Journal of Database Management
Volume 33 • Issue 1

HyTM-AP Hybrid Transactional Memory Scheme Using Abort Prediction and Adaptive Retry Policy for Multi-Core In-Memory Databases

Hyeong-Jin Kim, Chonbuk National University, South Korea

Hyun-Jo Lee, Chonbuk National University, South Korea

https://orcid.org/0000-0002-1316-6822

Yong-Ki Kim, Vision College of Jeonju, South Korea

https://orcid.org/0000-0003-3828-5980

Jae-Woo Chang, Chonbuk National University, South Korea*

ABSTRACT

Recently, works on integrating HTM with STM, called hybrid transactional memory (HyTM), have been intensively studied. However, the existing works consider only the prediction of a conflict between two transactions and provide a static HTM configuration for all workloads. To solve the problems, the authors propose a hybrid transactional memory scheme based on both abort prediction and an adaptive retry policy, called HyTM-AP. First, the HyTM-AP can predict not only conflicts between concurrently running transactions, but also the capacity and other aborts of transactions by collecting the information of transactions previously executed. Second, the HyTM-AP can provide an adaptive retry policy based on machine learning algorithms, according to the characteristic of a given workload. Finally, through the experimental performance analysis using the STAMP benchmark, the HyTM-AP shows 12~13% better performance than the existing HyTM schemes.

KEYWORDS

Concurrency Control, HTM, Multi-Core In-Memory Database, STM, Transactional Memory

1. INTRODUCTION

Parallel query processing algorithms have recently been studied for efficient database management, such as spatial database, skyline computation and data warehousing (Andrzejewski & Boinski, 2015; Endres & Kießling, 2015; Bellatreche, Cuzzocrea, & Benkrid, 2012). For parallel algorithms, a lock is a well-known synchronization mechanism used for shared memory in multithreaded programs. However, developing software that correctly uses locks is notoriously challenging; therefore, transactional memory (TM) has been proposed as an attractive alternative to lock-based synchronization schemes (Herlihy & Moss, 1993). Unlike lock-based approaches, where programmers identify shared data and

DOI: 10.4018/JDM.299555

*Corresponding Author

specify how to synchronize concurrent access to it, the TM paradigm needs only to identify which portions of the code must be executed atomically, instead of considering how atomicity should be achieved (Pankratius & Adl-Tabatabai, 2011). TM can be classified into two categories: software transactional memory (STM), where transactions are implemented in the software, and hardware transactional memory (HTM), where transactions are implemented in the cache. STM tends to perform poorly at low or medium threads, as compared with fine-grained locking techniques. On the other hand, HTM has been developed to manage conflicts between transactions on multithreads using a cache coherence protocol.

HTM is very efficient, but has some restrictions. First, the size of a transaction is limited to the HTM. The Intel Haswell architecture is limited to the size of the L1 cache (256 KB). This implies that it is impossible to execute a database transaction as an HTM transaction. Second, when a transaction fails due to conflicts, the cache can be cleared by the operating system (OS) using context switching in the quantum cycle. Thus, large-sized transactions might be aborted at any time. Finally, because HTM has a best-effort nature, Intel's RTM (Restricted Transactional Memory) does not guarantee that transactions will ever commit in hardware, even in the absence of conflicts. Therefore, programmers must provide a software fallback path in case of a hardware transaction abort.

To overcome these limitations, hybrid transactional memory (HyTM), the integration of HTM with STM, has been intensively studied. HyTM processes read-only or short transactions using HTM, while long transactions are processed using STM. First, Dalessandro et al. (2011) proposed a hybrid version of the efficient NOrec STM (Dalessandro, Spear, & Scott, 2010), called Hybrid NOrec. The NOrec STM requires minimal instrumentation to ensure the consistency of active transactions for validation. When a transaction fails during its execution using HTM, the Hybrid NOrec goes into a software-based fallback path. Second, Calciu, Gottschlich, Shpeisman, Herlihy, & Pokam (2014) proposed a hybrid transactional memory (Invyswell) that uses Intel's RTM as the HTM and the modified version of STM (Dalessandro, Spear, & Scott). When a hardware transaction fails, Invyswell uses a software-based fallback path. This process is described in more detail in Section 2.3.

However, the existing HyTM schemes have the following limitations. First, they do not support the abort prediction of transactions. Even though two concurrent transactions have a high conflict probability, they attempt to execute transactions using HTM as many times (as a given number of retries) before forwarding transactions to STM or a serial execution. Thus, it causes the degradation of overall transaction execution performance. Second, they do not provide the optimal HTM parameter setting for various types of workloads. Third, they do not provide an efficient memory management technique because the memory pool for memory allocation/free is used based on a lock mechanism to process transactions. As a result, the efficiency of transaction processing degrades as the number of threads in the multi-core in-memory database increases.

To address this problem, we propose a hybrid transactional memory scheme based on both abort prediction and an adaptive retry policy for multi-core in-memory databases. In in-memory databases, there are various types of workloads that have multiple attributes and properties. For example, because a generic application generally contains short transactions that perform complex computations, the application requires a smaller number of retries. On the other hand, a K-means clustering algorithm requires a large number of retries for stable performance. Hence, a solution with a static configuration cannot yield the best performance across all possible workloads.

Thus, our study was undertaken and provides the following contributions:

- We propose a hybrid transactional memory scheme based on both abort prediction and adaptive retry policy (called HyTM-ARP) for multi-core in-memory databases.
- We provide an abort prediction algorithm that predicted not only conflicts between concurrently running transactions, but also the capacity and other aborts of transactions.
- We provide a near-optimal HTM configuration setting using an adaptive retry policy based on machine learning algorithms, according to the characteristics of a given workload.