Evolving ext4 for SMR drives

(LWN 구독을 고려해봐)

Linux 스토리지를 위한 2017 Vault 컨퍼런스에서 Ted Ts'o와 Abutalib Aghayev는 shingled magnetic recording SMR 드라이브에서 ext4 파일 시스템이 더 잘 작동하도록 하기 위해 몇 가지 수행 작업에 대해 이야기함. 이러한 장치는 기존 드라이브와 몇 가지 주요 차이점이 있어 임의 쓰기 성능이 매우 저조. ext4에 적용된 아주 작은 변경 사항은 SMR 드라이브의 성능에 극적인 영향을 미쳤으며 기존 자기 기록 CMR 미디어에서도 메타데이터가 많은 작업 부하에 대한 성능 향상을 제공.

Ts'o는 그와 Carnegie Mellon University의 학생인 Aghayev가 ext4 변경 사항을 개발했다고 말함. CMU의 Garth Gibson 교수와 Northeast의 Peter Desnoyers 교수도 유용한 정보와 조언을 제공.

SMR basics

SMR 드라이브는 platter의 track을 겹쳐서 CMR 드라이브와 동일한 공간에 더 많은 데이터를 포장. 순차 쓰기는 SMR에서 잘 작동하지만, 기존 데이터를 덮어쓰려면 인접 track에서 데이터를 복사하고 순차 방식으로 다시 써야 함. SMR은 백업, 대용량 미디어 파일 및 유사한 사용 사례를 대상으로 함.

Ts'o는 회전 드라이브가 계속 유지되는 한 SMR은 우리와 함께할 것이라고 말함. SMR과 호환되는 "coming down the pike(???)" 추가 기술이 존재. 수백만 개의 SMR 드라이브가 출시되었으며 소비자들까지도 이 기술을 백업에 사용하고 있으며 더 빠른 액세스가 필요한 데이터에는 SSD를 사용.

SMR 드라이브에는 드라이브 관리 및 호스트 관리의 두 가지 종류가 존재. 이 강연은 운영 체제가 스토리지 장치를 능동적으로 관리해야 하는 호스트 관리가 아닌 드라이브 관리 SMR에 초점을 맞춤. 드라이브 관리형 SMR의 경우 SSD용 FTL(flash translation layer)과 유사한 STL(shingle translation layer)이 존재. STL은 크기가 256M이고 순차적으로 작성되어야 하는 SMR 장치의 다양한 영역을 숨깁니다. 장치의 제약 조건을 가리는 블록 인터페이스를 나타냄.

SMR 디스크에는 일반적으로 CMR 디스크보다 훨씬 빠른 영구 캐시가 존재. 이론은 유휴 시간이 있고 enterprise 설정에 있는 대부분의 디스크에 일부가 있을 경우 해당 시점에 데이터를 영구 캐시에서 디스크 자체로 순차적인 방식으로 이동할 수 있다는 것. 또한, 유휴 시간은 다양한 청소 및 가사 작업을 허용. 영구 캐시에 공간이 있는 한 장치에 대한 쓰기는 매우 빠르지만, 가득 차면 처리량이 급격히 떨어짐.

영구 캐시는 공급업체가 크기 및 기타 특성을 쿼리하는 마법 명령을 제공하지 않는 한 커널에 표시되지 않음. STL의 정확한 동작은 공급업체에 따라 다르며 FTL 구현 상황과 마찬가지로 변경될 수 있음. 그러나 플래시는 너무 빨라서 번역 계층이 다른 위치에 쓰기를 선택할 때 차이를 알아차리기 어려움. STL의 경우 영구 캐시에 쓰는 것이 디스크에 쓰는 것보다 훨씬 빠름.

STL은 순차적 쓰기를 인식하고 이에 대한 영구 캐시를 우회하려고 시도. Ts'o는 어떤 면에서 영구 캐시는 ext4 저널과 비슷하다고 말함. 임의 쓰기 워크로드의 경우 영구 캐시가 가득 차면 각 쓰기가 대규모 읽기-수정-쓰기 작업이 됨. 영구 캐시의 정확한 세부 정보, 예를 들어 디스크에 있는 캐시의 양과 위치는 다양함. 그들이 테스트한 일부 드라이브에는 25GB의 영구 캐시가 있었고 다른 드라이브는 다름.

Small changes

그와 Aghayev가 한 작업은 극적인 차이를 만든 ext4에 아주 작은 변경(40줄의 코드 수정, 새 파일에 600줄 추가)을 가하는 것. 이러한 변경으로 메타데이터가 적은 워크로드의 성능이 1.7~5.4배 향상되었습니다. 메타데이터가 많은 워크로드의 경우 2~13배 향상.

그는 ext4가 디스크를 사용하는 방식이 특히 SMR 장치에 좋지 않다고 언급. 메타데이터가 디스크 전체에 퍼져 있기 때문. 메타데이터 쓰기는 4KB 임의 쓰기이며, 이는 SMR에서 가능한 최악의 상황. 이러한 쓰기는 SMR에 적합한 대용량 비디오 스트림을 저장하는 경우에도 STL이 수행해야 하는 작업을 지배(dominate)할 수 있음. 유휴 시간이 많으면 변화가 그렇게 극적이지는 않지만 그렇지 않은 경우 STL이 임의 쓰기를 순차적 쓰기로 바꾸는 동안 성능이 크게 떨어짐.

Ext4는 미리 쓰기(writeahead) 로깅을 사용. 즉, 순차적으로 메타데이터를 저널에 기록하지만 저널이 채워지거나 dirty 쓰기 저장 시간 초과(writeback timeout)에 도달하면 메타데이터를 최종 위치에 저장하기 위해 임의 쓰기를 수행. 즉, 모든 메타데이터 블록은 저널에 한 번, 최종 대상에 한 번으로 두 번 작성; 저널 항목을 권위 있는 항목으로 사용하지 않는 이유는 무엇인가? 메모리의 블록은 저널 위치에 매핑되고 페이지 캐시에서 깨끗한 것으로 표시될 수 있음; 제거되어 다시 필요한 경우 저널에서 조회할 수 있음.

그러나 저널이 가득 차면 해야 할 일이 있음. 저널의 많은 블록은 나중에 저널의 항목에 의해 업데이트되었기 때문에 실제로 중요하지 않음. 여전히 유효한 경우 디스크의 최종 위치에 복사하거나 단순히 순차 쓰기로 새 저널에 복사할 수 있음. "눈을 가늘게 뜨면" 메타데이터를 위한 로그 구조의 파일 시스템처럼 보임.

이 모든 작업을 수행하기 위해 저널을 128MB에서 10GB로 늘림. "8TB 드라이브에서 친구 사이에 10GB는 무엇입니까?" 그들은 더 작은 저널을 시도했지만 저널은 더 빨리 채워져 더 많은 복사가 필요.