An Adaptive Fault Tolerant Scheduling System for Desktop Grid

Dr. Jyoti Bansal[1,]* and Dr. Geeta Rani[2]

[1] Baba Farid College of Engg & Technology, Bathinda , Punjab Technical University, erjyoti.2009@rediffmail.com
[2]Department of Computer Applications, Rayat Bahra University, Mohali, mailtogeeta@gmail.com

Abstract

In Desktop Grid, managing faults is very crucial and challenging job.So a fault tolerant system is mandatory requirement in desktop grid for fault identification and their resolution.A fault tolerant system allows applications to continue execution despite having faults without termination.In this paper,an adaptive fault tolerant scheduling system is presented that combines dynamic replication with rescheduling.The system initially schedules jobs depending upon the completion time and fault rate of resources and then fault-tolerant strategies are applied depending upon availability of resources.To measure the performance,experiments has been carried out and it has been observed that proposed scheduling system outperforms by a factor of 4.8% w.r.t. Average task response time and 0.02% w.r.t. Average flowtime as compared to existing system.On the other hand,there is no significant improvement is observed when BoT completion time and average execution time are compared to existing system.

Keywords: Grid Computing, Scheduling, fault tolerant, dynamic replication.

Received on 26 February 2019, accepted on 29 April 2019, published on 06 May 2019

doi: 10.4108/eai.13-7-2018.158528

1. Introduction

Internet proves to be a boon when it comes to communication and sharing, along with the availability of powerful computers and high speed communication networks for sharing resources has led to the sharing of spatially distributed resources and using it as an integrated computing machine. A wide range of resources are joining together to form a cluster to use them as a single integrated resource, thereby known as Grid Computing [1].

Specifically Grid computing can be elaborated not only as a coordinated sharing of resources but also as a solution to the complexity of dynamic and multi-organization collaborations[2]. It can be inferred that Grid computing can be used to solve a single yet huge problem, with the cluster of multiple resources which act as a single computational machine.

Grid computing has a wide range of applications both in the field of science and industry. Some of the applications include concentrated to data, information and computation [3]. A coordinated resource sharing is required for all the applications to compute, as the sharing is not only the exchange of files but rather resources are directly accessed such as software, memory, data, information and so on. Along these lines, sharing is highly restricted with resource contributors and clients those are accessing these resources. Organizations and/or foundations have to delineate sharing regulations, known as Virtual Organization (VO) [4].

In Desktop Grid, Information-free schedulers are using much more resources than necessary, so cannot make use of full potential of Desktop Grid. Prior information about resources and tasks will help in taking better scheduling decisions. For example if the scheduler has the knowledge about the resource reliability then it can be decided that which resource is able to complete the execution of task efficiently. Due to diverse nature of Desktop Grid, managing faults is very crucial and challenging task. So a fault tolerant system is a mandatory requirement in Desktop Grid for identification of faults and their resolution. A fault tolerant system allows

*Corresponding author. Email: erjyoti.2009@rediffmail.com applications to continue its execution despite having faults without termination. Also the system has to maintain a bare minimum quality of service.

2. Related Work

Three fundamental challenges that portray Grid computing are: diversity, scalability and element flexibility. A Grid includes a variety of heterogeneous resources and might traverse various organizational domains. As the size of Grid increases, raises the issue of performance deprivation. Finally, in a Grid with a variety of resources, the likelihood of resource failures is normally high. In addition, the hosts those are contributing in terms of donating their resources can be recovered by them whenever required, and is difficult to know ahead of time whether and when they will get to be distinctly accessible again. The resources administrators or applications must tailor their conduct effectively in order to get the maximum performance from the accessible resources and administrations.

Despite these issues, Grid computing must offer a simple way to use or access unlimited computing and heterogeneous resources, so it must have the ability to find, allot, supervise, control and deal with the utilization of system available abilities so as to accomplish different worldwide Quality of Service (QoS) also called resource administration [5]. In conventional computing systems, resource administration is an all-around considered issue. Resource administrators are composed and work under the suspicion that they have complete control of a resource and in this manner can implement or use the approaches required for efficient utilization of that resource in isolation. Tragically, this presumption does not have any significant bearing to Grids in view of the fundamental issues already portrayed. This circumstance is complex by the general absence of information available about the present framework and the contending needs of clients.

Therefore, great parts of the previous work in Grid resource administration concentrated on prevail over the essential problem of diversity, for instance through the various resource administration protocols [6] and communicating resource and task prerequisite mechanisms [7].

The underlying difficulties of Grid computing relating to how to execute a task, how to exchange substantial files, how to deal with various client accounts on various frameworks have been resolved primarily, so clients and scientists can now deal with the issues that will permit more efficient resource utilization. Significant challenges remain, in any case, in seeing how these systems can be effectively consolidated to make consistent virtual perspectives of hidden resources and administrations. The enormous reputation of internet has produced huge prospects for Grid computing i.e., many desktop personal computers, whose idle cycles can be changed to run Grid applications, are joined with wide-zone systems both in the business enterprises and in the home. These new stages for high throughput applications are called Desktop Grids [8].

A Desktop Grid consists of aggregation of spatially distributed resources across diverse associations with varying security constraints and attributes into a single system. In such environment, volunteers allows sharing of resources for solving extensive problems in weather forecasting, high energy physics, etc[9]. The distributed management, heterogeneity and inconsistency of desktop grid resources often results unavailability of resources as compared to traditional distributed system [10]. In Grid systems, a task is said to be failed if an allocated resource is not able to complete that task within the given time period [11]. Due to these failures, application performance degrades in terms of execution time. Moreover providing fault tolerance despite maintaining system performance in terms of application execution time is a challenging job [12].

Anousha et al. [13] presented a scheduling system that depends upon estimating the completion time of the tasks on each of resources. The proposed system will generate the scheduling decisions as per the value of estimated completion time. The results show that the proposed scheduling system improves total completion time in comparison to Min-Min strategy. S.K Panda et al. [14] presented a scheduling system that depends upon utilizing the Round Trip Time (RTT) to discover failures and after identification of failures checkpointing strategy is being used to recover from failures. Results show that the proposed scheduling system improves total completion time and lessen the makespan in comparison to Min-Min and Max-Min strategies. In N.M. Reda et al. [15], the proposed strategy is based on finding appropriate resources by getting the average value via sorting list of completion time of each task. Finally, the task having the maximum average is allocated to the machine that has the minimum completion time. Results show that the proposed strategy outperforms almost other strategies in terms of resources utilization and makespan. K. Kousalya et al. [16] proposed a QoS based Task Rescheduling algorithm (QTR) in which scheduled tasks are collected and then rescheduled using Minimum Completion Time(MCT) value. The results of the computations show that the QTR algorithm reduces makespan in comparison to existing strategies. J.Y Maipan et al. [17] proposed an algorithm MinExt which calculates the average completion time of all tasks. Then tasks having more than average completion time value are scheduled first followed by the set of tasks less than or equal to average completion time value. The results indicate that the proposed strategy minimizes total completion time value and utilizes the idle resources effectively in comparison to existing strategies.

Fault tolerant systems can be proactive or post-active based upon whether they handled prior to scheduling of tasks on desktop grid resources [18] or otherwise. Nazir et al. [18] proposed a fault tolerant scheduling system that works on retaining the value of fault index to handle job failures. In this paper grid scheduler generates the scheduling decisions on the basis of value of fault index.

Nandagopal et al. [12] proposed another scheduler that adds response time in addition to use fault index to make scheduling decisions. In [19] Mohammed Amoon et al. presented a fault tolerant scheduler that is based on using a scheduling indicator to select resources. This scheduling indicator consists of response time and fault rate of Grid resources. During scheduling the proposed scheduler will use this scheduling indicator to generate scheduling decisions. A most recent scheduling system presented by R.M.R Kovvur et al. [20] incorporated rescheduling on account of resource failures during execution of tasks.

So it has been seen that due to diverse nature of Desktop Grid, a single strategy does not work well in all situations.

3. Proposed Methodology

The main contribution of the proposed system is to present an adaptive fault tolerant scheduling system that combines dynamic replication with rescheduling. Initially the scheduling is being performed on the basis of completion time and fault rate of resources. Furthermore an appropriate fault tolerant strategy is used depending on the availability of resources. If the number of available resources are more than the number of tasks requesting for allocation then dynamic replication is preferred and if the number of resources are less as compared to number of tasks then rescheduling is preferred. The proposed system is then compared with the system presented in [15]. The improvement in terms of terms of average task response time and average flowtime value is clearly presented.

At the first stage the proposed system schedules the resources to tasks on the basis of scheduling criteria. Then in the second stage, the system checks the number of resources available. In case of more number of available resources in comparison to tasks then the system computes a dynamic threshold value for each task and replicates the task on multiple resources and if the number of available resources are less in comparison to tasks requesting for allocation then the proposed system perform rescheduling on account of resources failures during execution. The main components of proposed system and their interconnections are shown in figure 1.

3.1. User Interface/Desktop Grid Portal

Through this interface user submits the jobs to the resource broker for execution along with information like number of jobs, type of job and size of jobs etc.

3.2. Resource Broker

The Resource Broker performs the following functions:

a) Send request to Grid Information Server (GIS) for the number of available resources.

b) Select the best list of resources in terms of scheduling criteria (Sc).

c) Sort the list of resources as per the value scheduling criteria (Sc).

d) Communicates with Desktop Grid Scheduler and send the sorted list of resources along with the list of tasks to be executed.

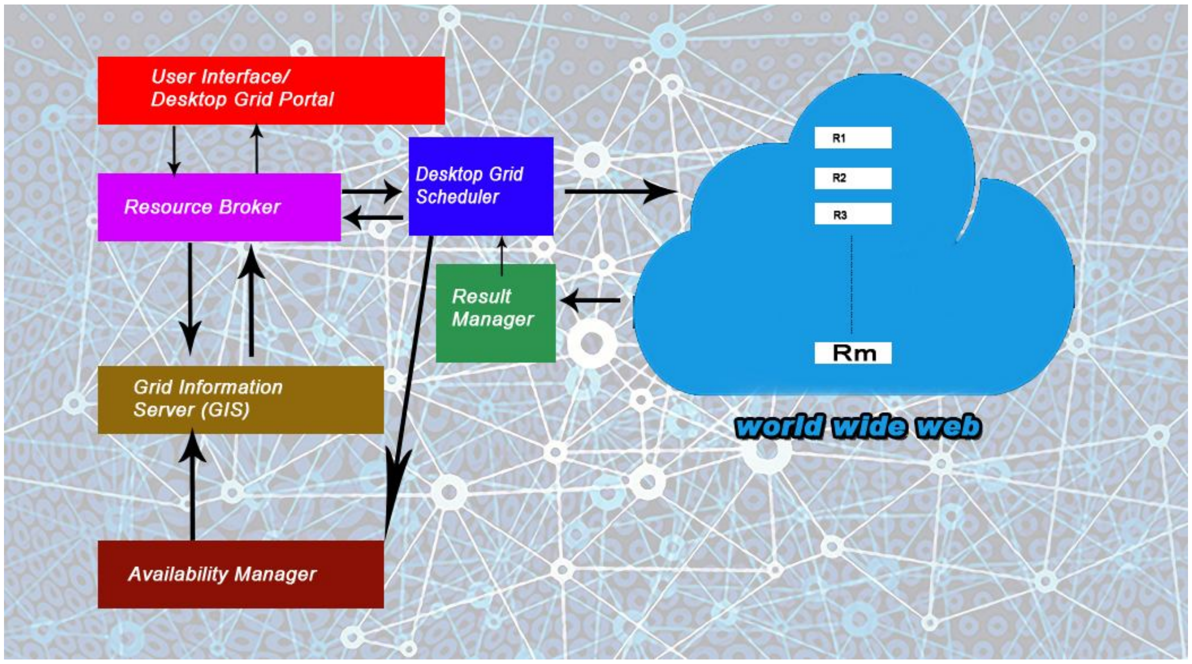


Figure 1. Adaptive Fault Tolerant Scheduling System

3.3. Desktop Grid Scheduler

The Desktop Grid Scheduler performs the following functions:

a) Allocates tasks to the first available resource according to the list received from resource broker. If it fails then will go to the next available resource and will continue to do until it find a suitable resource for execution of task.

b) Maintains a list of faulty resources

c) Will check the number of available resources. If number of available resources is more in comparison to number of tasks then the scheduler will do dynamic replication for each task. And in case of less number of resources in comparison to number of tasks then the scheduler will do rescheduling on account of resource failures.

d) Communicates with availability manager and send the list of faulty resources along with fault rate value.

3.4. Fault Tolerant Manager

The fault tolerant manager performs the following functions:

a) Keep updating the resource history as per the value of fault rate received from desktop grid scheduler.

b) Keep updating Grid information server with updated resource information along with the number of available resources.

3.5. Grid Information Server (GIS)

The Grid Information Server performs the following functions:

a) Will send the available resource list to the resource broker as per the information received from availability manager.

3.6. Outcome Manager

The Grid Information Server performs the following functions:

a) Will send the available resource list to the resource broker as per the information received from availability manager.

4. Adaptive Fault Tolerant Scheduler's Process

Desktop Grid Scheduler receives tasks for execution from user interface along with information about tasks i.e. number of tasks, type of task and size of task and so on. Then it allocates task to the most efficient resource for execution. Efficient resource selection is based upon value of fault rate and completion time value stored in the history of Grid Information server. The fault rate and completion time is defined as follows:-

Fault Rate (Fr): It is the ratio of the number of tasks failed to execute to the total number of tasks. Fault rate is maintained and updated dynamically in resource history table. Thus fault Rate (Fr) of jth resource is:

$$Fr_j = T_f*100/N \ . \qquad (1)$$

Here T_f is number of tasks failed to execute (T_f)
N is the total number of tasks

Completion time (C_{ij}): It can be computed by adding the Expected Execution Time (E_{ij}) of Job J_i on Resources R_j & ready time of resource R_j.

$$C_{ij} = E_{ij} + R_j \qquad (2)$$

The value of fault rate (Fr) and completion time (C_{ab}) is used by the desktop grid scheduler during taking scheduling decisions. The resource having the minimum value of fault rate (Fr) and completion time (C_{ab}) is called as most efficient resource.

To achieve its objective, the scheduler will calculate the value of Scheduling criteria (Sc) as per the following formula

$$Sc_j = C_{ij} * (1+Fr_j) \qquad (3)$$

During generating scheduling decision the value of scheduling criteria (Scj) is utilized. The most efficient resource selection will be done with minimum value of scheduling criteria (Scj).

The flow of work of the proposed system is shown by the following steps.

1. Resource broker collects tasks for execution from user Interface along with information like number of jobs, type of job and size of jobs etc.
2. Resource Broker will send request to Grid Information Server (GIS) for the number of available resources.
3. Grid Information Server (GIS) will send the available resource list to the resource broker as per the information received from availability manager.
4. After getting the resource list from GIS, the resource broker will performs the following functions:
    a) Select the best list of resources in terms of scheduling criteria (Sc).
    b) Sort the list of resources as per the value of scheduling criteria (Sc).
    c) Communicates with Desktop Grid Scheduler and send the sorted list of resources along with the list of tasks to be executed.
5. After getting the resource list along with tasks for execution the Desktop Grid scheduler will do the following tasks:-
    a) Allocates tasks to the first available resource according to the list received from resource broker. If it fails then will go to the next available resource and will continue to do until it find a suitable resource for execution of task.
    b) Maintains a list of faulty resources
    c) Will check the number of available resources. If number of available resources is

more in comparison to number of tasks then the scheduler will do dynamic replication for each task. And in case of less number of resources in comparison to number of tasks then the scheduler will do rescheduling on account of resource failures.

d) Communicates with fault tolerant manager and send the list of faulty resources along with fault rate value.

6. Fault tolerant manager will keep updating the resource history as per the value of fault rate received from desktop grid scheduler and also updates Grid information server(GIS) with updated resource information along with the number of available resources.

7. Outcome manager will send gathered outcomes of tasks to the desktop grid scheduler those have successfully completed their execution.

8. Finally the desktop grid scheduler will send the result back to the resource broker and the resource will send the collected result to the user interface.

5. Results and Discussion

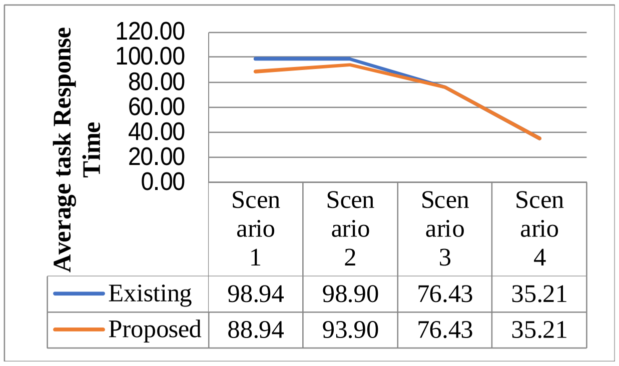
The proposed scheduler, an adaptive fault tolerant scheduling system is compared with existing scheduling system for performance analysis by using Java based simulation environment known as GridSim Toolkit. The performance analysis is based upon average task response time, BoT completion time (makespan), average execution time and average flowtime.

In order to verify proposed scheduling system in comparison with the existing scheduling system [7], number of experiments has been performed. The factors used for simulation are shown in Table 1.

Table 1. Factors for Simulation

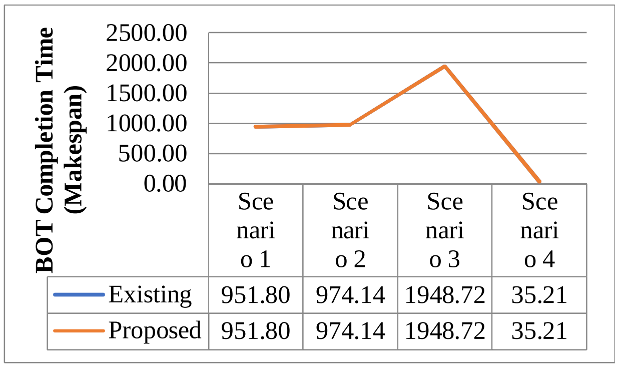
Factors	Scenario			
	1	2	3	4
Number of tasks	10	20	80	160
Number of resources	40	40	40	40
Processor speed (MIPS)	100-300			
Task size(MI)	18850-113100			

For exhaustive analysis of the proposed strategy, four scenarios are taken in which the number of resources are kept constant i.e. 40 in all cases. In first two scenarios, there is less number of tasks in comparison to number of available resources and in last two scenarios, the number of task are more in comparison to number of resources. For comparison purpose 25% and 50% of tasks in comparison to 40 numbers of resources are taken in first two scenarios and 200% and 400% of tasks in comparison to 40 numbers of resources are taken in last two scenarios. In each scenario, 30-35% resource failures are induced for analysis. Random failures are induced with the assumption that resources may fail anytime. The complete comparison analysis w.r.t. average task response time, BoT completion time also called makespan, average execution time and average flowtime with various scenarios is illustrated in figures 2, 3, 4 and 5 respectively.



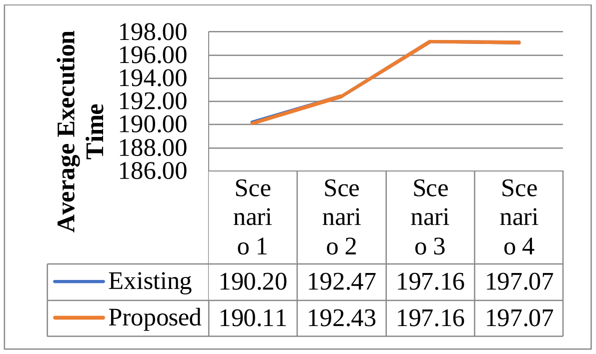
	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Existing	98.94	98.90	76.43	35.21
Proposed	88.94	93.90	76.43	35.21

Figure 2. Comparison of existing & proposed system w.r.t Average Task Response Time



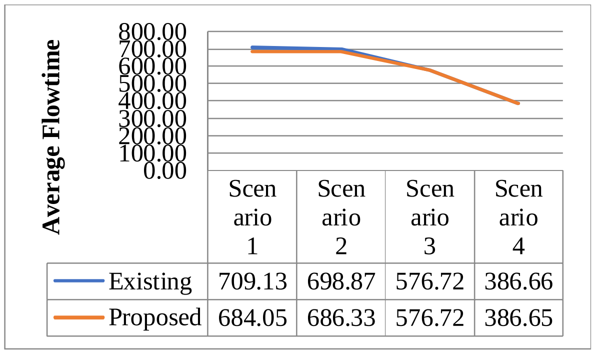
	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Existing	951.80	974.14	1948.72	35.21
Proposed	951.80	974.14	1948.72	35.21

Figure 3. Comparison of existing & proposed system w.r.t BoT Completion Time (Makespan)



	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Existing	190.20	192.47	197.16	197.07
Proposed	190.11	192.43	197.16	197.07

Figure 4. Comparison of existing & proposed system w.r.t Average Execution Time



	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Existing	709.13	698.87	576.72	386.66
Proposed	684.05	686.33	576.72	386.65

Figure 5. Comparison of existing & proposed system w.r.t Average Flowtime

As per the analysis, it has been observed that the proposed scheduling system outperforms existing scheduling system by a factor of 4.8% w.r.t. Average task response time. Furthermore when Average flowtime is taken into account then the proposed system shows a performance improvement of 0.02% as compared to existing scheduling system. Additionally, when BoT completion time and average execution time are taken in to account, then the performance of the proposed scheduling system remains same with the existing scheduling system. So it has concluded that the proposed system outperforms in every case as compared to the existing scheduling system. Furthermore the proposed system must be preferred for the applications where high response time is desired.

6. Conclusion and Future Scope

In this paper an adaptive fault tolerant scheduling system is presented that combines dynamic replication with rescheduling. The system initially schedules jobs depending upon the completion time and fault rate of resources. Furthermore, if the number of available resources are more than the number of tasks requesting for allocation then dynamic replication is preferred and if the number of resources are less as compared to number of tasks then rescheduling is preferred. The proposed adaptive scheduling system is evaluated under different scenarios with latest fault tolerant scheduling system that depend upon using the rescheduling on account of resource failures during execution of tasks. The parameters used for evaluation are average task response time, BoT completion time (makespan), average execution time and average flowtime.

As per the analysis, it has been observed that the proposed scheduling system outperforms existing scheduling system by a factor of 4.8% w.r.t. Average task response time. Furthermore when Average flowtime is taken into account then the proposed system shows a performance improvement of 0.02% as compared to existing scheduling system. Additionally, when BoT completion time and average execution time are taken in to account, then the performance of the proposed scheduling system remains same with the existing scheduling system. So it has concluded that the proposed system outperforms in every case as compared to the existing scheduling system. Furthermore the proposed system must be preferred for the applications where high response time is desired.

In future, scheduling can be done for dependent tasks also. Further improvements can also be done by specifying the time zones on which scheduling can be done because especially in night time zones resources are almost free to use. Replication and Checkpointing policies can be improved further. Furthermore, more than one BoTs concurrently can be considered for execution.

References

[1] Baker, M., Buyya, R. and Laforenza, D. (2000) The Grid: International efforts in global computing. In *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, IAquila, Rome, Italy, July 31-August 6, 2000.

[2] Nabrzyski, J., Schopf, J.M. and Weglarz, J. eds. (2012) Grid resource management: state of the art and future trends. *Springer Science and Business Media* **64**.

[3] Foster, I. and Kesselman, C. eds. (2004) *The Grid2: Blueprint for a New Computing Infrastructure*, 2nd edn. (Boston: Morgan Kaufmann Publishers Inc., Elsevier).

[4] 4. Foster, I., Kesselman, C. and Tuecke, S. (2001) The anatomy of the grid: Enabling scalable virtual organizations. *International journal of high performance compx`uting applications* 15(3): 200-222.

[5] Foster, I., Kesselman, C. (2004) The Grid in a Nutshell. In Nabrzyski, J., Schopf, J.M. (eds.) *Grid Resource Management: State of the Art and Future Trends* (Boston: Kluwer Academic Publisher)

[6] 6. Chandra, P., Fisher, A., Kosak, C., Ng, T.E., Steenkiste, P., Takahashi, E. and Zhang, H. (1998) Customizable resource management for value added network services. In *Proceedings of Sixth International Conference*, Darwin, October 1998 (Darwin: IEEE), 177-188.

[7] Raman, R., Livny, M. and Solomon, M. (1999) Matchmaking: An extensible framework for distributed resource management. *Cluster Computing* **2**(2): 129-138.

[8] Kondo, D., Chien, A.A. and Casanova, H. (2004) Resource management for rapid application turnaround on enterprise desktop grids. In *Proceedings of ACM/IEEE conference on Supercomputing*, Pittsburgh, 2004 (IEEE Computer Society), 17-30.

[9] Khan, F.G., Qureshi, K. and Nazir, B. (2010) Performance evaluation of fault tolerance techniques in grid computing system. *Computers & Electrical Engineering* **36**(6): 1110-1122.

[10] Zheng, Q., Veeravalli, B. and Tham, C.K. (2009) On the design of fault-tolerant scheduling strategies using primary-backup approach for computational grids with low replication costs. *IEEE Transactions on Computers* **58**(3): 380-393.

[11] Nandagopal, M. and Uthariaraj, V.R. (2010) Fault tolerant scheduling strategy for computational grid environment. *International Journal of Engineering Science and Technology* **2**(9): 4361-4372.

[12] Lee, H., Chung, K., Chin, S., Lee, J., Lee, D., Park, S. and Yu, H. (2005) A resource management and fault tolerance services in grid computing. *Journal of Parallel and Distributed Computing* **65**(11): 1305-1317.

[13] Anousha, S. and Ahmadi, M. (2013) An improved Min-Min task scheduling algorithm in grid computing. In *Proceedings of International Conference on Grid and Pervasive Computing*, (Berlin Heidelberg: Springer LNCS), 103–113.

[14] Panda, S.K., Khilar, P.M. and Mohapatra, D.P. (2014) FTM2: Fault tolerant batch mode heuristics in computational grid. In *Proceedings of International Conference on Distributed Computing and Internet Technology*, Feb 2014 (Cham: Springer LNCS), 98–104.

[15] Reda, N.M., Tawfik, A., Marzok, M.A. and Khamis, S.M. (2015) Sort-Mid tasks scheduling algorithm in grid computing. *Journal of advanced research* **6**(6): 987-993.

[16] Kousalya, K. and Kumar, P.K. (2016) QoS based Task Rescheduling in Computational Grid Environment. *Asian Journal of Research in Social Sciences and Humanities* **6**(6): 1975-1991.

[17] Maipan-uku, J.Y., Konjaang, J.K. and Baba, A.I. (2016) New Batch Mode Scheduling Strategy for Grid Computing System. *International Journal of Engineering and Technology* **8**(2): 1314-1323.

[18] 18. Nazir, B., Qureshi, K. and Manuel, P. (2009) Adaptive checkpointing strategy to tolerate faults in economy based grid. *The Journal of Supercomputing* **50**(1): 1-18.

[19] 19. Amoon, M. (2012) A fault-tolerant scheduling system for computational grids. *Computers & Electrical Engineering* **38**(2): 399-412.

[20] Kovvur, R.M.R. and Ramachandram, S. (2016) A Hybrid Fault Tolerant Scheduler for Computational Grid Environment. In *Proceedings of International Congress on Information and Communication Technology*, Jun 2016 (Singapore: Springer), 505–512.