플래시메모리 기반 저장시스템의 성능 분석을 위한 I/O 트레이스 가시화 도구 개발

윤경훈*O 정호영* 박성민* 차재혁* 강수용+
*한양대학교 정보통신공학과
+한양대학교 컴퓨터교육과
{rumimaru,sykang^O}@hanyang.ac.kr

Implementation of I/O Trace Visualization Tool for Flash Memory based Storage Systems

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

요 약

최근 플래시 메모리는 여러 장점들 때문에 다양한 휴대기기에서 많이 사용되고 있다. 반면 내구성에 약점을 갖고 있는 플래시 메모리의 특성 때문에 최대한 소거 동작을 적게 하여 오랫동안 사용하는 FTL 알고리즘을 개발하는 연구가 필요하다. 이러한 FTL 알고리즘을 실험하고 평가하기 위해서 트레이스 데이터를 연구에 활용하는 일이 많아지면서, 쉽게 트레이스 데이터에 대한 분석도구를 개발하였다. 우리는 트레이스 데이터를 다양한 그래프로 그려주고 통계치를 산출해주는 도구를 개발하였고, 이를 바탕으로 트레이스 분석 작업을 쉽게 할 수 있도록 하였다. 마찬가지로 이러한 도구는 버퍼 교체정책을 실험하고 평가하는 일에도 사용 될수 있다. 그리고 각 그래프를 설명하면서 트레이스에 데이터 대한 설명과 함께 분석을 통하여 버퍼교체 알고리즘및 FTL 알고리즘에 어떻게 활용 할 수 있는지 설명하였다.

1. 서 론

최근 디지털 카메라, MP3 플레이어, 핸드폰과 같이 이동성이 중요한 디지털 기기들을 많이 사용하고 있다. 이에 따라서 소형화, 저전력화, 비휘발성, 충격에 강한 메모리가 필요하게 되었고 Flash Memory는 이러한 요구사항을 만족시키는 메모리로 기존에 주로 사용되던 하드디스크에 비해 많은 장점을 가지고 있다. 그렇기에 점점 H 하드디스크를 대체하여 대표적인 저장 장치로 사용될 것이다.

플래시 메모리는 하드디스크와 다르게 Inplace-Update가 되지 않기 때문에 쓰기전 소거(Erase-before-write) 연산을 해야 한다. 이러한 연산은 한 블록에 평균적으로 10만번 이상 사용하면 해당 블록의 신뢰성에 결함이 생기기 때문에 플래시 메모리의 전체 블록을 평준하게 Erase 연산이 발생되도록 하는 Flash Translation Layer (이하 FTL)이 필요하다. 플래시 메모리를 저장 장치로 사용하기 위한 FTL 알고리즘의 성능을 평가하기 위해서는 디스크 I/O 패턴으로 실험을 해야 한다.

플래시 메모리의 FTL에서의 버퍼 교체 정책 또는 파일 시스템을 연구하고 성능을 평가하기 위해서 우리가 컴퓨터에서 하는 일상적인 작업(예를 들어, 문서 편집을 한다거나 웹브라우저를 사용하는 등..)들이 하드디스크 또는 버퍼상에서 어떠한 패턴을 갖고 있는지 Workload를 볼 수 있어야 한다. 이러한 Workload를 알아보기 위해서 트레이스를 추출하는 작업이 필요하며 우리는 트레이스를 추출하고 트레이스를 분석하기 위해서 다양한 그래프를 그리는 윈도우즈용 도구를 개발하였다.

본 연구에서는 트레이스를 추출하는 계층에 대해 설명하고, 추출된 트레이스 데이터를 다양한 그래프로 표현해주는 도구를 사용하여 그래프를 사용한 트레이스 데이터의 분석에 대해 쓰였다. 또한 우리는 메인 메모리의 양에 따라서 트레이스 패턴의 변화를 알아보기 위해서 메인 메모리의 양을 변화시키면서 실험을 하였다.

2. 트레이스 데이터

트레이스 데이터란 물리적인 저장장치에 요청되는 LBA(Logical Block Address)의 집합이며, 저장장치 관련된 알고리즘에 대한 실험과 평가를 위해서 사용 된다. 트레이스는 추출하는 Layer에 따라서 디스크 트레이스와 버퍼 트레이스로 나눠진다.

2.1 디스크 트레이스

디스크 트레이스는 운영체제의 버퍼를 거치고 물리적인

디스크 장치에 I/O를 요청하는 트레이스를 의미하며, 이러한 트레이스는 플래시 메모리에서 쓰기 패턴을 연구하는데 필요하다.

디스크 트레이스를 추출하기 위해서는 운영체제의 디바이스 드라이버 층에서 실제 물리적인 하드디스크에 요청되는 I/O에 대해 알아내고, 이것을 파일로 저장하여 추출 할 수 있다. 우리의 연구에는 이러한 도구로서 BYU Disk Trace Buffer(이하 DTB)를 사용하였다. DTB는 *Brigham Young University Performance Evaluation* Laboratory의 Trace Distribution Center (http://tds.cs.byu.edu/tds/)에서 배포하는 리눅스에서 디스크 트레이스를 추출하는 도구이다.[2]

2.1.1 메타데이터

파일 시스템에는 메타데이터를 저장하는 영역이 따로 할당 되어 있다. 메타데이터란 데이터를 표현하는 데이터, 즉 하드 디스크에 저장되어있는 실제 데이터들에 대한 정보를 가지고 있는 데이터이다.

우리는 리눅스 Ext2 파일시스템에서 추출한 트레이스 데이터로 실험을 하였고, 해당 파일시스템의 메타데이터 블록 번호의 범위를 알아내서 통해 관련 연구를 진행하였다. Ext2 파일시스템의 경우 디스크가 여러 개의 블록 그룹으로 나뉘어져 있으며, 각각의 블록 그룹들은 메타데이터 블록과(슈퍼 블록, 블록 비트맵, inode 비트맵, inode 테이블)와 데이터 블록으로 구분되어 있다[1].

Boot Sector	Block Group 1	Block Group 1	.....	Block Group N

<표 2. The Physical Structure of Ext2 Filesystem >

Super Block	FS descriptors	Block Bitmap	Inode Bitmap	Inode Table	Data Blocks

<표 3. The Physical Structure of Each Block Group >

Ext2 파일시스템에서 메타 데이터 블록의 범위를 알아내고, 우리가 제작한 그래프기반 시각화 도구를 통해 메타데이터의 패턴을 알 수 있다. 3.2절의 접근 패턴 그래프를 통해 메타 데이터에 대한 영역과 일반 데이터에 대한 읽기/쓰기 트레이스를 분석할 수 있다. 전체 트레이스중 메타데이터에 대한 I/O가 얼마나 많이 일어났고, 자주 일어났는지 그 패턴에 대해 알 수 있기 때문에 이에 대한 패턴 연구를 통해서 플래시 메모리에 일반 데이터를 저장하고 NVRAM(Non-Volatile RAM)에 메타데이터를 저장하는 연구에서 활용할 수 있다.

2.2 버퍼 트레이스

버퍼 트레이스는 운영체제의 버퍼 층에 요청되는 I/O 트레이스이며, 운영체제의 버퍼 교체 정책을 연구하는데 좋은 자료가 된다. 이러한 상위 레벨에서의 버퍼 교체에 대한 연구가 이루어져서 효율적인 버퍼 교체 정책이 개발되면, 플래시 메모리를 저장 장치로 사용하는 시스템에서 효과를 볼 수 있기 때문에, 플래시 메모리와의 연관성도 깊다고 볼 수 있다.

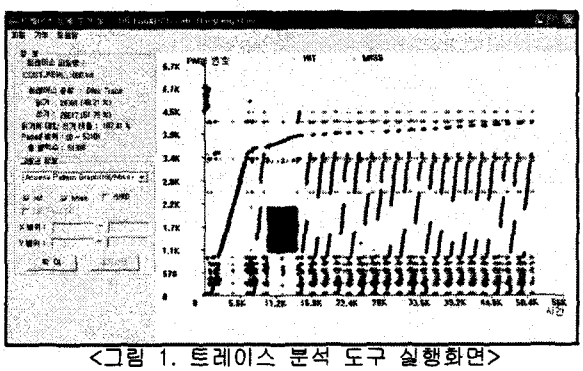
3. 트레이스 분석 가시화 도구

3.1 트레이스 분석

일반적으로 페이지 번호와 쓰기/읽기로만 구성된 텍스트 트레이스 데이터는 가독성이 떨어지기 때문에 사용자가 트레이스를 분석하기 힘들며, 다른 사람들이 이해할 수 있도록 자료를 준비해야하는 시간도 길어지기 때문에 불편한 점이 많다. 하지만, 그 결과를 쉽게 해석할 수 있는 그래프로 트레이스 정보를 보여주고, 통계 정보(읽기/쓰기 횟수, 요청된 페이지 수 등..)를 볼 수 있다면 쉽게 해석할 수 있는 장점이 있다.

우리의 연구에서는 트레이스 데이터를 입력받아서 그래프를 그려주는 윈도우즈 운영체제에서 사용할 수 있는 도구를 개발하였다. 다음절에서는 이 도구를 이용하여 그래프를 보고, 각 그래프를 분석하는 방법에 대해 설명하겠다.

3.1 트레이스 분석 도구

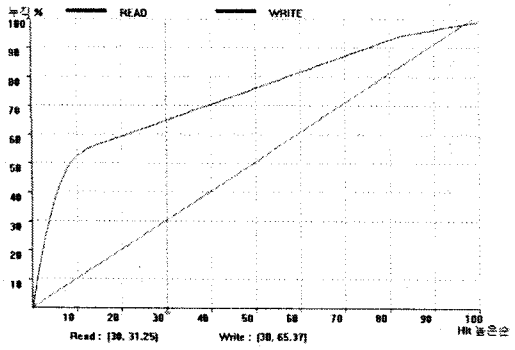


<그림 1. 트레이스 분석 도구 실행화면>

이 도구의 주요 기능은 트레이스 데이터를 읽어서 로컬리티 그래프(누적, 횟수)를 그려줌으로써 데이터의 집적도에 대해 알 수 있고, 접근 패턴 그래프를 그려줌으로써 패턴을 분석 할 수 있다. 또한 트레이스에 대한 통계 정보(읽기/쓰기 횟수, 읽기에 대한 쓰기의 비율, 페이지 번호 범위, 요청된 페이지 수)를 표현 해준다.

그림1을 보면 왼쪽 상단의 그룹에서 트레이스에 대한 통계 정보를 표현하고 있고, 오른쪽은 그래프를 그려주는 공간이다. 왼쪽 하단의 그룹은 오른쪽 그래프를 다양하게 보기 위해서 옵션을 지정해 줄 수 있는 컨트롤들이 위치한 공간이다.

3.1 누적 로컬리티 그래프(Accomulate Locality)



<그림 2. 누적 로컬리티 그래프>

□□ 그래프 설명

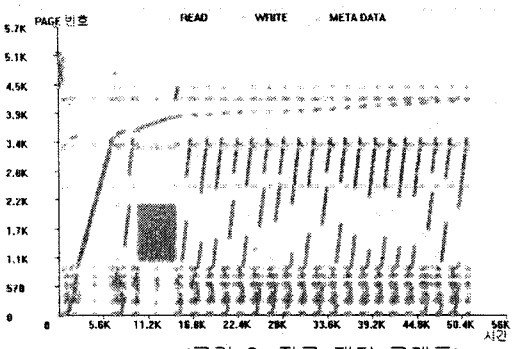
트레이스 데이터의 누적 로컬리티를 표현하는 그래프이다. 누적 로컬리티란 전체 트레이스 데이터 중에서 많이 참조되는 페이지 번호순서로 정렬하여 누적시켜 가면서 그리는 그래프이다. X축은 Hit가 높은 Page 순서로 정렬시켜 퍼센트(%)로 표현하였고, Y축은 로컬리티의 누적 퍼센트(%)를 표현하였다.

□□ 그래프 해석

누적 로컬리티 그래프는 특정 영역에 어느 정도의 로컬리티를 갖고 있는지 알아 볼 수 있는 그래프로서, 전체 트레이스 데이터중에 자주 Hit되는 영역이 어느 정도인지 파악하여 버퍼 교체 정책 등의 관련 연구에 사용 할 수 있다. 보통 10%의 영역에 90%의 데이터가 집적된 것 을 10/90 이라고 표현하며, 이러한 트레이스 데이터의 경우 좁은 영역에서 I/O가 일어난 것으로 Hit Rate가 높다고 판단한다.[3]

<그림2>에서 보이는 그래프는 리눅스에서 vi editor로 특정 파일을 편집하고 저장하는 작업의 디스크 트레이스 데이터에 대한 그래프 이며, 쓰기 데이터는 상위 30%에 전체 트레이스 데이터의 65.37%가 집적되어 있어 로컬리티가 30/70 정도로 높은 편 이고, 읽기 데이터는 상위 30%에 전체의 31.25%가 집적되어 있어 로컬리티가 50/50 정도로 낮은 편이다. 실제 디스크에 요청되는 I/O 트레이스는 운영체제의 버퍼 층을 거쳐서 버퍼에 없는 페이지 번호를 요청하기 때문에, 이미 한번 읽어들인 데이터는 버퍼공간이 작아서 스왑(Swap) 되기 전 까지 버퍼에 존재하게 된다. 그렇기 때문에 중복되는 페이지 번호가 아닌 새로운 페이지 번호를 요청하게 되고, 결과적으로 읽기 데이터의 경우 낮은 로컬리티를 갖게 된다.

3.2 접근 패턴 그래프(Access Pattern)



<그림 3. 접근 패턴 그래프>

□□ 그래프 설명

트레이스 데이터의 접근 패턴(Access Pattern)을 표현한 그래프로서, 트레이스의 분포도를 보면서 접근 패턴에 대해 알아 볼 수 있다. 시간이 지나면서 어떠한 페이지들이 어떠한 시간에 Hit 되었는지 한눈에 알아 볼 수 있으며, 메타데이터의 경우 2.1.1절에서의 설명한 내용을 바탕으로 추출한 범위 데이터를 파일로 입력받아서 해당 범위의 페이지들은 짙은 분홍색으로 표시되어 메타데이터를 이용한 연구에서도 활용 할 수 있다.

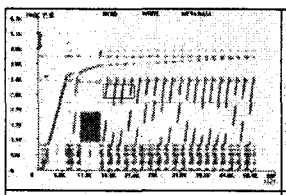
또한 전체적인 그래프의 모습을 보고 사용자가 보다 자세히 알아보고 싶은 영역이 있다면, 확대/축소 기능을 이용하여 특정 영역을 마우스로 드래그하여 클릭하면 트레이스의 분포를 상세하게 볼 수 있기 때문에 활용도가 높은 그래프이다. X축은 트레이스의 처음부터 끝까지의 상대적인 시간을 표현한다. 즉 트레이스 순서라고 생각할 수 있다. Y축은 Hit된 페이지 번호이다.
그리고 데이터를 쉽게 구분하여 해석할 수 있도록 데이터의 형태에 따라 그래프에 그려지는 점의 모양과 색깔이 다르다.

√ Read : 빨간색 +
√ Write : 파란색 ◇
√ Read&Write : 초록색 O
√ Metadata : 진한분홍색 X

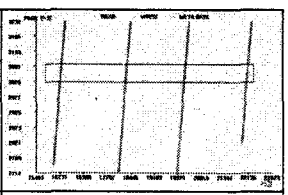
□□ Graph 해석

접근 패턴 그래프는 페이지의 시간/ 공간 지역성을 개괄적으로 보여주며, 트레이스의 전체적인 패턴을 알아 볼 수 있고, 여러 다른 그래프의 특성이 이 그래프에서 나타나기 때문에 대표적인 분석 그래프라고 볼 수 있다. 순차 접근 요청(Sequential Reuqest)인지 임의 접근 요청(Random Request)인지 판단하고, 자주 Hit되는 영역을 구분하여 플래시 메모리의 FTL 연구에 사용 할 수 있다.

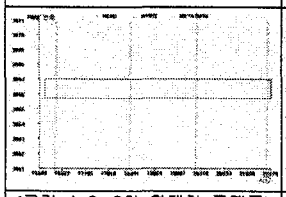
<그림 3>에서 보이는 그래프는 postgresql 데이터베이스의 디스크 I/O 트레이스 데이터에 대한 그래프이다. 전체적으로 동일한 패턴으로 반복되어 동일한 페이지가 요청되는 패턴을 쉽게 알아 볼 수 있다. 그래프는 압축된 형태의 패턴을 보여주기 때문에 동일한 패턴이 나타난다 하더라도, 실제로 동일한 페이지번호가 다시 참조되는지에 대해서는 알 수 없다. 그림 4.1~4까지의 과정은 동일한 페이지가 다시 참조된 트레이스인지, 순차적인 트레이스인지 알아보기 위해 특정 영역을 지정해서 계속 확대해 가면서 본 과정이며, 그 결과 다시 참조되는 Hit임을 알 수 있다.



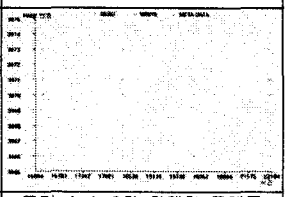
<그림 4-1. 초기 그래프>



<그림 4-2. 1차 확대한 그래프>



<그림 4-3. 2차 확대한 그래프>

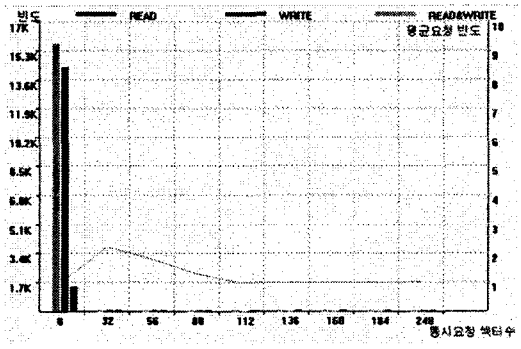


<그림 4-4. 3차 확대한 그래프>

3.3 동시 요청 섹터 그래프(Request Sector)

□□ Graph 설명

Request Sector Graph는 동시에 요청되는 Page의 수를 표현한 그래프이다. 동시에 요청되는 페이지 수의 단위는 4K가 한 Block인 파일시스템에서는 8의 배수로 요청되며, 각각 독립적인 데이터이기 때문에 막대그래프로 표현하였다. X축은 동시에 요청되는 페이지의 수이며, 왼쪽 Y축은 동시에 요청되는 단위별 횟수이며, 오른쪽 Y축은 해당 X축의 값에서 특정 페이지에 요청되는 빈도. 값이 높으면 동일한 페이지가 다시 요청되는 경우가 많다는 뜻이다.



<그림 5. Request Sector Graph>

□□ Graph 해석

동시 요청 섹터 그래프는 한번에 디스크로 요청되는 페이지의 크기를 기준으로 횟수를 누적시키고, 평균적으로 요청되는 빈도를 그려주는 그래프이다. 디스크 트레이스 데이터의 경우 적은 숫자의 쓰기 요청은 잦은 쓰기가 발생하게 되어 플래시 메모리의 성능 저하의 원인이 될 수 있으며, 쓰기 버퍼 캐시에서 작지만 자주 발생되는 쓰기를 잘 처리해주는 정책이 필요하다.

<그림 5>에서 보이는 그래프는 리눅스에서 Netscape Web-Browser를 사용하여 10분가량 웹서핑을 한 작업의 트레이스 데이터에 대한 그래프 이다. 8페이지씩 요청되는 트레이스 데이터가 대부분임을 쉽게 알 수 있으며, 평균요청빈도 그래프가 1에 근접하기 때문에 대부분 재참조되지 않는 트레이스임을 알 수 있다. 특정 페이지에 32~80 페이지단위로 재참조되는 트레이스가 있다는것을 x축의 32,56,80부분이 볼록하게 나와있는 그래프의 모습으로 알 수 있으며, 웹서핑 트레이스 데이터의 특성상 캐시에 데이터를 덮어쓰기(over-write)하는 부분으로 해석 할 수 있다.

4. 추후 연구

본 논문에서 다룬 내용은 트레이스 데이터를 분석하여 이를 운영체제의 버퍼교체정책과 플래시 메모리의 FTL 알고리즘의 실험 및 성능평가에 활용하는 내용 이였으며, 우리는 현재 트레이스 데이터를 입력받아 운영체제의 버퍼 계층을 통해 DBMS 시스템의 버퍼 계층과 플래시 메모리의 FTL 계층까지 연결하여, 시스템의 총체적인 접근 패턴및 Hit rate를 통계 분석하는 도구를 개발하

고 있다. 이러한 추후 연구의 결과는 각 계층 간의 버퍼 교체정책에 따라 시스템의 성능의 전반적인 효율성을 판단하고 분석하는 목적으로 활용 될 수 있을 것이다.

5. 결 론

다양한 휴대용 디지털 기기들이 많이 등장하고 있고, 우리의 일상생활에서 그 활용도가 매우 높아짐에 따라 대부분의 휴대용 디지털 기기에서 저장 장치로 사용되는 플래시 메모리가 점점 많이 사용되고 있다. 플래시 메모리를 효율적으로 사용하기 위해서 성능이 좋은 FTL 알고리즘이 중요하며 FTL 알고리즘을 연구하기 위해 실험 및 평가과정에서 트레이스 데이터를 추출하고 활용하는 것 이 중요하다. 본 연구에서는 리눅스 기반에서 트레이스 데이터를 추출하고, 추출된 트레이스 데이터를 그래프로 시각화 해서 보여주는 도구를 개발하여 이를 기반으로 플래시 메모리의 연구에 활용하였다.

실험을 위해 몇 가지의 트레이스 데이터를 추출하였지만, 앞으로 다양한 실험및 성능평가를 위해서는 좀 더 다양한 환경에서 추출된 트레이스 데이터가 필요하며 본 논문에서 소개한 트레이스 추출 방법 및 그래프 기반 시각화 도구가 유용하게 사용될 것이다.

5. 참고 문헌

[1] Rey Card, "Design and Implementation of the Second Extended Filesystem", Proceedings of the First Dutch International Symposium on Linux.

[2] BYU Disk Buffer Trace Tool Document, http://traces.byu.edu/new/Documentation/

[3] Jen-Wei Hsieh, Li-Pin Chang, and Tei-Wei Kuo, "Efficient On-line Identification of Hot Data for Flash-Memory Management," ACM SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA

[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.