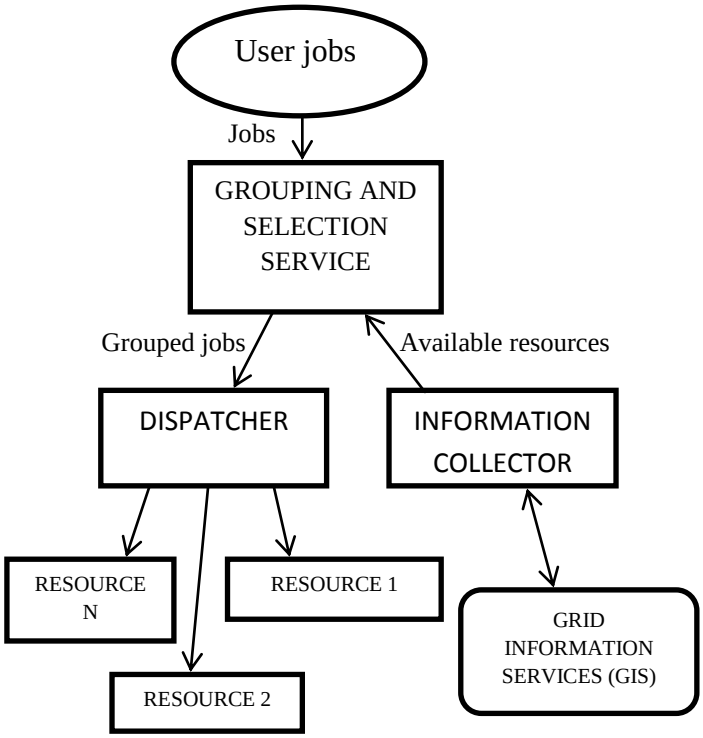

Fig. 1: Scheduling Architecture

The details of various elements in scheduling architecture are:

User: The user submits the grid application to the resource broker for processing [15].

Resource Broker: A Resource Broker is a grid portal that allows trusted users to create and handle computational Grid, by exploiting a simple and friendly web based GUI.

The resource broker is made up of no. of components:

- A scheduler
- A dispatcher
- Information collector
- Grouping and selection service.

The job scheduler is a service that resides in a user machine. When the user creates a list of jobs in the user machine, these jobs are sent to the job scheduler for scheduling [15]. The job scheduler obtains information of available resources from the Grid Information Service (GIS).The information collector collects information from the Grid Information Service (GIS). It assembles the resource availability and processing capability to the resource information table. The grouping and resource selection service is responsible for grouping of job based on information collected by the information collector from GIS. The dispatcher acts as a sender that sends grouped jobs to their respective resources [12].

Grid Information Service (GIS): GIS keeps the information of all resources that is needed by the broker for scheduling the jobs in efficient way. Then the resource broker chooses the best resource for the job to be executed. The information about the status of resource is updated, after successful completion of job. This procedure is repeated until all the jobs are assigned to the resources.

3.2 Scheduling activity

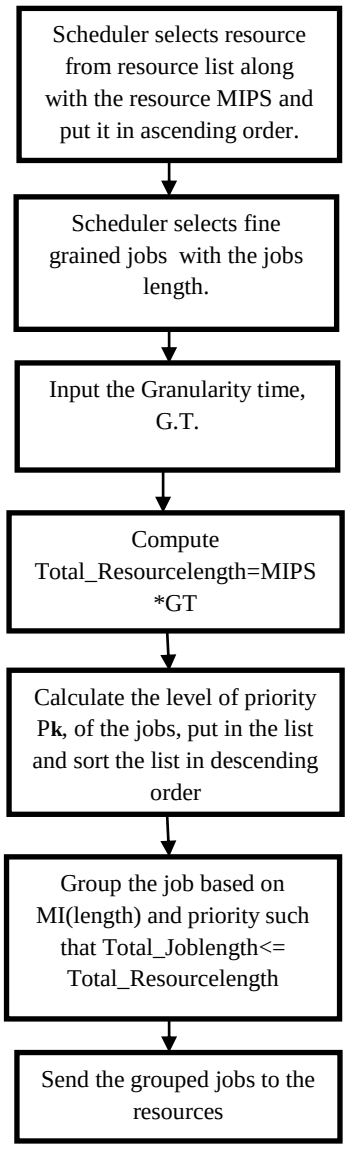

Fig. 2 Scheduling Activity Model

4. Proposed Model

The proposed model consists of two parts:

1) Cost based job prioritization model.
2) Job grouping scheduling model.

4.1 Cost based job prioritization model

1) Scheduler selects the resource with minimum cost and high priority jobs are scheduled on it.
2) In our work, cost of every individual resource is different.

The priority level of job is calculated using equation 1.

$$P_k = (L_k \ / \ R_{r,k}) * RC_{r,k} \quad \textbf{(1)}$$

$R_{r,k}$: The processing capability of rth resource used by the kth job.

$RC_{r,k}$: The cost of the rth individual use of resources by the kth job.

L_k : Length of the kth job.

P_k : The priority or cost of the kth job.

4.2 Job grouping scheduling model

The job grouping scheduling takes into account:

1) The length of the job (MI).
2) Grouping strategy is based on processing capability (MIPS) of available resources.
3) Granularity time (GT) defined by user.

Jobs are grouped according to the capability of the selected resource.

For e.g., the condition for the grouping strategy is that the total job length should not exceeds the total resource length [8].

Total_joblength<=Total_resourcelength

Here, Total_joblength is the total length (MI) of the grouped jobs [8].

Total_resourcelength = Resource_MIPS* GT

MIPS is the processing capability of the resource calculated as:

Resource_MIPS = total number of processing elements (PE) * MIPS (million instruction per second) of PE [8].

The job grouping is done based on a particular granularity time (GT). Granularity time is a user defined parameter which is used to measure the total number of jobs that can be completed within a specified period of time.

4.3 Proposed algorithm

The scheduler receives the number of tasks to be scheduled on number of available resource. Grid Information Service (GIS) registers the information of Grid resource. The job scheduler requests resources information from GIS. GIS sends the information of available resources to the scheduler.

4.3.1 Cost Based Job Prioritization Algorithm

1) For all available jobs, calculate their priority according to equation 1.
2) Calculate the priority of every coming job and put it in the list.
3) Sort the jobs based on priority in descending order.
4) Allocate high priority jobs to minimum cost resource using Job grouping scheduling algorithm.

4.3.2 Job Grouping Scheduling Algorithm

1) Initialize Joblist, Resourcelist, Targetlist to zero
2) Input the MIPS of resource.
3) The length (MI) of each job is generated.
4) Sort the resource list in ascending order based on cost of the resource (low to high).
5) Multiply the MIPS of the jth resource with granularity time as specified by user to get TOT_resourcelength.
6) Sort the job list in descending order based on priority of jobs (high to low)
7) If TOT_resourcelength is less than joblength.
8) The job cannot be allocated to the resource.
9) Input the MIPS of next resource.
10) GOTO step 6
11) If the TOT_resourcelength is greater than Joblength execute step 11.1 to 12.1 while

11.1) Total length of all jobs is less than or equal to the TOT_resourcelength.
11.2) Add previous total length and current Joblength and assign to current TOT_Joblength.
11.3) Get the length of next job.

12) If the total length exceeds TOT_resourcelength

12.1) Subtract the length of job from TOT_Joblength

13) If the TOT_Joblength is not zero then repeat steps 13.1 to 13.4

13.1) Create a new job group of length equal to TOT_Joblength

13.2) Allot a unique ID to the newly created job group.
13.3) Insert the job group into a new job group list.
13.4) Insert the allocated resource ID into the Targetlist of each grouped job.

14) Repeat the above until all the jobs in the list are grouped into job-groups.
15) When all the jobs are grouped and assigned to a resource, send all the job groups to their corresponding resources list of grouped jobs.
16) After the execution of the job-groups by the assigned resources send them back to the Targetlist.

5. Simulation & Experimental Setup

5.1 Grid simulation environment

Grid computing environment is dynamic in nature, so studying the environment and performing repeated experiment is a tedious job. To solve the problem, Gridsim simulation toolkit is used. The Gridsim toolkit [1] provides means for the modeling and simulation of various Gridsim entities like resources, Gridlets (jobs), schedulers, users (single or multiple) and network connectivity with different capabilities, configurations and domains. The Gridsim also supports modeling and simulation of broad range of heterogeneous resources like multiprocessors, SMPs and distributed memory machines such as PCs, workstations etc.
Application schedulers in Grid environment, called resource brokers, perform resource discovery, selection and aggregation of various set of distributed resource for an individual user[1].
A Gridsim toolkit 5.2 is used to build the Grid environment. The Grid environment is simulated using My Eclipse 6.0 by writing java code and implementing various Grid entities. The various Grid entities that used in this research work, are initialized by initializing the Gridsim package.

Create Gridlet( ): This method will create Gridlets statically and dynamically.

Create GridResource(): This method creates one Grid resource. A Grid resource contains one or more machines. Similarly, a Machine contains one or more PEs (Processing Elements or CPUs).

Create grid user( ): Single or multiple user can be created. Each user must have a unique id. In our research work, only one user is taken into account.

5.1.1 Input parameters

Gridlets : The number of gridlets.

A_MI : Average gridlet length in MI.

Deviate% : MI deviation percentage used to create different number of gridlets that have different lengths.

GT : Granularity time is defined by user. It is a measure of number of jobs that can be completed within a particular time.

OH_Time : Gridlet overhead time. In real world, overhead time depends on current network, load and speed.

Resources : Resources are selected from the resource list.

5.1.2 Performance metrics

In this work, two performance metrics are used.

Total processing time

The total processing time [8] is calculated in seconds based on:

- Overhead processing time each Gridlets.
- Time taken for performing Gridlet grouping process.
- Time taken for sending Gridlets to the resources.
- Time taken for processing Gridlets at the resources.

- Time taken for receiving back the processed Gridlets.

Total processing Cost
The total processing cost [8] is calculated based on:

- Cost rate associated with each Grid resource.
- Total processing time for job execution.

5.2 Experimental setup

Three cases are considered in our research work :

Case1: Simulation without job grouping.

Case2: Simulation with job grouping.

Case3: Simulation with cost based job prioritization grouping scheduling.

The above three cases are tested using seven different MIPS and cost per resource

Table 1: Grid resource setup for the simulation.

Resource ID	MIPS	Cost per sceond
R1	150	100
R2	110	200
R3	240	300
R4	320	350
R5	210	400
R6	300	450
R7	410	500

Simulations are conducted to analyse and compare the difference between the three cases. The input parameters to the simulation are as follows:

- Total number of Gridlets :25-150
- Average MI(A_MI) of Gridlets : 200
- MI(D_%) deviation percentage: 20%
- OH_Time: 10 sec

GridSim Random function provides static methods for incorporating randomness in data used for this simulation.

Experiment 1: Simulation without job grouping
In this experiment, jobs are coming in a sequential manner and each job is allocated to the resource one by one. The job length varies between 200 to 240.

Experiment 2: Simulation with job grouping
For simulation with job grouping, all the above input parameters are same. In this approach, the jobs are not send one by one rather they are send in groups. The Granularity time is 10 secs. Each incoming job is grouped and then send to the corresponding resource such that,

Total_joblength<=Resource_MIPS*GT

Table 2: Simulation without job Grouping

No. Of Gridlets	Without job Grouping	
	Process_cost	Process_time
25	11740	51
50	23620	104
75	36040	158
100	45780	210
125	60040	262
150	69400	310

Table 3: Simulation with job Grouping(without prioritization)

No. of Gridlets	With Grouping (without prioritization)		
	Number of Groups	Process_Cost	Process_Time (Sec)
25	1	400	4
50	1	1100	9
75	2	1900	13
100	2	2700	16
125	3	3300	18
150	3	4200	21

Experiment 3: Simulation with cost based job prioritization algorithm

In our proposed approach i.e., cost based job prioritization algorithm, the jobs are prioritized on the basis of jobs profit (job execution cost) in descending order. The resource list is to be sorted in ascending order and then the job grouping scheduling algorithm is applied to send grouped jobs to the resource.

The cost based job prioritization proves slightly better than job grouping scheduling algorithm

Table 4 : Simulation with cost based job prioritization grouping scheduling algorithm.

No. of Gridlets	Cost based job prioritization algorithm		
	Number of Groups	Process_Cost	Process_Time (secs.)
25	1	400	4
50	1	800	8
75	1	1200	10
100	2	1500	14
125	2	1900	18
150	3	2800	20

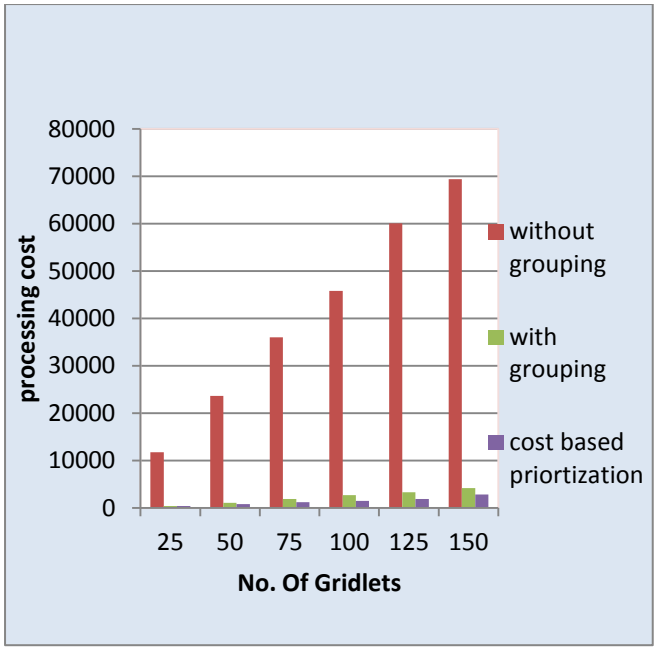

Fig. 3: Comparison between processing cost and Gridlet size with granularity time of 10 sec

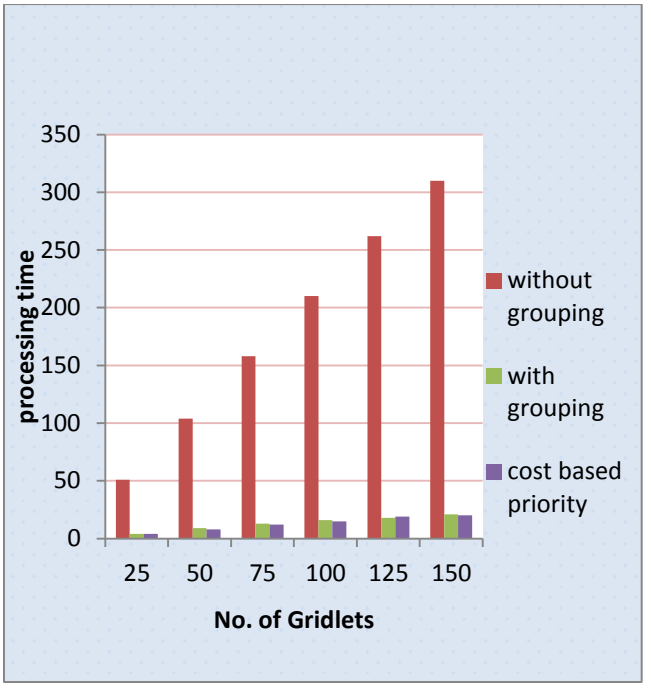

Fig. 4: Comparison between processing time and Gridlet size with granularity time of 10 sec

5.3 Experimental Results

Simulations were conducted with a granularity time of 10 seconds, the Gridlets average MI of 200 in a virtual Grid environment using Gridsim toolkit [1] and comparison of the algorithms is done in terms of processing time and cost. The comparison is done between the proposed cost based prioritization with without job grouping [8] and with job grouping [8] scheduling algorithms. The jobs are independent in nature and the input parameters are given dynamically.

5.3.1 Observation 1

We observed that the processing time and cost are higher when fine grained jobs are send individually without grouping. On the other hand, when individual jobs are grouped into a number of coarse grained jobs the processing time as well as cost reduces. In fig. (3) and fig. (4) the processing cost and processing time of the without grouing algorithm increased drastically as compared to with grouping algorithm and the proposed algorithm.

5.3.2 Observation 2

We observed that the processing time and cost is improved by applying grouping strategy but jobs are grouped on First come First serve basis. In our proposed algorithm jobs are prioritized based on the job execution cost. So, the higher priority jobs will be executed first thus satisfying the user's requirement. In fig. (3) and fig. (4) the processing cost and processing time of the with grouping algorithm is higher as compared to the proposed algorithm.

6. Conclusion and Future Work

When job grouping scheduling strategy is combined with cost based job prioritization scheduling the processing time as well as processing cost is reduced. In job grouping, without prioritization the jobs are grouped on first come first serve basis in a sequential manner but in cost based prioritization scheduling the jobs with highest priority are grouped first and allocated to the resource with minimum cost. This type of scheduling is beneficial to both user and resource broker in grid.

For the future work the QoS (Quality of Service) parameters can be taken into account before performing the grouping method. Moreover in our research work we considered the independent task in future dependent task can also be considered.

7. References

[1] Buyya. R, and Murshed. M, " GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", pp. 1-37, 2003.

[2] Selvarani. S, and Sadhasivam. G.S, "Improved cost-based algorithm for task scheduling in Cloud computing", Computational Intelligence and Computing Research (ICCIC), IEEE, pp.1-5, 2010.

[3] N. Muthuvelu, I. Chai, and C.Eswaran, "An adaptive and parameterized job grouping algorithm for scheduling grid jobs", International conference on Advanced communication technology, pp. 975-980, 2008.

[4] Amudha. T, Dhivyaprabha. T.T. "QoS Priority Based Scheduling Algorithm and Proposed Framework for Task Scheduling in a Grid Environment", IEEE-International Conference on Recent Trends in Information Technology, pp.1-6, 2011.

[5] Q. Liu, Y. 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.

[6] Keat, N.W, Fong, A.T, Chaw. L.T, and Sun. L.C, "Scheduling framework for bandwidth-aware job grouping- based scheduling in Grid computing", Malaysian Journal of Computer Science, Vol. 19, pp. 117 – 126, 2006.

[7] 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.

[8] Muthuvelu. N, Liu. J, Soe. N.L, venugopal. S and Buyya. R, "A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global Grids". Australian workshop on Grid computing and e research Australian Computer Society, Inc . vol. 44, pp. 41-48, 2005.

[9] Rosemarry. P, Singh. R, Singhal. P, and Sisodia. D, "Grouping Based Job Scheduling Algorithm Using Priority Queue And Hybrid Algorithm in Grid Computing." International Journal of Grid Computing & Applications (IJGCA) Vol.3, No.4, December 2012.

[10] Soni. V. K, Sharma. R, and Mishra. M. K, "Grouping-Based Job Scheduling Model in Grid Computing." World Academy of Science, Engineering and Technology 41, 2010.

[11] Chauhan. S. S, and Joshi. R. C, "A Heuristic for QoS Based Independent Task Scheduling in Grid Environment", International Conference on Industrial and Information Systems, ICIIS(IEEE), 2010.

[12] P. Rosemarry, Singhal, P, and Singh, R. "A Study of various job & resource scheduling algorithms in Grid Computing" International Journal of Computer Science and Information Technologies(IJCSIT), 2012.

[13] Soni V.K., Sharma. R, and Mishra. M. K, and Das. S, "Constraint-Based Job and Resource scheduling in Grid Computing", IEEE, 2010.

[14] Zheng. G, and Liu. Y, "A Selective Algorithm Based on.Multiple QoS Constraints for Grid Task Scheduling" First International Conference on Intelligent Networks and Intelligent Systems (IEEE), 2008.

[15] Wang. Y, Hu. S, and Wang. G, "A Strategy of Resource Scheduling for Grid Computing Based On QoS". International Conference on Information Science and Engineering, 2009.