Grouping-based Scheduling with Load Balancing for Fine-Grained Jobs in Grid Computing

Rabab Mohamed Ezzat
Department of Computer Science
Faculty of Computers and Information
Helwan University
Cairo, Egypt

Amal Elsayed Aboutabl
Department of Computer Science
Faculty of Computers and Information
Helwan University
Cairo, Egypt

Mostafa Sami Mostafa
Department of Computer Science
Faculty of Computers and Information
Helwan University
Cairo, Egypt

***Abstract*—Grid computing is characterized by the existence of a collection of heterogeneous geographicallydistributed resources that are connected over high speed networks.Job scheduling and resource management have been a great challenge to researchers in the area of grid computing.Very often, there are applications having alarge number of fine-grainedjobs.Sending these fine-grained jobsindividually to be executed on grid resources that have high processing power reduces resource utilization and is thus uneconomical. This paper presents efficient grouping-based scheduling models that group fine-grained jobs to form coarse-grained jobs which are sent for execution on grid resources. Our groupingstrategy is based on the processing capability of resources and the processing requirements of grouped jobs.A load balancing approach is also presented to achieve efficient utilization of resources. Simulation experiments were conducted using the Gridsim toolkit. Results show that thetotal simulation time and the cost are improved by grouping.Furthermore, our load balancing approach enhances resource utilization and achieves load balancing among resources.**

Keywords—grid computing;job scheduling; job grouping; load balancing; resource utilization; Gridsim

I. INTRODUCTION

Grid Computing is a computing paradigm that emerged in the late 1990's [10]. The emergence of this paradigm was mainly due tothe spread of powerful computers that have high computing power at low cost in addition to the popularity of the internet and availability of high speed networks [1]. Grid computing allows sharing and using of geographically distributed resources including supercomputers, data sources and specialized devices that are owned by different organizations [2].There are large scale compute-intensive problems in different fields such as engineering, science and economics that need high computing power to be solved. Grid computing enables sharing resources that are connected through the internet for solving these problems.

Resource management and job scheduling have been a cause of great challenge to researchers in the field of grid computing[7]. Grid scheduling is a complex process which differs from scheduling in traditional distributed systems because of the characteristics of grid computing environment:

- Resources are geographicallydistributed over different multiple administrative domains.
- Resources are not under central control.
- Resources are heterogeneous; different in architecture and management policies.
- Jobsin the grid are from different usershaving different requirements.

Many applications consist of a large number of fine-grained jobs having small scale processing requirements. Sending these jobs individually to be executed on grid resources that have high processing power reduces resource utilization and is thus uneconomical. Moreover, the total communication time for transmitting each fine-grained job to the resource may exceed the total computation time of that job on the resource.In grid computing, and for such type of applications, having coarse-grained jobs is more efficient and cost-effective than fine-grained jobs [12]. Therefore, instead of sending such jobs individually, coarse-grained jobs can be created by collecting a suitable number of jobs [3]. Grouping fine-grained jobs together to form coarse-grained jobs reduces the transmission time and increases resources utilization [4]. The total processing time needed for each fine-grained job includes:scheduling time that is the time spent to schedule the job, Sending time that is thetime spent to send the job to a resource, execution time that is the time spent to execute the jobandreceiving time that is the time spent to receive the job from a resource after execution.

Scheduling is the process of assigning or mapping jobs to suitable resources that execute jobs achieving the following goals [4]:

- Minimizing the processing time.
- Minimizing processing cost.
- Achieving load balancing among resources.

A *grid computing scheduler* is responsible for selecting the most suitable machine or computing resource for processingeach job to achieve maximum system throughput. In case of having fine-grained jobs requesting service on the grid, these fine-grained jobs are grouped to form coarse-grained jobswhich are then handled by the scheduler in such a way that achieves loadbalancing [7].

Load balancing is a mapping strategy that distributes applications load among resources so that there will be

efficient utilization of resources and hence the performance of the system is improved [1] [6]. Load balancing algorithms of traditional parallel and distributed systems cannot be used in grid computing because of the special characteristics of grid environments. In grid computing environment, resources differ in their computational power. Efficient load balancing algorithms are neededto maximize resources utilization and prevent the condition where some resources may be overloaded and other resources may be idle [2].

The paper is organizes as follows. Section II discusses previous works in the area of grouping-based grid scheduling.Section III presents our proposed models. Section IV provides a detailed description of our simulation, experiments and results. Finally, Section V provides a conclusion of this work.

II. Previous Work

Grid scheduling is a complexprocess that has been a challenge for researchers due to the heterogeneity of the grid environment. There have been a number of attempts in the area of grid scheduling in the literature since the grid computing paradigm emerged. In particular, we focus here onmodels that were developed to group and schedule fine-grained jobsin grid environments.

Constraint-based job and resource scheduling in grid computing [15] is a model in which resources are arranged in a hierarchical manner so that the resource with the highest computation power can be found using tree heap sort while jobs are grouped according to the processing capability, bandwidth and memory size of resources. A job scheduling model based on grouping was developed in [10]. In this model, resources are sorted in ascending order of their processing capability and then jobs are grouped according to the processing capability, bandwidth and memory size of the selected resources.

Two other grouping-based models were developed in [12] and [13] where resources are sorted according to bottleneck bandwidth and group jobs according to processing capability and bandwidth of the resources. In both the two models using the bandwidth strategy is not efficient to transfer the jobs. In [13] grouping strategy does not utilize the resources sufficiently. Fine-grained jobs are scheduled in [11] according to processing capability and bandwidth of selected resources.A dynamic job scheduling approach which is based on grouping is proposed in [14] for deploying applications with fine-grained tasks on global grids and is based on the processing capability of resources. This model reduces the processing time and communication time but does not utilize the resources sufficiently.A time minimization dynamic job grouping based scheduling is proposed in [5]. Resources are sorted in descending order based ontheir processing capability then fine-grained jobs are grouped according to processing capability of the selected resource by taking one job from the front of the sorted job list and one job from the end.An agent-based dynamic resource scheduling model with FCFS job grouping strategy is presented in [9]. Another algorithm which is based on grouping and takes into consideration both the memory requirements and execution time of jobs is presented in [7].

All of the previous researches depend on grouping fine-grained jobs to obtain coarse-grained jobs which are then sent to selected resources to be executed. The focus in previous researcheshas mainly been reducing processing and communication time. However, increasing the efficiency of resource utilization bybalancing load among resources has not been sufficiently addressed.Two load balancing approaches for computational grids were presented in [1] and [6] where a job is sent to the resource that has minimum queue length. Load balancing is achieved in the grid but at the expense of high overall execution time caused by increased communication time incurred by sending fine-grained jobs individually.

III. Proposed Models

Our proposed model consists of two parts; a grouping strategy and a scheduling model. Our grouping strategy groups fine-grained jobs to form a smaller number of coarse-grained jobs depending on the processing capability of the selected resource and the jobs' processing requirements. The UFF (User-Finished-First) scheduling model and URS (Users-Resources-Sharing) scheduling model are two different proposed models that group fine grained jobs and schedule these jobs in two different ways. RMQ (Resource with Minimum Queue Length) scheduling model is another proposed model that group fine grained jobs and schedule these jobs to the resource with minimum number of waiting jobs. This model is a load balancing approach based on the queue length of the available resource.

A. Grouping Strategy

The job scheduler groupsfine-grained jobsbased on both the processing requirements of each job and the processing capability of each resource.First, the scheduler selectsa resource from the ResourcesList and computes the product of MIPS (million instructions per second) which is used to define a resource's processing capabilityand G.T (granularity time)that is a user defined parameterwhich is used to measure the total number of jobs that can be completed within a specified period of time. In UFF scheduling model and URS scheduling model the resources in the ResourcesList are sorted in ascending order based on processing capability of each resource. In RMQ scheduling model select the resource that have minimum queue length (minimum number of waiting jobs).Second, the scheduler selectsfine-grained jobs from GridletList one after the other. Collect the MI of the selected fine-grained jobs. Each job's MI (million instructions) defines the computational power needed to execute the job. The grouping step ensures that the total required computational power of grouped jobsdoes not exceed the processing capability of the resource.The term gridlet is used here to refer to a job that can run independently and sequentially on a grid resource.

The detailed steps of the grouping strategy are listed below.

1) Availablegridletsare sent to the job scheduler for scheduling.

2) Grid resources register their information atthe Grid Information Service (GIS).

3) The job scheduler requestsresources information from GIS. GIS sends the information of available resources to the scheduler.

4) Sort resources in ResourcesList in ascending order based on resource processing capability (MIPS).

5) Sort gridlets of each user in a separate GridletsListin ascending order based on gridlet length (MI).

6) Get the first user.

7) Initialize indices of GridletsList, ResourcesListand Grouped_Gridlets, named I, X and J respectively, all initialized to 0.

8) Selectthe resource specified by ResourcesList [X].

9) Select job specified by GridletList [I] of current user.

10) Groupedjob_length=0.

11) MI of ResourcesList[X]= MIPS of ResourcesList[X] Granularity time.*

12) For(I=0 to GridletsList size-1)

13) {

14) If(Groupedjob_length< MI of ResourcesList[X]))

15) Groupedjob_length= Groupedjob_length +length of GridletsList [I]

16) Else

17) {

18) Groupedjob_length= Groupedjob_length - length of GridletsList[I]

19) Create a new job,Grouped_Gridlet [J],whose length is equal to Groupedjob_length.

20) Increment J.

21) Break;

22) }

23) }

Starting from line 24, two different sequences of steps are presented reflecting two different scheduling models.

B. UFF *Scheduling*

After forming the grouped jobs, the question is whether to map the grouped jobs of a certain user to the available resources or let a number of users share the resources. In the *UFF (User-Finished-First)*scheduling model, the grouped jobs of a user are assigned to the available resources in parallel before proceeding with the next user. Accordingly, the first user sends grouped job 0 to resource 0 then grouped job 1 to resource 1 and so on till the last resource is reached. The next user is selected and the same steps are repeated. In lines 24-29, Total_Resources refers to the number of resources.

UFF (User-Finished-First) Scheduling Model

*24)*Submit Grouped_Gridlet [J] to ResourcesList[X].
*25)*X++.
*26)*If (X== Total-Resources)
27){ X==0
*28)*Get Next_User}
*29)*Go to step 8.

C. URS *Scheduling*

In this model, grouped jobs of different users are assigned to available resources in parallel. Therefore, grouped job0 of n different users are sent to the first n resources in the list. If the number of users is less than the number of resources, repeat starting from the first user with grouped job 1 and so on. When the last resource is reached, start from the first resource.

URS (Users-Resources_Sharing) Scheduling Model

24) Submit Grouped_Gridlet [J] to ResourcesList [x].

25) Get Next_User.

26) X++.

27) If (X== Total-Resources)

28) X==0.

29) Go to step 8.

D. RMQ (Resource with Minimum Queue Length) Scheduling Model

In this model, the scheduler selects the resource that has minimum queue length (minimum number of waiting jobs). Then, the scheduler uses the grouping strategy that based on the processing capability of the resource in addition to the processing requirements of the jobsto group fine-grained jobs and form-grouped jobs. Then send these grouped jobs to the resources to be executed. This model reduces the processing time and cost and achieves load balancing among resources. The following steps show the grouping strategy together with the load balancing approach.

1) Available gridlets are sent to the job scheduler for scheduling.

2) Grid resources register their information at the Grid Information Service (GIS).

3) The job scheduler requests resources information from GIS. GIS sends the information of available resources to the scheduler.

4) Sort gridlets of each user in a separate GridletsList in ascending order based on gridlet length (MI).

5) Get first user.

6) Initialize indices of GridletsList, ResourcesListand Grouped_Gridlets, named I, X and J respectively, all initialized to 0.

7) Select ResourceList [X] such that it is the resource with minimum QLength.

8) Select job specified byGridletList [I] of current user.

9) Groupedjob_length=0

10) MI of ResourceList[X]= MIPS of ResourceList[X] Granularity time*

11) For(I=0 to GridletList size-1)

12) {

13) If(Groupedjob_length< MI of ResourceList[X])

14) Groupedjob_length= Groupedjob_length +length of GridletList [I]

15) Else

16) {

17) Groupedjob_length= Groupedjob_length - length of GridletList[I]

18) Create new job Grouped_Gridlet [J] with length equal to Groupedjob_length

19) Increment J.

20) Break

21) }
22) }
23) Submit Grouped_Gridlet [J] to ResourceList [X] (resource with minimum QLength)
24) QLength[X]= QLength[X]+1
25) Get Next_User
26) Go to step 7

IV. Experimental Work And Results

Grid computing environment is a dynamic environment so it is extremely difficult to perform repeated experiments and studies on this environment in practice.Using simulation in such studies helps in performing a large number of experiments with various parameters. In this work, Gridsim was used for simulating the grid resources, jobs as well as our grouping strategy together with the proposed scheduling and load balancing models.

Most of the previous researches in the area of grid scheduling and resource allocation are based on a single user. Inthis work, multiple users are assumed. Each user has a number of independent jobs (Gridlets) that will be scheduled and then executed on heterogeneous resources taking into consideration resources load balancing.

A. Gridsim Simulation Environment

Gridsim is a java based discrete event grid simulator toolkit that allows modeling and simulation of grid computing system entities: resources, gridlets, scheduler, grid information service and users.Gridsimis also used to test scheduling and load balancing models [2]. Gridsim users are able to model and simulate the characteristics of grid resources and networks with different configurations. It, therefore, allows researchers to study grids and test new algorithms and strategies in a controlled environment.In Gridsim terminology,Gridlets are jobs that could run independently and sequentially on grid resources.

A grid environment is built using the Gridsim5-2 toolkit. After installing the Gridsim5-2 toolkit, the Gridsim package is imported. The grid environment is simulated using Jcreator by writing java code and implementingour grid entities:

- Create grid user(s):Multiple users are allowed. Each user in Gridsim must have a unique id.
- Create grid resources: Resources in Gridsimare defined by *resource name*, *communication speed*, *resource characteristics* (operating system, architecture, and cost), *and number of machines*.Each machine may consist of a *number of processing elements* each processing element is defined by a*unique id* and *processing capability*in*MIPS* (millions of instructions per second).
- Create gridlets:A gridlet is defined by *gridletlength*;*input file size* and*output file size*.

B. Simulation Input and Output

A number of simulation parameters are fed into the simulator:

- Gridlets: the number of gridlets.
- A_MI: average gridlet length in MI reflecting the processing requirementsof the job.Based on a gridlet's MI, the resource that hasa suitable processing capabilityis selected to execute this gridlet.
- Deviate%:MI deviation percentagewhich is used to create different number of gridlets that have different lengths.
- G_Time: Granularity time (expected job processing time). It is a measure of thenumber of jobs that can be completed within a certain time on a particular resource [14].
- OH_Time: Gridlet overhead time. In real environments, overhead time for each job depends on the current network load and speed. In our simulation, the overhead time of each gridletis an input value.
- Resources: Resources to be used in a simulation experiment are selected from the resources list.

After simulation input parameters have been defined, we conduct oursimulation experiments with andwithout grouping. This allows us to compare between scheduling fine-grained jobs with grouping and scheduling fine-grained jobs without grouping. This also allows us to study the effect of grouping the input gridlets on the overall performance. Two performancemetrics are used in this respect: *total processing time* and *total processing cost*. Total processing time is computed based on:

- Gridlet overhead processing time.
- Time taken to perform grouping.
- Time taken for sending gridlets to the resources.
- Time of processing the gridlets at the resources.
- Time taken for receiving the processed gridlets.

The total processing cost is computed based on:

- The time taken for computing the gridlets at the grid resource
- The cost specified at the grid resource.

C. Results

Tables 1, 2 and 3 show three different sets of simulation input parameters denoted by SI1, SI2 and SI3. Three users are assumed where each user is defined by a number of gridlets, MI and Deviate % of these gridlets as explained before. SI1, SI2 and SI3 assume 4, 7 and 5 resources respectively. Resources MIPS are varied to simulate resources' heterogeneity. Granularity and overhead time are also input.

TABLE I. Simulation Inputs SI1

User	Gridlets	MI	Deviate %
User1	100	10	10
User2	50	20	20
User3	150	30	30
Resources: R7,R5,R3,R1			

Resources_MIPS: 66,60,39,20
G_Time =5, **OH_Time**=5

TABLE II. SIMULATION INPUTS SI2

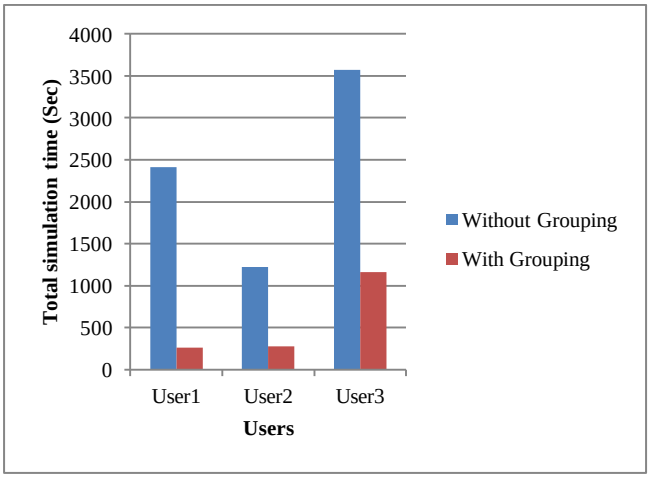
User	**Gridlets**	**A_MI**	**Deviate %**
User1	100	10	10
User2	50	20	20
User3	150	30	30
Resources: R4,R6,R7,R5,R3,R2,R1			
Resources_MIPS: 120,72,66,60,39,24,20			
G_Time =5, **OH_Time**=5			

TABLE III. SIMULATION INPUTS SI3

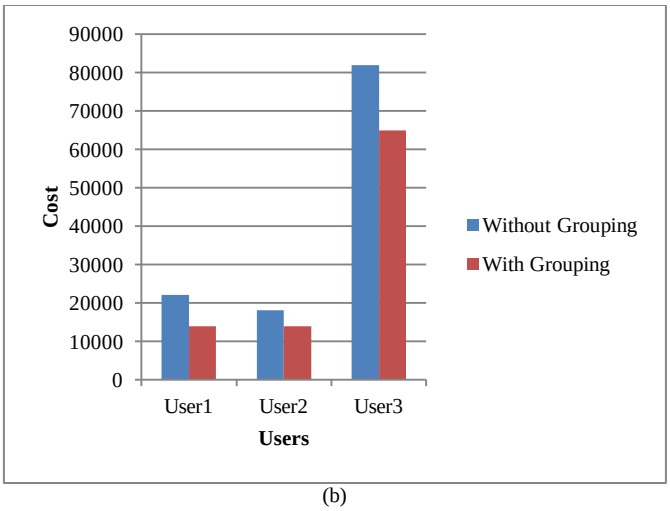
User	**Gridlets**	**MI**	**Deviate %**
User1	100	10	10
User2	150	20	20
User3	200	30	30
Resources:R4, R5,R3,R2,R1			
Resources_MIPS:120,60,39,24,20			
G_Time =5, **OH_Time**=5			

Figures 1, 2 and 3 show the results of total simulation time and cost with and without grouping in case of using simulation input parameters in table 1 for the three proposed modelsrespectively. Simulation time and cost are improved by using the grouping strategy.

This is due to the fact that total communication time is higher in case of scheduling the fine-grained jobs individually without grouping. On the other hand,when the grouping strategy is used, fine-grained jobs are grouped into a fewer number of coarse-grained jobs thus reducing the overall communication time.Similar results are obtained using simulation input parameters SI2 in table 2 as in Figures 4, 5 and 6.

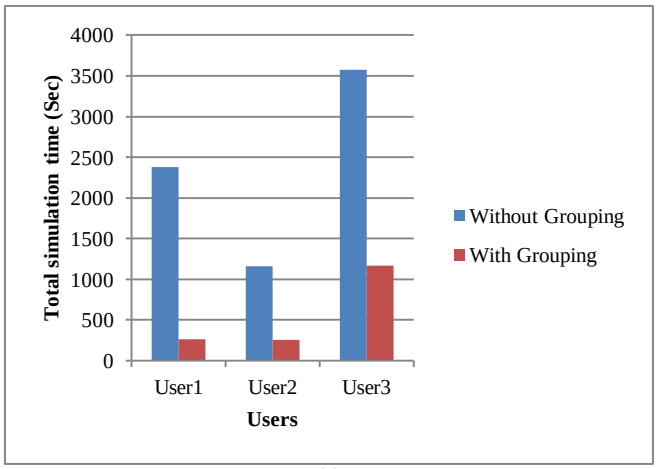


(a)

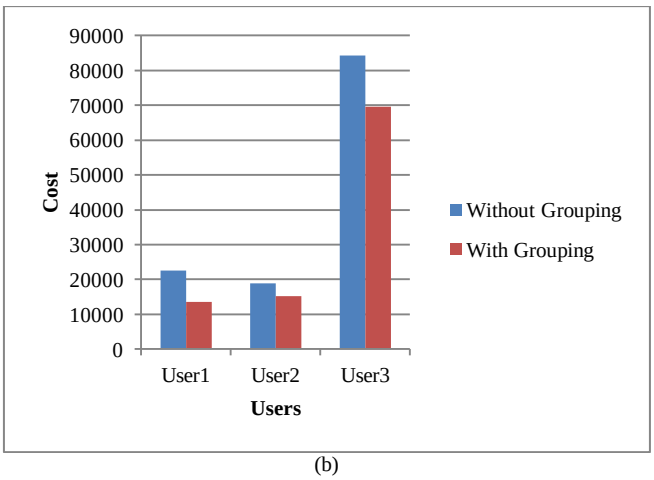


(b)

Fig. 1. Simulation time and processing cost using simulation inputs parameters SI1 in table 1 using UFF.

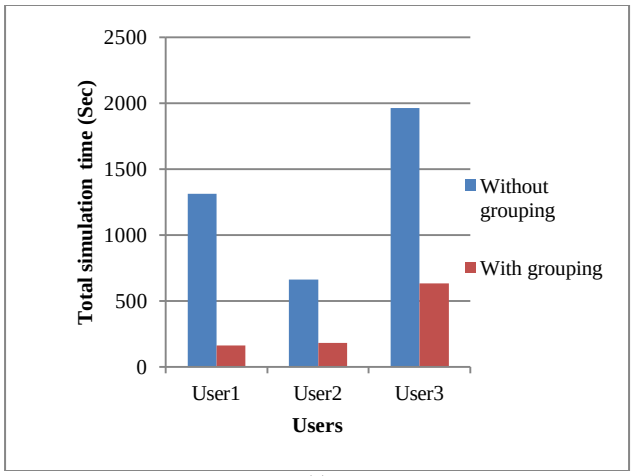


(a)

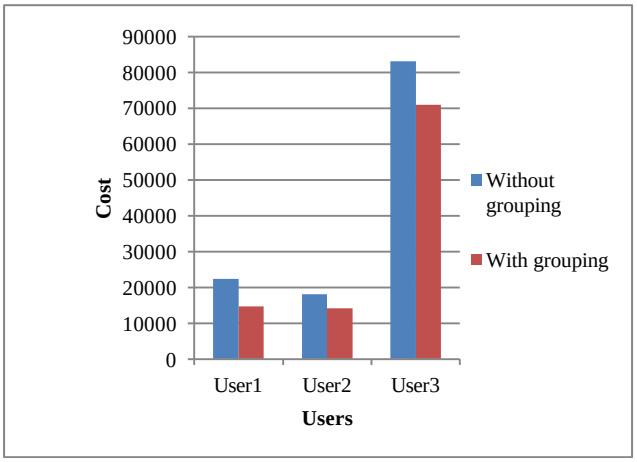


(b)

Fig. 2. Simulation time and processing cost using simulation input parameters SI1 in table 1 using URS.

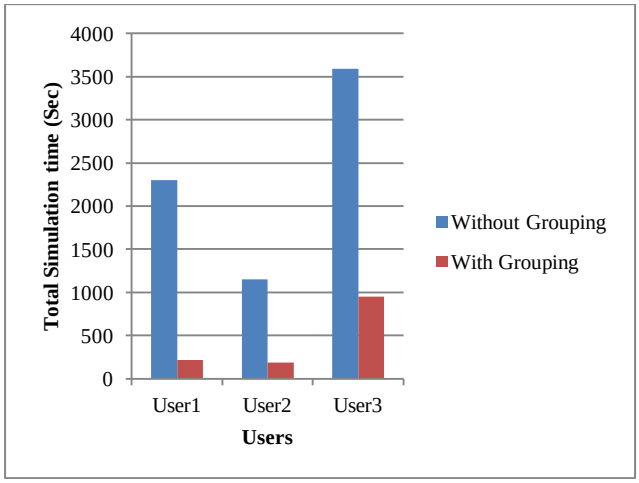


(a)

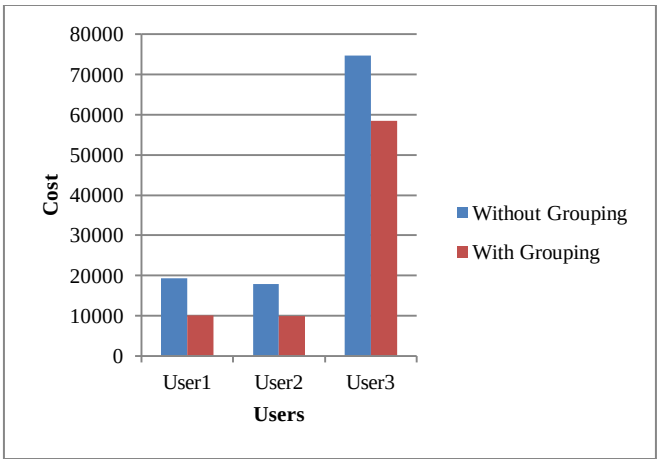


(b)

Fig. 3. Simulation time and processing cost using simulation inputs parameters SI1 in table 1 using RMQ.

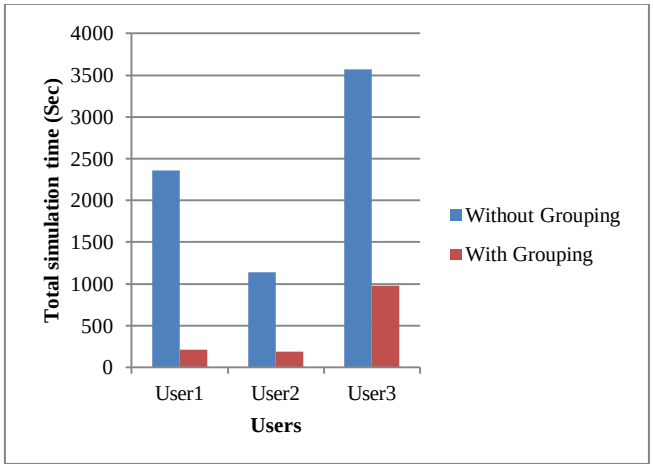


(a)

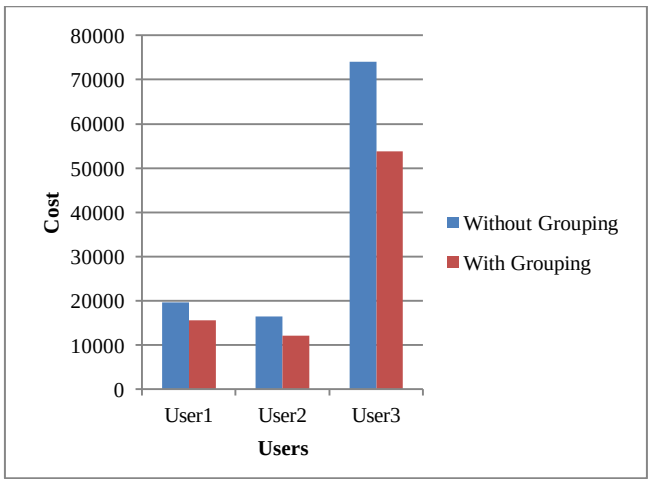


(b)

Fig. 4. Simulation time and processing cost using simulation input parameters SI2 in table 2 using UFF.

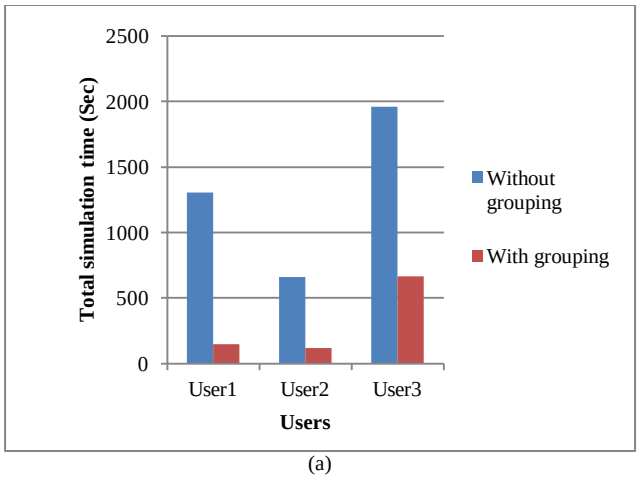


(a)

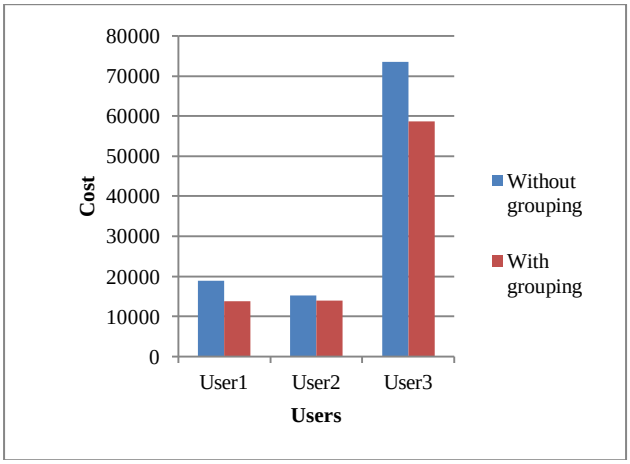


(b)

Fig. 5. Simulation time and processing cost using simulation input parameters SI2 in table 2 using URS



(a)



(b)

Fig. 6. Simulation time and processing cost using simulation input parameters SI2 in table 2 using RMQ.

The effect of varying the number of resources on the total simulation time is shown in Figures 7 and 8 using UFF and URS models. The simulation time in case of using sevenresources is less than the Simulation time when using four resources.

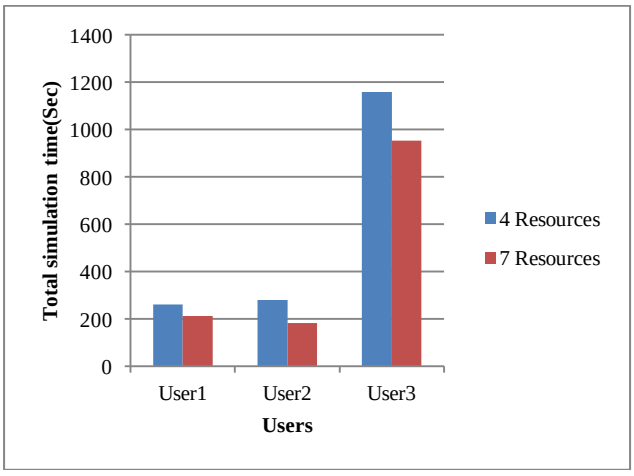


Fig. 7. Effect of varying the number of resources on total simulation time of jobs for different users using UFF.

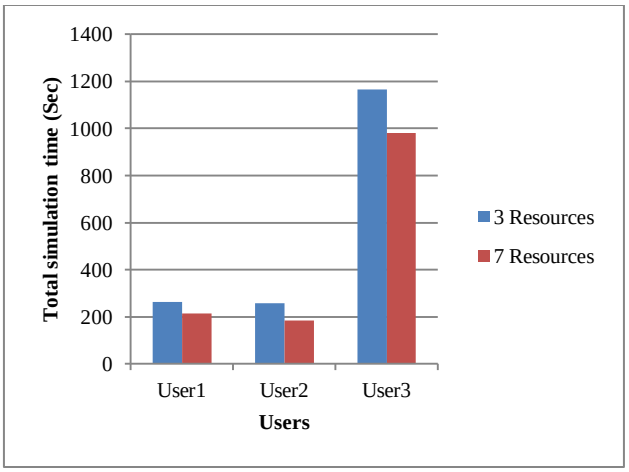


Fig. 8. Effect of varying the number of resources on total simulation time of jobs for different users using URS.

Figure 9 shows the results of total simulation time in case of using simulation input parameters SI3in table 3. The figure shows the simulation time of executing different number of gridlets of various users using 5 resources and different granularity time.Before starting simulation, thegiven granularity time is multiplied by the resource processing capability (MIPS). The result is the total (MI) that the resource can process within the given granularity time. Hence, higher granularity time means that the total (MI) that the resource can process will be also higher. Results shown are for experiments using simulation input parameters SI3 in table 3 and different values for granularity time: 5, 10, 15, and 20.The results show that the total simulation time for granularity time of 20 seconds is less than the total simulation time for granularity time of 15, 10 and 5 seconds.The 100 gridlets of user1 are grouped in three groups when granularity time is 5 seconds and are grouped in one group when granularity time is 10, 15, and 20. When the granularity time is equal to5, the product of granularity time and the resource's MIPS is equal tothe total MI that the resource can process within 5 seconds which is less than the total MI that the resource can process in 10, 15 and 20 seconds.When granularity time is less, more resources are needed to process the given gridlets within the same granularity time.

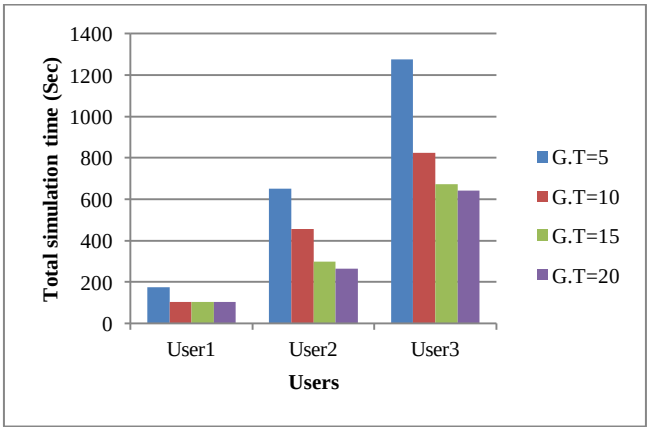


Fig. 9. Effect of varying the granularity time on total simulation time of jobs for different users usingsimulation input parameters SI3 in table 3.

Comparing the results of experiments with and without grouping, it is found that the total simulation time is reduced when grouping is applied reaching 9% to 33% of the total simulation time obtained without grouping.

The actual percentage depends on the number of gridlets and the processing requirements of each gridlet. Total cost in case of using grouping strategy is reduced reaching 52% to 91% of the total cost obtained without using grouping strategy.

Tables 4, 5 and 6 show the load distribution of gridlets on resources using the three proposed models.

TABLE IV. LOAD DISTRIBUTION ON GRID RESOURCES FOR USER1 USING SIMULATION INPUT PARAMETERS SI1 AND THE UFF SCHEDULING MODEL.

Grouped-Gridlet ID	Gridlets	Resource Name
0	0-32	R7
1	33-62	R5
2	63-81	R3
3	82-90	R1
4	91-99	R7

TABLE V. LOAD DISTRIBUTION ON GRID RESOURCES FOR USER1 USING SIMULATION INPUT PARAMETERS SI1 AND THE URS SCHEDULING MODEL.

Grouped-Gridlet ID	Gridlets	Resource Name
0	0-29	R1
1	30-38	R7
2	39-57	R3
3	58-88	R5
4	89-99	R7

TABLE VI. LOAD DISTRIBUTION ON GRID RESOURCES FOR USER1 USING SIMULATION INPUT PARAMETERS SI1 AND THE RMQ SCHEDULING MODEL.

Grouped-Gridlet ID	Gridlets	Resource Name
0	0-8	R1
1	9-39	R7
2	40-69	R5
3	70-89	R3
4	90-99	R1

V. CONCLUSION AND FUTURE WORK

Three models for scheduling fine-grained jobs in a grid computing environment are presented. First, the jobs are grouped to reduce the overall communication time incurred by sending individual jobs to grid resources. Then, grouped jobs are mapped to resources based on the proposed UFF, URS and RMQ scheduling models.

Our experiments were conducted using the Gridsim toolkit.Results indicate that total simulation time and cost are improved by grouping fine-grained jobs. Total simulation time with grouping is 9%-33% of that without grouping. Total cost in case of grouping reaches 52%-91% of that without grouping.

Furthermore, grouping enhancesutilization of resources processing capability. The grouping strategy is based on both the processing requirements of individual jobs and the processing capability of resources.

A load balancing scheduling model is also presented. The queue length of a resource is taken into consideration when a resource is selected. Granularity time has been used to indicate the number of gridlets that can be processed by a resource within a specified time. Results show that the total simulation time decreases as granularity time increases.

Future workin this area will be directed towards developing a grouping model based onthe bandwidth of resources in addition to their processing capability.

REFERENCES

[1] Er.Sourabh Budhiraja, “A Dynamic Load Balancing Approach in Grid Environment”, International Journal of Engineering Research and Technology (IJERT), Vol.1, Issue 9, November 2012.

[2] Dinesh S.Gawande, Rajesh C.Dharmik and Chanda Panse, “A Load Balancing in Grid Environment”, International Journal of Engineering Research and Applications (IJERA), Vol.2, Issue 2, PP.445-450, March-April 2012.

[3] Simrat Kaur and Sarbjeet Singh, “Comparative Analysis of Job Grouping based Scheduling Strategies in Grid Computing”, International Journal of Computer Applications, Vol.43, No.15, April 2012.

[4] P.Suresh and P.Balasubramanie, “Grouping Based User Demand Aware Job Scheduling Approach For Computational Grid”, International Journal of Engineering Science and Technology, Vol.4, No.12, December 2012.

[5] Manoj Kumar Mishra, Prithviraj Mohanty, G.B.Mund, “A Time-minimization Dynamic Job Grouping-based Scheduling in Grid Computing”, International Journal of Computer Applications, Vol.40, No.16, February 2012.

[6] Manpreet Singh, Sandip Kumar Goyal and Vishal Gupta, “An Adaptive Load Balancing Algorithm for Computational Grid”, Journal of Engineering and Technology, Vol.1, Issue 2, July-December 2011.

[7] S.Gomathi and D.Manimegalai, “An Analysis of MIPS Group Based Job Scheduling Algorithm with other Algorithms in Grid Computing”, International Journal of Computer Science Issues (IJCSI), Vol.8, Issue 6, No.3, November 2011.

[8] Simarjit Kaur and Sukhjit Singh, “Role Based Access Control For Grid Environment Using Gridsim”, Journal of Engineering Research and Studies (JERS), Vol. I, Issue I, PP.111-117, July-September 2010.

[9] Raksha Sharma, Vishnu Kant Soni, Manoj Kumar Mishra, Prachet Bhuyan and Utpal Chandra Dey, "An Agent Based Dynamic Resource Scheduling Model with FCFS-Job Grouping Strategy in Grid Computing", World Academy of Science, Engineering and Technology Journal, vol. 64, PP.467-471, 2010.

[10] Vishnu Kant Soni, Raksha Sharma, and Manoj Kumar Mishra, "Grouping-Based Job Scheduling Model In Grid Computing", World Academy of Science, Engineering and Technology Journal, vol. 65, PP.781-784, 2010.

[11] Yeqing Liao and Quan Liu, “Research on Fine-grained Job Scheduling in Grid Computing”, Modern Education and Computer Science journal, PP.9-16, 2009.

[12] T.F Ang, W.K.Ng, T.C Ling, L.Y. Por and C.S. Liew, “A Bandwidth-Aware Job Grouping-Based Scheduling on Grid Environment” Information Technology Journal, Vol.8, NO.3, pp. 372-377, 2009.

[13] Ng Wai Keat, Ang Tan Fong**,** Ling Teck Chaw and Liew Chee Sun "Scheduling Framework For Bandwidth-Aware Job Grouping-Based Scheduling In Grid Computing", Malaysian Journal of Computer Science, Vol.19, No. 2, pp. 117-126, 2006.

[14] Nithiapidary Muthuvelu, Junyang Liu, Nay Lin Soe, Srikumar Venugopal, Anthony Sulistio and Rajkumar 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.

[15] Vishnu Kant Soni, Raksha Sharma, Manoj Kumar Mishra and Sarita Das, "Constraint-Based Job and Resource scheduling in Grid Computing", 3rd International Conference On Computer Science and Information Technology, IEEE, 2010.

AUTHORS PROFILE

Rabab Mohamed Ezzat is currently a Masters Student at the Computer Science Department, Faculty of Computers and Information, Helwan University, Cairo, Egypt. She received her B.Sc. in Computer Science from Helwan University, Cairo, Egypt. She worked as Teaching Assistant in Modern Sciences and Arts University in Egypt for three years. Her current research interests include parallel computing, computer networks and human computer interaction.

Amal Elsayed Aboutabl is currently an Assistant Professor at the Computer Science Department, Faculty of Computers and Information, Helwan University, Cairo, Egypt. She received her B.Sc. in Computer Science from the American University in Cairo and both of her M.Sc. and Ph.D. in Computer Science from Cairo University. She worked for IBM and ICL in Egypt for seven years. She was also a Fulbright Scholar at the Department of Computer Science, University of Virginia, USA. Her current research interests include parallel computing, image processing and software engineering.

MostafaSami M. Mostafais currently a Professor of computer science, Faculty of Computers and Information, Helwan University, Cairo, Egypt. He worked as an Ex-Dean of faculty of Computers and Information Technology, MUST, Cairo. He worked also as an Ex-Dean of student affairs and Ex-Head of Computer Science Department, faculty of Computers and Information, Helwan University, Cairo, Egypt. He is a Computer Engineer graduated 1967, MTC, Cairo, Egypt. He received his MSC 1977 and his PhD 1980 from University of Paul Sabatier, Toulouse, France. His research activities are in Software Engineering and Computer Networking. He is awarded supervising more than 80 Masters of Sc. and 18 PhDs in system modeling and design, software testing, middleware system development, real-time systems, computer graphics and animation, virtual reality, network security, wireless sensor networks and biomedical engineering.