하이브리드 스토리지 기반의 데이터 패턴을 고려한 적응적 스토리지 클래스 메모리 관리 기법

정상원, 이태훈, 정기동

부산대학교 정보컴퓨터공학과

{jsw0921, withsoul, kdchung}@ melon.cs.pusan.ac.kr

Adaptive Storage Class Memory Management Policy Based-on Hybrid Storage Considering Data Access Pattern

Sangwon Jung, TaeHoon Lee, kidong Chung

Dept of computer Science and Engineering , Busan University

요 약

기존의 스토리지 클래스 메모리와 플래시 메모리를 결합한 하이브리드 스토리지는 고정적인 공간의 스토리지 클래스 메모리를 사용하여서 I/O 패턴에 따라 공간적인 비효율성을 보여주었다. 본 논문에서는 데이터 패턴과, 데이터 접근 지역성에 따라 스토리지 클래스 메모리의 영역들이 적응적으로 변화하는 스토리지를 제시하고 있다. 시뮬레이션 결과 적응적 스토리지 클래스는 고정적 스토리지 클래스 메모리와 비교하여 iozone같은 경우 15.3%, postmark의 경우 13.1%의 공간 절감 효과를 보였다.

1. 서 론

플래시 메모리는 무게가 가볍고, 비휘발성이고, 하드디스크에 비해 읽기 속도가 빠르며, 내구성이 강하다는 장점 때문에 새로운 스토리지로 부각되고 있다. 플래시 메모리는 임베디드 시스템과 모바일 시스템뿐만 아니라 개인용 컴퓨터와 서버용 컴퓨터에서의 대용량 스토리지로 사용되고 있다.

플래시 메모리는 위와 같은 여러 장점들이 존재 하지만, 물리적 특성이 기존에 존재하는 하드 디스크와 다르다. 같은 페이지에 대해 업데이트가 일어 날 경우, 램 또는 하드디스크와는 다르게 제자리 덮어쓰기가 불가능 하다. 같은 오프셋에 데이터를 적기 위해서는 삭제연산(블록 단위)이 선행 되어야 한다. 삭제 연산에 따른 시간 지연과 오버헤드가 있기 때문에, 이를 해결하기 위해서 플래시 메모리는 다른 영역에 업데이트 할 데이터를 적고, 이전에 쓰인 데이터는 무효 표시를 하여 유효하지 않은 데이터로 간주 하여 삭제 연산을 미루는 방식을 취하고 있다.

"이 논문은 2009년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2009-0087838)"

또한, 더 이상 빈 공간이 없을 시에, 공간을 확보하기 위해 가비지 컬렉션 과정을 거치게 된다. 이 작업은 블록 내의 무효 표시된 블록들을 삭제하고, 유효한 데이터들만 따로 모은 뒤, 여유 블록들을 생성 해낸다. 이런 과정들 때문에 주소 변환 과정이 필요하고, 응용 프로그램 또는 운영체재가 원활하게 스토리지를 사용 하는 것을 지원하는 플래시 변환 계층(Flash translation layer)이 필요 하게 된다[4].

위와 같은 플래시 메모리만의 특징들 때문에, 데이터들을 플래시 메모리에 배치하는 방식인 플래시 변환 계층의 효율적인 알고리즘과, 가비지 컬렉션의 횟수를 줄이고 효과적인 가비지 컬렉션 알고리즘에 대한 연구가 중요한 이슈가 되고 있다.

플래시 메모리와 더불어 새로운 저장장치로 각광받고 있는 것은 스토리지 클래스 메모리이다. 스토리지 클래스 메모리는 플래시 메모리와 같이 비휘발성 속성을 가지면서, 고속의 바이트 단위의 접근이 가능하다. 따라서 플래시 메모리가 가지는 단점에서 자유로울 수 있어서, 플래시 메모리와 하드 메모리를 대체할 스토리지로 떠오르고 있다. 그렇지만 스토리지 클래스 메모리는 아직 고가의 스토리지이고, 개인용 컴퓨터나 대용량의 서버를

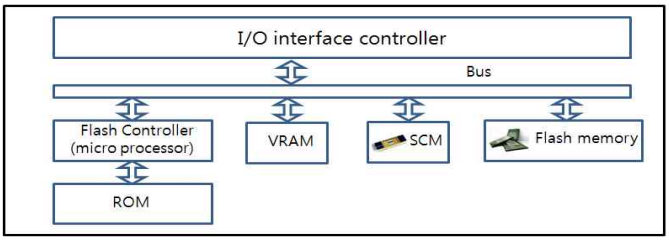
커버할 만한 수준은 되지 못한다[3, 5].

이렇듯, 플래시 메모리와 스토리지 클래스 메모리는 여러 장점으로 인해 새로운 스토리지로 떠오르고 있지만, 여러 단점들이 존재한다. 이들의 장점을 얼마나 살리면서, 단점들을 보완 할 수 있는 알고리즘 연구들의 중요함이 널리 인식 되고 있는 추세이다.

본 논문에서는 스토리지 클래스 메모리와 플래시 메모리를 결합한 하이브리드 저장장치를 제시하고 있다. 기존의 연구에서는 스토리지 클래스 메모리가 입력 데이터 패턴을 고려하지 않고 사용 되었다. 입력 데이터 패턴에 따라 필요한 버퍼 크기가 다르고, 데이터 접근의 지역성에 따라 필요한 매핑 정보의 범위가 달라지게 되는데, 이런 가변적인 사항들에 대한 고려가 없이 연구되어져 왔다. 따라서, 저장공간이 효율적으로 이용되지 못하였다. 본 연구에서는 데이터 패턴과, 데이터 접근 지역성에 따라 스토리지 클래스 메모리의 영역들이 적응적으로 변화하여 스토리지 클래스 메모리의 공간 효율성을 높이고, 최소한의 크기로 최대의 효율을 달성하고자 한다.

2. 관련 연구

2.1. 하이브리드 스토리지 모듈



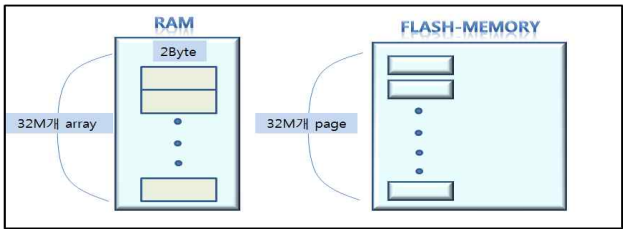
[그림 3] 스토리지 클래스 메모리를 버퍼로 활용한 스토리지 모듈

플래시 메모리와 스토리지 클래스 메모리가 결합된 스토리지 모듈은 [그림 1]과 같은 구조를 가지게 된다.[6] 모듈은 모듈을 전체적으로 제어하는 Flash controller, 펌웨어와 매핑 알고리즘을 저장하고 있는 ROM, 상위 계층과 의사소통을 위한 I/O interface controller, 임시 저장장치인 VRAM이 있다. 그리고 저장장치로써, 스토리지 클래스 메모리(SCM), 플래시 메모리가 있다. 이때, 스토리지 클래스 메모리가 버퍼로써 사용 되어 진다면, 기존의 연구에서는 크기는 고정적으로 정해지게 된다.

2.2. 스토리지 클래스 메모리의 매핑 정보 저장

하나의 논리적 주소와 플래시 메모리의 하나의 물리적 주소와 1:1 매핑의 시키는 방식을 섹터 매핑이라고 한다. 시스템이 초기화 될 때, 시스템은 매핑 정보를 알기 위해서 각 블록의 첫 번째 페이지에 기록 되어 있는 매핑 정보를 탐색해야 한다. 만약 32GB 낸드 플래시 메모리 시스템(페이지 크기 : 1024 bytes, 각 블록의 페이지 개수: 32) 을 가정 한다면 블록의 개수는 1,048,576 (32GB / 1024 X 32)이 되게 된다. 따라서 1,048,576번의 플래시 메모리 스캔 과정이 필요하게 되는데 한번의 access time이 30us이므로 총 약 30초의 탐색 과정이 필요 하게 된다. 많은 시간 소요가 있는 것을 알 수 있다.[6]

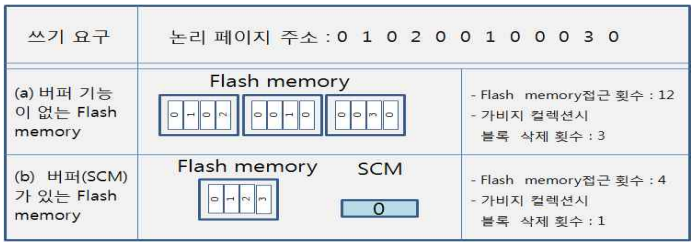
램에 저장되는 매핑 정보의 크기는 [그림 2]과 같다, 플래시 메모리의 페이지 개수와 같은 페이지 개수가 램상에 있어야 되고, 하나의 페이지를 2Byte라고 가정 한다면 총 64MB라는 큰 크기의 매핑 정보를 램상에 유지하여야 한다.



[그림 4] 섹터 매핑의 경우 매핑 정보의 크기

2.3. 스토리지 클래스 메모리의 버퍼 활용

스토리지 클래스 메모리는 바이트 단위의 임의 접근이 가능 하므로, 저용량으로 지역성이 높은 데이터들을 효과적으로 처리 할 수 있다. [그림3]는 하나의 논리적 페이지를 하나의 물리적 주소만 연결시키는 섹터 매핑을 가정한 저장장치에서 버퍼 여부에 따른 지역성이 높은 데이터 처리 방식을 보여주고 있다. 쓰기 요구를 살펴보면 '0'번에 해당하는 페이지에 대한 쓰기 요구가 많은 것을 알 수 있다. 버퍼가 존재하지 않는 (a)의 경우는 주소 '0'번에 해당 하는 요구가 들어오면 플래시 메모리는 덮어 쓰기가 불가능 하므로 그림과 같이 다른 페이지에 쓰는 방식을 취하는 것을 알 수 있다. (b)의 경우는 스토리지 클래스 메모리를 버퍼로 사용하는 경우 인데 스토리지 클래스 메모리는 덮어쓰기가 가능하므로 그림과 같이 '0'번의 주소를 지역성이 높은 데이터로 인식하여 처리하는 것을 알 수 있다, (a)와 같은 경우는 상대적으로 느린 플래시 메모리에 (b)에 비해 많이 접근 하는 것을 알 수 있다. 또한 추후에 이 블록들이 삭제 되어 질 경우 (a)의 경우 3번의 삭제가 (b)의 경우 1번의 삭제가 이루어지게 된다, 그림에서 쉽게 알 수 있듯이 (b)의 경우가 효과 적이라는 것을 알 수 있게 된다. 따라서 접근하는 데이터들의 주소를 윈도우 형식으로 계속 추적하여, 연속적으로 들어오는 데이터들은 플래시 메모리에 기록 하지 않고 스토리지 클래스 메모리에 기록 하여 높은 성능을 보일 수 있다.

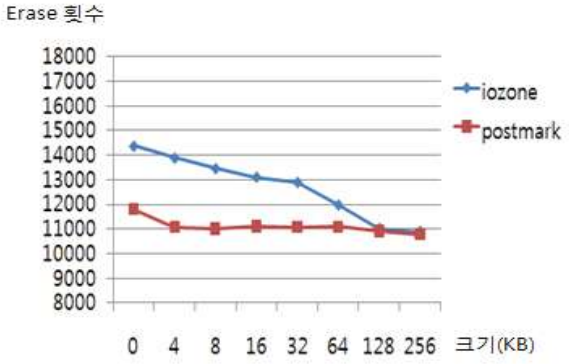


[그림 5] 버퍼여부에 따른 지역성이 높은 데이터 처리 방식

2. 기존 연구의 문제점 및 연구 동기

2.1. 기존 연구의 문제점

기존의 연구에서는 스토리지 클래스 메모리를 버퍼로 활용 할 때, 입력 데이터 패턴을 고려하지 않고 사용 되었다. 입력 데이터 패턴에 따라 필요한 버퍼 크기가 다르게 된다. [그림 4]는 스토리지 클래스 메모리 버퍼 크기에 따른 블록 삭제 연산 횟수를 나타내는 그래프이다. 블록 삭제 연산은 읽기, 쓰기 연산에 비해 많은 시간이 소모 되므로 삭제 연산이 많을수록 시스템의 속도 및 성능이 떨어지게 된다. 실험 트레이스 파일은 순차적인 접근의 구조를 가지고 같은 주소에 대해 여러번 접근하는 iozone과 임의접근과 순차접근의 복합적 구조를 가지면서 같은 주소에 대한 반복적인 접근이 없는 postmark이다.

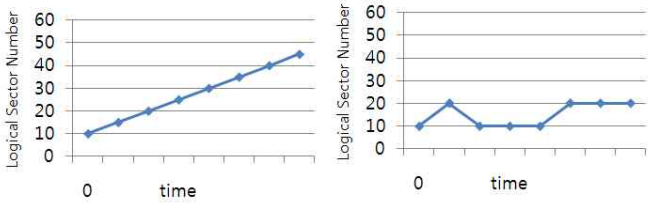


[그림 6] 스토리지 클래스 메모리 버퍼 크기(KB)에 따른 블록 삭제연산 횟수

iozone은 순차적인 데이터접근이 같은 주소값에 대해서 여러번 연속적으로 나타나기 때문에 버퍼크기가 증가 할수록, 삭제 연산 횟수가 감소하는 것을 볼 수 있다. 반면, postmark는 최신에 쓰여 졌던 데이터가 다시 재사용되는 않고, 데이터접근이 임의적인 주소에 일어나기 때문에 버퍼크기가 증가해도 삭제 연산이 크게 증가하지 않는 것으로 나타났다. 즉 iozone 같은 경우는 필요한 버퍼 크기가 상대적으로 크고, postmark같은 경우는 필요한 버퍼 크기가 상대적으로 작다. 기존 연구와 같이 버퍼 크기가 고정적이라면 파일의 데이터 접근 특성에 따라 버퍼가 큰 경우 버퍼 공간이 사용되지 않고 낭비 되는 경우 발생 할 것이다, 반대로 버퍼가 작은 경우 필요한 버퍼가 부족한 경우도 발생 하게 된다. 이와 같이 고정된 버퍼크기를 사용하는 정책은 상당히 효율이 떨어지는 것을 알 수 있다.

이와 유사하게 매핑 정보를 저장 함에 있어서도 문제점이 존재한다. [그림 5]는 시간에 따라서 접근하는 논리적 주소를 나타내고 있다. (a)의 경우는 거의 모든 데이터 주소에 접근을 하고, (b)의 경우는 한정된 주소값에만 여러 번 접근하는 지역성을 띄는 모습을 보여준다.

앞서 언급하였듯이 매핑 정보를 구성하는 데는 시간적으로, 용량적으로 많은 자원 소모가 있다. (b)와 같은 경우 모든 주소에 대한 매핑 정보가 필요 하지 않은 것을 볼 수 있다. 따라서 필요한 매핑 정보만 구성하는 것이 효과적일 것이다. 기존 연구에서는 이런 데이터 파일 접근의 지역성을 고려하고 있지 않기 때문에 자원 낭비 측면에서 비효율성을 보여준다.



(a) 지역성을 띄지 않는 경우 (b) 지역성이 높은 경우

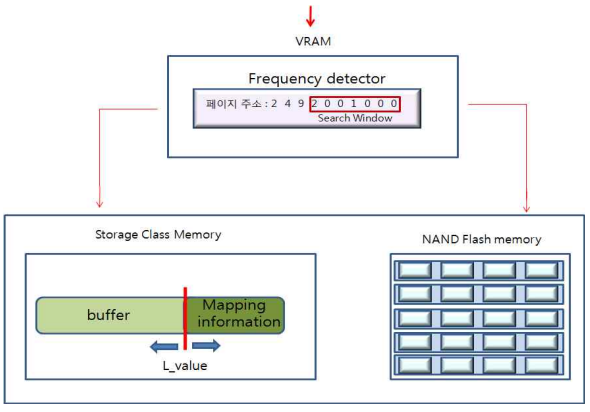
[그림 7] 데이터 접근의 지역성이 높은 경우와 낮은 경우

2.2 버퍼크기와 매핑정보 크기의 관계

데이터 접근의 지역성에 따라 필요한 버퍼크기와 매핑 정보 크기는 상관관계를 가진다. [그림 5]에서 (a)의 경우 여러 주소 값을 골고루 접근 하고 있기 때문에 필요한 매핑 정보량이 많을 것을 알 수 있다. 그에 반면, 지역성이 없기 때문에 많은 크기의 버퍼가 필요 없게 된다. (b)의 경우 특정 주소 값만을 접근하는 것을 볼 수 있다. 따라서 매핑 정보는 특정 주소 값만을 가지도 있으면 되므로 필요한 매핑 정보는 작아지게 된다. 그에 반면, 특정 지역을 계속 해서 접근하기 때문에 버퍼 크기가 클수록 효율적 이라는 것을 쉽게 예측 할 수 있다. 이렇듯 데이터 접근의 지역성에 따라 필요한 버퍼크기와 매핑 정보의 크기는 반비례하게 된다.

3. 적응형 하이브리드 스토리지

3.1. 스토리지 모듈



[그림 8] 적응형 스토리지 구조

적응형 하이브리드 스토리지는 [그림 6]과 같은 구조를 가진다. 스토리지 클래스 메모리와 플래시 메모리가 스토리로 사용 되어 지고, 데이터 패턴과 데이터 접근 지역성을 추적하고 분석하기 위해 VRAM(Volatile RAM)이 사용 된다. VRAM의 Frequency dectector는 상위 계층에서 들어오는 최신의 요청들을 저장하게 된다. Search Window를 통해 데이터 패턴을 분석 하게 된다.

분석을 통해 스토리지 클래스 메모리의 매핑 정보를 저장하는 크기와 버퍼로 사용되는 크기를 조정하게 된다. 데이터가 특정 지역 주소를 반복적으로 접근 한다면 매핑 정보를 저장하는 공간은 작아지게 되고, 반복적으로 접근 하는 값들을 저장하는 버퍼 공간을 커지게 된다. 반대로, 데이터가 여러 주소를 접근하게 된다면 여러 매핑 정보를 저장해야 하므로 매핑 정보를 저장하는 공간을 커지게 되고, 반복적으로 접근하는 주소값의 수가 작아지므로 버퍼의 크기는 작아지게 된다.

[그림 6]을 예로 살펴보면 최신의 데이터들은 페이지 '0~2'에만 반복적으로 접근하는 지역성이 높은 것을 알 수 있다. 따라서 매핑 정보는 이들의 정보들만 기록하게 되고, 매핑 정보의 크기는 작아지게 될 것이다. 매핑 정보를 저장 하는 공간이 작아졌으므로, 버퍼의 크기는 커지게 된다. 커진 버퍼 크기에 페이지 '0~2'에 해당하는 정보를 저장하게 되는 것이다. 기존 방식에서는 고정적인 크기의 이처럼 데이터 패턴과 데이터 접근 지역성을 분석하여 스토리지 클래스 메모리가 적응적으로 변화함으로써, 효율적인 공간 활용성을 보여준다.

3.1. 스토리지 클래스 메모리의 구성 및 역할

스토리지 클래스 메모리는 논리적 주소와 물리적 주소의 매핑을 저장하는 공간과 반복적으로 접근하는 데이터 주소에 해당하는 값을 저장하는 버퍼로 구성된다. 매핑 정보는 논리적 주소에 해당하는 물리적 주소 값을 저장하고 있으며, 이때 물리적 주소는 버퍼를 가르키고 있을 수도 있다. 여기서는 페이지 매핑을 가정하고 있으므로, 요청된 논리적 페이지가 어느 물리적 페이지에 저장되어 있는지에 대한 정보가 저장된다.

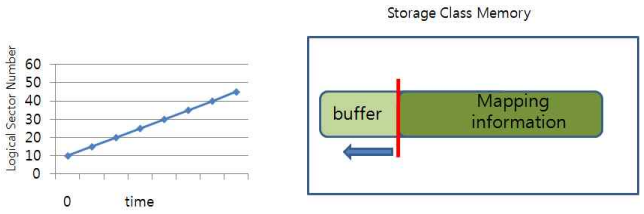
버퍼는 반복되는 주소들의 데이터 값을 저장 하고 있다. 플래시 메모리는 덮어쓰기가 불가능 하므로 반복적으로 들어오는 주소에 대해서 효율적으로 대처하지 못한다. 플래시 메모리는 다른 영역에 업데이트 할 데이터를 적고, 이전에 쓰인 데이터는 무효 표시를 하여 유효하지 않은 데이터로 간주 하여 삭제 연산을 미루는 방식을 취하고 있다. 이에 반면 스토리지 클래스 메모리는 덮어쓰기가 가능 하므로, 효율적인 처리가 가능하다.

3.2 Frequency detector의 역할

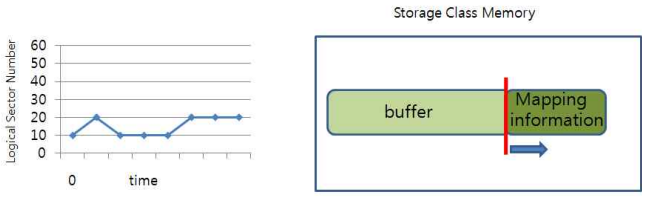
Frequency detector는 데이터 패턴과 데이터 접근 지역성을 추적하고 분석하여서, 스토리지 클래스의 적응적인 변화를 유도한다. 최근의 페이지 번호들을 저장하여 window 방식으로 분석을 하게 된다. 최근의 페이지 번호들이 여러 주소들에 대해 접근이 있었다면, 매핑 정보를 담는 공간을 늘리게 되고, 같은 주소들에 대한 접근이 많다면 버퍼의 크기를 상대적으로 늘려주게 된다.

3.3. 스토리지 클래스 메모리의 적응적 변화

필요한 매핑 정보와 버퍼크기는 반비례 한다는 것을 앞서 살펴보았다. 고정된 스토리지 클래스의 크기를 이 반비례하는 관계를 이용하여 두 영역이 적응적으로 공유하게 되는 것이다. [그림 8]은 데이터 패턴에 따른 스토리지 클래스 메모리의 적응적인 변화를 보여준다. (a)의 왼쪽 그림은 시간에 따라 여러 주소를 접근하면서, 같은 주소에 대해 반복적으로 접근하지 않는 데이터 패턴을 보여주고 있다. 여러 주소에 대해 접근 하므로 스토리지 클래스의 매핑 정보를 담는 공간의 크기가 증가하는 것을 볼 수 있다. 반면 반복적인 접근이 없으므로 버퍼의 크기는 작아졌다. (b)의 왼쪽 그림은 같은 주소에 대해 반복적으로 접근 하면서, 제한된 주소들에 대해서만 접근 하는 데이터 패턴을 보여주고 있다. 이에 맞게 스토리지 클래스의 공간이 적응적으로 변화하여, 매핑 정보의 공간은 작아지게 되고, 반복적으로 같은 주소에 접근하는 데이터들을 버퍼링하기 위해 버퍼의 공간이 커지는 것을 볼 수 있다.



(a) 지역성을 띄지 않는 경우 스토리지 클래스 메모리의 변화



(b) 지역성이 높은 경우 스토리지 클래스 메모리의 변화

[그림 7]데이터 패턴에 따른 스토리지 클래스 메모리의 적응적인 변화

4. 실험 및 평가

실험 트레이스 파일은 순차적인 접근의 구조를 가지고 같은 주소에 대해 여러번 접근하는 iozone과 임의접근과 순차접근의 복합적 구조를 가지면서 같은 주소에 대한 반복적인 접근이 없는 postmark이다. 이 파일들을 시뮬레이션 하였고, 적응적 스토리지 클래스 메모리와 고정적 스토리지 클래스 메모리

를 비교 실험 하였다. 적응적 스토리지 클래스 메모리와 같은 성능을 내기 위해서는 얼마만큼의 고정적 스토리지 클래스 메모리의 크기가 필요한지에 대한 실험이다. 실험 개요는 [표 1]과 같다.

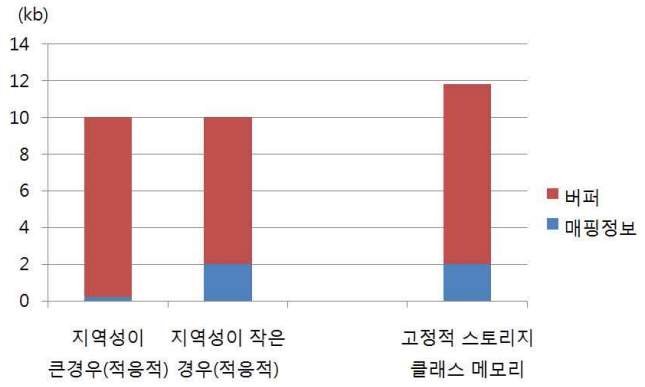
[표 1] 실험 개요

트레이스	-iozone : 같은 주소에 대해 여러번 접근하는 구조 -postmark : 임의접근과 순차접근의 복합적이고 같은 주소에 대해 자주 접근하지 않는 구조
Search window 크기	128
블록당 페이지 수	32
페이지의 크기	512byte
플래시 메모리 크기	64Mbyte
스토리지 클래스 크기	10Kbyte

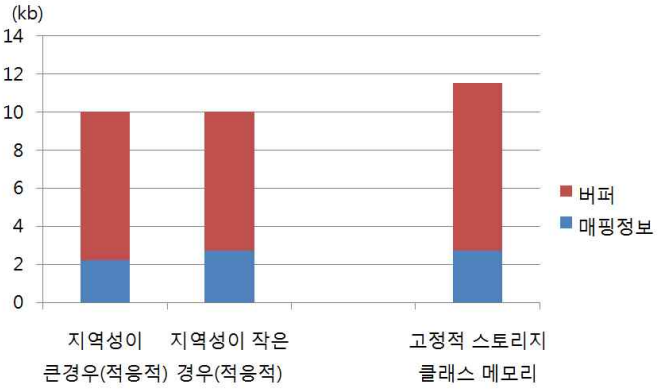
[그림 8]은 iozone의 적응적 스토리지와 고정적 스토리지의 필요 크기를 나타낸 그래프이다. 왼쪽 그래프 두 개는 적응적 스토리지 클래스 메모리에서 버퍼와 매핑정보의 비율을 나타낸다.

지역성이 큰 경우를 보면 같은 주소 값에 대해서 반복적으로 접근 하므로 매핑정보를 담는 크기가 작은 것을 볼 수 있다. 그에 반면, 반복적인 데이터를 담기위해 버퍼가 크다. 지역성이 작은 경우는 여러 주소에 대해 접근하기 때문에 매핑정보의 크기가 큰 것을 알 수 있고, 같은 주소에 대해 자주 접근하지 않으므로 필요한 버퍼 크기가 작게 된다. 가장 왼쪽의 막대는 적응적 스토리지 클래스 메모리와 같은 성능을 보이기 위해서 필요한 고정적 스토리지 클래스 메모리 크기를 나타낸다. 같은 성능을 나타내기 위해서는 우선 버퍼의 크기는 지역성이 가장 큰 경우에 필요한 버퍼의 크기와 같아야 하고, 매핑정보의 크기는 지역성이 작은 경우에 필요한 매핑정보의 크기와 같아야 된다. 따라서 다음과 같은 결과가 나오게 된다.

[그림 9]는 postmark의 적응적 스토리지와 고정적 스토리지의 필요 크기를 나타낸 그래프이다. 실험 결과는 iozone과 유사하다. 고정적 스토리지 클래스 메모리와 똑같은 성능을 내기 위해서 적응적 스토리지 클래스는 iozone같은 경우 84.7%의 크기만 있으면 되고, postmark의 경우 86.9%의 크기만 있으면 된다. 즉, 적응적 스토리지 클래스는 고정적 스토리지 클래스 메모리와 비교하여 iozone같은 경우 15.3%, postmark의 경우 13.1%의 공간 절감 효과를 보였다. 적응적 스토리지 클래스 메모리는 작은 크기의 스토리지 클래스 메모리로 높은 공간 효율성을 보여준다고 할 수 있다.



[그림 8] 적응적/고정적 스토리지의 필요 크기 (iozone)



[그림 9] 적응적/고정적 스토리지의 필요 크기(postmark)

5. 결론 및 향후 연구 계획

최근의 스토리지는 여러 패턴의 데이터 접근에 대해 적응적으로 대처를 할 수 있어야 한다. 기존의 연구에서는 스토리지 클래스 메모리를 활용 할 때, 입력 데이터 패턴을 고려하지 않고 사용 되었다. 본 연구에서는 I/O 패턴 기반 분석의 적응적 스토리지를 제시 하였다. 데이터 패턴에 따라 적응적으로 스토리지의 공간이 변화함으로써, 공간 효율성이 높아지게 된다. 반비례 관계에 있는 필요 버퍼 크기와 필요 매핑 관계를 이용하여 보다 효율 적인 스토리지 모듈을 설계하였다.

추후, 매핑정보의 크기와 버퍼의 크기의 비율을 적절하게 조절하는 알고리즘이 연구 되어야 한다.

6. 참고문헌

[1] Jesung Kim, Jong Min Kim, Sam H. Noh "A space-efficient flash translation layer for compactflash systems", IEEE Transactions on Consumer Electronics, 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 Transactions on Embedded Computing Systems, vol. 6, Feb. 2007.
[3] Po-Chun Huang, Yuan-Hao Chang, Tei-Wei Kuo, Jen-Wei Hsieh, Miller Lin,"The Behavior Analysis of Flash-Memory Storage Systems", 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC), 2008
[4] T. Kgil, D. Roberts, and T. Mudge. "Improving NAND Flash Based Disk Caches". In ISCA 08: Proceedings of the 35th annual international symposium on Computer architecture, 2008 pages 327--338, 2008.
[5] Young-Sik Lee, Dawoon Jung. "Memory Management Scheme for Cost-Effective Disk-On-Modules in Consumer Electronics Devices " IEEE Transactions on Consumer Electronics. Vol. 54, No. 4, NOVEMBER 2008
[6] J. Kim, H. Lee and K. Bahng, "A PRAM and NAND Flash hybrid architecture for high-performance embedded storage subsystems," ACM & IEEE EMSOFT'08, October 2008.
[7] 도인환, 노삼혁, " 스토리지 클래스 메모리 도입에 따른 컴퓨팅 패러다임과 시스템 소프트웨어의 변화", 정보과학회지, 2009
[8] Hyung Gyu Lee. "High-Performance NAND and PRAM Hybrid Storage Design for Consumer Electronics ". IEEE Transactions on Consumer Electronics, Vol. 56, No. 1, FEBRUARY 2010