Hierarchical composition of coordinated checkpoint with pessimistic message logging

Ndeye Massata Ndiaye[#*1], Pierre Sens[*2], Ousmane Thiare[#3]

[#]*Department of computer science, Gaston Berger University*
BP. 234 Saint-Louis, Senegal

[*]*Regal Team, LIP6*
4 Place Jussieu 75525 Paris Cedex 05, France

[1]ndeye-massata.ndiaye@lip6.fr

[2]Pierre.Sens@lip6.fr

[3]ousmane.thiare@ugb.edu.sn

***Abstract*—Grid computing mutualizes more computing resources working in a calculation or a common task. The increase in the number of components in the system leads also increases the number of fault. These failures result in a loss of several cycles of running applications. It is therefore essential to be able to tolerate faults so that the computation can continue to execute and finish despite failures, all while maintaining maximum performance. One advantage of coordinated checkpoint is its capacity to have a very low overhead as long as the execution stays fault free. On the contrary, due to the fact that uncoordinated checkpoint requires being complemented by a message log protocol, this adds a significant penalty for all message transfers, even in case of fault-free execution. With message log, these problems do not arise simply because it processes checkpoint and restart autonomously. These differences suggest that the best approach depend on the fault frequency. In this paper, we propose a hierarchical composition of algorithms: Chandy-Lamport protocol and pessimistic message logging protocol. We have implemented and compared the performance of these protocols in grid computing using the Omnet++ simulator [3].**

Keywords—Grid computing; coordinated checkpointing; chandy-lamport; pessimistic message logging

I. INTRODUCTION

Currently, many applications require significant computing power in order to run. Thus, some applications need a high performance-computing environment that is able to provide results within a specified period, while others require the ability to use the available computing power.

Today, grids enable the scientific community access to shared resources. The large number of federated resources by the grid leads to an increase of the probability of failure. Fault tolerance is an essential feature to ensure continuity of service in these platforms grid. These must provide mechanisms for detection and correction of errors in the execution of court applications.

To meet this need, several fault tolerances have been proposed in the literature. Our study is based on the protocols based on rollback recovery classified into two categories: checkpoint-based rollback recovery protocols and message logging protocols. However, there is a direct dependence between the characteristics of system, network and applications on the one hand, and the performance of a protocol on the other hand.

In [2], the performances of these protocols have implemented and compare in clusters and grid using the Omnet++ simulation [3].

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 [2]. 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 [2].

We propose a hierarchical composition of the following protocols: Chandy-Lamport protocol and pessimistic message logging protocol. Section 2 describes the protocols Chandy-Lamport and Pessimistic message logging used for the hierarchical composition. The details of the hierarchical composition are explained in section 3. The experimental setup and results obtained by executing the protocol is presented in Section 4. In section 5, we present the related work and finally section 6 concludes.

II. CHANDY-LAMPORT AND PESSIMISTIC MESSAGE LOGGING

Fault tolerance in parallel applications is provided by two categories of protocols: checkpointing protocols and message logging protocols [1]. To ensure the reliability of applications, the need to preserve the state of an application to preserve complete the calculation in the case of system failure is extremely important. Rollback recovery techniques are a common form of this type of conservation state, and have received much attention from the research community.

A process is modeled as a sequence of intervals state. Each state interval starts with a non-deterministic events, such as a user input or an incoming message, but is to run up to the deterministic state interval is attained.

Checkpointing and rollback recovery techniques usually assume a fail-stop model of defects. It is assumed that each process has access to some sort of stable storage, which can still be viewed after the process has failed. "Stable Storage" can come in a variety of forms, depending on assumptions made by the recovery protocol. It should not be a real disk. If a system should tolerate a single failure, stable storage could be set implemented using volatile memory of other processes in the system. If failures are assumed to be transient, the local hard drive of a host process can be used. If failures are not transient, the local hard drive from a host cannot be used because this record will not be accessible after a host failure. In this case, storage-stable process has to be found at distance from the host process.

A set of checkpoints is consistent if there are no orphan messages between processes. An orphan message in a global state is a message that was sent after a checkpoint belonging to this global state and received before a checkpoint belonging to this global state.

To recover from a process, the recovery procedure must ensure that the internal state of the process is recovered in line with the observed state of the system before failure. This is accomplished by identifying the latest set of consistent checkpoints and restore system to saved state in this set. This set of checkpoints is called the recovery line.

In [1], a comparison of different protocols backup checkpoint and logging was done. The results showed us that the Chandy-Lamport algorithm executed in grids gives better performance compared to other protocols backup checkpoint. Similarly, the pessimistic message logging is better suited to architectures grids especially since it does not produce orphaned process. That is why we chose these two protocols for conducting assessments of performance in cluster mode and grid mode. We will give a brief description of these protocols.

A. Non-blocking coordinated checkpoint protocol

The goal of fault tolerance is to allow the system to continue to provide the service despite the occurrence of faults. Depending on application requirements, a fault tolerant system must be able to support a given number of faults (which may be simultaneous), and different types of faults.

A global state corresponds to the set of states of individual processes and states participating communication channels. A consistent global state is a condition that could have occurred in the absence of faults during a correct execution of a parallel application. This is not necessarily a condition that occurred before the takeover. During a failure, the protocols based checkpointing restore the global state of the system from local checkpoints of each process forming coherent global state called the most recent recovery line [4].

To form this recovery line, three strategies are possible [1]: uncoordinated checkpointing, coordinated checkpointing, or communication induced checkpointing. On resumption, it is sufficient to restore all of the application process from their respective checkpoint. For our study we will use the protocols coordinated checkpoint. This technique ensures that the set of checkpoints forms a consistent global state and thus avoids the domino effect. Furthermore, the recovery is very simple. Another advantage of this technique is that the storage space needed to keep the checkpoints is minimized since it is only necessary to keep only one per process.

The main drawback of this strategy is that it involves significant latency when saving the checkpoint, since it requires a global coordination of all processes. Two problems arise here, therefore, the loss of performance even in the absence of faults, and management of scaling. Indeed, over the number of processes increases, the latency may be large, each process to wait until the end of the operation before it can resume its normal execution.

This approach, used by the first implementations, can be optimized. Also, to improve backup performance coordinated, several techniques have been proposed:

- Checkpointing coordinated non-blocking [1]
- Checkpointing with clock synchronization [1]
- Coordinated checkpointing to synchronization minimal [1]

The “distributed snapshot” algorithm of Chandy and Lamport [5] is the most classical algorithm of non-blocking coordinated checkpoint. It operates under the assumption of channel FIFOS. It is made of coordinating processes to ensure that all states of the processes form a consistent global state. The protocol allows the state identifier communication channels through "markers". During a backup step, each process saves its local state and then broadcasts a marker immediately on all communication channels. Then, it saves all messages received on each communication channel until it receives a marker. This set of messages corresponding to the state of the communication channel, which will then be saved on stable storage.

The advantage of the coordinated backup is that it is not sensitive to the domino effect during recovery. Only the last backup is necessary for a restart, which reduces the extra cost of storage.

B. Pessimistic message-logging protocol

Message-logging protocols are popular when building systems that can tolerate process crash failures [7]. These protocols require that each process periodically record its own local state and log the messages received right after. When a crash occurs, a new process is created instead: the new process is given the appropriate (recorded) local state and the logged messages are issued in the order they were originally received.

The principle of fault tolerance through logging is to safeguard the history of the application. Logging protocols by using both local backup of the process status and event logging deterministic to allow resumption of the operation. It is then possible to resume the process execution failed (and only faulty processes) from their last backup by replaying the nondeterministic events saved. The pessimistic message logging protocol assumes (pessimistic) that a failure may

occur immediately after a nondeterministic event. The principle of this protocol is not to allow a process to depend on a non-deterministic event. Specifically, if we consider that non-deterministic events are only receiving messages, this protocol requires processes to save all messages received before issuing a message to another process. The backups should be performed synchronously. The state of each process in a pessimistic logging system is always recoverable and this property leads to the following advantages [1]:

- A process can commit output to the outside world without running a special protocol.
- Recovery is simplified because the effects for a failure are confined only to the processes that fail.
- Functioning processes continue to operate and never become orphans.
- Processes restart from their most recent checkpoint upon a failure, therefore limiting the extent of execution that has to be replayed.
- There is no need to run a complex garbage collection protocol

In [2], the authors show that during the failure free execution, protocol overhead of nonblocking coordinated checkpoint (Chandy-Lamport algorithm) is less compared to other approaches because the phase synchronization is limited to the cluster and the second concerns only the leaders of each cluster. 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. In pessimistic approach, the number of rollbacks is minimal since only faulty processes need to be rolled back.

III. HIERARCHICAL COMPOSITION

We propose to make a composition using the hierarchical algorithm Chandy-Lamport and pessimistic logging. Indeed, the pessimistic message logging be applied within each cluster and between clusters the nonblocking coordinated checkpoint protocol will be used.

Figure 2 shows a simplified view of the grid architecture used by our algorithm. It is composed of a set of clusters connected by a WAN- type network. The cluster consists of multiple nodes connected by a broadband network. In each cluster, there is one leader connected to all other nodes of its cluster. All leaders are connected together (Fig.1). The leader assumes the role of intermediary in the inter-cluster communications.

Within each cluster, the messages are saved using the pessimistic message logging protocol. Indeed, when a process sends a message, it saves it in its volatile memory. Similarly, the identifier of the receiving process and the issue date of the message are saved. After receiving the message, the receiving process sends an acknowledgment to the transmitter, which will store the message on stable storage. All the sending processes save all messages that flow into the grid before interacting with other processes, which prevents orphan messages.

The backup takes place in four phases [2]:

- 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.

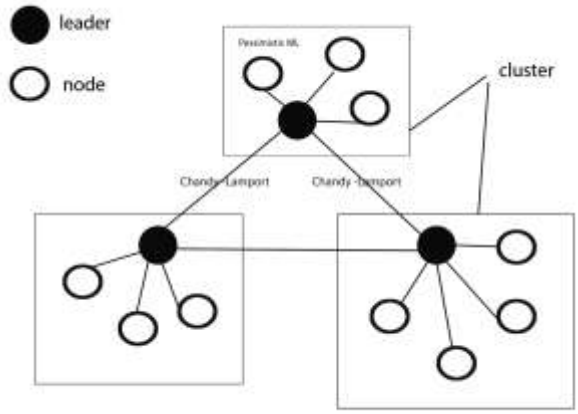

Fig. 2: Hierarchical checkpointing for grids

IV. PERFORMANCE EVALUATION

To evaluate our hierarchical algorithm, we did experiments on both types of application:

- "Token-application": at the beginning of the execution of the application, a node sends a message initiator. Upon receipt of the message, it is in turn sent to another process and so on, until the end of the application at run. As the grid is organized as a cluster (group of nodes), each message passes through its leader before being routed to its final destination.
- An application for broadcasting messages (represented as "Broadcast-Application" in the figures): all 30s, a message is broadcast to all nodes of the grid until the end of the execution of the application. Initially, the message is sent to all leaders of clusters, and then they distribute them within their clusters.

The experiments were conducted in two phases. The characteristics of defect-free protocols are studied in the first phase. Finally, in the second phase, the same sets of experiments were made with fault injection.

We have implemented our algorithm and application in the Omnet++ simulator [3].

A. Failure free performance

To measure the impact of logging messages on the response time of applications, we varied the size of messages. The figure 2 (Fig.2) shows the curves of changes in application response time using the following protocol: the hierarchical composition of Chandy-Lamport [5] algorithm with pessimistic message logging [6] (represented as "CLPML" in the figures) and the non-blocking coordinated checkpoint protocol only [2] ("CL" in the figures). Obviously, the "Token-Application", being highly communicative, has a response time higher than the application to broadcast messages. We note that the response time increases gradually as the message size increases. This is due to the data added to messages during the backup of the latter by their issuers. By cons, response time exhibits only slight variation by adding message logging to Chandy-Lamport protocol. In summary, the pessimistic logging used in the composition of our hierarchical algorithm has a very low impact on the response time of the application.

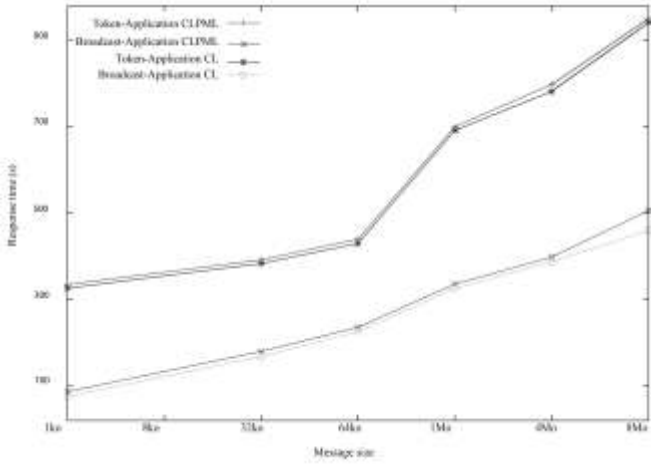

Fig. 2: Failure free performance, Checkpoint interval=180s, Execution time=900s

B. Recovery time with fault injection

The following figure shows the variation in response time of the "Token-Application" during recovery applications. Applying the Chandy-Lamport algorithm, all processes are restored from the most recent checkpoint. To finalize the calculation of the overall state of the grid, messages sent after the last point are replayed. To measure the impact of messages on application performance, we injected faults on a scale of 1 to 10.

We find that even the number of replayed messages is high; it has a very low impact on response time. Indeed, thanks to the pessimistic message logging, the messages sent are stored at the sending process before interacting with the system. So the process does not regenerate these messages, they will simply re-emit by the process.

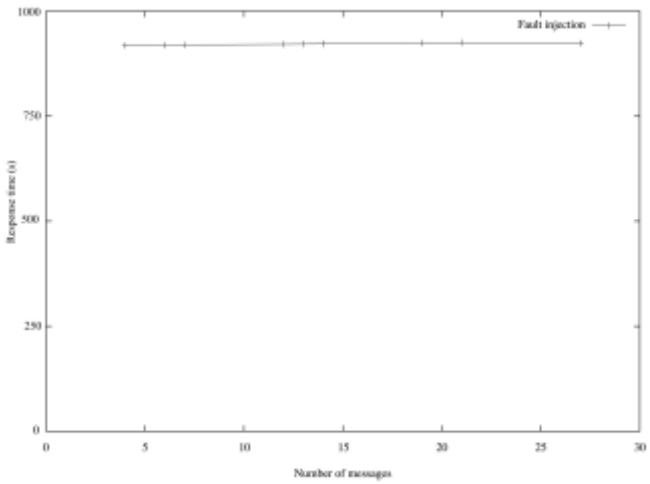

Fig. 3 Response time of Token-Application, checkpoint interval=180s, execution time=900s,numbers of fault=10

V. RELATED WORK

Many protocols with a combination between checkpointing and rollback-recovery have been proposed. Bhatia et al. [8] suggest a hierarchical causal logging protocol that addresses the scalability problems of causal logging. The protocol uses a network of proxies that cached recovery information while routing application. Indeed, the traditional causal logging algorithms are used successfully in small-scale systems. They are known to provide a low overhead during failure-free executions sending no extra messages. But they are not scalable since each application process needs to maintain a data structure, which grows with the number of processes in the system. Authors have reduced the data structure by an exponential amount. They have proposed 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 message. However, the use of proxies decreases the performance of recovery since the recovery information is spread on the proxies.

In [9], authors present an uncoordinated checkpointing protocol for send-deterministic [10] HPC applications. This protocol just requires logging a small subset of application messages to avoid the domino effect. Furthermore, like in message logging protocols, it does not require all processes to rollback in case of failure. Since there is no domino effect, a simple and efficient garbage collection can be done. Finally, by allowing checkpoint scheduling, this protocol appears to be some good way to address the problem of burst accesses to the I/O system.

VI. CONCLUSION

In this paper, we propose a hierarchical composition of algorithms: Chandy-Lamport protocol and pessimistic message logging protocol. We have implemented and compared the performance of the algorithm Chandy-Lamport only and hierarchical combination of pessimistic logging

protocols and Chandy-Lamport, using the simulator OMNeT + + [3].

We found that the highly interconnected applications like, "Token-application" have a high response time. But the use of pessimistic message logging protocol has an insignificant impact on performance when running applications. Indeed, 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. The hierarchical composition of the Chandy-Lamport algorithm and pessimistic message logging protocol is a good alternative to fault tolerance in computational grids.

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] Ndiaye, N.M, Sens, P. & Thiare, O. (2012). Performance comparison of hierarchical check- point protocols grid. Advances in Intelligent and Soft Computing, 2012, Volume 151/2012, 339-346, DOI: 10.1007/978-3-642-28765-7-40

[3] http://www.omnetpp.org

[4] Elnozahy, E.N., Johnson, D.B. & Zwaenepoel, W. (1992). The performance of consistent checkpointing. In Symposium on Reliable Distributed Systems, pp. 39-47.

[5] Chandy, K.M. & Lamport, L. (1985). Distributed snapshots: Determining global states of distributed systems. In Transactions on Computer Systems, vol. 3(1). ACM, pp. 63-75.

[6] Johnson, D.B. & Zwaenepoel, W. (1987). Sender-Based Message Logging. In Digest of Papers: The 17th Annual International Symposium on Fault-Tolerant Computing, pp. 14-19.

[7] Alvisi, L. Marzullo, K. (1998). Message logging: Pessimistic, optimistic, causal, and optimal. IEEE Trans. Software Eng, vol. 24, no. 2, pp. 149-159.

[8] 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.

[9] Guermouche, A., Ropars, T, Brunet, E., Snir, M. and Cappello, F. (2011). Uncoordinated Checkpointing Without Domino Effect for Send-Deterministic Message Passing Applications. In 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS2011), Anchorage, USA, pp. 989-1000.

[10] 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.

[11] Himadri, S. Paul, Gupta, A. Badrinath, R. (2002). Hierarchical Coordinated Checkpointing Protocol. In International Conference on Parallel and Distributed Computing Systems, pp. 240--245.

[12] 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. 127, New York, NY, USA.