Quantifying and Reducing Execution Variance in STM via Model Driven Commit Optimization

Girish Mururu
School of Computer Science
Georgia Institute of Technology
girishmururu@gatech.edu

Ada Gavrilovska
School of Computer Science
Georgia Institute of Technology
ada@cc.gatech.edu

Santosh Pande
School of Computer Science
Georgia Institute of Technology
santosh@cc.gatech.edu

Abstract

Simplified parallel programming coupled with an ability to express speculative computation is realized with Software Transactional Memory (STM). Although STMs are gaining popularity because of significant improvements in parallel performance, they exhibit enormous variation in transaction execution with non-repeatable performance behavior which is unacceptable in many application domains, especially in which frame rates and responsiveness should be predictable. Thus, reducing execution variance in STM is an important performance goal that has been mostly overlooked. In this work, we minimize the variance in execution time of threads in STM by reducing non-determinism exhibited due to speculation by first quantifying non-determinism and generating an automaton that models the behavior of STM. We used the automaton to guide the STM to a less non-deterministic execution that reduced the variance in frame rate by a maximum of 65% on a version of real-world Quake3 game.

CCS Concepts • **Computing methodologies → Parallel programming languages**;

Keywords STM, Variance Reduction, Non-determinism

ACM Reference Format:
Girish Mururu, Ada Gavrilovska, and Santosh Pande. 2018. Quantifying and Reducing Execution Variance in STM via Model Driven Commit Optimization. In *Proceedings of PPoPP '18: 23nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Vienna, Austria, February 24–28, 2018 (PPoPP '18),* 2 pages. https://doi.org/10.1145/3178487.3178530

1 Introduction

The execution time varies significantly across runs for parallel programs. The execution of a critical section in various orders of thread-interleaving increase non-determinism and timing variance in execution of the application. Many applications, especially responsiveness oriented and soft-real time applications require a certain amount of repeatability of timing behaviors and in some cases prescribe a loosely acceptable bound on the amount of variance in execution time that could be tolerated [1]. When such applications are developed using transactional memory (TM), a parallel programming paradigm free of locks, the variance increases because of speculative execution. For example, in SynQuake, a version of Quake3 game, the variance in frame rate processing time was observed to be 2.17 seconds (31%), a value at which game play will be ridden with jitters.

In STM, several threads can simultaneously execute the code within a transactional section during which the STM checks for conflicts before commit which otherwise is aborted and retried. Therefore, a thread can abort any number of times before finally committing a transaction thus unbinding non-determinism, unlike non-speculative execution in which the maximum non-determinism is the distinct orders of n threads completing a critical section (i.e. $n!$). To account for unbounded aborts, we envision a model based on a probabilistic automaton that captures the state of concurrency of threads and determines the most common commit paths emanating from each state. Guiding the execution along only such high probability paths can lead to less non-deterministic, globally minimal abort execution with less variance and without sacrificing speculation. We reduced the variance in frame rate by up to 65% on a version of a real world gaming application, Quake3, without degradation in timing.

2 Methodology

The guided execution consists of four phases as follows:
Profile Execution: The application is executed multiple times to gather the sequence of aborts and commits during the run cycle of the application. This sequence is called the *transaction sequence (Tseq)*, which is parsed to create thread transactional states (TSS)– a tuple consisting of thread IDs coupled together with their respective transaction IDs of a commit and corresponding aborts.
Model generation: The transaction sequence also captures the transition from one state to another. Thus each state can transition to several states with corresponding probability called transition probability. Each state is related to its

neighbors with the transition probabilities by constructing a Thread State Automaton (TSA)– a finite state automaton that shows the transition of the system from one state to another with a certain probability thus modeling the application behavior.

Model Analysis: If the probability distribution for transition are approximately equal, then the model is deemed unfit for optimizing variance because the bias required for guiding the execution in a low variance path simply does not exist.

Guided Execution: The generated automaton that passed the analyzer provides a probabilistic model of execution. High probability transitions in the automaton indicate most common cases of execution.The model is used by the modified STM to guide the application towards a subset of high probability transitions by restricting transactions that wander to low probable states thus reducing variance. For example, in Figure 1, a thread transactional state excerpted from K-means executing on 8 cores, the current state $\{<a6>, <b7>\}$ can high likely transition to states in set $D = \{\{<a1>\}, \{<b3>\}, \{<a2>\}, \{<a4>\}, \{<b0>\}, \{<a5>\}\}$ and less likely to $\{<c7>\}$ and $\{<a6>, <b5>\}$. If transactions $a6$ and $c7$, not part of tuples of D, are the prevailing transactions, then these transactions are blocked and retried and execution can lead to distinct paths in two different scenarios. In one case, any of the thread-transaction in D will proceed and change the current state. For example, if $b3$ is committed, the current state would change to $\{<b3>\}$ from which $c7$ will be allowed to continue. However, if none of the state-changing transactions execute within k (set to 3) retries, to avoid deadlock transactions $a6$ and $c7$ will be allowed to progress.

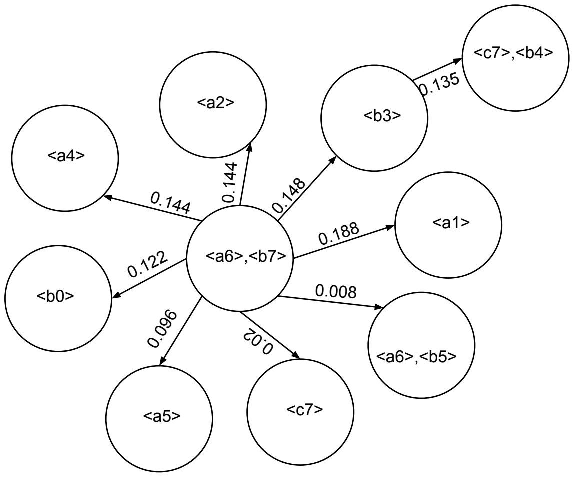

Figure 1. TSA excerpt from kmeans

3 Optimizing a real world game in LibTM

We experimented on 8 and 16 core machine using **SynQuake,** a 2D version of the real world Quake 3 game implemented using LibTM STM both of which is developed by [2], with 1000 players and two input quests namely, 4worst_case and 4moving, for training the model and two other quests– 4quadrants and 4center_spread6–for testing. Quests are specific areas in the map that attract players thus simulating a high interest area in the game play and the associated player movement patterns. The variance in processing frame rate was reduced by as much as 64.7% in guided STM as shown in Table 1. Because the model analysis shows higher scope of optimization for 16 threads than 8 threads, 16 threads comparatively show better reduction in variance. SynQuake experiences a speedup of almost 33% with 4quadrants and 12% with 4center_spread6 in case of 8 threads. However, in 16 threads, shown in Table 1, because of the increase in model data, speedup from reduced aborts is compensated by cache pressure exerted by the model.

Input	**FR 16(8)**	**AR 16(8)**	**Speedup 16(8)**
4Quadrants	58.26(7.09)	4(2.5)	-3(33)
4Center_spread6	64.7(18.09)	57.86(5.37)	3.3(12)

Table 1. Percentage Improvement in Frame Ratio Variance (FR), Abort Ratio (AR) and Speedup for 16 threads (8 threads)

4 Related Work

Several authors have focused on Contention Managers (CMs) with the intent to reduce aborts. A recent work DeSTM [3] guarantees completely deterministic execution of STM for easier debugging of STM based applications. However, none of these techniques, unlike this work, deal with the unique problem of high variance and non-determinism that exists in STM due to extremely speculative execution.

5 Conclusion

In this work, we propose a model driven approach that optimizes variance in execution time for applications developed with STM. We successfully demonstrated our technique by reducing variance in frame rate of a version of Quake3 by a maximum of 65% in 16 cores and 18% in 8 cores.

Acknowledgement

The authors gratefully acknowledge [2] for LIBTM and also the support of NSF grant CNS 1320752.

References

[1] Tushar Kumar, Jaswanth Sreeram, Romain Cledat, and Santosh Pande. 2007. A Profile-driven Statistical Analysis Framework for the Design Optimization of Soft Real-time Applications *(ESEC-FSE companion '07)*. ACM, 529–532. https://doi.org/10.1145/1295014.1295033

[2] Daniel Lupei, Bogdan Simion, Don Pinto, Matthew Misler, Mihai Burcea, William Krick, and Cristiana Amza. 2010. Towards Scalable and Transparent Parallelization of Multiplayer Games Using Transactional Memory Support *(PPoPP '10)*. ACM, 325–326. https://doi.org/10.1145/1693453.1693496

[3] Kaushik Ravichandran, Ada Gavrilovska, and Santosh Pande. 2014. DeSTM: Harnessing Determinism in STMs for Application Development *(PACT '14)*. ACM, 213–224. https://doi.org/10.1145/2628071.2628094