Comparative Analysis of Job Grouping based Scheduling Strategies in Grid Computing

Simrat Kaur
University Institute of Engineering & Technology
Panjab University, Chandigarh

Sarbjeet Singh
University Institute of Engineering & Technology
Panjab University, Chandigarh

ABSTRACT

Grid computing is a form of distributed computing that provides a platform for executing large-scale resource intensive applications on a number of heterogeneous computing systems across multiple administrative domains. Therefore, Grid platforms enable sharing, exchange, discovery, selection, and aggregation of distributed heterogeneous resources such as computers, databases and visualization devices. Job and resource scheduling is one of the key research area in grid computing. In a grid computing environment, a scheduler is responsible for selecting the best suitable computing resources in the grid for processing jobs to achieve high system throughput. Further, grouping the fine grained jobs according to the processing capability of available resources results in better throughput, resource utilization and low communication time. Motivation of this study is to encourage and help the amateur researcher in the field of grid computing, so that they can understand easily the concept of scheduling, job grouping and can contribute in developing more efficient and practical scheduling algorithm. In this paper, we compared three job grouping based scheduling algorithms that will benefit interested researchers to carry out further work in this thrust area of research.

Keywords

Grid Computing, Job scheduling, Job grouping

1. INTRODUCTION

The emergence of high speed networks has made it possible to share geographically distributed resources such as supercomputers, storage systems, databases and scientific instruments in order to gather, process and transfer data smoothly across different administrative domains. Aggregations of such distributed resources, called computational grids [1][2] provide computing power that has made it possible to solve large scale problems in science, engineering and commerce.

In a Grid computing environment, a scheduler is responsible for selecting the best suitable machines or computing resources in the grid for processing jobs to achieve high system throughput [3][4]. Typically, an application requires an execution set that consists of several jobs, where each job is considered as the atomic unit of computation .In the case of an application with a large number of jobs with small scale processing requirements, the total communication time between each job and the resource seems to be more than the total computation time of each job at the resource. However, coarse-grained jobs can be created by aggregating a suitable number of jobs at the user-level, and submitted the aggregated jobs to the scheduler for deployment. This, however, creates a programming burden on the application developer as he/she will have to be aware of the complexities of Grid environment. Alternatively, the small scaled jobs can be submitted individually. This option leads to high communication time and cost, since each small job is associated with transmitting and processing overhead time and cost. Therefore, there is a need for a scheduling strategy to group the jobs at the scheduling level according to the processing capabilities of the available resources, and proceed with the job scheduling and deployment activities [5]. In recent years, the researchers have proposed several efficient scheduling algorithms that are used in grid computing to allocate grid resources with a special emphasis on job grouping based scheduling.

This paper is organized as follows: Section II describes the concept of a basic job grouping framework in brief. Section III presents a literature review of job grouping based scheduling algorithms in grid computing. Section IV presents a detailed study of three job grouping based scheduling algorithms proposed by researchers in grid computing and Section V provides a comparison and simulation result among the three surveyed papers. Section VI presents conclusion of this paper and future work and lastly the references.

2. BASIC JOB GROUPING FRAMEWORK

When the user creates a list of jobs in the user machine, these jobs are sent to the job scheduler for scheduling arrangement. The information collector gathers resource information from the Grid information service (GIS). The grid information service (GIS) is a facility that provides information about all the registered resources in a grid. Based on the information, the job scheduling algorithm is used to determine the job grouping and resource selection for grouped jobs. Once all the jobs are put into groups with selected resources, the grouped jobs are dispatched to their corresponding resources for computation[6].

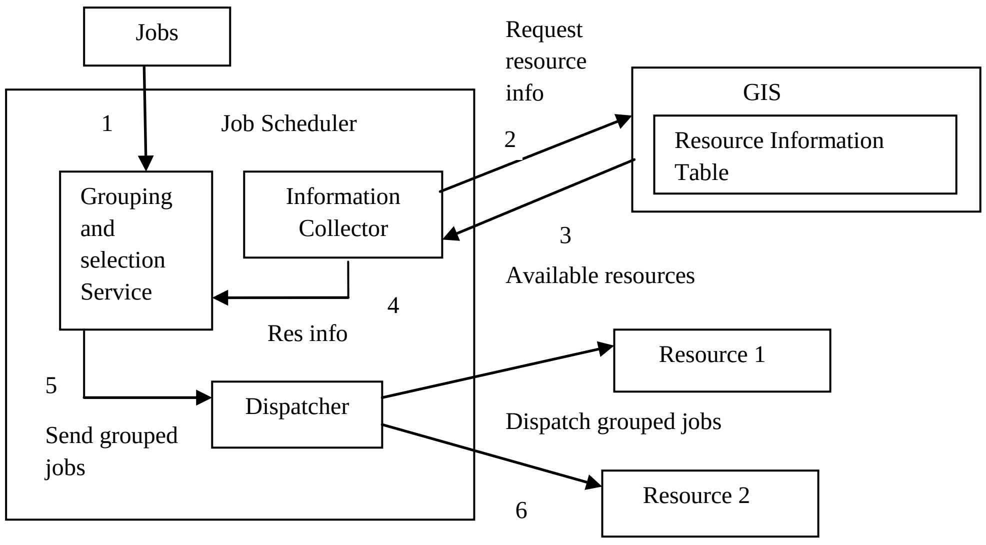

Figure 1. Framework for job Scheduling [6]

The scheduling framework illustrated in figure 1 depicts the design of the job scheduler and its interactions with other entities.

The grouping and selection service serves as a site where matching of jobs is conducted. The strategy for matching jobs is based on the information gathered from the information collector. There are two steps involved during the matching of jobs. They are job grouping and job selection. In the job grouping process, jobs submitted by the user to the scheduler are collected and they are grouped together based on the information of resources. The size of a grouped job depends on the processing requirement length expressed in Million Instructions (MI). At the same time, job selection is also being conducted where a grouped job corresponds to the resource in question. The process is performed iteratively until all the jobs are grouped according to their respective resources. The dispatcher functions as a sender that transmits the grouped jobs to their respective resources. The dispatcher forwards the grouped jobs based on the schedule made during the matching of jobs with resources. The dispatcher also collects the results of the processed jobs from the resources through input ports.

3. RELATED WORK

In this section, the various job grouping algorithms proposed in literature for job scheduling in grid environment are discussed. As stated by Buyya, Date, et. al. [5], the need for a job grouping method became an imperative research area after the emergence of distributed analysis of brain activity data. The Magneto encephalography (MEG) helmet is used for recording information about brain activities. A 64-sensored MEG instrument produces 0.9 GB of data in an hour and the data is used to generate 7257600 analysis jobs which take about 102 days on a commodity computer. Global grids enable the partnering doctors to share the MEG instrument and allow the analysis jobs to be computed among the distributed computing resources. Large amount of computation power reduces the total time taken for completing the analysis jobs. The main issue is the expense caused from the overhead communication time. This necessitates grouping of jobs.

A dynamic job grouping-based scheduling algorithm [7], groups the jobs according to MIPS (Million Instructions per Second) of the available resources. The proposed job scheduling strategy takes into account: *(i)* the processing requirements for each job, *(ii)* the grouping mechanism of these jobs, known as job grouping, according to the processing capabilities of available resources, and *(iii)* the transmitting of the job grouping to the appropriate resource. This model reduces the processing time and communication time of jobs, but the algorithm doesn't take the dynamic resource characteristics into account and the grouping strategy can't utilize resource sufficiently.

Scheduling framework for Bandwidth-Aware Job Grouping Based strategy [8] groups the jobs according to MIPS and bandwidth of the resource. The principle behind the bandwidth-aware scheduling is the scheduling priorities taking into consideration not only their computational capabilities but also the communication capabilities of the resources. The bandwidth-aware scheduling approach uses the network bottleneck bandwidth of resources to determine the priority of each resource. But the deficiencies of the algorithm are first, grouping strategies does not utilize resource sufficiently, and second, consideration of bandwidth strategy is not efficient to transfer the job.

A Bandwidth-Aware Job Grouping-Based scheduling strategy [9], 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.

Grouping-based fine-grained job scheduling algorithm [10] presents job scheduling algorithm that schedule the group jobs according to resource MIPS and Bandwidth. Greedy algorithm is used to cluster lightweight jobs. The job will join the first job group that still meet the constraint conditions after the job joins in. And if the job is a coarse-grained job, it will be allocated to an appropriate resource without grouping. But the problem of the algorithm is preprocessing scheduling time of the job is high, time complexity of the scheduling algorithm is high and finally, it does not give any attention to the memory requirement of file-size.

In Constraint-Based Job and Resource scheduling (CBJRS) algorithm [11] grouping is done based on processing capability (in MIPS), bandwidth (in Mb/s), and memory-size (in Mb) of the available resources. The resources are arranged in hierarchical manner where Heap Sort Tree (HST) is used to obtain the highest computational power resource or root node, so as to make balanced and effective job scheduling.

Memory aware job scheduling Model [12] presents and evaluates an extension to Computational-Communication Memory size based job grouping scheduling strategy that tries to maximize the utilization of Grid resources and their processing capabilities, and also reduces processing time and network delay to schedule and execute the jobs on the Grid. The proposed job scheduling is based on job grouping concept taking into account memory constraint together with other constraints such as processing power, bandwidth, expected execution and transfer time requirements of each job.

An Improved Resource Scheduling Approach Using Job Grouping strategy in Grid Computing [13] proposes grid level resource scheduling with Job Grouping strategy that maximizes the resource utilization and minimizes processing time of jobs.The resource and job scheduling model is based on a hierarchical approach. The model is divided into three levels, user level, top level (global level) and local level (cluster level). Whenever an application is submitted into grid at the global level, highest computational power cluster is selected and if its total computational power is higher than submitted application then next application enters and this time total required computational power of application (first + second) is compared with selected cluster computational power. The same process is repeated until total computational power requirements of the applications in that group is less than or equal to the available computational power of the selected cluster. This grouped of applications is submitted to the cluster having highest computational power and then local level scheduling is done according to the ability of nodes within the cluster.

In improved heuristic approach based on Particle Swarm Optimization (PSO) algorithm to solve task scheduling problem in grid is proposed. In improved PSO algorithm user jobs were grouped in an un-uniform manner. The percentage of the processing capability of a resource on the total processing capability of all the resources is calculated. Then using this percentage, the processing capability of a resource based on the total length of all tasks to be scheduled is calculated. By this way the jobs are allocated to the available resources not uniformly, but the utilization of resources will be increased. The scheduler groups the jobs according to the calculated processing capability. The new job group is scheduled to execute in the available resources. This process of grouping and scheduling is repeated until all the user jobs are grouped and assigned to selected grid resources. [14].

4. STUDY OF SCHEDULING ALGORITHMS

4.1 Constraint-Based Job and Resource scheduling in Grid Computing

Job scheduling is the mapping of jobs to specific resources but assigning a single job to the specific resource takes high processing time and communication time. So, processing and communication time can be reduced by considering a grouping strategy [11].This Grouping strategy is based on processing capability (in MIPS), bandwidth (in Mb/s), and memory-size (in Mb) of the available resources. Jobs are put into the job group until all the following conditions are satisfied:

$$\text{Groupedjob\_MI} <= \text{Resource\_MIPS} * \text{Granularity size} \quad (1)$$

$$\text{Groupedjob MS} <= \text{Resource MS} \quad (2)$$

$$\text{Groupedjob\_MS} <= \text{Resource\_baudRate} * \text{Tcomm} \quad (3)$$

Where, MI (Million Instruction) is job's required computational power, MIPS (Million Instruction Per Second) is processing capability of the resource and Granularity size is user defined time which is used to measure total no. of jobs that can be completed within that specified time, Groupedjob_MS is required Memory Size of group jobs and Resource_MS is available Memory of the resource. Size of resource, Baud Rate is the bandwidth capacity of resource, Tcomm is the job's communication time. Equation (1) required computational power of grouped jobs shouldn't exceed to the resource's processing capability. Eq (2) Memory-size requirement of grouped job shouldn't exceed to the resource's memory-size capability. In Eq (3) Memory-size of the grouped job shouldn't exceed to resource's transfer capability within a given time period. These are the main factors in job grouping strategy that influences the way job grouping is performed to achieve the minimum job processing time and maximum resource utilization of the Grid resources.

4.1.1 Pseudocode of the Algorithm [11]:

```
1. $Groupedjob_i$:=0;
2. Sort(JobList_size) in Ascending order according to MI and
   assign ID.
3. Resources selected by HST;
4. For i:=0 to ResourceList size-l Do
5. $(Groupedjob_i)_{MI}$:= 0;
6. $R_{i\ MI}$ := $ResourceList_{i\ MIPS}$ * Granularity size;
7. $R_{i\ BW}$:= baud Rate * $T_{comm}$
8. For j:=0 to JobList_size-1
9. while ( j <=Joblist_ size-1)
10. {
11. Groupedjob:= Groupedjob+ $Job_j$ ;
12. if((($(Groupedjob)_{MI}$ <= $R_{i\ MI}$) &&($(Groupedjob)_{MS}$ <= $R_{i\ MS}$)
    &&((Groupedjob)$_{MS}$ <= $R_{i\,BW}$)))
13.{
14.   j++;
15. }
16. Else
17.{
18. Groupedjob:= Groupedjob – $Job_j$ ;
19. }
20. j--;
21. break;
22. } //End while
23. Create a new job with total MI less or equals to Resource
    MI;
24. Assign a unique ID for the newly created Groupedjob;
25. Place the Groupedjobj to Target ResourceListj for
    Computation;
26. Receive computed Groupedjob from ResourceListj;
27. i++;
28. Endfor;
29. End;
```

In this algorithm, after gathering the details of user jobs and the available resources, the system selects jobs in order after sorting them in descending order of their MI to form different job groups. Resources are arranged in hierarchical manner, where Heap Sort Tree (HST) is used to obtain the highest computational power resource or root node, so as to make balanced and effective job scheduling. When the resources join into the grid, they are arranged in a tree by Heap sort Tree using their computational power. The root node of the tree having highest computational power in whole grid system is ready to compute the jobs .Jobs are put into a job group one after another until sum of the resource requirements of the jobs in that group is less than or equal to amount of resource available at the selected resource site. In this way jobs are subsequently gathered or grouped one by one according to the resulting MI, Memory size and Bandwidth of the resource until the condition on which it is based is satisfied. As soon as a job group is formed, the scheduler submits the grouped job to the corresponding resource for job computation.

4.2 A Memory-Aware Dynamic Job Scheduling Model in Grid Computing

Memory aware job scheduling Model presents and evaluates an extension to Computational-Communication Memory size based job grouping scheduling strategy that tries to maximize the utilization of Grid resources and their processing capabilities, and also reduces processing time and network delay to schedule and execute the jobs on the grid. The model groups the jobs according to jobs requirement and available resource capability. The size of a grouped job depends on the processing requirement length expressed in Million Instructions, Bandwidth expressed in Mb/s and Memory size requirement expressed in Mb, expected execution and transfer time in seconds [12].

The processing requirement of the Grouped job or coarse-grained job shouldn't exceed to the resource processing capability at any point of time during grouping of the jobs. Memory size requirement of the grouped job shouldn't exceed to the resource memory size capability. Memory size of the grouped jobs shouldn't exceed to resource transfer capability at any point of time during grouping of the jobs. Communication time of the grouped jobs should not exceed computation time of the grouped jobs. These are the main constraints in job grouping strategy that influences the way job grouping is performed to achieve minimum job execution time and maximum resource utilization in the Grid system.

4.2.1 Pseudocode of the Algorithm [12]:

```
1. $Groupedjob_k$:=0, j:=0, k=0;
2. Sort resources in descending order according to their
   MIPS;
3. Jobs are taken in FCFS order and assigned each an ID;
4. For i:=0 to ResourceList size-1 Do
5. {
6. $T_{EG}$ :=0,C= 0, M:=0, $T_{EG}$ :=0
7. while ( j<=Joblist_size-1)
8. {
9. $T_{EG}$ :=+ $t_{ej}$;
10. C:=+ $\alpha_{cj}$;
11. M:=+ $\mu_{mj}$;
12. $T_{TG}$ :=+ $t_{ij}$;
13. $R_{i\,MI}$ := $p_{pi}$ * $T_{EG}$;
14. $R_{i\,BW}$:= $\beta_{bi}$ * $T_{TG}$;
15. if(((C<= $R_{i\,MI}$) && (M <= $\Pi_{mi}$)) && ((M <= $R_{i\,BW}$)))
16. {
17.    $Groupedjob_k$:= $Groupedjob_k$+ $Job_j$ ;
18.    j ++;
19. }
20. Else
21. {
22.    $T_{EG}$ := $T_{EG}$ - $t_{ej}$ ;
23.   C=C - $\alpha_{cj}$ ;
24.   M:=M -$\mu_{mj}$;
25.   $T_{TG}$ := $T_{TG}$- $t_{ij}$;
26.   Submit the Groupedjob$_k$ to $R_i$
27.   Set computational power of the resource $R_i$ to zero
28.   k++;
29.  }//End if
30.  Break;
31.  } // End while
32. If (Prearranged Time of Reconstruction is arrived)then
33. {
34. Reconstruct the Job queue and Resource queue;
35. Go to step 1;
36. }
37. }// End for
38. End;
```

In this algorithm, after gathering the details of user jobs and the available resources, the system selects jobs in FCFS order to form different job groups. The scheduler selects resources in FCFS order after sorting them in descending order of their MIPS. Jobs are put into a job group one after another until sum of the resource requirements of the jobs in that group is less than or equal to amount of resource available at the selected resource site. In this way jobs are subsequently gathered or grouped one by one according to the resulting MI, Memory size and Bandwidth of the resource until the condition on which it is based is satisfied. As soon as a job group is formed, the scheduler submits the grouped job to the corresponding resource for job computation. After executing the job group, the results goes to the corresponding users and resource is again available to Grid system with its available power and ready to execute another job.

4.3 Improved Job-Grouping Based PSO Algorithm For Task Scheduling In Grid Computing

PSO is one of the latest population-based search models and has been applied successfully to a number of optimization problems. A PSO algorithm contains a swarm of particles in which each particle includes a potential solution [14]. The user jobs were grouped in an un-uniform manner based on the percentage of a particular resource processing capacity on the total processing capacity of all the resources available in the grid, which improves computation/communication ratio and utilization of resources.

4.3.1 Particle Swarm Optimization for scheduling

Particle Swarm Optimization (PSO) is a swarm-based intelligence algorithm influenced by the social behaviour of animals such as a flock of birds, finding a food source or a school of fish protecting themselves from a predator. A particle in PSO is analogous to a bird or fish flying through a search (problem) space. The movement of each particle is co-ordinated by a velocity which has both magnitude and direction. Each particle position at any instance of time is influenced by its best position and the position of the best particle in a problem space. The performance of a particle is measured by a fitness value, which is problem specific. In PSO, the population is the number of particles in a problem space. Particles are initialized randomly. Each particle will have a fitness value, which will be evaluated by a fitness function to be optimized in each generation. Each particle knows its best position pbest and the best position so far among the entire group of particles gbest. The pbest of a particle is the best result (fitness value) so far reached by the particle, whereas gbest is the best particle in terms of fitness in an entire population. In each generation the velocity and position is updated using following equation:

$$v_i^{k+1} = \omega v_i^{k} k + c_1 * rand_1 * \left(pbest_i - x_i^{k}\right) + c_2 * rand_2 * \left(gbest_i - x_i^{k}\right)$$

And $x_i^{k+1} = x_i^{k+1} + v_i^{k+1}$

where V_i^k: velocity of particle i at iteration k

V_i^{k+1}: velocity of particle i at iteration k + 1

ω : inertia weight

c_j: acceleration coefficients; j = 1, 2

The PSO algorithm starts with random initialization of particle's position and velocity. In this problem, the particles are the task to be assigned and the dimension of the particles is the number of tasks in a workflow. The value assigned to each dimensions of a particles are the computing resources indices. Thus the particles represent a mapping of resource to a task. The evaluation of each particle is performed by the fitness function. The particles calculate their velocity using above given Equations. The evaluation is carried out until the specified number of iterations (user-specified stopping criteria). PSO algorithm provides a mapping of all the tasks to a set of given resources based on the processing capability of the available resources.

4.3.2 Pseudo code of the Algorithm [14]:

```
1.    The scheduler receives Number of Gridlets 'n' and Number of Resources 'm'.
2.    Scheduler receives the Resource-list R[].
3.    Set Tot-MIR (Sum of the processing capacity of all the resources) to zero.
4.    Set Tot-GMI (Sum of the length of all the gridlets) to zero.
5.    The Gridlets created by the system are submitted to the scheduler.
6.    Set the resource ID j to 1 and the index i to 1.
7.    While j is less than or equal to m repeat steps 7.1 to 7.4.
7.1.    Get the jth resource from the resource list.
7.2.    Multiply the MIPS of jth resource with granularity time specified by the user.
7.3.    Find Tot-MIR by adding previous Tot-MIR with the value got from step.
7.4.    Get the MIPS of the next resource.
```

```
8.    Assign the gridlets to the resources using PSO algorithm.
9.    While i is less than or equal to n repeat steps 9.1 to 9.3.
9.1.  Get the length of the ith Gridlet ($G_i$-MI).
9.2.  Find Tot-GMI by adding previous Tot-GMI and $G_i$-MI
9.3.  Get the length of the next Gridlet.
10.   While j is less than or equal to m repeat steps 10.1 to 10.3.
10.1  Calculate the processing capability of jth resource by multiplying MIPS of jth resource and granularity time.
10.2  Calculate the processing capability of jth resource on the processing capability of all the available resources ($PTot-GMI_j$) by dividing processing capability of jth resource by Tot-MIR.
10.3  Calculate the processing capability of jth resource on the total length of all available gridlets by multiplying $PTot-GMI_j$ and Tot-GMI.
11.   Set k to zero
12.   While i is less than or equal to n repeat 13 to 15
13.   While j is less than or equal to m repeat steps 13.1 to step 15
13.1  Set Tot-Jleng to zero.
13.2  While Tot-Jleng is less than equal to $PTot-GMI_j$ and i is less than n repeat:
      Begin
          Calculate Tot-Jleng by adding previous Tot-Jleng and length of the ith Gridlet ($G_i$-MI)
      End
14.   If Tot-Jleng is greater than $PTot-GMI_j$ then subtract $G_i$-MI (length of the last Gridlet) from Tot-Jleng.
15.   If Tot-Jleng is not zero repeat steps 15.1 to 15.4.
15.1. Create a new Grouped-gridlet of length equal to Tot-Jleng.
15.2. Assign a unique ID to the newly created Grouped-gridlet
15.3. Insert the Grouped-gridlet into a new Grouped-gridlet list $GJ_k$
15.4. Insert the allocated resource ID into the Target resource list $TargetR_k$
15.5. Increment the value of k
16.   When all the gridlets are grouped and assigned to a resource, send all the Groupedgridlets to their corresponding resources.
17.   After the execution of the grouped-gridlets by the assigned resources send them back to the Target resource list.
18.   Display the Id of the resource, start time, end time, simulation time and task execution cost of each executed grouped-gridlet.
```

The percentage of the processing capability of a resource on the total processing capability of all the resources is calculated. Using this percentage, the processing capability of a resource based on the total length of all tasks to be scheduled is calculated. By this way the jobs are allocated to the available resources. The scheduler groups the jobs according to the calculated processing capability. The new job group is scheduled to execute in the available resources. This process of grouping and scheduling is repeated until all the user jobs are grouped and assigned to selected grid resources.

5. EXPERIMENTAL RESULTS

GridSim toolkit [15] is used to conduct the simulations by setting values to the number of jobs from 100 to 500. Processing time is recorded to analyze the feasibility of the algorithms. The system accepts total number of user jobs, processing requirements or average MI of those jobs, allowed deviation percentage of the MI, granularity size of the job grouping activity and the available Grid resources in the Grid environment. Details of the available Grid resources are obtained from Grid Information Service (GIS) entity that keeps track of the resources available in the Grid environment. Each Grid resource is described in terms of their various characteristics, such as resource ID, name, total number machines in each resource, total processing elements (PE) in each machine, MIPS of each PE, and bandwidth speed. In this simulation, the details of the Grid resources used are as follows:

Table 1. Grid Resources Setup For the Simulation

Resource Name	MIPS	Cost per sec
R1	200	100
R2	160	200
R3	210	300
R4	480	210

The tests are conducted using four resources of different MIPS as shown in table 1. The MIPS of each resource is computed as follows:

$$Resource\ MIPS = Total\_PE * PE\_MIPS,$$

where

Total_PE = Total number of PEs at the resource,

PE_MIPS = MIPS of PE

Each resource has its own predefined cost rate for counting the charges imposed on a Grid user for executing the user jobs at that resource. The MIPS and cost per second are selected randomly for the simulation purpose.

The total processing cost is computed based on the actual CPU time taken for computing the Gridlets at the Grid resource and at the cost rate specified at the Grid resource, as summarized below:

$$Process\_Cost = T * C,$$

where

T = Total CPU Time for Gridlet execution, and

C = Cost per second of the resources.

Table 2 and figure 2 shows the results obtained for processing time required to execute 100 to 500 gridlets using different job grouping based algorithms keeping the same resources and job specification

Table 2. Comparision Between The Algorithms

Accoridng To Their Processing Time

No. of Jobs	**PROCESSING TIME(In Simulation Sec)**		
	Constraint Based Job and Resource Scheduling (CBJRS)	**A Memory-Aware Dynamic job Scheduling Model(MDJS)**	**Improved Job-Grouping Based PSO algorithm for task Scheduling (IJGPSO)**
100	227	234	177
200	329	361	256
300	357	464	374
400	470	623	410
500	536	726	427

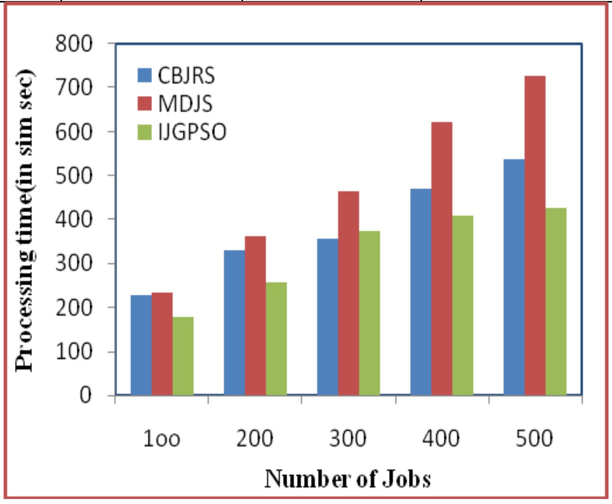

Figure 2. Processing time of different algorithms for executing 100 to 500 gridlets

Table 3 and figure 3 shows the results obtained for processing cost required to execute 100 to 500 gridlets using different job grouping based algorithms with the same resource and job specification. . From the results obtained, it is seen that heuristic based “Improved Job Grouping based PSO algorithm in grid Computing” algorithm takes less time and cost than the other two algorithms.

Table 3. Comparision Between The Algorithms

Accoridng To Their Processing Cost

No. of Jobs	**PROCESSING COST**		
	Constraint Based Job and Resource Scheduling (CBJRS)	**A Memory-Aware Dynamic job Scheduling Model (MDJS)**	**Improved Job-Grouping Based PSO algorithm for task Scheduling (IJGPSO)**
100	68723	61751	58383
200	125092	129727	78257
300	180607	185281	118562
400	243468	243848	197262
500	298575	312321	215113

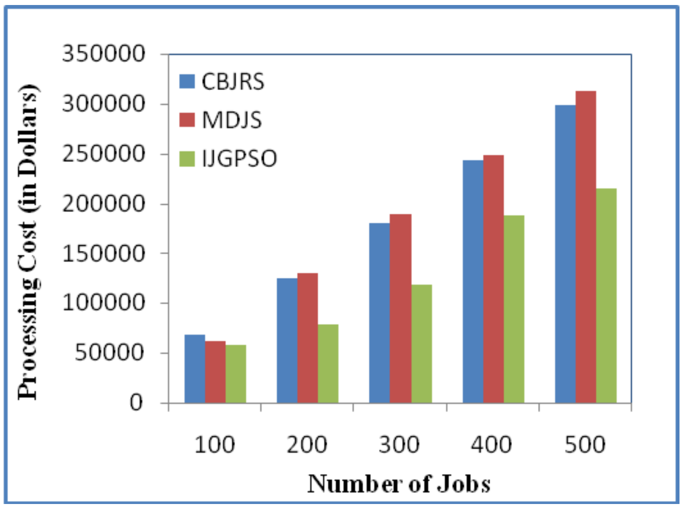

Figure 3. Processing cost for executing different number of gridlets

6. CONCLUSION AND FUTURE WORK

In this paper, we analyzed various job grouping based scheduling algorithms in grid computing. Simulation result has shown their processing time and cost with respect to number of jobs without considering the preprocessing time of the scheduling algorithm and results of the simulation may vary in different simulating environment. From the results obtained, it is clearly shown that heuristic based “Improved Job Grouping based PSO algorithm in grid Computing” algorithm takes less time and cost than the other two algorithms. As the both other two algorithms follow constraints (memory size constraint, bandwidth constraint), in that case “Constraint-Based Job and Resource scheduling in Grid Computing” gives better results in the terms of both processing time and cost.

In future, research on job scheduling can be carried out in various directions depending upon minimizing complexity of the scheduling algorithm, load balancing at local site, various load factors, tolerant, user’s demand and price etc. Future work may involve developing a more comprehensive job grouping-based scheduling system that takes into account QoS (Quality of Service) requirements of each user job before performing the grouping method and handle more complicated scenario involving dynamic factors such as dynamically changing grid environment for e.g. network failure, hardware failure at a node etc. The above constraints and issues can be taken into account in designing a more efficient and practical scheduler, that will help the society to realize the benefit and implementation of the real grid computing system.

7. REFERENCES

[1] Foster and C. Kesselman, “*The Grid: Blueprint for a Future Computing Infrastructure*” Morgan Kaufmann Publishers,San Francisco, CA, USA, 1999.

[2] I. Foster, C. Kesselman, S. Tuecke, “*The Anatomy of Grid: Enabling Scalable Virtual Organizations”*, International Journal of Supercomputer Applications, 2001.

[3] Mark Baker, Rajkumar Buyya, and Domenico Laforenza, "*Grids and Grid technologies for Wide-Area Distributed Computing*", Software - Practice and Experience – SPE , vol. 32, no. 15, pp. 1437-1466, 2002

[4] Berman, F., Fox, G. and Hey, A.,” *Grid Computing– Making the Global Infrastructure a Reality”* London,Wiley,2003.

[5] Buyya, R., Date, S., Miizuno-Matsumoto, Y., Venogopal, S. and Abramson, D., “*Neuroscience Instrumentation and Distributed Analysis of Brain Activity Data: A case for eScience on Global Grids*”, Journal of Concurrency and Computation: Practice and Experience. Vol 17, No. 15, pp.1783-1798 ,2004.

[6] V. Rajendran,G. Sudha Sadasivam, “*An Efficient Approach to Task Scheduling in Computational Grids*”, International Journal of Computer Science and Application, vol. 6, No. 1, pp. 53-69, 2009.

[7]. Muthuvelu. N, Liu. J, Lin Soe. N, Venugopal. S, Sulistio. A and Buyya. R, 2005, “*A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids*”, in Proceedings of Australasian Workshop on Grid Computing and e-Research (AusGrid2005), , pp. 41-48,2005.

[8]. Ng. W. K, Ang. T. F, Ling. T. C, and Liew. C. S, “*Scheduling Framework for Bandwidth-Aware Job Grouping-based Scheduling in Grid Computing*”, Malaysian Journal of Computer Science, Vol. 19, pp.117-126,2006.

[9]. T.F. Ang, W.K. Ng, “*A Bandwidth-Aware Job Scheduling-Based Scheduling on Grid Computing*”, Asian Network for Scientific Information, vol. 8, No. 3, pp. 372-277, 2009.

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

[11] M.K.Mishra,V.K.Soni,R. Sharma,Sarita Das,**“***Constraint-Based Job and Resource scheduling in Grid Computing***”** ,3rd International Conference On Computer Science and Information Technology,IEEE,2010.

[12] M.K.Mishra,V.K.Soni,R. Sharma, B. R. Parida, R. K. Das**,“***A Memory-Aware Dynamic Job Scheduling Model in Grid Computing***” ,** International Conference On Computer Design And Appliations,IEEE,2010.

[13] M.K.Mishra,V.K.Soni,R. Sharma, “*An Improved Resource Scheduling Approach Using Job Grouping strategy in Grid Computing***”,** International Conference on Educational and Network Technology,IEEE,2010.

[14] S.Selvarani, G.Sudha Sadhasivam “*Improved Job-Grouping Based PSO Algorithm For Task Scheduling In Grid Computing”,* International Journal of Engineering Science and TechnologyVol. 2(9), 2010.

[15] R. Buyya and M. Murshed*, “GridSim; A toolkit for the modeling and simulation of distributed management and scheduling for grid computing*”, Concurrency and Computation: Practice and Experience , Volume 14, Issue 13-15,2002.