Proc. of the Intl. Conf. on Advances In Computing, Communication and Information Technology – CCIT 2014.

ISBN: 978-1-63248-010-1 doi: 10.15224/ : 978-1-63248-010-1-106

Providing Fault Tolerance in Grid Computing Systems

Torki Altameem

***Abstract*—In grid computing, resources are used outside the boundary of organizations and it becomes increasingly difficult to guarantee that resources being used are not malicious. Also, resources may enter and leave the grid at any time. So, fault tolerance is a crucial issue in grid computing. Fault tolerance can enhance grid throughput, utilization, response time and more economic profits. All mechanisms proposed to deal with fault-tolerant issues in grids are classified into: job replication and job checkpointing techniques. These techniques are used according to the requirements of the computational grid and the type of environment, resources and virtual organizations it is supposed to work with. Each has its own advantages and disadvantages which forms the subject matter of this paper.**

***Index Terms*—Fault tolerance, Grid computing, Checkpointing, Job replication.**

I. Introduction

Grid computing technology is developed for solving large scale computational and data intensive problems in science, engineering, and commerce [1]. It is distinct from conventional distributed computing by autonomic heterogeneous resources and large scale resource sharing [2]. It uses resources of many separate computers connected by a network (usually internet). The tremendously large number and the heterogeneous potential of grid resources cause scheduling of user jobs to these resources is the key technology in grid computing [1], [3], [4].

Grid computing systems have been used for execution of applications that need more time. During execution, the computation cannot complete if any resource failure is encountered. The possibility of failures occurring is exacerbated by the fact that many grid resources will be used in performing long tasks that may need several days of computation. Also, since grid environments are extremely heterogeneous and dynamic, with components joining and leaving the system all the time, more faults are likely to occur in grid environments. Therefore, fault tolerance has become a crucial issue in grid computing systems.

The fault tolerance techniques compromise between efficiency and reliability of the resource in order to complete the execution even in the presence of failures. The main objective usually is to preserve efficiency hoping that failures will be less. However, the computational resources have increased in grid but its dynamic behavior makes the environment unpredictable and failure prone.

All techniques proposed to deal with fault-tolerant issues in grids are classified into two categories. The first one is called space redundancy or job replication. In this category, the same job is replicated to be executed on multiple undependable resources to guard the job against a single point of failure. The second category is called time redundancy or checkpointing and rollback. In this category, the state of a running job is saved to a stable storage. This state can be used later in case of any fault to resume execution of the job instead of restating it. An adaptive technique uses both job replication and checkpointing to achieve the fault-tolerant [5]. To overcome the drawbacks present with job replication and checkpointing, fault tolerance is factored into grid scheduling.

This paper presents the most commonly used fault tolerance techniques in grid computing systems. Also, it considers the most parameters used for evaluating the performance of grid computing systems.

This paper is organized as follows: section 2 briefly explains fault tolerance in grid computing. In Section 3, the standard metrics used to measure the performance of fault tolerance techniques are presented. Section 4 elaborates the techniques of the fault tolerance in grids. Section 5 discusses the process of selecting the fault tolerance technique. Section 6 concludes the paper.

II. Fault Tolerance in Grid Computing

Fault tolerance is preserving the delivery of expected services despite the presence of fault-caused errors within the system itself. Errors are detected and corrected and permanent faults are located and removed while the system continues to deliver acceptable services [6]. In computational grids, fault tolerance is important as the dependability of grid resources may not be guaranteed. It is needed to enable the grid to continue its work when one or more resources fail. In this sense, a fault-tolerant service must be included to detect errors and recover from them and thus avoiding the failure of the grid.

The heterogeneous nature of grid resources means that applications will be performing in environments where faults are more likely to occur between disparate resources. if there is no fault tolerance provided, the grid cannot survive to continue when one or more resources fail and the whole application crashes. Thus a fault tolerant technique is needed that would enable grid to continue executing even in the presence of faults.

The architecture of basic fault tolerance employed in grid computing systems is shown in Fig. 1. The system has five main components: Grid Interface, Allocator, Information Server, Fault Handler, and the Grid. Grid Interface provides an interface to users to submit their jobs for execution. Allocator selects the optimal resources to execute the job. The allocation decisions of the Allocator are based on the Quality of Service (QoS) requirements of users. The Information Server (IS) contains information about all resources in the grid. The information can include computation speed, memory available, load, and so on. The IS supplies the scheduler with the required information. The Fault Handler is responsible for detecting failure of

Proc. of the Intl. Conf. on Advances In Computing, Communication and Information Technology – CCIT 2014.

ISBN: 978-1-63248-010-1 doi: 10.15224/ : 978-1-63248-010-1-106

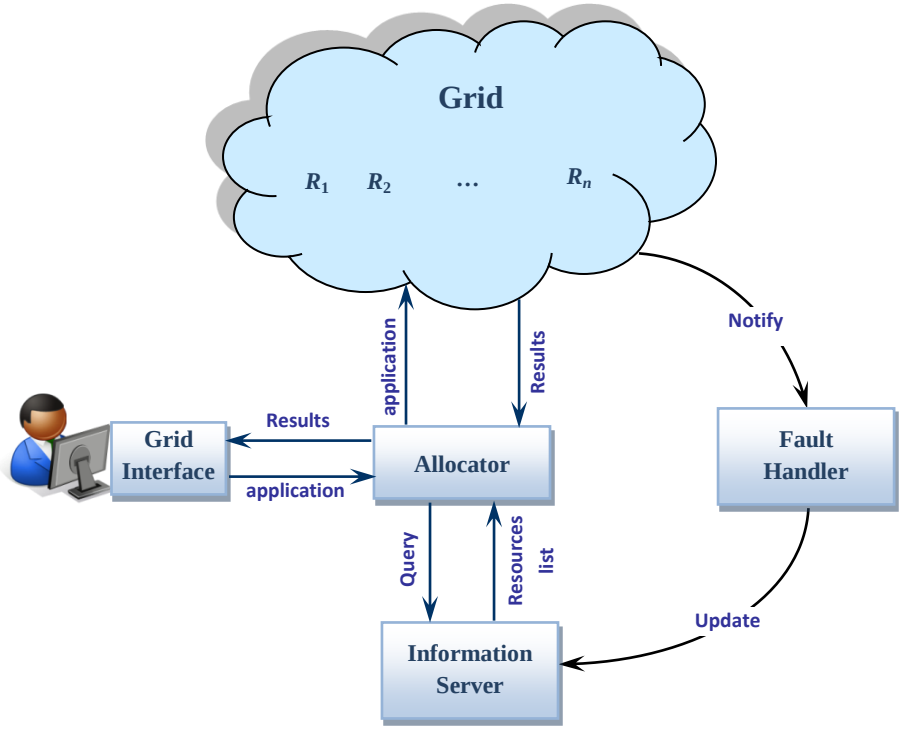

Fig. 1. The basic architecture of fault tolerance in grid computing.

resources and estimating the required information for fault tolerance process.

III. Performance Metrics

The performance of fault tolerance techniques is measured by standards metrics turnaround time, throughput, fail tendency and grid load. The parameters are defined as:

Turnaround time: It is an important parameter for determining the performance of different fault tolerance techniques. It is the only parameter users pay attention for. It can be defined as the interval of time elapsed from submission of a job to the time of its completion.

Throughput: Historically, throughput has been a measure of the comparative effectiveness of high performance computers that run many programs concurrently. An early throughput measure was the number of batch jobs completed in a certain period of time. More recent measures assume a more complicated mixture of work or focus on some particular aspect of computer operation.

Throughput is used to measure the ability of the grid to accommodate jobs. It is defined as:

$$Throughput(n) = \frac{n}{T_n},$$

where n is the total number of jobs submitted and T_n is the total amount of time necessary to complete n jobs.

Fail tendency: It is the percentage of the tendency of grid resources to fail and is defined as:

$$FailTendency = \frac{\sum_{j=1}^{m} p_{fj}}{m} \times 100\%,$$

where m is the total number of grid resources and p_{fj} is the failure rate of resource j. Through this metric, we can expect the faulty behavior of the system.

Grid load: It represents the amount of extra computations encountered by the grid to alleviate the effect of resources failures.

IV. Fault Tolerance Techniques

Providing fault tolerance in a grid environment, while optimizing resource utilization and response time, is a challenging task. A large number of research efforts have already been devoted to fault tolerance in the area of distributed computing. However, a little work has been done for fault tolerance in grid environments. Aspects that have been explored include the design and implementation of fault detection services, as well as the development of failure prediction, and recovery strategies. The recovery strategies are classified into job replication (space-sharing) and checkpointing (Time-sharing).

A. Job replication

Job replication is a key mechanism for developing fault-tolerant and highly available grids. It is commonly used by fault tolerance mechanisms in order to enhance the availability of the grid. Replication is based on the assumption that the probability of a single resource failure is much higher than of a simultaneous failure of multiple resources. It avoids job recomputation by starting several copies of the same job on diff
redundant copies of a job, the grid ca

Proc. of the Intl. Conf. on Advances In Computing, Communication and Information Technology – CCIT 2014.

ISBN: 978-1-63248-010-1 doi: 10.15224/ : 978-1-63248-010-1-106

service in spite of failure of some grid resources carrying out job copies without affecting the performance of the grid.

J. Abawajy [7] presented a distributed fault-tolerant scheduling (DFTS) algorithm that couples job scheduling with job replication. He assumed that grid is divided into sites and each site has a scheduling manager for resources in this site. Each scheduling manager acts as a backup for another scheduling manager. His algorithm is static because it depends on using a fixed number of replicas for each job. Each job replica is scheduled to a different site to be executed. The number of replicas is specified by the user at the time of job submission.

K. Srinivasa, G. Siddesh and S. Cherian [8] proposed an adaptive replication middleware which depends on data replication at different sites of the grid. The middleware dispatches replicas to different nodes and enables data synchronization between multiple heterogeneous nodes in the grid. Data sources are synchronized by using TCP/IP transfer protocol.

M. Chetepen et al [9] provided some scheduling heuristics based on job replication and rescheduling of failed jobs. Their heuristics do not depend on particular grid architecture and they are suitable for scheduling any application with independent jobs. Scheduling decisions are based on dynamic information on the grid status and not on the information about the scheduled jobs.

In [10], C. Jiang and et al proposed a replication based fault tolerant algorithm which schedules jobs by matching the user security demand and the resource trust level. The number of job replications changes adaptively according to the security level of the grid environment.

M. Amoon [5] considers adaptive job replication technique in order to create a proactive fault-tolerant scheduling system. In his system, two algorithms are proposed. One algorithm is for determining the number of replicas for each job, namely and the other algorithm is used for selecting the resources that execute these replicas. Both of algorithms depend on using the fault rate of the resources. The number of replicas is dynamic and is determined according to the fault rate of resources scheduled for jobs.

The main disadvantage of job replication technique is the additional resources used in executing the same job. This can cause grid over provisioning and can lead to great delays for other jobs waiting these resources to become free. Also, most of the existing replication based techniques are static. This means that the number of replicas of the original job is decided before execution and it is fixed number. Static job replication leads to excessive utilization of resources and also to excess load on the grid.

On the other hand, adaptive job replication can alleviate this extra load resulting from using fixed number of replica. Adaptive job replication techniques determine the number of replica according to the failure history of the primary resource allocated to execute the job. Thus, the number of replica will be different for each job according to the failure behavior of each resource in the past. Bad failure history means big number of replica and good failure history means small number of replica.

Fig. 2 shows the comparison between using a static job replication technique and using and adaptive one. It is shown that adaptive replication techniques provide less grid load than the static ones. So, using adaptive replication techniques is better than using static replication techniques.

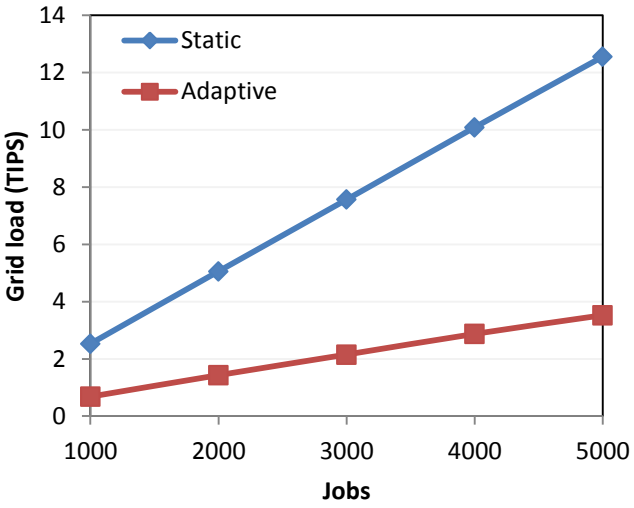

Fig. 2. Comparison between static and dynamic job replication.

B. *Job checkpointing*

Checkpointing is the ability to save the state of a running job to a stable storage. In case of any fault, this saved state can be used to resume execution of the application from the point in computation where the check-point was last registered instead of restarting the application from its very beginning. This can reduce the execution time to a large extent. Thus, the purpose of checkpointing is to increase fault-tolerance and to speedup application execution on unreliable systems.

Many real time applications in distributed system [11], [12], [13 have used checkpointing for performance optimization. F. G. Khan, K. Qureshi and B. Nazir [14] presented a performance evaluation of most commonly used fault-tolerant techniques (FTTs) in grid computing. These FTTs include retrying, checkpointing, alternate resource and alternate task. The metrics used in their evaluation are throughput, turnaround time, waiting time and network delay.

B. Nazir, K. Qureshi and F. G. Khan [15] presented an adaptive fault tolerant job scheduling strategy for grids. Their strategy is checkpointing-based. It maintains the fault index of grid resources. The scheduler makes scheduling decisions according to the value of the fault index of resources and response time of resources.

In [16], M. Nandagopal and V. R. Uthariaraj combined the mechanism developed in [15] with Minimum Total Time to Release (MTTR) job scheduling algorithm. Also, when making scheduling decisions, their scheduler depends on using the fault index and the response time of resources.

J. Mehta and S. Chaudhary [17] assumed that short running jobs can be resubmitted from scratch if they failed and presented a fault tolerant scheme that should applied to long running jobs using checkpointing.

In [18], P. Domingues, J. Silva and L. Silva presented a study about the effects of sharing checkpoints on turnaround time in desktop grid systems. In [19], M. Chtepen et al provided an algorithm called MeanFailureCP. This algorithm is designed to modify a job
as a function of mean failure freque

Proc. of the Intl. Conf. on Advances In Computing, Communication and Information Technology – CCIT 2014.

ISBN: 978-1-63248-010-1 doi: 10.15224/ : 978-1-63248-010-1-106

the job is being executed, and the total job execution time. In [20], they developed the MeanFailureCP+ algorithm which is a modification of the MeanFailureCP that deals with checkpointing of grid applications with execution times that are unknown a priori.

The main disadvantage of checkpointing mechanism is that it performs identically regardless the stability of the resource. This inappropriate checkpointing can delay the job execution and can increase the grid load. Furthermore, each job maintains multiple checkpoints and has to periodically invoke a garbage collection algorithm to reclaim the checkpoints that are no longer useful.

The efficiency of a checkpointing technique strongly depends on a good choice of a checkpointing interval. The checkpointing interval is the duration between two checkpoints. Short checkpointing interval leads to a large number of redundant checkpoints, which delay job processing by consuming computational and network resources. On the other hand, when a checkpointing interval is too long, a substantial amount of work has to be redone in case of a resource failure. Each interval starts when a checkpoint is established and ends when next checkpoint is established. There are several benefits of using checkpointing, including: fault recovery, better response time, and better system utilization.

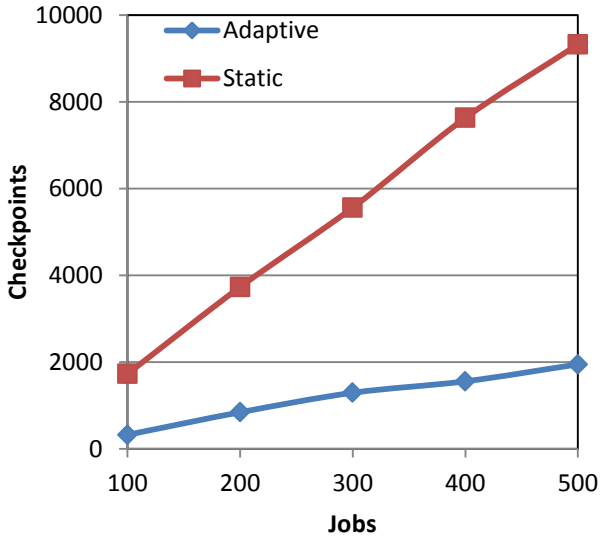

Fig. 3. Comparison between static and dynamic job checkpointing.

Most of job checkpointing techniques use a static or fixed number of checkpoints which leads to excessive utilization of resources and also to excess load on the grid. Adaptive job checkpointing techniques can alleviate this extra load resulting from using fixed number of checkpoints. These techniques determine the number of checkpoints according to the failure rate of the primary resource allocated to execute the job. Thus the number of checkpoints will be different for each job. Fig. 3 shows the comparison between using a static checkpointing technique and using and an adaptive one. It is shown that adaptive checkpointing techniques provide less grid load than the static ones. So, using adaptive checkpointing techniques is better than using static checkpointing techniques.

V. Fault Tolerance Technique Selection

Fault tolerance is widely adopted to increase the overall system performance and reliability of grids. Since grid computing systems usually include a large number of distributed resources, selecting the most suitable fault tolerance technique reduces the overhead of system developers and helps to achieve optimal scheduling of resources.

Different fault tolerance techniques have different features. For example, the response time of a checkpointing technique is not good compared with the job replication technique. This is due to the extra time needed to migrate the job to another resource when a resource fails. On the other hand, job replication technique does not need to migrate jobs between resources and the first returned response is employed. The required networking and computing resources of job replication techniques are much higher than those of checkpointing techniques. Checkpointing has another cost when writing checkpoint data to stable storage whenever a checkpoint is taken. This cost is proportional to the size of the checkpoint data.

Thus, we can use checkpointing strategy for the resources constrained grids and job replication technique for real time applications. However, determination of the number of replica and the number and intervals of checkpoints are still big challenges.

VI. Conclusions

Fault tolerance plays an important role in order to achieve good performance of a grid system. The most famous standards metrics used to evaluate the performance of fault tolerance techniques are turnaround time, throughput, fail tendency and grid load. Replication and Check pointing are the major techniques used in any fault-tolerant grid management system. In this paper, some works that have been done using the two techniques are surveyed. It is shown that adaptive fault tolerant technique provides better performance than static one.

References

[1] P. Huang, H. Peng, P. Lin and X. Li, “Static Strategy and Dynamic Adjustment: An Effective Method for Grid Task Scheduling,” J. Future Generation Computer Systems 25, 884–892 (2009).

[2] L. Lu and S. Yang, “DIRSS_G: An Intelligent Resource Scheduling System for Grid Environment Based on Dynamic Pricing,” Int. J. Information Technology 12 (4), 120–127 (2006).

[3] L. Chunlin, Z. J. Xiu, and L. Layuan, “Resource Scheduling with Conflicting Objectives in Grid Environments: Model and Evaluation,” J. Network and Computer Applications 32, 760–769 (2009).

[4] T. Altameem and M. Amoon, “An Agent-Based approach for dynamic adjustment of scheduled Jobs in Computational Grids”, Journal of Computer and Systems Sciences International, vol. 49, no. 5, pp. 765–772, Oct. 2010.

[5] M. Amoon, “Design of a Fault-Tolerant Scheduling System for Grid Computing,” Proc. of International Conference on Networking and Distributed Computing (ICNDC2011), 21-24 Sep., 2011, Beijing, P. R. China, pp. 104-108.

[6] M. Amoon, “A job checkpointing system for computational grids,” Central European Journal of Computer Science, vol. 3, no. 1, pp. 17-26, March 2013.

[7] J. Abawajy, "Fault-tolerant scheduling policy for grid computing systems," Proc. of 18th IEEE International Parallel and Distributed Processing Symposium, April 26-30, 20

[8] K. Srinivasa, G. Siddesh and S. Cherian, "
grid computing," Proc. of 12th IEEE

ISBN: 978-1-63248-010-1 doi: 10.15224/ : 978-1-63248-010-1-106

High Performance Computing and Communications, Melbourne, Australia, pp. 635-640, Sep. 1-3, 2010.

[9] M. Chtepen, B. Dhoedt, F. Cleays and P. Vanrolleghem, "Evaluation of replication and rescheduling heuristics for gird systems with varying resource availability," Proc. of 18th International Conference on Parallel and Distributed Computing Systems, Anaheim, CA, USA, pp. 622-627, Nov. 13-15, 2006.

[10] B. Khoo and B. Veeravalli, "Pro-active failure handling mechanisms for scheduling in grid computing environments," J. Parallel and Distributed Computing, vol. 70, no. 3, pp. 189-200, 2010

[11] Buyya R (2002) Economic-based distributed resource management and scheduling for grid computing. Ph.D. Paper, Monash University, Melbourne, Australia, 12 April 2002

[12] Foster I, Kesselman C, Tueke S (2001) The anatomy of the grid: enabling scalable virtual organizations. Int J Supercomput Appl.

[13] Nazir B, Khan T (2006) Fault tolerant job scheduling in computational grid. In: Proceedings of 2nd IEEE international conference on emerging technologies (ICET'06), Peshawar, Pakistan, 13–14 November 2006, pp 708–713

[14] F. G. Khan, K. Qureshi and B. Nazir, "Performance Evolution of Fault Tolerance techniques in Grid Computing System," Journal of Computing and Electrical Engineering, vol. 36, pp. 1110-1122, 2010.

[15] B. Nazir, K. Qureshi and F. G. Khan, "Adaptive checkpointing strategy to tolerate faults in economy based grid," Journal of Supercomputing, vol. 50, pp. 1-18, 2009.

[16] M. Nandagopal and V. R. Uthariaraj, "Fault Tolerant Scheduling Strategy for Computational Grid Environment," International Journal of Engineering Science and Technology, vol. 2, no.9, pp. 4361-4372, 2010.

[17] J. Mehta and S. Chaudhary, "Checkpointing and recovery mechanism in grid," Proc. of Sixteenth Intl. Conf. on Advanced Computing and Communication (ADCOM 2008), Chennai, 14-17 Dec. 2008, pp. 131-140.

[18] P. Domingues, J. Silva and L. Silva, "Sharing Checkpoints to Improve Turnaround Time in Desktop Grid Computing," Proc. of the 20th Intl. Conf. on Advanced Information Networking and Applications (AINA'06), Vienna, Austria, 18-20 April 2006.

[19] M. Chtepen et al, "Adaptive Task Checkpointing and Replication: Toward Efficient Fault-Tolerant Grids," IEEE Trans. Parallel and Distributed Systems, vol. 20, no. 2, pp. 180-190, Feb. 2009.

[20] M. Chtepen, F. Claeys, B. Dhoedt, F. Turck, P. Demeester, and P. Vanrolleghem, "Adaptive checkpointing in dynamic grids for uncertain job durations," Proc. of the 31st Intl. Conf. on Information Technology Interfaces (ITI), Dubrovnik, Croatia, 22-25 June 2009.