MMDB에서 캐시 친화를 고려한 최적의 레코드 저장노드 크기

김경석[O†], 이익훈[§], 이상구[§]
[†]LG이노텍㈜ 시스템연구소, [§]서울대학교 지능형 데이타베이스 연구실
gskimf@lginnotek.com[O], {ihlee, sglee}@europa.snu.ac.kr

Optimal Size of Record Storage Node in MMDB Considering Cache-consciousness

Gyeong-seok Kim[O†], Ig-hoon Lee[§], Sang-goo Lee[§]
[†]System R&D Center, LG Innotek
[§]Intelligent Database System Lab., Seoul National University

요 약

MMDB에서 성능에 가장 큰 영향을 미치는 부분은 로깅, 체크포인트, 락이다. 기존 연구는 이것들의 성능에 많은 초점을 맞추었으며, 이는 레코드 저장노드 크기에 관한 연구도 마찬가지다. 하지만 초고성능을 요하는 최신 MMDB 응용에서는 성능 조건을 충족시키기 위해 로깅, 체크포인트, 락을 포기하고 기본적인 레코드 저장구조 기능만으로 MMDB를 운용하기도 한다. 이 경우 레코드 저장구조 성능이 중요하게 된다. 이 논문에서는 실험을 통해 최적의 레코드 저장구조 성능을 보이는 레코드 저장노드 크기를 구한다. 그리고 실험 결과를 CPU 캐시 친화 관점에서 분석한다. 최종적으로는 MMDB 시스템 전체 성능의 최적화 관점에서, 실험 결과 얻어진 레코드 저장노드 크기를 검토한다.

1. 서 론

MMDB(Main Memory DataBases)는 금융, 이동통신 같은 응용분야에서 요구되는 초당 트랜잭션 처리 성능이 디스크 기반 DB로는 불가능함에 따라 등장하였다. MMDB에서 성능에 가장 큰 영향을 미치는 부분은 로깅, 체크포인트 등 백업과 관련된 부분이고, 다음으로는 락이다. 그래서 MMDB 관련 기존 연구는 로깅, 체크포인트, 락 성능에 관해 많은 관심을 가져왔다.

이 논문에서 다루고자 하는 레코드 저장노드 크기와 관련된 연구도 비슷한 경향을 보여왔다. 즉 백업, 락 성능 향상의 관점에서 레코드 저장노드 크기를 선택해왔던 것이다. 여기서 "레코드 저장노드"란 레코드들을 묶어 저장하는 블록을 지칭한다(예, slotted page).

그런데 초고성능의 트랜잭션 처리 능력이 요구되는 최신 MMDB 응용 분야에서는 MMDB의 트랜잭션 처리 능력으로도 한계를 느끼게 되었다. 이런 분야에서는 로깅, 체크포인트, 락 등을 포기하고 레코드 관리, 인덱스 등 MMDB의 기본적인 부분만으로 시스템을 운영하게 되었다. 이에 따라 레코드 관리, 인덱스의 성능이 중요한 성능 요소로 등장하게 되었다.

이 논문에서는 레코드 관리, 인덱스를 묶어 "레코드 저장구조"라는 용어로 표현하고자 한다. 지금까지의 레코드 저장노드 크기 연구는 백업, 락 성능에만 관심을 가져왔을 뿐 레코드 저장구조 성능에 초점을 맞춘 경우는 없었다. 이 논문에서는 최신 MMDB 응용에서 중요한 요소가 된 레코드 저장구조 성능에 초점을 맞춰, 레코드 저장구조 성능을 최적화하는 레코드 저장노드 크기를 실험을 통해 연구하고자 한다. 특히 기존에 어느 정도 연구가 진행되어있는 인덱스보다는 레코드 관리부 성능에 초점을 맞출 것이다. 최종적으로는 MMDB 시스템 전체 성능의 최적화 관점에서, 실험 결과 얻어진 레코드 저장노드 크기를 검토해보려고 한다.

논문의 진행은 다음과 같다. 2장에서는 기존 연구를 살펴보고, 3장에서는 실험 변수 및 방법에 대해 설명하며, 4장에서는 실험 결과를 보이고 이를 캐시 친화 관점에서 분석한다. 5장에서는 MMDB 시스템 전체 성능 최적화 관점에서 실험 결과를 검토해보고 결론을 내린다.

2. 기존 연구

기존 MMDB에서 사용된 레코드 저장노드 크기는 세 가지이다. 페이지 크기, 세그먼트 크기(페이지 크기의 정수배), L2 캐시라인 크기이다. 페이지 크기를 사용한 대표적 논문은 [1]로, 유닉스의 mlock 시스템콜에 의한 효율적인 페이지 단위 락을 구현하였다. 세그먼트 크기를 사용한 대표적 논문은 [2]로, 세그먼트 단위의 효율적인 백업·복구 알고리즘을 구현하였다. [3]의 경우 세그먼트 크기를 사용하여, 백업·복구를 효율적으로 구현한 것은 물론 mlock 시스템콜에 의해 락도 효율적으로 구현하였다. L2 캐시라인 크기를 사용한 기존 연구는 인용하지 않는다. L2 캐시라인 크기를 사용한 것으로 보이는 기존 연구가 있지만, 사용했다는 명확한 언급은 되어있지 않기 때문이다. 많은 기존 MMDB는 레코드 저장노드 크기로 페이지 혹은 세그먼트 크기를 사용하였으며, 이는 백업·복구, 락을 효율적으로 구현할 수 있기 때문이다.

3. 실험 방법

실험에서는 레코드 저장노드 크기에 따른 레코드 저장구조 성능을 구하려고 한다. 레코드 저장구조 성능이라 하면, 레코드 검색, 삽입, 삭제 등 여러 경우의 성능을 포함한다. 이 논문에서는 이 중 레코드 검색 성능만을 대상으로 한다. 그 이유는 MMDB는 일반적으로 레코드 검색 연산이 삽입, 삭제 연산에 비해 절대적으로 많기 때문이다(예, 검색:타 연산 = 10,000:1). 또 이 논문에서

는 레코드 저장노드를 단순히 레코드들의 배열이라 가정한다. 레코드 저장노드의 대표적인 구조인 slotted page도 기본적으로는 레코드들의 배열이고, 특정 구조에 논의를 한정시키지 않기 위해서이다. 레코드는 선두 4바이트에 레코드 번호가 있고 나머지는 쓰레기 값이 들어간다. 인덱스는 레코드 번호에 대해 건다. 인덱스로는 노드 크기를 L2 캐시라인 크기로 한 B+-트리를 사용한다. 이것은 MMDB에서 널리 쓰이는 인덱스이다.

레코드 저장구조 성능에 영향을 미치는 변수는 레코드 저장노드 크기(w) 외에도 여러 가지가 있다. 실험에서는 이 중 레코드 수(#r), 레코드 크기(rs), 검색 모드(mode), 연산(op), 검색 비율(scan)의 5개 변수를 선택한다. w를 포함하면 6개의 변수가 된다. 모든 변수를 고려하여 실험을 하는 것은 쉽지 않기 때문에 상대적으로 중요한 변수를 선택한 것이다. 각 변수가 가질 수 있는 모든 값에 대해 실험한다는 것도 무리가 따르기 때문에, 각 변수가 가질 수 있는 값도 몇 개로 한정한다. 이는 표 1에 나타나 있다.

표 1 실험 변수

변수	단위	가능 값
레코드 저장노드 크기(w)	L2 라인 크기	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024
레코드 수(#r)	개	500, 5k, 50k, 500k
레코드 크기(rs)	바이트	8, 32, 64, 128, 256
검색 모드(mode)		물리적 순차(PhySeq), 인덱스 순차(IdxSeq), 인덱스 랜덤(IdxRan)
연산(op)		레코드 검색(Retrieve), 필드 합(Sum)
검색 비율(scan)	%	5, 20, 45, 80

w의 가능한 값은 L2 캐시라인 크기의 배수로 한정하였다. 그 이유는 메인메모리와 L2 캐시 사이의 데이터 전송 단위가 L2 캐시라인 크기이기 때문에, L2 캐시라인에 맞추지 않은 레코드 저장노드 크기는 비효율성을 수반하기 때문이다. #r은 테이블 내의 레코드 수를 말한다. #r 값에서 k는 1,000을 곱해주라는 의미이다. mode에서 순차 검색은 물리적 순차와 인덱스 순차로 나뉜다. 물리적 순차(PhySeq)란 레코드의 물리적 저장 순서가 순차 검색되는 순서로 저장되어있는 경우를 말한다. 이에 비해 인덱스 순차(IdxSeq)는 물리적 저장 순서와는 상관없이 인덱스 상에서의 순서대로 검색하는 경우를 말한다. op에서 Retrieve는 사용자가 쿼리한 레코드를 가공 없이 그대로 사용자에게 돌려주는 경우를 말한다. 이에 비해 Sum은 특정 필드의 합을 구하는 집단 함수(aggregation function)를 말한다. Retrieve는 메모리 복사 연산의 비중이 큰데 비해, Sum은 메모리 복사 연산의 비중이 작고 산술 연산의 비중이 크다. scan은 순차 검색에만 적용되는 변수이며, 테이블 내 전체 레코드 수 중 검색되는 레코드 수의 비율을 의미한다.

실험은 다음과 같은 과정으로 수행한다.

① 6개 변수에 값을 대입, 실험하여 레코드 검색 시간을 측정한다. 동일한 변수 값 조합에 대해 10회 실험하여 그 평균을 해당 변수 값 조합의 검색 성능으로 한다. 이를 perf(w, #r, rs, mode, op, scan)라 하자.
② 모든 가능한 변수 값 조합에 대해 perf를 구한다.
③ mode가 IdxRan인 경우 모든 가능한 scan 값에 대해, perf(w, #r, rs, mode, op, scan) := perf(w, #r, rs, mode, op)로 한다. 이것은 IdxRan이 PhySeq, IdxSeq와 동일한 가중치를 부여받기 위해서이다.
④ perf를 정규화한다. 수식으로 표현하면, norm(w, #r, rs, mode, op, scan) := {perf(w, #r, rs, mode, op, scan) − avg_w(#r, rs, mode, op, scan)} / $stdev_w$(#r, rs, mode, op, scan) 이다. avg는 평균, stdev는 표준편차를 의미한다.
⑤ w별로 norm 값들을 더해 정규화된 합을 구한다. 수식으로 표현하면, normSum(w) := $\Sigma_{(\#r,\ rs,\ mode,\ op,\ scan)}$ norm(w, #r, rs, mode, op, scan) 이다.
⑥ 가로축 w, 세로축 normSum으로 그래프를 그린다.

normSum(w)은 가능한 모든 변수 조합에 대해 (정규화된) 검색 시간을 가중치 없이 합한 것이다. 따라서 normSum(w)이 클수록 해당 w에서 종합적인 검색 성능은 나쁜 것을 의미한다.

실험 환경은 표 2와 같다.

표 2 실험 환경

SUN UltraSPARC-IIi 332MHz	8kB
2MB	48엔트리*8kB = 384kB
64B	62cycles
8cycles	512MB
70cycles	

4. 실험 결과 및 분석

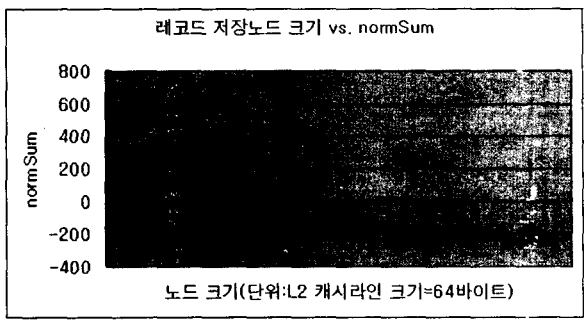

그림 1 레코드 저장노드 크기 vs. normSum

실험 결과 얻어진 그래프는 그림 1과 같다. 그림 1은

레코드 저장노드 크기가 "페이지 크기 또는 그 이상일 때" 레코드 저장구조 성능이 최고임을 보여주고 있다. 보다 정확히 표현하면 "페이지 크기 또는 그 정수배일 때"이다. 페이지 크기의 정수배가 되지 않을 경우 필연적으로 성능 상의 비효율성을 수반하게 되기 때문이다. 참고로 동일한 실험을 HP rp5405(2×CPU) 서버에서도 수행해보았는데 거의 동일한 양상의 그래프를 얻었다.

MMDB의 레코드 저장구조 성능을 좌우하는 것은 캐시 친화성(cache-consciousness)이다. 여기서 CPU 캐시와 친화성 개념에 대해 간단히 설명한다. MMDB에서 데이터 원본은 메인메모리에 위치하는데, 메인메모리 속도는 CPU 연산속도에 비해 상당히 느리다. 이 격차를 완화하기 위한 것이 CPU 캐시이다. CPU 캐시는 메인메모리와 CPU 연산유닛 사이에 위치하며 CPU 연산속도와 비슷한 속도로 동작한다. 캐시 친화성을 높인다는 것은 CPU 캐시 미스로 인한 소모 시간을 줄이는 것을 의미한다.

CPU 캐시는 L1 캐시, L2 캐시, TLB(Translation Lookaside Buffer)로 구성된다. 이 중 메인메모리와 직접 데이터를 주고받는 것은 L2 캐시와 TLB이다. 따라서 L2 캐시와 TLB의 캐시 친화성이 성능에 있어 중요하다. L2 캐시와 메인메모리 사이의 데이터 전송단위는 L2 캐시라인 크기로 보통 64바이트이다. 같은 L2 캐시라인 내에서의 메모리 접근은 L2 캐시 미스를 발생시키지 않는다. 이에 비해 TLB의 데이터 전송단위는 TLB 엔트리 크기로 이는 페이지 크기와 같다. 전형적인 페이지 크기는 8kB이다. 같은 페이지 내에서의 메모리 접근은 TLB 미스를 발생시키지 않는다.

그런데 최근의 하드웨어는 L2 캐시라인의 병렬 prefetch 기술이 크게 발달되었다. 병렬 prefetch란 동시에 여러 개의 L2 캐시라인을 병렬적으로 읽어들이는 것을 말한다. 90년대 초·중반의 하드웨어가 4개 정도의 병렬 prefetch를 지원했던 반면, [4]에서 사용된 컴팩 ES40 머신은 15개의 병렬 prefetch를 지원한다. 더욱 최신의 머신은 20~30개의 병렬 prefetch를 지원한다. 병렬 prefetch의 발전에 의해 L2 캐시라인 크기의 수배~수십배에 달하는 레코드 저장노드도 마치 한 개의 L2 캐시라인을 읽어오듯이 읽어올 수 있게 되었다. 따라서 신형 머신에서 레코드 저장노드 크기에 미치는 L2 캐시의 영향력은 상대적으로 작아지게 되었다.

이에 비해 레코드 저장구조에서 발생하는 TLB 미스는 레코드 저장노드 크기가 페이지 크기에 가까워질수록 줄어들다가 페이지 크기 또는 그 정수배일 때 최소가 된다. 그 이유를 예로써 설명한다. 레코드 저장노드 크기가 페이지 크기일 때 테이블이 p개의 페이지를 사용하여 저장되었다고 하자. 이 때 레코드 저장노드 크기를 페이지 크기의 절반으로 할 경우 테이블은 최대 2p개의 페이지에 흩어져 저장될 수 있다. 더 많은 페이지를 사용하여 저장될수록 레코드 저장구조의 TLB 미스는 늘어나게 된다. TLB 미스의 수는 TLB 친화성과 반비례하며, TLB 친화성은 성능과 비례한다. 따라서 레코드 저장노드 크기가 페이지 크기 또는 그 정수배일 때 TLB 친화성은 최대가 되며 그에 따라 성능도 향상된다.

지금까지의 논의를 정리하면, 신형 머신에서 레코드 저장노드 크기에 따른 레코드 저장구조 성능은 L2 캐시 친화성보다는 TLB 친화성에 의해 좌우된다. 신형 머신에서 TLB 친화성이 중요하다는 결과는 최근의 다른 연구결과들로부터도 뒷받침된다. [5] 등 기존의 MMDB 인덱스는 L2 캐시 친화성을 생각하여 노드 크기를 L2 캐시라인 크기로 하는 것이 일반적이었다. 그런데 [6] 등 신형 머신을 기반으로 발표된 최신의 연구결과에 따르면, 인덱스 노드 크기를 L2 캐시라인 크기보다 훨씬 크게 하는 것이 성능을 더 좋게 한다는 것이다. [6]은 성능 향상의 중요한 이유로 TLB 친화성 향상을 꼽고 있다.

5. 결론

실험 결과 레코드 저장구조 성능이 최대로 되는 것은 레코드 저장노드 크기가 페이지 크기 또는 그 정수배일 때였다. 그리고 그 이유는 레코드 저장노드 크기가 페이지 크기 또는 그 정수배일 때 레코드 저장구조의 TLB 친화성이 최대가 되기 때문이었다.

이제 레코드 저장구조 성능에 한정짓지 말고 MMDB 시스템 전체 성능의 관점에서 생각해보자. MMDB 시스템 성능에서 가장 큰 비중을 차지하는 것은 로깅, 체크포인트 등 백업 관련 부분과 락이라는 것은 앞서 언급한 바 있다. 그런데 많은 기존 연구에 의하면 백업, 락 성능이 최대가 되는 것은 레코드 저장노드 크기가 페이지 크기 또는 그 정수배일 때이다. 따라서 레코드 저장노드 크기를 페이지 크기 또는 그 정수배로 하는 것은 MMDB 시스템 전체적으로도 최적의 선택이 되는 것이다.

6. 참고 문헌

[1] H.V.Jagadish, et al., "Dali: A High Performance Main Memory Storage Manager", Proceedings of the 20th International Conference on Very Large Data Bases(VLDB '94), 1994

[2] J.Lin, et al., "Segmented Fuzzy Checkpointing for Main Memory Databases", Proceedings of the 7th International Conference on Databases and Expert Systems Applications(DEXA '96), 1996

[3] J.Baulier, et al., "DataBlitz: A High Performance Main Memory Storage Manager", Proceedings of the 24th International Conference on Very Large Data Bases(VLDB '98), 1998

[4] S.Chen, et al., "Improving Index Performance through Prefetching", Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data(SIGMOD 2001), 2001

[5] J.Rao, et al., "Cache Conscious Indexing for Decision-Support in Main Memory", Proceedings of the 25th International Conference on Very Large Data Bases(VLDB '99), 1999

[6] R.Hankins, et al., "Effect of Node Size on the Performance of Cache-Conscious B+-trees", Proceedings of the International Conference on Measurements and Modeling of Computer Systems(SIGMETRICS 2003), 2003