An Simple and Efficient Fault Tolerance Mechanism for Divide-and-Conquer Systems

Gosia Wrzesińska, Rob V. van Nieuwpoort, Jason Maassen, Henri E. Bal
Dept. of Mathematics and Computer Science
Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
{*gosia, rob, jason, bal*}*@cs.vu.nl*

Abstract

With the rapid development of grid technology, fault tolerance becomes an issue of crucial importance. Existing fault tolerance techniques, like checkpointing or message logging, have high overhead even if no fault occurs, and do not perform well in large scale, heterogeneous environments such as computational grids. We study if fault tolerance can be made simpler and more efficient by exploiting the structure of the application. More specifically, we study divide-and-conquer parallelism, which is a popular and effective paradigm for writing parallel grid applications.

The divide-and-conquer paradigm presents several advantages when implementing fault tolerance. In a divide-and-conquer application, function execution will always produce the same outputs if given the same inputs, a property also known as referential transparency. Exploiting this feature of the divide-and-conquer paradigm makes it possible to create a fault tolerance mechanism based on redoing the work lost in crashes of processors. Such a mechanism can have very low overhead, as no synchronization between processes is needed and no data need to be stored on stable storage. A number of such techniques have been proposed in the literature. However, the common problem of those techniques is redundant computation - work done by alive (i.e. still working) processors that has to be discarded and redone as a result of crashes of other processors.

We have designed a novel fault tolerance mechanism for divide-and-conquer applications that reduces the amount of redundant computation by storing results of discarded work in a global (replicated) table. These results can later be reused, thereby minimizing the amount of work lost as a result of a crash. The execution time overhead of our mechanism is close to zero. Our mechanism can handle crashes of multiple processors or entire clusters at the same time. It can also handle crashes of the root node that initially started the parallel computation.

We have incorporated our fault tolerance mechanism in Satin, which is a Java-based divide-and-conquer system. Satin is implemented on top of the Ibis communication library. The core of Ibis is implemented in pure Java, without using any native libraries. The Satin runtime system and our fault tolerance extension also are written entirely in Java. The resulting system therefore is highly portable allowing the software to run unmodified on a heterogeneous grid.

We evaluated the performance of our fault tolerance scheme on a cluster of the Distributed ASCI Supercomputer 2 (DAS-2). In the first part of our tests, we show that the execution time overhead of our mechanism is close to zero. The results of the second part of our tests show that our algorithm salvages most of the work done by alive processors. Finally, we carried out tests on the European GridLab testbed. We ran one of our applications on a set of six heterogeneous parallel machines (four different operating systems, four different architectures) located in four different European countries. After manually killing one of the sites, the program recovered and finished normally.

0-7803-8430-X/04/$20.00 ©2004 IEEE