A Practical Approach of Storage Strategy for Grid Computing Environment

Kalim Qureshi

Abstract -- An efficient and reliable fault tolerance protocol plays an important role in making the system more stable. The most common technique used in High Performance Computing is rollback recovery, which relies on the availability of checkpoints and stability of storage media. Unstable media can result in failure of the nodes of the grid. Furthermore dedicating powerful resources solely as checkpoint storage results in loss of computation power of these resources which may become bottlenecks when the load on the network is high. A new protocol based on replication is proposed in this paper. To ensure the availability of valid checkpoints even in the case of checkpoint server or a whole cluster failure, the checkpoints are replicated on all checkpoint servers in the same cluster as well as on other clusters. To minimize the wastage of computational power of the most stable nodes in the cluster, our protocol utilizes the CPU cycles of dedicated servers in the case of high loads on the network.

Index Terms: **Checkpoint storage; fault tolerance in grid; checkpoint replication**

I. INTRODUCTION

The development of Wide Area Networks and the availability of powerful resources are changing the way servers interact. Technological advancements make it possible to utilize geographically distributed resources in multiple owner domains to solve large-scale problems in the fields of science, engineering and commerce. Therefore Grid and cluster computing have gained popularity in High Performance Computing. However, the technologies pose many challenges with respect to computational nodes, storage media and interconnection mechanism, which affect overall system reliability. As the size of Grid increases the Mean Time Between Failure (MTBF) decreases to a critical level [1]. Large applications (e.g. scientific and engineering) which require a computing power of hundreds or thousands of nodes create problems with respect to reliability which occur due to node or link failure giving rise to the need for dynamic configuration of the grid over runtime. Thus fault tolerance is an indispensable characteristic of Grid. In traditional implementation a failure causes the whole distributed application to shutdown requiring it to be restarted manually [2]. To avoid restarting the application from the beginning a technique called rollback recovery is used which is based on the checkpointing concept. In checkpoint-based strategies, the executed portion of the process is periodically saved to a stable storage medium as a checkpoint, which is not subject to failures. In the case of a failure, further computation is started from one of these previously saved states. Checkpointing can be classified into three categories [3]:

Kalim Qureshi is a faculty member of Information Science Department, Kuwait University, Kuwait. His research interests include network parallel distributed computing, thread programming, concurrent algorithms designing, task scheduling, and performance measurement. His email: kalimuddinqureshi@gmail.com.

- Uncoordinated checkpointing: Every process independently saves its checkpoints, thus, eliminating the need to communicate with other processes in order to initialize checkpoints but this technique suffers from domino effect
- Coordinated checkpointing: Processes have to coordinate with each other in order to form a consistent global state of checkpoints, so there is no chance of domino effect in this technique, and
- Communication-induced checkpointing: Processes initiate some of their checkpoints independently while a domino effect is prevented by the addition of checkpoints that are forced by protocol.

The fault-tolerance mechanism must have the ability to save generated checkpoints on a stable storage [4]. Usually this can be achieved by installing dedicated checkpoint servers, but these dedicated servers may become a bottleneck when the grid size increases. To overcome this problem, grid's shared nodes can be used to store checkpoint data. The following figure illustrates the concept of checkpoint storage over a stable storage media.

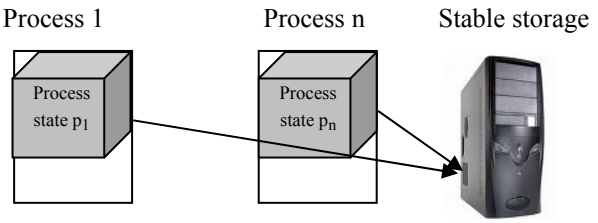

Figure 1. Fault tolerance by checkpoint/restart.

One way to provide checkpoint storage reliability is replication, in which multiple copies of a checkpoint are stored on different nodes so that data can be recovered even when part of the system is unavailable. Another approach used is to break data into fragments and add some redundancy so that data can be recovered from a subset of the fragments. The most common technique used to break data into redundant fragments is the addition of parity information. Parity calculation is less reliable as compared to replication, since failure of only one checkpoint can be tolerated, failure of two consecutive checkpoints results in loss of data, while replication can provide N-1 failure tolerance where N is the total number of nodes in the system. Checkpoints stored on storage disks are called disk-based checkpointing and checkpoints stored in memory are called diskless checkpointing.

Beside checkpoint storage, a fault tolerant protocol must also address the heterogeneity and dynamic nature of resources which is unavoidable in a Grid environment.

In this paper a new protocol named Reliable Checkpoint Storage Strategy (RCSS) is proposed which ensures

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

availability of valid checkpoints even in the case of checkpoint server or a whole cluster failure. It also shortens the checkpoint wave completion time by having the checkpoint server partially acknowledge the client when it records the chunk and then replicate it over other servers as well as on other clusters. Also our protocol utilizes the CPU cycles of dedicated servers in the case of high network load; hence, it minimizes the wastage of processing power of these most stable nodes of clusters.

II. RELATED WORK

In order to tolerate faults the strategies based on checkpointing save the executed portion of processes on stable storage so that the executed portion can be retrieved in case of failure and further computation can be carried out. Checkpoints can be initiated in three ways which have been discussed in the previous section. Chandy and Lamport [5] were the first to introduce a coordinated checkpointing protocol for distributed applications.

A. Disk-based Checkpointing

Disk-based checkpointing is advantageous in the sense that it can tolerate the failure of N-1 nodes, where N is the total number of nodes in the system. Bouabache et al [2] consider the scenario of a cluster of clusters; replication of checkpoints is done over stable storage and the number of these stable storage called checkpoint servers, are fixed in each cluster. Checkpoint servers are responsible for replicating the received chunk of the checkpoints on all other checkpoint servers. Two variants of replications are presented in [2]

- *Simple Hierarchical Replication Strategy (SHRS)*: The checkpoint server receiving the chunk of a checkpoint from a client becomes primary checkpoint server for that chunk and is responsible to replicate the checkpoint on all the checkpoint servers of its group and to checkpoint servers of other groups. So each primary server replicates the chunk to ($s+i^0$ mode $[2^m]$) nodes. But in this technique the intermediary servers have no role to play.

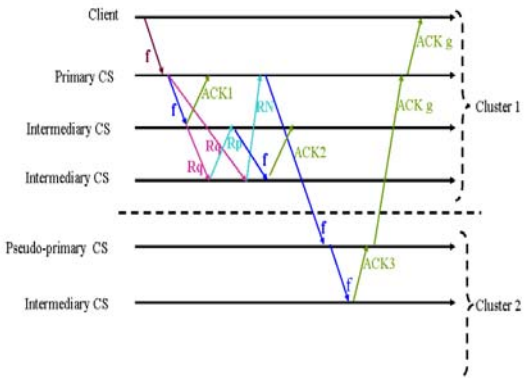

Figure 2. Checkpoint replication.

- *Greedy Hierarchical Replication Strategy (GHRS)*: Here the replication process is accelerated by the involvement of intermediary servers in the replication. For this a set of checkpoint servers is defined for each checkpoint server using a formula $\{s, s+2^0, s+2^1, \ldots, s+^{2m-1}\}$ called children of checkpoint server 's'. However this technique suffers from the overhead of time to store the checkpoint on stable storage [6].

- The Figure 2 shows the checkpointing replication transaction steps. Here f is the chunk to be replicated, CS is Checkpoint Server, Rq is Request to inquire whether the chunk has been received or not, Rp and Rn are the response of the request Rq. The client sends its checkpoint to the checkpoint server which becomes primary for that chunk. The primary checkpoint server then replicates the chunk to its children according to the formula given above as depicted in figure 2. Then each intermediary server sends request to its children to inquire about arrival of the chunk, if the children send negative reply then the checkpoint is forwarded to children.

B. Diskless Checkpointing

To reduce the overhead of disk-based checkpointing Plank et al. [7] introduced the concept of diskless checkpointing where the stable storage media is replaced by memory for checkpoint storage. Z. Chen at al [8] presented checksum-based checkpointing strategy which relies on diskless checkpointing. The technique is scalable as the overhead to survive K failures does not increase with increasing number of application processes. The key idea in this technique is pipelining of data segments. All the computational processors and the checkpoint processor are organized in chain, where each processor receives the data segment from its predecessor, calculates the checksum and sends it to the next processor in the chain and the process continues until the segment reaches the checkpoint server which is at the end of the chain. The checkpoint server will receive a segment at the end of each step and the checkpoint will be completed as soon as the last segment is received at the checkpoint server.

In Checkpoint Mirroring (MIR) [7], the checkpoint mirroring technique is used in which each processor saves a copy of its checkpoint to another processor's local disk. In the case of processor failure the copy of that checkpoint will be available for a spare processor to continue the execution of that process. The drawback of this technique is the need for space to store m+1 checkpoints per processor. The phenomenon is shown in Figure 3. N+1 Parity (PAR) [9] is motivated diskless checkpointing technique; it overcomes the space overhead of checkpoint mirroring with parity calculation that is stored on the central disk. The PAR checkpointing process is presented in Figure 4.

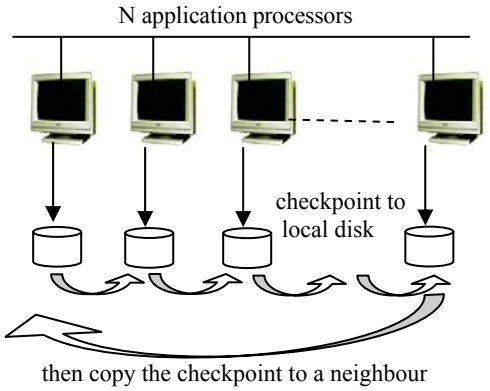

Figure 3. MIR checkpointing.

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

P. Sobe [9] presented two variants of parity calculation based on a Redundant Array of Independent Disks (RAID)-like storage scheme. In Parity Grouping over Local Checkpoints (PG-LCP) parity is calculated over local checkpoints and then stored on an additional node. If the size of the local checkpoint is different from the checkpoints received, then to calculate parity, the size of each checkpoint has to be enlarged to the size of the biggest checkpoint and these unused bits are assumed to be zero in calculation. To restart a single failed process, its last saved state has to be reconstructed which is done by XOR-ing all other checkpoints and parity. This process requires the transfer of N-1 checkpoints and parity. The unit of the parity scheme is the entire checkpoint here. The Phenomenon is elaborated in Figure 5.

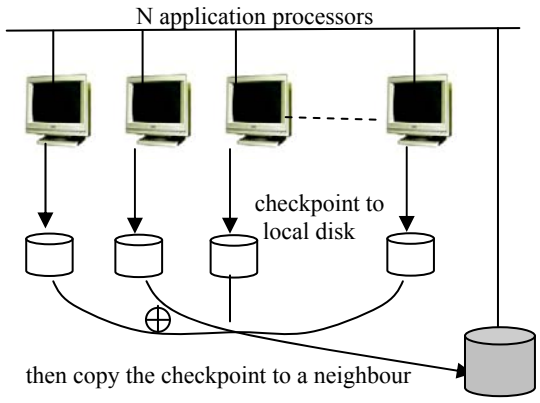

Figure 4. PAR chekpointing.

In Intra Checkpoint Distribution (ICPD), the checkpoint is divided into strips at local nodes and the parity is then calculated over these stripes (as shown in Figure 6). Then distributed software system transfer and write the parity and strips to disks. Here the unit of parity scheme is a strip. In addition to the parity information the length of the checkpoint is also stored at the parity node so that the checkpoint can be recovered in the original length even if the strip and information related to it is lost.

However diskless checkpointing incurs the overhead of high memory for storing checkpoints. While calculating the parity each processor has to communicate with the parity processor which may cause a communication bottleneck. Also to recover a failed processor, checkpoints from all other processors as well as from the parity node are required which is also an expensive task in terms of communication.

III. SYSTEM MODEL

As the Grid is the collection of powerful resources that are managed by different administrative domains, in this paper the Grid is considered as a cluster of clusters. The Grid architecture is defined by the following:

- There are “K” clusters in the Grid.
- Each cluster has “N” nodes.
- Among these “N” nodes some are dedicated checkpoint servers, and some have dual roles to play, i.e., in high load, these nodes will perform the processing and also store chunks, and in low load, they will only perform the processing, and the rest of the machines in the cluster will perform processing only.
- The clusters are connected through front-end machines.
- It is quite possible for any component to fail at any time in a Grid environment. A coordinated checkpoint protocol will handle the client failures.
- In this paper a strategy will be outlined to deal with the failure of checkpoint servers to guarantee the reliability of the storage service even when a checkpoint server fails. Failure can be any of the following types:
- A checkpoint server in a cluster may be disconnected due to link failure.
- A cluster may be disconnected from rest of the grid due to failure of the front-end machine of the cluster.
- Simultaneous failure of all the components of the cluster could occur**.**

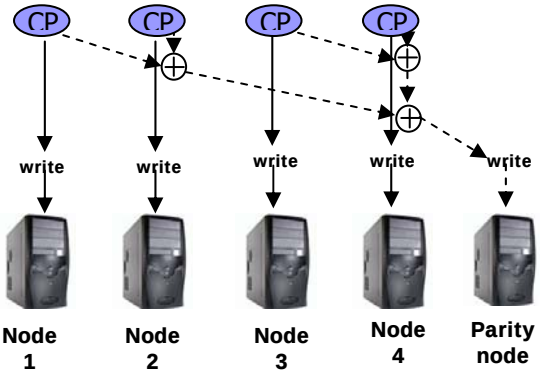

Figure 5. Parity grouping of local checkpoints

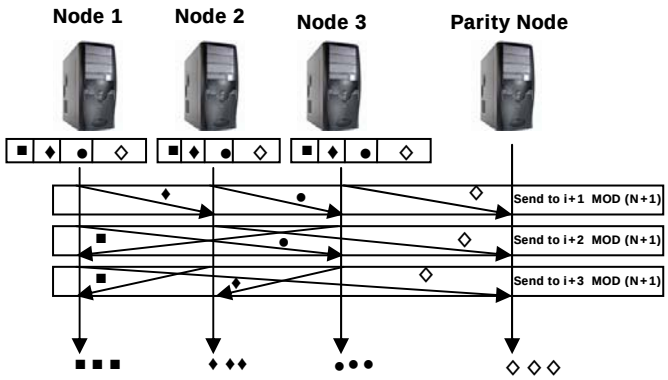

Figure 6. Intra-checkpoint distribution

Further assumptions regarding failure are as follows:

- A group failure will occur only if any connection to the checkpoint server is lost, for example, a front-end machine failure due to crash.
- The system will crash only if K-1 clusters get disconnected.
- In the case of a cluster disconnected or group failure, the processes that were being executed in this cluster will be restarted in another one.
- There will be no more than x-1 checkpoint-server failures in a cluster, where ‘x’ is total number of checkpoint servers in that cluster.

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

IV. RELIABLE CHECKPOINT STORAGE STRATEGY

Our protocol works in two steps, in storage phase the checkpoints are stored to checkpoint servers. The recovery phase is executed when any of calculation nodes fail and the last valid checkpoint images are downloaded.

A. *The Storage Phase*

In this phase, all the compute nodes send their chunks to the checkpoint servers in a distributed manner i.e**.,** in round robin fashion. The server receiving the chunk from the compute node becomes the primary server for that chunk and is responsible for replicating it on all the servers in the same cluster as well as on other clusters in the Grid. The intra-cluster replication is done in a hierarchical way according to the following formula:

$$\{s, s+2^0 \bmod [m], s+2^1 \bmod [m], \cdots, s+2^{n-1} \bmod [m]\} \quad \textbf{(1)}$$

where

s is the checkpoint server,

m is the No. of checkpoint server, and

n is the bit identifier of the checkpoint server (e.g. for 7 checkpoint servers 3 bits are used as their identifiers).

Each checkpoint server receiving the image from another checkpoint server sends back an ACK after properly storing that chunk. Meanwhile the checkpoint server also sends its chunk to checkpoint servers in other clusters; this replication is also done in a hierarchical manner according to the following formula:

$$\{c, c+2^0 \bmod [m], c+2^1 \bmod [m], \cdots, c+2^{n-1} \bmod [m]\} \quad \textbf{(2)}$$

where

c is the cluster number,

m is the total number of clusters, and

n is the bit identifier of cluster.

The server that receives the checkpoint becomes pseudo-primary for that chunk. To shorten the checkpoint wave completion time the pseudo-primary server saves the chunk and sends back a partial ACK to the primary server so that the primary server can acknowledge the client for that chunk and the client may proceed with processing. Then the checkpoint server ensures that all the clients of the same application have recorded their checkpoints correctly and validates the checkpoint wave locally. Concurrently the pseudo-primary servers replicate their respective chunks on all other checkpoint servers and on getting ACKs from each of them, the pseudo-primary servers send back a full ACK to the primary server.

B. *The Recovery Phase*

At the start of the recovery phase, different checkpoint servers conduct an agreement for the last valid checkpoint wave. In this agreement process all checkpoint servers send their last valid checkpoint wave number and the greatest number agreed by the majority of the servers becomes the result and then processing starts from that point onward. For this process, model B, also used by F. Bouabache in [2] is used. The model has the following properties

- After some time 'x' all correct processes will realize a process that was failed.
- After some time 'y' most of the processes will realize a process that is correct.

For each checkpoint server the technique defined in [1] is used to keep the list of failed checkpoints and to ensure that the agreement for last valid checkpoint wave is conducted by some correct process. After a failure a client first asks for a valid checkpoint wave number and checks it locally or it sends a request to the checkpoint servers of the same cluster for the chunks. The recovery is also done in a distributed fashion. All the checkpoint servers send the chunks to the requesting client for which these servers are primary. On receiving the chunks the image is reconstructed and the client is restarted.

V. EXPERIMENTAL RESULTS

The experiments were conducted in a GridSim simulator. For each cluster it was assumed that all the checkpoint servers are connected in a complete graph and all the intra-cluster links are faster than all the inter- cluster links

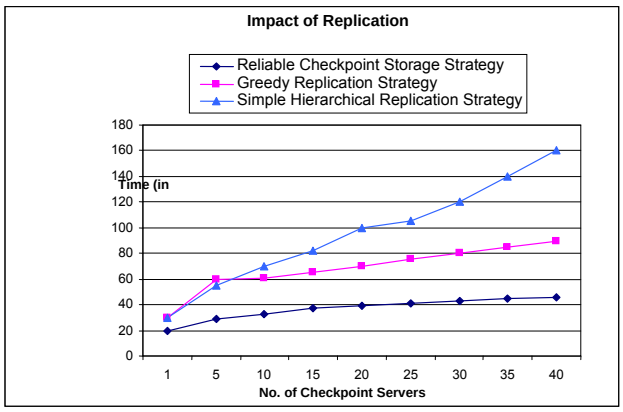

Figure 7. Impact of replication with respect to different numbers of Checkpoint Servers.

Initially, to compare the performance of three strategies i.e. Simple Hierarchical Replication, Greedy Hierarchical Replication and Reliable Checkpoint Storage Strategies, the clients' number was set to 200, the cluster number was set to 1, the checkpoint size per client to 1 Megabytes and the checkpoint server numbers are varied. Figure 7 presents the simulation results. Figure 7 represents the impact of replication within a cluster with respect to different numbers of checkpoint servers. It can be seen from the figure that the Reliable Checkpoint Storage Strategy replicates checkpoints to all of the servers in the group much faster the than other two strategies. The reason is that in Simple Hierarchical Replication the primary checkpoint server replicates the chunk over all other checkpoint servers and waits for the acknowledgement from them, and then it acknowledges the client to proceed with its processing.

In Greedy Hierarchical Replication, although the checkpoint servers are arranged in tree to replicate checkpoints but the servers exchange request and reply messages to inquire about the arrival of chunk and then act accordingly which results in long checkpoint wave completion time. While our strategy do not exchange control messages among checkpoint server to inquire about chunk arrival rather each checkpoint server sends the chunk to its peers with sequence number. If the receiver has already received the chunk with the same sequence number, it discards the received chunk; otherwise the receiver saves it. This clearly decreases the checkpoint wave execution time which is evident from figure 7. The Simple Hierarchical

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Replication was not considered for further experiments as it proved to be the slowest in accomplishing the task.

In figure 8, to investigate the scalability of clients' numbers within a cluster, the cluster and checkpoint server number were fixed at 1 and 6, respectively. It was observed during experiments that the checkpoint wave completion time depends on the number of clients; since, more the number of clients, greater is the data to be stored. To identify the step (client communication time and replication time) that affects the checkpoint wave completion time the most the two steps are isolated in the following figures.

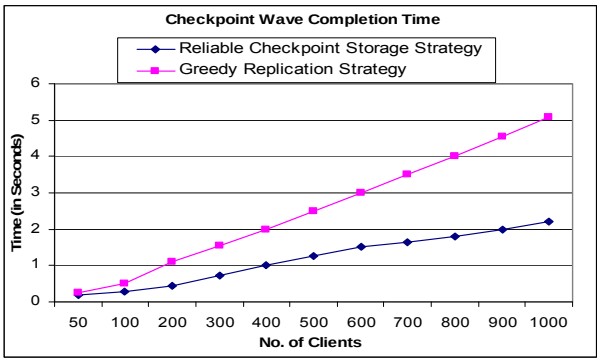

Figure 8. Checkpoint wave completion time with respect to different numbers of clients.

Figure 9 shows the effect of client communication time (the time it takes the client to send its chunks to checkpoint servers in a round robin fashion). When the number of clients were greater than 400 dual role nodes were used to store their own checkpoints locally which reduced the clients' communication time significantly. (For number of clients greater than 500, some stable nodes store their own checkpoints locally, and for number of clients' grater then 600 some checkpoint servers perform processing too).

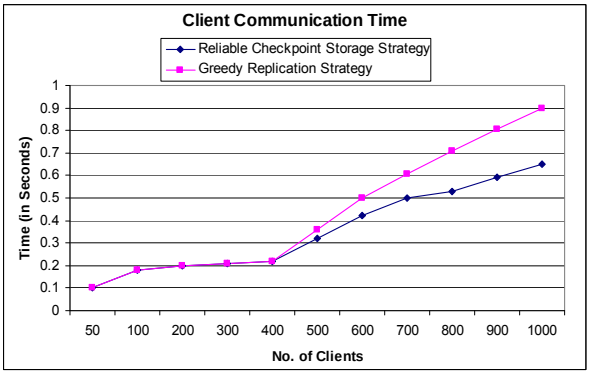

Figure 9. Impact of storage with respect to different numbers of clients.

Figure 10 shows the time it takes to replicate the chunks in the system with respect to increasing number of clients. The graph clearly depicts that replication time influences the checkpoint wave completion time the most. As compared to GHRS, RCSS performs well because of two reasons:

- There is no use of control packets.
- Dual role nodes are used for checkpoint storage, which decreases the overall checkpoint wave execution time.

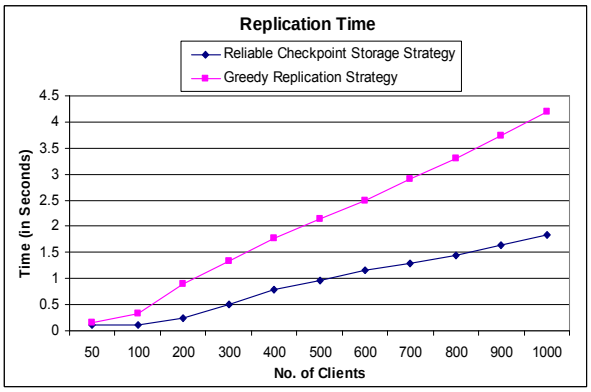

Figure 10. Impact of replication with respect to different numbers of clients.

Figure 11 depicts that by using tree topology RCSS outperforms the GHRS which replicates the chunks in all clusters in a flat order (primary checkpoint server in a cluster sends its chunks to all the clusters in the grid). For this experiment the number of clients 'c' was set to 100, number of checkpoint servers 'cs' was set to 30 and the number of clusters 'k' was varied. So there were c/k clients and cs/k checkpoint servers per cluster. The initial increase in checkpoint wave completion time is due to the increase in inter-cluster communication links which, as discussed earlier, are slower than intra-cluster communication links. But after 20-22 clusters, RCSS gave stable results because of topology. To make the graph easy to understand, client communication time and replication time are shown in two separate graphs as follows.

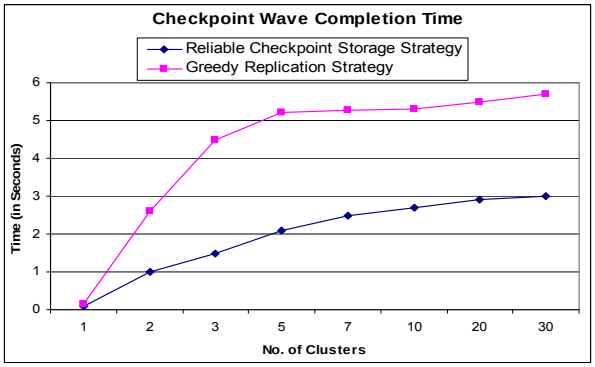

Figure 11. Impact of topology.

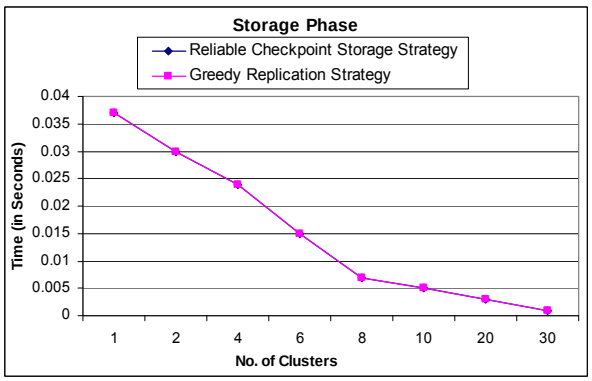

Figure 12. Impact of topology on client communication step.

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Figure 12 represents that as the number of cluster increases the number of clients per cluster decreases; and hence, the client communication time decreases too. In this graph the communication time of both RCSS and GHRS is same because the number of clients per cluster is less and so RCSS do not use the checkpoint servers for job execution.

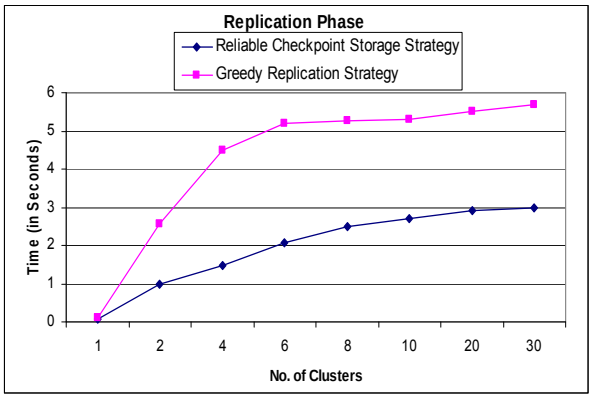

Figure 13. Impact of topology on replication step.

Figure 13 shows the impact of topology on the replication step with decreasing number of clients and checkpoint servers per cluster. It is evident from the figure that RCSS performs better than GHRS. The reason is that checkpoint servers in RCSS do not exchange requests and reply messages for chunks and secondly, when the cluster number increases, the topology used in RCSS overcomes the drawback of slower inter-cluster communication links. For this experiment clients number per cluster was set at 100 and checkpoint servers' number per cluster at 20. RCSS in figure 14 outperforms GHRS because of its intra-cluster and inter-cluster replication strategy that has been discussed in earlier section.

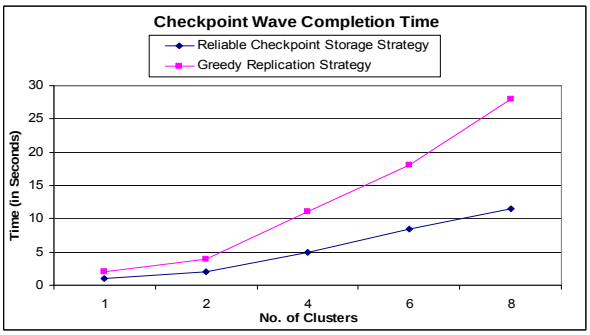

Figure 14. Impact of topology with fixed number of clients and checkpoint servers.

VI. SUMMARY

Fault tolerance is an important characteristic of High Performance Computing (Grid, Cluster). To make the system fault tolerant rollback recovery is a most commonly used technique which relies on availability of checkpoints at the time of recovery. Most often special devices are dedicated to store checkpoints and it is assumed that these devices can never fail but the reality contradicts this assumption. In Grid or Cluster environment any device can fail at any time. Furthermore dedication of powerful resources for checkpoint storage only results in wastage of these resources when they are needed the most.

We propose a protocol that ensures the availability of checkpoints in the case of checkpoint server failure or even in the case of cluster failure. Also our protocol utilizes the CPU cycles of some of dedicated checkpoint servers for computation when the load on network will be high. During practical measurement, in order to provide reliability the increase in network traffic was neglected. Placement of the jobs to resources was done in a random manner; the performance of this strategy can be improved with the help of a new and robust scheduling scheme.

REFERENCES

[1] J. Yu, R. Buya, " A taxanomy of work flow management systems for Grid computing", Journal of Grid Computing, Vol. 3, pp. 29, 2005.

[2] F. Bouabache, T. Herault, G. Fedak, "Hierarchical replication techniques to ensure checkpoint storage reliabilty in Grid environment", 8th IEEE International Symposium on Cluster Computing and the Grid, pp. 475 – 483, 2008.

[3] R.Y.De Camarge, F. Kon, "Strategies for checkpoint storage on opportunistic Grids" IEEE Computer Society, Vol. 7, No.9, pp. 1 - 1 September 2006.

[4] C. W. Cheng, J. J.Wu, P. Liu, " QoS-aware, access-efficient, and storage-efficient replica placement in grid environments" The Journal of Supercomputing, Vol. 49, No. 1, July, 2008.

[5] K. M. Chandy and L. Lamport. "Distributed snapshots:Determining global states of distributed systems. ACM Transactions on Computer Systems", Vol. 3, pp. 63–75, Feb 1985.

[6] Fiaz Gul Khan, Kalim Qureshi, Baber Nazir, "Performance Evaluation of Fault Tolerance Techniques in Grid Computing System", International Journal of Computer and Electrical Engineering, Volume 36, Issue 6, November 2010, *Elsevier.*

[7] J. S. Plank, "Improving the Performance of Coordinated Checkpointers on Networks of Workstations using RAID Techniques" IEEE Trans. Parallel Distributed System., pp. 76 – 85, 1996.

[8] Z. Chen, J. Dongarra "A Scalable Checkpoint Encoding Algorithm for Diskless Checkpointing", 11th IEEE High Assurance Systems Engineering Symposium, pp. 71 - 79 2008.

[9] P. Sobe "Stable checkpointing in distributed systems without shared disks", Parallel and Distributed Processing Symposium, pp. 8, 2003.

[10] Kalim Qureshi, Fiaz Gul Khan, Paul Manuel, Babar Nazir "A Hybrid Fault Tolerance Technique in Grid Computing System" published in Journal of Supercomputing, *Springer*, January 19, 2010. DOI:10.1007/S11227-009-0345-Y.

ISBN: 978-988-19251-1-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)