

Controlling Recovery Time with Message Logging

Kuo-Feng Ssu

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Bin Yao and W. Kent Fuchs

School of Electrical & Computer Engineering
Purdue University
West Lafayette, IN 47907

1 Introduction

Numerous analytical approaches have been proposed to determine the optimal checkpoint interval for minimizing the total execution time of an application in the presence of failures. These mathematical solutions are often not applicable due to the lack of accurate data on the probability distribution function of failures [1]. Current checkpoint libraries typically require application users to define a fixed time interval for checkpointing [2]. Since the checkpoint interval implies the approximate maximum recovery time for single process applications, users who do not have accurate information on the mean time to failure (MTTF) determine the fixed checkpoint interval based on their preferred maximum recovery time.

The maximum recovery time can be much smaller than the checkpoint interval when message logging is used [3]. Due to the faster recovery time, the checkpoint interval may be set smaller than needed and unnecessary execution overhead introduced. In this abstract, a new checkpointing mechanism for message logging is developed to bound the user-defined recovery time. This mechanism avoids the ambiguity of the fixed checkpoint interval for message logging. In addition, users can control the upper bound on the recovery time more precisely using the checkpointing mechanism.

The checkpointing mechanism has been implemented and evaluated with a receiver-based message logging on a wireless mobile network. With 0.2% average overhead, the checkpointing mechanism precisely maintained the user-defined fixed maximum recovery times for several traces with varying computation patterns. The checkpointing mechanism not only incurs low overhead and avoids unnecessary checkpointing, but also improves execution time performance.

This research was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract DABT 63-96-C-0069, and in part by the Office of Naval Research under contract N00014-97-1-1013.

2 Estimation of Recovery Time

In our approach, checkpointing is triggered by a maximum recovery time instead of by a fixed checkpoint interval. The difference between execution time and recovery time is an important aspect of bounding the recovery in message-passing systems. During normal execution, processes exchange computation results by sending and receiving messages. Processes can be forced to stop further computation because the needed results from other processes have not arrived. During recovery, the messages do not have to be reproduced because they have been logged either at the receiver or sender process. The failed process accesses messages from logs directly so the completion time of the program is reduced. Therefore, the main difference between the execution time and recovery time for message logging is due to the time necessary to obtain the messages for further execution. The fixed maximum recovery time can be maintained as long as the execution delay and the message access time can be measured. The time to access messages can be obtained before actual execution but the execution delay must be measured during execution.

Instead of using a periodical polling technique, our checkpointing mechanism utilizes a two-timer implementation to reduce performance overhead. A *message timer* measures the total time for sending requests and waiting for replies. The message timer excludes the time to obtain the messages from logs. A *checkpoint timer* is set to the maximum recovery time defined by users. When the checkpoint timer expires, it checks the value of the message timer. If the value is larger than zero, the checkpoint timer is adjusted to the value of the message timer and the message timer is reset to zero. Eventually, the checkpoint timer will detect that the message timer is zero. A checkpoint is created at this moment. After the checkpoint is complete, the checkpoint timer is set to the user-defined recovery time again. A similar implementation can also be used for sender-based logging.

Table 1: Timer Overhead

Trace Program	Normal	Normal with Timers	
	Exec Time (sec)	Exec Time (sec)	Overhead (%)
T1	4963.3	4969.7	0.12%
T2	4227.0	4228.7	0.04%
T3	3854.0	3863.3	0.24%

3 Experimental Results

The performance of our approach was measured in a wireless mobile environment. Two Pentium II 300MHz PCs equipped with 2Mbps Lucent WaveLAN wireless interface served as mobile hosts. The PCs each had 256MB memory and ran Windows NT 4.0. A Sun Sparc 20 workstation with 320MB memory running Solaris 2.6 served as the mobile support station. The workstation used Lucent WavePOINT-II to communicate with the PCs. The workstation was connected to a file server, a dual-processor Sun Ultra Sparc 2 with 512MB RAM running Solaris 2.6, by a 155Mbps ATM network.

An independent checkpointing protocol with receiver-based pessimistic logging was used for our experiments [4]. There were two processes simulating web browsing behavior using pre-generated traces¹ on each mobile host. The processes sent requests and replies to each other. Each message was sent to the mobile support station through the wireless channel and then forwarded to the destination. The mobile support station was also responsible for storing message logs and checkpoints as requested by the processes on mobile hosts.

The experimental results show that measuring the time for sending requests and waiting for replies did not cause significant overhead (see Table 1). The average overhead for the timers was less than 0.2%. In our implementation, five lines of Java code were inserted for each `write-read` pair.

The procedures for measuring recovery time are briefly described as follows. During failure-free execution, the number of events executed between two checkpoints was recorded. During recovery, a timer was started when the program began to load the latest checkpoint and stopped after the same number of traces was completed. The results in Table 2 demonstrate that the checkpointing mechanism precisely maintained the user-defined maximum recovery time. The recovery times were all approximately five minutes for three traces that had different computation and communication patterns. In contrast, the fixed checkpoint interval approach did not accurately bound the recovery time. With a five minute fixed checkpoint interval, the maximum recovery time ranged from 49 seconds to 119 seconds.

¹The details for generating the traces can be found in [4].

Table 2: Maximum Recovery Time

Trace Program	Fixed Interval	Our Checkpoint Mechanism
	Recovery Time (sec)	Recovery Time (sec)
T1	119.3	306.0
T2	73.3	303.7
T3	48.7	297.3

Table 3: Comparison of Execution Time

Trace	Normal	Fixed Interval		Our Mechanism	
	Exec Time (s)	Exec Time (s)	No. of Ckps	Exec Time (s)	No. of Ckps
T1	4963.3	5174.7	16	5035.7	5
T2	4227.0	4446.3	14	4268.7	3
T3	3854.0	4055.0	13	3891.0	1

Table 3 shows the simulated number of checkpoints and the execution times for the three traces using both a five-minute fixed checkpoint interval and our mechanism for maximum recovery time. The checkpoint size for all traces was 1 MB. Our approach prevented excessive checkpoints during execution and achieved better performance.

4 Conclusion

The checkpointing mechanism for message logging described in this abstract inserts checkpoints dynamically to provide an approximate upper bound on recovery time defined by users. This mechanism provides enhanced control of the recovery time compared to fixed checkpoint intervals.

The mechanism has been implemented and evaluated with a receiver-based message logging protocol on a wireless network. The results demonstrated that the mechanism preserved accurate fixed maximum recovery times for a variety of traces and incurred low overhead. The approach prevented unnecessary creation of checkpoints and improved the total execution time.

References

- [1] J. S. Plank and W. R. Elwasif, "Experimental Assessment of Workstation Failures and Their Impact on Checkpointing Systems," *Proceedings of IEEE Fault-Tolerant Computing Symposium*, pp. 48–57, June 1998.
- [2] Y.-M. Wang, Y. Huang, K.-P. Vo, P.-Y. Chung, and C. Kintala, "Checkpointing and its Applications," *Proceedings of IEEE Fault-Tolerant Computing Symposium*, pp. 22–31, June 1995.
- [3] G. Suri, B. Janssens, and W. K. Fuchs, "Reduced Overhead Logging for Rollback Recovery in Distributed Shared Memory," *Proceedings of IEEE Fault-Tolerant Computing Symposium*, pp. 279–288, June 1995.
- [4] B. Yao, K. Ssu, and W. K. Fuchs, "Message Logging in Mobile Computing," *Proceedings of IEEE Fault-Tolerant Computing Symposium*, June 1999.