로그 구조 파일시스템의 SSR 모드로 인한 단편화 현상 분석

Analyses on File Fragmentation caused by SSR Mode of Log-structured Filesystem

요 약

로그 구조 파일시스템은 여유 공간 확보를 위해 가비지 컬렉션을 수행한다. 하지만 가비지 컬렉션은 추가적인 쓰기 요청 및 I/O 처리 지연 현상을 유발하는 단점이 있다. 이에 대표적인 로그 구조 파일시스템인 F2FS는 가비지 컬렉션 대신 무효화 된 블록에 대해 in-place update를 수행하는 SSR 모드를 지원한다. SSR 모드의 로그 구조 파일시스템은 높은 I/O 성능을 제공하지만 임의 쓰기 요청을 유발한다. 본 논문에서는 로그 구조 파일시스템에서 가비지 컬렉션 및 SSR 모드를 수행할 경우, 각각의 I/O 성능 및 쓰기 패턴에 대해 비교, 분석을 수행하였다. 실험 결과, SSR 모드를 수행할 경우 기존 가비지 컬렉션을 수행할 때에 비해 2.46배 더 높은 I/O 성능을 보였지만 파일 단편화가 1.7배 더 많이 발생되는 것을 확인하였다.

1. 서 론

최근 모바일 디바이스 시장이 커져감에 따라 높은 성능, 작은 크기, 낮은 전력 소모량 등의 장점을 가지고 있는 플래시 스토리지가 널리 사용되고 있다. 또한 플래시 스토리지는 out-of-place update, 읽기/쓰기 단위와 지우기 단위가 다르다는 등의 특징을 가지고 있다. 이러한 구조적 한계를 극복하기 위해서 임의 쓰기를 순차 쓰기로 변경해 처리하는 로그 구조 파일시스템이 제안되었다[1]. 로그 구조 파일시스템은 호스트가 임의 쓰기를 발행하여도 이를 순차 쓰기로 변경해주기 때문에 기존 파일시스템보다 더 높은 성능을 보여준다.

로그 구조 파일시스템은 새로운 빈 공간(free space)을 확보하기 위해 가비지 컬렉션(garbage collection, GC)을 수행한다. 가비지 컬렉션을 수행하게 되면 유효 데이터를 지닌 블록을 다른 위치의 빈 공간으로 옮기게 되는데 이러한 과정은 추가적인 쓰기 요청을 유발하고 I/O 처리 성능을 떨어트리는 주요 원인이 된다. 이와 같은 성능 오버헤드를 완화시키기 위해 무효화된 블록(invalid block)에 쓰기 동작을 수행하도록 하는 slack space recycling (SSR) 모드가 제안되었다[2]. 하지만 SSR 모드는 임의 쓰기를 발행하기 때문에 플래시 스토리지의 성능을 떨어트리는 단편화 문제를 발생시킨다.

플래시 스토리지는 내부 단편화 문제 해결 및 빈 공간 확보를 위하여 스토리지 내부의 가비지 컬렉션을 사용한다. 스토리지 가비지 컬렉션은 파일시스템 가비지 컬렉션과 마찬가지로 여러 데이터를 빈 공간에 옮겨야 하는 추가적인 쓰기 요청을 수행한다. 즉, 임의 쓰기로 인해 발생하는 단편화 문제는 빈번한 스토리지 가비지 컬렉션을 유발한다. 이로 인해 추가적인 쓰기 요청이 생기고 이는 시스템 I/O 성능을 떨어트림과 동시에 플래시 스토리지의 내구성 또한 악화시킨다.

본 논문에서는 대표적인 로그 구조 파일시스템인 F2FS에서 SSR 모드 및 가비지 컬렉션을 수행할 때 I/O 성능 및 파일시스템 단편화 정도를 측정해본다.

2. 배 경

2.1. 로그 구조 파일시스템(log-structured filesystem)

로그 구조 파일시스템이란 데이터를 업데이트 시 덮어쓰기 방식이 아니라 기존 유효 데이터를 무효화하면서 데이터를 순차적으로 저장하는 파일시스템이다[1]. LFS는 스토리지 공간을 세그먼트 단위로 나누어 관리하는데 각 세그먼트는 512개의 4KB 블록으로 이루어져 있다. LFS는 임의 쓰기 요청이 들어와도 순차 쓰기로 변환시켜 수행하기 때문에 더 높은 쓰기 성능을 보여준다. 하지만 무효화된 블록을 회수하기 위한 가비지 컬렉션 작업이 필요하다. 이러한 가비지 컬렉션은 유효 블록들을 복사하는 과정의 성능 오버헤드를 유발하게 된다. 이에 가비지 컬렉션으로 인한 성능 오버헤드를 완화시키기 위한 SSR 모드가 제안되었다.

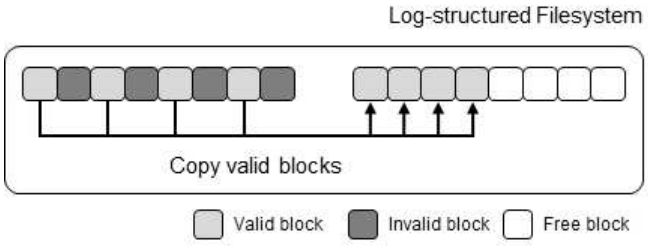

그림 1 garbage collection 동작 방식

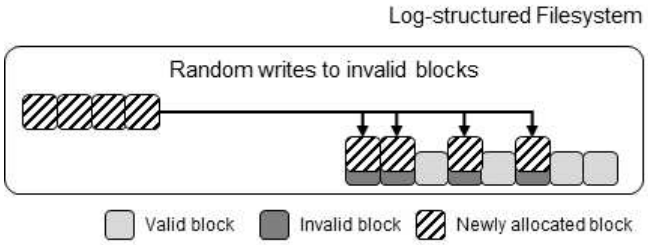

그림 2 Slack space recycling 동작 방식

2.2. 가비지 컬렉션(garbage collection)

로그 구조 파일시스템은 빈 공간 확보를 위하여 가비지 컬렉션을 사용한다. 가비지 컬렉션은 foreground와 background 두 가지 종류로 나눌 수 있는데 foreground 가비지 컬렉션은 유효 세그먼트가 충분하지 않을 때 수행된다. 만약 기존 쓰기 요구 처리되는 중에 가비지 컬렉션이 수행된다면 기존 쓰기 요청 처리는 가비지 컬렉션이 완료될 때까지 지연되어 응답시간에 큰 악영향을 주게 된다. 반면에 background 가비지 컬렉션은 I/O가 수행되지 않을 때 주기적으로 처리되므로 성능에 큰 영향을 주지 않는다.

가비지 컬렉션은 그림 1과 같이 동작한다[4]. 우선 희생 세그먼트(victim segment) 선정 알고리즘을 통해 유효 블록들을 이주시킬 희생 세그먼트를 선정한다. 그 다음, 희생 세그먼트 내에 존재하는 유효 블록들을 빈 세그먼트로 복사한다. 그리고 마지막으로 희생 세그먼트를 프리 세그먼트(free segment)로 변경하여 가비지 컬렉션을 끝내게 된다. 가비지 컬렉션의 과정 중 유효 블록들을 빈 세그먼트로 복사하면서 추가적인 쓰기 요청 오버헤드가 발생하기 때문에 파일시스템 I/O 성능이 저하된다.

2.3. Slack space recycling(SSR)

SSR 모드란 로그 구조 파일시스템에서 foreground 가비지 컬렉션 성능 오버헤드를 극복하고자 제안된 기법이다[2]. 로그 구조 파일시스템에서 기존의 순차 쓰기와는 다르게 SSR 모드는 가비지 컬렉션을 수행하는 대신 무효화된 블록에 임의 쓰기를 수행한다. 이 때 새로운 데이터가 무효화된 블록에 덮어써짐으로써 가비지 컬렉션으로 인한 오버헤드를 피할 수 있다.

SSR 모드에서는 빈 공간이 적을 때 상대적으로 더 높은 성능을 보이지만 임의 쓰기를 수행한다는 단점이 있다. SSR 모드로 인한 임의 쓰기는 파일시스템을 단편화시키고 이로 인해 추후 읽기 및 쓰기 성능을 떨어트리게 된다. 또한 이러한 임의 쓰기는 플래시 스토리지 내부의

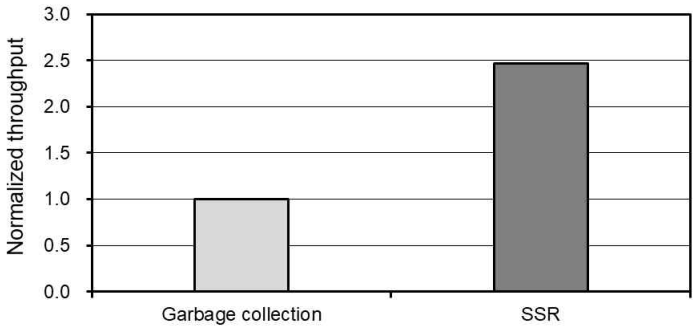

그림 3 가비지 컬렉션과 SSR 모드 간의 성능 비교

단편화를 발생시켜 플래시 스토리지 내부의 가비지 컬렉션을 더 빈번하게 발동시키고 수명 또한 단축시킨다[3].

3. 실험 환경 및 실험 결과

3.1. 실험 환경

대표적인 로그 구조 파일시스템인 F2FS에서 가비지 컬렉션 및 SSR 모드를 사용하였을 때 각각 성능을 측정해보고 SSR 모드로 인한 임의 쓰기 문제를 분석해본다. 이를 위해 표 1과 같이 Intel i7-3770K, 8GB DRAM, Samsung SSD 850 PRO 128GB 장비를 사용하였다. 4.15.0 버전의 커널과 Ubuntu 16.04 시스템에서 실험을 진행하였다. 버전을 사용하였다. 또한 파일시스템으로는 로그 구조 파일시스템 중 하나인 F2FS를 사용하였다[4, 5].

성능 측정 및 쓰기 패턴 분석에 앞서 8GB 크기의 파티션으로 나누어 스토리지 내부 가비지 컬렉션에 영향을 받지 않도록 세팅하였다. 7.5GB를 128KB 단위로 300초 동안 임의 쓰기 하여 성능을 측정하였고 가비지 컬렉션 및 SSR 모드에서 각각 발생하는 쓰기 요청의 패턴 분석을 위해 *blktrace*를 사용하였다.

표 1 실험 환경

CPU	Intel i7-3770K 3.50GHz
Memory	8GB (DDR4 4GB * 2)
Storage	Samsung SSD 850 PRO 128G
Operation system	Ubuntu 16.04 LTS
Filesystem	F2FS

3.2. 가비지 컬렉션과 SSR 모드 간 성능 비교

그림 3은 가비지 컬렉션을 사용하는 경우와 SSR 모드를 사용하는 경우의 I/O 처리량을 비교한 그래프이다. SSR 모드를 사용할 때 I/O 처리량이 가비지 컬렉션을 사용할 때보다 2.46배 더 높은 것을 확인 할 수 있다. 이는 앞서 언급하였듯 가비지 컬렉션은 유효 블록을 옮겨 적는 오버헤드가 존재하기 때문이다. 게다가 가비지 컬렉션이 완전히 동작을 마칠 때까지 I/O 요청이 처리되는 것을 중지시키기 때문에 I/O 응답시간 및 처리량을 상당히 떨어트린다. SSR 모드는 유효 블록을 빈 세그먼트로 복사하는 대신 무효화된 블록에 데이터를 저장함으로써 가비지 컬렉션을 수행할 때보다 더 높은 성능을 보여준다.

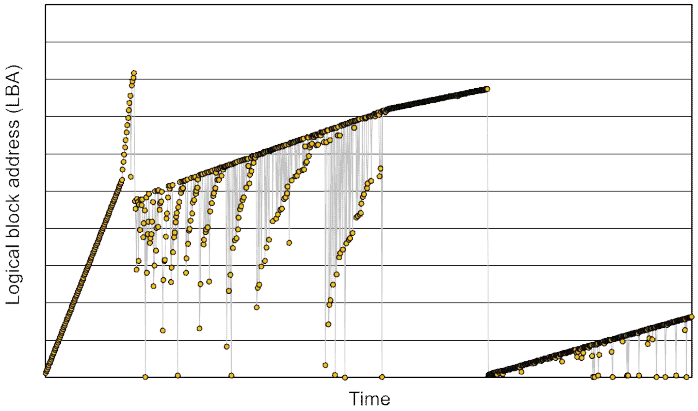

그림 4 가비지 컬렉션을 수행하는 로그 구조 파일시스템의 쓰기 패턴

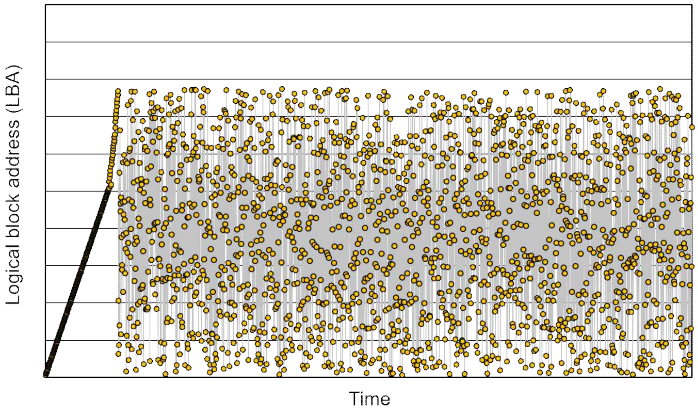

그림 5 SSR 모드로 수행하는 로그 구조 파일시스템의 쓰기 패턴

3.3. SSR 모드 및 가비지 컬렉션에서의 쓰기 패턴 비교

로그 구조 파일시스템은 데이터를 순차적으로 저장하는 파일시스템이다. 하지만 스토리지 용량의 제한으로 인해 순차적으로 데이터를 쓸 수 있는 양은 제한적이다. 따라서 빈 공간 즉, 빈 세그먼트를 확보하기 위해 앞서 언급했던 가비지 컬렉션이나 SSR 모드가 사용된다. 이는 데이터의 순차적 저장을 방해하는데 이때 각 기법이 쓰기 요청 패턴에 어떤 영향을 미치는지 알아보기 위하여 *blktrace*를 통하여 logical block address(LBA)를 확인해 보았다.

그림 4는 가비지 컬렉션 모드에서 쓰기 요청 LBA를 나타낸 그래프이다. 초반부에서는 데이터가 순차적으로 쓰이는 것을 확인 할 수 있지만 빈 공간이 소진됨에 따라 가비지 컬렉션이 수행되는 모습을 확인할 수 있다. 또한 가비지 컬렉션 작업에 이어서 데이터를 순차적으로 쓰기 요청을 처리하는 모습을 확인 할 수 있다. 그림 5는 SSR 모드에서의 쓰기 요청 LBA를 나타낸 그래프이다. 처음에는 빈 공간에 데이터를 저장하므로 가비지 컬렉션 모드의 초반부와 같이 순차적으로 쓰기 요청을 처리하는 모습을 보여준다. 하지만 빈 공간을 대부분 소진하게 되면 가비지 컬렉션 모드는 유효 블록을 옮겨 저장하며 빈 공간을 만들어 주는 것과 달리 SSR 모드에서는 임의 쓰기를 수행하는 것을 볼 수 있다. 이는 남아있던 빈 세그먼트에 데이터가 모두 쓰여서 무효화된 블록에 임의쓰기를 수행하였기 때문이다.

이러한 SSR 모드로 인한 임의 쓰기는 파일시스템 단편화 문제를 심화시킨다. 그림 6은 각 모드에서 저장한 파일 데이터의 LBA가 몇 조각으로 나누어져 있는 지를 확인하고 비교한 그래프이다. 리눅스의 *hdparm* 명령어를 사용해 파일이 얼마나 단편화 되어있는 지 확인한 결과, SSR 모드는 가비지 컬렉션보다 1.7배 더 단편화 된 것을 확인 할 수 있었다.

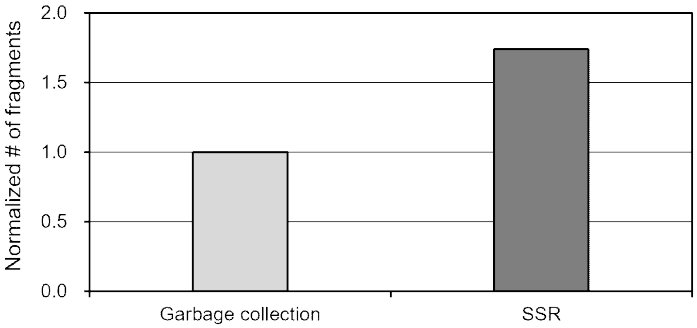

그림 6 가비지 컬렉션과 SSR 모드 간의 단편화

3. 결론 및 향후 계획

본 논문에서는 로그 구조 파일시스템에서 지원하는 가비지 컬렉션과 SSR 모드의 차이점을 알아보고 각 기법을 적용했을 때 I/O 처리 성능 및 임의 쓰기 문제에 대해 분석해 보았다. 가비지 컬렉션은 로그 구조 파일시스템에서 빈 공간 확보를 위해 사용되지만 I/O 처리 성능을 크게 떨어트린다. 이에 반해 SSR 모드는 가비지 컬렉션을 수행하지 않아 높은 I/O 성능을 보여주는 대신 임의 쓰기를 수행하기 때문에 플래시 스토리지 내부의 단편화 문제를 발생시키는 문제점이 있다. 추후 연구에서는 SSR 모드로 인해 발생하는 임의 쓰기 요청이 플래시 스토리지 내구성 및 I/O 처리 성능에 끼치는 영향을 확인하고 그에 따른 개선점을 연구할 예정이다.

참 고 문 헌

[1] M. Rosenblum and J. K. Ousterhout, "The Design and Implementation of a Log-structured File System," *ACM Transactions on Computer Systems,* Vol. 10, No. 1, pp. 26-52, 1992.

[2] Y. Oh, E. Kim, J. Choi, D. Lee, and S. H. Noh, "Optimizations of LFS with Slack Space Recycling and Lazy Indirect Block Update," *Proc. of the Annual Haifa Experimental Systems Conference,* pp. 1-9, 2010.

[3] C. Min, K. Kim, H. Cho, S. W. Lee, and Y. I. Eom, "SFS: Random Write Considered Harmful in Solid State Drives," *Proc. of the USENIX Conference on File and Storage Technologies,* pp. 273-286, 2012.

[4] C. Lee, D. Sim, J. Y. Hwang, and S. Cho, "F2FS: A New File System for Flash Storage," *Proc. of the USENIX Conference on File and Storage Technologies,* pp. 273-286, 2015.

[5] J. Kim, "F2FS," https://www.kernel.org/doc/Documentation/filesystems/f2fs.txt.