Limited Optimism: A Method for Software Fault-Tolerant Operation

Anthony Joseph*
Large-Scale Parallel Software Group, MIT Laboratory for Computer Science
email: adj@lcs.mit.edu

1 Introduction

Advances in hardware technology have made it plausible to construct a parallel computer with hundreds of thousands or even millions of processors sometime in the next decade. Many technical obstacles must be overcome to build and make effective use of such a machine; of these, *reliability* seems the most difficult. Such systems could have mean time between failures (MTBF) on the order of a few hours or days and without mechanisms to limit the amount of work lost when a failure occurs, the range of problems that can be solved on such large machines will be limited to those that take relatively little time to compute.

Hardware approaches to fault-tolerant operation (e.g., improving component reliability or providing redundant components) produce limited improvements in overall system reliability and are very expensive. We believe that software techniques are a more cost-effective approach; they provide more flexibility in tailoring the level of redundancy and reliability to the needs of applications, and also make it easier to reconfigure machines dynamically when failures occur.

We present an overview of an software-based that we are developing to provide a low-overhead and scalable approach to hardware and communication fault-tolerance. The method is application-independent and requires no modifications to the application. All failures are hidden from applications using this method.

2 Application Independent Fault-Tolerance

The traditional pessimistic approach for application-independent software fault-tolerance assumes that failures are fairly frequent; thus, it trades fast failure recovery for slower normal processing. Pessimistic recovery methods allow a process to process an incoming message only when it has guaranteed that it can reconstruct its current state after a failure. This can delay the processing of messages, but after a failure a process can recover to the point immediately preceding the failure without involving any other processes.

Optimistic methods [1, 2] recognize that although failures do occur, they typically occur very infrequently. Thus, these methods emphasize normal processing performance over failure recovery. They do not delay the sending of messages; this means that state information may be lost after a failure. States at other processes that are dependent upon this lost information become invalid or "orphan" states. To detect these states at other processes, requires that state dependency information be included with messages. Using this information during failure recovery, it is possible to determine which states have been lost and which states at other processes are dependent on those lost states and need to be rolled back.

The problems with purely optimistic methods are that they: can involve all processors in failure recovery, this does not scale well since recovery could take long enough that another processor fails; do not provide a reasonable upper bound the amount of information that may be lost after a failure, as a result useful forward progress could occur very slowly. We are developing an optimistic recovery, called limited optimism, that addresses both of the above problems.

3 Limited Optimism

We base our method on the work by Strom and Yemini, and also by Sistla and Welch [1, 2]. Each process has a primary and at least one backup, running on separate processors. The backup receives periodic checkpoints from the primary, as well as logged messages. The messages are stored in memory, since the probability that two processors fail is substantially smaller than the probability that one will fail. When a failure of the primary is detected, the backup takes over, creates a new backup to replace the primary, and starts a recovery algorithm. After recovery is complete, the backup starts running. If the backup fails, a new backup is created to take its place, and the primary continues running.

*Supervised by Professor William E. Weihl. Supported in part by the National Science Foundation under Grant CCR-8716884, by the Defense Advanced Research Projects Agency (DARPA) under Contract N00014-89-J-1988, by an equipment grant from Digital Equipment Corporation, and by an IBM Graduate Fellowship.

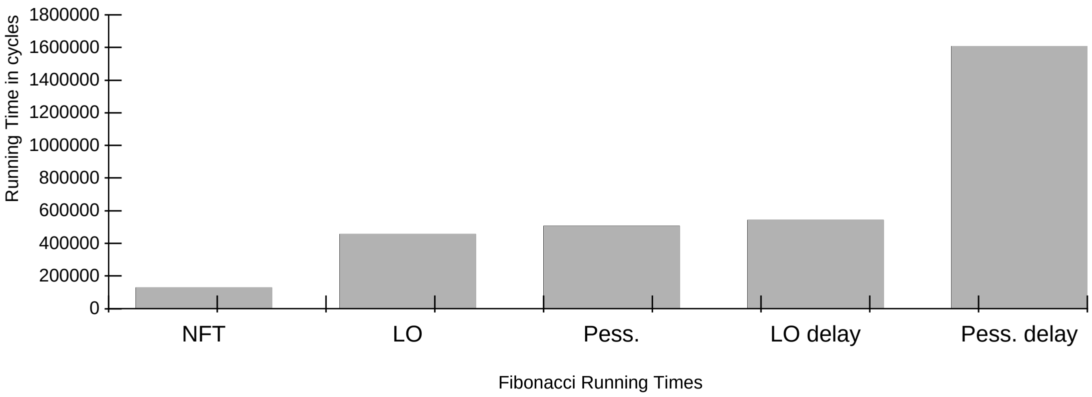

Figure 1: SOR Failure-free performance

To solve the problems with existing optimistic methods discussed earlier, we make the following extensions and modifications:

- Purely optimistic methods allow dependency chains of arbitrary length to develop without forcing data to be logged to a backup. The failure of one processor with unlogged data could cause an unbounded number of processes to rollback. Instead of recording just the *set* of dependencies for each process, our method records the maximum length of a message chain involved in each dependency and requires that their length is bounded by k. The delays introduced by limiting chain length should be reasonably small compared to pessimistic systems, since a process can send a message and the receiver can start processing it right away without waiting for data to be logged by the sender. If k is bigger than one, the receiver of the first message can send messages to other processes, which can send further messages, and so on up to a chain of length k.
- To avoid broadcasting to all processes that a failure has occurred, each process keeps track of its "friends," which are the processes with which it communicates frequently. A process can send a message to a friend without waiting, but before it can send a message to a non-friend, it must inform its backup that the message is being sent. This allows the backup to know which processes might be directly affected by a failure. The "friends" mechanism also keeps track of a processes k-level friends (those that depend on it with a chain of length up to k). In many applications, the set of processes that can depend on a particular process through a dependency chain of length less than k is small and does not change during the computation. In such applications, there should be almost no overhead resulting from maintaining the set of k-level friends. Also, we can use the friends information to narrowcast information about states that are logged to the affected processors.
- We have designed an incremental recovery method that requires only the processes affected by a failure to be involved in recovering from it. The method works by piggybacking on messages information about both direct dependencies (as in Sistla and Welch's method) and indirect dependencies involving dependency chains up to some fixed length d. Increasing d reduces recovery time, but increases overhead during normal operation. Thus, the method provides simple control over the tradeoff between the overhead during normal operation and recovery time.

4 Performance

The initial results show that the failure-free performance of limited optimism is slightly better than a pessimistic method. If the backups store logged messages to stable storage, then the performance of the limited optimism method is substantially better than the pessimistic method [4]. This difference is visible in Figure 1.

References

[1] Robert E. Strom and Shaula Yemini. Optimistic recovery in distributed systems. *ACM Transactions on Computer Systems*, 3(3):204–226, August 1985.

[2] A. Prasad Sistla and Jennifer L. Welch. Efficient distributed recovery using message logging. In *Proceedings of the 8th ACM Symposium on Principles of Distributed Computing*, pages 223–238. ACM, August 1989.

[3] David B. Johnson and Willy Zwaenepoel. Sender-based message logging. In *Proceedings of the 17th International Symposium on Fault-Tolerant Computing*, pages 14–19. IEEE Computer Society, July 1987.

[4] Anthony Joseph and William Weihl. Limited optimism: A method for fault-tolerant operation. Parallel Software Group Design Note *forthcoming, MIT Laboratory for Computer Science, 1991.*