A Recoverable Distributed Shared Memory Integrating Coherence and Recoverability

Anne-Marie Kermarrec, Gilbert Cabillic, Alain Gefflaut, Christine Morin, Isabelle Puaut

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Recoverable Distributed Shared Memory Integrating Coherence and Recoverability

Anne-Marie Kermarrec, Gilbert Cabillic, Alain Gefflaut,
Christine Morin, Isabelle Puaut

N° 2481

Janvier 1995

PROGRAMME 1

ISSN 0249-6399

INRIA
RENNES

A Recoverable Distributed Shared Memory Integrating Coherence and Recoverability

Anne-Marie Kermarrec, Gilbert Cabillic, Alain Gefflaut,
Christine Morin, Isabelle Puaut

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projet SOLIDOR

Rapport de recherche n°2481 — Janvier 1995 — 18 pages

Abstract: Large-scale distributed systems are very attractive for the execution of parallel applications requiring a huge computing power. However, their high probability of site failure is unacceptable, especially for long time running applications. In this paper, we address this problem and propose a checkpointing mechanism relying on a recoverable distributed shared memory (DSM). Although most recoverable DSM require specific hardware to store recovery data, our scheme uses standard memories to store both current and recovery data. Moreover, the management of recovery data is merged with the management of current data by extending the DSM's coherence protocol. This approach limits the hardware development and takes advantage of the data replication provided by a DSM in order to limit the amount of transferred pages during the checkpointing.

The paper also presents an implementation and preliminary performances evaluation of our recoverable DSM on an Intel Paragon with 56 nodes. In particular, it shows performance degradation introduced by the fault tolerance mechanisms on failure-free executions, compared to a standard DSM on the same architecture.

Key-words: Distributed Shared Memory, Large-scale Distributed Systems, Fault-Tolerance, Backward Error Recovery, Checkpointing

(Résumé : tsvp)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Proposition d'une mémoire partagée distribuée recouvrable intégrant cohérence et disponibilité

Résumé : Les systèmes distribués extensibles sont particulièrement adaptés à l'exécution d'applications parallèles nécessitant une importante puissance de calcul. Toutefois, ils ont, en raison du nombre important d'éléments dont ils sont composés, une forte probabilité de défaillance qui est innaceptable pour les applications de longue durée. Dans ce rapport, nous proposons un mécanisme de prise de point de récupération pour une mémoire partagée distribuée à haute disponibilité. De nombreuses mémoires partagées distribuées à haute disponibilité repose sur l'utilisation de matériel spécifique pour stocker les données de récupération. Notre proposition, au contraire, est d'utiliser les mémoires standard pour stocker à la fois les données courantes et les données de récupération. De plus, la gestion des données de récupération est entièrement intégr'ee à celle des données courantes par une extension du protocole de cohérence de la mémoire partagée distribuée. Cette approche limite le développement matériel et en profitant de la replication de données inhérente au fonctionnement de la mémoire partagée distribuée, limite la quantité de pages tranférées lors de la prise du point de récupération. Dans ce rapport, une mise en œuvre de notre mémoire partagée distribuée à haute disponibilité sur une Intel Paragon composée de 56 nœuds et quelques résultats préliminaires de performances sont présentés. Ces résultats montrent notamment la dégradation de performance dûe à la mise en œuvre des mécanismes de tolérance aux fautes en comparant notre proposition à une mémoire virtuelle partagée standard sur la méme architecture.

Mots-clé : Mémoire partagée distribuée, système distribué extensible, tolérance aux fautes, retour arrière, point de recupération

1 Introduction

Distributed shared memory (DSM) systems implement a shared memory programming model on top of a distributed system without hardware support for shared memory (i.e. distributed memory multicomputers, networks of workstations). DSM is attractive from the programmer's point of view, since it simplifies data partitionning and load balancing, which are two of the toughest problems in parallel machine programming.

One of the main advantages of distributed systems is their scalability. They are thus very attractive for the computation of parallel applications requiring both long running execution times and powerful environments of computation. Nevertheless, as the number of components in these systems grows, the probability that a failure occurs also increases. This behaviour is unacceptable, especially when long-running applications are to be executed. As restarting a long-running application from its initial state must be avoided, it is desirable for a DSM to be recoverable and to allow an application to restart its execution from a state saved beforehand, when a failure occurs. In order to preserve the vocation of distributed systems, the implementation of such error recovery mechanisms must be done without sacrificing the system scalability.

Backward error recovery is a well-known fault tolerance technique [Lee & Anderson 90]. It avoids the need of restarting an application from its initial state in case of failure by using a previous image of the system to restart the execution. Most recoverable DSM assume the presence of a stable storage device to ensure the stability of recovery data. These stable storages are usually implemented with specific hardware devices such as stable memories [Banâtre *et al.* 93], or more conventional devices such as disks. The first solution is efficient but too expensive, especially for large scale systems, whereas the second one is cheap but may rapidly limit the system scalability. In this paper, we propose a backward error recovery implementation that avoids the use of specific hardware, still ensuring the stability of recovery data by using the replication mechanisms provided by a DSM system. Moreover, this recovery scheme integrates the management of recovery data in the management of current data by extending the standard coherence protocol used by a DSM. The paper also describes a first implementation of the proposed recoverable DSM on a 56 node Intel Paragon [Int 93]. The application reconfiguration, in particular the way to restart processes after a failure, is outside the scope of this paper.

The remainder of the paper is organized as follows. The next section presents our base DSM and failure model. Section 3 then describes the integration of recovery data management in the standard coherence protocol and the way to make a DSM recoverable. Section 4 presents our implementation and some measures we have made to show the performance degradation introduced by the fault tolerant mechanisms. It also studies the scalability of this proposal. Section 5 presents related works and finally, section 6 concludes with a summary of our proposal.

2 System Model

The system we consider is composed of a set of nodes connected by an interconnection network. Each node is composed of one or more processors and of a memory module. In the evaluation section, this system is an Intel Paragon but any other hardware platform could be envisaged.

This section first describes the base DSM and especially the coherence protocol which will be extended and then lists up the assumptions of our scheme.

2.1 Distributed Shared Memory

In a DSM, local memories are considered as large caches of the global shared address space. Pages are replicated on demand in the local memory of processors requesting them, and can be loaded either from disk or from another local memory. In this paper, for the sake of simplicity, we do not consider the swapping management and consider that the set of memories can contain all virtual pages.

Since several copies of the same page may exist in the system, coherence between these copies has to be maintained. Our DSM is managed by a static distributed manager [Li & Hudak 89] implementing a sequential consistency model. Coherence is maintained in the system by directories statically distributed among the nodes. This technique avoids the bottleneck inevitably created by a central manager in a large-scale distributed system. The coherence protocol uses a write-invalidate strategy.

For each page, we distinguish its *owner node* from its *manager node*. The owner of a page has a valid copy of the page; it is the only one to have a copy of the page and the page is accessed through a write operation. The manager of a page keeps a directory entry for that page containing coherence information related to it: its owner, its state (*Invalid, Shared or Modified-exclusive*) and its *copyset list* which contains the list of the nodes having a copy of the page in their local memory. The manager of a page is statically determinated by a dedicated function (for instance a modulo between the page's address and the number of nodes). When a page fault occurs, a request for the page is sent to the manager of the page which forwards the request to its owner. The manager of a page is also in charge of performing the necessary invalidations before a write access on a replicated page. The standard coherence protocol is depicted by the transition diagram of figure 1.

A page p on a node n may be in one of the following states:

- *Invalid*: the local memory of n does not contain a copy of p.
- *Shared*: the local memory of n has a copy of p that can only be read. Other copies of p may exist in *Shared* state in other local memories.
- *Modified-exclusive*: the local memory of n has a copy of the page that can be read or written. No other copy of the page exists in the system and n is the *owner* of the page.

The *Shared* and *Modified-exclusive* states respectively correspond to the *Read* and *Write* state in Li's terminology [Li & Hudak 89].