NAND 플래시 메모리를 위한 지역성 기반의 로그 블록 교체 기법

이성진*, 김영진*, 김지홍*, 신동군**
* 서울대학교 컴퓨터공학부
** 성균관대학교 정보통신공학부
e-mail : chamdoo@davinci.snu.ac.kr

A Locality-Based Log Block Replacement Technique for NAND Flash Memory

SungJin Lee*, YoungJin Kim*, Jihong Kim*, Dongkun Shin**
* School of Computer Science and Engineering, Seoul National University
** School of Information and Communications Engineering, Sungkyunkwan University

요 약

플래시 메모리는 휴대폰, MP3 플레이어, 개인휴대정보단말기(PDA), 휴대용 멀티미디어 플레이어(PMP), 디지털 카메라 및 캠코더와 같은 이동성이 강한 소형기기에서 가장 많이 사용되는 저장 매체이다. 최근 대용량의 값싼 플래시 메모리가 등장하면서 랩톱이나 데스크톱과 같은 일반적인 컴퓨팅 환경을 지닌 기기들에서도 그 사용이 확대되고 있는 추세이다. 플래시 메모리가 보다 범용적인 저장 장치로 사용되기 위해서는 일반 컴퓨팅 환경에서의 복잡한 작업 부하에서도 우수한 성능을 제공할 수 있는 플래시 변환 계층(Flash Translation Layer)이 반드시 필요하다. 아쉽게도 현재까지 연구된 FTL 기법들은 소형기기의 단순한 작업 부하에 알맞도록 설계되어 있으며, 일반 컴퓨팅 환경과 같이 복잡한 작업 부하를 지닌 환경에서는 우수한 성능을 제공하지 못한다는 단점을 가지고 있다. 본 논문에서는 일반 컴퓨팅 환경의 복잡한 작업 부하에 대해서도 우수한 가비지 수집 성능을 제공하는 새로운 로그 블록 교체 기법을 제안하였다. 실험을 통한 평가 결과, 제안한 기법은 기존 기법 대비 평균 35% 정도의 가비지 수집 부하를 감소시키는 것으로 나타났다.

1. 서론

현재 플래시 메모리는 휴대폰, MP3 플레이어, 개인정보단말기(PDA), 휴대용 멀티미디어 플레이어(PMP), 디지털 카메라 및 캠코더와 같은 이동성이 강한 소형기기에서 가장 많이 사용되는 저장 매체이다. 이는 플래시 메모리가 지닌 특성들 즉, 저전력성, 비휘발성, 고성능, 물리적 안정성 및 휴대성 등이 소형기기를 위한 저장 장치로 적합하기 때문이다. 최근 대용량의 값싼 플래시 메모리가 등장하면서 랩톱과 같은 일반적인 컴퓨팅 환경을 지닌 기기들에서도 플래시 메모리의 사용이 확대되고 있는 추세이다[1].

하드디스크와는 달리 플래시 메모리는 덮어 쓰기 연산을 지원하지 못한다는 단점을 가진다. 이런 제약으로 인해 플래시 메모리의 특정 페이지에 저장된 데이터를 수정해야 할 경우 먼저 새로운 데이터를 빈 페이지에 저장한 후 이전 데이터를 가진 페이지를 무효화 시켜야만 하는 과정이 필요하다. 이러한 이유로 플래시 메모리 기반의 저장 장치는 파일 시스템으로부터 받은 논리 페이지 번호를 실제 플래시 메모리 내의 물리 페이지 번호로 변환 시켜 주는 주소 사상 기법을 필요로 하며, 또한 무효화된 페이지들을 재사용할 수 있게 삭제 해주는 일종의 가비지 수집 정책을 필요로 한다. 이를 위해 파일 시스템과 플래시 메모리 장치 사이에 위치한 플래시 사상 계층(Flash Translation Layer, 이하 FTL)이라 불리는 소프트웨어 모듈이 일반적으로 사용된다[2, 3]. 플래시 메모리의 높은 삭제 비용으로 인해, 전반적인 I/O 성능은 삭제 연산의 횟수에 의존적이며, 따라서 많은 FTL 기법들은 가비지 수집 시 발생하는 삭제 연산 횟수를 얼마나 많이 줄일 수 있을지에 대해 초점을 두고 있다.

기존 FTL 기법들은 소형기기 환경에서는 좋은 성능을 보여주지만, 일반 컴퓨팅 환경에서 사용되는 SSD 와 같은 플래시 기반의 드라이브에 적용될 경우 매우 낮은 성능을 보이는 문제점을 지니고 있다. 일반적으로 소형기기에서의 쓰기 패턴은 다수의 연속적인 쓰기 요청과 소수의 임의쓰기로 이루어진 반면 일반 컴퓨팅 환경에서의 쓰기 패턴은 소형기기에 비해 매우 복잡하며, 특히 높은 지역성과 다수의 임의 쓰기로 이루어진다. 따라서 이러한 성능 저하는 기존 FTL 기법이 일반 컴퓨팅 환경에서의 작업부하의 특징을 적절하게 활용하지 못함에 원인을 둔다.

본 논문에서는 작업 부하의 지역성을 적절하게 활용함으로써 일반 컴퓨팅 환경의 작업 부하에서도 우수한 가비지 수집 성능을 제공하는 지역성 기반의 로그 블록 교체 기법(Locality-Based Log Block Replacement Technique, 이하 L^2BR)을 제안한다. 시뮬레이션을 통한 성능 평가 결과 본 논문에서 제안한 기법은 기존 FTL 기법 대비 약 35% 정도의 가비지 수집 부하를 감소시킬 수 있었다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 논문이 기반을 두고 있는 로그 버퍼 기반의 FTL 기법에 대하여 설명하며, 3 장에서는 높은 지역성을 지닌 작업 부하에서의 로그 버퍼의 특징에 대하여 설명한다.

4 장에서는 제안한 로그 블록 교체 기법에 대하여 논하며, 5 장에서는 실험 결과에 대하여 분석한다. 이후 6 장에서는 본 연구의 결론을 내린다.

2. 관련 연구

2.1 로그 버퍼 기반 FTL 기법

현재까지 제안된 FTL 기법들은 주소 사상 방식에 따라 페이지 기반[4], 블록 기반[5], 그리고 혼합 기반[6,7]의 기법으로 분류된다. 혼합 기반의 기법은 적은 메모리 자원을 요구하면서도 높은 성능을 제공하기 때문에 많은 소형기기에서 사용되며, 이중 BAST[6]와 FAST[7]기법과 같은 로그 버퍼 기반의 FTL 기법이 널리 알려져 있다. 본 논문은 로그 버퍼 기반의 FTL 기법, 특히 FAST 기법에 기반을 두고 있기 때문에 이후 FAST 기법을 위주로 설명을 하도록 한다.

로그 버퍼 기반의 FTL 기법은 플래시 블록들을 데이터 블록과 로그 버퍼로 나누어 관리한다. 데이터 블록은 일반적인 데이터 저장을 위해 사용되며 실제 사용자에게 보여지는 공간이다. 로그 버퍼는 작은 개수의 로그 블록으로 구성되어 있으며 덮어 쓰기 요청 시 해당 데이터를 임시로 저장하기 위한 공간으로 사용된다. 일반적으로 데이터 블록은 블록 단위로 관리되며 로그 버퍼는 페이지 단위로 관리된다.

이러한 혼합 기반의 주소 사상을 통해, 로그 버퍼 기반의 FTL 기법은 추가적인 I/O 연산 없이 각각의 읽기 및 쓰기 연산을 수행할 수 있다는 장점을 가진다. 하지만 로그 버퍼 공간이 고갈될 경우 FTL 은 합병 연산이라 불리는 값 비싼 가비지 수집 연산을 통해 새로운 빈 공간을 확보해야만 한다. 합병 연산은 크게 1) 교체될 로그 블록을 선택하는 과정 2) 교체될 로그 블록과 연관된 데이터 블록들을 선정하는 과정 3) 선택된 플래시 메모리 블록 내의 유효한 페이지들을 빈 블록에 복사하는 과정 4) 이전 블록들을 삭제하고 주소 사상 테이블을 수정하는 과정으로 이루어진다. 일반적으로 과정 3)과 4)에서의 복사 및 삭제 연산이 전반적인 합병 연산의 비용을 결정한다.

합병 연산은 크게 교체 합병과 완전 합병으로 분리된다. 교체 합병은 교체될 로그 블록 내에 페이지들이 논리적으로 연속되어 쓰여진 경우에만 수행 가능하며, 페이지 복사 연산 없이 한번의 삭제 연산만을 요구하기 때문에 비용이 싸다는 장점을 가진다. 반면 완전 합병은 교체될 로그 블록이 다수의 데이터 블록들과 연관될 수 있으며, 교체될 블록내의 페이지들 역시 논리적으로 연속되어 쓰여지지 않을 수 있기 때문에 복사 및 삭제 비용이 매우 높다. 합병 연산 과정에 대한 상세한 설명은 [6,7]을 참고하기 바란다.

2.2 기존 기법의 문제점

일반 소형 기기에서의 작업 부하는 대부분 연속적인 쓰기 요청과 매우 작은 수의 임의 쓰기로만 구성되어 있다. 따라서 값싼 교체 합병 연산의 비중이 값비싼 완전 합병 연산에 비해 매우 높다. 이는 로그 버퍼 기반의 FTL 기법은 소형기기에서 우수한 가비지 수집 성능을 제공해 주는 이유이다. 반면 임의 쓰기와 지역성이 높은 작업 부하 환경에서는 값 비싼 완전 합병 연산의 비중이 크게 증가하며, 따라서 가비지 수집에 따른 부하가 매우 커지게 된다. 본 논문에서는 일반 컴퓨팅 환경에서 높은 비중을 차지하는 완전 합병 연산의 비용을 감소시킴으로써 전반적인 가비지 수집 성능을 향상 시키고자 한다

3. 지역성 높은 작업 부하에서의 로그 버퍼의 특징

3.1 합병 비용 모델

본 논문에서는 완전 합병에 의해 발생하는 추가적인 연산 비용을 합병 비용(merge cost)라고 정의 내린다. 식(1)에서 볼 수 있듯이, 합병 비용은 페이지 복사 비용(page migration cost)과 블록 삭제 비용(block erase cost)으로 이루어진다. 복사 비용은 합병 연산 중 유효한 페이지들을 빈 블록으로 복사할 때 발생하며 블록 삭제 비용은 페이지 복사 후 데이터 블록과 교체될 로그 블록을 삭제하는 과정에서 발생한다.

$$merge\ cost = page\ migration\ cost + block\ erase\ cost \quad (1)$$

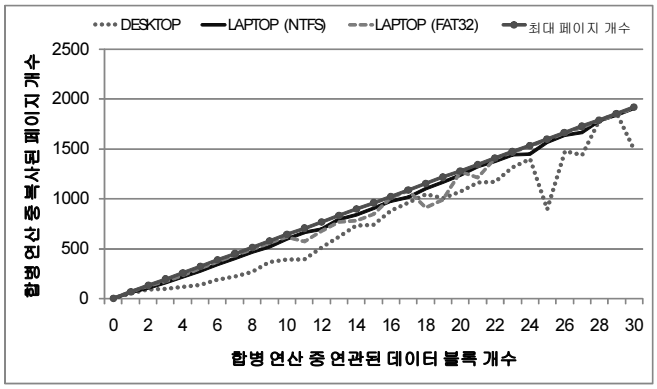
$$= \sum_{i=0}^{N} \{P(b_i) \times C_c\} + C_e \times (N+1) \quad (2)$$

$$\approx \sum_{i=0}^{N} \{P_{\max} \times C_c + C_e\} + C_e \quad (3)$$

$$\approx N \cdot c + C_e \quad (4)$$

완전 합병 시 N 개의 데이터 블록들$(b_1,...,b_n)$이 교체될 로그 블록에 연관되어 있을 경우, 합병 비용은 식(2)와 같아진다. 여기서 $P(b_i)$는 데이터 블록 b_i 에 대응되는 유효한 페이지들의 개수를 나타내며, C_c 는 하나의 페이지 복사 시 발생하는 비용을, C_e 는 하나의 블록을 삭제할 때 발생하는 비용을 나타낸다. 이때 데이터 블록의 개수 N 과 각 데이터 블록에 해당되는 $P(b_i)$의 값은 어떤 로그 블록이 교체 대상으로 되었느냐에 따라 달라지기 때문에 실제로 측정하기 매우 힘들다. 대신 우리는 0 부터 n 까지의 i 에 대하여 각 $P(b_i)$는 P_{max} 와 동일한 값을 가진다 가정하고 식(2)를 식(3)로 근사 시켰다. 여기서 P_{max} 는 하나의 플래시 블록당 페이지 개수를 나타낸다. 일반적으로 $P(b_i)$는 P_{max} 보다 같거나 작을 수 밖에 없다. 만약 해당되는 데이터 블록 b_i 의 모든 페이지가 사용 중 이라면 $P(b_i)$는 P_{max} 와 같아진다. 반면 데이터 블록 b_i 의 대부분이 빈 공간이라면 $P(b_i)$는 매우 작을 것이다. 우리는 식(3)의 합병 비용 근사가 정확하지를 확인하기 위해 여러 작업 부하를 대상으로 검증 작업을 수행했다.

그림 1 은 합병 연산 시 실제로 복사된 페이지의 개수와 우리가 가정한 페이지 복사 개수를 비교한 것이다. 가로 축은 합병 연산 시 연관된 데이터 블록의 개수를 나타내며, 반면 세로축은 합병 연산 시 실제로 복사된 페이지의 개수를 보여준다. 그림에서 '최대 페이지 개수'는 합병 연산 시 P_{max} 만큼의 페이지 복사가 발생했다고 가정했을 때의 페이지 복사 횟수이고 나머지는 실제 복사 연산이 일어난 횟수를 나타낸



(그림 1) 합병 연산 중 발생한 실제 페이지 복사 횟수

다. 그림 1 에서 나타내듯 식(3)에서의 가정이 합병 비용을 정확하게 근사하고 있음을 알 수 있다.

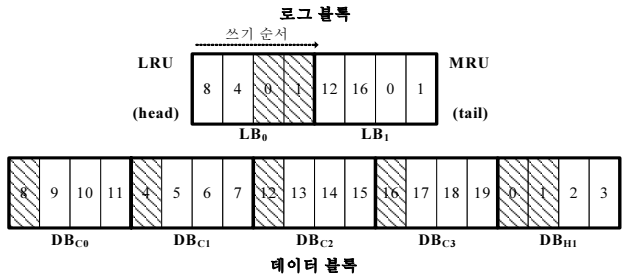
식(3)에서 P_{max}, C_e, 그리고 C_c 는 모두 상수이기 때문에 우리는 합병 비용 모델로서 보다 간단한 식(4)를 사용할 수 있다. N 은 어떤 로그 블록을 교체될 블록으로 선택했는지에 따라 결정되며, 결국 합병 비용은 합병 연산 중 어떤 블록을 교체될 블록으로 선택했는지에 의존적이 된다. 따라서 우리는 앞으로 식(4)의 상수 c 와 C_e를 각각 1 과 0 으로 가정할 것이다.

3.2 합병 비용과 로그 버퍼 지역성간의 관계

본 절에서는 지역성이 높은 작업 부하 수행 시 나타나는 로그 버퍼와 합병 비용간의 특징에 대하여 설명한다. 그림 2 는 완전 합병 연산 시의 로그 버퍼와 데이터 블록의 상태를 보여주고 있다. 이 예제에서 FTL 은 두 개의 로그블록 LB_0 와 LB_1, 다섯 개의 데이터 블록 DB_{C0}~ DB_{C3}, DB_{H1} 으로 구성되어 있다. DB_{C0}~ DB_{C3} 는 지역성이 낮은 페이지들이 저장된 반면, DB_{H1} 은 지역성이 높은 페이지들이 저장되어 있다. 우리는 각 페이지에 대한 쓰기 요청이 (8, 4, 0, 1, 12, 16, 0, 1) 순서로 발생하고 요청된 순서대로 로그 버퍼의 왼쪽에서 오른쪽으로 쓰여진다고 가정한다. 페이지 0 과 1 은 높은 지역성을 지니고 있기 때문에 다른 페이지들에 비하여 여러 번의 쓰기가 요청되었다.

본 예제에서 만약 FTL 이 LB_0 를 교체될 블록으로 선택하였을 경우 합병 비용은 2 가 된다. 이는 두 개의 데이터 블록 DB_{C0} 과 DB_{C1} 이 합병 연산에 개입되었기 때문이다. 반면 FTL 이 LB_1 을 교체될 블록으로 선택한 경우 3 개의 데이터 블록 DB_{C2}, DB_{C3} 그리고 DB_{H1} 이 합병 연산 시 연관되기 때문에 총 합병 비용은 3 으로 증가된다.

이 예제는 지역성이 높은 작업 부하 수행 시 로그 버퍼의 특징을 잘 보여준다. 첫 번째로 지역성이 보다 높은 페이지를 로그 버퍼에 오래 유지시키는 것이 합병 비용 감소에 도움이 된다. 지역성이 높은 페이지들은 자주 쓰여지기 때문에 이전에 기록된 페이지들이 로그 버퍼에서 쫓겨나기 전에 다시 로그 버퍼에 기록된다. 이때 FTL 은 데이터 일관성 보장을 위해 기존의 데이터를 가지고 있는 페이지를 반드시 무효화 시켜야만 하며, 따라서 지역성이 높은 작업 부하를 수행할 경우 로그 버퍼 내에 많은 수의 무효화된



(그림 2) 로그 블록의 덮어 쓰기 과정

페이지들이 생성되게 된다. FTL 은 합병 연산 시 무효화된 페이지들을 무시하기 때문에 교체될 로그 블록 내의 무효화된 블록의 개수가 많으면 많을수록 연관된 데이터 블록의 개수는 점차 감소하게 된다.

두 번째로 로그 버퍼의 동작은 LRU 리스트의 동작과 유사하다. 지역성이 높은 페이지들은 로그 버퍼의 끝(tail)에 빈번하게 다시 쓰여지며 동시에 이전 데이터를 저장하고 있는 페이지들은 무효화를 시키기 때문에, 로그 버퍼의 끝(tail)과 가까운 로그 블록들은 지역성이 높은 페이지를 저장하고 있을 가능성이 높은 반면, 로그 버퍼의 처음(head)과 가까운 로그 블록들은 지역성이 낮은 페이지들과 무효화된 페이지들을 가지고 있을 가능성이 높아지게 된다. 따라서 로그 버퍼의 처음에 위치한 로그 블록(LB_0)을 교체될 블록으로 선택하는 것이 합병 비용 감소에 보다 유리하다.

4. 로그 블록 교체 기법

교체될 블록을 선택할 때 우리가 고려해볼 수 있는 첫 번째 접근 방법은 LRU 방식을 사용하는 것이다. 로그 버퍼가 마치 LRU 리스트와 유사하게 동작하기 때문에, 이 기법은 매우 작은 계산 비용을 요구하는 반면 비교적 우수한 합병 비용 감소를 기대할 수 있다. 하지만 LRU 기법은 합병 비용을 직접적으로 고려하고 있지 않기 때문에, 가비지 수집에 따른 부하를 줄이기 위한 기회들을 효과적으로 활용하지 못한다. FAST 기법은 라운드 로빈(round robin) 방식의 로그 블록 교체 기법을 사용하는데[7], 로그 버퍼의 특성상 이는 LRU 와 동일하게 동작한다.

로그 버퍼 내에서 가장 합병 비용이 낮은 블록을 교체될 블록으로 선택하는 greedy 기법 역시 알맞은 대안 기법으로 제시될 수 있다. greedy 기법을 이용할 경우 합병 비용이 낮은 로그 블록을 우선적으로 제거함으로써 전반적 합병 비용을 낮출 수 있으며, 더불어 합병 비용이 충분히 감소될 때까지 합병 비용이 높은 로그 블록을 로그 버퍼 내에 유지 시킴으로써 향후 합병 비용을 감소 시키는데 기여할 수 있다. 직관적으로 보면 greedy 기반의 교체 기법은 합병 비용 감소에 이득이 될 것 같다. 하지만 오히려 greedy 기법은 특정한 경우에 있어서 합병 비용을 증가시키는 경향을 가지고 있다. 이러한 문제점은 greedy 기법이 향후 합병 비용이 크게 감소될 가능성이 높은 로그 블록을 교체될 블록으로 선택하기 때문에 나타난다.

이러한 문제점들을 해결하기 위해, 본 논문에서 제안한 L^2BR 기법은 교체될 블록 선택 시 합병 비용과

지역성을 함께 고려한다.

$$cleaning\ factor_i = freq_i \bullet merge\ cost_i \quad (5)$$

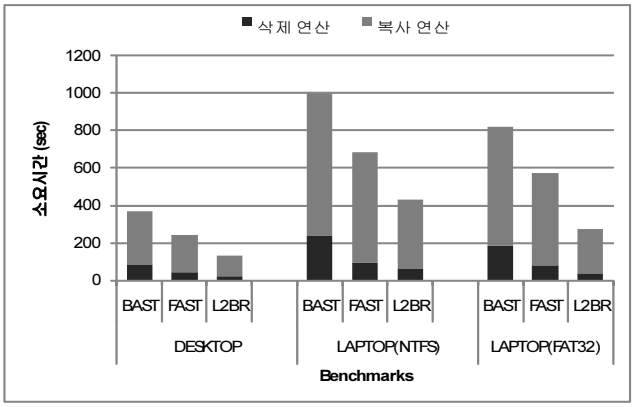
여기서 $freq_i$와 $merge\ cost_i$는 각각 로그 블록 i에 남아 있는 페이지들의 접근 빈도수의 합과 합병 비용을 나타낸다. 우리는 로그 버퍼 내에 가장 작은 *cleaning factor*를 가진 로그 블록을 교체될 블록으로 선택한다. 식(5)는 높은 지역성을 지닌 로그 블록에게 높은 가중치를 적용함으로써 해당 블록이 교체될 블록으로 선정되는 것을 막는다. 반면 유사한 지역성을 지닌 로그 블록의 경우에는 합병 비용이 작은 블록을 우선 교체 대상으로 정한다.

Cleaning factor 계산을 위해 FTL 은 내부적으로 접근 빈도 테이블과 합병 비용 테이블을 유지 한다. 접근 빈도 테이블은 제한된 개수의 페이지 요소들로 구성되어 있는데 각 요소들은 자주 접근된 페이지들의 주소와 길이, 접근 횟수를 기록하고 있다. 합병 비용 테이블은 각 로그 블록의 합병 비용을 기록한다. 각 테이블들은 모두 SRAM 에 유지되기 때문에 테이블 갱신에 따른 부하는 무시할만한 수준이다.

5. 실험 결과

제안한 기법의 성능 평가를 위해 우리는 작업 부하 기반의 FTL 시뮬레이터를 개발하였다. 성능 비교를 위해 두 가지 로그 버퍼 기반의 FTL 기법인 BAST 기법과 FAST 기법을 추가로 구현하였다. 사용한 플래시 메모리 모델은 삼성 K9F1G08X0A NAND 플래시 메모리이다. 기법 평가를 위해 수집된 모든 작업 부하들은 실제로 많이 사용되는 응용들(문서 편집기, 웹 브라우저, MP3 플레이어, 게임 등)을 Microsoft Windows XP 기반의 노트북 및 데스크톱에서 수행하며 추출되었다. 가비지 수집에 따른 부하는 각 읽기, 쓰기, 지우기 연산 별 수행 횟수에 해당되는 소요시간 값을 곱한 후 이를 합하여 계산하였다.

그림 3 은 로그 버퍼의 크기가 256 MB(=2048 개의 로그 블록)일 때 각 FTL 기법 별 가비지 수집 부하를 보여준다. 만약 NAND 플래시 메모리가 32 GB 용량의 SSD 인 경우, 로그 버퍼가 전체 사용공간에서 차지하는 비율은 1% 이하로 매우 작다. 평가된 모든 기법들 중 L^2BR 는 가장 우수한 가비지 수집 효율을 보여주었다. L^2BR 은 FAST 기법과 비교하여 약 32~40%의 가비지 수집 부하를 감소시켰다. BAST 기법은 모든 FTL 기법 중 가장 나쁜 성능을 보여주는데 이는 높은 지역성과 다수의 임의 쓰기가 있는 작업 부하를 효과적으로 활용 못하기 때문이다. FAST 기법은 BAST 기법에 비해 개선된 가비지 수집 성능을 보였지만, 합병 비용과 지역성 고려 없이 단순한 LRU 방식에 따라 로그 블록을 교체했기 때문에 L^2BR 에 비해 낮은 성능을 보였다. 더불어 NTFS 와 FAT 등 상이한 파일 시스템에 따른 가비지 수집 성능의 차이는 거의 발생하지 않았다. 다음으로 우리는 각 FTL 기법들이 요구하는 메모리 공간의 크기를 비교했다. 플래시의 총 용량이 32 GB 이고 로그 버퍼의 크기가 256



(그림 3) 가비지 수집 부하

MB 인 경우 L^2BR 은 총 1.99 MB 의 메모리 공간을 필요로 한 반면 BAST 와 FAST 기법은 모두 1.5 MB 의 메모리 공간을 요구 했다. BAST 와 FAST 기법은 대부분의 메모리 공간을 페이지 사상 테이블(512 KB)과 블록 사상 테이블(1 MB)을 위해 사용했다. 반면 L^2BR 은 접근 빈도 테이블(410 KB)과 합병 비용 테이블(96 KB)을 위해 추가적인 메모리를 필요로 했다. 하지만 이는 사상 테이블의 크기는 실제 시스템에 적용되는데 큰 장애가 되지 않을 정도로 충분히 작았다.

6. 결론

본 논문에서는 작은 자원을 요구지만 일반 컴퓨팅 환경과 같은 복잡한 작업 부하 환경에서도 훌륭한 가비지 수집 성능을 제공하는 새로운 로그 블록 교체 기법인 L^2BR 을 제안하였다. 시뮬레이터를 통해 실험해본 결과 L^2BR 은 FAST 기법 대비 평균 35%의 가비지 수집 비용을 감소시킬 수 있었다.

감사의 글

이 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사 드립니다. 이 논문은 2007 년도 두뇌한국 21 사업에 의하여 지원되었으며, 또한 2007 년도 정부(과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구(No. R0A-2007-000-20116-0)입니다.

참고문헌

[1] G. Lawton, “Improved flash memory grows in popularity,” IEEE Computer, vol. 39, no. 1, pp. 16-18, 2006.
[2] Intel Corporation. “Understanding the flash translation layer (FTL) specification,” http://developer.intel.com/.
[3] CompactFlash Association. http://www.compactflash.org
[4] A. Kawaguchi, S. Nishioka, and H. Motoda. “A flash-memory based file system,” In Proc. of the USENIX Winter Technical Conference, pp. 155-164, 1995
[5] A. Ban. “Flash file system,” United States Patent, no. 5,404,485, April 1995.
[6] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, “A space-efficient flash translation layer for compact flash systems,” IEEE Transactions on Consumer Electronics, vol. 48, no. 2, pp. 366-375, 2002.
[7] S.-W. Lee, W.-K. Choi, D.-J. Park, “FAST: an efficient flash translation layer for flash memory,” The 1st International Workshop on Embedded Software Optimization, 2006.