이종 SSD로 구성된 스토리지 서버에서 저장장치 최적화를 위한 논리 블록 계층 관리

조유현O*, 진용석*, 박지성, 김지홍
서울대학교 컴퓨터공학부
{yhjo, ysjin, jspark, jihong}@davinci.snu.ac.kr

Logical Block Layer Management to Optimize Devices on Storage Servers Configured with Heterogeneous SSDs

Yoohyun JoO*, Yongseok Jin*, Jisung Park, Jihong Kim
Department of Computer Science and Engineering, Seoul National University

요 약

병렬성을 최대화하여 대역폭을 높인 SSD는 최근 스토리지 서버에서 많이 사용되고 있다. 하지만, 병렬성을 최대로 활용하는 방식은 저장장치 내부에 서로 다른 데이터가 혼재되어 시스템의 성능을 저하시킨다. 또한, 이 문제를 해결하는 기존 연구들은 다양한 물리 블록 크기의 이종 SSD로 구성된 스토리지 서버 시스템에는 적용할 수 없다. 본 논문에서는 이종 SSD로 구성된 시스템에 적용 가능한 논리 블록 계층 관리를 제안한다. 논리 블록 계층에서 적절한 크기의 논리 블록 단위로 흩뿌려 저장하는 방식을 통해 기존 방식 대비 평균 83.6%의 쓰기 증폭 지수가 감소하여 전체 시스템 성능 향상을 확인하였다.

1. 서 론

최근 스토리지 서버에서는 높은 대역폭과 낮은 전력소모, 적은 오류 발생 등의 이유로 기존 저장장치로 사용하던 하드디스크를 SSD (Solid State Drive)로 대체하고 있다.

보통 SSD는 그림 1과 같이 내부적으로 많은 물리 블록들로 이루어진 플래시 메모리에 데이터를 최대한 흩뿌려 저장하는 방식(full striping)을 사용한다. 이 방식으로 높은 대역폭을 만족시킬 수 있지만 NAND칩 내부에 여러 데이터가 혼재되어 저장된다. 혼재된 데이터는 가비지 컬렉션(GC) 과정에서 쓰기 증폭을 증가시켜 전체 시스템의 성능을 저하시킨다.

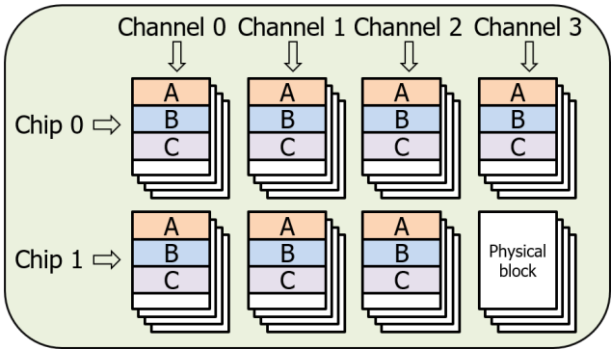

그림 1. SSD 내부 구성과 Full Striping 방식의 데이터 저장

스토리지 서버에서 일반적인 full striping 방식의 SSD를 사용하면 생기는 성능 저하 문제를 해결하기 위해 플래시 메모리를 시스템에서 직접 관리하여 SSD의 대역폭과 실제 사용 공간을 증가시킨 Software-Defined Flash[4], 여러 사용자에게 제공한 가상 SSD에서 요청을 처리할 때 실제 SSD 내부에서 요청 간 충돌이 발생하는 문제를 물리적인 공간 분리로 해결한 FlashBlox[5]등의 관련 연구가 있다. 이런 연구에서는 SSD와 시스템의 성능을 최대화 하기 위하여 시스템에서 블록 단위로 데이터를 관리하고, 저장장치에 요청한다. 그런데 블록 크기를 단일 SSD로 구성된 환경을 가정하여 정한 것이라는 한계가 있다. 거대한 스토리지 서버 시스템에서는 단일 SSD만으로 구성될 가능성이 낮고, 물리 블록 크기가 다른 이종 SSD (heterogeneous SSD)가 사용될 확률이 높다. 시스템에서 관리하는 블록의 크기가 SSD의 물리 블록 크기와 어울리지 않으면 기존 full striping 방식처럼 데이터가 혼재되어 전체 시스템과 저장장치 성능이 저하된다.

본 논문에서는 이종 SSD로 구성된 스토리지 서버에서 전체 시스템의 성능을 최대화하기 위한 논리 블록 계층(logical block layer)관리를 제안한다. 논리 블록 계층에서는 SSD 내부 NAND칩을 직접 관리하여 full striping 방식이 아닌, 데이터를 논리 블록 크기 단위로 흩뿌려 저장하는 방식(intra striping)을 사용한다. SSD의 물리 블록 크기에 적합한 논리 블록 크기 단위로 데이터를 모아서 저장하기 때문에 데이터가 혼재되는 비율이 작고, GC과정에서 쓰기 증폭이 적게 발생해 전체 시스템 성능이 개선된다. intra striping 방식의 추가 최적화로 물리 블록 크기에 맞게 데이터를 저장한 후 남은 자투리 데이터만 따로 저장하는 방식(remainder block)을 구현하였다.

논리 블록 계층의 성능을 확인하기 위한 실험으로 SSD 에뮬레이터[1]로 여러 물리 블록 크기의 SSD를 구성한 다음 LSM-Tree[2] trace 기반의 워크로드 I/O 요청을 처리할 때 발생한 쓰기 증폭 인자(WAF, Write Amplification Factor)를 측정했다. 실험 결과 일반 full striping 방식과 비교하여 intra striping 방식에서 평균 83.6% WAF가 감소하였고, remainder block 방식은 intra striping 방식 대비 최대 5.4% WAF가 감소하였다.

*공동 제1저자

논문의 구성은 다음과 같다. 2장에서는 대상 시스템의 구조와 논리 블록 인터페이스에 대해 설명한다. 3장에서는 논리 블록 계층에서 플래시 메모리를 관리하는 기법들에 대한 설명과 관련 연구를 소개한다. 4장에서는 실험 환경에 대한 설명과 실험 결과 및 분석을 보인다. 5장에서 추후 연구에 대해 말하고, 6장에서 결론을 맺는다.

2. 대상 시스템 구조

2.1 개요

그림 2는 본 연구의 대상이 되는 시스템의 전체적인 구조를 나타낸다. 저장장치로 물리 블록 크기가 다른 여러 이종 SSD들이 구성되어 있고, 응용 계층(application layer)은 논리 블록 계층을 통하여 SSD를 사용하는 모습을 보여준다.

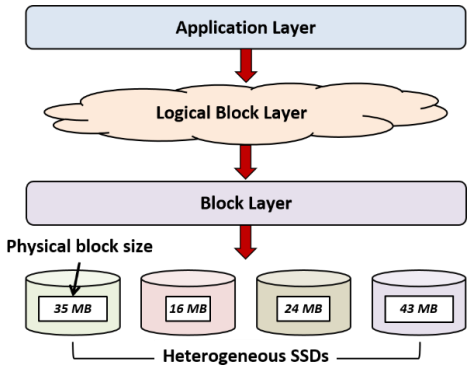

그림 2. 대상 시스템 구조

2.2 논리 블록 인터페이스

논리 블록 계층에서는 응용 계층이 저장장치를 사용할 수 있도록 다음과 같은 세 종류의 인터페이스 함수를 제공한다. (1)**`get_block`** 함수를 통해 응용 계층은 사용할 수 있는 지정된 크기의 논리 블록을 할당 받는다. 리턴 값으로 논리 블록 번호를 받아 논리 블록을 사용하는 요청들의 인자로 사용한다. (2)**`read_block/write_block`** 함수를 통해 특정 논리 블록에 데이터를 읽거나 쓴다. 인자로 논리 블록 번호, offset, 데이터의 크기를 사용한다. 리턴 값으로 읽거나 쓴 데이터의 크기를 반환한다. (3)**`trim_block`** 함수를 통해 특정 논리 블록의 데이터를 invalidate시킨다. 리턴 값으로 TRIM된 데이터의 크기를 반환한다. NAND칩 플래시 SSD의 경우 TRIM 명령어를 통해 저장장치 내부의 유효한 데이터를 판단하여 GC도중 쓰기 증폭을 감소시키기 때문에 필수적으로 사용된다.

3. 논리 블록 계층의 플래시 메모리 관리

3.1 Intra striping 방식의 데이터 저장

기존 full striping 방식에서는 데이터들이 혼재되어 저장되기 때문에 GC과정에서 WAF가 증가한다. 이 문제를 해결하기 위해 논리 블록 계층에서는 플래시 메모리를 직접 관리하여 데이터를 논리 블록 단위로 모아 저장장치에 저장하는 intra striping 방식을 구현하였다. 기존 SSD 내부 컨트롤러에서 FTL (Flash Translation Layer)모듈이 담당하는 L2P mapping, GC등의 작업을 논리 블록 계층에서 수행한다. 응용 계층에서 논리 블록을 요청하면, 저장장치에서 논리 블록 크기의 물리 블록을 할당하고, 이후 해당 영역에서 데이터 쓰기와 읽기가 이루어 진다. 그림 3은 물리 블록 크기보다 논리 블록 크기를 크게 설정한 경우이고, intra striping 방식으로 저장장치에 데이터(A, B, C)가 저장된 모습을 나타낸다.

그런데, Intra striping 방식으로 데이터를 저장하는 경우 SSD의 병렬성을 최대로 활용하지 못하는 문제점을 생각해볼 수 있다. 하지만 스토리지 서버 시스템에서는 응용이 많은 수의 논리 블록을 할당하여 사용하거나 또는 여러 응용들이 할당 받은 각 논리 블록을 동시에 사용하는 경우가 대부분일 것이다. 이 경우 SSD는 실질적으로 full striping 방식의 경우와 동일하게 최대 대역폭으로 사용되어 성능의 저하는 없으면서 데이터 혼재 비율만 감소시킬 수 있다.

3.2 Remainder block 방식의 데이터 저장

Intra striping 방식에서도 논리 블록 크기와 물리 블록 크기 비율에 따라 데이터가 혼재되는 부분이 생기는 것을 알 수 있다. 이에 추가적인 최적화로 자투리 부분을 따로 모아서 저장하는 remainder block 방식을 구현하였다. 그림 4는 remainder block 방식의 데이터 저장을 나타낸다. 일부 block들을 자투리 부분을 저장하기 위한 remainder block으로 미리 지정해 두고, 물리 블록 크기에 맞게 데이터를 저장한 후 남은 데이터를 remainder block에 저장한다. 이 방식으로 intra striping 방식보다 더 적은 데이터 혼재를 기대한다.

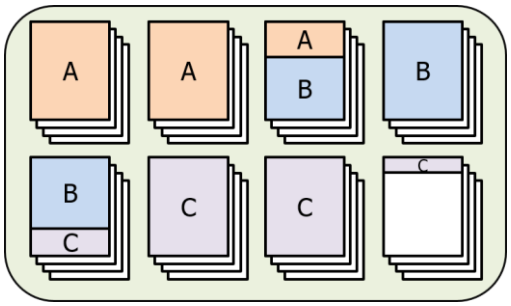

그림 3. Intra Striping 방식의 데이터 저장

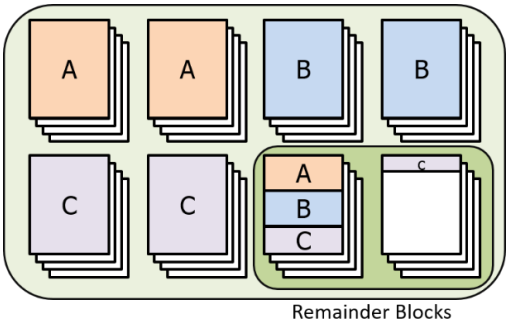

그림 4. Remainder Block 방식의 데이터 저장

3.3 관련 연구

저장장치에 데이터가 혼재되는 것을 막기 위한 노력으로 최신 SSD에서는 Multi-Stream방식의 쓰기를 지원한다.[3] 쓰기 요청 시 특정 stream을 지정하여 전달하면 저장장치에서 데이터를 stream단위로 구분하여 저장한다. 하지만 Multi-Stream SSD는 최대로 활용 가능한 stream개수가 상당히 적어 많은 응용이 사용하는 스토리지 서버 시스템에서는 대응에 한계가 있다. 논리 블록 계층의 플래시 메모리 관리 기법들은 수많은 응용에 대해서도 각각 논리 블록을 할당해 주는 방식으로 확장성 있는 대응이 가능하다

4. 실 험

4.1 실험 환경

실험은 Intel i7-7700 3.6GHz 8-thread CPU, 32GB 메모리, Linux 4.14.11 환경에서 SSD 에뮬레이터[1]를 이용하여 진행하였다. 실험에 사용한 워크로드는 4096*1024*4개의 key range로 구성된 LSM-Tree[2]에서 유효한 데이터의 비율을 90%로 설정 후 발생한 약 1억7천만개의 I/O 요청 trace를 기반으로 논리 블록 계층의 인터페이스를 통해 읽기, 쓰기, TRIM 명령을 요청한다. 받은 요청을 SSD에서 (1)full striping 방식, (2)intra striping 방식, (3)remainder block 방식으로 데이터를 저장하는 경우에 발생한 WAF를 각각 측정하였다.

이종 SSD를 사용하는 환경을 고려하여 SSD의 물리 블록 크기를 30MB, 35MB, 40MB, 50MB로 바꾸어가며 실험했으며, 논리 블록 크기는 82MB로 고정하였다.

4.2 실험 결과 및 분석

그림 5에서는 여러가지 논리 블록 크기 대 물리 블록 크기 조합에 대한 full striping (FS)방식, intra striping (IS)방식, remainder block (RB)방식으로 데이터를 저장할 때 측정된 WAF의 비교를 보여준다. FS 방식의 경우 블록 크기 비율에 큰 영향을 받지 않아서 하나의 블록 크기 비율(82:30)에 대한 결과만 나타내었다. FS 방식에서 측정된 WAF는 IS 방식을 사용할 때 대비 압도적으로 높게 측정되었다. IS 방식에서 측정된 WAF는 FS 방식에서 측정된 WAF 대비 평균적으로 83.6%의 감소를 보이고, 최대 86.11%의 개선을 나타낸다. FS 방식의 데이터 저장에서는 저장장치에서 많은 데이터들이 혼재되어, GC 과정에서 많은 유효 데이터 복사 작업이 수행되고 그 결과 큰 WAF가 측정되었다. IS 방식의 데이터 저장에서는 데이터들이 저장장치 내에서 논리 블록 크기로 모여서 저장되므로 데이터들의 혼재 비율이 작아져 동일한 워크로드를 수행하는 과정에서 작은 WAF가 측정되었다.

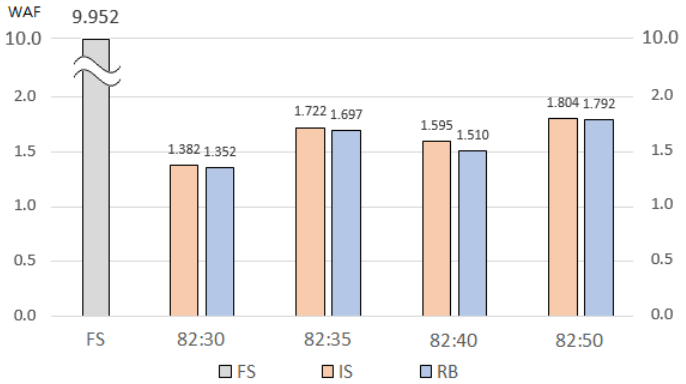

그림 5. Full Striping 방식, Intra Striping 방식, Remainder Block 방식의 WAF 비교

IS 방식에서 추가적인 최적화로 진행한 RB 방식의 데이터 저장은 여러 블록 크기 조합에서 IS 방식 대비 평균적으로 2.5%의 적은 성능 개선이 있었고, 물리 블록 크기를 40MB로 설정하였을 때 가장 큰 5.4%의 성능 개선이 있었다. 실험 결과 IS 방식의 데이터 저장이 데이터 혼재를 충분히 잘 대응하여 RB 방식이 추가적인 개선을 보이기 어려웠다고 판단된다. 다만 remainder block이 잘 활용되는 특정 블록 크기 조합(82:40)일 때 RB 방식이 좋은 효과를 보였다.

5. 추후 연구

논리 블록 크기 단위로 데이터를 저장하더라도 데이터의 혼재는 일부 여전히 존재하고 특히, 혼재된 데이터의 수명(데이터가 유효한 기간)에 따라 WAF에 많은 영향을 미치기 때문에 이에 대한 고려가 필요하다. 예를 들어, 수명이 짧은 데이터(hot data)들과 수명이 긴 데이터(cold data)가 혼재되어 저장되는 경우 GC 과정에서 cold data를 옮기는 과정 때문에 WAF가 증가하게 된다. 만약, hot data와 cold data를 논리 블록 계층에서 판단하여 분리해 저장할 수 있다면 WAF를 감소시켜 전체 시스템의 성능을 개선시킬 수 있다. 관련 연구로 논리 블록 주소 기반으로 hot data와 cold data의 분리를 하는 AutoStream[6]이 있다.

또한, 여러 종류의 이종 SSD를 동시에 사용하는 경우 어떤 논리 블록 크기가 적절한지 자동으로 판단하는 최적화 방향이 있다. 실험에서는 논리 블록 크기를 고정하고, 물리 블록 크기를 바꾸어가며 제한적인 실험 결과만을 확인하였다. 실제 SSD의 서로 다른 물리 블록 크기에 적합한 논리 블록 크기를 자동으로 설정해주는 기법을 적용한다면 SSD 구성의 변화가 잦은 서버 환경에서 효과가 기대된다.

6. 결 론

Full striping 방식으로 병렬성을 최대화하여 대역폭을 높인 SSD는 저장장치 내부에 데이터가 많이 혼재되어 시스템의 성능을 저하시킨다. 이 문제를 해결한 기존 연구들은 물리 블록 크기가 다른 이종 SSD로 구성되는 경우를 고려하지 않았다. 본 논문에서는 이종 SSD로 구성된 스토리지 서버 시스템을 대상으로 논리 블록 계층을 통해 intra striping 방식 또는 remainder block 방식으로 데이터를 저장하여 SSD 내부 데이터 혼재로 발생하는 시스템 성능 저하 문제를 해결하였다.

감사의 글

이 연구를 위해 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드립니다. 이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(NRF-2018R1A2B6006878). (교신저자: 김지홍)

참 고 문 헌

[1] Sungjin Lee et al., “Application-Managed Flash,” in *Proceedings of the 14th FAST*, pp. 339-353, 2016.

[2] O’Neil et al., “The Log-Structured Merge-Tree (LSM-Tree),” in *Acta Informatica*, pp. 33-351, 1996.

[3] KANG, J.-U et al., “The Multi-Streamed Solid-State Drive,” in *Proceedings of the 6th HotStorage*, 2014.

[4] Jian Ouyang et al., “SDF: Software-Defined Flash for Web-Scale Internet Storage” in *Proceedings of the 19th ASPLOS*, pp. 471-484, 2014.

[5] Jian Huang et al., “FlashBlox: Achieving Both Performance Isolation and Uniform Lifetime for Virtualized SSDs,” in *Proceedings of the 15th FAST*, pp. 375-390, 2017.

[6] Jingpei Yang et al., “AutoStream: Automatic Stream Management for Multi-streamed SSDs,” in *Proceedings of The 10th ACM International Systems and Storage Conference*, pp. 3:1-3:11, 2017.