Adaptive Conflict Detection Algorithm Based on Rochester Software Transactional Memory

Ying Liu*, Junting Liu and Fuxiang Gao
College of Computer Science and Engineering, Northeastern University, Shenyang, China
*Corresponding author email: liuying@cse.neu.edu.cn

Abstract. When the transactional memory system detects conflicts, the more read-write addresses are, the higher the false positive rate of this algorithm is. This paper studies the problem and proposes a new signature based optimization algorithm, adaptive increase signature algorithm (AISA). The algorithm can calculate the saturation value of the number of read-write addresses, and dynamically adjust the size of bit string through the calculated saturation value, thus greatly reducing the false positive rate. The experimental results show that AISA can control the false positive rate at a low level on the basis of considering the space cost, and its effect is almost the same as that of 6 times of space, which will improve the performance of transactional memory system greatly.

1. Introduction
Inspired by the concept of transaction in database, J. Eliot B. Moss and Maurice Herlihy [1] have defined transactions in transactional memory. Transactional memory must have three functions including conflict detection, data version management and conflict resolution [2-4]. At present, the research on transactional memory [5,6] is mainly about hardware transactional memory (HTM), hybrid transactional memory (HyTM) [7] and software transactional memory (STM). HTM [8] was first proposed by J. Eliot B. Moss and Maurice Herlihy in 1993. Although its performance is very good, it also has many shortcomings, such as poor portability and heavy burden on programmers. The main reason for these is that the implementation of hardware transactional memory system depends too much on the cache size, and the cache size supported by hardware must be known when programming. The classic implementation of HTM is Stanford's TCC (transactional memory coherence and consistency). The earliest STM was implemented by Shavit and Touitou in 1995 by using the technology helping. It needed to allocate memory blocks to transactions, and each block is exclusive to transactions, and all input and output sets should be predicted. These limitations bring great difficulties to programming. In 2003, STM for dynamic sized data structures (DSTM) was proposed by Maurice Herlihy Nir Shavit Victor Luchangco and Michael Spear [9]. This design adopts dynamic data structure, so that programmers do not have to specify transactions for memory storage units. In 2006, Rochester STM (RSTM) implemented by C++ was proposed by the research group of the University of Rochester. RSTM adopts non-blocking data structure, so as to avoid deadlock and priority inversion caused by lock mechanism. This paper is mainly based on RSTM. Signature accelerated transactional memory (SigTM) is a classical hybrid transactional memory system. SigTM not only ensures the isolation of transaction and running environment, but also effectively reduces the system overhead, which has been widely concerned. In the architecture of SigTM, hardware signature is used to save read-write sets. Signature [10] in SigTM is also implemented based on Bloom filter.
2. Adaptive Conflict Detection Algorithm-AISA

Based on the idea of Bloom Filter [11], a conflict detection algorithm based on Signature is implemented to improve the existing conflict detection algorithm. This algorithm maps the read-write address set into a bit array through a number of different hash functions, which we call the bit array Signature. This mechanism can store an infinite number of read-write addresses on a limited bit string. The conflict detection algorithm based on Signature mainly includes True Bloom, Hash Bloom, Cuckoo Bloom, etc. Most of these algorithms focus on the saving storage space, while ignoring the problem of false positive rate. False positive refers to the concurrent transactions in the transactional memory system, which is judged to be a conflict when there is no conflict in the actual situation. Hash Bloom and Cuckoo Bloom is improved on the basis of True Bloom, although reduce the false positive rate relatively. But they did not solve the problem of high false positive rate when the address set is large, the algorithm complexity is higher, and the system has much overhead [12]. Therefore, the conflict detection algorithm based on Signature of fixed size bit string can be improved.

Analysis of benchmark showed that the sizes to read-write set of most transactions are in a relatively concentrated area, such as the size of most write sets don’t more than 30KB, and the size of read set can reach 350KB. The scope of the read-write set is difficult to predict in advance because it is related to the actual program. In this case, the problem for fixed size Signature will be obvious. Less elements waste storage while more elements increase false positive rate. So the size of the bit string should be adjusted according to the running environment.

Due to the difficulty of determining the appropriate length for bit string, it is unnecessary to overburden the system with too small or too large elements. A conflict detection algorithm by increasing length of bit string adaptively is designed in this paper, and the size of the bit string can be adjusted dynamically according to the number of read-write set in actual running environment in order to solve the problem mentioned above [14]. The algorithm can save storage space and keep false positive rate at lower levels.

In this paper, for research on false positive rate influenced by memory address of a Signature, the number of false positive rate of True Bloom should be estimated. In order to simplify the model, before the estimate, assumptions \(k \times n < m\), and the hash function is completely random. The elements in the set \(A = \{x_1, x_2, \ldots, x_n\}\) is mapped by \(k\) hash functions mapped to a bit array with \(m\) bits. \(1/m\) is the probability that in a mapping a bit is chosen by arbitrary hash function. \((1 - 1/m)\) is the probability that in a mapping a bit is not chosen by the hash function. Set \(A\) needs to complete all the mappings by \(k \times n\) times, so the probability of a bit that is not chosen after \(k \times n\) times mapping is \((1 - 1/m)^{k \times n}\). The probability that one of \(V\) is still 0 after that set \(A\) is mapped into \(V\) with \(m\) bits by the \(k\) hash functions is \(p = (1 - 1/m)^{k \times n} \approx e^{-k \times n/m}\). By using the formula for conversion, according to \(\lim_{x \to \infty} (1 - 1/x)^{-x} = e^{-1}\), we can get \(p = e^{-k \times n/m}\). Let \(\hat{\lambda}\) represents the probability that one of \(V\) is still 0 after that set \(A\) is mapped into \(V\) completely, \(E(\hat{\lambda}) = p\). So \((1 - \lambda)^k\) is the probability that one of \(V\) is still 1 after that set \(A\) is mapped into \(V\) completely. \((1 - \lambda)^k\) can approximately represent the probability that the one is all 1 after \(k\) hash mappings, and it can be approximate the false positive rate in True Bloom. After the derivation and calculation of the formula, false positive rate can be calculated as \((1 - \lambda)^k \approx (1 - p)^k \approx (1 - p)^k\). \(p\) and \(p\) are substituted into the upper formula, get the formula \(f = (1 - (1 - 1/m)^{k \times n})^k = (1 - p)^k\), \(f = (1 - e^{-k \times n/m})^k = (1 - p)^k\). Comparing \(p\) and \(f\), \(p\) and \(f\) is convenient in analysis. In True Bloom, the formula (1) of false positive rate is derived[15].

\[
f = (1 - e^{-k \times n/m})^k
\]
Where $f$ is the false positive rate, $k$ is the number of hash functions used in the address mapping, $m$ is the length of the bit string used in True Bloom, and $n$ is the number of elements to be stored. The formula (2) can be derived.

$$n = -m \times \ln(1 - f^{1/k}) \frac{k}{k}$$

(2)

True Bloom maps elements to be stored to bit string Signature depends on multiple hash functions. In experiment $k$ is 3, and the data is calculated. By analysing different bit string length $m$, $f$ correspondence relation between address number $n$ and false positive rate $f$ in different situations with different length of bit string. As shown in Figure 1, random address number has an impact on false positive rate.

![Figure 1. The relationship between the false positive rate and the number of addresses.](image)

It can be observed that there is no regularity between address increment and rate of false positive rate increasing through relation diagram of the address number and false positive rate. We guess the false positive rate must be influenced by the number of 1 in the bit string, and we call the proportion of the number of 1 in the bit string to the length of the bit string saturation. And this will be verified by formula. $p$ and $f$ are used for calculation, $f = (1 - e^{-ln/m})^k = e^{k \ln(1 - e^{-ln/m})}$. Let $g = k \ln(1 - e^{-ln/m})$, and that is $f = e^g$. Minimize $g$, and $f$ is also minimum. According to $p = e^{-ln/m}$, $g = (-m/n) \ln(p) \ln(1 - p)$. The formula function curve is shown in Figure 2.

![Figure 2. Function curve of formula.](image)

From the figure of function curve of formula $g = (-m/n) \ln(p) \ln(1 - p)$, we know that when $p = 1/2$, $g$ is the minimum value. Because of $p = e^{-ln/m}$, $g$ is the minimum value when $k = m/n \ln 2$. At this point, the minimum false positive rate $f$ is approximately equal to $(1/2)^k \approx (0.6185)^m/n$ that map all the elements of set $A$ into $m$-length bit array $V$ fully. In addition, $p$ is the probability that the element of $V$ is still 0 when $A$ is fully mapped into $V$, and $V$ has the lowest false positive rate because of $p = 1/2$. That means
false positive rate is lowest when half of 0 and the other half is 1. In other words, if you want to maintain a low false positive rate, it's best to keep half of V empty. The proportion of that element value is 1 in the bit array is defined as the saturation of an array, which means that he false positive rate will keep a lower value when the saturation is 50 percent. Then we'll implement Adaptive Increase Signature Algorithm in RSTM system according to the running environment.

The element insertion flowchart is shown in Figure 3. When inserting an element, the returned result is judged according to the saturation. If true is returned, it will be inserted into a new bit string; if it is returned to false, it will be inserted into the original string.

![Flowchart of inserting elements.](image)

3. Experiment

RSTM is the classic software transactional memory system. There are multiple STM library implementations in RSTM, and a variety of benchmarks with various features. RingSW \[15\] is one of the STM library implementations. RingSW stores the read-write address set with Bloom Filter, and it will publish its Bloom Filter globally when the transaction commits. Transactions can detect conflict quickly by intersecting their read-write sets. Therefore, adaptive increase bit string algorithm is modified based on RingSW. In order to verify the adaptive increase bit string conflict detection algorithm AISA can better adapt to the situation with uncertain number of addresses. Multiple benchmarks are adopted in this experiment. The algorithm is tested in the case of different number of threads at the same time to verify whether it is optimized for most programs. AISA and True Bloom are tested by Monte Carlo Method. In the simulation experiment, the value of C is 30, and the experimental results are shown in Figure 4 and Figure 5.

In Figure 5, the saturation curve diagram of the current used bit string is displayed, which shows the expansion process of Signature for 6 times during the experiment. False positive rate has remained at a low level using AISA for expansion. With the number of memory addresses increasing, for the True Bloom which Signature size is 1 KB, false positive rate will reach an unacceptable level. From Fig.4 True Bloom which Signature size is 6 KB can also make the false positive rate is low. But if the address continues increasing, its false positive rate is also rising. And it is clearly not the right thing to choose a fixed large Signature, which will cause waste of memory address.
The experimental results show that AISA is similar in the less number of addresses as the algorithm of fixed-size bit string. With the number of addresses increasing, the false positive rate of fixed-size Signature increases significantly, but false positive rate of AISA can be maintained at a lower level. The more the number of addresses is, the more obvious the performance is. In conclusion, AISA can better adapt to the un-certainty number of addresses. While maintaining the false positive rate at a lower level it can utilize the space reasonably. AISA solves the problem of that false positive rate is too high in the conflict detection algorithm based on the fixed-size bit string. In both high and no matter on high or low concurrency situations, AISA conflict detection algorithm can reduce the transaction abort rate effectively. However, due to the contention of system resources, the results of AISA in high concurrency and low concurrency environment are not very different. But, AISA reduces the abort rate to a certain extent in high concurrency and low concurrency environments.

4. Conclusion
The false positive rate is an important index of conflict detection algorithm in transactional memory system. This paper proposes a new signature based optimization algorithm, adaptive increase signature algorithm (AISA). The algorithm can calculate the saturation value of the number of read-write addresses, and dynamically adjust the size of bit string through the calculated saturation value, thus greatly reducing the false positive rate. The experimental results show that AISA can control the false positive rate at a low level on the basis of considering the space cost, and its effect is almost the same as that of 6 times of space, which will improve the performance of transactional memory system greatly. In the future, we will study the impact of AISA on the abort rate to further improve the system performance.

Acknowledgments
This paper is supported by Key Research and Development Program of Liaoning Province under Grant No. 2019JH2/10100019, and "the Fundamental Research Funds for the Central Universities" under Grant No. N181606001, N2016011 and N2024005-1
References

[1] Herlihy Maurice and Moss J.Eliot B Transactional memory Architectural support for lock-free data structures Proceedings of the 20th Annual International Symposium on Computer Architecture p 289-300

[2] Xiaoqun Wang Zhenzhou Ji Chen Fu and Mingzeng A Review of Hardware Transactional Memory in Multicore Processors Information Technology Journal (vol 8) chapter 7 p 965-970

[3] Peng Li Lunguo Xie Xiaoliang Zhang and Xinyan Xie Conflict detection via adaptive signature for software transactional memory 2nd International Conference on Computer Engineering and Technology p 306-310

[4] Johannes Schneider and Roger Wattenhofer Bounds on Contention Management Algorithms Theoretical Computer Science (vol 412) chapter 32 pp 4151-4160

[5] Wang Y An H Guo R Transactional memory architecture supporting speculative parallelization 2011 International Conference on Network Computing and Information Security p 89-93

[6] Tu X Fan X Jin H Transactional memory consistency: A new consistency model for distributed transactional memory 3rd International Joint Conference on Computational Science and Optimization p 364-368

[7] Xiang Li Jing Zhang and Junhuai Li Hardware/hybrid transactional memory 2011 International Conference on Computer, Mechatronics, Control and Electronic Engineering p 68-71

[8] Ljungkvist K Tillenius M Black-Schaffer D Using hardware transactional memory for high-performance computing 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum p 1660-1667

[9] Maurice Herlihy Nir Shavit Victor Luchango and Michael Spear The Art of Multiprocessor Programming (San Francisco: Morgan Kaufmann) p 12-35

[10] Sanchez D Yen L Hill M D Implementing signatures for transactional memory Proceedings of 40th Annual IEEE/ACM International Symposium on Microarchitecture p 123-133

[11] Swathy Priyadharsini P Arulanand Natarajan and A M Natarajan 2012 Optimization of Bloom Filter Using Evolutionary Algorithm Proceedings of the 2012 International Conference on Recent Advances in Computing and Software System p 86-90

[12] Bai, Lei, and Dong Min Li. Using Sample and Multilayer Compressed Counting Bloom Filter Algorithm to Realize Elephant Flows Identification Applied Mechanics and Materials (vol 651-653) p 2228-2232

[13] Michael F. Spear. "Fastpath Speculative Parallelization", Lecture Notes in Computer Science, 2010

[14] Kamala Subramaniam Alan Tharp and Arne Nilsson Location Updates In Cellular Networks Using Bloom Filters 30th Annual International Computer Software and Applications Conference (vol 02) p 3-9

[15] Spear M F Michael M M and von Praun RingSTM: scalable transactions with a single atomic instruction Proceedings of the 20th annual symposium on Parallelism in algorithms and architectures p 275-284