long 트랜잭션 처리를 위한 HTM 기반 트랜잭션 분할 기법

HTM-based transaction splitting technique for processing long transactions

김 형 진, 마 현 국, 이 종 찬, 장 재 우**
전북대학교 컴퓨터공학과**

Hyeong-Jin Kim, hyeon-guk Ma, Jae-Woo Chang**
Chonbuk National Univ. Computer engineering

요약

최근 Hardware Transaction Memory (HTM)으로 트랜잭션을 처리하는 기술이 각광을 받고 있다. 그러나 HTM의 처리 실패 요인 중 하나는 캐시를 사용하여 트랜잭션을 처리하여 용량에 제한이 존재한다. 이러한 이유로 long 트랜잭션의 경우 용량을 초과하여 처리가 불가능한 경우가 빈번히 발생한다. 이를 해결하기 위해 본 논문에서는 long 트랜잭션 처리를 위한 HTM 기반 트랜잭션 분할 기법을 제안한다. 제안하는 기법은 먼저 HTM 으로 수행하여 캐시 용량을 초과하는 경우, long 트랜잭션을 다수의 트랜잭션으로 분할한다. 분할된 트랜잭션이 수행이 완료되면, 부분 커밋(commit)을 수행하고 이에 대한 정확성을 제공하기 위해 validation을 수행한다. 분할된 모든 트랜잭션의 수행이 완료되면 최종적인 커밋을 수행한다. 이를 통하여 기존 HTM 으로 처리하기 불가능한 long 트랜잭션을 속도가 우수한 HTM을 기반으로 효율적인 트랜잭션 처리가 가능하다.

I. 서론

기존의 lock 를 통해 트랜잭션을 처리하는 기술은 매우 높은 오버헤드를 요구하며, 트랜잭션 처리 성능이 낮은 단점이 존재한다. 이를 개선하기 위해 트랜잭서널 메모리 기술이 제안되었다. 그 중에서 HTM[1]은 cache를 통해 트랜잭션을 처리하기 때문에, 처리[1)]해야하는 트랜잭션의 크기가 cache 용량보다 크거나 정해진 clock cycle이 지나면 실패하는 한계점을 지닌다. 이를 개선한 기법이 PART-HTM[2]으로 capacity abort가 발생하면 해당 트랜잭션을 작은 트랜잭션의 단위로 나눠 처리하는 기법이다. 그러나 해당 기법은 트랜잭션을 수동으로 나눠야하는 단점이 존재하며, 아울러 해당 기법으로 분할을 하여도 cache 용량보다 큰 경우 및 contention이 발생하는 경우는 처리하지 못하는 단점을 지닌다. 이를 해결하기 위해, 본 논문에서는 long 트랜잭션을 머신 러닝 기법[3]을 이용하여 다수의 단위로 분할하는 트랜잭션 분할 기법을 제안한다. 제안하는 기법는 머신 러닝을 활용한 분할 정책 및 Self-tuning 기법[4]의 최적retry 수를 사용함으로써 cache 용량에 최적화된 분할 트랜잭션을 제공한다. 아울러 contention이 많이 발생하는 경우를 고려하여 Seer[5]의 충돌 예측 기법을 적용한다.

1) 이 논문은 2016년 교육부와 한국연구재단의 지역혁신창의인력양성사업의 지원을 받아 수행된 연구임(NRF-2016H1C1A1065816). 또한 이 논문은 미래창조과학부 및 정보통신기술진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음(IITP-2016-R0113-15-0005).

II. 관련 연구

PART-HTM[2]은 capacity abort 발생한 트랜잭션을 분할하여 처리하는 기법이다. PART-HTM[2]은 크게 3가지로 구성된다. 첫 번째는 First-trial HTM으로 트랜잭션을 기본 HTM으로 수행하는 단계이다. 일반적인 트랜잭션의 경우 First-trial HTM으로 처리가 가능하다. 두 번째, 만약 First-trial HTM에서 capacity abort가 발생하면 Sub-HTM을 수행한다. Sub-HTM은 트랜잭션에서 분할할 크기를 정하고, 크기만큼 트랜잭션을 수행하고 commit을 수행한다. 마지막 단계는 Global Lock 단계이다. Sub-HTM으로도 트랜잭션이 처리되지 못할 경우, 이를 처리하기 위해 Lock을 획득하고 트랜잭션을 처리한다.

III. long 트랜잭션 처리를 위한 HTM 기반 트랜잭션 분할 기법

제안하는 long 트랜잭션 처리를 위한 HTM 기반 트랜잭션 분할 기법 (ML-HTM) 은 먼저 머신 러닝 기법을 이용하여 다수의 단위로 트랜잭션을 분할하며, 최적의 retry 를 계산하기 위해 Self-tuning[4] 기법을 사용하고, contention이 많이 발생하는 경우를 고려하여 seer[5]의 충돌 예측 기법을 적용한다.

첫째, 트랜잭션 분할 정책은 PART-HTM[2]의 수동적분할 방법을 개선하여 머신 러닝 알고리즘[3]을 적용하여 최적의 분할을 수행한다. PART-HTM의 경우 분할된 트랜잭션의 크기가 cache 용량보다 크면 capacity abort가 발생하여 Global Lock으로 처리한다. 분할 정책이 수동

적으로 수행되면 Global Lock으로 처리되는 트랜잭션의 수가 증가하고 이는 성능 저하의 원인이 된다. 따라서 머신 러닝 알고리즘[3]을 적용하여 분할된 트랜잭션을 효율적으로 처리가 가능하다.

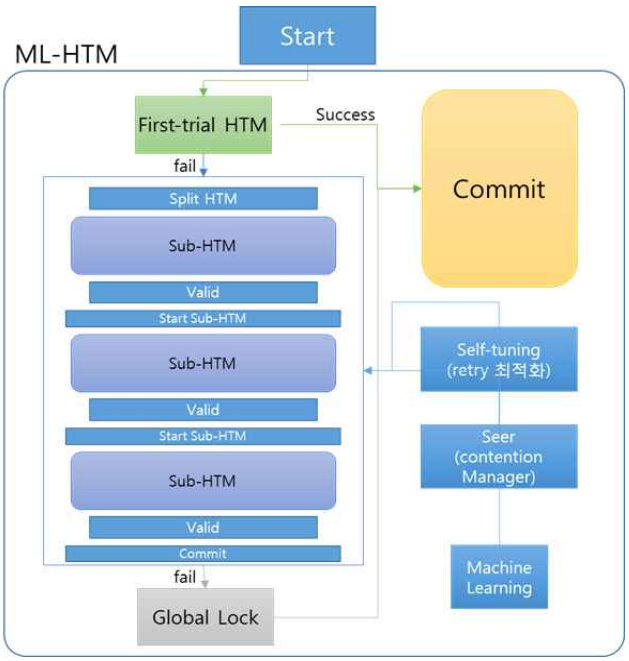
둘째, self-tuning[4]은 트랜잭션이 처리 시 abort가 발생하면, 최적의 retry 횟수를 설정하는 기법이다. PART-HTM 은 capacity abort에 대해서만 효율적 처리가 가능하고 retry 정책에 대해서는 고려하지 않고 있다. 따라서 머신 러닝 알고리즘을 활용한 최적의 retry 정책을 사용하여 보다 우수한 성능을 얻는다.

마지막으로 Seer[5]는 과거에 처리된 트랜잭션의 commit/abort 정보를 축적하고 이를 확률적으로 수치화하여, 현재 동작하고 있는 트랜잭션들의 충돌 확률을 예측한다. 이로써, HTM 상에서 충돌이 발생할 수 있는 원인을 사전에 차단하여 트랜잭션 처리 효율을 증대시킨다. PART-HTM 은 conflict abort에 대해서는 고려를 하고 있지 않기 때문에, 제안하는 기법에서는 Seer를 적용하여 contention이 많이 발생하는 경우를 처리할 수 있다.

세 가지 설계를 통해 conflict abort 및 capacity abort가 처리 가능한 long 트랜잭션 처리를 위한 새로운HTM 기반 트랜잭션 분할 기법 제안한다. 그림 1은 제안하는 ML-HTM의 전체 구조이다. 첫째, First-trial HTM을 수행하며, 이는 다른 thread와 동기화 문제를 해결하기 위해 Seer의 충돌 예측 기법을 사용한다. 둘째, First-trial HTM가 capacity abort가 발생할 경우 Self-tuning의 retry 수 정책만큼 재시도를 수행한다. Self-tuning의 retry 정책은 트랜잭션의 성질에 따라 retry 수를 조정함으로써 성능이 감소되는 것을 방지한다. 셋째, Self-tuning의 retry 수가 모두 소모되면 Sub-HTM을 수행한다. Sub-HTM은 트랜잭션의 read와 write의 수로 트랜잭션의 수를 파악하고 분할 처리하는 기법으로 기존의 PART-HTM의 수동적인 분할 방식이 아닌 머신 러닝을 통한 분할 정책을 수행한다. Sub-HTM 또한 하나의 트랜잭션으로 간주되어 처리되기 때문에 Seer의 충돌 예측 기법을 적용하여 Sub-HTM 처리 도중 conflict abort가 발생하는 것을 방지한다. 넷째, 하나의 Sub-HTM 수행이 완료되면 수행된 Sub-HTM 의 validation을 수행한다. validation 이 완료되면 Sub-HTM 은 부분 commit 이 완료된다. 마지막으로 모든 Sub-HTM 이 완료되고 validation 수행이 완료되면, 전체 트랜잭션에 대한 commit 이 완료된다. HTM 이 long 트랜잭션을 처리하지 못한다는 한계점을 극복하기 위해, 제안하는 기법은 분할 트랜잭션 처리 기법을 적용하고, conflict abort에 대한 효율적 처리를 위해 Seer의 충돌 예측기법을 적용함으로써 효율적인 처리가 가능하다.

Ⅳ. 결론

기존 PART-HTM[2] 은 capacity abort가 발생하는 트랜잭션에 대해 분할을 수행하고 분할된 트랜잭션을 HTM으로 처리한다. 그러나 해당 기법은 capacity abort에 대해서만 고려하며, 분할 정책이 수동적이라는 한계점을 지닌다. 따라서 본 논문에서는 conflict abort를 처리하는 seer[5], HTM의 retry 횟수를 조정하는 self-tuning[4] 및 머신 러닝[3]을 이용한 최적의 분할 정책을 지원하는 ML-HTM을 제안한다. ML-HTM은 머신 러닝[3]을 바탕으로 conflict abort, capacity abort, retry 정책을 관리함으로써 다양한 환경의 벤치마크를 효율적으로 처리가 가능하다.



▶▶ 그림 1. 제안하는 기법은 전체 구조

▌참 고 문 헌▐

[1] H. Chen et al. “Using restricted transactional memory to build a scalable in-memory database.” ACM Proceedings of the Ninth European Conference on Computer Systems, 2014.

[2] Mohamedin, Mohamed, et al. “Managing Resource Limitation of Best-Effort HTM.” IEEE Transactions on Parallel and Distributed Systems (2017).

[3] Nuno Diegues, and Paolo Romano. “Self-tuning intel transactional synchronization extensions.” 11th International Conference on Autonomic Computing (ICAC 14). 2014.

[4] Nuno Diegues, Paolo Romano, and Stoyan Garbatov. “Seer: Probabilistic scheduling for hardware transactional memory.” Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures. ACM, 2015.

[5] Pedregosa, Fabian, et al. “Scikit-learn: Machine learning in Python.” Journal of Machine Learning Research 12.Oct (2011): 2825-2830.