NVDIMM 기반 PostgreSQL WAL로그 성능평가

심주보○, 이상원
성균관대학교
{joobo95, swlee}@skku.edu

Performance Analysis of NVDIMM-Based PostgreSQL WAL Log

Joobo Shim○, Sang-Won Lee
Sungkyunkwan University

요 약

비휘발성 메모리인 NVDIMM은 데이터베이스의 durability를 위해 DBMS에서 필수적인 로깅 시스템에 변화를 주어 성능 저하를 야기하는 latency문제를 해결할 수 있다. 본 논문에서는 상용 오픈소스 DBMS인 PostgreSQL의 로깅 시스템에 NVDIMM의 특성이 어떤 영향를 줄 수 있는지 알아보고 적용한 뒤, 기존 시스템과 변경된 시스템의 성능평가를 시행했다. 실험은 TPC-C벤치마킹을 통해 이루어졌으며, 기존의 시스템과 변경된 시스템의 트랜잭션 수행 능력, CPU 사용량의 측면에서 평가했다. NVDIMM의 특성을 적용했을 때 트랜잭션 수행 능력과 CPU 사용비율이 높아졌고, 이는 컴퓨터 장치의 효과적인 사용과 DBMS시스템의 성능 향상을 의미한다.

1. 서 론

DBMS는 안전한 트랜잭션 수행을 보장하기 위해 ACID(Atomicity, Consistency, Isolation, Durability)의 성질을 지킨다. 이 중 Durability는 작업 수행 도중 전력공급의 문제나 갑작스런 오류(system crash)에 의해 발생하는 파일손상에도 commit된 트랜잭션은 온전히 보존되거나 복구 되어야 함을 의미한다. DBMS는 commit전 변경된 내용에 대해 로그를 먼저 저장하는 WAL 로깅 시스템을 적용해 durability를 지원한다.

휘발성 DRAM을 주 기억장치로, 비휘발성 SSD나 HDD를 보조 기억장치로 사용하는 메모리 계층 구조에서 로그파일은 주 기억장치에서 작성되고 보조 기억장치에 저장된다. 주 기억장치에 write된 파일은 system crash에 안전하지 못하기 때문에 DBMS에서 fsync를 통해 보조기억장치에 저장 한다. 로그 파일을 저장하는 과정은 CPU의 처리속도 보다 느리며, 로그 파일이 완전히 저장되기 전에 다음 작업을 수행할 수 없으므로 DBMS는 요청한 작업이 완료될 때 까지 기다리게 된다. 이렇게 발생한 대기 지연(latency)은 그 빈도가 높지 않더라도 한번 수행에 걸리는 시간이 비교적 크기 때문에 DBMS 성능을 상당히 저하시킨다.

다중 코어 중앙처리장치, 대용량 주 기억장치, 높은 성능의 보조기억장치 등 점점 성능이 좋아지는 컴퓨터 장치들로 인해 최근 데이터 관리 시스템의 정보 처리 속도는 계속해서 발전하고 있다. 그러나 언제 일어날지 모르는 system crash를 대비하기 위한 작업 때문에 컴퓨터 장치들의 최대 성능을 누리지 못하고 있다. 이를 해결하기 위한 노력으로 PRAM, BBU(배터리 백업)DRAM 과 SSD 하이브리드 드라이브와 같은 NVRAM(Non-volatile DIMM)이 등장하고 있으며, 관련된 다양한 연구 활동이 이루어지고 있다. NVRAM중 하나인 NVDIMM의 비휘발성은 기존 휘발성 주 기억장치를 사용하는 DBMS에 적용된 로그 시스템에 큰 변화를 줄 수 있다[1, 2].

따라서 본 논문에서는 상용 오픈소스 DBMS인 PostgreSQL의 로깅 시스템에 NVDIMM의 특성을 적용하여 줄일 수 있는 작업을 찾아보고, 적용해본다. 변경된 시스템이 OLTP benchmark 워크로드를 얼마나 잘 수행하는지 보기 위해 TPC-C 벤치마킹을 통해 성능을 평가했다. 그 결과를 통해 NVDIMM을 적용한 DBMS가 가질 수 있는 성능향상에 대해 논한다.

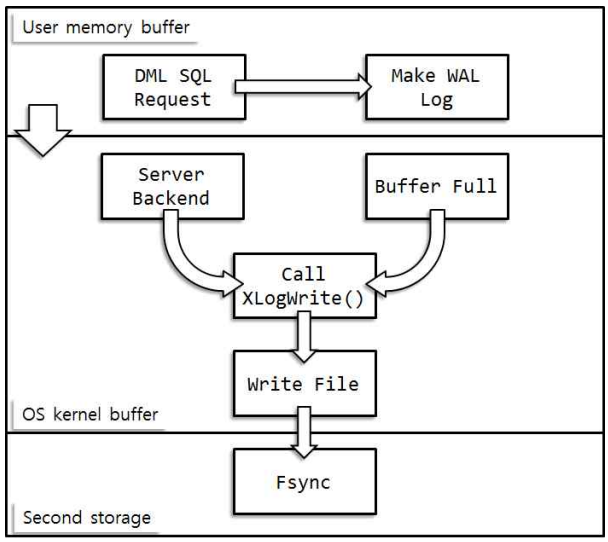
본 논문의 구성은 다음과 같다. 2장에서 PostgreSQL의 WAL 로그 시스템을 분석하고, NVDIMM의 특성이 적용될 때 불필요한 작업을 살펴본다. 3장에서 기존 시스템, 변경된 시스템의 성능평가 환경과 결과를 보인다. 4장에서 결론을 맺고 향후 연구를 제시하며 논문을 마친다.

2. PostgreSQL의 WAL 로그 시스템과 NVDIMM

PostgreSQL의 초기 설계 구조는 매우 간단한 로깅 시스템을 제안하지만[3], 현재는 durability의 문제로 인해 그 구조를 사용하지 않는다. PostgreSQL에서 이루어지는 WAL 로깅의 과정은 [그림 1]과 같다. 클라이언트에서 DELETE, INSERT, UPDATE등의 DML(Data Manipulate Language) 쿼리를 요청하면 서버가 이를 수행하고 WAL 로그를 작성한다. 작성된 로그는 서버가 요청을 처리하는 도중 system crash가 발생하면 다시 복구할 수 없으므로 DBMS에서 설정한 Timout이 발생하거나 일정 횟수의 트랜잭션이 수행되었을 때 서버의 Backend에서 로그 저장을 시도하고, 지정된 버퍼사이즈에 더 이상 용량이 없을 때 로그 저장을 시도한다. 로그 저장은 XLogWrite 함수가 수행하며 해당 함수 안에서 파일을 열어 작성된 로그를 저장하고 fsync를 호출한다. [4, 5]

SQL문에 의해 WAL로그가 작성될 때 로그 내용은 프로그램 내부 변수로써 User memory buffer에 존재하게 되며, WAL로그를 파일에 저장할 때 OS kernel buffer에

쓰이게 되고, fsync가 호출될 때 보조 기억장치에 완전히 저장된다. 이 때 사용되는 write 함수와 fsync함수는 공용 데이터를 접근하는 과정이므로 synchronous하게 이뤄져야 하며, 해당 작업은 exclusive lock을 요구한다. 기억장치의 데이터 처리 속도가 CPU보다 현저히 느리기 때문에 lock을 가지지 못한 트랜잭션은 모두 대기하고 있어야하며, latency 때문에 전체 데이터베이스 시스템이 병목현상을 겪게 된다.



[그림 1] PostgreSQL WAL 로그 시스템 구조

NVDIMM은 system crash에도 데이터가 손상되지 않는 특성을 가지지만, 트랜잭션 롤백이나 recovery에 사용될 로그는 여전히 필요하다. WAL 로깅에 있어서 주 기억장치를 비휘발성 NVDIMM으로 대체하게 된다면, User memory buffer와 OS kernel buffer가 system crash에도 정보를 잃지 않게 되며 write와 fsync과정 없이도 로그 파일의 durability를 만족할 수 있다. [1]

NVDIMM의 용량이 가득 찰 때마다 현재 사용하고 있지 않은 부분에 저장된 로그 파일을 독립적으로 보조 기억장치로 옮기면, write 와 fsync과정 없이 [그림 1]의 시스템과 동일한 작업을 수행할 수 있다. 결과적으로 한 트랜잭션이 가지게 되는 latency가 줄어들게 되고, 다른 트랜잭션을 기다리는데 소비하는 시간이 짧아진다. 즉, 데이터베이스 시스템에서 WAL로깅에 의한 병목현상이 완화되며 CPU를 최대한으로 활용하여 기존보다 빠른 트랜잭션 수행 성능을 기대할 수 있다.

3. 성능 평가 및 분석

본 실험에서 DBMS는 PostgreSQL 9.4.5 를 사용했고, 벤치마크 툴은 BenchmarkSQL 4.1.0 를 사용했다. 자세한 실험 환경은 [표 1]과 같다.

TPC-C 벤치마킹 설정에서 warehaouse의 값은 100(약 11GB)이며, 클라이언트 수는 100개이다. 한 클라이언트에서 1000개의 트랜잭션을 수행하며, 총 100000개의 트랜잭션을 처리한다. 로그 파일은 DML 쿼리에서 생성되므로 TPC-C workload를 write-intensive (Delivery 4%, OrderStatus 4% ,Payment 43%, StockLevel 4%, NewOrder 45%) 하게 설정했다.

운영체제	Linux (커널 3.13.0-74)
CPU	Intel® CoreTM i5-2500k CPU @ 3.30GHz (4 CPUs)
메모리(RAM)	12.00GB
저장장치	Samsung 840 PRO SSD 256GB
DBMS	PostgreSQL 9.4.5
벤치마크 툴	BenchmarkSQL 4.1.0

[표 1] 실험 환경

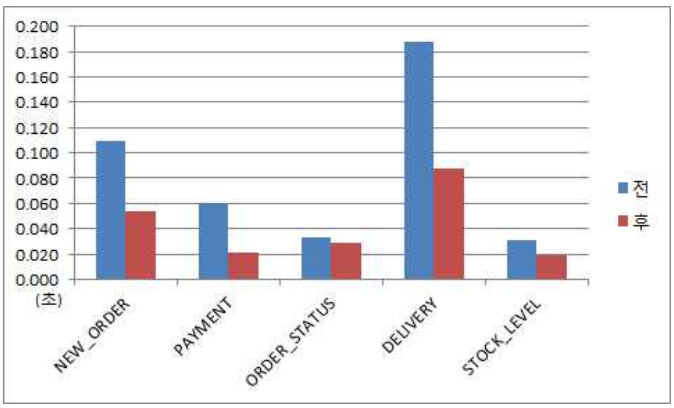
Original 데이터베이스 설정은 소스코드 원본을 사용했으며, Modified는 NVDIMM의 특성을 적용한 PostgreSQL를 사용했다. 이는 PostgreSQL의 소스코드 원본 파일 중 XLogWrite의 write, issue_xlog_fsync 함수를 주석처리 하고, PostgreSQL 서버 설정에서 fsync를 사용하지 않도록 한 것이다.

각 실험에서 트랜잭션별 latency와 tmpC, tmpTOTAL, CPUIdleMAX, CPUIdleAVG를 측정했으며 5회씩 실행 후 평균값을 기준으로 평가했다.

트랜잭션별 latency는 5가지 종류의 트랜잭션에서 1개의 트랜잭션이 평균적으로 가지는 latency를 측정한 것이다. tmpC는 신규주문과 처리과정 전체를 하나의 작업으로 본 분당 작업처리량을 말하며, tmpTOTAL은 실행 중 분당 트랜잭션 처리 개수를 의미한다. CPUIdleMAX와 CPUIdleAVG는 각각 실행 중 CPU idle상태의 최대 비율, 평균 비율을 보여준다. 전후비율은 Modified/Original 값이다.

	Original	Modified	전후비율
NEW_ORDER	0.109 sec	0.053 sec	49%
PAYMENT	0.060 sec	0.022 sec	36%
ORDER_STATUS	0.033 sec	0.028 sec	87%
DELIVERY	0.188 sec	0.087 sec	46%
STOCK_LEVEL	0.031 sec	0.019 sec	60%

[표 2] 트랜잭션 종류별 latency 결과 표

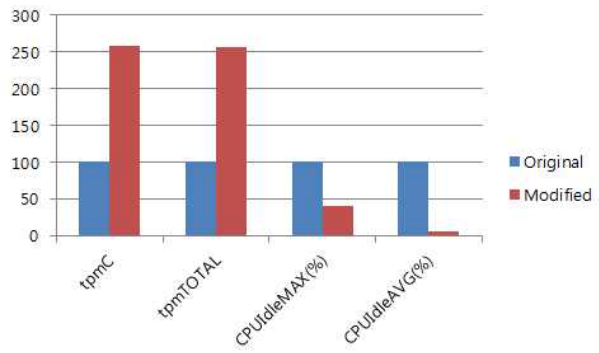


[그림 2] 트랜잭션 종류별 latency 결과 그래프

트랜잭션에 따른 latency의 결과는 [표 2], [그림 2]와 같다. Original 대비 Modified의 비율이 최대 약 1/3까지 감소했다. 전체 트랜잭션에 많은 비율을 차지하는 Payment와 New_order 항목에서 각각 49%, 36%로 줄어들어 전체 시스템 수행시간에 영향을 줄 것으로 예상했고, 결과도 그러했다. [표 3], [그림 3]은 성능 평가 결과를 나타낸다.

	Original	Modified	전후비율
tpmC	9578.89	24645.87	257%
tpmTOTAL	21292.98	54675.28	257%
CPUIdleMAX	51.18%	20.46%	40%
CPUIdleAVG	21.18%	1.08%	5%

[표 3] 성능 평가 결과



[그림 3] Original을 100으로 본 Modified의 비율

TPC-C 벤치마크 수행 결과 Modified 시스템은 Original 시스템에 비해 성능의 향상을 보였다. 트랜잭션 처리량이 257%로 증가하였으며, 사용되지 않는 CPU의 비율이 최대치는 40%로, 평균치는 5%로 감소하였다.

4. 결론 및 향후 연구

본 논문에서는 PostgreSQL의 WAL 로그 시스템을 분석하고, NVDIMM의 특성이 적용될 때 불필요한 작업을 살펴보았다. PostgreSQL에 NVDIMM의 특성이 적용된다면 user memory buffer의 durability를 보장받을 수 있고, write와 fsync의 과정을 생략할 수 있다. 이에 따라 트랜잭션이 lock을 잡기 위해 대기하는 시간과 lock을 잡고 명령을 처리하는 수행시간의 latency를 줄여준다.

성능 평가 결과, 변경된 시스템의 성능이 OLTP benchmark workload 상에서 향상됐다. DBMS의 트랜잭션 처리량을 2배 이상 향상시킬 수 있었고, I/O latency에 의해 사용되지 않는 CPU의 평균 수치를 5%까지 낮추어 컴퓨터의 계산 성능을 효과적으로 사용할 수 있음을 보였다.

향후 DBMS의 로깅시스템에서, NVDIMM뿐만 아니라 새로운 디바이스의 특성을 적용함으로써 어떤 작업을 삭제, 간소화 할 수 있는지 알아보고, 성능향상을 위한 연구해보고자 한다. 또한 PostgreSQL의 초기 설계 단계에 적용된 WAL로깅 구조가 현재 사용되지 못하는 문제점을 해결하는 방법을 연구해볼 계획이다.

사사

이 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (R0126-16-1108, 비휘발성 메모리 기반 개방형 고성능 DBMS 개발)

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 SW중심대학지원사업의 연구결과로 수행되었음 (R2215-16-1005)

5. 참고문헌

[1] 이상원, “NVRAM과 데이터베이스 지속성”, 정보과학회지, 81p, 2015.2

[2] 홍대용, 오기환, 강운학, 이상원, “비휘발성 캐시를 사용하는 플래시 메모리 SSD의 데이터베이스 로깅 성능 분석, 정보과학회논문지, 제42권, 1호, 107p, 2015.1

[3] M. Stonebraker, “The design of the Postgres storage system”, In Proc. of Intl. Conf. on VLDB, Sept. 1987.

[4] PostgreSQL 9.4.5 source, http://www.postgresql.org/ftp/source/v9.4.5/

[5] Michael Stonebraker, Lawrence a. Rowe, Michael Hirohama, “The Implementation of POSTGRES”, IEEE Transactions on Knowledge and Data Engineering, Volume 2, Issue 1, 125-142p, Mar 1990