플래시메모리SSD 상에서 IOSTAT 지표의 한계점

이재훈°, 이상원
성균관대학교
{ljhh0611, swlee} @ skku.edu

Limitations of IOSTAT Metrics on Flash Memory SSDs

Jae-Hun Lee°, Sang-Won Lee
Sungkyunkwan University

요 약

플래시메모리 저장장치 기술의 급격한 발달로 인해 플래시메모리 SSD가 주류 저장장치로 자리 잡는 all-flash 시대가 도래하고 있다. 한편, 플래시메모리 SSD는 그 내부의 풍부한 병렬성 등 기존 하드디스크와 다른 여러 가지 특성을 가진다. 리눅스 환경에서 IO 장치의 사용률 지표를 보여주는 대표적인 도구인 IOSTAT은 주로 하드디스크의 특성을 반영해서 개발되었다. 따라서, 이 도구를 플래시메모리SSD상에서 수행할 때, 플래시메모리SSD의 사용률을 정확하게 반영하지 못하게 된다. 본 논문에서는 플래시메모리SSD상에서 보여지는 IOSTAT 지표의 한계점을 설명하고 그 원인을 규명한다.

1. 서 론

컴퓨터과학계에서는 연구와 실험 중 많은 부분이 컴퓨팅 성능을 향상시키는 목적으로 수행된다. 이 때, 스토리지 또는 디바이스 관련 연구는 디바이스의 성능을 모니터링하는 방법이 주요 쟁점으로 작용한다. 정확하게 성능을 측정해야 실험의 결과를 신뢰할 수 있기 때문이다.

과거엔 대부분의 컴퓨터 저장장치로 HDD를 사용하여 연구를 진행해왔다. 하지만 현재는 SSD와 차세대 플래시 메모리를 새로운 저장장치로 사용하고, 연구 또한 HDD를 대체한 SSD에 관련한 많은 연구가 수행되고 있다[1][2]. 그러나 실제로 HDD와 SDD는 완전히 다른 방법으로 동작하기 때문에 공통점보다 차이점이 더 많은 디바이스이다. 그렇기에 연구부분에서도 HDD를 사용하여 했을 때 문제가 되지 않았던 부분에서 SSD로 바뀌면서 문제가 될 여지가 충분히 있다.

이에 따라 본 논문에서는 iostat의 디바이스 utilization이 SSD를 사용한 실험 측정에 문제가 되는 이유를 분석하고 실험을 통해 실제로 문제가 발생함을 확인한다.

본 논문은 2장에서 디바이스의 모니터링 방법 중 널리 쓰이는 iostat에 대해 살펴보고, 3장에서 SSD와 HDD의 차이점에 대해 간단히 알아본 뒤, 4장에서는 SSD를 이용하여 실험을 했을 때 iostat 사용에 어떤 문제가 있는지 분석한다. 5장에서는 4장에서 도출된 문제점이 실제로 발생하는지 확인하는 실험을 진행한다. 마지막으로 6장에서 실험의 결론과 어떻게 문제를 해소할 수 있을지에 대한 향후 연구를 제시하며 논문을 마친다.

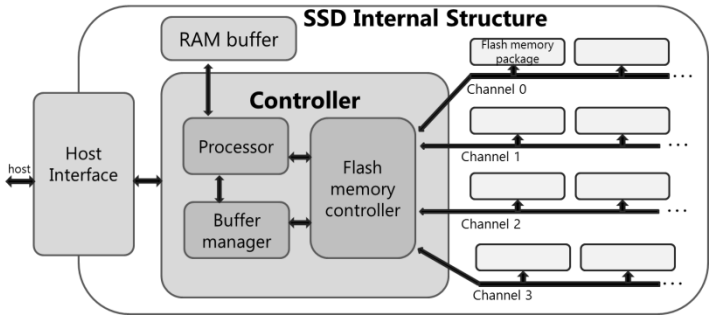
2. 디바이스 모니터링 방법

디바이스를 모니터링 하는 방법으로 리눅스에서 제공하는 iostat을 이용하여 실험을 진행하는 경우가 많다. iostat은 일정한 주기마다 실험을 진행 중인 컴퓨터의 CPU 사용률과 컴퓨터와 연결된 디바이스의 IO 상태정보를 보여주기 때문이다. CPU의 사용률은 유저 프로그램이 사용한 시간, 시스템이 사용한 시간, IO를 기다리는 시간, 아무것도 하지 않은 시간 등으로 나누어 보여준다. 또한 디바이스 IO에 대해서는 CPU보다 다양한 정보를 보여주는데, 초당 읽기/쓰기 각각의 트랜잭션 횟수나 양, 초당 디바이스 큐의 크기, IO요청 후 요청에 대한 결과를 반환한 시간, 그리고 디바이스의 사용률(util)에 대한 정보를 제공한다. 연구자들은 이 정보 중, CPU의 util과 디바이스의 util정보를 이용해 실험 상황에서 CPU util이 100%면 CPU를 완전히 사용하고 있다고 판단하고, 디바이스의 util이 100%면 디바이스의 성능을 완전히 사용한다고 판단한다. 그리고 이에 따라 실험을 수정하거나 실험의 결과로 사용한다.

3. HDD와 다른 SSD의 IO특징

플래시 메모리 SSD는 HDD와 많은 부분에서 다르다. 그 중 IO의 관점에서 살펴보면, HDD는 기계의 물리적 움직임을 이용해 데이터가 쓰여진 곳, 또는 데이터를 쓸 곳으로 이동해 한번에 단 하나의 읽거나 쓰는 일을 수행할 수 있다. 반면, SSD는 [그림1]의 내부 구조를 기본으로 구성되어있다[3]. 기본적으로 메모리 패키지로 이루어져 있기 때문에 물리적인 움직임 없이 전기적 신호를 이용해 IO를 수행한다. 또한 이 다수의 메모리 패키지는 메모리 컨트롤러를 통해 제어되어 접근된다. 메

모리 컨트롤러는 패키지들을 채널이라 불리는 단위로 나누고, 각 채널마다 IO를 독립적으로 수행한다. 따라서 SSD는 여러 IO를 동시에 수행할 수 있다. [그림1]과 같이 채널이 4개라면 SSD는 최대 4개의 IO요청을 동시에 수행 할 수 있다.



[그림 1] SSD 내부 구조

4. SSD 관련 실험에서 iostat 사용의 문제점

SSD를 사용한 실험에서의 iostat 사용의 문제점은 iostat이 디바이스 util을 계산하는 방법과 관련이 있다.

Linux kernel[4]은 CPU와 디바이스의 사용률이나 IO 정보들을 실시간으로 저장해놓고, 주기적으로 저장해 둔 정보를 /proc/diskstats 파일에 쓴다. Iostat[5]은 매 주기마다 이 /proc/diskstats 파일을 읽고 CPU와 IO 정보를 받아와 그 값들을 계산해서 사용자에게 필요한 값을 출력한다. 그 중, 디바이스 util과 관련된 정보는 cpu가 디바이스에 IO request를 보낸 후, 응답을 받을 때까지 걸린 시간을 축적해놓은 값인 tot_ticks라는 값과 관련이 있다. 이 tot_ticks는 kernel에서 IO관련 함수가 불릴 때마다 디바이스가 동작하고 있다면 동작한 tick만큼을 더하는 방식으로 tot_ticks값을 업데이트 한다.

문제가 되는 부분은 iostat에서 tot_ticks 값 만으로 디바이스 util을 계산한다는 것이다. iostat에서 디바이스 util은 아래의 공식을 통해 계산 된다.

$$\text{util}(\%) = \frac{(\text{측정 시간동안 } tot\_ticks \text{ 변화량}) * (1tick \text{ 당 걸리는 시간})}{(\text{측정 단위 시간})} * 100$$

이를 해석하면, 디바이스 util은 단위 시간 동안 디바이스가 IO를 처리하고 있는 시간의 비율이다. HDD 디바이스라면 디바이스 내에서 한번에 하나의 IO밖에 처리하지 못하기 때문에, util이 100%라면 더 이상 디바이스의 성능을 끌어올릴 수 없다고 판단할 수 있을 것이다. 그러나, SSD와 같이 여러 IO를 동시에 수행하는 능력이 있는 디바이스에서는 위와 같은 util이 100%라고해서 디바이스의 성능을 완전히 끌어올렸다고 확신 할 수 없다. 측정 시간 동안 계속해서 디바이스가 동작했다고 해서 모든 채널에서 계속 IO를 수행했는지에 대한 정보가 없기 때문이다. 측정 시간 동안 하나의 채널에서만 IO가 계속 처리되고 있어도 util은 100%라 하지만 실제로 다른 채널에도 IO를 수행하면 성능을 더 끌어올릴 수 있다.

다음 장에서는 이 문제점이 실제 실험 수행 상황에서는 어떻게 나타나는 지 확인하였다.

5. 실험 및 분석

실험은 리눅스에서 디바이스 성능을 테스트 하는 fio를 사용하였다. 자세한 실험 환경은 [표 1]과 같다.

[표 1] 실험 환경

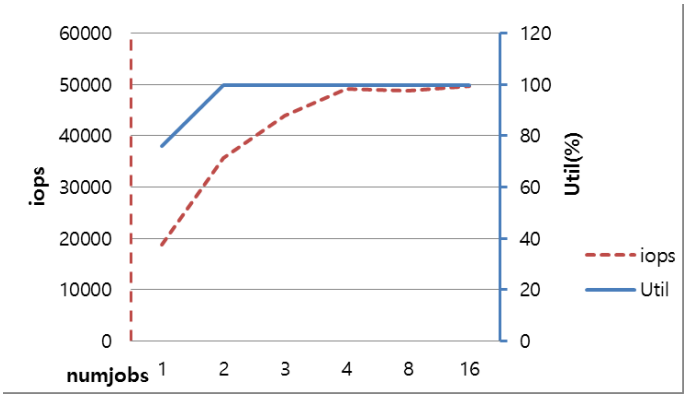
운영체제	Ubuntu 14.04LTS
프로세서	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz (X24)
메모리	2GB
저장장치	Samsung SSD 850 PRO
fio 주요 옵션	bs=4K time_based runtime=300 rw=randwrite ioengine=libaio iodepth=1 fsync=0 direct=1 numjobs=${numjobs}

fio의 옵션 중 numjobs는 동일한 fio테스트를 몇 개나 동시에 수행하도록 명령할지 지정하는 옵션이다. 이 실험에서는 numjobs의 값을 각각 1, 2, 3, 4, 8, 16으로 주어 5분동안 디바이스에 IO를 주어서 그때의 IOPS와 iostat의 디바이스 util값을 측정했다. 실험의 결과는 아래의 [표 2]와 같다.

[표 2] fio 실험 결과

jobs	1	2	3	4	8	16
util(%)	75.9	99.96	99.92	99.92	99.9	99.79
IOPS	18842	35628	43944	49095	48755	49675

[표 2]를 그래프로 그리면 아래의 [그림 2]과 같다.



[그림 2] fio 실험 결과

fio 실험 결과, 실제 job의 개수에 따른 IOPS 의 변화와 util의 변화가 동일하지 않다는 것을 알 수 있다. util의 경우, job이 두 개일 때부터 그 이후로 계속 100%를 유지하고 있다. util의 결과값으로 디바이스의 성능을 본다면, 실험 디바이스는 job이 한번에 두 개 이상 실행되는 경우 디바이스의 성능을 더 끌어올릴 수 없다는 결론이 나온다. 그러나 IOPS의 결과값으로 디바이스의 성능을 본다면, 동시에 실행되는 job이 4개가 되는 지점까지 IOPS가 계속 증가함을 알 수 있다. 이는 job이 4개가 되기 전까지는 디바이스 utilization이 100%가 나오지 않았다는 결론이 나온다. 이 상충되는 결론은 앞서 4장에서 제기했던 문제인 SSD를 디바이스로 사용 할 때 iostat util이 실제 SSD의 사용률을 제대로 알려주지 못하는 문제점과 동일한 문제임을 알 수 있다.

6. 결론 및 향후 연구

본 논문에서는 현재 리눅스에서 활발하게 사용되고 있는 디바이스 모니터링 프로그램 중 하나인 iostat이 SSD 디바이스를 제대로 모니터링하지 못하는 문제점에 대해 살펴보았다.

프로그램이 어떻게 실행되는지에 관한 분석과 실제 실험 결과 모두 iostat의 디바이스 util은 SSD의 실제 utilization을 알려주지 못했고, 이로 인해 모니터링의 신뢰도가 떨어지는 문제점이 있는 것을 보였다.

향후 이러한 문제의 원인인 SSD가 채널 수에 따른 동시에 다중 IO가 가능하다는 것을 host 상에서 알 수 있도록 하여 채널마다 모니터링을 할 수 있게 하거나, host에서 디바이스의 최대 동시에 할 수 있는 IO의 개수를 hint로 받았을 때 해당 자원을 이용해 좀 더 효율적으로 디바이스의 성능을 모니터링 할 수 있게 하는 등의 방향으로 현재의 문제점을 해결하는 방법을 연구할 계획이다.

참 고 문 헌

[1] Nguyen, Trong-Dat, and Sang-Won Lee. "I/O characteristics of MongoDB and trim-based optimization in flash SSDs." Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory. ACM, 2016

[2] Gihwan Oh, Chiyoung Seo, Ravi Mayuram, Yang-Suk Kee, Sang-Won Lee, "SHARE Interface in Flash Storage for Databases", SIGMOD'16, San Francisco, June, 2016

[3] SSD Internal Structure, https://www.usenix.org/legacy/events/usenix08/tech/full_papers/agrawal/agrawal_html/index.html/

[4] linux kernel source code, https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.10.13.tar.xz

[5] sysstat-11.2.0 source code, http://pagesperso-orange.fr/sebastien.godard/sysstat-11.2.0.tar.xz

사사

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 SW컴퓨팅산업원천기술개발사업(SW스타랩)의 연구결과로 수행되었음" (IITP-2015-0-00314)