Fault Tolerance Mechanism for Computational Grid Using Checkpoint Algorithm

Mr.Ramesh Prajapati
Dept. of Computer Science Engineering
Jagannath University, Jaipur, India
rtprajapati@gmail.com

***Abstract*-- Computational grids have solving large-scale scientific applications using heterogeneous and geographically distributed resources Grid infrastructure is a large set of nodes geographically distributed and connected by a communication. In Computational grid, fault tolerance is one of the main research areas. Fault tolerance is a necessary by the distribution that create a number of problems related to the heterogeneity of hardware, operating systems, networks, middleware, applications, the dynamic resource, the scalability. In this research paper our main focus is on the development of fault tolerance system for computational grids. we have studied existing fault tolerance in Computational Grid in detail, and have ascertained the frequent causes of failures in it. So Checkpoint is process as a designated place in a program at which normal processing is interrupted specifically to preserve the status information necessary to allow resumption of processing at a later time. Checkpointing is the process of saving the status information. The probability of fault occurrence increases, as the number of resources involved in grid increases. For this we had setup a computational grid based on the Alchemi middleware. Alchemi is a .NET based grid computing framework that provides the runtime machinery and programming environment required to construct computational grid. After setting up grid environment we had generate the different checkpoint result and compare with chandy-Lamport result.**

***Keywords*-- Grid computing, fault tolerance, check pointing**

I. INTRODUCTION

In Large-scale science and engineering are done through the interaction of people, heterogeneous computing resources, information systems, and instruments, all of which are geographically and organizationally dispersed. The overall motivation for "Grids" is to facilitate the routine interactions of these resources in order to support large-scale science and Engineering. There are many Opportunity of grid in eScience and eBusiness suppose for Physicists worldwide pool resources for peta-op analyses of peta bytes of data, for Civil engineers collaborate to design, execute, & analyze shake table experiments, for An insurance company mines data from partner hospitals for fraud detection, for An application service provider offloads excess load to a compute cycle provider, and for An enterprise configures internal & external resources to support eBusiness workload.

There are many Challenges technical requirement for the grid. [2]

- Dynamic formation and management of virtual organizations
- Online negotiation of access to services: who, what, why, when, how
- Establishment of applications and systems able to deliver multiple qualities of service
- Autonomic management of infrastructure elements

So that in the grid Computing, main element for concept on Resource sharing, Coordinated problem solving, and Dynamic, multi-institutional virtual organizations. Grid resources are also heterogeneous in nature so resources may enter and leave the grid at any time, in many cases outside of the applications' control. So therefore interaction faults may be likely to occur between disparate grid nodes. Also resource may be outside of the organization so that is not guaranteed that a resource being used is not malicious. So here challenging issue is that faults and failure in grid. So ***fault tolerance*** is a crucial aspect for grid computing.

Section II describes the related work. In section III, we talk about Types of fault tolerance. Checkpoint recovery of for grid is described in section IV. The Implementation scheme is presented in Section V.In section VI described Performance Evaluation. In section VII, we present the Conclusion. Future Enhancements describe in section VIII.

II. RELATED WORK

Fault tolerance[6] is the system to perform its function correctly even in the presence of faults. The fault tolerance makes the system more dependable. A complementary but separate approach to increase dependability is fault prevention. This consists of techniques, such as inspection, whose intent is to eliminate the circumstances by which faults arise. A failure occurs when an actual running system deviates from this specified behavior. The cause of a failure is called an error. An error represents an invalid system state that does not complete the system specification. The error itself is the result of a defect in the system or fault. In other words, a fault is the root cause of a failure. However, a fault may not necessarily

result in an error; nevertheless, the same fault may result in multiple errors. Similarly, a single error may lead to multiple failures.

Fault tolerance[33] is an important property in grid computing, since the resources are geographically distributed. Check pointing method is used to improve system performance in the presence of failure, their effectiveness largely depends on tuning runtime parameters such as the checkpointing interval there are still many issues that need to be explored. In current system chandy-lamport algorithm[3] there are no. of checkpoint are taken during computation, due to that we required more space and bandwidth to store that checkpoints. The current algorithm is weak in terms of task completion time in both of fault-free and faulty situations. By using checkpoint algorithm, every node saves its context in stable storage. So that system has context saving overhead. Basic algorithm does not reduce that overhead. here in all checkpoint algorithm it could not generate the information about which node has crash and how to recover the failure node. Then after I had compared my proposed algorithm with chandy and Lamport algorithm.

III. TYPES OF FAULT TOLERANCE

A. *Types of Fault Tolerance*

There are many types of fault [4] that we faced in grid computing related to hardware (like faults of CPU, Disk storage), application and operating system (Memory link and resource unavailable), and network (like node failure and packet losses) see the figure 1. While considering all these factors, we have to consider the desirable properties of failure detectors and correctors that are Completeness, Accuracy, Consistency, Detection Latency, Scalability, Flexibility, adaptiveness.

The main objective of fault tolerance [3] is to preserve 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 service.

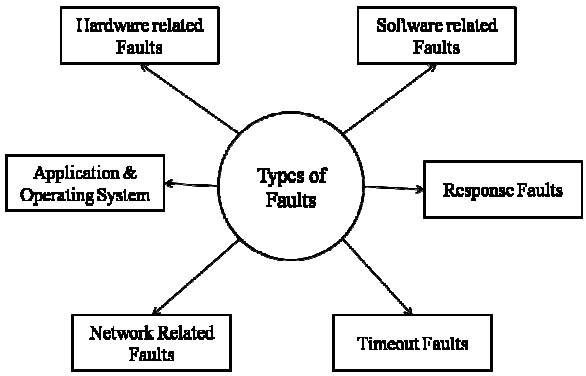

Figure 1 Type of Faults

TABLE I DETAILED TYPES OF FAULTS

Sr.No	Main Type	Detailed Types of faults
1	Hardware Relate Faults	CPU Memory Storage
2	Application & Operating System Oriented Faults	Operating System Specific • Memory Leaks • OS Faults Application Specific
3	Network Related Faults	Node fault Network oriented • Packet corrupted • Packet Losses
4	Software Related Faults	Unhandled Exception Unexpected Input
5	Timeout Faults	Timeout Run Exception Timeout Job Exception
6	Response Relate Faults	Value Byzantine

B. *Problem Definition*

From a user's point of view, a distributed application should continue despite failures. To achieve the automatic ways to deal with failures, various fault tolerance mechanisms are there. Some of these fault tolerance mechanisms are (See Figure 2)

1. **Application Dependent**: fault-tolerance[15] experts write algorithms and encapsulate them into reusable code artifacts, or modules.
2. **Monitoring Systems**: In this a fault monitoring unit is attached with the grid. The base technique which most of the monitoring units follow is heartbeating technique.
3. **Checkpointing recovery**: Checkpointing[1] involves saving enough state information of an executing program on a stable storage so that, if required, the program can be re-executed starting from the state recorded in the checkpoints.
4. **Fault Tolerant Scheduling**: fault tolerant scheduling [16]algorithms are using the coupling of scheduling policies with the job replication schemes such that jobs are efficiently and reliably executed. Scheduling policies are further classified on basis of time sharing and space sharing.

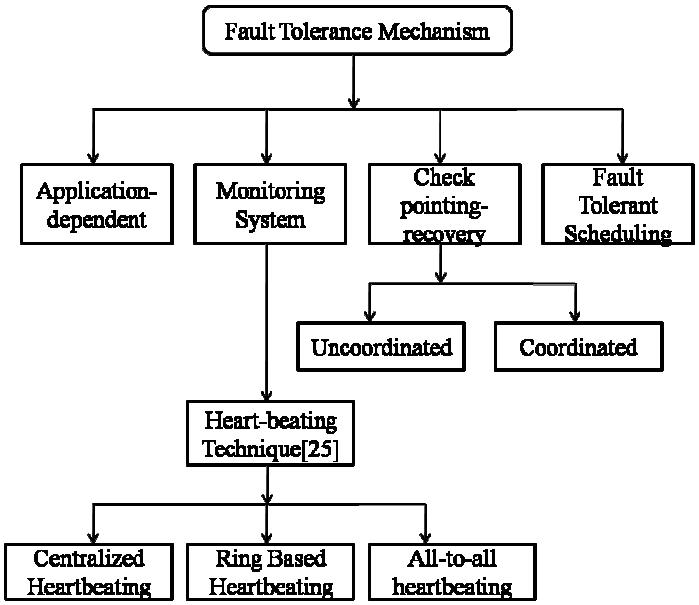

Figure 2 Fault Tolerance Mechanisms

IV. CHECKPOINT RECOVERY

Check pointing, we can restore the partially completed job from the last checkpoint saved and then starting a job from scratch is avoided. The main disadvantage of checkpointing mechanism [4] is that it performs identically regardless the stability of the resource. The check pointing is one of the most popular techniques to provide fault-tolerance on unreliable systems. It is a record of the snapshot of the entire system state in order to restart the application after the occurrence of some failure. The checkpoint can be stored on temporary as well as stable storage. However, the efficiency of the mechanism is strongly dependent on the length of the check pointing interval.

V. IMPLEMENTED SCHEME

Check pointing is one of the techniques of fault tolerance. Till now many middleware in the grid environment are not fully faults tolerant. Different middleware have different levels of fault tolerance. Some of the middleware like Alchemi.NET do not have a robust fault tolerance mechanism. Therefore, in this research work Alchemi.NET has been chosen and a checkpointing algorithm has been designed for it. Alchemi.NET [32] is an open source software framework that allows you to painlessly aggregate the computing power of networked machines into a virtual supercomputer and develop applications to run on the grid. Alchemi.NET includes:

- The runtime machinery (Windows executables) to construct grids.
- A .NET API and tools to develop .NET grid applications and grid enabled legacy applications.

A. Checkpoint Algorithm with certain time period

Algorithm: Checkpoint Algorithm with certain time period
Input: Enter maximum limit for prime number (1000000)
Host
Port Number
Host Name
Password
Time interval to create consecutive checkpoint
Output: Generate Checkpoint, Executor Log file

Algorithm Step:

S1: Initialization Set Checkpoint time Interval initialize no of Processes.

S2: Send message to manager for alive connection.

Check Failure condition If failure occur then go to step 4. Otherwise continue to step 3

S3: while process Pi till the end. Initialize checkpoint no Initialize new checkpoint flag value as true. Initialize CT
If Process Current time= Checkpoint Intervals set flag 1 for taking checkpoint Take stable checkpoint with current process state increase checkpoint number Update checkpoint interval time set flag false after successful checkpoint Else Take stable checkpoint with current process state. Increase checkpoint number Update checkpoint interval time set flag false after successful checkpoint.

S4: After checkpoint flag is checked, process Pi send and receive message and Check manager checkpoint number greater than Current checkpoint number. If Manager checkpoint no = Current checkpoint no then flag=0 Set Current checkpoint no = Manager checkpoint no Else Take new checkpoint with current process state Set Current checkpoint no = Manager checkpoint no Set flag = 1

VI: PERFORMANCE EVALUATION

A. Results Taken From Alchemi Tool

This study was done on the bases on 15 executors running with one manager to execute prime number generation application. Each node comprises of one CPU core with 512MB ram and storage spaces of 40 GB.I have worked three parameters.

Total Execution Time: The experiment was done on the bases two policies. First policies comparison of Different Number of Executer with Different input range with respect to time.

TABLE 2 COMPARISON OF DIFFERENT NUMBER OF EXECUTER WITH DIFFERENT INPUT RANGE WITH RESPECT TO TIME

Input Range to find out prime number	Total No of Executer Running	Total Time to Complete execution (In Second) (CPG-CTP Algorithm)	Total Time to Complete execution (In Minute) (Chandy-Lamport Algorithm)
Prime of 10000	2	2.0781	02:03:0423
	3	1.0884	01:09.0937
	4	1.0381	01:01.0265
Prime of 50000	1	3.0224	03:02:1423
	2	2.0432	02:03:0124
	3	1.0625	01:03:1295
	4	1.0112	01:02:0298
Prime of 100000	2	2.0825	02:04:0825
	3	1.0893	01:02:0654
	4	1.0318	01:01:0345
	5	1.0121	01:00:0124
	10	1.0081	01:00:0081

Table 1 shows result summary of prime number generation with different executers, different input range given by user and fix time interval of 300 ms. so we can compare result with Chandy-Lamport algorithm with different input range given and total time to complete the application. And second policies comparison of comparison of Different Number of Executer with Different input range with respect to Generation of checkpoint.

TABLE 3 COMPARISON OF DIFFERENT NUMBER OF EXECUTER WITH DIFFERENT INPUT RANGE WITH RESPECT TO GENERATION OF CHECKPOINT

Input Range to find out prime number	Total No of Executer Running	Total Checkpoint Generated CPG-CTP Algorithm	Total Checkpoint Generated Chandy-Lamport Algorithm
Prime of 10000	2	5	15
	3	3	11
	4	4	9
Prime of 50000	1	6	18
	2	9	23
	3	8	21
	4	13	26
Prime of 100000	2	15	29
	3	13	24
	4	17	32
	5	19	34
	10	21	39

Table 2 shows result summary of prime number generation with different executers, different input range given by user and fix time interval of 300 ms . so we can compare result of number of checkpoint generated with Chandy-Lamport algorithm and Implemented algorithm.

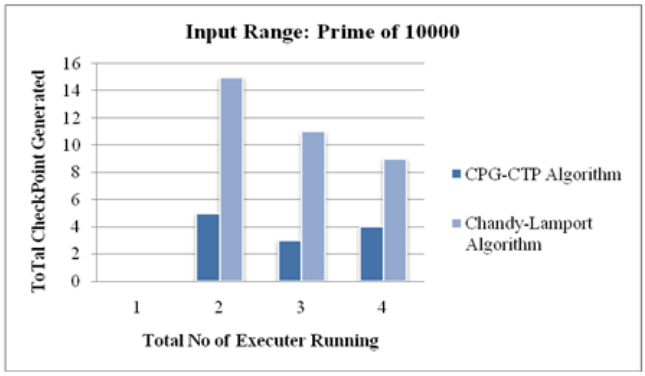

Figure 3 Comparison of checkpoint generation (Input Range: 10000)

Figure 3 shows result summary of prime number generation with different executers, different input range given by user and fix time interval of 300 ms .so we can compare result with Chandy-Lamport algorithm with different input range given and total time to complete the application Implemented algorithm total time taken by application is small compare to Chandy-Lamport algorithms in different input ranges.

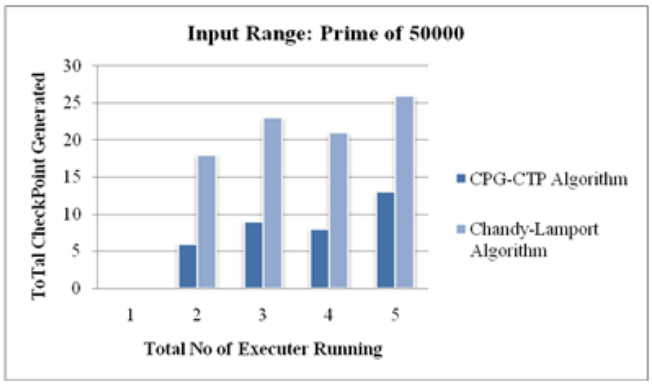

Figure 4 Comparison of checkpoint generation (Input Range: 50000)

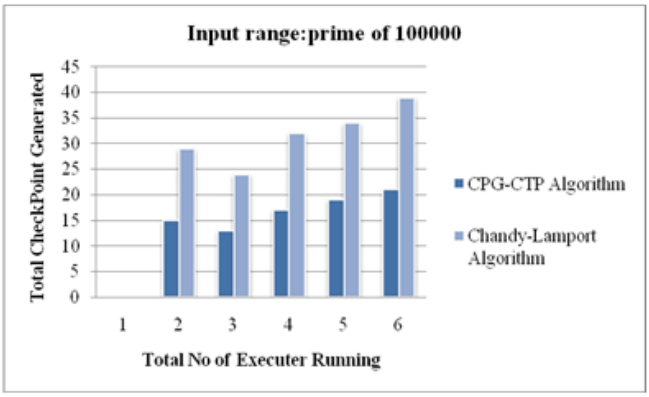

Figure 5 Comparison of checkpoint generation (Input Range: 100000)

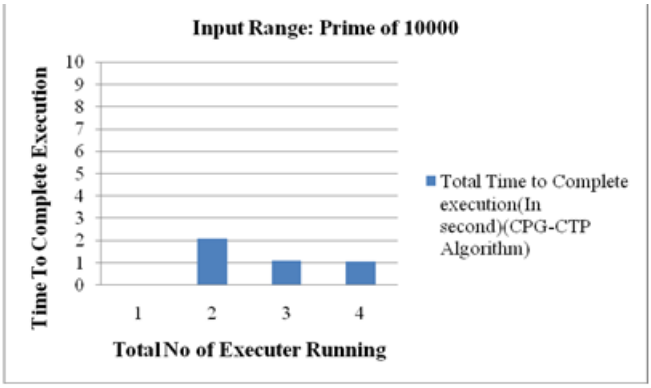

Figure 6 Execution Time (CPG-CTPAlgorithm with Prime of 10000)

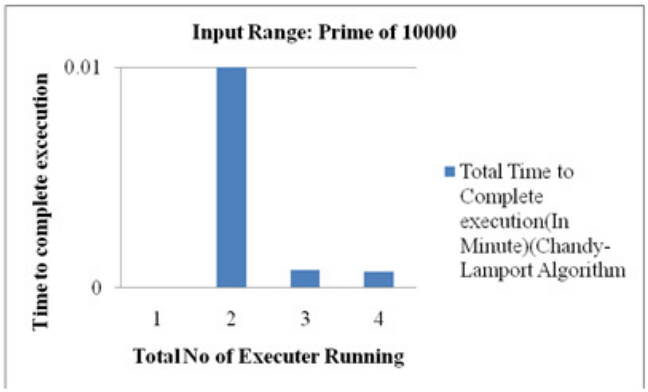

Figure 7 Execution Time (Chandy-Lamport Algorithm with Prime of 10000)

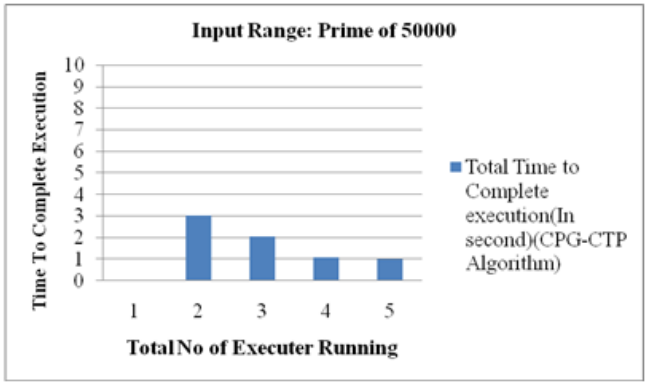

Figure 8 Execution Time (CPG-CTPAlgorithm with Prime of 50000)

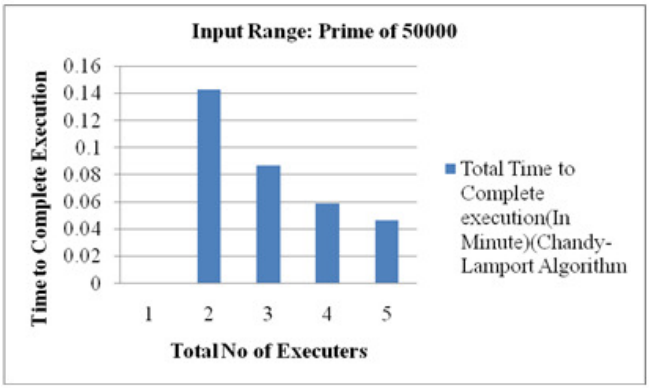

Figure 9 Execution Time (Chandy-Lamport Algorithm with Prime of 50000)

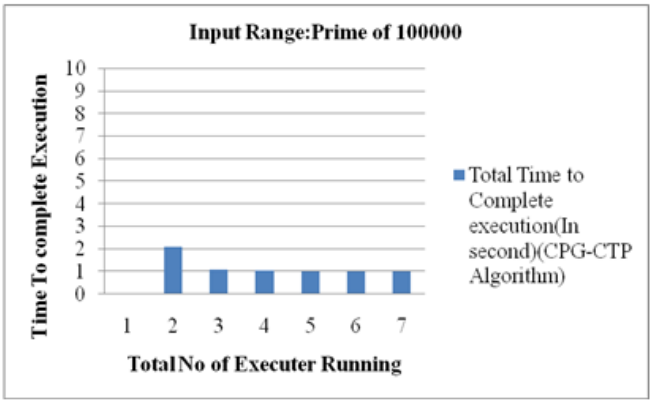

Figure 10 Execution Time (CPG-CTPAlgorithm with Prime of 100000)

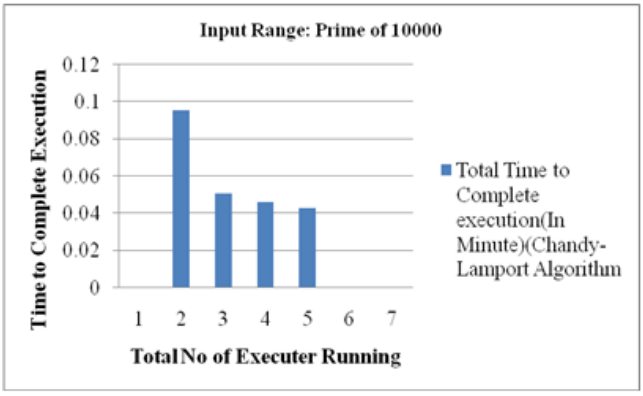

Figure 11 Execution Time (Chandy-Lamport Algorithm with Prime of 100000)

In this entire figure we had compared in our algorithm with number of executor increases it takes less execution time to complete application and in chandy-Lamport algorithm it takes more execution time to complete application.

VII. CONCLUSION

In this paper the sharing of computational resources is a main motivation for constructing Computational Grids in which multiple computers are connected by a communication network.Alchemi.NET for handling the faults dynamically has been identified. On the basis of identified shortcomings, proposed and updated fault tolerance in Alchemi.NET middleware with the help of check pointing algorithm..Implemented checkpoint concept that will help in avoiding the grid to become inaccessible if the Manager or Executor fails. This implementation done in Alchemi.Net Grid with fix time interval to take checkpoint of the running system. Number of checkpoint created is proportional to time taken by system to execute the job. In case executer fails and restarts it starts the job from the last checkpoint which increases the performance in both given approaches. Handles large amount of data with multiple executers which increases the overall performance i.e. taken less time by the system in this approach.

VIII .FUTURE ENHANCEMENTS

Checkpoint can be extended to work with load balancing technique. Backup manager concept that will help in avoiding the grid to become inaccessible if the central manager fails. This implementation is kind of third party software that will improve the grid fault tolerance. But this can be integrated into the Alchemi.NET middleware code to help the user to easily use it.

RERERENCES

[1] Sriram Krishnan, "An Architecture for Checkpointing and Migration of Distributed Components on the Grid" PhD Thesis, Department of Computer Science, Indiana University, November 2004

[2] Dheeraj Bhardwaj. Grid Computing Concepts, Applications, and Technologies Department of Computer Science and Engineering Indian Institute of Technology, Delhi

[3] Paul Townend and Jei Xu, "Fault Tolerance within a Grid Environment" Proceedings of AHM2003

[4] Foster, I., Kesselman, C., "The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann", pp. 15-51, 1999.

[5] Prakash R. and Singhal M.Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems. ,IEEE Transaction On Parallel and Distributed Systems, vol. 7,no. 10, pp. 1035- 1048, October1996.

[6] Chandy K. M. and Lamport L., "Distributed Snapshots: Determining Global State of Distributed Systems," ACM Transaction on Computing Systems, vol. 3, No. 1, pp. 63-75, February 1985.

[7] Koo. R. and S.Toueg. .Checkpointing and Rollback-Recovery for Distributed Systems. .IEEE Transactions on Software Engineering, SE- 13(1):23-31, January 1987.

[8] Silva L, Silva J 1992 Global checkpointing for distributed programs. Proc. IEEE 11th Symp. On Reliable Distributed Syst. pp 155-162.

[9] J.L. Kim and T. Park. "An efficient protocol for checkpointing recovery in Distributed Systems" IEEE Transaction On Parallel and Distributed Systems, 4(8):pp.955-960, Aug 1993.

[10] G. Cao and M. Singhal. "On impossibility of Min- Process and Non-Blocking Checkpointing and An Efficient Checkpointing algorithm for mobile computing Systems". OSU Technical Report #0SU-CISRC-9/97-TR44, 1997, pp 37-44.

[11] D.V. Subba Rao and MM Naidu: A new, efficient corrdinated checkpointing protocol combined with selective sender based message logging, IEEE, 2008, Page(s): 444 – 447.

[12] "An Adaptive Index-based Algorithm using Time-coordination in Mobile Computing", by Yanping Gao, Changhui Deng, Yandong Che in the Proceedings of the 2008 International Symposium on Information Processing (ISIP 08), May 2008, pp.578-585.

[13] Ajay D Kshemkalyani: "A symmetric O(n log n) message distributed snapshot algorithm for large scale systems", IEEE, 2010, pp 1-4

[14] Ajay D Kshemkalyani " Fast and message efficient global snapshot algorithms for large scale distributed systems" IEEE 2010. Page(s): 1281 – 1289.

[15] Anh Nguyen-Tuong, "Integrating Fault-Tolerance Techniques in Grid Applications" PhD Thesis, University of Virginia, August 2000

[16] J.H. Abawajy, "Fault-Tolerant Scheduling Policy for Grid Computing Systems", IPDPS'04

[17] www.gridsim.org

[18] gridsimulator.http://www.buyya.com/gridbus/gridsim/,relea sed on Apr 08, 2009

[19] http://www.gridcomputingplanet.com

[20] www.Alchemi.NET.net

[21] Krishna Nadiminti, Akshay Luther, Rajkumar Buyya, "Alchemi: A .NETbased Enterprise Grid System and Framework" December 2005

[22] www.wiki.gridpp.ac.uk/wiki/Grid middleware
[23] Amit Jain and R.K. Shyamasundar, "Failure Detection and Membership Management in Grid Environments" Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04) pp. 44-52
[24] Neeraj Kumar Rathore Ms. Inderveer Chana, "Comparative Analysis of Checkpointing" IMR Third National IT Conference-2008 on "Prestige Institute of Management and Research", Indore (M.P), India.
[25] Liang PENG, Lip Kian NG, "N1GE6 Checkpointing and Berkeley Lab Checkpoint/Restart" Dec 28, 2004
[26] Paul Townend and Jie Xu-"Fault Tolerance within a Grid Environment",University of Durham, DH1 3LE, United Kingdom.
[27] C.Kesselman. I. Foster. Computational grids. In The Grid: Blueprint for a New Computing Infrastructure., chapter 2. Morgan-kaufman edition, 1999.
[28] Yuuki Horita, Kenjiro Taura, Takashi Chikayama, "A Scalable and Efficient Self-Organizing Failure Detector for Grid Applications" 6th IEEE/ACM International Workshop on Grid Computing, Grid 2005
[29] en.wikipedia.org/wiki/Grid_computing
[30] Leu P-J, Bhargava B "Concurrent robust checkpointing and recovery in distributed systems". *Proc. Int. Conf. on Data Engineering* pp 154-163,1988
[31] Yanping Gao, Changhui Deng, Yandong Che: an adaptive index based algorithm using time coordination in mobile computing, International symposiums on information processing, IEEE, 2008, pp 578-585.
[32] Rajkumar Buyya and Srikumar venugopal , "A Gentle Introduction to Grid Computing and Technologies", CSI Communications, pages 9-19, Vol.29, No.1, ISSN 0970-647X, Computer Society of India (CSI), Mumbai, India, July 2005.
[33] J. Joseph, C. Fellenstein, "Grid Computing", Prentice Hall/IBM Press, Edition 2004
[34] Akshay Luther Rajlumar Buyya Rajiv Ranjan,"Peer-to-Peer Grid Computing and a .Net-based Alchemi Framework", the University of Melbourne, Australia