플래시메모리 SSD를 이용한 확장형 버퍼 관리
(Extended Buffer Management with Flash Memory SSDs)

심 도 윤 † 박 장 우 † 김 성 탄 † 이 상 원 †† 문 봉 기 †††
(Doyoon Sim) (Jangwoo Park) (Sungtan Kim) (Sangwon Lee) (Bongki Moon)

요 약 최근 들어, 플래시메모리의 가격이 지속적으로 낮춰지고, 플래시메모리 기반 SSD 컨트롤러 기술이 급격하게 발전하면서 중저가의 고성능 플래시 SSD가 시장에 널리 보급되고 있다. 하지만, 데이터베이스 분야에서 가격 등의 이유로 당분간 플래시 SSD가 하드디스크를 완전히 대체하기는 쉽지 않을 것이다. 대신 플래시 SSD의 빠른 성능을 캐시 용도로 활용하는 접근법이 현실적이고, 실제로 하드디스크와 플래시메모리를 하이브리드 형태로 사용하는 접근법들이 제시되었다.

본 논문에서는 기존의 접근법들과는 달리, 플래시 SSD를 데이터베이스의 버퍼에서 밀려나는 페이지들을 순차적으로 저장하고, 재 참조될 때 하드디스크 대신 플래시 SSD에서 읽혀지도록 하는 확장 버퍼 아키텍처를 제안한다. 플래시 SSD를 저장장치 레벨에서 캐시로 사용하는 기존 방법들에 비해, 플래시 SSD를 호스트 시스템에서 확장 버퍼로 사용함으로써 읽기 측면에서 주 버퍼에서 밀려나는 웜 페이지(warm page)들에 대해 상당한 성능 개선을 이룰 수 있다. TPC-C 트레이스를 사용한 시뮬레이션 결과, 주 버퍼에 없는 페이지들이 확장 버퍼에서 찾아지는 적중률이 60%를 넘는 사실을 알 수 있었다. 이 확장 버퍼 아키텍처는, 동일한 비용을 지불하는 다른 접근법, 즉 DRAM을 버퍼로 추가하는 기법과 하드디스크를 추가하는 기법에 비해 가격 대비 성능 개선 효과가 높다.

키워드 : 버퍼 관리, 플래시메모리 SSD, 확장형 버퍼

Abstract As the price of flash memory continues to drop and the technology of flash SSD controller innovates, high performance flash SSDs with affordable prices flourish in the storage market. Nevertheless, it is hard to expect that flash SSDs will replace harddisks completely as database storage. Instead, the approach to use flash SSD as a cache for harddisks would be more practical, and, in fact, several hybrid storage architectures for flash memory and harddisk have been suggested in the literature.

In this paper, we propose a new approach to use flash SSD as an extended buffer for main buffer in database systems, which stores the pages replaced out from main buffer and returns the pages which are re-referenced in the upper buffer layer, improving the system performance drastically. In contrast to the existing approaches to use flash SSD as a cache in the lower storage layer, our approach, which uses flash SSD as an extended buffer in the upper host, can provide fast random read speed for the warm pages which are being replaced out from the limited main buffer. In fact, for all the pages which are missing from the main buffer in a real TPC-C trace, the hit ratio in the extended buffer could be more than 60%, and this supports our conjecture that our simple extended buffer

· 본 논문은 서울시의 지원으로 수행한 서울시 산학연 협력사업(PA090903)의 연구 결과입니다.
· 본 논문은 언어지식베이스(ITRC 지능형 HIC융합 연구센터 3 세부과제, 2005.9~2013.8, IITA)의 연구 결과입니다.

† 비 회 원 : 성균관대학교 정보통신공학부
sdy7777777@naver.com
sliod@naver.com
sungtan.kim@samsung.com
†† 정 회 원 : 성균관대학교 정보통신공학부 교수
swlee@skku.edu
††† 비 회 원 : Univ. of Arizona Dept. of Computer Science 교수
bkmoon@cs.arizona.edu

논문접수 : 2010년 7월 6일
심사완료 : 2010년 9월 27일

정보과학회논문지: 데이타베이스 제37권 제6호(2010.12)

approach could be very effective as a cache. In terms of performance/price, our extended buffer architecture outperforms two other alternative approaches with the same cost, 1) large main buffer and 2) more harddisks.

Key words : Buffer Management, Flash Memory SSD, Extended Buffer

1. 서 론

최근 들어 플래시메모리 칩 기술 및 플래시메모리 SSD(Solid State Disk, 이하 플래시 SSD라 칭함.), 컨트롤러 기술의 급격한 발전으로 인해서, 낮은 가격의 고성능 플래시 SSD가 상업적으로 이용가능하다. 예를 들어, 4KB 크기의 페이지 기준으로 초당 30,000회 읽기 연산과 3,000회의 쓰기 연산이 가능한 80GB 크기의 플래시 SSD가 200$ 근처의 가격으로 시장에서 구매가능하다. 기계적으로 동작하는 하드디스크에 비해, 플래시 SSD는 전자적인 특성으로 인해 성능 면에서 빠르고 균일한 접근 속도를 보장하며, 충격, 소음, 발열, 전력소비 등에서 매우 유리하다. 이러한 장점으로 인해서, 최근 들어서 플래시 SSD는 모바일 및 PC 분야이외에, 대용량 데이터베이스를 위한 엔터프라이즈 환경에서도 가격과 성능 측면에서 하드디스크보다 우수한 저장장치로 인식되고 있다[1,2]. 그렇지만, 여전히 비싼 가격과 새로운 저장장치의 안정성 등을 이유로 적극적인 도입을 꺼리는 심리적인 이유로, 플래시 SSD는 데이터베이스 분야에서 당분간 하드디스크를 완전히 대체하는 저장장치로는 한계가 있을 수 있다. 이러한 이유로, 적은 용량의 플래시 SSD를 일종의 캐시로 사용하여 전체 시스템 성능을 높이는 방안에 대한 연구들이 데이터베이스 분야에서 진행되었다. 예를 들어, 데이터베이스의 객체들에 대한 접근 빈도를 미리 분석하여 자주 사용되는 데이터를 플래시 SSD에 두는 방법[3]과 자주 참조되는 데이터를 실행 시에 구분하여 데이터를 그 접근 빈도와 유용성에 따라서 하드디스크와 플래시 SSD간에 동적으로 이동하는 방법[4]이 제시되었다. 하지만, 이러한 접근법들은 데이터 갱신이 자주 일어나는 환경에서는 실제 성능 향상이 크지 않거나[3], 실행 시에 데이터에 대한 접근 패턴 구분과 데이터 이동을 위한 사용하는 알고리즘 복잡도, 그리고 접근 패턴 정보를 저장하기 위한 공간 오버헤드[4]가 존재한다. 특히, 이러한 기법들은 플래시 SSD에 대한 임의 쓰기(random write)를 유발하고, 이는 덮어쓰기가 불가능한 플래시메모리 특성으로 인해 성능 저하를 유발할 수 있다.

기존 연구들의 이런 문제점들을 해결하기 위해, 본 논문에서는 플래시 SSD를 확장 버퍼(extended buffer)로 사용하는 아키텍처를 제안한다. 즉, DRAM으로 구성된 버퍼(이하 주 버퍼(main buffer)라 칭함)에서 밀려나는 페이지를 일종의 큐(queue) 형태인 플래시 SSD에 순차적으로 쓰고, 플래시 SSD에서 하드디스크로 다시 밀려나기 전에 읽기요청이 발생하면 플래시 SSD에서 주 버퍼로 읽어 들인다. 특이한 점은 읽기 전용의 페이지도 주 버퍼에서 밀려나는 경우, 플래시 SSD에 쓰여 진다. 이런 측면에서 플래시 SSD는 주 버퍼에서 밀려나는 페이지들에 대해 확장된 버퍼(이하 확장 버퍼라 칭함.)로써 역할을 한다. 이 아키텍처는, 기존 방법들에 비해 1) 플래시 SSD에 캐싱을 위한 별도의 복잡한 알고리즘이 필요하지 않기 때문에 실행 시에 관리 방법이 단순하고, 2) 큐 형태의 순차 쓰기만 발생하기 때문에 임의 쓰기가 유발되지 않는 장점을 제공한다.

본 논문에서, 우리는 기존 데이터베이스 분야의 대표적인 버퍼 교체 정책들에 플래시 SSD를 확장 버퍼로 사용하도록 변형하고, TPC-C 벤치마크 수행시의 접근 패턴을 사용해서 각 방법들을 시뮬레이션하고 성능을 측정하였다. 논문에서 제안한 아키텍처는 동일한 비용의 다른 기법들에 비해, 즉, 확장 버퍼로 사용한 플래시 SSD와 동일한 가격만큼의 1) DRAM을 추가한 경우와 2) 하드디스크를 추가한 경우에 비해, 성능 면에서 훨씬 우수한 것으로 나타났다. 특히, 주 버퍼에 존재하지 않는 전체 페이지들에서 확장 버퍼로 사용된 플래시 SSD에서 찾아지는 적중률이 70%에 이르고 플래시 SSD의 빠른 임의 읽기가 성능향상에 직접적인 영향을 준 것으로 분석된다.

본 논문의 구성은 다음과 같다. 2장은 기존의 버퍼 교체 정책 및 플래시 SSD를 확장 버퍼로 사용한 기법에 대해 설명한다. 3장에서는 기존의 기법과 새로운 기법을 시뮬레이터로 구현하여 실험한 결과를 보여주고 플래시 SSD 확장 버퍼 기법의 성능 분석을 한다. 마지막으로 4장에서는 논문을 요약하고 마무리한다.

2. 플래시 SSD를 이용한 확장 버퍼 아키텍처

2.1 버퍼 교체 정책

데이터베이스 분야의 버퍼 교체 정책들은 페이지 접근의 시간 지역성(temporal locality)을 기반으로 자주 접근되는 페이지를 가능한 주 버퍼에 유지함으로써 주 버퍼의 적중률을 높여 하드디스크에 대한 접근을 최소화해서 성능을 높인다[5,6]. 지금까지 데이터베이스 분야에서 제안된 주요 버퍼 교체 정책은 크게 두 가지, 즉

1) 처음 참조되는 모든 페이지는 우선 주 버퍼에 반입시키고, 참조가 덜 되는 페이지들을 주 버퍼에서 적극적으로 밀어내는 정책(예를 들어, LRU와 LUR-K[5]), 그리고 2) 참조가 빈번할 것으로 예상되는 페이지(hot page, 이하 핫 페이지라 칭함.)를 적극적으로 주 버퍼로 반입하는 정책으로(예를 들어, 2Q[6]) 구분할 수 있다. 이 두 관점에서 주 버퍼에 대한 확장 버퍼로써 플래시 SSD의 역할과 장점에 대해 알아보자. 본 장에서는 두 가지 대표적인 버퍼 교체정책인 LRU, 2Q에 대해 플래시 SSD를 주 버퍼를 보조하는 확장 버퍼로써의 역할과 장점을 제시하고 플래시 SSD를 확장 버퍼로 사용하는 아키텍처를 설명한다.

2.2 LRU

우선 LRU의 경우, 주 버퍼에서 밀려나는 페이지(victim page, 이하 희생 페이지라 칭함.) 중에는 조만간 다시 재 참조가 될 가능성이 있음에도 불구하고 제한된 주 버퍼 크기로 인해서 어쩔 수 없이 밀려나는 경우가 있다. 그런데, 이와 같이 조만간 재 참조의 가능성이 높은 희생 페이지(warm page, 이하 웜 페이지라 칭함.)가 주 버퍼에서 밀려난 후 재 참조가 되는 경우 다시 주 버퍼로 읽어 들이기 위해서는 하드디스크의 높은 접근 시간(latency) 비용을 치러야 한다. 따라서 LRU 정책에 따른 희생 페이지들을 주 버퍼에서 밀려나는 순서대로 플래시 SSD에 순차 쓰기를 통해 저장하고, 재 참조되는 경우 하드디스크가 아닌 플래시 SSD에서 읽어 들임으로써 하드디스크의 높은 접근시간을 피할 수 있다.

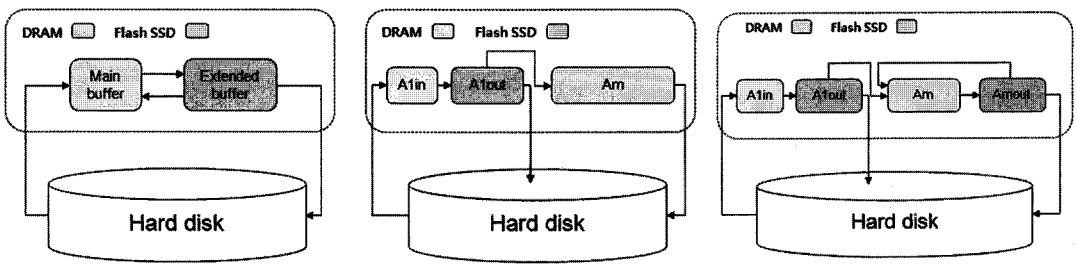
이러한 역할을 위해, 그림 1의 (a)처럼, 플래시 SSD를 주 버퍼의 LRU 큐에서 밀려난 페이지를 순차적으로 플래시 SSD에 저장하도록 하였다. 플래시 SSD에는 라운드 로빈 형태로 순차적으로 희생 페이지들을 기록한다. 플래시 SSD에 저장된 페이지들의 페이지 ID와 플래시 SSD 내에서의 위치를 관리하는 일종의 디렉토리를 DRAM에서 별도로 관리하고 있는 것으로 가정한다. 플래시 SSD의 정해진 용량이 다 찬 경우, 플래시 SSD 확장 버퍼에서 FIFO(first-in first-out) 방식으로 페이지들을 밀어내야 한다. 주 버퍼에서 갱신된 페이지의 경우, 이를 하드디스크로 반영해야 하고, 주 버퍼에서 읽기만 수행된 페이지 경우 하드디스크로 쓰기 연산이 필요 없다.

한 가지 주목할 점은 주 버퍼에서 읽기 연산만 수행된 페이지의 경우에도 플래시 SSD에 기록한다는 점이다. 독자들 중에서 이 부분이 큰 오버헤드로 생각할 수도 있지만, 다음과 논리로 성능상의 장점이 더 많게 된다. 4KB 페이지 기준으로, 주 버퍼에서 읽기만 수행된 특정 페이지를 플래시 SSD 확장 버퍼에 쓰고(0.3ms 소요) 재 참조에 의해 읽는 시간(0.03ms 소요)를 필요로 하는데, 하드디스크로부터 재 참조를 위해 읽는데 소요되는 시간은 일반적으로 5ms 이상이다. 따라서 희생 페이지가 확장 버퍼에서 밀려나기 전에 10%의 이상의 확률로 재 참조되면 성능상의 이점이 많아지게 된다. 뒤의 성능평가 장에서 제시하겠지만, 확장 버퍼내의 페이지들에 대한 적중률 60% 이상이기 때문에 이점이 훨씬 더 많다.

2.3 2Q

앞에서 설명한 바와 같이, 2Q 기법은 핫 페이지를 미리 선별해서 적극적으로 주 버퍼에 반입하는 대표적인 버퍼교체 정책이다[6]. 이를 위해, 2Q 기법은 주 버퍼를 A1in과 Am/A1out 영역으로 구분하고 있다. 페이지가 처음으로 참조되면 A1in 큐에 우선 반입하는데, A1in은 FIFO 방식으로 관리되는 큐이기 때문에 A1in에 있는 동안 재 참조가 되더라도 큐의 순서가 바뀌지 않는다. 결국 A1in 큐에서 반출되는 페이지에 대한 접근 히스토리 정보를 A1out 큐에 유지한다. 다시 참조되는 페이지 중에서 A1out큐에 히스토리 정보가 있는 페이지는 핫 페이지로 간주해서 Am 큐로 반입한다. 즉, 2Q 알고리즘은 A1in/A1out큐를 이용해서 접근 지역성을 기반으로 핫 페이지들만 주 버퍼로 반입해서 주 버퍼의 적중률을 높이는 기법이다.

이와 같은 2Q의 특성상, 플래시 SSD를 확장 버퍼로 사용하는 한 가지 방법은, 그림 1의 (b)처럼, A1in 큐에서 밀려난 페이지들을 A1out 큐에 순차적으로 저장하는 것이다. 기존 2Q에서 A1out 큐에는 A1in에서 밀려난



(a) LRU (b) 2Q (c) 2Q-Flash

그림 1 플래시 SSD 확장 버퍼 구조

페이지들에 대한 메타 정보를 유지하고 있었는데 반해, 플래시 SSD를 확장 버퍼로 사용하는 경우 페이지 자체를 A1out 큐의 형태로 플래시 SSD에 저장하는 차이가 있다. A1in에서 밀려나는 페이지들을 확장 버퍼에서 관리하는 방법은 LRU의 경우와 동일하다. 비록 기존 2Q의 A1out 큐가 핫 페이지를 구분해내는 능력은 갖고 있었지만, 원본 페이지가 하드디스크에 존재하기 때문에 핫 페이지를 주 버퍼 Am으로 읽어 들일 때 하드디스크의 느린 접근 시간을 개선하는 효과는 전혀 없다. 이에 반해서, 플래시 SSD를 A1out에 속하는 페이지들에 대한 확장 버퍼로 사용하는 경우, 플래시 SSD의 빠른 순차 쓰기와 빠른 임의 읽기로 상당한 성능 개선의 효과가 예상된다. LRU의 경우와 달리, 주 버퍼 Am으로 반입될 핫 페이지들에 대한 빠른 접근 속도를 제공하는 것이 이 아키텍처의 목적이다.

그런데, 3장에서 설명하겠지만, 이 아키텍처의 경우 플래시 SSD 확장 버퍼의 용량이 작은 경우에 LRU 기법에 비해 좋은 성능을 보이지만, 확장 버퍼의 크기가 커지면서 이 아키텍처로 인한 성능 개선이 LRU보다 낮아지게 된다. 이 이유는 2Q에서 그리 크지 않은 A1out 정보만을 유지해도 핫 페이지를 구분해 낼 수 있고, 추가적인 공간을 투입해도 더 이상의 성능 개선 효과는 없기 때문이다. 따라서 여분의 공간이 있는 경우, A1out에 투입하는 대신, LRU와 유사하게 동작하는 Am 버퍼를 위한 확장 버퍼로 사용하는 것이 더 유용하다. 따라서 이러한 경우를 위해, 그림 1의 (c)와 같이, 플래시 SSD를 A1out과 Am을 위한 확장 버퍼(Amout)로 사용하는 아키텍처(이하 2Q-Flash라 칭함.)도 제안한다. 2Q-Flash 아키텍처의 목적은 플래시 SSD의 일부분을 활용해서 Am으로 반입될 핫 페이지에 대한 접근 속도를 개선하고, 또한, LRU 확장 버퍼와 마찬가지로, Am에서 밀려나는 웜 페이지들에 대한 재 참조 성능을 높이는 데 있다. A1out과 Amout은 플래시 SSD를 논리적으로 두 영역으로 나누어 할당한다. 각각의 큐는 FIFO 방식이지만 실제 플래시 SSD에서는 원형 링 형태로 구현된다. 이 아키텍처 하에서 자연스럽게 나타나는 튜닝 이슈는 일정한 용량의 플래시 용량이 주어졌을 때, 어떤 비율로 A1out과 Amout으로 할당할 때 최적의 성능을 얻을 수 있는가 하는 점이다. TPC-C 벤치마크 워크로드를 사용한 실험결과 그림 2와 같이 Amout과 A1out의 비율이 6:4일 때 가장 좋은 결과를 보여주었고, 워크로드에 따라서 약간의 차이가 있을 것으로 예상한다.

3. 실험 및 결과

3.1 실험 환경

OLTP 워크로드에서 페이지 접근 트레이스를 얻기 위해 Postgres에서 약 1시간 동안 표준 TPC-C 벤치마크를 Benchmark SQL을 이용해서 수행시키고, 이 때 페이지들에 대한 논리적인 읽기, 쓰기 트레이스를 수집하였다. 읽기, 쓰기 트레이스 수집을 위해 Postgres 버퍼관리자 모듈의 ReadBuffer() 함수와 트랜잭션 로그관리자 모듈의 XLogInsert() 함수를 수정하였다. TPC-C 벤치마크 데이터베이스 크기는 1.5GB이며 디폴트 페이지 크기는 4KB로 설정하였다. 이렇게 얻어진 트레이스를 그림 1에서 제시한 세 가지 아키텍처, 즉 LRU, 2Q, 2Q-Flash를 위한 시뮬레이터를 모두 구현해서 적용하였다. 시뮬레이션 시 주 버퍼의 크기는 전체 데이터베이스의 4% (즉, 60MB) 로 설정하였고, 확장 버퍼인 플래시 SSD 크기는 전체 데이터베이스의 5%부터 50%까지 5%씩 늘려가면서 수행하였다.

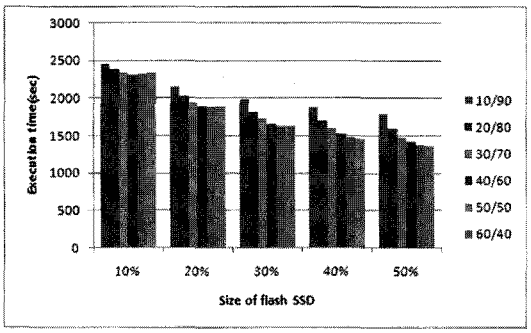


그림 2 Amout/A1out 비율에 따른 성능 변화

각 알고리즘별 성능은 데이터 페이지가 각 큐에서 이동한 횟수와 이동할 때의 시간을 곱해 계산하였다. 예를 들어 메인 메모리에서 플래시 SSD로 페이지가 밀려난 경우 "플래시 SSD 순차 쓰기 시간 * 이동 횟수"로 계산하였다. 보다 정확한 시뮬레이션을 위해 비용 계산에 사용된 수치를 ORION(Oracle IO Calibration) Tool[7]을 사용하여 실제 하드 디스크와 플래시 SSD를 측정하였다(표 1).

2Q/2Q-Flash는 A1in의 크기는 전체 데이터의 25%, Amout과 A1out은 2.3절에서 설명한 바와 같이 6:4의 비율로 설정해서 실험하였다.

표 1 플래시 SSD와 하드디스크 사양: 성능, 가격, 용량

장치 구분	접근 속도(ms)
플래시 SSD: 임의 읽기	0.03
플래시 SSD: 순차 쓰기	0.33
단일 하드디스크(임의 읽기/쓰기)	2.6
하드디스크*2 (RAID0)	1.6

플래시SSD : Intel X25-M G2 (80GB, $225)
하드디스크 : Seagate Cheetah 15K.7 (320GB, $350)
읽기와 쓰기는 모두 4KB 페이지 크기 기준

3.2 성능 평가

3.2.1 수행시간

그림 3은 확장 버퍼로 사용한 플래시 SSD 영역의 크기에 따른 각 알고리즘별 성능이다. 가로축은 실험에 사용한 전체 데이터베이스 용량(1.5GB) 대비 확장 버퍼로 사용한 플래시 SSD 용량의 비율을 나타내며, 세로축은 해당 크기의 확장 버퍼를 사용한 경우 트레이스의 수행 시간을 나타내다. 플래시 SSD 확장 버퍼를 적용한 세 가지 아키텍처 각각이, 하드디스크만 사용한 방법보다 성능이 200% 이상 향상되었고, 특히 플래시 SSD의 크기가 클수록 수행시간이 줄어들었다. 세 아키텍처 중에서, 2Q의 성능이 가장 좋지 못했고, 2Q와 LRU의 장점을 동시에 갖는 2Q-Flash 성능이 가장 좋게 나왔다.

그림 4는 각 아키텍처의 주 버퍼 적중률을 보여주고 있는데, 실제로 주 버퍼의 적중률은 SSD의 크기가 클수록 비슷해진다. 원래 2Q 알고리즘의 경우, A1out의 비율을 아주 낮게 잡아서 핫 페이지들만 Am 주 버퍼에 유입될 수 있도록 하였다. 그런데, A1out, 즉 플래시 SSD 확장 버퍼의 크기를 키우는 경우 핫 페이지가 아

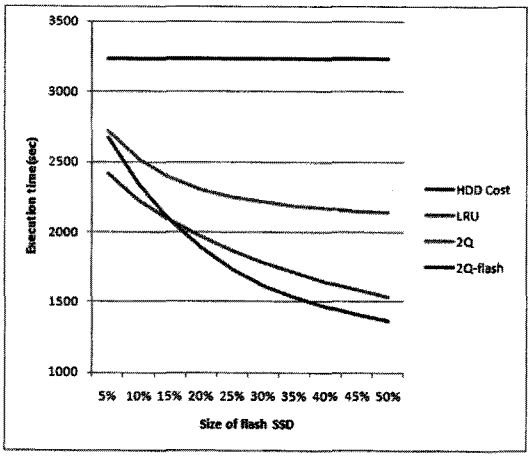


그림 3 전체 수행 시간

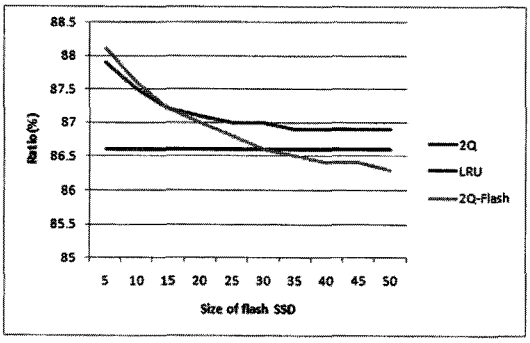


그림 4 주 버퍼 적중률

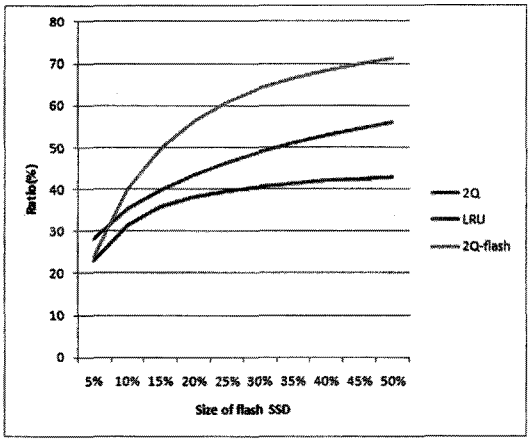


그림 5 확장 버퍼 적중률

닌 페이지들이 Am 주 버퍼로 유입될 확률이 높아지고 따라서 주 버퍼의 적중률이 조금씩 낮아지게 된다. 그럼에도 불구하고, A1out의 확장 버퍼로 플래시 SSD를 사용함으로써 주 버퍼로 유입되는 페이지들에 대한 하드디스크와 플래시 SSD의 접근 속도 차이 때문에 성능 향상이 훨씬 크다(그림 3, 5 참조).

그림 5는 주 버퍼에 존재하지 않는 페이지들 중에서 확장 버퍼에서 찾아지는 비율, 즉 확장 버퍼 적중률을 보여주고 있는데, 이 확장 버퍼 적중률과 그림 3의 수행시간이 밀접한 상관관계가 있음을 알 수 있다. LRU와 2Q는 확장 버퍼 크기가 각 35%, 20% 근처에서 수행시간과 확장 버퍼의 적중률이 정체되었다. 반면 2Q-Flash의 경우 확장 버퍼 크기가 40%에 이를 때까지 적중률이 상승했고, 수행시간 역시 감소했다. 확정 버퍼를 사용하는 아키텍처의 경우, 시스템 전체의 성능은 주 버퍼 적중률보다 확장 버퍼 적중률에 더 큰 영향을 받음을 알 수 있다.

각 아키텍처의 전체 수행시간에 가장 큰 비중을 차지하는 것이 하드디스크 I/O 시간이다. 기존 방법에서 주 버퍼의 적중률이 높아지면 필요한 하드디스크 I/O가 감소하고 따라서 전체 성능이 올라간다. 하지만 플래시 SSD를 확장 버퍼로 사용하는 경우 주 버퍼에서 없는 페이지의 경우 확장 버퍼에서 적중되는 경우 하드디스크보다는 훨씬 빠른 접근 시간을 보장하게 된다. 2장에서 기술한 바와 같이, 플래시 SSD를 확장 버퍼로 사용하는 경우 웜 페이지 적중률을 높여 하드디스크 I/O를 높이는 것이 목적이다. LRU 기법은 지역성이 높은 페이지부터 낮은 페이지까지 실시간으로 정렬을 하기 때문에 웜 페이지가 자동으로 분류가 된다. 따라서 확장 버퍼 적중률이 올라가 성능이 좋았다. 반면 2Q 기법은 주 버퍼의 적중률이 상대적으로 높음에도 불구하고

A1out에서 지역성을 구분하는 메커니즘이 없어 확장 버퍼 적중률이 떨어져 성능이 낮았다. 2Q-Flash 기법은 2Q의 높은 주 버퍼 적중률과 LRU의 높은 확장 버퍼 적중률을 모두 가졌기 때문에 성능이 가장 좋은 것으로 분석된다.

3.2.2 타 아키텍처와 가격 대비 성능 비교

그림 6은 2Q-Flash 기법에 투입된 플래시 SSD의 확장 버퍼 와 동일한 비용을 갖는 두 가지 다른 아키텍처, 즉, 1) 동일한 가격의 DRAM을 주 버퍼로 추가한 경우와 2) 동일한 가격의 하드디스크를 추가한 경우와 성능을 비교한 결과다. 현재 MLC 방식의 플래시 SSD와 DRAM의 가격 차이는 GB당 약 1:10 정도로 DRAM이 더 비싸다[8]. 따라서 그림 6에서 가로축의 50%의 플래시 SSD에 대응하는 DRAM의 비율은 5%로 가정한다. 실험 결과, 플래시 SSD의 크기가 클수록 상대적인 차이가 증가하고 본 실험 조건에서 최대 40% 정도의 차이를 보인다. 원래 실험에서 주 버퍼의 크기를 데이터베이스 전체 크기의 4% 정도로 설정하였는데, 실제 TPC-C의 경우 접근 지역성 때문에 주 버퍼의 크기가 4% 이상의 경우, 추가적으로 주 버퍼를 늘려도 급격한 성능 개선은 기대하기 힘들고 추가된 주 버퍼의 크기에 주 버퍼 적중률이 조금씩 선형적으로 증가할 뿐이다[9]. 반면 확장 버퍼는 상대적으로 크기가 크고 하드 디스크에 비해 성능이 월등히 좋기 때문에 확장 버퍼 적중률이 증가하면서 성능이 향상된다.

다음으로, 플래시 SSD 대신 2개의 하드디스크를 RAID-0로 구성한 방법과 비교하면, 이 경우 역시 일정 크기 이상 플래시 SSD의 크기가 커지면 하드디스크를 추가한 경우보다 성능 차이가 계속해서 커진다. 엔터프라이즈 환경에서는 데이터 전체 용량과 관계없이 IOPS (IO per Second, 초당 IO 수)를 높이기 위해 빠른 하드 디스크 다수를 병렬로 연결하여 사용하는데, 표 1의 가격처럼 MLC 기반의 플래시 SSD 가격이 고성능 하드디스크 가격보다 더 싸기 때문에 IOPS 측면에서 플래시 SSD 확장 버퍼 방식이 더 효율적임을 알 수 있다. 플래시 메모리의 가격은 당분간 지속적으로 하락할 것이며, 플래시 SSD 컨트롤러 기술의 계속적인 발전으로, 본 논문에서 제안한 아키텍처가 다른 두 가지 아키텍처에 비해 가격 대비 성능 측면에서 상대적인 우월성이 더 커질 것이다.

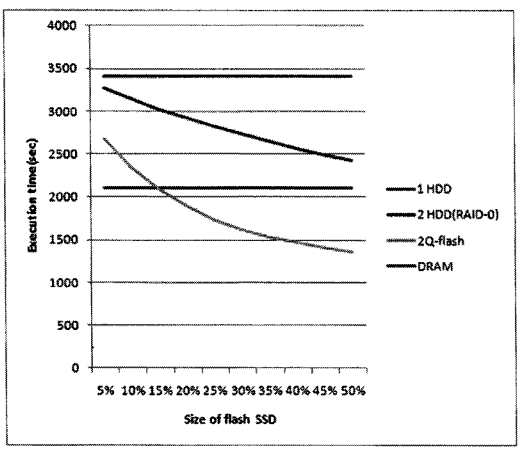


그림 6 가격대비 성능 비교

4. 결론 및 향후 연구

본 논문에서는 플래시 SSD를 데이터베이스에서 DRAM으로 구성된 주 버퍼의 확장 버퍼로 사용하는 새로운 접근법을 제시하였다. 플래시 SSD를 확장 버퍼로 사용함으로써, 동일한 비용을 필요로 하는 다른 접근법들에 비해 훨씬 더 나은 성능을 보임을 알 수 있었다. 또한 플래시 SSD를 확장 버퍼로 사용한 접근에서 주 버퍼의 적중률 보다 확장 버퍼의 적중률이 성능에 직접적인 영향을 주는 것을 알 수 있었다.

앞으로 우리는 보다 다양한 워크로드(예를 들어, 데이터베이스 이외의 분야)를 통해 확장 버퍼의 유용성을 검증하고, 확장 버퍼 아키텍처에 적합한 새로운 버퍼 교체 알고리즘 등에 대한 연구도 수행할 예정이다.

참 고 문 헌

[1] Intel Corporation. Oltp performance comparison: Solid state drives vs. hard disk drives. Test report, Jan. 2009.

[2] S.-W. Lee, B. Moon, and C. Park. Advances in flash memory ssd technology for enterprise database applications. *Proceedings of SIGMOD*, 2009.

[3] M. Canim et al. An Object Placement Advisor for DB2 Using Solid State Storage. Proceedings of the VLDB Endowment, vol.2, no.2, pp1318-1329, 2009.

[4] I. Koltsidas and S. D. Viglas. Flashing Up the Storage Layer. Proceedings of VLDB, 2008.

[5] E. J. O'neil et al. The LRU-K Page Replacement Algorithm For Database Disk Buffering. Proceedings of SIGMOD, 1993.

[6] T. Johnson and D. Shasha. 2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm. Proceedings of VLDB, 1994.

[7] Oracle, Oracle I/O Numbers Calibration Tool. http://www.oracle.com/technology/software/-tech/orion/.

[8] DRAMeXchange, Price Quites. http://www.dramexchange.com/Price/NationalDramDetail.aspx

[9] T. F. Tsuei et al., Database buffer size investi-

gation for OLTP workloads. Proceedings of SIGMOD, pp.112-122, 1997.
[10] 김성탄, 심도윤, 박장우, 이상원, 2Q-Flash: 플래시 SSD를 사용한 확장 버퍼, 전자공학회 하계학술대회, 2010.

심 도 윤
2009년 순천향대학교 정보기술공학부 졸업(학사). 2009년~현재 성균관대학교 임베디드소프트웨어학과 석사과정. 관심분야는 데이터베이스 시스템, 플래시메모리

박 장 우
2009년 건국대학교 소프트웨어과 졸업(학사). 2009년~현재 성균관대학교 임베디드소프트웨어학과 석사과정. 관심분야는 데이터베이스 시스템 구조 및 개발, 플래시메모리

김 성 탄
2008년 성균관대학교 컴퓨터공학과 졸업(학사). 2010년 성균관대학교 임베디드소프트웨어학과 졸업(석사). 2010년 8월 삼성전자 반도체. 관심분야는 데이터베이스 시스템, 플래시메모리, SSD

이 상 원
정보과학회논문지 : 데이타베이스
제 37 권 제 4 호 참조

문 봉 기
정보과학회논문지 : 데이타베이스
제 37 권 제 3 호 참조