가상 머신 간 성능 간섭 회피를 위한 SSD 내부 맵 캐시 분배 기법

박대준°, 신동군
성균관대학교 전자전기컴퓨터공학과
pdaejun@skku.edu, dongkun@skku.edu

Partitioning of Map Cache in Solid-State Drives for Performance Isolation of Virtual Machines

Daejun Park°, Dongkun Shin
Department of Electrical and Computer Engineering, Sungkyunkwan University

요 약

가상화 환경에서 호스트의 자원은 가상 머신이 서로 공유하므로, 하이퍼바이저는 스케줄링을 통해 성능 간섭이 발생하지 않도록 자원을 분배하여야 한다. 특히 HDD나 SSD와 같은 저장장치를 여러 가상 머신에게 성능 간섭이 발생하지 않도록 스케줄링 하는 기법이 연구되었다. 한편 SSD는 낸드 플래시 메모리 공정이 고도화됨에 따라 점차 저장 용량이 늘어나고 있다. DFTL은 저장 용량에 비해 적은 매핑 테이블 크기로 대용량 SSD를 관리할 수 있는 기법이다. 그러나 map miss가 일어나면 성능 저하가 불가피한 단점이 있다. 가상화 환경에서 DFTL이 적용된 SSD를 사용하면 가상 머신들이 cached mapping table을 공유하지만 이에 대한 관리는 부재한 상황이다. 이로 인해 특정 가상 머신으로 인해 로딩된 맵이 과도하게 맵 캐시에 존재하여 다른 가상 머신의 입출력 수행 시 map miss를 유발할 수 있다. 본 논문에서는 가상 머신 별로 분리된 맵 캐시를 제공하여 특정 가상 머신의 과도한 맵 캐시 점유 현상을 해결하였다.

1. 서 론

가상화(virtualization) 환경에서 여러 가상 머신들은 호스트의 자원을 공유하여 사용한다. 하이퍼바이저는 호스트의 자원을 가상머신들에게 제공할 때, 성능 상 공정성(fairness)과 고립(isolation)을 고려하는 스케줄링을 사용해야 한다. 그러므로 하이퍼바이저를 거치면서 추가적으로 발생하는 계층으로 인해 성능 상의 부하가 발생한다. SR-IOV(Single Root I/O Virtualization)는 가상화 환경에서 가상머신이 직접 입출력 장치를 접근 가능 할 수 있도록 하는 기법으로, 하이퍼바이저를 거치면서 발생한 성능 상의 부하를 피할 수 있는 장점이 있다. SSD(solid-state disk)를 위해 새롭게 제안된 NVMe(Non-Volatile Memory express) 규격은 SR-IOV를 지원하여, 가상화 환경에서 입출력 가상화를 위해 하이퍼바이저에서 존재하던 부하를 감소시켜준다. 그러나 하이퍼바이저에 존재하던 스케줄링을 거치지 않음으로써 가상머신간의 성능 공정성과 고립에 대한 문제가 발생할 수 있다. 이러한 문제를 해결하기 위해 SSD의 여유 저장 공간(over-provisioning space)을 분리하여 가상머신의 성능을 고립시키는 기법[1]과 SSD를 가상머신에게 성능이 간섭 받지 않는 채널 단위로 할당하여 물리적으로 성능 분리하는 방법[2]이 제시되었다.

한편 낸드 플래시 메모리 공정의 고도화로 인해 점차 SSD의 저장용량이 크게 증가하고 있다. 그로 인해 주소 변환을 위해 FTL(flash translation layer)에서 필요로 하는 L2P(logical to physical) 매핑 테이블의 크기가 함께 증가한다. 그런데 L2P 매핑 테이블의 크기가 SSD 내부 메모리에 로드 할 수 없을 정도로 큰 경우, L2P 매핑 테이블에 대한 요구 로딩 기법인 DFTL[3]과 같은 기법을 사용하게 된다. 그런데 DFTL에서 map miss가 발생하게 되면, 낸드 플래시 메모리에 읽기와 쓰기 연산이 발생하므로 성능 하락이 발생한다.

그런데 가상화 환경에서는 각 가상머신이 요청한 입출력 명령에 의해 로딩된 맵이 제한된 CMT(cached mapping table)를 공유해야 하므로 가상머신 간에 간섭을 받을 수 있다. 예를 들어 하나의 가상머신이 map miss를 자주 일으키는 워크로드를 가지고 있을 경우, 다른 가상머신이 사용 중인 맵이 인접한 시간 내에 다시 접근 가능하더라도 제거가 될 수 있다. 이 경우, 특정 가상머신의 워크로드로 인해 다른 가상 머신은 map miss가 발생하여 성능 하락이 발생하고, 이는 가상머신간의 성능 간섭에 해당한다.

본 논문에서는 DFTL이 적용된 SSD를 사용할 때, 가상화 환경에서 발생하는 map miss로 인한 가상 머신간의 성능 간섭을 방지하기 위해, 각 가상머신이 사용하는 CMT를 구분지어 관리하고자 한다. 이를 적용시키면, 특정 가상 머신이 발생시키는 map miss가 다른 가상 머신의 map miss에 영향을 미치지 않게 된다. 그러나 CMT를 같은 크기로 분배 한다면, 가상 머신 별 작업 집합(working set)에 따른 CMT에 대한 공간 요구를 만족시킬 수 없다. 그래서 가상 머신 별로 구분된 CMT를 동적으로 재분배 하여, 각 가상 머신의 워크로드에 따라 필요한 작업 집합을 충족시킬 수 있는 기법을 제안하고자 한다.

2. 관련 연구

가상화 환경에서 SSD를 사용할 때, 가상머신 별 입출

이 논문은 2013년도 정부(교육부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(2013R1A1A2A10013598)

력 성능 고립화를 위한 연구로, OPS isolation, VSSD, WABCS가 있다. OPS isolation은 여러 가상머신이 하나

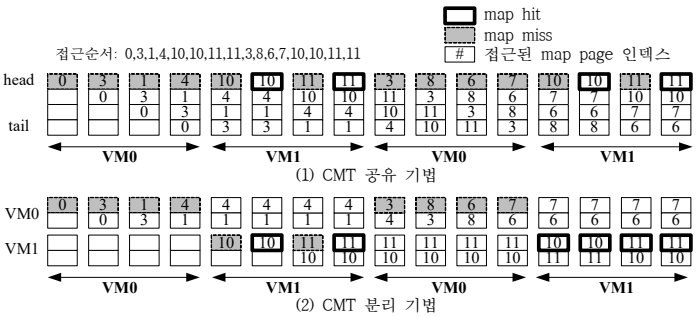

그림 1 기법 별 두 가상머신의 CMT 사용

의 SSD를 공유하는 환경에서, GC(garbage collection) 동작을 성능 고립의 방해 요인으로 지적하였다. 그리고 이를 해결하기 위해 GC가 가상 머신 별 독립적으로 수행되도록, SSD의 여유 저장 공간 영역을 가상 머신별로 할당하였다. VSSD는 SSD에서 여러 사용자간에 성능 간섭에 대해 GC를 원인으로 지적하고, GC 문제를 해결하기 위해 SSD에서 성능 고립의 물리적인 단위인 채널을 각 사용자에게 할당하였다. WABCS[4] 또한 GC를 성능 간섭의 원인으로 지적하고 이를 해결하고자 하였다. WABCS는 I/O 비용을 계산할 때, 각 가상 머신의 GC 기여도를 추가적으로 고려하고, 이를 기반으로 한 스케줄링을 통해 GC로 인한 성능 간섭을 보상하는 기법을 사용하였다. 그러나 이러한 연구는 전체 L2P 매핑 테이블이 SSD 내부 메모리에 로드 되어 있는 상황에서 GC로 인한 성능 저하만을 고려하였다. 그러므로 DFTL에서 발생하는 map miss와 이로 인한 가상 머신간의 성능 간섭 문제에 대해서는 해결 할 수 없다.

3. 가상머신 간 CMT 경쟁

DFTL은 SSD의 전체 L2P 매핑 테이블에 비해 작은 내부 메모리를 CMT(cached mapping table)로 사용하여 대용량 SSD를 관리할 수 있는 장점이 있지만, map miss가 발생하면 map flush와 map load를 위한 읽기와 쓰기 연산으로 인해 추가적인 부하가 발생하는 단점도 존재한다.

단일 사용자 환경과 달리, 여러 가상 머신이 호스트의 자원을 공유하는 가상화 환경에서는 DFTL에서 발생하는 map miss로 인해 가상 머신 간 성능 간섭 문제가 발생할 수 있다. 각 가상 머신이 자신에게 할당된 논리 공간을 가지고 있으므로, 가상 머신 간에는 map entry가 공유되지 않는다. 결국 특정 가상 머신의 CMT 독점은 다른 가상 머신의 map miss를 유발시켜, 입출력 성능을 낮출 수 있다.

그림 1은 가상화 환경에서 DFTL을 사용할 때, 시간 별로 각 가상머신이 I/O 서비스를 제공받을 때 접근하는 map entry에 대한 그림이다. 각 map entry는 플래시 메모리 페이지 단위로 관리되기 때문에 CMT에는 map page 단위로 로딩된다. 2개의 가상 머신이 SSD를 함께 사용할 때, 임의 접근을 자주 하는 VM0는 여러 map entry를 로딩하게 된다. 그러므로 map miss를 매번 발생시키고 동시에 map flush를 유발한다. 그러나 VM1은 순차 접근으로 인해 하나의 map entry만 접근하게 되어, map loading으로 인한 map flush를 자주 일으키지 않는다. CMT가 충분히 크지 않을 경우 VM0이 발생시킨 map flush에 의해 VM1이 접근하는 map entry가 flush 될 수 있다. 그림 1의 (1)을 보면 CMT가 최대 4개의 map page를 로딩할 수 있다. VM1은 10번과 11번 map page를 자주 사용함에도 불구하고, VM0의 잦은 map load로 인해 항상 10번과 11번 map page를 접근 할 때 map miss가 일어나게 된다. 그러므로 총 12번의 접근 중에서 4번의 map hit만 존재하게 되고, 특히 VM1은 반복적인 접근에도 불구하고 8번의 접근 중 50%의 map hit만 발생하게 된다.

이 문제는 DFTL로 인해 CMT가 가상 머신간의 공유자원이 되어 경쟁이 발생 함에도, 이를 관리하지 않기 때문에 발생한 것이다. 이를 해결하기 위해서는 CMT를 가상 머신 별로 구분하여, 과도한 map loading을 발생시키는 가상 머신이 CMT를 과도하게 점유하지 않도록 해야 한다.

4. CMT 분리 기법

DFTL을 사용하는 환경에서 특정 가상 머신이 CMT를 잠식하는 되어 다른 가상 머신에서 map miss가 일어나는 현상이 발생할 수 있다. CMT를 가상 머신 별로 따로 관리하면 이러한 현상을 막을 수 있다.

가장 직관적인 방법으로, CMT를 가상 머신이 똑같이 분배하여 사용할 수 있다. 그림 1의 (2)는 가상 머신별로 CMT를 동일한 크기로 분배하고, (1)과 같은 접근 패턴을 적용한 것이다. (1)에서는 VM0에 의해 10번과 11번 map page가 CMT에서 LRU에 의해 flush되었지만, (2)에서는 CMT를 서로 공유하지 않으므로 VM0의 map loading이 VM1에 영향을 끼치지 않는다. 그러므로 VM1의 8번의 접근 중에서 6번의 map hit가 발생하여, (1)에서 일어나는 CMT의 간섭 문제를 피할 수 있다.

이 방법을 사용하면 성능 고립과 공정성을 확실히 보장할 수 있지만, CMT를 충분히 활용하지 못하여 사용률이 낮아지는 문제가 생길 수 있다.

이를 해결하기 위해, 각 가상 머신의 워크로드에서 필요로 하는 작업 집합의 변화에 따라 동적으로 CMT 크기를 조절하고자 한다. 이를 위해 S-CAVE[5] 논문에서 참고한 rECS를 활용하였다.

$$rECS = \frac{\text{접근한 고유 } map page \text{ 수}}{\text{할당된 } CMT \text{크기}}$$

rECS는 할당받은 메모리의 크기 당 작업 집합의 크기를 의미한다. rECS는 주기적으로 측정되는데, 각 가상 머신별로 SSD 내부에서 수집한 데이터를 기반으로 한다. 이 때 rECS의 변화값인 delta rECS를 통해 작업 집합의 변화를 관찰 할 수 있다.

Delta rECS가 0보다 크다면, 해당 가상 머신의 작업 집합의 크기가 증가하고 있고, map을 위한 공간이 더 필요로 하다는 것을 의미한다. Delta rECS가 0보다 작으면, 해당 가상 머신의 작업 집합의 크기가 줄어들고 있고, map을 위한 공간이 충분하다는 것을 의미한다. Delta

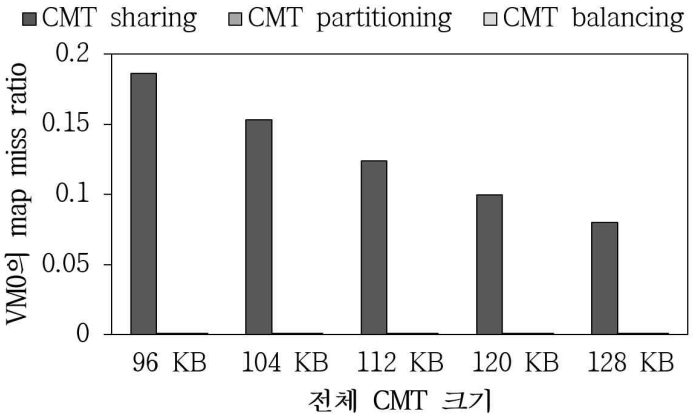

그림 2 VM0의 CMT 크기에 따른 기법별 map miss ratio

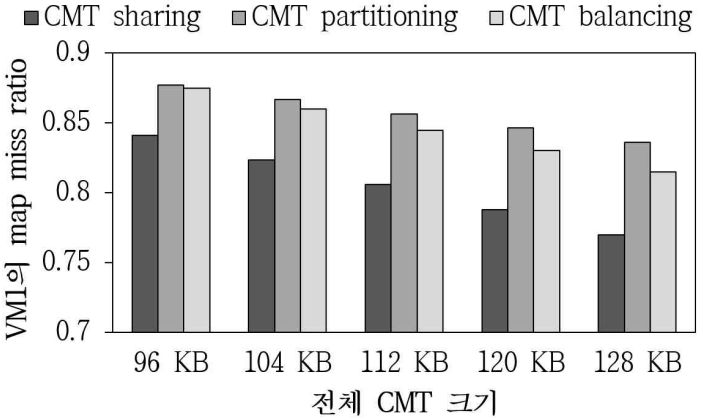

그림 3 VM1의 CMT 크기에 따른 기법별 map miss ratio

rECS을 통해, 가장 큰 값을 가지는 가상 머신에게 하나의 map page에 해당하는 크기를 CMT에서 추가 할당 하였고, 가장 작은 음수 값을 가지는 가상 머신에게는 하나의 map page에 해당하는 크기를 CMT에서 반환 시켰다.

5. 실험

본 논문에서 제안된 SSD 내부 메모리 분리 기법을 평가하기 위해, 트레이스를 입력으로 받는 이벤트 주도 SSD 시뮬레이터[6]를 사용하였다. SSD 내부 스케줄러로는 BCQ[7]를 사용하였다. 여유 저장 공간은 전체 크기의 3%로 지정하였다. SSD의 총 크기는 8GB로, 구성으로 8개 채널이 있고, 각 채널 당 2개의 웨이를 가지고 있다. 플래시 메모리의 페이지는 4KB이고, 프로그래밍 시간은 0.55ms, 읽기 시간은 65μs, 블록 삭제 시간은 1.5ms로 설정하였다. CMT partitioning은 가상 머신별로 CMT를 똑같이 분배한 것이고, CMT balancing은 10초마다 각 가상 머신에서 delta rECS 값을 통해 CMT 크기를 조절하였다.

실험에서는 VM0와 VM1, 두 개의 가상 머신을 설정하였다. 두 가상 머신은 모두 4KB 크기의 임의 쓰기를 하는데, VM0의 임의 쓰기는 시간적 지역성과 공간적 지역성이 존재하므로, VM1에 비해 map miss가 적게 일어나는 특징이 있다.

그림 2는 CMT partitioning과 CMT balancing이 CMT 공유로 인한 map miss 증가를 억제하는지 살펴보기 위해, VM0과 VM1의 IO을 동시에 SSD에서 수행하면서 VM0에서 발생하는 map miss ratio를 측정한 결과이다. 이 때, 전체 CMT 크기를 증가시키면서 기법 별로 map miss ratio를 비교하였다. 전체적으로 CMT sharing이 CMT balancing과 CMT partitioning에 비해 map miss가 많은 것을 볼 수 있는데, 이는 CMT 공유로 인해 map loading을 많이 발생시킨 VM1이 VM0의 map page를 flush 시키기 때문이다.

그림 3은 CMT partitioning으로 인한 VM1의 map miss 증가를 살펴보고 CMT balancing이 이를 얼마나 줄여주는 지 살펴보기 위해 VM1의 map miss ratio를 측정한 것이다. 그래프를 보면 CMT 크기 변화에 따라 CMT sharing이 CMT partitioning에 비해 낮은 map miss ratio를 가지고 있다. CMT partitioning은 가상 머신 별로 분리된 CMT를 제공하기 때문에, 각 가상 머신은 자신에게 할당된 CMT만을 사용할 수 있고, 자유롭게 가상 머신간에 CMT를 공유하는 기법에 비해 CMT 활용율이 떨어 질 수 있다. 그러므로 각 가상 머신에게 map miss에 대한 성능 고립은 보장 할 수 있지만, 전체적인 map miss는 늘어날 수 있다.

CMT balancing을 살펴보면 CMT partitioning에 비해 낮은 map miss ratio를 보여주는데, 이는 VM0의 작업 집합에 충분한 크기 이상의 CMT를 할당 받게 되었다고 판단되면 남는 CMT를 VM1에게 제공하였기 때문이다.

6. 결론

가상화 환경에서 가상 머신간에 발생하는 공유하는 자원은 경쟁이 발생하므로 성능 간섭을 피하기 위한 설계가 필요하다. 본 논문에서는 DFTL 환경에서의 SSD 내부 메모리 경쟁으로 인한 성능 간섭을 개선하기 위해 가상 머신별로 CMT를 분배하는 기법을 제안하였다. CMT 분배 기법을 통해 특정 가상 머신이 SSD 내부 메모리를 잠식하는 현상을 막아, 다른 가상 머신의 map miss가 증가하는 현상을 줄일 수 있었다.

참 고 문 헌

[1] Kim, Jaeho, Donghee Lee, and Sam H. Noh. "Towards slo complying ssds through ops isolation." 13th USENIX Conference on File and Storage Technologies (FAST 15). 2015.
[2] Chang, Da-Wei, Hsin-Hung Chen, and Wei-Jian Su. "VSSD: Performance Isolation in a Solid-State Drive." ACM Transactions on Design Automation of Electronic Systems (TODAES) 20.4 (2015): 51.
[3] Gupta, Aayush, Youngjae Kim, and Bhuvan Urgaonkar. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. Vol. 44. No. 3. ACM, 2009.
[4] Jun, Byunghei, and Dongkun Shin. "Workload-aware budget compensation scheduling for NVMe solid state drives." Non-Volatile Memory System and Applications Symposium (NVMSA), 2015 IEEE.
[5] Barik, Rajkishore, Jisheng Zhao, and Vivek Sarkar. "S-cave: Effective ssd caching to improve virtual machine storage performance." Parallel Architectures and Compilation Techniques (PACT), 2013 22nd International Conference on. IEEE, 2013.
[6] Y. Hu et al., "Performance Impact and Interplay of SSD Parallelism Through Advanced Commands, Allocation Strategy and Data Granularity," in Proc. of the International Conf. on Supercomputing, 2011, pp. 96.107.
[7] Q. Zhang et al., "An Efficient, QoS-Aware I/O Scheduler for Solid State Drive," in Proc. of 2013 IEEE International Conf. on Embedded and Ubiquitous Computing, 2013, pp. 1408.1415.