Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Performance evaluation of fault tolerance techniques in grid computing system [☆]

Fiaz Gul Khan [b], Kalim Qureshi [a,*], Babar Nazir [b]

[a] Information Science Department Kuwait University, Kuwait
[b] Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad, Pakistan

A R T I C L E I N F O

Article history:
Received 9 January 2010
Received in revised form 3 February 2010
Accepted 29 April 2010
Available online 26 May 2010

Keywords:
Grid computing
Fault tolerance
Resource management
Distributed system

A B S T R A C T

As fault tolerance is the ability of a system to perform its function correctly even in the presence of faults. Therefore, different fault tolerance techniques (FTTs) are critical for improving the efficient utilization of expensive resources in high performance grid computing systems, and an important component of grid workflow management system.

This paper presents a performance evaluation of most commonly used FTTs in grid computing system. In this study, we considered different system centric parameters, such as throughput, turnaround time, waiting time and network delay for the evaluation of these FTTs. For comprehensive evaluation we setup various conditions in which we vary the average percentage of faults in a system, along with different workloads in order to find out the behavior of FTTs under these conditions. The empirical evaluation shows that the workflow level alternative task techniques have performance priority on task level checkpointing techniques. This comparative study will help to grid computing researchers in order to understand the behavior and performance of different FTTs in detail.

1. Introduction

Grid computing enables it users to use Grid for large-scale compute and data intensive applications, in science, engineering and commerce [1,2]. Such applications include, molecular modeling for drug design [3], brain activity analysis [4], high energy physics, protein modeling, ray tracing [5] and weather forecasting, etc. Grid computing enables these compute intensive applications to integrate large scale, geographically distributed and heterogeneous resources in different administrative domains with different resource usage and security policies in order to compute their jobs.

Fault tolerance or graceful degradation is the property of distributed computing system which distinguishes it from sequential computing. This property enables distributed computation to carry on it computation even on individual component's failure without terminating the entire computation [6,7]. Due to the diverse nature of grid and large scale applications on grid, fault tolerance becomes a challenge on developing, deploying and running applications on the grid environment [8]. Thus, the inclusion of fault tolerance related features in grid computing system is not a supplementary optional feature [9] rather a prerequisite.

Due to scale of complexity and heterogeneous nature of grid as compared to traditional computing systems, existing fault tolerance techniques of traditional systems are not enough to manage the faults in grid computing. Therefore, we require additional fault tolerance technique that could work well in complex and heterogeneous nature of grid. Consequently over

☆ Reviews processed and proposed for publication to the Editor-in-Chief by Associate Editor Dr. Sahin.
* Corresponding author. Tel.: +965 4895351; fax: +965 4985253.
E-mail address: Kalim_Qureshi@hotmail.com (K. Qureshi).

the years researchers have yielded a considerable body of theoretical and practical knowledge of fault detection, handling, and recovery techniques [6]. Rajkumar Buyya, in his taxonomy of workflow management systems for grid computing [10] divided the fault tolerance techniques in to two main levels that are; (a) task level and (b) workflow level fault tolerance. He also cataloged different fault tolerance techniques under the above two levels [10].

Taking into account the large number of different fault tolerance techniques (FTTs) available, choosing the best FTT under different levels of faulty system with different system's performance requirements is not an easy task. The system centric approach of FTT is attractive because it enhanced the system throughput, utilization, and complete execution at the earliest possible time [11] and another benefit of using system centric approach for grid computing is economic profits [12].

In this paper we present a simulation study that comprises of four most commonly used fault tolerance techniques in terms of their impact on the performance of grid computing system. We considered the system centric parameters such as throughput, turnaround time, waiting time, and transmission delay in order to evaluate the relative performance of grid computing system using various fault tolerance techniques. We also considered various simulation conditions in which we vary the average percentage of faults in a system along with different workloads, and showed the experimental results.

Rest of the paper is organized as follow. In Section 2, we discussed different fault tolerance techniques in grid computing, and our choice of FTTs Section 3 contains the performance parameters which we have used. Section 4 includes the methodology that we have adopted while in Section 5 we present experimental results and discussion, finally in Section 6 we draw conclusion of our work.

2. Performance analysis of fault tolerance techniques in grid

In grid computing we can categories resource management and job scheduling in to two broad categories that are (a) system centric and (b) user centric. The system centric approach [23] aims to maximize the overall system utilization by considering the parameters such as throughput, turnaround time, transmission delay, waiting time, etc. On the other hand, in user-centric approach, the main focus is to deliver maximum utility to the grid users based on their QoS requirements, i.e., deadline guarantees by which the user wants his jobs to be completed and/or budget he can pay for the required service. As most of the grid resource management and scheduling system such as Condor [22], AppLeS PST, PUNCH, Netsolve, etc., works on system centric approach [23].

2.1. Fault tolerance techniques

Jia Yu and Rajkumar Buyya in their taxonomy of workflow management systems for grid computing [10] divided the fault tolerance techniques in to two main levels that are (a) task level and (b) workflow level fault tolerance.

2.1.1. Task level failure handling techniques

Task-level techniques [8] refer to recovery techniques that are to be applied in the task level to mask the effect of task crash failures without having to know about the task context [13] (i.e., failure-type independent failures). Task level [14] fault tolerance techniques include:

1. *Retrying* – Retrying technique for fault tolerance is the simplest technique [15] being used. After failure it retries the same task regardless the cause of failure on the same grid resource, up to some threshold value and hopes that failure will not come in successive retries.
2. *Alternate resource* – The alternate resource technique [17] works just like the retry except it retries on the alternate resource rather than retrying on the same resource again and again.
3. *Checkpoint/restart* – The checkpointing technique periodically saves the state of an application [18], and on failure it migrate the task on other resource, and starts the execution from the last saved checkpoint. In general, the checkpoint in the application can be [19] either kernel, user or application level checkpointing.
4. *Replication* – The replication technique in fault tolerance [20,21] runs different replicas of same task on different grid resources simultaneously, hoping that at least one of them will complete successfully.

Because of the simplicity of implementation, retry and alternate resource techniques are being mostly used [23], while due to the complexity in the implementation of replication and checkpointing/restart [22] techniques they have been studied more theoretically, yet few implement replication or checkpointing.

2.1.2. Workflow level failure handling techniques

Workflow level or failure type sensitive failures [8] handling techniques change the flow of execution on failure, based on the knowledge of task execution context. Workflow level techniques [10] can be classified:

1. *Alternative task* – is just like retry technique the only difference is that, it exchanges a task with other implementation of same task but with different execution characteristics, on failure of the first one.

2. *Redundancy* – While for redundancy techniques [23] we require many different implementations of same task with different execution characteristics, which run in parallel as opposed to task level replication technique where same tasks are replicated on different grid resources.
3. *User defined exception handling* – In user defined exception handling technique [16,10] user specifies the particular treatment to workflow of a task on failure.
4. *Rescue workflow* – The rescue workflow technique [10] allows the workflow to continue even if the task fails until and unless it becomes impossible to move forward without catering the failed task.

2.2. Selected FTTs for performance evaluation

We have selected three of task level FTTs and one from workflow level due to two main reasons. First the basic requirement for first three workflow level techniques is that you must have multiple implementations of same task [10] with different execution characteristics. But most of the time this is not feasible to have multiple implementation of same task. Secondly, from literature review we came to know that most of the current projects have different task level FTTs [10], as shown in Table 1. We selected the following four most commonly used FTTs for empirical evaluation study in grid computing system that are; Retrying; Alternate resource; Checkpoint/restart and Alternative task.

3. Performance metrics

The comparative performance of FFTs is measured by standard matrices [12] throughput, turnaround time, waiting time and transmission delay. The terms are defined:

Average throughput – We calculated the average throughput of 'n' number of gridlets submitted divided by the total amount of time T necessary to complete 'n' gridlets successfully i.e.,
Throughput $(n) = n/Tn$, where
n is a total number of gridlets submitted.
Tn is a total amount of time necessary to complete n gridlets.
Average turnaround time – from particular gridlet/job point of view, the important criterion is how long the system takes to execute that gridlet. The interval from the time of submission of a gridlet to the time of completion is the turnaround time. While average turnaround time, is averaged over all gridlets submitted (see Fig. 1).
Average waiting time – is the average amount of time between submitting a gridlet to grid resource and starting the gridlet on one of the executions host (see Fig. 1).
Transmission Delay – Transmission delay [23] is a function of the packet's length and the transmission delay. It has nothing to do with the distance between the two nodes. This delay is proportional to the packet's length in bits; it is given by the following formula:
$D_T = N \,/\, R$, where
D_T is the transmission delay
N is the number of bits, and
R is the rate of transmission (say in bits per second).

Table 1
FTTs used in different projects.

Project name	Fault tolerance techniques used	
DAGMan [23]	*Task level* – Alternate resource – Retrying	*Workflow level* – Rescue workflow
Taverna [22]	*Task level* – Retry – Alternate resource	
GridFlow [23]	*Task level* – Alternate resource	
Askalon [22]	*Task level* – Retry – Alternate resource	*Workflow level* – Rescue workflow
Karajan [23]	*Task level* – Retry – Alternate resource – Checkpoint/restart	*Workflow level* – User-defined exception
Kepler [24]	*Task level* – Alternate resource	*Workflow level* – User-defined exception handling – Workflow rescue
Gridbus workflow [25]	*Task level* – Alternate resource	

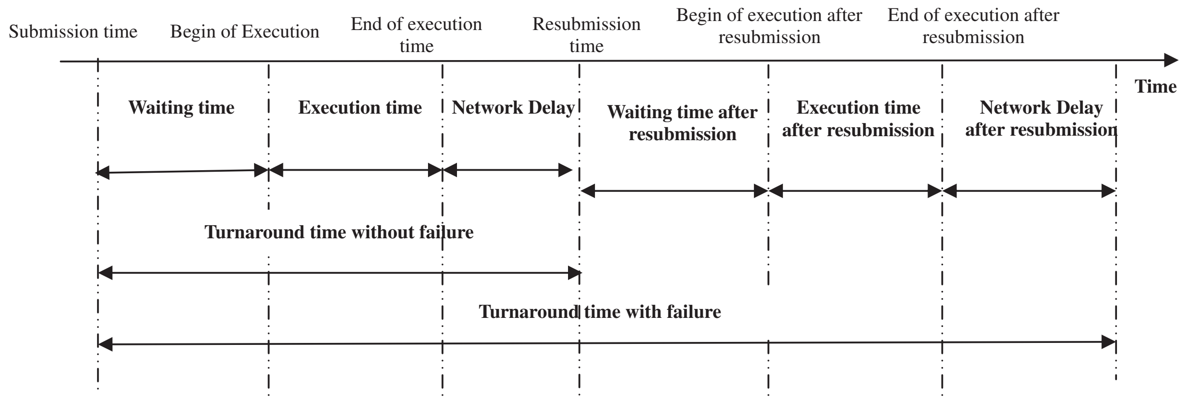
Submission time
Begin of Execution
End of execution time
Resubmission time
Begin of execution after resubmission
End of execution after resubmission
Time
Waiting time
Execution time
Network Delay
Waiting time after resubmission
Execution time after resubmission
Network Delay after resubmission
Turnaround time without failure
Turnaround time with failure

Fig. 1. Definition of timing parameters.

4. Experimental methodology

We carried out our study by mean of GridSim simulator [22]. GridSim supports modeling and simulation of heterogeneous grid resources, users and application models. It provides infrastructure for creation of application tasks, mapping of tasks to resources, and their management. We measured the throughput, turnaround time, waiting time and transmission delay of Retrying, Alternative, Checkpoint and Alternative-task fault tolerance techniques. The experiment setup is same for all FTTs evaluation. In this study we model and create Grid resources, applications and jobs as gridlets in GridSim environment.

4.1. Resource modeling

In our experiments we modeled different space-shared resources; we took the configurations, characteristics, and capabilities of these resources from those in WWG testbed [23].

4.2. Application modeling

In GridSim the job and its length in Million of Instructions, the size of job input and out data in bytes are grouped together and is called a 'Gridlet'. In our experiments we modeled applications with different number of gridlets from 200 to 1000. We have used 56 Mbps (Megabits per second) communication link in our setup. We took the same file size that is 300 MB and length that is 10,000 million of instruction (MI) for all jobs, and these specifications remain same for measuring performance of all fault tolerance techniques.

5. Experimental results and discussion

This section present the measured results of different FTTs. We have conducted different experiments with variation in the total number of jobs/gridlets submitted to the grid resources, by injecting different percentage of failures in a system, and got average throughput, average turnaround time, average waiting time, and average transmission delay of a system using different fault tolerance techniques mentioned in Section 2.2.

5.1. Average throughput

Throughput is an important parameter for determining the performance of different FTTs. It becomes more important when a user have to wait for the completion of all jobs before proceeding further. Figs. 2–6 depicts average throughput of different FTTs considered in our study.

Figs. 2–6 present the measured throughput of all four FTTs, with different percentage of faults injected in a system, and on different number (200, 400, 600, 800 and 1000) of gridlets submitted. With different percentage of faults injected, and number of gridlets submitted, the overall difference among all four FTTs is less than 18% for average throughput.

In general the average throughput of all techniques decreases with increase in the percentage of faults injected and with the increase in the number of gridlets submitted to the system. Among different fault tolerance techniques, average throughput of a system increases when we move from 'retrying' to 'alternative task technique' of fault tolerance, but with some exclusion like in Figs. 2 and 3. In Fig. 2 the average throughput of checkpointing is more than that of Alternative task technique under 25–30% of faults injection. This is because in checkpointing the whole gridlet is divided in to different

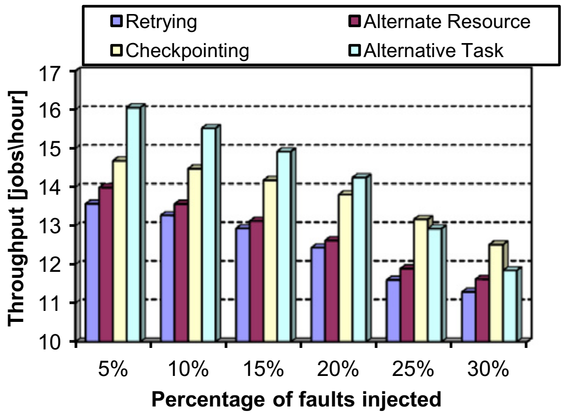

Fig. 2. Total number of gridlets/jobs submitted = 200.

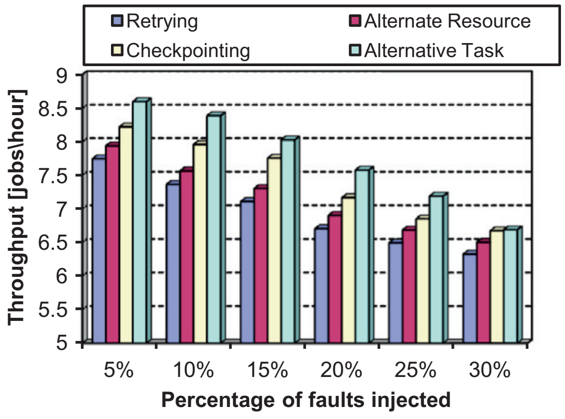

Fig. 3. Total number of gridlets/jobs submitted = 40.

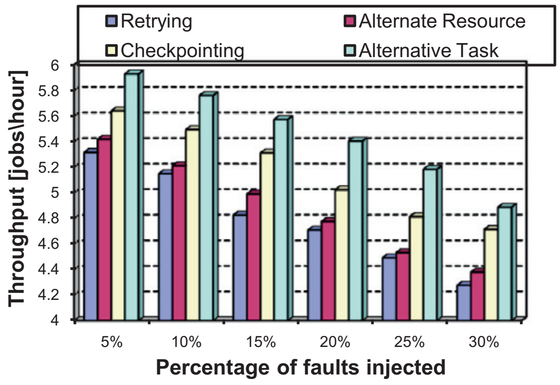

Fig. 4. Total number of gridlets/jobs submitted = 600.

checkpoints, it may happen that on first time when a gridlets fails it would have completed its execution up to some initial checkpoints inserted in the gridlet and you have to resubmit it again for further execution from last successfully completed checkpoint onward. In this way the number of resubmission with checkpointing may increase specially in case when you have more faulty system and with large number of gridlets submitted. This would ultimately increase the network delay in case of checkpointing. Due to this reason the average throughput of checkpointing in Fig. 2 is more than the average throughput of alternative task, but with increase in the number of gridlets submitted the average throughput of alternative task starts increasing then that of checkpointing because of more network delay in checkpointing. The overall difference between ‘retrying’ and ‘alternate resource’ is less than 3.5% for average throughput.

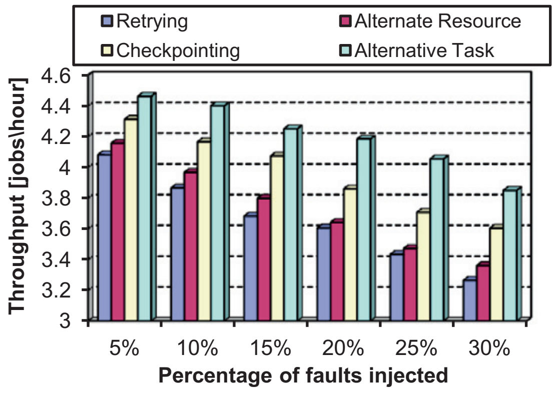

Fig. 5. Total number of gridlets/jobs submitted = 800.

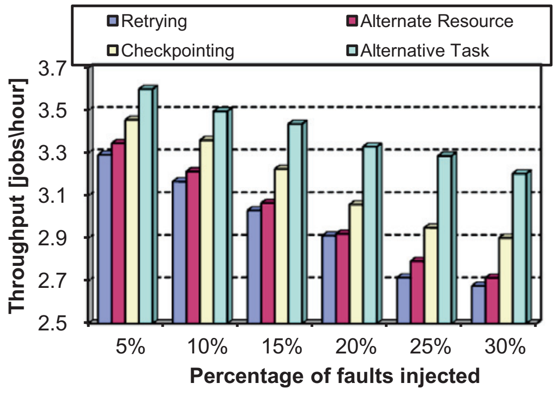

Fig. 6. Total number of gridlets/jobs submitted = 1000.

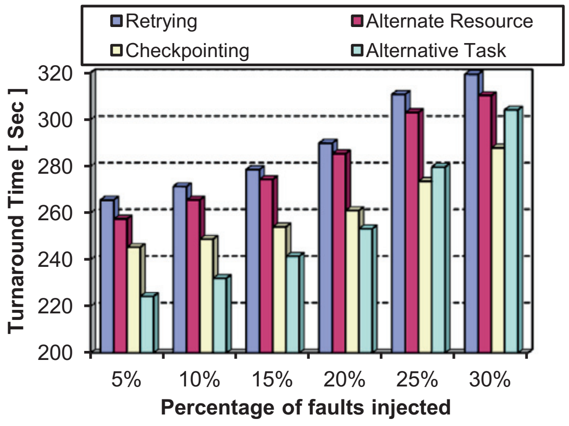

Fig. 7. Total number of gridlets/jobs submitted = 200.

In summary we see that with less number of gridlets submitted that is under 400 and high faulty system task level checkpointing gives high throughput as compare to other FTTs in our experiment. On the other hand, with increase in the number of gridlets submitted as Figs. 4–6 depicts workflow level alternative task technique gives high throughput as compare to other FTTs present in experiment, under all percentages of faults. There is a significant difference between the throughput given by retrying and alternate resource as compared to checkpointing and alternative task under all sorts of conditions.

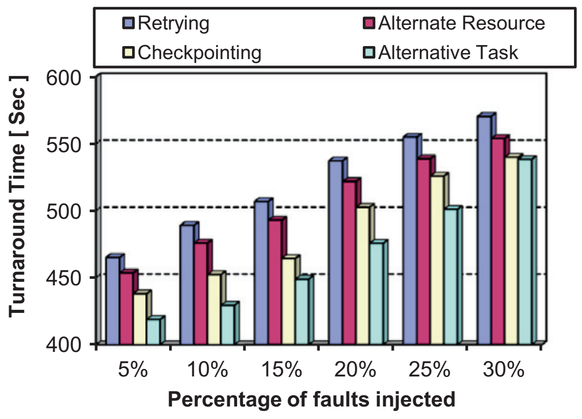

Fig. 8. Total number of gridlets/jobs submitted = 400.

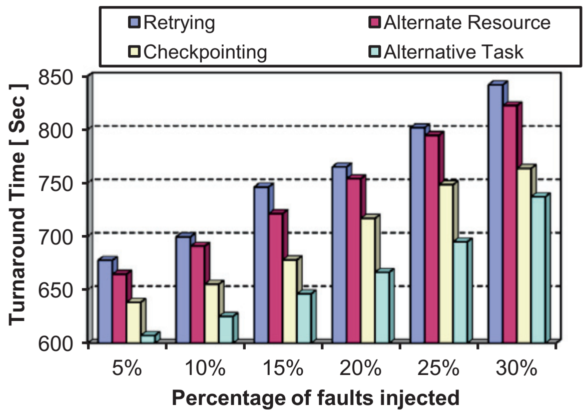

Fig. 9. Total number of gridlets/jobs submitted = 600.

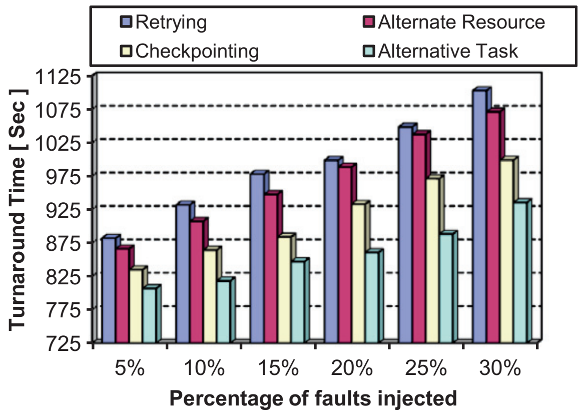

Fig. 10. Total number of gridlets/jobs submitted = 800.

5.2. Average turnaround time

Figs. 7–11 depict ‘turnaround time’ of different FTTs, with different percentage of faults injected in a system, and on different number (in count) of gridlets submitted. With different percentage of faults injected, and number of gridlets submitted, the overall difference among all four FTTs is less than 21% for average turnaround time.

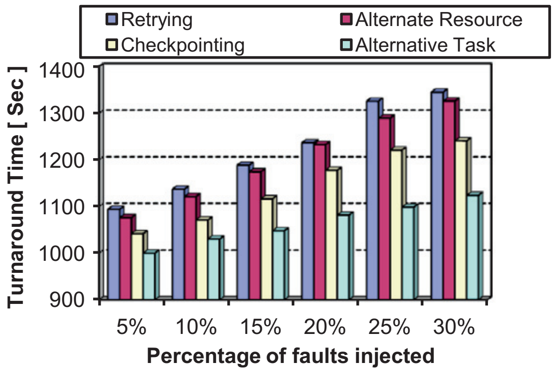

Fig. 11. Total number of gridlets/jobs submitted = 1000.

In general the average turnaround time of all techniques increases with increase in the percentage of faults injected and with the increase in the number of gridlets submitted to the system as shown in Figs. 7–10. Among different fault tolerance techniques in general, average turnaround time also decreases when we move from 'retrying' to 'alternative task technique' of fault tolerance, but with some exclusions. Like in Fig. 7, turnaround time of 'checkpointing' is less than that of 'alternative task technique'.

This is because in 'alternative task' we have different implementations of same task [8] with different executions characteristics, and in our experiment after the failure of first implementation which is fast but unreliable we resubmits the second implementation that is slower than first one but more reliable, which increases the turnaround time of a task/gridlet and it increase more with the increase in percentage of failures injected. But with increase in the number of gridlet in checkpointing the likelihood of resubmitting the failed task also become increases as described in Section 5.2 which eventually increases the turnaround time of checkpointing as compared to that of 'alternative task'. This trend further elaborated when there is a significant increase in the number of gridlets submitted as shown in Fig. 11.

In summary we see that under less numbers of gridlets submitted and presence of high percentage of faults in a system the checkpointing gives good results as compared to other FTTs as shown in Fig. 7. With increase in the number of gridlets submitted its performance goes down with respect to alternative task, but it still works well as compared to retrying and alternate resource under all sorts of condition.

5.3. Average waiting time

Figs. 12–16 depict 'average waiting time' of different FTTs, with different percentage of faults injected in a system, and on different number (in count) of gridlets submitted. With different percentage of faults injected, and number of gridlets submitted, the overall difference among all four FTTs is little less than 21% for average 'waiting time'.

In general the average waiting time of all techniques increases with increase in the percentage of faults injected and with the increase in the number of gridlets submitted to the system. Among different fault tolerance techniques the average waiting time of checkpointing is less in all conditions. The waiting time of alternative task is more because of the slow task that we have resubmitted on failure of fast but unreliable task. Due to the same reason the waiting time of alternative task is also

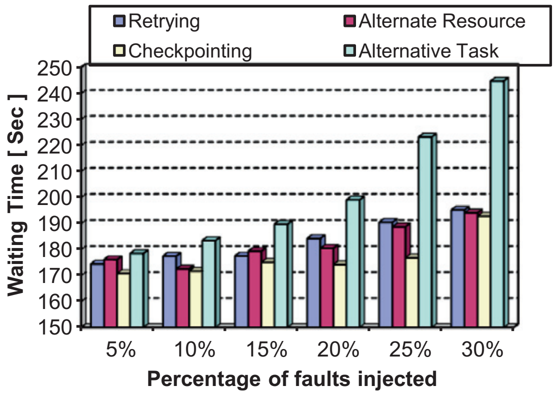

Fig. 12. Total number of gridlets/jobs submitted = 200.

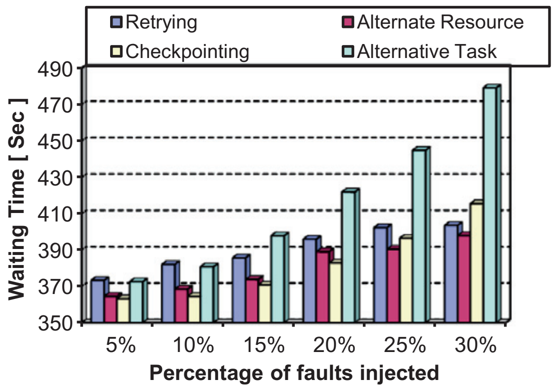

Fig. 13. Total number of gridlets/jobs submitted = 400.

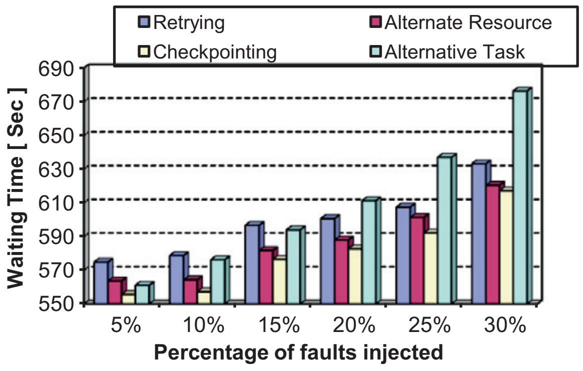

Fig. 14. Total number of gridlets/jobs submitted = 600.

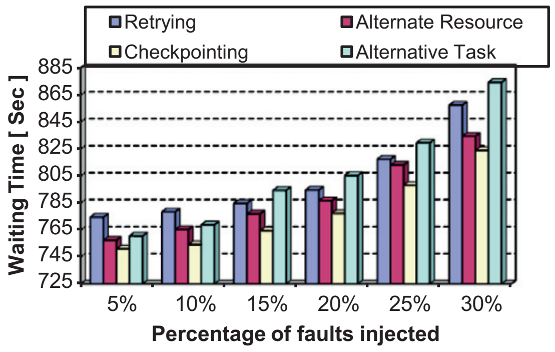

Fig. 15. Total number of gridlets/jobs submitted = 800.

more than that of retrying and alternative task techniques too when there is a high percentage of faults occurrence. While the waiting time of alternate resource technique alters with the load on the resource on which it has been migrated on failure.

5.4. Average transmission delay

Figs. 17–21 depict 'average transmission delay' of different FTTs, with different percentage of faults injected in a system, and on different number (in count) of gridlets submitted. With different percentage of faults injected, and number of gridlets submitted, the overall difference among all four FTTs is less than 9% for average 'transmission delay'.

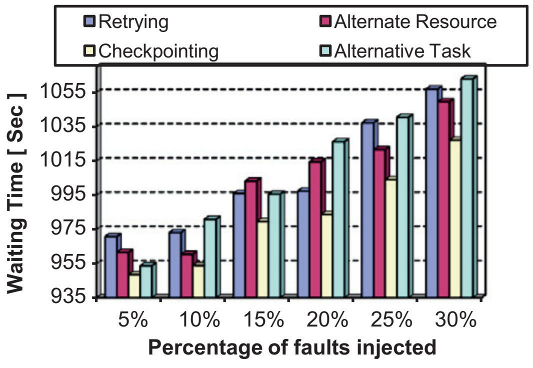

Fig. 16. Total number of gridlets/jobs submitted = 1000.

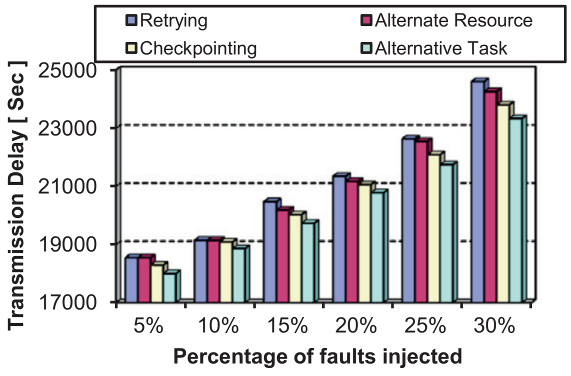

Fig. 17. Total number of gridlets/jobs submitted = 200.

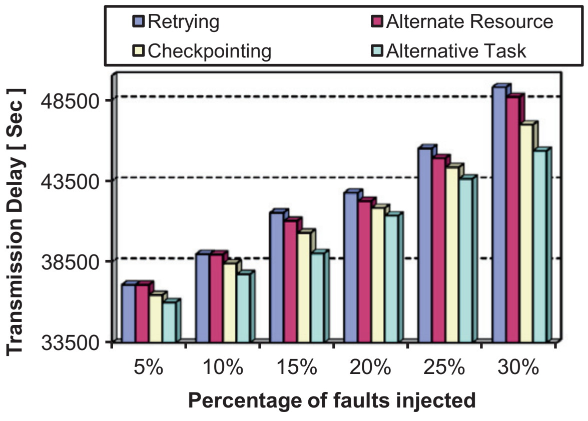

Fig. 18. Total number of gridlets/jobs submitted = 400.

In general the average transmission delay of all techniques increases with increase in the percentage of faults injected and with the increase in the number of gridlets submitted to the system. Among different fault tolerance techniques the average transmission delay of alternative task is less in all conditions, which is the main cause of its high throughput and low turn-around time.

The reason for increase in the transmission delay of checkpointing is the same as described in Section 5.1. The transmission delay of alternate resource and retrying is almost the same, because in both on resubmission gridlet's size remains same and both works on same number of retries on failure.

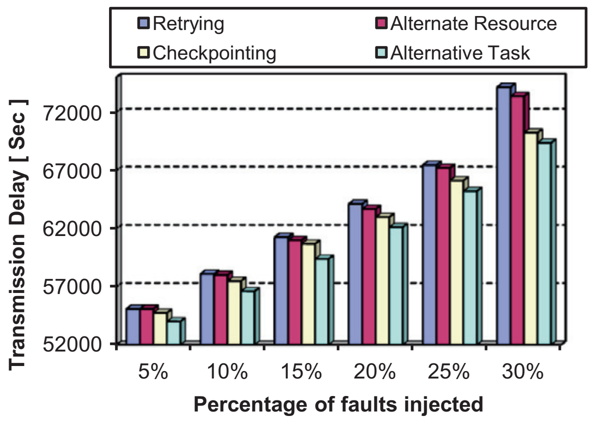

Fig. 19. Total number of gridlets/jobs submitted = 600.

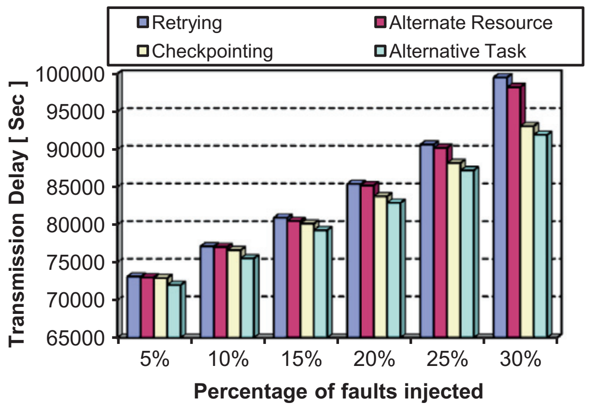

Fig. 20. Total number of gridlets/jobs submitted = 800.

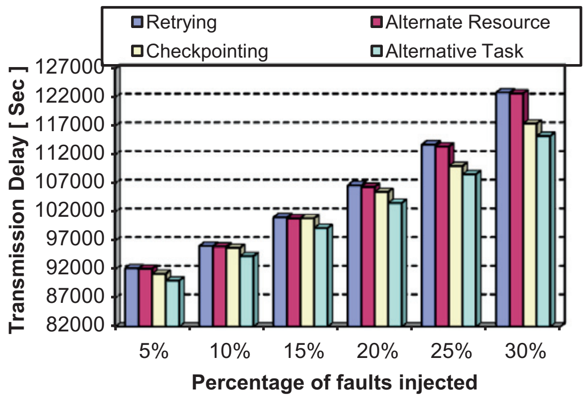

Fig. 21. Total number of gridlets/jobs submitted = 1000.

6. Conclusion

In this paper we carried out a comparative study of four most commonly used fault tolerance techniques in grid computing, these techniques include ‘checkpointing’, ‘retrying’, ‘alternate resource’, and ‘alternative task’. The performance of these techniques is evaluated in different conditions, on different parameters such as throughput, turnaround time, waiting time, and transmission delay. The waiting time of ‘alternative task’ techniques is high it is due to the resubmission of slow task, but it works well under high workloads and with different percentages of faults injected in a system.

The reason for good performance of ‘alternative task’ is to takes less network delay as compare to other FTTs. On the other hand when we have low workload and with high percentage of faults in a system ‘checkpointing’ give better results than alternative task and other FTTs. Our comparative study will help other researchers in order to understand the behavior and performance of different FTTs for Grid computing environment.

References

[1] Fahringer Thomas, Truong Hong Linh. ASKALON: a tool set for cluster and grid computing. Concurrency and Computation: Practice and Experience, vol. 17, Nos. 2–4. NY: Wiley InterScience; 2005. ISSN: 1532-0626.
[2] Zheng Qin, Veeravalli Bharadwaj, Tham CK. Fault-tolerant scheduling for differentiated classes of tasks with low replication cost in computational grids, HPDC’07, June 25– 29, 2007, Monterey, California, USA, ACM 978-1-59593-673-8/07/0006; June 2007. p. 239–40.
[3] Buyya Rajkumar, Abramson David, Venugopal Srikumar. The grid economy, special issue on grid computing. In: Parashar Manish, Lee Craig, editors. Proceedings of the IEEE, vol. 93, No. 3. New York, USA: IEEE Press; 2005. p. 698–714. ISSN: 0018-9219.
[4] von Laszewski Gregor. Workflow Concepts of the Java CoG Kit. J Grid Comput 2006;3(3–4):239–58.
[5] Buyya Rajkumar, Abramson David, Giddy Jonathan, Stockinger Heinz. Economic models for resource management and scheduling in grid computing, concurrency and computation: practice and experience (CCPE), vol. 14, Nos. 13–15. USA: Wiley Press; 2002.
[6] Stelling Paul, DeMatteis C, Foster Ian, Kesselman Carl, Lee Craig, Von Laszewski Greger. A fault detection service for wide area distributed computations. In: 7th IEEE international symposium on high performance distributed computing, Washington, DC, USA; July 1998. p. 268, ISBN: 0-8186-8579-4.
[7] Fault-tolerant System: http://en.wikipedia.org/wiki/Fault-tolerant_system.
[8] Hwang Soonwook, Kesselman Carl. A flexible framework for fault tolerance in the grid. J Grid Comput 2003;1(3):251–72. doi:10.1023/B:GRID.0000035187.54694.75.
[9] Abawajy Jemal H. Fault tolerant scheduling policy for grid computing systems. In: 18th International parallel and distributed processing symposium (IPDPS’04), Santa Fe, New Mexico. Los Alamitos, CA, USA: IEEE Computer Society (CS) Press; 2004. p. 238–44.
[10] Yu Jia, Buyya Rajkumar. A taxonomy of workflow management systems for grid computing. J Grid Comput 2005;3(3–4):171–200. doi:10.1007/s10723-005-9010-8.
[11] Qureshi Kalim, Khan Faiz Gul, Manuel Paul, Nazir Babar. A hybrid fault tolerance technique in grid computing system. J Supercomput 2010.
[12] Chunlin Li, Layuan Li. A system-centric scheduling policy for optimizing objectives of application and resource in grid computing. J Comput Ind Eng 2009;57:1052–61.
[13] Buyya Rajkumar, Economic-based distributed resource management and scheduling for grid computing. PhD thesis, Monash University, Melbourne, Australia, April 12; 2002.
[14] Condor Manual: http://www.cs.wisc.edu/condor/manual/v7.0/-condor-V7_0_1-Manual.pdf [August 2008].
[15] Gartner Felix C. Fundamentals of fault-tolerant distributed computing in asynchronous environments. J ACM Comput Surv 1999;31(1):1–26.
[16] Gioiosa Roberto, Sancho Jose Carlo, Jiang Song, Petrini Fabrizio, Davis Kei. Transparent incremental check-pointing at kernel level: a foundation for fault tolerance for parallel computers. In: Proceedings of the 2005 ACM/IEEE SC|05 conference (SC’05); 2005.
[17] Jankowski Gracjan, Januszewski Radoslaw, Mikolajczak Rafal. Grid checkpointing architecture – a revised proposal, core GRID TR-0036, May 30; 2006.
[18] Hwang Soonwook, Kesselman Carl. Grid workflow: a flexible failure handling framework for the grid. In: 12th IEEE international symposium on high performance distributed computing (HPDC’03), Seattle, Washington, USA. Los Alamitos, CA, USA: IEEE CS Press; 2003.
[19] Vanderster Daniel Colin, Dimopoulos Nikitas J, Sobie Randall J. Intelligent selection of fault tolerance techniques on the grid. In: Third IEEE international conference on e-science and grid computing. Washington, DC, USA: IEEE Computer Society; 2007. ISBN: 0-7695-3064-8.
[20] Anglano Cosimo, Canonico Massimo. Fault-tolerant scheduling for bag-of-tasks grid applications, vol. 3470. Berlin/Heidelberg: Springer; 2005. p. 630–9, ISSN: 0302-9743 (Print), 1611-3349 (Online), ISBN: 978-3-540-26918-2.
[21] Transmission Delay: http://en.wikipedia.org/wiki/Store_and_forward_delay.
[22] Buyya Rajkumar, Murshed Manzur. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. J Concurrency Comput: Pract Exp 2002;13(13–15):1.
[23] WWG Testbed: http://gridbus.cs.mu.oz.au/sc2003/list.html.
[24] Mandal Nandita, Deelman Ewa, Mehta Gaurang. Integrating existing scientific workflow systems: the Keper/Pegasus. In: Proceedings of the 2nd workshop on Workflows in support of large-scale science. ACM; 2007. p. 21–8. ISBN: 978-1-59593-715-5.
[25] Yu Jia, Buyya Rajkumar. Gridbus workflow enactment engine. Available from: http://www.buyya.com/GridbusWorkflowEgine2008.pdf.

Fiaz Gul Khan is a Ph.D. Student in Wireless Systems and Related Technologies at University of Politecnico di Torino, Italy. His research interest includes Distributed Computing, Wireless Networks, Network architecture, Communication protocols, and Simulation tools. He received his Masters degree from COMSATS Institute of Information technology Abbottabad, Pakistan. Contact him at fiazgul.khan@studenti.polito.it.