

Tolerating Node Failures in Cache Only Memory Architectures

Michel Banâtre, Alain Gefflaut, Christine Morin

▶ **To cite this version:**

Michel Banâtre, Alain Gefflaut, Christine Morin. Tolerating Node Failures in Cache Only Memory Architectures. [Research Report] RR-2335, INRIA. 1994. inria-00074341

HAL Id: inria-00074341
https://inria.hal.science/inria-00074341

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Tolerating Node Failures in Cache Only Memory Architectures

Michel Banâtre, Alain Gefflaut, Christine Morin

N° 2335

Août 1994

PROGRAMME 1



ISSN 0249-6399



Tolerating Node Failures in Cache Only Memory Architectures

Michel Banâtre*, Alain Gefflaut*, Christine Morin*

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

Rapport de recherche n° 2335 — Août 1994 — 28 pages

Abstract: COMAs (Cache Only Memory Architectures) are an interesting class of large scale shared memory multiprocessors. They extend the concepts of cache memories and shared virtual memory by using the local memories of the nodes as large caches for a single shared address space. Due to their large number of components, these architectures are particularly susceptible to hardware failures and so fault tolerance mechanisms have to be introduced to ensure a high availability. In this paper, we propose an implementation of backward error recovery in a COMA which minimizes performance degradation and requires little hardware modifications. This implementation uses the features of a COMA to implement a stable storage abstraction using the standard memories of the architecture. Recovery data are replicated and mixed with current data in node memories both of which are managed in a transparent way using an extended coherence protocol.

Key-words: scalability, multiprocessor architecture, shared memory, availability, backward error recovery, simulation

(Résumé : tsvp)

The work presented in this paper is partially funded by the DRET research contract number 93.34.124.00.470.75.01. This paper will also appear in the proceedings of SuperComputing'94, Washington DC, November 14-16, 1994.

*{banatre}{gefflaut}{morin}@irisa.fr

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 architecture extensible COMA tolérant les défaillances de nœuds

Résumé : Les architectures COMA (Cache Only Memory Architectures) sont une classe intéressante des architectures multiprocesseurs extensibles à mémoire partagée. Elles étendent les concepts de mémoires cache et de mémoire virtuelle partagée par l'utilisation des mémoires locales des nœuds comme caches de grande taille d'un espace d'adressage partagé unique. Compte tenu de leur grand nombre de composants, ces architectures sont particulièrement sujettes aux défaillances matérielles rendant nécessaire l'introduction de mécanismes de tolérance aux fautes pour garantir une haute disponibilité. Dans cet article, nous proposons la mise en œuvre d'un mécanisme de retour arrière dans une architecture COMA qui minimise la dégradation des performances et requiert peu de modifications matérielles. Cette mise en œuvre tire profit des caratéristiques inhérentes aux architectures COMA pour offrir une abstraction de mémoire stable en utilisant les mémoires standard de l'architecture. Les données de récupération sont répliquées et conservées avec les données courantes dans les mémoires des nœuds. Les deux types de données sont gérés de façon transparente par un protocole de cohérence étendu.

Mots-clé : extensibilité, architecture multiprocesseur, mémoire partagée, disponibilité, retour arrière, simulation

1 Introduction

Scalable Shared Memory Multiprocessors (SSMM) are thought to be a good solution to achieving the teraflops computing power needed by grand challenge applications such as climate modeling or humane genome. These architectures consist of a set of computation nodes containing processors, caches and memories, connected by a high-bandwidth low latency interconnection network. Scalability, achieved by the scalable interconnection network and the distributed main memory, allows a large number of processors to be used with a good efficiency. Shared memory provides a flexible and powerful computing environment. Two variations of these architectures have emerged: Cache Coherent Non Uniform Memory Access machines (CC_NUMA) [1, 18], which statically divide the main memory among the nodes of the architecture, and Cache Only Memory Architectures (COMAs) [14, 10] which convert the per node memory into a large cache of the shared address space, called an Attraction Memory (AM).

Due to their increasing number of components both CC-NUMA machines and COMAs have, despite an important increase in hardware reliability, a very high probability to experience hardware failures. Tolerating node failures is therefore very important for architectures used for long running computations. Fault tolerance becomes then mandatory rather than optional for large scale shared memory multiprocessor architectures.

In this paper, we propose a new solution to cope with multiple transient and single permanent node failures in a COMA. Our approach uses a backward error recovery scheme [17] where the replication mechanisms of a COMA are used to ensure the conservation and replication of recovery data in the AMs of the architecture. To implement this, an extended coherence protocol manages transparently both current and recovery data. This solution avoids the need for specific hardware and minimizes performance degradation by using the memories and the interconnection network to handle recovery point establishment. Other aspects of fault tolerance, such as detection and error confinement are out of the scope of the paper and in the remainder we assume a fault-free network and fail-stop nodes. Detection of faulty nodes is provided through time-outs on inter-node communications.

The remainder of this paper is organized as follows. Section 2, gives an overview of COMA machines. In Section 3, after introducing the principles of our solution, we describe how the coherence protocol is extended to tolerate node failures. Performance evaluation results obtained by simulation are given in Section 4 for an implementation of the protocol in a slotted ring COMA similar to a single ring KSR1 [10]. Section 5 concludes our presentation.

2 Cache Only Memory Architectures

CC-NUMA architectures and COMAs have similar organizations. They both have a distributed main memory and a scalable interconnection network. In contrast to CC-NUMA machines, COMAs convert the per-node memory into a huge cache of the shared address space by adding tags to lines in main memory. A consequence of this is that the location of a data item in the machine is totally decoupled from its physical address, and a data item is automatically migrated and replicated in the memories following the memory reference pattern of the executed application. From a fault tolerance point of view, this feature constitutes a clear advantage of COMAs over CC-NUMA architectures since memory items located in a faulty node can be re-allocated on functioning nodes transparently without modification to their physical address. This is the reason why we have chosen to investigate COMAs.

COMA usually use hierarchical organization to locate memory items on a miss. Directories at each level of the hierarchy maintain information about memory item copies located in a sub-hierarchy. Such an organization is used in the DDM [14] and KSR1 [10] architectures.

Basically, all COMAs use the same coherence protocol. This protocol can be simplified to four basic item states which change according to requests received from the local processor (*Pread/Pwrite*), or from remote nodes over the network (*Nread/Nwrite*). Figure 1 depicts the standard coherence protocol used by the AMs of a COMA architecture.

- *Invalid*: the local AM does not contain a valid copy of the item.
- *Shared*: the local AM has a read-only copy of the item.