FTL(Flash Translation Layer)을 위한 비휘발성 메모리 기반 쓰기 버퍼의 활용

박성민*○ 정호영* 윤경훈* 차재혁* 강수용+
*한양대학교 정보통신공학과
+한양대학교 컴퓨터교육과
{syrilo○}@hanyang.ac.kr

Utilization of Non-Volatile RAM Write Buffer for FTL

Sungmin Park○ Hoyoung Jung* Jaehyuk Cha* Kyeonghoon Yoon* Sooyong Kang+
Dept. of *Informations and Communication Engineering, Hanyang University
Dept. of +Computer Science Education, Hanyang University

요 약

최근 낸드 플래시 메모리는 임베디드 저장 장치로서 많이 사용되고 있을 뿐만 아니라 플래시 메모리의 저장 용량의 대용량화로 하드 디스크를 대체하는 SSD(solid state disk) 같은 제품이 출시되고 있다. 플래시 메모리는 하드디스크에 비하여 저전력, 빠른 접근성, 물리적 안정성 등의 장점이 있지만 읽기와 쓰기의 연산의 불균형적인 비용과 덮어 쓰기가 안 되고 쓰기 전에 해당 블록을 지워야하는 부가적인 작업을 수행해야 한다. 이와 같은 특징은 플래시 메모리의 쓰기 성능을 저하 시키고 기존의 하드디스크를 대체하는 것을 어렵게 만든다. 이와 같은 플래시 메모리의 단점을 해결하기 위해서 본 논문에서 비휘발성 메모리와 플래시 메모리를 함께 사용하는 방법을 제안한다. 최근 MRAM, FeRAM, PRAM과 같은 차세대 메모리 기술의 발전과 배터리 백업 메모리의 가격 하락으로 인하여 비휘발성 메모리의 상품적 가치가 높아지고 있다. 하지만 아직까지 용량 대비 가격이 비효율적이기 때문에 소용량의 비휘발성 메모리를 활용하여 플래시 메모리의 쓰기 연산에 대한 단점을 보완하는 방법을 제안한다. 본 논문에서는 FTL 에서 비휘발성 메모리를 쓰기 버퍼로 이용한 여러 가지 버퍼 관리 정책을 실험하였고 각 관리 정책에 따른 플래시 메모리의 성능 향상을 측정하였다. 실험을 통하여 최대로 읽기의 횟수는 90% 감소, 쓰기 횟수는 33% 감소, 소거 횟수는 50% 감소 효과를 보였다.

1. 서 론

최근 낸드 플래시 메모리는 디지털 카메라, MP3 플레이어, 핸드폰, 개인 휴대폰 단말기(PDA) 등 임베디드 저장 장치로서 많이 사용 되고 있다. 뿐만 아니라 플래시의 저장용량도 급속도로 대용량화 되어가는 추세여서 기존 하드디스크를 대체하는 SSD(solid state disk) 같은 제품도 출시되고 있다. 플래시 메모리는 소비 전력이 적고 전원이 꺼지더라도 저장된 정보가 사라지지 않은 특성을 지닌 비휘발성 기억 장치로 하드디스크에 비해 데이터 접근 성능이 빠르고 크기가 매우 작다. 또한 물리적인 충격에 강하고 무게가 가볍다는 장점을 가지고 있다.

하지만 플래시 메모리는 하드 디스크와 달리, 특정 섹터(sector)에 쓰기 연산(write)을 하기 위해서는 해당 섹터가 깨끗한 상태로 비어져 있어야만 한다. 다시 말해서 데이터가 쓰여 져 있는 섹터에 대해 덮어쓰기(overwrite) 허용되지 않는다. 따라서 이런 경우에는 섹터를 포함하고 있는 블록 전체를 소거연산(erase)를 통해서 깨끗이 지우고 쓰기 연산을 수행해야 한다. 하지만 블록의 소거 연산에 걸리는 시간은 쓰기 연산과 읽기 연산에 비해 훨씬 크다. 이러한 플래시 메모리의 특징은 플래시 메모리가 기존의 하드디스크를 대체하는 것을 어렵게 만들고 플래시 메모리 전체의 성능을 저하를 가져온다. 즉 쓰기 연산에 의해 유발되는 소거연산의 횟수를 줄이는 것이 플래시 메모리 성능에 절대적인 영향을 미친다. 또한 플래시 메모리에서는 시스템 초기에 플래시 메모리의 주소 사상 테이블을 만들기 위해서 플래시 메모리 전체를 스캔하게 되어 부팅시간 길어지는 단점도 있다.

이와 같은 플래시 메모리의 문제점을 해결하기 위해서 본 논문에서는 비휘발성 메모리(Non Volatile RAM, NVRAM)와 플래시 메모리를 함께 사용하는 방법을 제안한다. 최근 차세대 메모리의 기술 발전으로 비휘발성 메모리의 상품화가 급속도로 진행되고 있지만 아직까지 용량 대비 가격이 비효율적이다. 따라서 소량의 비휘발성 메모리를 활용하여 플래시 메모리의 쓰기, 소거 연산을 효율적으로 줄이고 플래시 메모리의 주소 사상 테이블을 비휘발성 메모리에 저장함으로써 부팅 시간을 단축함으로써 플래시 메모리의 단점을 보완하는 방법을 제안한다. 본 논문에서는 소량의 비휘발성 메모리를 쓰기 버퍼로 활용 했을 때의 다양한 알고리즘을 제시하고 성능

을 평가 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로서 FTL의 기본 개념과 FTL 알고리즘의 대표적인 로그 블록 기법에 대하여 살펴본다. 3장에서 본 논문에서 제시하는 시스템 구조를 설명한다. 4장에서는 쓰기 버퍼로 활용했을 때의 다양한 알고리즘을 제시한다. 5장에서는 기존의 FTL 알고리즘과 비휘발성 메모리를 쓰기 버퍼로 활용했을 때의 알고리즘들의 성능을 비교하고 5장에서 결론을 맺고 향후 연구 과제를 설명한다.

2. 관련 연구

2.1. FTL

FTL은 플래시 메모리에서 논리적인 섹터/블록 번호를 물리적 섹터/블록 번호로 사상해주는 계층이다. FTL의 주소 사상 기법은 단순히 논리적 번호를 물리적 번호로 사상 시키는 것 뿐 만 아니라 합병연산(erase)을 하는 방법이나 시점 그리고 쓰기나 읽기 연산에 많은 영향을 미치기 때문에 어떤 알고리즘을 사용하느냐에 따라 성능 차이를 나타나게 된다. FTL 알고리즘의 예로는 SSR, Mitsubishi, FMAX, LEXAR, 로그 블록 기법[1]이 있으며 이 중에서 로그 블록 기법이 성능이 가장 우수하다. 따라서 로그 블록 기법에 대하여 간략히 설명하고 성능 평가에서 로그 블록 기법과 비교한다.

2.2. 로그 블록 기법

로그 블록 기법은 블록 사상과 섹터 사상을 사용하는 혼합 사상방식이다. 데이터 블록에 대해서는 블록 사상을 하고 in-place로 쓰기 연산을 수행한다. 소수개의 로그 블록에 대해서는 섹터 사상을 하고 out of place로 쓰기 연산을 수행한다. 데이터 블록에 이미 데이터가 쓰여져 있고 이 페이지에 대하여 쓰기 요청이 발생하면 데이터 블록에 로그 블록을 할당하고 덮어쓰기 연산을 수행한다. 로그 블록의 상태에 따라 합병연산이 발생하게 되는데 그 방법은 아래 그림1) 과 같다.

- switch : 가장 비용이 적게 드는 합병연산으로서 로그 블록이 in place로 쓰기가 요청 되었을 때 로그 블록을 데이터 블록으로 바꾸고 이전의 데이터 블록을 소거한다.
- switch copy : 다음으로 비용이 적게 드는 합병 연산으로서 로그 블록이 in place 로 쓰기가 되는 도중에 합병연산이 발생하게 되면 남은 페이지들 중에 데이터 블록에 유효한 데이터가 있는지 살펴보고 있으면 복사를 해 온 후에 로그 블록을 데이터 블록으로 바꾸고 이전의 데이터 블록을 소거한다.
- smart copy : 이 연산은 가장 비용이 많이 드는 연산으로 로그 블록이 out place로 쓰기가 일어난 경우에 합병연산이 발생하게 되면 데이터 블록과 로그 블록에 유효한 데이터를 새로운 블록에 복사 하는 방법이다. 새로운 블록에 유효한 데이터를 복사 한 후에 데이터 블록과 로그 블록은 소거 한다.

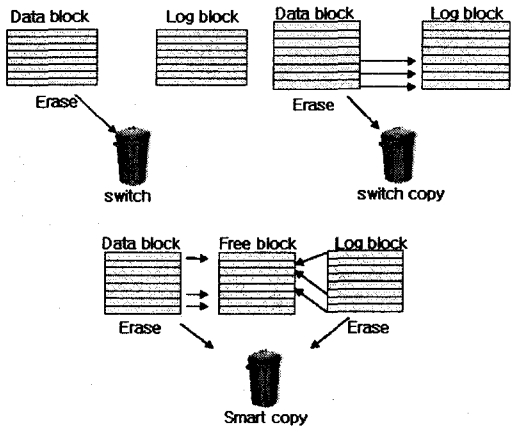

그림 1 합병 연산

합병 연산이 일어나는 시점은 로그 블록이 모두 사용되고 있을 때 로그 블록을 할당 받지 못한 데이터 블록이 새로운 로그 블록을 요청할 때나 어떤 데이터 블록이 할당된 로그 블록을 모두 사용하고 새로운 로그 블록을 요청할 때 발생하게 된다. 합병 연산을 살펴보면 Flash 메모리의 성능을 저하시키는 주요 원이 되는 데이터들은 smart copy 합병 연산을 일으키는 임의 쓰기 요청(samll random write)이다.

3. FTL-NVRAM 내부 구조

FTL-NVRAM 내부 구조는 비휘발성 메모리를 사상테이블을 저장하는 공간과 낸드 플래시 메모리의 쓰기 버

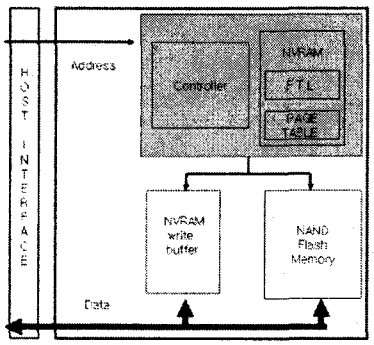

그림 2 FTL-NVRAM 내부 구조

퍼로 사용하는 공간으로 활용한다. 그림2)은 본 논문에서 제안하는 시스템 구조이다. 그림2)에서 PAGE TABLE은 비휘발성 메모리의 쓰기 버퍼에 저장된 데이터에 대해서는 섹터 사상을 하고 NAND 플래시 메모리에 저장된 데이터에 대해서는 블록 사상을 한다.

4. 쓰기 버퍼 관리 정책

FTL에서 비휘발성 메모리를 쓰기 버퍼로 이용했을 때 다양한 알고리즘의 성능은 다음 세 가지 조건에 좌우된다.

1. 비휘발성 메모리에서 쓰기 요청에 대하여 재 참조를 얼마나 시킬 수 있는가?
2. 비휘발성 메모리에서 쓰기 요청을 수용할 수 있는 빈 공간을 얼마나 확보할 수 있는가?
3. 비휘발성 메모리에서 낸드 플래시 메모리로 내려가는 데이터가 Flash 메모리에 발생 시키는 연산의 비용은 얼마인가?

위 세 조건을 만족시키기 위하여 각 알고리즘에서는 쓰기 데이터의 시간 참조성을 고려한 스케줄링 정책과 쓰기 버퍼에서 플래시 메모리로 내려 보내는 정책, 그리고 데이터가 내려갈 때 플래시 메모리에서의 합병 연산에 대하여 고려해야 할 것이다. 또한 비휘발성 메모리는 상대적으로 Flash 메모리 보다 용량이 매우 작다고 가정하기 때문에 효과적으로 비휘발성 메모리를 관리하기 위해서는 Flash 메모리의 성능을 저하시키는 주요 원이 되는 데이터들인 임의 쓰기 요청을 효율적으로 관리하는 정책이 필요할 것이다. 따라서 블록에 대하여 연속적인 쓰기 요청이 발생하는 데이터는 발견하여 우선적으로 교체 대상으로 삼는다. 블록에 연속적인 쓰기 요청을 발견하는 방법은 어렵지 않다. 각 블록에 연속 쓰기 변수를 두고 이 변수를 통해서 판별한다. 그림3)은 연속 쓰기 블록을 판별하는 알고리즘이다.

```
Sequential Detect :
Write request Sector x :
 offset  is x % PAGES_PER_BLOCK
 logical block is x / PAGES_PER_BLOCK
 if(offset is 0)
   set sequential_count of logical block to 1
 else if(offset is sequential_count of logical block)
   add sequential_count of logical block to 1
 else
   set sequential_count of logical block to 0
 endif
 if(sequential_count is PAGES_PER_BLOCK)
   this block is sequetial block ;
 endif
```

그림 3 sequential detect algorithm

4.1. 플래시 메모리와 비휘발성 메모리의 합병 연산

비휘발성 메모리에서 각 페이지들은 독립된 개체로서가 아니라 블록 단위로 간주된다. 비휘발성 메모리에서 교체 대상 블록이 내려갈 때 플래시 메모리에 새로운 블록을 할당하고 데이터 블록과 비휘발성 메모리 블록에 유효한 데이터를 새로운 블록에 복사 하는 방법을 사용한다. 새로운 블록에 유효한 데이터를 복사 한 후에 데이터 블록은 소거한다. 이 합병 연산은 로그 블록 기법의 smart copy 연산과 흡사하다. 따라서 이 연산을 NF_smart copy로 정의 한다. 이 연산은 smart copy 보다 소거 연산이 한번 적게 일어난다.

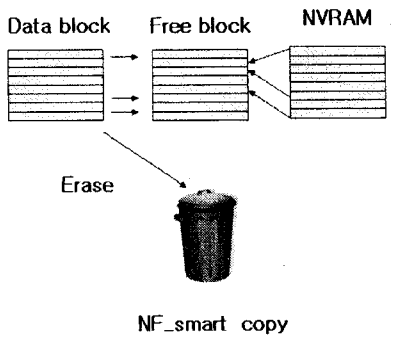

그림 4 NF_smart copy

4.2. 스케줄링 정책

4.2.1. BLOCK LRU

이 스케줄링 기법은 시간 참조성을 고려하여 블록 단위의 LRU 기법을 사용한다. 비휘발성 메모리에서 플래시 메모리로 페이지를 내려 보내야 할 때 교체 대상 블록에 속한 모든 페이지들을 플래시 메모리에 내려 보내는 정책이다.

4.2.2. PAGE NUM

이 스케줄링 기법은 블록에 속한 페이지가 많은 블록을 교체 대상으로 선정하고 이 블록에 속한 모든 페이지들을 플래시 메모리에 내려 보내는 정책이다. 이 정책에서는 시간 참조성은 무시되고 단순히 블록의 페이지의 개수만을 가지고 힙으로 구성한다.

4.2.3. BLOC LRU + CLEAN LIST

이 스케줄링 기법은 BLOCK LRU 기법에 PAGE NUM 속성을 적절히 혼합하기 위하여 블록에 대한 PAGE의 갯수가 일정한 개수가 넘으면 CLEAN LIST에 유지하고 CLEAN LIST에 속한 페이지가 우선적으로 교체 대상으로 선정하고 이 블록에 속한 모든 페이지들을 플래시 메모리에 내려 보내는 정책이다. 이 정책은 시간 참조성과 PAGE NUM의 두 요소를 적절히 조합하였다.(그림5)

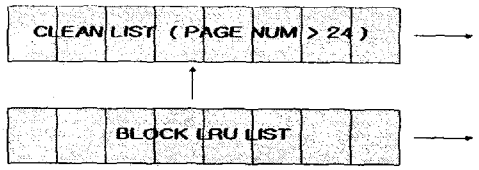

그림 5 BLOCK LRU + CLEAN LIST

4.2.4. GREEDY DUAL

이 스케줄링 기법도 역시 BLOCK LRU 기법과 PAGE NUM 속성을 적절히 혼합하기 위한 기법이다. 이 기법에서는 블록에 대한 LRU 에서의 위치에 대한 값과 블록에 해당하는 페이지 개수의 값을 가지고 Greedy Cost 값을 산출하고 그 값을 가지고 힙으로 구성하였다. 아래는 현재 사용된 식이다. (그림6)

$$\text{Greedy Cost} = \text{PAGE NUM} * \text{LRU\_Value}$$

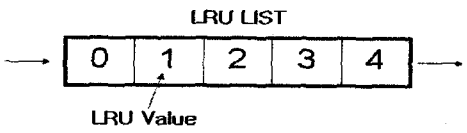

그림 6 Greedy Cost and LRU_Value

5. 실험

5.1. 실험 데이터

트레이스 데이터는 리눅스 EXT2 파일 시스템에서 블록 사이즈를 4K, 1K로 설정하고 각각 일반적인 작업을 하여 쓰기 요청만을 추출하였다. 작업은 컴파일 작업, vi 문서 작업, webbrowser, gimp 작업, mail 송수신 작업 등이다. 블록 사이즈 4K, 1K 트레이스는 쓰기 요청이 각각 381904, 2883502 건이다. 각 트레이스에 작은 임의쓰기 (<= 8 sector) 비율은 전체 쓰기 요청의 60%을 차지하고 쓰기 데이터 양에는 각각 20%, 5% 를 차지하였다.

5.2. 실험 환경

실험 환경은 낸드 플래시 메모리는 SLC, small block을 비휘발성 메모리는 256k ~ 2M 크기를 가정하였다. 로그 블록 기법과 성능 비교 평가는 로그 블록 개수를 100개로 비휘발성 메모리 크기는 2M를 가정하였다.

5.3. 실험 결과

5.3.1. 읽기 연산

로그 블록 기법에서는 쓰기 연산이 일어나기 전에 항상 데이터 블록에 데이터가 있는지 읽어 보고 없으면 데이터 블록에 쓰고 있으면 로그 블록에 쓰는 정책을 쓴다. 본 논문에서 제안하는 알고리즘은 비휘발성 메모리에서 플래시 메모리로 데이터가 내려갈 때 항상 새로운 블록을 할당 받아 쓰기 때문에 불필요한 읽기 연산을 줄였다. 그림7)에서 읽기 연산은 로그 블록 기법과 비교하여 90% 감소 효과를 보였다.

5.3.2. 쓰기 연산

쓰기 연산은 최대 4K 트레이스에서 33%, 1K 트레이스에서 20% 감소 효과를 보였다. 4K 트레이스는 1K 트레이스 보다 작은 임의 쓰기 데이터 양 비율이 높았기 때문에 비휘발성 메모리의 쓰기 버퍼 효과가 더 컸다.

5.3.3. 소거 연산

소거 연산은 최대 4K 트레이스에서 50%, 1K 트레이스에서 25% 감소 효과를 보였다. 쓰기 연산처럼 4K 트레이스는 1K 트레이스 보다 작은 임의 쓰기 데이터 양 비율이 높았기 때문에 비휘발성 메모리의 쓰기 버퍼 효과가 더 컸다.

5.3.4. 버퍼크기별 각 알고리즘 성능 비교

비휘발성 메모리 쓰기 버퍼가 증가할수록 선형적으로 추가적인 성능 개선이 이루어짐을 알 수 있다. 알고리즘의 성능은 쓰기 요청의 재참조율과 비휘발성 메모리에서 플래시 메모리로 내려가는 블록의 평균 페이지 개수에 따라 성능이 차이난다. BLOCK LRU 알고리즘은 다른 알고리즘보다 재참조율은 높으나 내려가는 블록의 평균 페이지의 개수가 적고, PAGE NUM 알고리즘은 다른 알고리즘보다 내려가는 블록의 평균 페이지 개수는 많으나 재참조율이 낮다. BLOCK LRU + CLEAN LIST 와 GREEDY DUAL 알고리즘 두 요소를 모두 고려했기 때문에 상대적으로 나머지 두 알고리즘보다 성능이 우수했다. 특히 GREEDY DUAL 알고리즘은 시간 참조성과 블록의 평균 페이지 개수를 비용 기반으로 계산하여 두 요소를 적절히 고려했기 때문에 BLOCK LRU + CLEAN LIST 알고리즘 보다 우수했다. GREEDY DUAL 알고리즘 다른 알고리즘보다 최대 10% 성능이 우수했다.

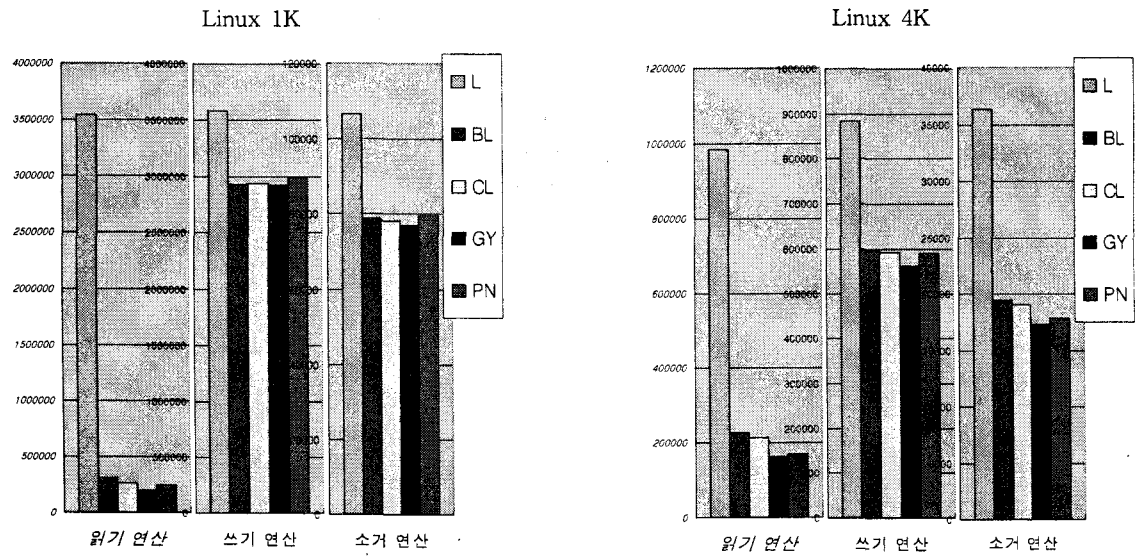

그림 7 실험 성능 평가 (L: LOG BLOCK, BL: BLOCK LRU, CL: BLOCK + CLEAN LIST, GY: GREEDY DUAL, PN : PAGE NUM)

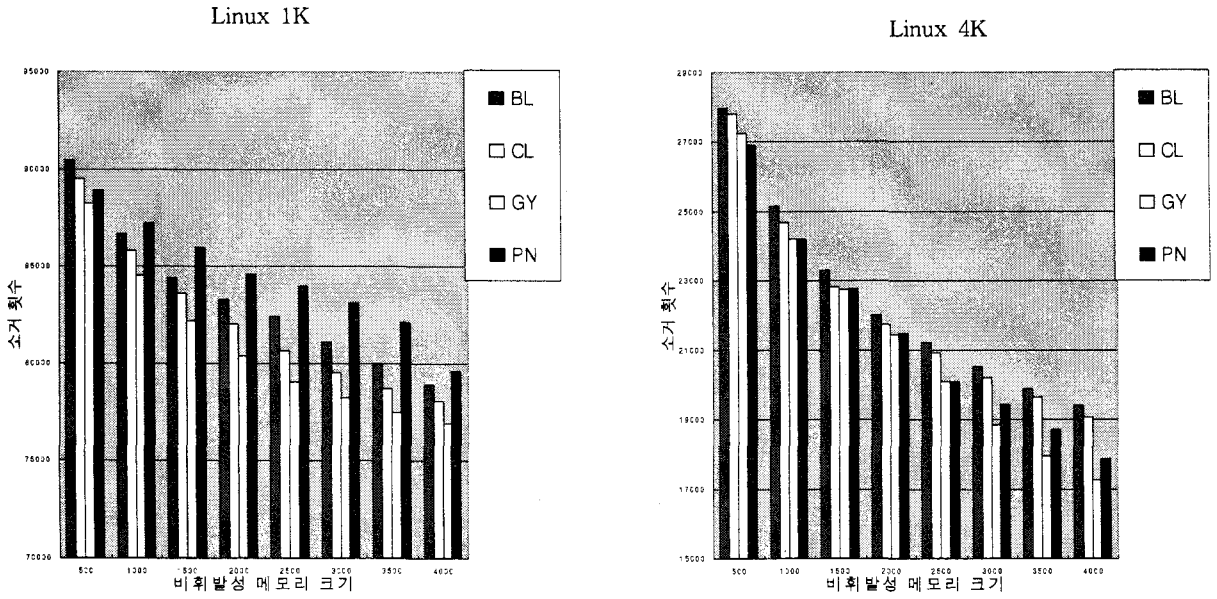

그림 8 버퍼 크기에 따른 성능 평가 (L: LOG BLOCK, BL: BLOCK LRU, CL: BLOCK + CLEAN LIST, GY: GREEDY DUAL, PN : PAGE NUM)

6. 결론 및 향후 연구

낸드 플래시 메모리에서 비휘발성 메모리를 쓰기 버퍼로 사용하기 위한 다양한 알고리즘을 제시하였다. 이 알고리즘들을 실험을 통해 소용량의 비휘발성 메모리로 낸드 플래시 메모리의 성능이 40% 정도 향상 될 수 있다는 것을 보였다. 낸드 플래시에서 최대 읽기의 횟수는 90% 감소 , 쓰기 횟수는 33% 감소, 소거의 횟수는 50% 감소되었다. 특히 소거 횟수의 감소는 플래시 메모리의 수명 연장에도 도움이 될 것이다. 이밖에 부가적으로 I/O 요청의 속도 향상, I/O의 안정성 향상과 부팅 시간의 단축 등의 효과도 있을 것이다.

향후에는 대용량 플래시 메모리에서 병렬로 구성했을 때에 대한 알고리즘에 대하여 연구할 것이다. 또한 성능이 가장 좋은 Greedy Dual 알고리즘은 실제로 구현에 비용이 크기 때문에 이를 효과적으로 대체할 수 있는 방법을 찾아 볼 수 있을 것이다.

참고 문헌

[1] Jesung Kim, Jong Min Kim, Sam H. Noh, Sang Lyul Min and Yookun Cho, "A Space-Efficient Flash Translation Layer for CompactFlash Systems", IEEE Transactions on Consumer Electronics, Vol. 48, No. 2, May 2002.

[2] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and JA. A. Tauber, "Storage Alternatives for Mobile Computers", In Proceedings of the 1st Symposium on Operation Systems Design and Implementation(OSDI), 1994.

[3] Petro Estakhri and Berhanu Iman, "Moving Sequential Sectors within A Block of Information in A Flash Memory Mass Storage Architecture", United States Patent, No. 5,930,815, 1999.

[4] John L. Hennessy and David A. Patterson, "Computer Architecture: A Quantitative Approach (2nd ed.)," Morgan Kaufmann, 1996.

[5] Bum Soo Kim and Gui Young Lee, "Method of Driving Remapping in Flash Memory and Flash Memory Architecture Suitable Therefore", United States Patent, No. 6,381,176, 2002.

[6] Takayuki Shinohara, "Flash Memory Card with Block Memory Address Arrangement", United States Patent, No. 5,905,993, 1999.

[7] Binny Gill and Dharmendra S. Modha "WOW: Wise Ordering for Writes-Combining Spatial and Temporal Locality in Non-Volatile Caches" USENIX File and Storage Technologies (FAST), December 13-16, 2005

[8] B. S. Gill and D. S. Modha, "SARC: Sequential prefetching in adaptive replacement cache," in USENIX, 2005.

[9] J.-F. Paris, T. R. Haining, and D. D. E. Long, "A stack model based replacement policy for a non-volatile cache," in Proc. IEEE Sym. Mass Storage Sys., pp. 217-224, March 2000