The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 16, No. 6, pp.177-181, Dec. 31, 2016. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2016.16.6.177
JIIBC 2016-6-22

가상화 환경에서 저널링 기법에 의한 입출력 성능저하 분석 및 개선

Analysis and Improvement of I/O Performance Degradation by Journaling in a Virtualized Environment

김성환*, 이은지**

Sunghwan Kim*, Eunji Lee**

요 약 본 논문에서는 저널링을 사용하는 전가상화 시스템에서 호스트 캐시의 효율을 높이기 위한 기법을 제안한다. 게스트의 저널링 데이터는 쓰기를 위해 호스트 캐시에 한번만 접근되는 패턴과 빈번한 sync 명령으로 인해 캐시의 효율을 감소시킨다. 이러한 성능 감소를 줄이기 위하여 본 논문에서는 게스트의 저널링 데이터가 호스트 캐시에 접근하는 것을 막는 PDC라는 기법을 제안한다. PDC는 Linux 4.14 버전에서 QEMU-KVM 2.1 버전을 기반으로 구현하였으며, 다양한 워크로드에서 3-32%의 성능 향상을 보였다.

Abstract This paper analyzes the host cache effectiveness in full virtualization, particularly associated with journaling of guests. We observe that the journal access of guests degrades cache performance significantly due to the write-once access pattern and the frequent sync operations. To remedy this problem, we design and implement a novel caching policy, called PDC (Pollution Defensive Caching), that detects the journal accesses and prevents them from entering the host cache. The proposed PDC is implemented in QEMU-KVM 2.1 on Linux 4.14 and provides 3-32% performance improvement for various file and I/O benchmarks.

Key Words : Virtualization, Caching, Journaling, File system

Ⅰ. 서 론

가상화는 개인 컴퓨터에서 클라우드 서버까지 현대 컴퓨터 시스템에서다양하게 널리 사용되고 있는 기술이다[1-9]. 가상화는 소프트웨어 플랫폼과 하드웨어 환경을 분리시켜 사용자에게 유연성과 확장성을 제공한다. 가상화 기술은 크게 전가상화 기술과 반가상화 기술로 나누어 지는데 최근 연구결과에 따르면 가상화 하이퍼바이저의 80% 이상이 Vmware, Hyper-V, QEMU-KVM과 같은 전가상화 시스템을 사용하고 있다[10].

전가상화 시스템은 게스트의 수정 없이 호스트에서 동작하기 때문에 그림 1과 같이 각각의 게스트는 자신만의 캐시를 가지고 있다. 하이퍼바이저의 설정에 따라 호스트 캐시를 우회할 수도 있지만 일반적으로는 호스트 캐시를 공유하여 사용한다. 이러한 계층적 캐쉬 구조는 중복 캐쉥 오버헤드가 발생하지만 호스트 캐쉬가 2차 공유 캐쉬로 동작함에 따라 버퍼링 및 캐쉥효과 제공하여 결과적으로 성능을 향상시킬 수 있다[11,12]. 그러나 게스트의 스토리지 접근 워크로드가 재참조 특성이 없고 버퍼

*준회원, 충북대학교 소프트웨어학과
**정회원, 충북대학교 소프트웨어학과 (교신저자)
접수일자 : 2016년 9월 22일, 수정완료 : 2016년 10월 22일
게재확정일자 : 2016년 12월 9일

Received: 22 September, 2016 / Revised: 22 October, 2016 / Accepted: 9 December, 2016
**Corresponding Author: eunji@cbnu.ac.kr
Dept. of Computer Science, Chungbuk National University, Korea

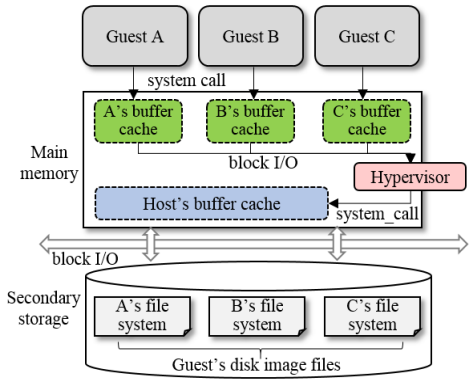


그림 1. 전가상화 시스템에서의 파일시스템 구조
Fig. 1. File system structure of a fully virtualized system.

링으로 인한 트래픽 절감 효과를 기대할 수 없다면 이중 캐쉬 구조가 성능저하를 발생시킬 수 있다[13,14].

본 논문에서는 성능 감소를 일으키는 다양한 요인 중 저널링과 연관된 요인을 분석하고 개선하였다. 저널링은 파일 시스템의 일관성을 위하여 Ext4 와 ReiserFS와 같은 파일 시스템에서 사용하는 기법이다[15,16]. 저널링 파일 시스템은 별도의 저널 영역에 데이터를 쓰고 성공적으로 쓰여진 데이터를 원래 위치에 반영함으로써 비일관성을 지닌 데이터가 발생하는 것을 방지한다. 그러나 저널링에 의해 발생하는 I/O 접근은 다음과 같은 특성이 있어 이중 캐쉬 구조에서 비효율적이다. 첫째, 저널 데이터는 시스템 크래시가 발생하지 않는 이상 다시 접근될 가능성이 없다. 둘째, 저널 영역은 데이터를 로깅하는 공간이기 때문에 큰 크기의 순차적인 쓰기 접근만이 발생한다. 이러한 순차 쓰기 접근은 하위 캐시에 존재하는 데이터를 밀어내는 효과를 가지고 있어 캐시 오염을 유발시킬 수 있다. 마지막으로 저널링은 잦은 동기화 (sync) 명령을 발생시키는데 빈번한 동기화 요청 시 호스트 캐쉬의 버퍼링 효과를 감소시켜 성능을 저하시키는 요인이 된다. 그림 2는 Ext4 파일시스템에서 저널링 트래픽의 특성을 분석한 그래프이다.

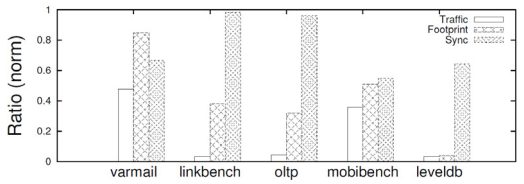


그림 2. 저널링 트래픽의 특성
Fig. 2. Characteristic of journaling traffic.

그림을 통하여 저널링 데이터는 전체 I/O 트래픽에서 평균적으로 19%를 차지하고 있으며, 저널 트래픽의 Footprint는 평균적으로 45.2% 이고, 전체 Fsync 횟수에서 86%는 저널링과 연관된 것을 알 수 있다.

본 논문에서는 이러한 저널 트래픽 특성 분석을 바탕으로 전가상화 환경에서 입출력 성능을 개선시키기 위한 캐쉥 기법에 대해 제안한다. 제안하는 PDC(Pollution-Defensive Caching) 기법은 게스트와 호스트 간에 저널 트래픽에 대한 정보를 공유함으로써 저널 데이터의 경우 호스트 캐쉬를 우회할 수 있도록 하는 캐쉥 정책이다. PDC 기법은 호스트 캐쉬에서 성능저하를 일으킬 수 있는 저널 데이터를 진입 제어함으로써 불필요한 데이터에 의해 캐쉬 공간이 오염되는 것을 방지하고 입출력 성능을 향상시킨다. 제안하는 기법은 Linux Kernel 4.14 와 QEMU-KVM 2.1 에 구현되었으며, 다양한 벤치마크를 통한 성능평가에서 기존 캐쉥 기법 대비 평균 17% 의 성능향상을 나타내었다.

본 논문은 다음과 같이 구성된다. Ⅱ장에서는 제안하는 저널 데이터가 호스트 캐쉬를 우회할 수 있도록 하는 캐쉥 정책의 알고리즘에 대해 자세히 기술하고, Ⅲ장에서는 제안하는 기법의 성능평가 및 장단점을 분석하고 Ⅳ장에서는 결론을 맺는다.

Ⅱ. Pollution Defensive Caching

PDC는 저널 데이터 호스트 캐쉬를 오염시키는 것을 방지하기 위하여 호스트 캐쉬를 우회할 수 있도록 하는 캐쉥 정책이다. PDC를 구현하기 위해서는 두 가지 문제를 해결해야 한다. 첫 번째는 수정되지 않은 게스트가 있을 때 어떻게 저널링 데이터를 구별할 것인가 이고, 두 번째는 그렇게 구별한 저널링 데이터를 어떻게 호스트 캐쉬를 우회할 수 있도록 동적으로 캐쉥 모드를 변경할 것인가이다.

첫 번째 문제부터 살펴보면, 일반적으로 저널 영역은 스토리지를 포맷할 때 그 영역이 지정되지만 LBA (Logical Block Address)를 통하여 그 위치를 찾기가 쉽지 않고, 스토리지의 크기에 따라 다양한 크기를 갖는다. 이를 해결하기 위한 한가지 방법은 명시적으로 저널 데이터를 표시하도록 하여 하이퍼바이저가 이를 구별할 수 있도록 하는 것인데, 이는 게스트를 수정해야 하기 때문에 전가상화의 개념에 위배된다. 이를 극복하기 위하여

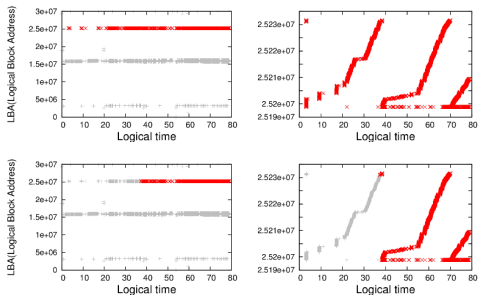


그림 3. 저널 데이터 예측의 정확도
Fig. 3. Accuracy of journal accesses prediction.

표 1. 정의된 posix_fadvise 플래그
Table 1. Action for posix_fadvise flags.

Flag	*Action*
POSIX_FADV_NORMAL	Read-ahead data on-demand
POSIX_FADV_SEQUENTIAL	Increase read-ahead window
POSIX_FADV_RANDOM	Read-ahead in a chunk size
POSIX_FADV_NOREUSE	Not implemented
POSIX_FADV_WILLNEED	Read-ahead requested pages
POSIX_FADV_DONTNEED	Invalidate data on cache

PDC는 휴리스틱 기법을 활용하였다. 현재 Ext4의 경우 디스크의 LBA 공간을 GROUP 단위 (128MB) 로 나누어 사용하는데 그 중 특정 그룹을 저널데이터로 사용한다. PDC는 각 그룹에 대해 최근에 접근했던 주소를 기억하고 순차적으로 반복 접근하는 부분을 저널데이터의 그룹으로 예측한다.

그림 3은 명시적으로 저널 구간을 나타내었을 때와 제안하는 저널 영역을 감지하는 알고리즘을 적용하였을 때의 감지된 구간을 비교한 그래프이다. 저널 구간을 예측하는 동안 한 번의 루프를 접근하는 부분은 감지되지 않지만 이후에 명시적으로 나타내었을 때와 같이 저널 데이터를 감지하는 것을 볼 수 있다.

다음으로 이렇게 추출한 저널 데이터를 하이퍼바이저가 호스트 캐쉬에 들어오지 못하도록 이를 동적으로 관리할 수 있어야한다. PDC는 posix_fadvise 시스템콜을 활용하여 이 문제를 해결하였다. posix_fadvise 시스템콜은 유저 어플리케이션이 OS에게 자신의 데이터 접근 패턴을 OS에게 명시적으로 알려 줄 수 있는 인터페이스로, 표 1과 같은 플래그들을 명시적으로 OS에게 줄 수 있다. PDC는 저널 데이터일 경우 posix_fadvise에 POSIX_FADV_NOREUSE 플래그를 설정하도록 하고, Direct I/O가 수행되도록 구현하여 저널 데이터가 호스트 캐쉬에 접근하는 것을 방지하였다.

표 2. 실험환경
Table 2. Experimental Setup

CPU	Intel Core i5-3470 3.2GHz, Quad-Core
Main memory	DDR3 Samsung 16GB
Guest Resource	Single core / 4GB Memory
Storage 1	Toshiba 1TB HDD
Storage 2	Sandisk 240GB SSD
OS	Ubuntu 14.04
Hypervisor	QEMU-KVM 2.1

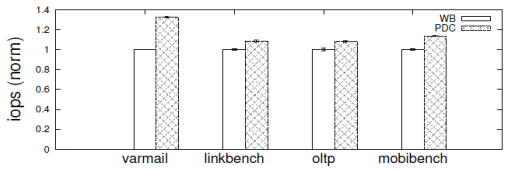
또 다른 PDC의 중요한 문제는 호스트 캐쉬와 스토리지 간의 일관성을 어떻게 맞출 것인가이다. 스토리지에 직접 쓰기를 수행하게 되면 사용되지 않는 데이터가 호스트 캐쉬에 남아 있게 되어 데이터의 일관성을 해칠 수 있다. 우리는 호스트 캐쉬의 사용하지 않는 데이터를 무효화시키기 위하여 스토리지로 쓰기를 수행할 때 POSIX_FADV_DONTNEED 플래그를 설정하도록 하였다.

Ⅲ. 성능 평가

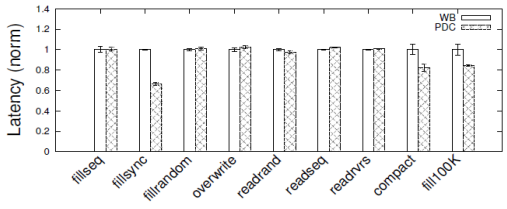
PDC의 성능을 평가하기 위해서 우리는 SSD에서 기존의 캐싱 정책인 Writeback 모드와 비교하여 측정하였다. PDC는 리눅스 커널 4.14버전에서 QEMU-KVM 2.1 버전에서 구현하였으며, 표 2와 같이 실험환경을 구성하였다. 우리는 각각의 시나리오를 10번씩 측정하고 평균을 사용하였다.

PDC로 인하여 기대되는 성능 향상은 두 가지가 있다. 첫 번째는 sync 명령어로 인한 호스트 캐쉬의 즉시 쓰기를 제거하여 쓰기 양이 줄어드는 것이고, 두 번째는 다시 재 참조되지 않는 데이터가 호스트 캐쉬를 오염시키지 못하도록 하여 캐쉬 히트율이 높아지는 것이다. 우리는 이 두 가지 요소들의 효과를 살펴보기 위하여 단일 게스트 일 때 성능을 측정하였다. 그림 4는 단일 게스트 일 때 SSD 에서 기존의 Write-back 캐싱 정책과 비교를 한 그래프이다. 그림 4(a)와 같이 PDC는 File I/O와 데이터베이스 벤치마크에서 기존의 캐싱기법 대비 8-32%의 성능 향상을 보였다. 특히 PDC는 varmail 워크로드에서 32%의 성능 향상을 보였는데, 이는 varmail 워크로드가 많은 양의 sync 명령을 보내기 때문이다. 그 외에도 PDC는 링크벤치와 OLTP에서는 각각 8.5%와 8% 성능 향상을 보였고, 모비벤치에서는 14% 성능 향상을 보였다.

그림 4(b)는 LevelDB Key-value store 어플리케이션이 사용되고 있을 때 WB와 PDC의 성능을 비교한 그래



(a) File I/O and database benchmarks



(b) Key-value store benchmark(LevelDB)

그림 4. 단일 게스트에서 WB와 PDC의 성능
Fig. 4. Performances of WB and PDC in a single guest

프이다. PDC는 읽기나 쓰기 명령에서는 큰 성능 향상을 보이지 못하였으나, fillsync와 compact 명령에서 각각 33%와 18%의 성능 향상을 보였다. 특히 compact 명령어는 주기적으로 높은 레벨(빠른 스토리지)의 유효한 데이터를 낮은 레벨(느린 스토리지)로 보내는 명령로, compact 명령어가 수행 될 때 많은 양의 I/O가 발생되고, 다른 명령어들은 수행 시간이 5us 이하인 반면, compact 명령어는 거의 1초가량 소요되기 때문에 주요한 성능 감소의 원인으로 밝혀져 있다[17]. 따라서 이러한 compact 명령어의 지연 속도가 18% 가량 감소한 것은 주목할 만한 성능 향상으로 볼 수 있다.

Ⅳ. 결론

본 논문에서는 전가상화 환경에서 게스트의 저널링이 호스트 캐시 성능에 어떠한 영향을 미치는지 분석하였다. 우리는 다시 재 참조되지 않고 바로 동기화 명령을 내리는 저널 데이터가 호스트 캐시에 악영향을 미치는 것을 발견하였고, Pollution-defensive caching이라는 새로운 캐싱 기법을 도입하여 이를 해결하였다. 그리고 우리는 실험을 통해 우리가 제안한 캐싱 정책이 가상화 시스템에서 3-32%의 성능이 향상됨을 보여주었다.

References

[1] M. Ben-Yehuda, M. Factor, E. Rom, A. Traeger, E. Borovik, and A. B. Yassour. “Adding advanced stor-age controller functionality via low-overhead vir-tualization.” In Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST), 2012.

[2] G. Casale, S. Kraft, and D. Krishnamurthy. A model of storage i/o performance interference in virtualized systems. In Proceedings of the Interna-tional Workshop on Data Center Performance (DCPerf), 2011.

[3] J. N. Christoffer Dall. Kvm/arm: The design and implementation of the linux arm hypervisor. In Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2014.

[4] A. Gulati, C. Kumar, and I. Ahmad. Storage workload characterization and consolidation in virtualized environments. In Proceedings of 2nd Interna-tional Workshop on Virtualization Performance, 2009.

[5] S. Hajnoczi. An updated overview of the qemu storage stack. LinuxCon Japan, 2011.

[6] D. Hildebrand, A. Povzner, R. Tewari, and V. Tarasov. Revisiting the storage stack in virtualized nas environments. In Proceedings of the Workshop on I/O Virtualization (WIOV), 2011.

[7] D. Le, H. Huang, and H. Wang. Understanding performance implications of nested file systems in a virtualized environment. In Proceedings of the 10th USENIX Conference on File and Storage Technol-ogies (FAST), 2012.

[8] R. Russell. virtio: towards a de-facto standard for virtual i/o devices. ACM SIGOPS Operating Systems Review, 42(5):95 - 103, 2008.

[9] V. Tarasov, D. Hildebrand, G. Kuenning, and E. Za-dok. Virtual machine workloads: The case for new nas benchmarks. In Proceedings of the 11th USENIX Conference on File and Storage Technol-ogies (FAST), 2013.

[10] Vmworld 2013, “Is vmware the mumford and sons

of the cloud?" URL http://www.techweekeurope.co.uk/work-space/vmworld-vmware-cloud-mumford-and-sons-12950.
[11] Virtualbox. URL http://www.virtualbox.org.
[12] K. Wolf. A block layer overview. KVM Forum, 2012. URL http://www.linux-kvm.org/page/KVM_Forum_2012.
[13] D. H. Kim, H. K. Bahn, "Buffer Cache Management of Smartphones Exploiting Write-Only-Once Characteristics", The Journal of The Institute of Internet, Broadcasting and Communication (JIIBC), VOL. 15 NO. 6, pp.129-134, 2015
[14] M.Y. Sung, "Performance Evaluation of Flash Memory-Based File Storages: NAND vs. NOR", Journal of the Korea Academia-Industrial cooperation Society(JKAIS), Vol. 9, No.3, pp.710-716, 2008.
[15] Ext4 wiki. URL https://ext4.wiki.kernel.org/index.php/Main_Page.
[16] Reiserfs. URL https://en.wikipedia.org/wiki/ ReiserFS.
[17] Rocksdb. URL https://github.com/facebook/rocksdb/wiki/RocksDB-Basics.

저자 소개

김 성 환(준회원)

- 2016년 2월: 충북대학교 소프트웨어학과 학사
- 2016년 3월~: 충북대학교 소프트웨어학과 석사과정

이 은 지(정회원)

- 2005년 2월: 이화여자대학교 컴퓨터공학과 학사
- 2012년 2월: 서울대학교 컴퓨터공학부 박사
- 2014년 3월~: 충북대학교 소프트웨어학과 조교수

※ 이 논문은 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No.NRF-2014R1A1A3053505)