https://doi.org/10.5392/JKCA.2017.17.07.093

고성능 플래시 SSD 환경에서 NoSQL 데이터베이스의 성능 평가 및 최적화

Performance Evaluation and Optimization of NoSQL Databases with High-Performance Flash SSDs

한혁
동덕여자대학교 컴퓨터학과

Hyuck Han(hhyuck96@dongduk.ac.kr)

요약

최근 사회 관계망 서비스, 클라우드 컴퓨팅, 슈퍼컴퓨팅, 기업용 스토리지 시스템 등의 분야에서 고성능 플래시 메모리 기반 저장 장치(플래시 SSD)에 대한 수요가 크게 증가하고 있다. 이러한 환경에서 최근 산업계 및 학계에서는 고성능 플래시 SSD를 위한 NVMe 규약을 만들었고, NVMe 규약을 따르는 고성능 플래시 SSD는 현재 시장에서 구할 수 있다. 본 논문에서는 NVMe 플래시 SSD를 이용하여 클라우드 컴퓨팅, 사회 관계망 서비스 등에서 많이 활용되고 있는 NoSQL 데이터베이스의 성능을 평가하고 분석하고자 한다. 성능 평가에 사용된 저장 장치는 삼성전자가 최근에 개발한 NVMe 기반 플래시 SSD이며 이 장치의 연속 읽기/쓰기 성능은 3.5GB/s 이다. NoSQL 데이터베이스는 MongoDB의 기본 스토리지 엔진으로 채택된 WiredTiger를 사용하였다. 실험 결과는 고성능 NVMe 플래시 SSD 환경에서 NoSQL 데이터베이스의 로그 처리 부분이 성능상의 가장 큰 오버헤드임을 보여준다. 이 결과를 바탕으로 로그 처리 부분을 최적화하였고 최적화된 WiredTiger는 기존 대비 최대 15배의 성능 향상을 보여준다.

■ 중심어 : | NVMe 플래시 SSD | NoSQL 데이터베이스 | WiredTiger |

Abstract

Recently, demands for high-performance flash-based storage devices (i.e., flash SSD) have rapidly grown in social network services, cloud computing, super-computing, and enterprise storage systems. The industry and academic communities made the NVMe specification for high-performance storage devices, and NVMe-based flash SSDs can be now obtained in the market. In this article, we evaluate performance of NoSQL databases that social network services and cloud computing services heavily adopt by using NVMe-based flash SSDs. To this end, we use NVMe SSD that Samsung Electronics recently developed, and the SSD used in this study has performance up to 3.5GB/s for sequential read/write operations. We use WiredTiger for NoSQL databases, and it is a default storage engine for MongoDB. Our experimental results show that log processing in NoSQL databases is a major overhead when high-performance NVMe-based flash SSDs are used. Furthermore, we optimize components of log processing and optimized WiredTiger show up to 15 times better performance than original WiredTiger.

■ keyword : | NVMe Flash SSD | NoSQL Database | WiredTiger |

* 이 논문은 2016년도 동덕여자대학교 학술연구비 지원에 의하여 수행된 것임.
접수일자 : 2017년 03월 06일
수정일자 : 2017년 06월 14일
심사완료일 : 2017년 06월 27일
교신저자 : 한혁, e-mail : hhyuck96@dongduk.ac.kr

I. 서 론

최근 플래시 메모리 기반 저장 장치는 (플래시 SSD) 데이터센터 및 사회 관계망 서비스, 기업 용 스토리지 서버 시스템에 널리 채택되어 사용되고 있다. 특히 플래시 SSD는 하드디스크에 비해 수십 배 향상된 대역폭과 빠른 지연시간을 제공한다. 현재 NetApp과 IBM과 같은 다양한 스토리지 시스템 제조업체들이 하드디스크를 플래시 SSD로 대체하여 Flash Array[1] 및 EF540[2] (NetApp), 그리고 FlashSystem 820[3] (IBM) 등과 같은 제품들을 출시하였다.

저장 장치내의 소자 측면의 발전과 더불어 저장장치와 호스트 컴퓨터 간 인터페이스 상의 발전은 플래시 SSD의 성능을 크게 향상시켜왔으며 이는 전반적인 컴퓨터 시스템의 성능을 높이고 있다. 기존의 하드디스크를 위한 SAS 또는 SATA 인터페이스들은 플래시 SSD의 성능 및 특징을 최대한 활용하지 못하게 하였고, 이러한 점을 해결하기 위해 산업계와 학계는 표준 NVMe (Non-Volatile Memory Express)[4] 규약을 고안하였다. 그리고 삼성, 인텔과 같은 주요 플래시 SSD 제조업체들이 PCIe 기반의 NVMe 플래시 SSD를 출시하였다. 이러한 PCIe 기반의 NVMe 플래시 SSD는 SAS/SATA 기반의 플래시 SSD보다 훨씬 더 좋은 성능을 보인다.

NVMe를 위한 호스트 컨트롤러 인터페이스는 확장성을 지녔으며 SATA 인터페이스와는 달리 CPU 코어당 전송/완료 큐를 각각 지원하여 병렬적으로 입출력 연산들을 처리할 수 있도록 해준다. 또한 64,000개의 입출력 큐들을 가지며 하나의 큐에는 64,000개의 입출력 명령어를 처리할 수 있도록 한다. 본 논문은 이러한 특징을 가진 PCIe 기반의 NVMe 플래시 SSD 장비를 탑재한 서버 시스템에서 NoSQL 데이터베이스의 성능을 평가하고 분석한다. 이를 위해 최근 MongoDB[5]의 기본 스토리지 엔진으로 채택된 WiredTiger [6] 시스템을 이용하였다. 본 연구에서 사용되는 플래시 SSD는 최근 삼성전자에서 개발된 NVMe 플래시 SSD이며 이는 4KB 입출력 요청 크기일 때 최대 읽기/쓰기 성능 3.5GB/s를 제공한다. 이는 기존 SATA 인터페이스 기반 플래시 SSD의 성능 (최대 읽기 성능 540MB/s, 쓰기 성능 520MB/s)에 비해 읽기는 6.4배, 쓰기는 6.7배 높은 성능을 보인다. 실험 결과는 입출력 성능이 우수한 NVMe 플래시 SSD를 이용하더라도 NoSQL 데이터베이스 시스템의 로그 레코드 쓰기 부분이 가장 큰 오버헤드임을 보여준다.

이러한 실험 결과를 바탕으로 WiredTiger의 로그 처리 부분을 최적화하였다. 기존 WiredTiger는 로그 쓰기 연산이 완료되어야 트랜잭션이 끝나지만, 최적화된 WiredTiger에서는 트랜잭션의 안전성을 훼손하지 않으면서 로그 쓰기 연산을 비동기 쓰기 연산으로 대체하여 성능 향상을 달성한다. 이와 같은 방법으로 최적화된 로그 처리가 가능한 WiredTiger는 기존 대비 최대 15배 정도의 더 향상된 성능을 보여준다.

이 논문의 나머지 부분은 다음과 같이 구성된다. 2장은 본 연구와 관련된 연구에 관해 설명하고, 3장에서는 본 논문과 관련된 배경에 대해 논의한다. 4장은 NoSQL 데이터베이스 시스템의 성능 평가 결과를 분석과 함께 설명한다. 5장에서는 4장의 결과를 바탕으로 최적화 기법과 그 효과를 설명한다. 마지막으로 6장에서는 논문의 결론을 내린다.

II. 관련 연구

최근에 비휘발성 메모리 기반 저장 장치의 성능을 평가하고 최적화하는 연구들이 진행되어 왔다. [10]의 연구에서는 입출력 시간 특히 읽기 지연 시간의 측면에서 플래시 메모리와 비교하여 PCM이 시스템의 성능을 향상시킬 수 있는지를 보였다. 또한 PCM을 스토리지 서버의 캐시로 사용하는 것이 비용 대비 성능이 좋아짐을 보였다. [11]의 연구에서는 플래시 SSD가 트랜잭션 처리에 있어서 하드 디스크 대비 수십 배 이상의 성능 향상을 가져올 수 있음을 보였다. [12]의 연구에서는 NVMe 기반 플래시 SSD과 하드 디스크를 이용하여 하둡 파일 시스템을 평가하였고, [13]에서는 NVMe 기반 플래시 SSD를 탑재한 서버 시스템에서 파일 시스템의 성능을 평가한 결과를 보였다.

또한 SSD를 탑재한 시스템에서 NoSQL 데이터베이스의 성능을 높이려는 연구들이 진행되고 있다. [14]의 연구에서는 SSD에 멀티 스트림이라는 새로운 기능을 구현하여 NoSQL 데이터베이스의 성능을 높일 수 있음을 보였다. [15]의 연구에서는 SSD에 최적화된 데이터 레이아웃을 가지게 하여 NoSQL 데이터베이스 중의 하나인 LevelDB의 성능을 높였으며 [16]의 연구에서는 NoSQL 데이터베이스의 주요 인터페이스인 Get/Put을 SSD에서 구현하여 NoSQL 기반 응용의 성능을 높였다. 본 연구는 이러한 연구들과는 달리 NoSQL 데이터베이스 로그 처리 부분에 초점을 맞춘다.

III. 배 경

이 장에서는 본 논문의 배경에 대해 설명한다. 먼저 고성능 플래시 SSD 및 그 인터페이스에 대해 설명하고 그 다음으로 본 연구에서 사용된 NoSQL 데이터베이스인 WiredTiger에 대해 설명한다.

2.1 고성능 플래시 SSD

플래시 SSD는 최근 많은 스토리지 시스템에서 활용되면서 고성능 입출력을 요구하는 분야에서 하드디스크를 대체하고 있다. 플래시 SSD는 하드디스크가 가지는 기계적 오버헤드가 없으며 이로 인해 하드디스크 대비 수십 배 낮은 지연시간과 높은 대역폭을 제공한다. 이러한 장점들로 인해 플래시 SSD를 위한 저장장치와 호스트 간 인터페이스 기술은 빠르게 발전하고 있다. 기존의 하드디스크에서 주로 사용되는 SATA, SAS 인터페이스는 플래시 SSD의 성능을 최대로 이용하는 것이 어려웠다. 이를 해결하기 위해 학계 및 산업계는 기존의 순차적인 입출력 처리에 초점을 맞춘 인터페이스 규약에서 병렬적 처리의 이점을 최대로 이용할 수 있는 NVMe 규약를 고안하였고, NVMe 규약은 컴퓨터 시스템 내부의 PCIe 버스와 함께 사용되었다. PCIe 버스는 기존의 다른 버스에 비해 플래시 SSD의 성능을 상당히 증가시켰다.

NVMe 규약은 확장성을 중점으로 설계되었으며 PCIe 기반 플래시 SSD의 성능을 최대한 이끌 수 있도록 설계되었다. NVMe 규약은 최적화된 레지스터, 명령어 집합 등을 정의하며 저장 장치 당 하나의 입출력 큐를 가지고 있는 기존 인터페이스 규약과는 달리 CPU 코어당 입출력 큐를 가져서 입출력 처리에 있어서 병렬성을 높였다. 이러한 설계는 입출력 큐에 대한 자료구조를 각각의 프로세서 코어의 캐시에 생성하고 큐에 대한 락 경합을 줄임으로써 입출력 병렬성을 극대화시킨다. 또한 NVMe 규약을 위한 리눅스 입출력 소프트웨어 구현은 기존 규약을 위한 입출력 소프트웨어보다 더 최적화된 커널 입출력 스택을 가진다. 기존 규약에서는 입출력 요청이 블록 계층, SCSI 및 SATA 인터페이스 등을 거쳐서 처리되기 때문에 소프트웨어 오버헤드가 발생되었지만, NVMe 규약 상에서는 블록 계층을 거쳐 바로 호스트 NVMe 명령어를 생성한 후 저장 장치로 전송하여 소프트웨어 오버헤드를 줄이고 있다.

2.2 NoSQL 데이터베이스 및 WiredTiger

최근 사회 관계망 등에서 사용되는 대용량 시스템의 특징은 특정 고객이 아닌 전 세계 사람들을 대상으로 한다는 점이며 이는 기존의 시스템에서 볼 수 없었던 매우 단순한 형태지만 매우 큰 규모의 데이터를 생산해 냈다. 대용량 단순 데이터 처리에 대한 요구가 가장 큰 구글과 아마존은 빅테이블(Bigtable)과 Dynamo 시스템을 개발하였다. 이것은 관계형 데이터베이스 중심의 데이터 처리 기술과는 다른 형태인 새로운 데이터 저장 기술인 NoSQL 기술의 대표적인 예가 되었다.

본 논문에서는 고성능 NVMe 기반 플래시 SSD를 탑재한 서버 시스템에서 WiredTiger NoSQL 데이터베이스의 성능을 평가한다. WiredTiger 시스템은 고성능, 확장성 있는 트랜잭션 처리를 상용화 수준의 품질로 제공해주는 오픈 소스 데이터 스토리지 엔진이다. [그림 1]과 같이 WiredTiger는 최근에 MongoDB, MySQL에서 사용되고 있다.

WiredTiger는 데이터의 영속성 보장을 위해 체크포인트 수준 영속성, 커밋 수준 영속성 방식을 제공하고 있다. 체크포인트 수준 영속성은 세션이 종료될 때 혹은 사용자 응용 프로그램이 체크포인트를 남기는 호출

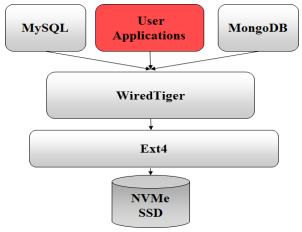

그림 1. WiredTiger의 에코 시스템

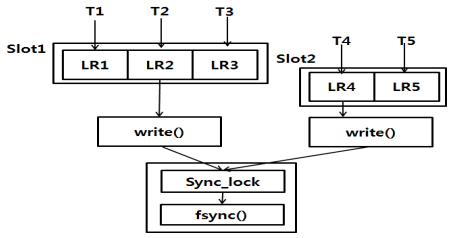

그림 2. WiredTiger 시스템의 로그 처리 과정
(T: 트랜잭션, LR: 로그 레코드, Slot: 로그 버퍼)

을 할 때 변경된 데이터가 저장 장치에 반영되는 방식으로 영속성을 보장해준다. 커밋 수준 영속성은 기존 데이터베이스에서 많이 사용하는 방식으로 WAL (write- ahead logging) 기법을 사용하여 영속성을 보장해준다. WiredTiger는 캐시 영역에 레코드들의 여러 버전을 유지하는 방법을 이용하여 다 버전 병행 제어 (Multiversion Concurrency Control/MVCC)를 제공한다. 바 버전 병행 제어의 read-committed 격리 수준에서는 읽기 트랜잭션은 가장 최근에 커밋된 버전의 레코드를 읽는다.

2.3 WiredTiger의 로그 처리

WAL을 사용하고 있는 데이터베이스에서는 모든 로그 레코드들은 로그 반영 순서를 위해 로그 시퀀스 숫자를 (LSN) 사용하여 구별된다. 즉, 로그 레코드 LR1의 LSN이 다른 로그 레코드 LR2의 LSN보다 작고 LR2가 저장 장치에 안전하게 써졌다면 LR1 역시 저장 장치에 써졌음이 보장된다. 커밋 수준 영속성을 보장하기 위해서는 데이터베이스 시스템은 트랜잭션의 로그 레코드를 위해 LSN을 부여하고 로그 레코드를 로그 버퍼에 저장하고 저장 장치의 미디어까지 쓰는 것을 락과 같은 상호 배제를 이용하여 안전하게 처리한다. 이것은 로그 레코드가 순서대로 저장 장치에 써지는 것을 보장하기 위해서이다.

기존의 로그 버퍼에 로그 레코드를 추가하여 처리하는 방식과는 달리 WiredTiger는 consolidation array 방법[7]을 이용하여 로그 처리 오버헤드를 줄여준다. 기존의 방법에서는 로그 레코드에 LSN을 부여하여 로그 버퍼에 추가하는 전체 과정이 락을 통하여 보호되지만, [그림 2]와 같이 consolidation array 방법에서는 로그 버퍼를 슬롯으로 나누어 하나의 슬롯에 여러 트랜잭션이 로그 레코드를 락 없이 동시에 복사할 수 있다. 즉, 로그 버퍼에 로그 레코드들이 병행적으로 복사될 수 있고 슬롯에 참여하고 있는 트랜잭션들 중에서 하나의 트랜잭션이 write와 fsync 함수를 호출한다. 즉, 상호 배제에 의한 자원 (로그 버퍼) 경쟁이 줄어드는 효과가 있으며, 슬롯에 추가된 로그 레코드 전체를 한 번에 저장 장치에 쓴다.

IV. 문제점 분석

3.1 실험 환경

이번 장에서는 WiredTiger 데이터베이스를 이용하여 고성능 플래시 SSD의 성능을 평가하기 위한 실험 환경에 대해 설명한다. 성능 평가를 위해 Intel Xeon CPU E5-2670 2.6GHz를 장착한 서버를 사용하였다. 이 서버 시스템은 16개의 코어와 16GB 메인 메모리를 가지고 있으며, 이 서버 시스템을 위해 Linux 커널은 3.14.3 버전과 EXT4 파일 시스템[8]을 구동하였다. WiredTiger의 트랜잭션 격리 수준은 read-committed 수준, 인덱싱을 위해서 B-tree를 사용하도록 설정하였다. WiredTiger 데이터베이스의 캐시 크기는 5GB, 로그 파일의 크기는 100MB이다. 트랜잭션 영속성은 트랜잭션 커밋 수준으로 설정되었다.

실험을 위한 트랜잭션은 다음과 같은 워크로드를 가진다. WiredTiger 데이터베이스의 하나의 엔트리의 키 크기는 20B, 값 크기는 1KB이며 각각의 트랜잭션은 하나의 엔트리만 업데이트한다. 이 워크로드는 Yahoo가

제공하고 있는 클라우드 컴퓨팅 벤치마크[9]의 (YCSB, Yahoo Cloud Serving Benchmark) 업데이트 워크로드와 유사하다. 총 엔트리의 개수는 50,000,000개이며 트랜잭션은 임의의 키를 생성하여 해당 엔트리를 업데이트한다. 데이터베이스 쓰레드는 100개이며, 각 쓰레드는 총 500,000개의 트랜잭션을 수행한다. 본 평가에서 사용되는 저장 장치는 최근 삼성전자가 개발한 NVMe 기반 플래시 메모리 SSD이며 이 장치는 용량이 4TB이며, 4KB요청 사이즈일 때 최대 읽기/쓰기 성능 3.5GB/s를 보여준다.

트랜잭션 처리 시간의 세밀한 분석을 위해 WiredTiger의 소스 코드를 수정하여 트랜잭션 처리 컴포넌트별 실제 실행 시간(wall clock time)을 측정하였다. 실험에 사용된 트랜잭션은 크게 5단계로 수행 단계를 나눌 수 있으며 각 단계는 다음과 같다. 트랜잭션 시작을 위한 BEGIN, 업데이트하고자 하는 엔트리의 키와 값을 생성하고 WiredTiger 자료구조에 채우는 SET_KEY_VALUE 단계, 생성된 키를 이용하여 엔트리를 찾는 SEARCH, 생성된 값을 이용하여 엔트리를 업데이트하는 UPDATE, 마지막으로 트랜잭션의 처리 완료를 담당하는 COMMIT 단계이다.

3.2 성능 평가 결과 및 분석

3.1절의 실험 환경에서 트랜잭션 처리 컴포넌트별 실제 실행 시간을 분석하였다. 트랜잭션 수행 단계들 중에서 COMMIT 단계가 전체 실행 시간의 99% 이상을 차지하고 있음을 확인할 수 있었고 이것은 NVMe 플래시 SSD와 같이 고성능 저장 장치를 탑재한 서버 시스템에서 트랜잭션의 영속성을 위한 COMMIT 부분이 가장 오버헤드가 된다는 것이다.

COMMIT 단계에서 대부분의 실제 실행 시간을 차지하고 있는 부분은 로그 레코드 처리 부분으로 약 99% 정도 된다. 2.3 절에서 설명한 바와 같이 WiredTiger 시스템은 consolidation array 기법을 사용하여 상호 배재에 의한 자원 경쟁을 줄이고 저장 장치의 효율을 높였다. 이 기법은 WiredTiger 소스 코드의 log_write_internal() 함수에 구현이 되어 있으며, 이 함수는 크게 다음의 4가지 함수를 수행하면서 로그 레코드를 처리한다. 트랜잭션이 슬롯에 참여하는 slot_join, 슬롯을 새로 시작한 트랜잭션이 해당 슬롯을 닫을 때까지 그 슬롯에 참여한 다른 트랜잭션이 기다리는 slot_wait, 슬롯에 마지막으로 로그 레코드를 추가한 트랜잭션이 그 슬롯의 모든 로그 레코드를 저장 장치에 쓰는 log_release, 슬롯의 로그 레코드들이 모두 저장 장치가 써질 때까지 다른 트랜잭션들이 기다리는 log_cond_wait 함수이다.

로그 처리를 위한 함수 별 실제 실행 시간의 비율을 분석한 결과는 [표 1]과 같으며, 로그 레코드를 쓰거나 다 써지기를 기다리는 데 전체 실행 시간의 82.8% 사용하고 있었다. 고성능 저장 장치가 사용되고 있음에도 로그 레코드를 저장 장치에 쓰는 것이 제일 큰 오버헤드이다. 또한, 로그 레코드들을 쓰는 연산이 비동기적으로 수행되는 것으로 볼 때 CPU의 자원이 효율적으로 사용되지 못하고 있음을 유추할 수 있다.

표 1. 함수 별 실행 시간 비율 분석 (로그처리)

함수	비율(%)
slot_join	0.02
slot_wait	17.18
log_release	13.8
log_cond_wait	69
합계	100

[표 2]는 CPU 사용률을 분석한 결과이다. 앞서 설명한 바와 같이 로그 레코드들을 비동기적으로 쓰거나 써지길 기다리는 부분이 많기 때문에 전체 실행 시간의 89% 이상을 CPU를 사용하지 않고 sleep하는데 보내고 있음을 확인하였다. 이 결과는 데이터베이스 쓰레드의 수를 늘려도 비슷한 결과를 얻었다. 이 결과는 NoSQL 데이터베이스 시스템은 고성능 저장 장치에 로그 레코드를 효율적으로 처리하고, CPU 활용률을 높이는 방법이 필요하다는 것을 의미한다.

표 2. CPU 사용률 분석

	비율(%)
WiredTiger	6
Linux Kernel	5
합계	11

V. 제안기법

이 장에서는 앞서 분석한 것과 같이 고성능 저장 장치를 탑재한 시스템에서 WiredTiger의 성능 상의 가장 큰 오버헤드인 로그 처리를 최적화 기법에 대해 설명한다. 기존의 slot array를 기반으로 한 로그 처리 기법은 로그 버퍼에는 트랜잭션들이 동시에 로그를 복사하고 이 중 하나의 트랜잭션을 처리하는 쓰레드가 저장 장치에 로그를 쓴 후에 모든 트랜잭션들이 COMMIT 완료된다. 이때 다른 쓰레드들은 로그가 써질 때까지 대기하는데 이것이 시스템의 CPU 효율을 저하시키는 원인이 된다.

표 3. 설계 중점 사항 및 설명

설계 중점 사안	설명
Sync lock 경쟁 완화	Sync lock이 기존의 fsync 호출 대신에 write I/O 버퍼로의 복사를 보호
CPU 사용률 개선	Flush thread가 로그 I/O를 수행하는 동안에 트랜잭션 쓰레드는 다른 트랜잭션 수행

[표 3]에서 제시된 바와 같이 최적화된 로그 처리 기법에서는 lock 경쟁을 완화하고 CPU 사용률을 개선하기 위해 다음과 같은 흐름으로 로그 처리를 진행한다. 로그 버퍼에 동시에 로그를 복사한 후 하나의 쓰레드가 별도의 I/O 버퍼에 복사를 한 후에 sync 락을 해제한다. 그 후 모든 쓰레드들은 수행 중이던 트랜잭션의 상태를 저장하고 로그 쓰기 연산이 수행되는 동안 다른 트랜잭션들을 수행한다. 이 때 기존의 트랜잭션의 상태는 COMMIT으로 변경되지 않기 때문에 트랜잭션의 안전성은 훼손되지 않는다. 로그 쓰기 연산은 별도의 Flush 쓰레드가 수행하며 로그 쓰기 연산이 완료되면 해당 로그와 연관되어 있는 트랜잭션의 상태를 COMMIT으로 변경한다. 그 후에 트랜잭션 쓰레드들은 COMMIT으로 표시된 트랜잭션 처리를 끝낸다.

[그림 3]은 최적화된 로그 처리의 예를 보여준다. 트랜잭션 쓰레드 T1, T2, T3는 Tx1, Tx2, Tx3 트랜잭션을 수행하고 있으며, T1, T2, T3 쓰레드들은 동시에 로그를 동시에 로그 버퍼에 복사를 한다. 그리고 세 개의 쓰레드 중에 T1 쓰레드가 별도의 I/O 버퍼에 로그 버퍼

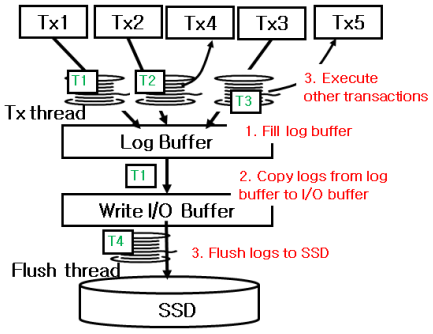

그림 3. 최적화된 로그 처리 기법

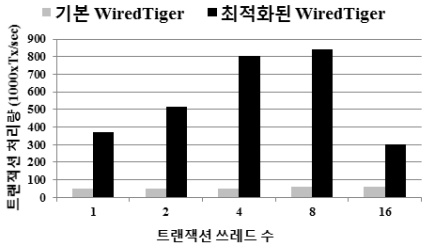

그림 4. 최적화된 로그 처리 기법의 성능 평가

에 저장되어 있는 로그를 복사한다. I/O 버퍼로 복사하는 과정이 sync 락에 의해 여전히 보호되지만 기존의 방법처럼 SSD에 써지는 동안에 보호되는 것보다는 보호 구간이 (critical section) 짧아진다. 이 때 Tx1, Tx2, Tx3 트랜잭션의 상태는 COMMIT으로 변경되지 않는다. 그 후에 T1, T2, T3 쓰레드들은 응용의 다른 트랜잭션 요청을 처리한다. 그림의 예에서는 T2 쓰레드는 Tx4 트랜잭션, T3 쓰레드는 Tx5 트랜잭션을 수행한다. 이와 별도로 Flush 쓰레드인 T4는 I/O 버퍼로 복사된 로그를 저장 장치에 쓴다. 쓰기 연산이 완료된 후에 T4는 다 써진 로그와 관련된 트랜잭션들의 상태를 COMMIT으로 변경한다. 즉, T4가 Tx1, Tx2, Tx3의 로그를 저장 장치에 쓴 후에 해당 트랜잭션의 상태를 COMMIT으로 변경한다. 그 후에 T1, T2, T3는 Tx1, Tx2, Tx3 트랜잭션들을 각각 완료 처리 한다.

이러한 최적화 기법은 sync 락으로 보호되는 구간이 짧아지기 때문에 락 경쟁을 줄이는 효과를 가진다. 그리고 로그가 저장 장치에 써지는 동안에 다른 트랜잭션

들을 수행할 수 있어서 성능을 개선시킬 수 있다. 하지만 COMMIT되지 않은 트랜잭션의 수가 늘어남에 따라 같은 키를 가지는 여러 버전의 value들을 (다 버전 병행 제어) 유지해야 할 확률이 높아지고 이것은 메모리 사용량을 높이는 결과를 가져올 수 있다. 극단적인 경우로 HDD를 저장 장치로 사용하게 되면 COMMIT되지 않은 트랜잭션의 수가 SSD의 경우보다 더 많아지게 이에 따라 다 버전 병행 제어로 인한 성능 병목이 생기게 된다. 실제로 로그 처리 최적화 기법을 적용한 WiredTiger와 기본 WiredTiger를 HDD 탑재 시스템에서 비교하였으며, 16개의 트랜잭션 쓰레드를 실행하였을 때 성능 개선 이 거의 없음을 확인하였다.

로그 처리 최적화 기법의 효과를 알아보기 위해 3절의 실험 환경에서 기본 WiredTiger와 최적화된 WiredTiger에 수행하였다. [그림 4]는 최적화된 로그 처리의 성능 결과를 보여준다. 성능 평가를 위해 YCSB의 update heavy 워크로드를 사용하였다. 4개의 트랜잭션 쓰레드가 트랜잭션을 처리하는 경우에 기본 WiredTiger는 초당 5만개, 그리고 최적화된 WiredTiger는 초당 80만개 정도 트랜잭션을 처리하여 약 15.9배 정도의 성능 향상 효과가 있음을 확인하였다. 이것은 sync 락 경쟁이 줄어드는 효과와 함께 트랜잭션의 로그가 장치에 써지는 동안에 해당 쓰레드가 다른 트랜잭션들을 수행하기 때문에 시스템 전체 측면에서 트랜잭션 처리량을 높이기 때문이다. 하지만 16개의 트랜잭션 쓰레드의 경우에는 성능 향상 효과가 5배 정도로 줄어드는데 그것은 데이터베이스 쓰레드의 수가 커지면서 sync lock 경쟁으로 인한 오버헤드가 커지기 때문이다.

표 4. CPU 사용률 비교 (쓰레드 수:8)

	기본 WiredTiger	최적화된 WiredTiger
WiredTiger	3.8%	51.7%
Linux Kernel	0.7%	1.4%
합계	4.5%	53.1%

[표 4]는 트랜잭션 쓰레드의 수가 8일 때, 최적화된 로그 처리를 적용한 경우와 그렇지 않은 경우의 CPU 사용률을 보여준다. 로그를 flush하는 동안에 트랜잭션 쓰레드가 다른 트랜잭션들을 수행할 수 있기 때문에 CPU 사용률을 높일 수 있다.

VI. 결론

본 논문에서는 여러 IT 분야에서 많이 활용되고 있는 고성능 NVMe 플래시 SSD를 이용하여 NoSQL 데이터베이스 시스템의 성능을 평가하고 분석하였다. 삼성전자의 최신 NVMe 플래시 SSD와 현재 가장 우수하다고 알려져 있는 로그 처리 알고리즘이 구현된 WiredTiger를 사용하여 성능을 분석하였다. 실험 결과는 고성능 NVMe 플래시 SSD를 탑재한 서버 환경에서는 NoSQL 데이터베이스 시스템의 로그 레코드 처리 부분이 여전히 가장 큰 오버헤드임을 보여준다. 이를 해결하기 위해 로그가 써지는 동안에 다른 트랜잭션을 수행하는 최적화 기법을 제안하고 구현하였으며 이 기법이 최대 15배 정도 성능을 향상시킬 수 있음을 보였다. 향후 연구에는 로그 처리 부분에서 락 경쟁을 완전히 제거할 수 있는 기법을 제안하고자 한다.

참 고 문 헌

[1] IBM system storage ds8000 easy tier. http://www.ibm.com/systems/storage/flash/720-820.

[2] Netapp. http://www.netapp.com/us/products/storage-systems/flash-accel.

[3] Flasharray, meet the new 3rd-generation flasharray. http://www.purestorage.com/flash-array.

[4] A. Huffman, “NVM Express Overview & Ecosystem Update,” In Proceedings of Flash Memory Summit 2013.

[5] MongoDB Inc., http://www.mongodb.com

[6] MongoDB Inc., http://www.wiredtiger.com

[7] Ryan Johnson, Ippokratis Pandis, Radu Stoica, Manos Athanassoulis, and Anastasia Ailamaki., “Aether: a scalable approach to logging,” Proc. VLDB Endow, Vol.3, Issue.1-2, 2010(9).

[8] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, and L. Vivier, “The new ext4 filesystem: Current status and future plans,” In Proceedings of the Linux Symposium, 2007.

[9] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears, “Benchmarking cloud serving systems with YCSB,” In Proceedings of the 1st ACM symposium on Cloud computing (SoCC10).

[10] Hyojun Kim, Sangeetha Seshadri, Clement L. Dickey, and Lawrence Chiu, “Evaluating phase change memory for enterprise storage systems: a study of caching and tiering approaches,” In Proceedings of the 12th USENIX conference on File and Storage Technologies (FAST'14).

[11] Sang-Won Lee, Bongki Moon, Chanik Park, Jae-Myung Kim, and Sang-Woo Kim, “A case for flash memory ssd in enterprise database applications,” In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08).

[12] Sangwhan Moon, Jaehwan Lee, Xiling Sun, and Yang-Suk Kee, “Optimizing the Hadoop MapReduce Framework with high-performance storage devices,” Journal of Supercomputing, Vol.71, No.9, 2015(9).

[13] Yongseok Son, Hara Kang, Hyuck Han, and Heon Young Yeom, “An empirical evaluation and analysis of the performance of NVM express solid state drive,” Cluster Computing, Vol.19, No.3, 2016(9).

[14] Jeong-Uk Kang, Jeeseok Hyun, Hyunjoo Maeng, and Sangyeun Cho, “The multi-streamed solid-state drive,” In Proceedings of the 6th USENIX conference on Hot Topics in Storage and File Systems (HotStorage'14).

[15] Lanyue Lu, Thanumalayan Sankaranarayana Pillai, Hariharan Gopalakrishnan, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, “WiscKey: Separating Keys from Values in SSD-Conscious Storage,” ACM Transations on Storage, Vol.13, No.1, Article.5.

[16] Yanqin Jin, Hung-Wei Tseng, Yannis Papakonstantinou, and Steven Swanson, "KAML: A Flexible, High-Performance Key-Value SSD," In Proceedings of the 23rd IEEE Symposium on High Performance Computer Architecture (HPCA2017).

저 자 소 개

한 혁(Hyuck Han) 정회원

- 2003년 8월 : 서울대학교 컴퓨터공학부(공학사)
- 2006년 2월 : 서울대학교 컴퓨터공학부(공학석사)
- 2011년 2월 : 서울대학교 컴퓨터공학부(공학박사)
- 2011년 3월 ~ 2012년 8월 : 서울대학교 컴퓨터공학부 박사후 연구원
- 2012년 9월 ~ 2014년 2월 : 삼성전자 메모리 사업부 책임연구원
- 2014년 3월 ~ 현재 : 동덕여자대학교 컴퓨터학과 조교수

<관심분야> : 데이터베이스 시스템, 병렬 프로그래밍, 분산 시스템