Distributed Memory Multiprocessor Pdf Free

Hardware Fault Containment in Scalable Shared-Memory Multiprocessors

Dan Teodosiu, Joel Baxter, Kinshuk Govil, John Chapin*,
Mendel Rosenblum, and Mark Horowitz

Computer Systems Laboratory
Stanford University, Stanford, CA 94305
http://www-flash.stanford.edu

Abstract

Current shared-memory multiprocessors are inherently vulnerable to faults: any significant hardware or system software fault causes the entire system to fail. Unless provisions are made to limit the impact of faults, users will perceive a decrease in reliability when they entrust their applications to larger machines. This paper shows that fault containment techniques can be effectively applied to scalable shared-memory multiprocessors to reduce the reliability problems created by increased machine size.

The primary goal of our approach is to leave normal-mode performance unaffected. Rather than using expensive fault-tolerance techniques to mask the effects of data and resource loss, our strategy is based on limiting the damage caused by faults to only a portion of the machine. After a hardware fault, we run a distributed recovery algorithm that allows normal operation to be resumed in the functioning parts of the machine.

Our approach is implemented in the Stanford FLASH multiprocessor. Using a detailed hardware simulator, we have performed a number of fault injection experiments on a FLASH system running Hive, an operating system designed to support fault containment. The results we report validate our approach and show that in conjunction with an operating system like Hive, we can improve the reliability seen by unmodified applications without substantial performance cost. Simulation results suggest that our algorithms scale well for systems up to 128 processors.

1 Introduction

Scalable shared-memory multiprocessors are becoming an increasingly common computing platform. Several companies, including HP-Convex [20], Sequent [11], and Silicon Graphics [10], are shipping multiprocessor systems with configurations of up to a few hundred processing nodes.

However, current shared-memory multiprocessors are inherently vulnerable to faults: any significant hardware or system software fault will cause the entire system to fail. Unless provisions are made to limit the impact of faults, users will perceive a decrease in reliability when they entrust their applications to larger machines. This is an important problem for the viability of large-scale shared-memory multiprocessors as general-purpose compute servers.

Fault containment is a reliability technique widely used in distributed systems, in which the effects of a fault are limited to a small portion of a system [15]. In a computing system that provides fault containment, the chance of failure for a task depends only on the amount of resources that the task uses, not on the size of the entire system.

When running on a large-scale shared-memory multiprocessor, applications of small to moderate size will benefit from the increased reliability offered by fault containment. Since these applications only require a limited amount of resources, they will be protected from failures that occur in the parts of the machine they do not use. Fault containment can also provide benefits to large parallel applications that use a substantial fraction of the machine, if these applications are structured to cope with the loss of some of their resources.

Several challenging issues arise when one tries to implement fault containment on a shared-memory machine. The tight coupling provided by shared memory allows the effects of hardware faults to spread much faster and to more nodes than in a traditional distributed system. In addition, many of the fault containment techniques used in distributed systems (such as performing consistency checks on all incoming messages) appear to be prohibitively expensive given the low latency requirements of a high-performance memory system.

This paper describes one method for implementing fault containment in a scalable shared-memory multiprocessor without substantially reducing its performance or increasing its cost. Rather than relying on expensive techniques that mask the effects of data loss, our approach is based on limiting the impact of faults and running a distributed algorithm to bring the machine back to an operational state after a failure. We have added support to the FLASH multiprocessor [7][9] to confine the effects of most hardware and system software faults. Using the SimOS simulation environment [19], we have validated our approach by performing a number of fault injection experiments on a detailed simulation of the machine.

Achieving fault containment in a shared-memory multiprocessor requires careful design of both its hardware and its operating system. Current multiprocessor operating systems are unable to cope with the loss of any essential hardware resource, such as the failure of a processor or a memory board. This issue has been addressed in our previous work on the Hive operating system [3][18] that was developed in conjunction with the FLASH multiprocessor. This paper focuses on the hardware and firmware support required for an operating system such as Hive that provides fault containment. The experimental results reported in the paper show that, in conjunction with Hive, our implementation offers fault containment benefits to unmodified applications running on FLASH.

ISCA '97 Denver, CO, USA

*Author's present address: Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.