카운팅 블룸 필터를 활용한 FTL 로그영역 탐색 비용 감소

강운학, 이상원
성균관대학교 전자전기컴퓨터공학과
e-mail:woonagi319@skku.edu

Diminishing FTL Log Area Searching Cost Using the Counting Bloom Filter

Woon-Hak Kang, Sang-Won Lee
Department of Electrical and Computer Engineering, Sung-Kyun-Kwan University

요 약

플래시 메모리은 많은 부분에서 기존의 저장장치인 HDD에 비해 좋은 성능을 지녔다. 하지만 HDD와는 달리 데이터의 덮어쓰기가 허용 되지 않는다. 이 문제를 해결하기 위해 플래시 메모리는 FTL(Flash Translation Layer)을 사용하고 있으며, FTL을 통해 기존의 저장장치와 동일하게 사용할 수 있다. FTL들 중 로그영역을 사용하여 성능을 개선한 것들이 많은데, 로그영역의 사용으로 인해 읽기/쓰기 작업시 반드시 로그영역을 탐색을 해야만 했다. 본 논문에서는 카운팅 블룸필터(Counting bloom filter)를 활용하여 불필요한 로그영역 탐색을 줄이는 기법을 제안하였고, 실험을 통해 로그영역에 최신 데이터가 없는 경우 탐색횟수를 크게 줄일 수 있는 것을 확인하였다.

1. 서론

플래시 메모리는 기존의 저장장치인 HDD에 비해 좋은 성능을 지녔다. 읽기/쓰기 속도도 빠르며, 충격에 강하고, 소음도 발생시키지 않는다. 이러한 특성으로 인해 휴대폰, MP3, 휴대용 저장장치 등 많은 곳에서 플래시 메모리를 활용하고 있다.

플래시 메모리는 물리적으로 페이지와 페이지의 집합인 블록으로 이루어져 있는데 대블록 낸드(Large Block Nand) 플래시 메모리의 경우 페이지의 크기는 2KB, 한 블록은 128KB(64개의 페이지)이다. 데이터의 읽기/쓰기는 페이지단위, 지우기는 블록단위로 처리한다. 플래시 메모리는 HDD와 달리 덮어쓰기가 되지 않는데, 덮어쓰기가 발생하면 지우기 명령을 실행한 후 쓰기를 수행하게 된다. 따라서 읽기/쓰기 명령을 읽기/쓰기/지우기 명령으로 바꾸어 주어야 하는데 이런 작업을 FTL(Flash Translation Layer)이 담당하고 있다.

많은 FTL들이 제안되었는데, 로그 영역을 사용하는 FTL로는 FAST[1], BAST[2]등이 있다. 이 FTL들은 읽기/쓰기 과정에서 최신 데이터를 찾기 위해 로그영역을 탐색해야만 하는데, 플래시 메모리의 용량이 증가하면 값비싼 탐색 비용으로 인해 성능이 저하된다.

본 논문에서는 삭제가 허용되는 카운팅 블룸필터(Counting Bloom Filter)[3]를 활용하여 로그영역에 원하는 페이지가 없는 경우 탐색을 회피할 수 있는 방법을 제안하였다. 실험을 통해 확인한 결과 블룸필터의 크기에 따라 약 80%이상의 불필요한 탐색을 회피할 수 있었다.

2. 관련연구

2.1 로그영역 기반 FTL

FAST와 BAST알고리즘 블록 매핑 기반 FTL이며, 로그영역을 이용한다. FAST는 페이지 단위로, BAST는 블록 단위로 로그영역을 이용하는 알고리즘이다. 로그영역을 사용함으로서 비용이 많이 드는 삭제명령과 쓰기 명령을 피하게 되고, 그로인해 성능향상을 이룰 수 있었다. 로그영역을 사용하는 FTL은 읽기 과정에서 최신 정보가 로그영역에 있는지 확인해야만 한다. 또, FAST는 쓰기 과정에서 로그영역과 데이터영역의 병합이 발생하면, 동일한 블록에 속한 페이지를 로그영역에서 모두 찾아야만 한다. 이러한 과정에서 로그영역에 원하는 페이지가 없는 경우에도 반드시 로그영역을 탐색해야만 했다.

본 논문에서는 FAST FTL 알고리즘을 기준으로 구현과 성능평가를 하였다.

2.2 블룸필터(Bloom Filter)

블룸필터는 비트벡터와 여러 개의 해시함수를 이용하여 어떤 집합에 원하는 항목이 존재하는지를 적은 메모리로

본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업(NIPA-2009-(C1090-0902-0046))과 2008년 정부(교육과학기술부)의 재원으로 한국학술진흥재단(KRF-2008-0641)의 지원을 받아 수행하였음.

빠르게 확인할 수 있는 기법이다. 블룸필터는 항목이 없다는 'No'와 항목이 존재할 지도 모른다는 'Maybe'를 결과로 반환한다. 블룸필터는 실제로는 항목이 집합에 존재하지 않지만 존재한다고 반환하는 '거짓양성(false positive)'을 포함하고, 거짓양성 비율은 비트벡터의 크기, 저장가능 항목의 개수 그리고 해시함수의 개수에 따라 달라진다. 블룸필터는 여러 항목이 같은 비트를 1로 만들 수 있는데, 항목이 집합에서 빠지더라도 '거짓음성(false nagative)'이 발생할 수 있기 때문에 해당 비트를 0으로 바꿀 수 없다.

항목의 삭제를 허용한 대체 기법으로 카운팅 블룸필터가 있다. 이 기법은 비트벡터의 각 비트에 카운터를 두어 항목이 추가되면 해당 비트의 카운터를 증가시키고 삭제되면 카운터를 감소하는 방식이다. 일반적으로 4비트를 사용하면 확률적으로 안전하다고 알려져 있다.[3]

3. 카운팅 블룸필터를 이용한 FAST FTL

본 논문에서는 FAST 알고리즘의 로그영역에 입력되는 항목들을 대상으로 카운팅 블룸필터를 추가하였다. 로그영역의 페이지 매핑을 해시 테이블로 구현하였는데, B+트리와 같은 트리 자료구조를 이용하여 할 수도 있으나 카운팅 블룸필터에 사용한 해시함수의 값을 그대로 이용할 수 있기 때문에 해시 테이블을 이용하였다.

카운팅 블룸필터는 어떤 항목이 집합에 존재하지 않는 것은 확실히 판단할 수 있기 때문에 불필요한 로그영역 탐색을 효과적으로 감소시킬 수 있다. 약 10%의 거짓양성이 발생 하더라도 로그 탐색 회피율이 80%이라면 전체적으로는 2%만이 불필요한 탐색이기 때문에 2%의 비용을 감수하고 80%의 로그 영역 탐색을 회피할 수 있다.

카운팅 블룸 필터를 사용한 FTL의 동작은 다음과 같이 이뤄진다. 쓰기 명령이 주어지면 입력된 논리적 주소를 블록 매핑 테이블을 통해 물리적 주소로 바꾸고 해당 물리적 주소공간이 비어있는지를 확인한다. 만약 물리적 주소공간이 비어있지 않다면, 주어진 데이터는 로그영역에 기록한다. 이 때, 로그영역의 기록되는 모든 페이지는 카운팅 블룸필터와 해시 테이블에도 매핑 정보를 기록한다.

카운팅 블룸필터와 해시 테이블에 매핑 정보를 기록하기 전에 로그영역을 탐색하여 이전에 기록된 동일한 페이지가 있는지를 검사하는데, 이전에 기록된 페이지는 곧 쓸모없는 데이터가 될 것이므로 매핑 정보를 최신 페이지로 변경해주어야 하기 때문이다. 블룸필터를 사용하여 테스트 결과 'No'라면 해당항목을 블룸필터와 해시 테이블에 등록하고 로그영역에 데이터를 기록한다. 테스트결과가 'Maybe'라면 페이지가 해시 테이블에 존재하는지 검사해서 해당 매핑 정보의 현재 입력될 항목으로 바꾸어준다. 'Maybe'를 반환했을 경우 해당 페이지가 로그영역에 존재하지 않을 수도 있기 때문에, 해시 테이블에 매핑 정보가 존재하지 않는다면 카운팅 블룸필터와 해시 테이블에 해당 정보를 각각 입력해준다.

읽기 명령이 주어지면, 원하는 페이지가 로그 영역에 존재하는 것인지 카운팅 블룸필터로 테스트한다. 반환 결과가 'No'이면 로그 영역 존재하지 않는 것이 확실하므로 블록 매핑 정보를 통해 물리적 주소를 반환할 수 있다. 'Maybe'를 반환하면 먼저 해시 테이블에 해당 페이지가 존재하는지 검사를 하여 매핑 정보를 얻어낸다. 해시 테이블에 매핑 정보가 없을 경우에는 로그영역에 원하는 페이지가 존재하지 않으므로 블록 매핑 테이블 정보를 이용하여 물리적 주소를 반환한다.

4. 성능평가

성능평가는 8Gbyte SSD를 가정하여 시뮬레이션으로 평가하였다. 리눅스에서 실제 8Gbyte SSD에 TPC-C 워크로드를 처리하는 동안 발생하는 IO명령을 blktrace를 이용하여 800만건을 저장하였다. 로그영역의 크기는 512MB 크기와 800MB크기로 하였으며, 카운팅 블룸필터의 비트벡터 크기는 로그영역에 저장 가능한 페이지 개수의 3~5배로 하였다. 해시 함수는 총 3개를 사용하였다.

<표 1> 블룸필터의 크기에 따른 회피율과 거짓양성비율

로그영역의 크기		3배	4배	5배
512MB	회피율	81.34%	84.89%	85.43%
	거짓양성	28.83%	12.10%	8.87%
800MB	회피율	73.77%	80.44%	80.97%
	거짓양성	13.45%	6.10%	3.45%

800MB의 로그영역을 사용할 경우 카운팅 블룸필터의 최대크기는 약 204KB이며, 거짓양성에 따른 불필요한 탐색은 전체 탐색 횟수의 약 0.69%에 불과하다.

5. 결론

본 논문에서는 적은양의 메모리 공간을 이용하여 FTL의 불필요한 로그영역 탐색비율을 효과적으로 줄일 수 있는 방안을 제안하였으며, 실험을 통해 실제로 많은 양의 탐색을 회피할 수 있는 것을 확인하였다.

따라서 플래시 메모리의 용량과 메모리 크기에 적합한 카운팅 블룸필터를 구성하면 플래시 메모리의 읽기/쓰기 명령을 더 빠르게 수행할 수 있을 것이다.

참고문헌

[1] J. Kim, J.M. Kim, S. Noh, S. L. Min, and Y. Cho. "A Space-efficient Flash Translation Layer for Compactflash Systems". Consumer Electronics, IEEE Transactions on, 2002.

[2] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, and H.-J. Song. "A log buffer-based Flash Translation Layer using Fully-associative sector translation:. ACM Trans. Embed. Comput. Syst., 2007.

[3] Li Fan , Pei Cao , Jussara Almeida , Andrei Z. Broder. "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol". IEEE/ACM Transactions on Networking. 1998.