RDBMS의 HDD와 SSD상에서 인터럽트 오버헤드 분석

이영석○, 이상원
성균관대학교
bdg97119@gmail.com, swlee@skku.edu

Analysis of RDBMS Interrupt Overhead on HDD and SSD

Yeong-seok Lee○ Sang-won Lee
Sungkyunkwan University

요 약

보조기억장치로 SSD가 HDD를 대체하고 있으며, SSD와 관련된 연구가 활발하게 진행되고 있다. 프로그램을 실행하기 위해서는 항상 운영체제 위에서 실행 될 수 밖에 없으며, 프로그램이 하드웨어와 커뮤니케이션을 하기 위해서는 커널에게 시스템 콜을 통해 요청을 할 수 있다. 본 논문은 MySQL의 벤치마크 프로그램 중 하나인 Linkbench를 이용하여 SSD와 HDD상에서의 인터럽트 오버헤드를 측정하였다. 인터럽트 오버헤드를 측정하는 도구로 strace와 /proc/stat 정보를 이용하여 분석하였다.

1. 서 론

보조기억장치로서 HDD를 대체할 수 있는 SSD가 개발됨에 따라 SSD에 대한 활발한 연구가 진행되고 있다. I/O 작업이 많이 일어나는 대표적인 프로그램인 RDBMS는 SSD가 개발됨에 따라 높은 성능을 낼 수 있게 되었다. 하지만 현재의 RDBMS 아키텍쳐는 1970년대 후반의 컴퓨터 기술에 최적화 되어있으며, 현재까지도 그 변화가 거의 없다.〔1〕

따라서 우리는 RDBMS가 HDD와 SSD상에서 운영체제와 어떠한 방식으로 요청을 처리하는지 분석할 필요가 있다. 우리가 어떤 프로그램을 실행하기 위해서는 항상 운영체제 위에서 실행되는데, 운영체제는 하드웨어와의 커뮤니케이션을 위해서 인터럽트를 사용한다.

본 논문은 I/O 명령이 주를 이루는 대표적인 어플리케이션 중 하나인 RDBMS의 벤치마크인 Linkbench를 통해 HDD와 SSD상에서의 인터럽트 오버헤드를 분석하였다.

2. 관련 연구

인터럽트는 크게 비동기적 인터럽트와 동기적 인터럽트로 나눌 수 있는데, 비동기적 인터럽트는 CPU상에서 어떠한 프로그램이 실행되고 있을 동안 타이머나, I/O 디바이스(키보드 입력 등)에서 처리를 요청하는 인터럽트로서 현재 CPU에서 실행되고 있는 프로그램과 상관없이 언제든지 일어날 수 있는 인터럽트이다. 반면에 동기적 인터럽트는 현재 프로그램에서 CPU에서 처리할 수 없는 사항을 프로그래머가 처리를 요청했을 경우 발생하는 인터럽트인데, 운영체제에서는 이것을 소프트웨어 인터럽트라고 정의하고, 시스템 콜(System Call)을 통해서 처리를 한다.〔2〕

시스템 콜은 운영체제 커널이 사용자 공간(userspace)의 프로그램이 직접 처리할 수 없는 사항을 커널 모드(kernel mode)에서 처리할 수 있도록 제공하는 인터페이스이다. 시스템 콜을 호출하게 되면 사용자 모드에서 커널 모드로 바뀌게 되고, 시스템 콜을 처리 후 다시 사용자 모드로 돌아가게 된다.

3. 실 험

실험은 MySQL와 Linkbench를 이용해서 HDD와 SSD상에서 동일한 조건으로 실험을 하였다. 시스템 콜과 인터럽트의 오버헤드를 측정하기 위해서 Strace 프로그램과 MySQL의 /proc/stat 정보를 이용하였다.

OS	Linux kernel 3.18
RDBMS	MySQL 5.6
InnoDB Buffer pool	128MB
Benchmark	Linkbench
Threads	16
Requests/thread	10000

표 1 실험 환경

Query Type	Query 개수
SELECT	110492
INSERT	18497
UPDATE	24607
DELETE	6406
합계	160000

표 2 Query Type에 따른 각 Query 개수

4. 실험 결과

(단위 : 초)

	HDD	SSD
user	55.3625	57.045
system	18.865	31.3175
idle	384.4175	41.795
iowait	2881.43	65.745
irq	0.0025	0
softirq	0.6075	0.6575
합계	3340.685	196.56

표 3 /proc/stat 결과

전체 16만개의 요청(Request)을 처리하는데 SSD는 201초, SSD는 3345초가 소요되었다. 표3는 각 디바이스에서 벤치마크를 실행하는 동안의 CPU가 어떤 동작을 했는지에 대해 평균 값을 보여준다. 멀티코어이기 때문에 각 CPU에서 한 동작시간을 평균값으로 나타내었다. 사용자 공간에서 동작한 시간이 실제로 MySQL이 벤치마크를 처리하는 부분이다. HDD와 SSD가 동일한 요청을 처리하므로 실제로 나타난 사용자 공간에서 처리 시간도 HDD와 SSD가 거의 동일한 것을 알 수 있다. 하지만 주목해야 할 부분으로 시스템 공간(Kernel space)에서 처리한 시간이 SSD가 약 1.7배 많은 것을 알 수 있다. 사용자 공간에서 실행된 프로그램이 시스템 공간에 접근을 하려면 시스템 콜을 이용해야 한다. 이는, SSD가 HDD에 비해 동일한 처리를 하는데 있어서 시스템 콜을 더 많이 요청했다는 것을 뜻한다.

% time	seconds	usec/call	calls	syscall name
48.59	1083.4	140	7722446	futex
38.2	851.9	9087	93756	poll
5.35	119.4	584	204315	pread
2.8	62.5	590	105919	pwrite
2.44	54.5	242115	225	select
1.86	41.5	1275	32564	fsync
0.4	8.9	19	477983	sched_yield
0.18	3.9	16	245687	sendto
0.17	3.7	8	487401	recvfrom

표 4 SSD에서 strace 결과

표4과 표5는 Strace를 이용하여 벤치마크 실행동안의 MySQL의 시스템 콜 호출횟수와, 차지한 시간을 나타낸 표이다. 동일한 요청을 처리하는 동안 대부분의 시스템 콜 횟수가 비슷하게 나타났는데, futex와 시스템 콜은 SSD에서 각각 3.2배 더 많이 호출되었는 것을 알 수 있다. Futex 시스템 콜은 Fast Userspace Mutex의 약자로서 사용자 공간에서 뮤텍스를 처리하는 함수인데, 경합(Contention)이 발생하지 않는다면 사용자 공간에서 바로 처리가 가능하며, 경합이 발생할 경우 futex 시스템 콜을 호출하게 된다.[3]

% time	seconds	usec/call	calls	syscall name
40.13	566.8	217	2606076	futex
34.67	489.7	4825	101491	poll
17.3	244.3	1083	225532	pread
4.22	59.7	528	112967	pwrite
1.88	26.6	731	36398	select
1.67	23.5	9237	2548	fsync
0.05	0.6	4	177443	sched_yield
0.04	0.6	3	246013	sendto
0.03	0.5	1	494190	recvfrom

표 5 HDD에서 strace 결과

이는 SSD에서 각 쓰레드가 공유 자원(Shared Resource)에 대해서 경합이 많이 발생함을 뜻한다. SSD의 특성상 HDD보다 읽기 속도가 수십 배 빠르기 때문에 CPU는 그만큼 처리할 수 있는 자원의 양이 많아지고 각 코어에서 동시에 자원에 대한 처리를 빠르게 할 수 있지만 그만큼 경합이 많이 발생할 수 있다는 것을 뜻하며, 표4의 futex 시스템 콜 횟수는 이를 뒷받침 해준다.

또한 여기서 각 시스템 콜에서 걸린 시간이 실제로 벤치마크가 실행된 시간보다 많이 측정 되었는데, 이는 strace에서 CPU에서 직접 실행된 시간을 측정한 것이 아니라, 쓰레드가 대기상태(Blocked)일 때도 포함해서 측정하였기 때문에 실제 실행시간보다 많게 측정되었다.

	Context Switches	switches/secs
SSD	40491085	201448
HDD	24000189	7174

표 6 컨텍스트 스위칭

표6은 컨텍스트 스위칭(Context Swtich)의 횟수를 나타낸 것이다. 초당 컨텍스트 스위칭 횟수가 SSD가 HDD에 비해서 약 30배 정도 많은 것을 알 수 있다. 이것은 SSD가 동일한 요청을 처리하는 데에 있어서 I/O 속도가 빠르기 때문에 훨씬 빠른 속도로 처리가 가능하지만 그만큼 컨텍스트 스위칭의 오버헤드가 더 많아짐을 뜻한다.

5. 결론 및 향후 연구

본 논문은 RDBMS의 벤치마크를 통해 인터럽트의 오버헤드와 컨텍스트 스위칭에 대해서 SSD와 HDD상에서 분석을 진행하였다. 동일한 벤치마크를 처리할 동안

SSD상에서 시스템 공간 상에서 HDD보다 약 1.7배의 시간을 소모하는 것을 확인 할 수 있었다. Futex 시스템 콜에서 SSD가 HDD에 비해 3배 이상 호출이 되었다. 또한, 컨텍스트 스위칭 또한 SSD가 시간당 30배 많이 일어나는 것도 확인 할 수 있었다.

향후에 SSD상에서 Futex 시스템 콜이 왜 많이 발생하는지와 정확한 시간 오버헤드 분석이 필요할 것이다. 또한, 컨텍스트 스위칭의 횟수를 줄일 수 있는 방안에 대한 연구를 진행하여야 한다. 또한, Query Type의 비율에 따라 SSD와 HDD의 System Call 비율을 조사하여 각 System Call 오버헤드 변화를 분석하는 연구를 진행하여야 한다.

사 사

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

이 논문은 2014년도 정부(산업통상자원부)의 재원으로 산업기술 기술혁신사업의 지원을 받아 수행된 연구임 (10049445, 모바일 저장장치용 UFS 2.0 제어기 SoC 및 임베디드 SW 개발)

참 고 문 헌

[1] HARIZOPOULOS, Stavros, et al. OLTP through the looking glass, and what we found there. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM, 2008. p. 981-992.

[2] BOVET, Daniel P.; CESATI, Marco. Understanding the Linux kernel. " O'Reilly Media, Inc.", 2005.

[3] Linux Programmer's Manual, Futex, http://man7.org/linux/man-pages/man2/futex.2.html