

Design and Implementation of Data Version Management Strategy Foresight in STM

Ying Liua, Fuxiang Gaob, Xin Sun

*College of Information Science and Engineering, Northeastern University,
110819 Shenyang Liaoning, China*

aemail: liuying@ise.neu.edu.cn, bemail: gaofuxiang@ise.neu.edu.cn

Abstract

First of all, several data version management strategies in software transactional memory have been studied in this paper, and both advantages and disadvantages are analyzed. Then a novel data version management strategy named Foresight in software transactional memory is put forward. Its design idea and detailed implementation is given. And its performance is tested in RSTM. The results show that as a data version management strategy Foresight has lower abort rate than any other data version management strategy. As we known, the low abort rate is beneficial to improve system performance. So the data version management strategy Foresight can help improve the whole performance of the software transactional memory system.

Keywords: Software Transactional Memory; RSTM; Data Version Management

Introduction

Referring to the concept of transaction in database, transactional memory is used to solve the problems in the parallel processing instead of the locks and semaphores. Each thread that can be executed in parallel would be processed as a transaction, and this can reduce the complexity of programming. Transactions here have three characteristics, atomicity, serializability, and isolation. Among them, atomicity means that a transaction should be executed completely and committed, or aborted and recovered back to the state before executing. This characteristics have played the same role in parallel processing as locks. According to this rule, in its execution processing, the thread should announce the beginning of the transaction, execute a series of operations, and commit this transaction. At present, transactional memory has become a new research issue in multi-core parallel processing.

An integral transactional memory system must have three key functions, data version management[1], conflict detection and conflict resolution Among the three functions, data version management is the basis of all functions. All the historical data for each stage should be saved completely to keep the system's atomicity. In this paper, we focus on the data version management strategies, and put forward a novel data version management strategy called Foresight. Its

design idea and implementation is given in detail. Finally, software transactional memory system, RSTM, is chosen as the experiment environment. And the experimental data have shown that using this adaptive data version management strategy abort rate can be lower, and the performance of the whole system can be improved.

Study on Data Version Management Strategies

Overview

For software transactional memory systems, Herlihy et al proposed the first dynamic software transactional memory system DSTM in 2003. The granularity for DSTM to access data is object. Firstly data is encapsulated into objects in a transaction. When the transaction accesses a data object, a corresponding data object replica will be produced. And change will happen to this replica. There is a special object to record the addresses of the new and old versions [2]. When the transaction is committed, the address of the data will be replaced by the replica.

Another typical software transactional memory system is RSTM [3]. There are many library implementations in RSTM. One of RSTM libraries, et, its granularity is word. So the value of the address should be exchanged into read/write set in words. And the read/write process can be executed according to the rule such as Eager or Lazy [4]. There are three combinations for the read/write process, Eager-Eager, Eager-Lazy, Lazy-Lazy [5]. The programmers can decide which strategy to set.

Although RSTM has all the data version management strategies and can switch according to the specific environment to improve efficiency. But it still needs the programmers to decide to choose which kind of data version management strategy. There is no dynamic data version management strategy in the software transactional memory system. Therefore we design an adaptive data version management strategy to improve the system performance according to the specific environment. And it has been implemented in RSTM.

Analysis

The data version management strategies Eager and Lazy both have advantages and disadvantages. For data version management strategy Eager, the modified data (new data) would be saved in the address where they are stored, while the unmodified data (old data) would be saved in the log. The transaction updates the values of variables directly, and commits. It deletes the records in the log when committing. This strategy can reduce the time delay. Once it aborts, the transaction will return to the state before transaction happened according to the records in the log. But in strategy Eager, all the write set of the transaction would be locked when committing or aborting. So the earlier they are stored, the longer the other conflict transactions will delay. The existing Eager data version management strategies like LogTM, OneTM, LogTM-SE, cannot reduce the isolation window to the minimum. They must write the old value into the undo-log in private space before updating the new value. And they must recover the old value when aborting. This leads to the extra operations of load and store when each transaction writing. The isolation window may be larger for

introducing the software method, which would introduce the conflict. Meanwhile, the operations of reading the old value and storing it into undo-log need two more caches, which would cause unnecessary consumption. So in the strategy Eager, storing the old values by software method will reduce the performance, and waste a lot of time before releasing the access permission.

The strategy Lazy must include the changed states in the hardware buffer, and submit them to the memory. This will bring two main problems.

The first one is committing delay. The old values are stored in the original addresses, while the new ones are stored in the buffer which is slower. The operation of commit is regular operation, but it's slower than the operation of abort. That will cause a lot of unnecessary overhead.

The other problem is the limit memory space that will make the new values overflow in the hardware buffer.

Data Version Management Strategy Foresight

The core of the data version management strategy Foresight is to track historical information of the transaction executing before and analyze these historical information by algorithms. Then the strategy decides which data version management strategy is suitable to execute the transaction by the selector.

There are four structures to decide to choose which data version management strategy.

The first structure is Transaction State Register (TSR). TSR is used to record the characteristic information of executing transactions currently. The part of TSR address is used to store the addresses of read/write set on memory. The parts of TSR commits and aborts are used to store the times of committing successfully and aborting for each transaction. The part of TSR strategy is used to store the data version management strategy for the transaction.

The second structure is History Execute Table (HET). HET is used to present the information of the transactions before. It includes several statistical data and address information used to store the addresses of read/write set on memory. The part of HET lem is used to present the execute strategy of the last committing instance of the transaction. The part of HET retc is used to track whether this address had been aborted. It can be computed by algorithm according to the parts of TSR commits and aborts.

The third structure is Write Version Management Selector (WVMS). When a transaction writes, WVMS will decide to choose which data version management strategy to execute the write operation. WVMS can set the strategy by the data from HET.

The fourth structure is Read Version Management Selector (RVMS). When a transaction reads, RVMS will decide to choose which data version management strategy to execute the read operation. RVMS can set the strategy by the data from HET.

The data transfer relationship among the structures is shown in Figure 1.

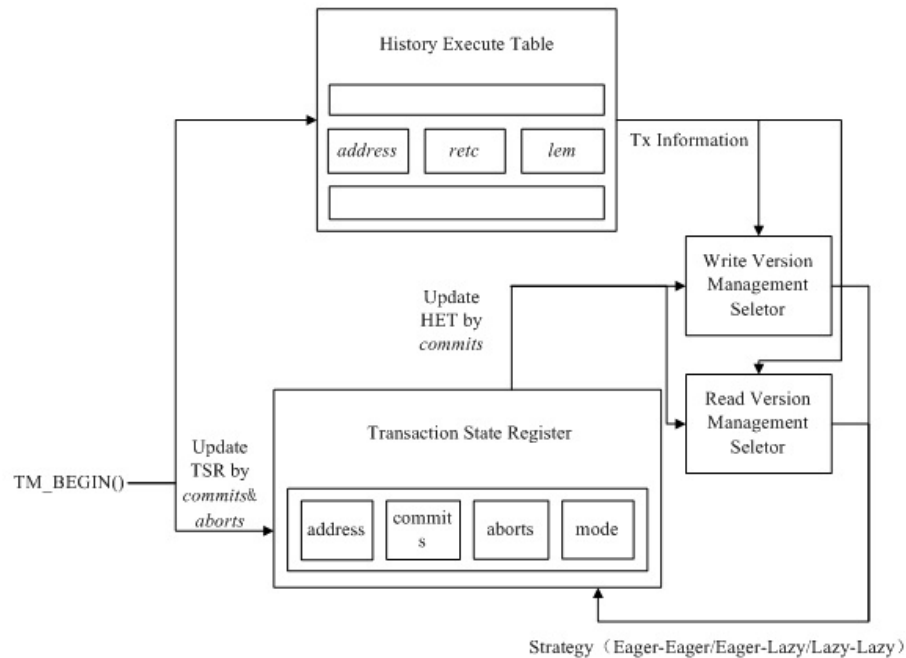


Fig. 1 Data Transfer Relationship among Structures

When a new transaction begins, VMS can compute the results for current transaction to decide which data version management strategy according to the *retc* of HET and the *commits* and *aborts* of TSR.

In the process of transaction executing, when writing, WVMS is called to decide which data version management strategy for this write operation by the information of HET and TSR. If the WVMS decides to choose Eager-Eager for current transaction write operation, the address will be locked which means the information of this address is being used and can't be modified by other read or write operations. The old value will be written into the undo-log, while the new value will be written into the memory. If the WVMS decides to choose Eager-Lazy for current transaction's operation writing, the address will be locked which means the information of this address is being used and can't be modified by other read or write operations. The new value will be written into the redo-log. The lock will be released until the transaction committing. If the WVMS decide to choose Lazy-Lazy for current transaction operation writing, the new value can be written into redo-log directly. While the old value will be remained in original address, and unchanged.

When the transaction commits or aborts, update the information *commits* and *aborts* by the specific condition. If a transaction is committed completely, HET should be updated by computing the *retc* using the *commits* and *aborts* of TSR.

Performance Evaluation

To test the performance of the data version management strategy Foresight, we choose a series of benchmarks in RSTM for strategy Foresight. These benchmarks include RBTree, LinkedList, Dlist, and LFUCache.

With data version management strategies, Eager-Eager (ee), Eager-Lazy (el), Lazy-Lazy (ll) and Foresight (a), the abort rates are shown in Figure2.

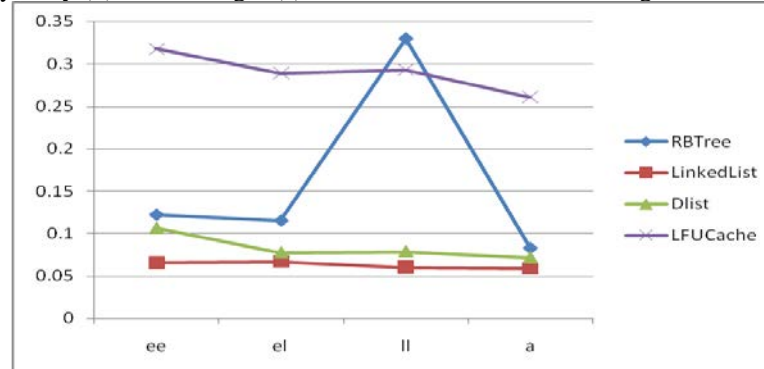


Fig. 2. Abort Rates for Different Strategies

The ability of pick the ball robot has reached theoretical calculation expected. The experiment started with no the institutions, the ball can not very well joint with pick cue, causing pick the ball dynamics change range is very large, and pick the ball height and the average of the theoretical calculation of the distance is smaller than the calculated assumption the height and distance.

After the ball add tape loading agencies, institutions force the ball the reverse spin, the ball close to pick the cue, pick the ball transfer fully energy to the ball, pick the ball effect is obvious stable. But it is found that the actual pick the ball after add tape loading agencies less than the theory calculated average distance.

Conclusion

The experiment shows that after running all kinds of benchmarks, the abort rate is reduced significantly by using data version management strategy Foresight, comparing with Eager-Eager, Eager-Lazy, Lazy-Lazy. The results prove that strategy Foresight has better performance than others. So using this adaptive data version management strategy can get lower abort rate than any other strategy. And the low abort rate is beneficial to improve system performance. So this would improve the system performance.

References

- [1] Liu Ying, Gao Fuxiang. Research of Conflict Detection Algorithm in STM[J]. Journal of Northeastern University, 2013, 34(6) 774-777.

- [2] Herlihy M, Luchangco V , Moir M, et al. Software transactional memory for dynamic-sized data structures [C]. Boston,MA: Proc 22nd Annual Symposium on Principles of Distributed Computing, 2003, 92-101.
- [3] Information on <http://www.cs.rochester.edu/research/synchronization/rstm/>
- [4] Lihang Zhao, Woojin Choi, and Jeff Draper. SEL-TM: Selective Eager-Lazy Management for Improved Concurrency in Transactional Memory [C], Proceedings of the 26th international parallel & distributed processing symposium, 2012, 1530-2075.
- [5] Zhao L, Draper J: On the Correctness of Mixing Lazy and Eager Version Management in Transactions [C], Parallel and Distributed Processing Symposium Workshops & PhD Forum, 2012, 2534-2537.