# Performance comparison of hierarchical checkpoint protocols grid computing

Ndeye Massata Ndiaye, Pierre Sens, Ousmane Thiare

# Performance Comparison of Hierarchical Checkpoint Protocols Grid Computing

Ndeye Massata Ndiaye, Pierre Sens and Ousmane Thiare

**Abstract** Grid infrastructure is a large set of nodes geographically distributed and connected by a communication. In this context, fault tolerance is a necessity imposed by the distribution as any node can fail at any moment and the average time between failures highly decreases. To improve the robustness of supercomputing applications in the presence of failures, many techniques have been developed to provide resistance to these faults of the system. Fault tolerance is intended to allow the system to provide service as specified in spite of occurrences of faults. To meet this need, several techniques have been proposed in the literature. We will study the protocols based on rollback recovery classified into two categories: checkpoint-based rollback recovery protocols and message logging protocols. However, the performance of a protocol depends on the characteristics of the system, network and applications running. Faced with the constraints of large-scale environments, many of algorithms of the literature showed inadequate. Given an application environment and a system, it is not easy to identify the recovery protocol that is most appropriate for a cluster or hierarchical environment, like grid computing. Hence there is a need to implement these protocols in a hierarchical fashion to compare their performance in grid computing. In this paper, we propose hierarchical version of these protocols. We have implemented and compare their performance in clusters and grid computing using the Omnet++ simulator.

Ndeye Massata Ndiaye
Gaston Berger Univsersity of Saint-Louis Senegal and Regal team Paris Jussieu, e-mail: ndeye-massata.ndiaye@lip6.fr

Pierre Sens
Regal team Paris Jussieu e-mail: Pierre.Sens@lip6.fr

Ousmane Thiare
Gaston Berger University of Saint-Louis Senegal e-mail: ousmane.thiare@ugb.edu.sn

## 1 Introduction

Today, grid computing technologies make it possible to securely share data and programs for multiple computers, whether desktop or personal supercomputers. These resources are networked and shared through software solutions. Many grids are appearing in the sciences, production grids are now being implemented in companies and among agencies: Grid'5000, TeraGrid, Sun Grid, Xgrid ... Grid computing will allow dynamic sharing of resources among participants, organizations and businesses in order to be able to pool, and thus run compute-intensive applications or treatment of very large volumes of data. Since the probability of failure increases with a rising number of components, fault tolerance is an essential characteristic of massively parallel systems. Such systems must provide redundancy and mechanisms to detect and localize errors as well as to reconfigure the system and to recover from error states. A fault tolerant approach may therefore be useful in order to potentially prevent a faulty node affecting the overall performance of the application. Fault tolerance appears then as an indispensable element in grid computing. Many protocols for distributed computing have been designed [1]. These protocols are classified into four different groups, namely, coordinated checkpointing, communication induced checkpointing, independent checkpointing and log-based protocols. We have implemented and compared the performance of these protocols in clusters and grid computing using the Omnet++ simulator [7]. Section 2 describes the protocols implemented in Omnet++. The experimental setup and results obtained by executing these protocols are presented in Section 4. In section 5, we present the related work and finally section 6 concludes.

## 2 Checkpoint and rollback-recovery protocols

Checkpointing is a standard method for the repairing of faults in systems. The idea is to save the state of the system on a stable period to prevent breakdowns (Figure. 1). That way, when you restart after a power failure, the state last saved is restored and the execution before the crash resumes. The overall status of a distributed system is defined by the union of local states of all processes belonging to the system.

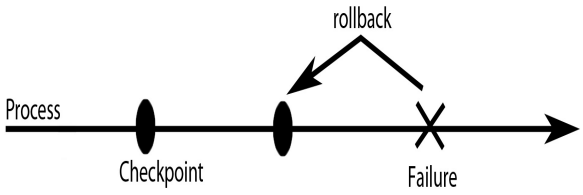


**Fig. 1** Rollback recovery

There are two main classes of protocols: coordinated checkpointing and message logging.

## *2.1 Coordinated checkpointing*

The protocol requires processes coordinate their checkpoints to form a consistent global state. A global state is consistent if it does not include any orphan messages (i.e a message received but not already sent). This approach simplifies the recovery and avoids the domino effect, since every process always restarts at the resume point later. Also, the protocol requires each process to maintain only one permanent checkpoint in stable storage, reducing the overhead due to storage and release of checkpoints (garbage collection) [1].
Its main drawback however is the large latency that require interaction with the outside world; in this case the solution is to perform a checkpoint after every input / output. To improve the performance of the backup coordinated, several techniques have been proposed. We have implemented as non-blocking coordinated checkpointing.

- Non-blocking coordinated checkpointing: the example of coordinated checkpoint non-blocking is that of Chandy and Lamport algorithm [2]. This algorithm uses markers to coordinate the backup, and operates under the assumption of FIFO channels. In [3], a comparison of protocols between a blocking and a non-blocking coordinated checkpoint has been made. Experiments have shown that the synchronization between nodes induced by the blocking protocol further penalize the performance of the calculation with a non-blocking protocol. However, using frequencies of taken checkpoints usual performance of the blocking approach is better on a cluster to high-performance communications.
- Communication induced checkpointing: this protocol defines two types of checkpoints [1]: local checkpoints taken by processes independently, to avoid the synchronization of coordinated backup and forced checkpoints based on messages sent and received and dependency information carried 'piggyback' on these posts, so to avoid the domino effect of uncoordinated backup, ensuring the advancement of online collection. Unlike coordinated checkpoint protocols, the additional cost due to the medium access protocol disappears because the protocol does not require any exchange of message to force a checkpoint: this information is inserted piggyback on the messages exchanged.

## *2.2 Message-Logging protocols*

The logging mechanism uses the fact that a process can be modeled as a sequence of deterministic state intervals, each event begins with a non-deterministic. An event may be the receiving or the issuing of a message or other events in the process. It is deterministic if from a given initial state, it always occurs at the same final

state [1]. The principle of Logging is to record on a reliable storage support any occurrences of non-deterministic events to be able to replay them in recovering from a failure. During execution, each process performs periodical backups of their states, and records log information about messages exchanged between processes. There are three message-logging categories: optimistic, pessimistic and causal.

- Pessimistic message-logging: this protocol was designed under the assumption that a failure may occur after any nondeterministic event (i.e. message reception). It is often made referring to the synchronized because when logging process logs an event of non-deterministic stable memory, it awaits for an acknowledgment to resume its execution. In a pessimistic logging system, the status of each process can be recovered independently. The main drawback is the high latency of communications which results in degradation of the applications response time.

- Optimistic message-logging: this protocol uses the assumption that the logging of a message on reliable support will be complete before a failure occurs. Indeed, during the execution of the process, the determinants of messages are stored in volatile memory, before being saved periodically on stable support. The storage of stable memory is asynchronous. Induced latency is then very low. However, a failure may occur before the messages are saved on stable storage support. In this case, the information stored in volatile memory of the process down are lost and the messages sent by this process are orphaned.

- Causal message-logging: this protocol combines the advantages of both previous methods. As optimistic logging, it avoids the synchronized access to the stable support, except during the input / output. As pessimistic logging, it allows the process to make interactions with the outside world independently, and does not create process orphan. Causal logging protocols piggyback determinants of messages previously received on outgoing messages so that they are stored by their receivers.

## 3 Hierarchical checkpointing for Grids

The architecture of a grid can be defined as a set of clusters connected by a WAN-type network. The cluster consists of multiple nodes connected by a broadband network. We adopt a hierarchical scheme. In each cluster, there is one leader connected to all other nodes of its cluster. All leaders are connected together (Figure. 2). The leader assumes the role of intermediary in the inter-cluster communications. The backup takes place in four phases:

- Initialization: an initiator sends a checkpoint-request to its leader,
- Coordination of leaders: the leader transfers the checkpoint request to the other leaders
- Local checkpointing : Each leader initiates a checkpoint inside its cluster

- Termination: When local checkpoint is over, each leader sends an acknowledgement to the initial leader.

The recovery follows the same rules as the backup: coordination phase of the leaders, and a phase of recovery limited to the cluster.

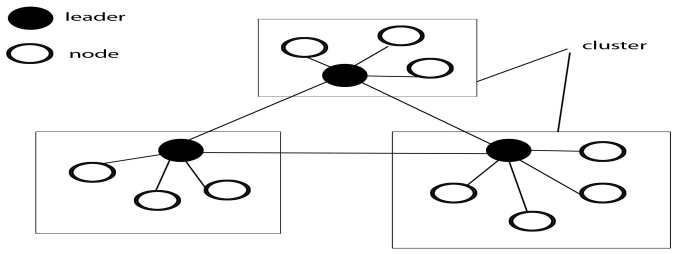


**Fig. 2** Hierarchical checkpointing for grids

## 4 Performance evaluation

In most of the previous studies, fault tolerance algorithms were tested in flat architectures, namely in a cluster. The aim of our study is to determine which algorithm best suits the architectural grid. For that purpose, we implement the seven checkpoint algorithms described in Section 2: the 3 main messages logging protocols (represented as ML in the figures), Chandy-Lamport, Communication induced protocol (CIC in figures), and blocking coordinated checkpointing (CheckpointCoord). We compare the performance of these algorithms in cluster and grid environments. We use the Omnet++ simulator [7]. The cluster is configured with 25 nodes. For the grid configuration, 25 nodes were uniformly spread in 5 clusters. The intra-cluster delay is fixed to 0.1 ms and the inter-cluster delay is fixed to 100ms. Our tests were carried out with 50 application processes. Messages between processes were randomly generated.

### *4.1 Failure free performance*

Figure. 3 presents the performance of the algorithms in both configurations. It is obvious that the time taken to run an application with checkpointing is longer than the time it takes to run it without checkpoint. Protocol overhead checkpoint coordinated non-blocking is less compared to other approaches to that phase synchronization is limited to the cluster and the second concerns only the leaders of each cluster. The additional cost of communications-driven approach is due to the forced checkpoints

during execution. Logging protocols are sensitive to characteristics of the application, especially in communications-intensive applications. Indeed, they produce a large overhead due to the backup of messages on stable storage and the increasing size of messages to piggyback determinants.

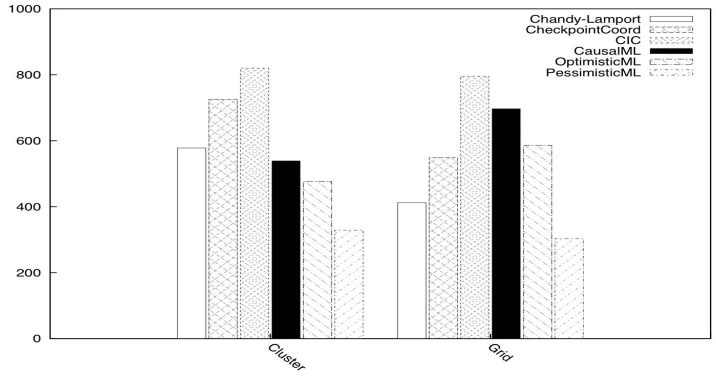


**Fig. 3** Failure free performance, Checkpoint interval=180s, Execution time=900s

## *4.2 Recovery time*

The recovery time depends on the number of checkpoints maintained by the protocol and the number of rollbacks. In coordinated checkpointing and pessimistic logging, recovery is simplified because the system is rolled back only to the most recent checkpoint. In the grid approach, the additional cost of recovery decreases slightly. Indeed, if the faulty node has no dependencies with nodes of other cluster nodes, the fault is confined to the cluster node's fault. So all the nodes of the grid do not perform the recovery procedure. By cons, if the inter-cluster communications are intensive, the overhead increases as in the case of causal and optimistic logging.

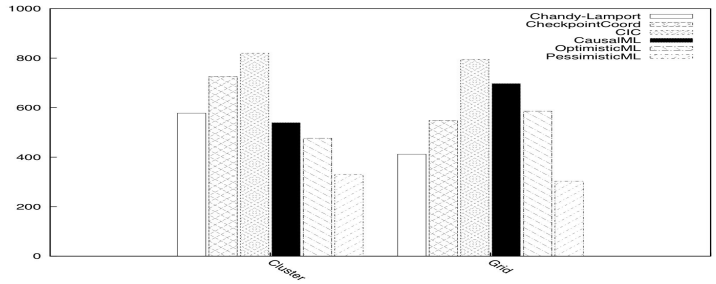


**Fig. 4** Overhead of recovery,checkpoint interval=180s,execution time=900s,numbers of fault=10

### *4.3 Number of rollbacks*

For coordinated checkpoint protocols, all processes must resume during recovery. The logging protocol reduce the number of rollback. This number is minimal in pessimistic approach since only faulty processes need to be rolled back. For the other logging protocol, this number depends on the information stored in backups and in the main memory of correct processes

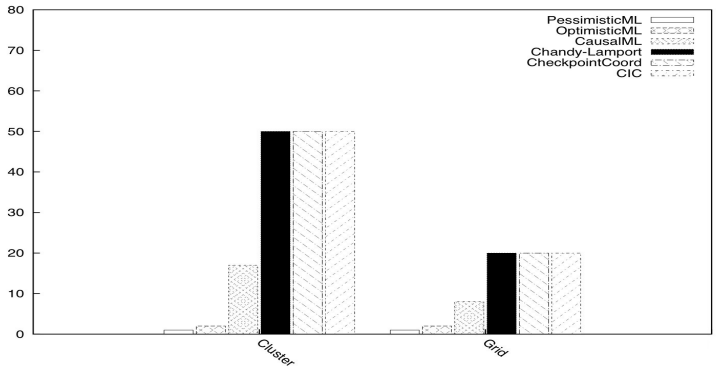


**Fig. 5** Number of process, Checkpoint interval=180s, Execution time=900s, Numbers of fault= 1

## 5 Related work

Paul and al. [4] proposes a hierarchical protocol based on coordinated checkpoint. The hierarchical checkpoint protocol is in two phases. The first phase is the execution of the blocking coordinated checkpoint algorithm limited to the cluster. The second phase is a coordinated checkpoint but the leaders of the clusters are the only participants, with the initiator, which acts as a coordinator. The experiments showed that the overhead of checkpointing in the hierarchical approach is lower than in the standard flat coordinated protocol. However the protocol hierarchy is sensitive to the frequency of messages between clusters. Indeed the extra cost of checkpoint increases progressively as the frequency of messages increases, and tends towards that of the checkpoint protocol standard.
Bhatia and al. [5] propose a hierarchical causal logging protocol that addresses the scalability problems of causal logging. Authors propose a hierarchical approach using a set of proxies spread on the network that act as a distributed cache. This approach highly reduces the amount of information piggybacked on each messages. However, the use of proxies decreases the performance of recovery since the recovery information is spread on the proxies.
Monnet and al. [6] propose a hierarchical checkpointing protocol which combines

coordinated checkpointing inside clusters and a checkpoints induced by communications between clusters. Simulation of the protocol shows that it generates a high number of forced checkpoints when the communication rate between clusters increases. Then, this approach is more suitable for code coupling applications where communications are mainly local inside clusters.

## 6 Conclusion

In this paper, we compared checkpoint protocols and message-logging in grid computing. We propose a hierarchical approach to combine different algorithms. We found that the protocols that require the recovery of all processes in case of single failure are poorly suited to systems with many processes. The message logging protocols are more suitable for large configuration with the exception of some causal logging approach which induces communications to all processes during the recovery. Non-blocking coordinated checkpoint are not sensitive to the rate of communications. They therefore represent an attractive solution for applications and highly interconnected grid architectures by reducing the number of markers sent during the synchronization phase.

## References


1. Elnozahy, E. N., Alvisi, L., Wang, Y.-M. & Johnson, D. B. (2002). A Survey of Rollback-Recovery Protocols in Message-Passing Systems. *ACM Computing Surveys, vol. 34, no. 3* (pp. 375–408).
2. Chandy, M., & Lamport, L. Distributed snapshots: Determining global states of distributed systems. *ACM Trans. Computing Systems, vol 3, no. 1* (pp. 63–75).
3. Coti, C., Herault, T., Lemarinier, P., Pilard, L., Rezmerita, A., Rodriguez, E., & Cappello, F. (2006). Blocking vs. non-blocking coordinated checkpointing for large-scale fault tolerant MPI. *In SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing* (pp. 127). New York: USA.
4. Himadri, S.-Paul, Gupta, A., & Badrinath., R. ( 2002). Hierarchical Coordinated Checkpointing Protocol. *In International Conference on Parallel and Distributed Computing Systems* (pp. 240–245).
5. Bhatia, K., Marzullo, K., & Alvisi, L. ( 2003). Scalable causal Message Logging for Wide-Area Environments. *Concurency and Computation: Practice and Experience, 15(3)* (pp. 873–889).
6. Monnet, S., Morin, C., & Badrinath, R. ( 2004). Hybrid Checkpointing for Parallel Applications in cluster Federations. *Proc. 4th IEEE/ACM International Symposium on Cluster Computing and the Grid* (pp. 773–782). Chicago: USA.
7. http://www.omnetpp.org.