International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 9 (2015) pp. 24463-24478

http://www.ripublication.com

A Comparative Study on Fault Tolerance Strategies For Job Scheduling In Grid Environment

B.Muthulakshmi[1], K.Somasundaram[2]
[1]*Assistant professor (SL.Grade), Department of Computer Science and Engineering, A.V.C. College of Engineering,Mayiladuthurai, Tamilnadu, India*
[2]*Professor, Department of Computer Science and Engineering, Jaya Engineering College – Avadi, Tamilnadu,India*

Abstract

Grid computing is a form of distributed computing, which involves coordinating and collaborative sharing of resources across geographically dispersed organizations.There are no best scheduling approaches for all grid environments since the scheduling onto the grid is NP-complete.Due to the task characteristics, machines, and network connectivity, an appropriate scheduling algorithm is chosen as an alternative to utilize in a given grid environment. Job scheduling is one of the key research areas in grid computing to accomplish high throughput of the system. The possibility of a failure is much greater in large-scale grids than in traditional parallel systems. Henceforth, fault tolerance has become a vital field in grid computing. It becomes progressivelyhardto assure that a resource being used is not malicious as it may also be used outside of organizational boundaries. This paper presents an extensive survey of various fault tolerance strategies for job scheduling, which can contribute in developing efficient scheduling algorithms.Moreover, the comparison between the numerous job scheduling and fault tolerance techniques are illustrated.

Index Terms: Grid Computing, Job Scheduling, Fault Tolerance, Checkpoint Recovery, Failure Detection Service.

Introduction

GRID computing is a new trend in the distributed computing systems, which enables the aggregation and sharing of resources for solving large-scale applications. Grid scheduling is the process of establishing scheduling decisions, including resources over several administrative domains to use a single machine. In an efficient way, the grid allows the management of heterogeneous and dynamically available resources. Job scheduling is the most critical problems in grid computing for running

applications. A software application, say, scheduler is used for scheduling and it enables an enterprise to schedule. In current years, the researchers have introduced various efficient scheduling algorithms. These algorithms are used in grid computing to allocate the resources of the grid with a special emphasis on job scheduling. The key parameters such as deadline, waiting time, process time, and turn around time is included once a task is considered. The classification of scheduling algorithms are:

- Centralized Algorithm
- Decentralized Algorithm

The global scheduling decisions are made by a central scheduler in a case of centralized algorithm. It has the single access point for the whole infrastructure. In decentralized algorithm, the decision is made by all the scheduling instances, which accepts tasks for execution.

More faults are likely to occur in grid environment since grid resources are highly heterogeneous and dynamic. Fault tolerance is basically important to accomplish the better performance of the computational grids. The failure of the resources affects job execution and the possibility of a failure is much greater than in traditional parallel computing. Therefore, it investigates the techniques of fault tolerance for grid computing. Fault tolerance [1]can be considered as the survival attribute of computer system in which the function of fault tolerance is described. Even in the presence of faults, it is the ability of a system to perform its function accurately. Fault tolerance makes the system more dependable and preserves the delivery of expected services. The service of fault tolerance is essential for satisfying the requirements of Quality of Service (QoS) in grid computing. Without any participation of any external agents, a fault tolerant service candetectand recover the error. The strategies to recover from error comprise (a) roll-back and (b) roll-forward. Different types of failures [2]with regards to the computing resource are given in the Table I.

Table 1: Failure Types

FAILURE	**DESCRIPTION**
Omission Failure	The reply to a request is omitted by a server
Response Failure	Incorrect reply is forward with a request by a server
Value failure	The wrong value is returned by a server
State transmission Failure	The incorrect state transition is made by the server
Timing Failure	A server does not reply in the specified real-time interval
Crash Failure	Until being restarted, a server frequentlyfails to reply to requests.

The fault tolerance service deals with several kinds of resource failures, which consists of:

- Process Failure
- Processor Failure
- Network Failure

This may lead to the failure of job/resource, Service Level Agreement (SLA), disrupting timing deadlines, and destroyed the user expected QoS. Fault tolerance makes to acquire the dependability of the system that related to the QoS aspects offered by the system. Dependability involves the attributes of reliability and availability, where the reliability specifies that a system can run constantly without any failure and the availability indicates that a system is instantaneously ready to employ. In order to properly schedule both the faulty and non-faulty tasks, fault-tolerant schedulers attempt to do so by incorporating the scheduling and fault management.

Many existing systems focused on the concept of the performance improvement to achieve low cost using the grid job replication. In order to improve the reliability and speed up of the system in the grid, the grid environment is created initially and further the jobs are collected from the user. To avoid the permanent failure of the job, the proposed concept is focused on the replica of the each job before scheduling. It will improve the performance of the grid environment. Based on the replica of each resource, the monitoring process will relocate the job.

The rest of the paper is organized as follows. Section II describes the various fault tolerant strategies for job scheduling in grid environment. Section III presents results and discussion about the several fault tolerant strategies. Section IV provides the proposed work and section V discusses conclusions.

Various Fault Tolerance Strategies For Job Scheduling In Grid Environment

An effective scheduling and computation has become one of the major challenge in the grid computing. Job scheduling is the job mapping to a particular physical resource, which reduce some of the cost function specified by the user. This states an NP-complete problem and an optimal solution is reached using different heuristics.

A. Highest Response Next (HRN) Scheduling

This scheduling approach [3]offers more response with time, memory, and the requirements of the CPU. Based on the priority of jobs and the capability of the processors, thejobs are allocated to the number of processors. Without any loss of performance, this approach is adaptive for local and remote jobs. It is also highly adaptive for grid environment.

1) Merits

- HRN scheduling approach completes all the jobs rapidly than the First Come First Serve (FCFS).
- It effectively utilizes the available resources with priority.
- It overcomes some of the weakness of both Shortest Job First (SJF) and FCFS.

2) Demerits

- Since determining job priority is a tedious one, it does not suit for large number of job allocations.
- Turn-around time is high.
- Wastage of CPU and memory.

B. Optimal Resource Constraint (ORC) Scheduling

According to the capability of the processors, ORC scheduling algorithm [4] allots the jobs. The best fit algorithm is applied initially and then it is followed by the Round Robin (RR) scheduling. It distributes the jobs among the available processors and is compared with various algorithms such as FCFS, SJF, and RR. In terms of turn-around time and average waiting time, the ORC provides better performance than the other algorithms. The efficiency of the load balancing and grid resource dynamicity capability are enhanced.

1) Merits

- As ORC is suitable for a large number of jobs, it overwhelms the difficulty of HRN and FCFS scheduling.
- Decrease the process allocation complexity.
- Minimize the turn-around time and waiting time of the jobs in the queue.
- ORC avoids the problem of starvation.

2) Demerits

- ORC has high communication overhead.

C. Hierarchical Job Scheduling (HJS)

For a cluster of workstations, the scheduling model is based on a hierarchical method[5]using two level scheduling. It comprises local level and top level, which is the global scheduling. For distinct type of the jobs, the global scheduler utilizes a single or separate queue for scheduling with the policy of FCFS, SJF, or FF. The global scheduler has various functions to match the resources that a request by a job and to obtain the best utilization of the available clusters. Whereas, the local scheduler is responsible for scheduling the jobs to a particular resource.

1) Merits

- Decrease the overall turn-around time.
- For high system loads,
  - ✓ It increases the system utilization.
  - ✓ It maintains the delay of job scheduling at global level using multiple queue.

2) Demerits

- Under-utilization of grid resources.
- HJS approach does not consider the dynamic behavior of the grid resources.

- HJS approach has the problem of starvation.

D. Resource Co-allocation for Scheduling Tasks with Dependencies (RCST)

The co-allocation scheduling method [6] offers a strategy to schedule the jobs along with the dependencies in grid environment. This method is applied on both inside and across the clusters. Based on the dependencies, each and every step merges or combines the clusters. The major objective of this algorithm is to enhance the efficiency with respect to the load balancing and minimum time for task execution.

1) Merits

- Decrease task execution time.
- At the earliest time, the tasks are allocated and scheduled.
- Due to the decentralized strategy, this method is more reliable than a centralized one.
- It is less subjective to the single point of failure.
- Better load balancing.

2) Demerits

- High communication overhead.
- A task requirement has not specified by the RCST.

E. Grouping based Job Scheduling

By taking into consideration of the resource computational and communication capabilities, this scheduling algorithm [7] schedules the jobs in grid systems. The resource priority is determined using the network bottleneck bandwidth of the resource. The scheduler retrieves the information about the resource processing capability by the job grouping approach. The total number of jobs group should be established in order to avoid the higher processing time. Therefore, the processing loads among the chosen resources are balanced. A hybrid of SJF, FCFS, and RRscheduling algorithm provides better performance than the SJF alone in terms of processing time.

1) Merits

- Increase the resource utilization.
- Decrease the processing time of the jobs.

2) Demerits

- The resource cannot manage the factors such as current load of resource, network delay, and the requirements of QoS.

F. Grouping-Based Adaptive Fine-grained Job Scheduling

The grouping strategy [8],[9] in the job scheduling model is based on the characteristics of the resources. An adaptive fine-grained job scheduling is focused on the light-weight job scheduling in grid environment. The grouping algorithm

incorporates the greedy and FCFS technique, to enhance the fine-grained jobs processing.

1) Merits

- By grouping the light weight jobs, the total overhead of fine-grained job scheduling can be minimized.
- Maximizes the resource utilization.
- Minimize the job execution time.
- Low network latency.
- Low total processing time.

2) Demerits

- High pre-processing time for job grouping.
- The memory size constraint does not consider in this scheduling algorithm.

G. Job Schedule Model

The job schedule model is based on the Maximum Processor Utilization and Throughput (MPUT) scheduling algorithm to accomplish the performance utilization. There are two major modules in this model, Supervisor Schedule Module (SSM) and Execute Schedule Module (ESM). The functions of the supervisor grid node[10]are: processing new jobs, processing transferred jobs, receiving completion of transferring jobs, grouping job information and grid information, executing job scheduling, and updating the information to the log file.

1) Merits

- At the condition of failure of the supervisor node, it uses the backup node.
- Therefore, it provides reliability with good load balance.
- Increases the CPU utilization.
- Maximizes the throughput.
- Decreases the turn-around time.

2) Demerits

- More communication overhead.
- This scheduling model does not consider any job and resource constraints.

H. Replication Fault Tolerance

A fault can occur while a grid resource is unable to complete its job in the given deadline and due to the failure of resource, job or network. The term check-pointing is the process of saving the running application state to the constant storage. If there is any fault occur, this saved state is used to resume execution of the application instead of restarting the application from the beginning.The check-pointing can be classified based on the attributes such as the requirement of storage space, level of abstraction, orphan messages, in-transit messages, the granularity of check-pointing, and scope of the check-point.

To accomplish the fault tolerance in grids, replication is the significant method. Several replication mechanisms are:

a. Job replication: The service of fault tolerant service is capable of receiving and executing the jobs, performing the operations of checksum, and transfer the results back [11]. The co-ordination service initially locates, receives, and votes upon the jobs that submitted by a user.
b. Component replication: If any machine fails or components are replicated on various machines in the grid, then that respective application could be relocated and run on another machine [12].
c. Data replication: To improve the availability in the grid, say, environments where the failure occurs, the replication is used by fault tolerance mechanisms. While a node hosting a data replica crash, other replicas are made available by other nodes [13]. It may further consist of two categories: (a) synchronous replication, and (b) asynchronous replication.

1) Merits

- Protect against malicious services.
- Maximize the functionality in the fault tolerance grid services.

2) Demerits

- High overhead.
- Voting could not commence until an appropriate number of outcomes have been produced.

I. Checkpointing Fault Tolerance

Based on the attribute of the level of the abstraction, the state of a process is being saved. The categories of level of abstraction are:

- System Level Check-Point (SLC)
- ApplicationLevel Check-Point (ALC)
- Mixed Level Check-Point (MLC)

At the level of operating system or middleware level, SLC[14] is a technique that offers the automatic, transparent check-pointing of applications. The mechanism of check-pointing has no knowledge about any of its characteristics as the application seemed as a black-box. In SLC, check-pointingthe bits that constitute the program counter contents, registers, and memory are stored on the constant storage. The Condor and Libckpt are the examples of systems that do SLC.

An application can acquired fault tolerance by offering their own code of check-pointing. It accurately restarts from several positions in the code via saving the specific information to a restart file. BONIC and XtremWeb[15] are the examples of ALC. The major distinguishing between SLC and ALC are listed as follows:

- Transparency
- Portability
- Size of check-point
- Flexibility

- Forced check-point generation

ALC cannot be created forcefully, since the process state can only be saved once the generated code of checkpoint is reached during execution[16]. Whereas, SLC can be saved at any moment andMLC is the integration of SLC and ALC.

1) Merits

- Provide transparency, flexibility, and portability.

2) Demerits

- Complex to build the MLC systems.

J. Scheduling Fault Tolerance

Fault tolerance is factored into grid scheduling to overwhelm the demerits present with the mechanisms of check-pointing and the replication. The policies of scheduling for grid system can be categorized into time-sharing [17]and space-sharing policy. The processors have shared over time in time-sharing scheme by implementing various applications on the same processors through distinct time interval. Whereas, the processor is split into disjoint sets in the space-sharing approach.

Many fault tolerance scheduling has been studied for load balancing, failure masking, enhanced scalability, and so on. Fault tolerance scheduling mechanisms such as Charlotte, Bayanih an, Javelin, GUCHA and Xtrem Web may tolerate crash and link failures. But, they have limitations like redundant re-computation and an independent live-lock problem [18]. To overcome these complications, Distributed Fault Tolerant Scheduling (DFTS) [19]and Volunteer Availability based Fault Tolerant Scheduling (VAFTS) [18] are described. The major components of the DFTS policy are: job placement algorithm and replica management. The algorithms are described as follows:

Job Placement Algorithm[19]

1. Poll all sites for availability information
   a. **IF** ($R \geq n$) **THEN**
      i. Choose the best n sites
      ii. Designate one of them as a backup home SM andnotify the n sites the backup
   b. **ELSE**
      i. Reserve n - R site that are expected to finish soon.

   ENDIF
2. **IF** there are at least n sites **THEN**
   a. Send a replica of the job to each site
   b. Update job table and backup scheduler.

   ENDIF

1) Merits

- Load balancing.
- Failure masking.

- Improved scalability.
- Volunteer autonomy failures.

2) Demerits

- Redundant re-computation
- An independent live-lock problem.

K. Heartbeat Mechanisms for Failure Detection

Failure detectors are described as an integral part of any fault tolerant distributed systems, where the traditional failure detectors do not execute efficiently while applied to the grid environments. Most of the real-life distributed systems implement the fault tolerant service through the heartbeat mechanisms[20]. Various heartbeat mechanisms are centralized, virtual ring based, all-to-all, heartbeat groups. The centralized approach has only one centralized monitor[21].This receives heartbeat signals of all the nodes at regular time interval. It is considered as failed, if the leader within the group has not received a fixed number of heartbeat signals from a specific node. The models of centralized approach are:

- Push model
- Pull model
- Dual model

A monitorable object transmits a heartbeat message to the monitor at regular interval in push model. It is said to be suspected once a message has not received within its expected time bounds. In the pull model, the monitor transmits a liveliness request to the monitorable object and it responds to the request.

1) Merits

- Simplicity.
- Scalability.
- Adaptability.

2) Demerits

- Single point of failure, i.e., if the monitor fails then the whole approach fails.

Results and Discussion

Various techniques for job scheduling and fault tolerance on grid environment are depicted. The results of the survey are shown in Table II. The Global Scheduler Fault Tolerance (GSFT) improves the reliability as well as the speed up of the system in the grid environment. From the survey, it is evident that the GSFT can result better fault rate for each process than the other fault tolerance job scheduling algorithm like heartbeat mechanism.

Table 2: Information About Various Fault Tolerance Strategies For Job Scheduling In Grid Environment

Techniques	Author & Reference	Year	Performance	Frame work	Quality measurement
Job Scheduling Approaches in Grid Environment					
Highest Response Next (HRN) Scheduling	Somasundaram et al [3]	2007	HRN is a novel scheduling algorithm in grid to offer more responses with the requirements of time, memory and CPU.	Nil	1. CPU utilization 2. Memory utilization 3. Waiting time of each job 4. Service time of each job
Optimal Resource Constraint (ORC) Scheduling	Somasundaram et al [4]	2008	ORC scheduling reduces the waiting time of the jobs in the queue and increases the processing time of the job.	Nil	1. Average waiting time of process 2. Jobs allocated to the processor 3. Turn-around time for running processes
Hierarchical Job Scheduling (HJS)	Santoso et al [5]	2000	HJS approach uses both the global and local scheduling. The policies of FCFS, SJF, and FF are applied at both the levels.	Nil	1. Load Balancing
Resource Co-allocation for Scheduling Tasks with Dependencies (RCST)	Moise et al [6]	2011	Co-allocation denotes the strategy to provide a task scheduling with dependencies in grid environment.	MonALISA and ApMon tool.	1. Optimized resource utilization 2. Load balancing 3. Task scheduling 4. Minimum execution time
Group based Job Scheduling Methods in Grid Computing					
Grouping based Job Scheduling	Rosemary et al [7]	2012	An algorithm is developed for an efficient job scheduling along with FCFS, SJF, and RR for jobs.	MATLAB using the parallel computing toolbox.	1. Processing time 2. Maximize the resource utilization
Bandwidth-Aware Job Grouping based Scheduling (BJGS)	Ang et al [26]	2009	BJGS focuses on grouping independent jobs and schedules them in accordance with network conditions.	GridSim Toolkit	1. Processing time with the number of jobs 2. Processing time with the granularity size
Resource Scheduling Model with Bandwidth-Aware Job Grouping Strategy	Sharma et al [27]	2010	The resource scheduling algorithm with BJGS is used to minimize the processing time and increase the grid resource utilization.	GridSim Toolkit	1. Processing time of the algorithms 2. Processing cost of the algorithms
Grouping-Based Adaptive Fine-grained Job Scheduling	Liu et al [8]	2009	To utilize the grid resources sufficiently, grouping based fine-grained job scheduling	GridSim toolkit.	1. Reduce the execution time from 20 to 30s. 2. Decrease the total cost

			is introduced. The grouping algorithm can be incorporated with greedy and FCFS approach.		
	Chang et al [9]	2012	An adaptive scoring method is introduced for scheduling the jobs in grid environment.	Taiwan UniGrid Platform, Globus Toolkit 4.	1. CPU speed of each resource 2. Makespan 3. Decrease the completion time of jobs
Job Schedule Model	Wu et al [10]	2007	The performance of the job schedule model system can be measured from the log file editorial policy.	Job information history database and supervisor grid information database.	1. Maximum processor utilization and throughput 2. Minimize the turn-around time
Fault Tolerance by Replication in Grid Environment					
Replication Fault Tolerance	Townend et al [11]	2004	Replication based fault tolerance uses an FT-Grid co-ordination service to locate, receive, and vote upon the jobs submitted by a client program.	DSS-Net and web hosting environment	1. Minimize the likelihood of faulty results. 2. Protect against the malicious services.
	Arshad et al [12]	2006	An automated failure method based on continuous monitoring and AI planning is presented for failure recovery.	Apache, Tomcat.	1. Recovery from machine failure. 2. Recovery from the failure of component.
	Antoniu et al [13]	2004	A software architecture is presented to decouple the consistency management from the fault-tolerance management.	JUXMEM platform.	1. Allocation cost based on the replication degree. 2. Cost of the basic primitives.
Fault Tolerance by Check-pointing in Grid Environment					
Checkpointing Fault Tolerance	Amoon [22]	2012	A checkpointing based scheduling system is presented for computational grids using the average failure time and failure rate of the grid resources.	Grid simulator	1. Throughput 2. Average turn-around time 3. Failure tendency
	Nandagopal et al [23]	2010	Fault tolerance based job scheduling with checkpoint replication	Globus Toolkit	1. Number of jobs completed for various deadlines.

			service can optimize the user-centric metrics in the presence of faults.		2. Execution time for number of jobs.
	Gokuldev et al [24]	2013	A fault index based re-scheduling algorithm is introduced for re-scheduling the jobs to other available resources.	GridSim Toolkit.	1. Dynamic adaptive checkpoint. 2. Kernel level checkpoint.
	Chtepen et al [25]	2009	Based on the information of the grid status, numerous heuristics are presented to offer high job throughput in the presence of the failure.	DSiDE grid simulator.	1. Average job execution time. 2. Average job length.
Fault Tolerance by Scheduling in Grid Environment					
Scheduling Fault Tolerance	Gannon et al [17]	2004	The common component architecture is introduced to describe the support for user-defined checkpointing and restart for distributed applications.	XCAT3 Framework.	1. Checkpoint time with data size.
	Choi et al [18]	2004	Volunteer autonomy failures are specified to solve an independent livelock problem	Nil	1. Total execution time according to the number of tasks.
	Abawajy et al [19]	2004	Fault tolerance job scheduling approach is presented to incorporate the scheduling and replication of the jobs.	Nil	1. Sensitivity to mean time between failures.
Fault Tolerance by Heartbeat Mechanisms in Grid Environment					
Heartbeat Mechanisms for Failure Detection	Jain et al [21]	2004	A highly scalable failure detector is presented along with a membership management algorithm.	GridSim	1. Processor utilization.
	Monnet et al [28]	2007	A hierarchical failure detection service is evaluated in a test environment.	Grid 5000 platform	1. The faults are detected efficiently.

Proposed Work: Global Scheduler Fault Tolerance (GSFT) Algorithm

Grid computing is an efficient technology to enhance the resource execution, which may follow the scheduling approaches. In general, the scheduling algorithms will enhance the resource utilization in the grid. A fault tolerance is one of the most

important factor of grid computing. The fault is measured level by level in the proposed system and the three major levels of monitoring the fault in the grid job execution are:

- Before Execution
- Before Scheduling
  - ✓ Scheduling Phase
  - ✓ Execution Phase
- After Execution

These three categories will improve the reliability as well as the fault recovery for the jobs in the resource. For scheduling the jobs, Peeking Order Scheduling System will introduced based on the fault tolerance. Level by level grid environment will connected with the global grid resource scheduler and local grid resource scheduler. Further, the global grid resource scheduler connects to the virtual network, which has multiple resources. Grid Information Server (GIS) will used to store the data of each virtual network status. Based on the GIS, the Global Scheduler Fault Tolerance (GSFT) algorithm will presented to schedule the jobs.

The QoS parameters for the virtual network are gathered and are considered by the GIS. The next level of scheduling will performed in the local grid resource scheduler. This scheduling will consider the QoS parameters for the individual resource that are connected to the virtual network. After performing this operation, the jobs will submitted to grid resources. Subsequently, it will analyzed to monitor the fault occurrence in the grid environment at the time of job execution.

3) Merits

- Achieve less fault rate for each process.
- Enhance the reliability.
- Improve the speed up of the system in the grid.

Conclusion

In this paper, an overview of various fault tolerance based job scheduling algorithms are presented. The survey is based on the techniques of the job scheduling, group based job scheduling, fault tolerance by replication, fault tolerance by checkpointing, fault tolerance by scheduling, and fault tolerance by heartbeat mechanisms. From the survey, it is finding out that the Global Scheduler Fault Tolerance (GSFT) algorithm is an efficient failure reduction service to minimize the job execution time. It results more improved reliability and the speed up of the system in the grid environment. GSFT also provides the less fault rate for each process than the other mentioned fault tolerance mechanisms.

References

[1] P. Townend and J. Xu, "Fault tolerance within a grid environment," *time-out,* vol. 1,no. S2, p. S3, 2003.

[2] N. Arshad, *et al.*, "A planning based approach to failure recovery in distributed systems," in *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*, 2004, pp. 8-12.

[3] K. Somasundaram, *et al.*, "Efficient Utilization of Computing Resources using Highest Response Next Scheduling in Grid," *Asian Journal of Information Technology*, vol. 6,no. 5, pp. 544-547, 2007.

[4] K. Somasundaram and S. Radhakrishnan, "Node allocation in grid computing using optimal resource constraint (ORC) scheduling," *IJCSNS International Journal of Computer Science and Network Security*, vol. 8,no. 6, pp. 309-313, 2008.

[5] J. Santoso, *et al.*, "Hierarchical job scheduling for clusters of workstations," in *Proceedings of the sixth annual Conference of the Advanced School for Computing and Imaging, Delft, Netherlands*, 2000, pp. 99-105.

[6] D. Moise, *et al.*, "Resource coallocation for scheduling tasks with dependencies, in grid," *arXiv preprint arXiv:1106.5309*, 2011.

[7] P. Rosemarry, *et al.*, "Grouping Based Job Scheduling Algorithm Using Priority Queue And Hybrid Algorithm in Grid Computing," *International Journal of Grid Computing & Applications (IJGCA) Vol*, vol. 3,2012.

[8] Q. Liu and Y. Liao, "Grouping-based fine-grained job scheduling in grid computing," in *Education Technology and Computer Science, 2009. ETCS'09. First International Workshop on*, 2009, pp. 556-559.

[9] R.-S. Chang, *et al.*, "An adaptive scoring job scheduling algorithm for grid computing," *Information Sciences*, vol. 207,pp. 79-89, 2012.

[10] H. Wu, *et al.*, "A Job Schedule Model Based on Grid Environment," in *Complex, Intelligent and Software Intensive Systems, 2007. CISIS 2007. First International Conference on*, 2007, pp. 43-52.

[11] P. Townend and J. Xu, "Replication-based fault tolerance in a grid environment," in *UK e-Science 3rd All-Hands Meeting*, 2004.

[12] N. Arshad, "A planning-based approach to failure recovery in distributed systems," University of Colorado, 2006.

[13] G. Antoniu, *et al.*, "Building fault-tolerant consistency protocols for an adaptive grid data-sharing service," 2004.

[14] G. Bronevetsky, *et al.*, "Recent advances in checkpoint/recovery systems," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006, p. 8 pp.

[15] P. Domingues, *et al.*, "Using checkpointing to enhance turnaround time on institutional desktop grids," in *e-Science and Grid Computing, 2006. e-Science'06. Second IEEE International Conference on*, 2006, pp. 73-73.

[16] R. Y. De Camargo, *et al.*, "Checkpointing-based rollback recovery for parallel applications on the integrade grid middleware," in *Proceedings of the 2nd workshop on Middleware for grid computing*, 2004, pp. 35-40.

[17] S. Krishnan and D. Gannon, "Checkpoint and restart for distributed components in XCAT3," in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, 2004, pp. 281-288.

[18] S. Choi, *et al.*, "Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment," in *Network Computing and Applications, 2004.(NCA 2004). Proceedings. Third IEEE International Symposium on*, 2004, pp. 366-371.

[19] J. H. Abawajy, "Fault-tolerant scheduling policy for grid computing systems," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 2004, p. 238.

[20] A. S. M. Noor and M. M. Deris, "Extended Heartbeat Mechanism for Fault Detection Service Methodology," in *Grid and Distributed Computing*, ed: Springer, 2009, pp. 88-95.

[21] A. Jain and R. Shyamasundar, "Failure detection and membership management in grid environments," in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, 2004, pp. 44-52.

[22] M. Amoon, "A Fault Tolerant Scheduling System Based on Check pointing for Computational Grids," International Journal of Advanced Science and Technology, vol. 48, 2012.

[23] 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

[24] V. M. Gokuldev S, "Fault Tolerant System for Computational and Service Grid," International Journal of Engineering and Innovative Technology (IJEIT), vol. 2, no. 10, 2013.

[25] M. Chtepen, et al., "Adaptive task checkpointing and replication: Toward efficient fault-tolerant grids," Parallel and Distributed Systems, IEEE Transactions on, vol. 20, no. 2, pp. 180-190, 2009.

[26] T. Ang, et al., "A bandwidth-aware job grouping-based scheduling on grid environment," Information Technology Journal, vol. 8, no. 3, pp. 372-377, 2009.

[27] R. Sharma, et al., "A New Resource Scheduling Model with Bandwidth Aware Job Grouping Strategy in Grid Computing," in Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, 2010, pp. 324-328.

[28] S. Monnet and M. Bertier, "Using failure injection mechanisms to experiment and evaluate a grid failure detector," in High Performance Computing for Computational Science-VECPAR 2006, ed: Springer, 2007, pp. 610-621.