가상화 지원 스왑 장치를 이용한 효율적인 페이징 기법

민창우*, 김인혁**, 김태형**, 엄영익**
*성균관대학교 휴대폰학과
**성균관대학교 정보통신공학부
e-mail : {multics69, kkojiband, kim15m, yieom}@ece.skku.ac.kr

Virtualization Aware Swap Device for Efficient Paging

Changwoo Min*, Inhyuk Kim**, Taehyoung Kim**, Young Ik Eom**
*Department of Mobile Systems Engineering
**School of Information and Communication Engineering
Sungkyunkwan University

요 약

가상화는 서버통합을 통하여 가상머신 간의 하드웨어 자원을 공유함으로써, 총 소유 비용을 줄일 수 있어서 널리 사용되고 있다. 하지만 메모리는 다른 장치와 달리 쉽게 공유되기 어려워서 서버 통합에 있어서 병목이 되고 있다. 이를 해결하기 위한 여러가지 방법들중 많은 방법이 공통적으로 가상머신모니터에서 페이징을 사용하고 있다. 하지만 게스트 운영체제와 가상머신모니터가 모두 페이징을 할 경우, 페이징이 급격히 증가가는 이중 페이징 문제가 발생할 수 있다. 본 논문에서는 이중 페이징 문제를 해결하기 위한 방법으로 가상머신모니터와 게스트 운영체제가 스왑 장치를 공유하는 가상화 지원 스왑 장치를 제안한다. 또한 실험을 통하여 가상머신모니터가 페이지 교환 알고리즘으로 LRU 를 사용할 경우 이중 페이징 문제가 크게 발생할 수 있음을 보인다.

1. 서론

가상화(virtualization)는 하나의 서버에 다수의 가상머신(VM, Virtual Machine)을 수행하는 기법으로, 가상머신간에 하드웨어 자원을 공유함으로써 총소유 비용을 줄일 수 있어서 널리 사용되고 있다. 가상머신모니터(VMM, Virtual Machine Monitor)는 하드웨어와 가상머신의 게스트 운영체제(guest operating system) 사이에서 하드웨어 리소스의 관리를 담당한다. 가상머신모니터는 시분할기법을 이용하여 CPU 와 IO 장치를 효율적으로 공유할 수 있다. 하지만 메모리의 경우, 시분할기법으로 가상머신간에 효율적으로 공유될 수 없어서 서버통합(server consolidation)에 병목이되고 있다.

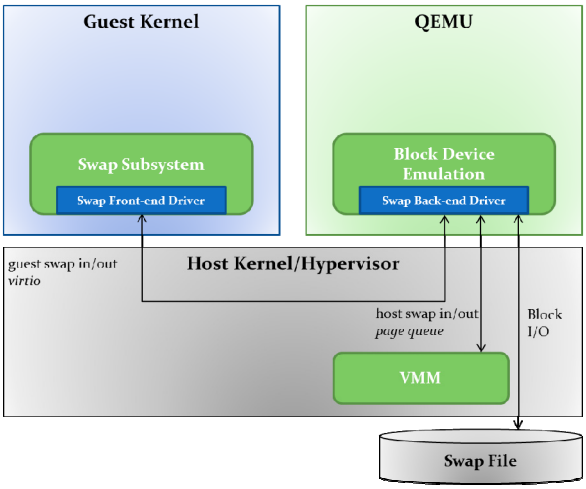
최근에는 가상머신간의 효율적인 메모리 공유를 위하여 ballooning 기법과 페이지 공유 기법등 다양한 기법들이 제안되었다 [1, 2, 3]. 본 논문에서는 여러가지 메모리 공유 기법에서 공통적으로 사용되는 가상머신모니터 페이징 기법을 기상화 지원 스왑 장치(swap device)를 이용하여 효율적으로 수행할 수 있는 방법을 제안한다.

본 과제(결과물)는 교육과학기술부. 지식경제부의 출연금으로 수행한 산학협력중심대학육성사업의 연구결과입니다.

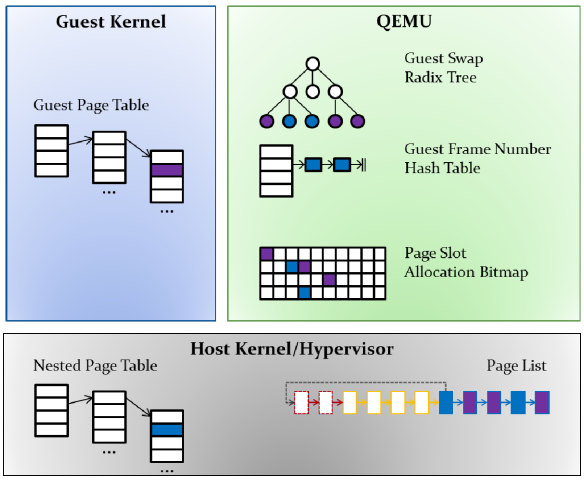
논문의 구성은 2 장에서는 관련연구에 대해서 살펴보고, 3 장에서는 제안기법의 설계 및 구현에 대해서 설명한다. 4 장에서는 실험 및 평가 결과를 보이며 5 장에서 결론을 맺는다.

2. 관련연구

가상머신간의 효율적인 메모리 공유를 위한 대표적인 기법으로는 VMWare ESX 에서 제안되어 널리 사용되고 있는 ballooning 기법과 내용기반 페이지 공유 기법이 있다[1, 2]. Ballooning 은 가상머신의 게스트 운영체제에 가상머신모니터와 직접통신하는 balloon 장치 드라이버가 필요하다. 가상머신모니터는 가상머신의 메모리 할당량을 변경할 필요가 있을 때, balloon 장치 드라이버에게 요청하게 된다. Balloon 장치 드라이버는 가상머신모니터의 요청에 따라 메모리 할당량을 줄일 필요가 있을 때는 게스트 운영체제에서 물리 페이지를 할당받아 가상머신모니터에게 알려주는 역할을 한다. 가상머신모니터는 balloon 장치 드라이버가 할당한 페이지를 다른 가상머신에게 할당함으로써 메모리 할당량을 변경하게 된다. 반대로 메모리 할당량을 늘릴 경우에는 balloon 장치 드라이버가 할당했던 메모리를 게스트 운영체제에서 해제함으로써 메모리 할당량을 변경한다.



(그림 1) 가상화 지원 스왑 장치 구조



(그림 2) 페이지 슬롯 리매핑 방법

내용기반 페이지 공유 기법은 가상머신모니터가 각 가상머신에 할당되어있는 페이지 중에서 내용이 동일한 페이지를 찾아서 한 개의 물리 페이지만 할당하도록 하는 기법으로 동일한 내용의 페이지가 많을 확률이 높은 유사 워크로드에서 공유 효율이 높은 것으로 알려져 있다. 공유되는 페이지에 변경이 가해질 경우에는 copy-on-write 기법을 이용하여 공유를 해제하게 된다.

Ballooning 기법의 경우, 가상머신모니터가 메모리 반환을 요청한 후에 가상머신의 게스트 운영체제로부터 반환이 실시간으로 이루어지지 않으므로 일시적으로 필요한 물리 메모리의 크기가 가상머신에 할당된 크기보다 큰 경우가 존재하게 된다. 비슷하게 내용기반 페이지 공유 기법에서도 copy-on-write 가 발생하여 새로운 페이지의 할당이 필요한 경우에 할당된 물리 메모리보다 더 많은 메모리가 필요하게 된다.

이러한 문제점을 해결하기 위해, 두 방법 모두 가상머신모니터에서 페이징 기법(host paging)을 사용한다. 하지만 가상머신모니터와 가상머신이 모두 페이징 기법을 사용하고, 가상머신에 필요한 메모리보다 적은 메모리가 할당되어 가상머신모니터에서 페이징이 발생하는 경우 이중 페이징 문제(double paging anomaly)가 발생할 수 있다. 이는 가상머신모니터가 특정 페이지를 swap-out 한후에 같은 페이지를 가상머신의 게스트 운영체제가 swap-out 하려고 시도할 경우, 가상머신모니터에서 swap-in 이 발생하여 페이징이 급격히 증가하는 문제이다. 특히 가상머신모니터와 가상머신이 유사한 페이지 교환 정책을 사용하는 경우에는 같은 페이지를 victim 으로 선정할 가능성이 높아진다 [4].

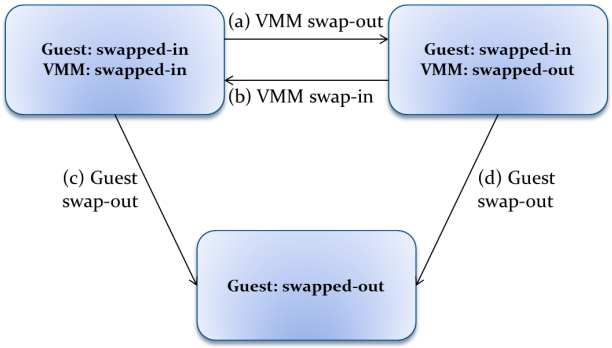
가상머신모니터에서 효율적인 페이징을 위하여 IBM 에서는 CMM(Collaborative Memory Management) 기법을 제안하였다 [3]. CMM 기법은 게스트 운영체제의 페이지 중에서 사용하고 있는 페이지와 swap-out 없이 회수가 가능한 페이지에 대한 정보를 가상머신모니터와 공유한다. 가상머신모니터를 이 정보를 바탕으로 swap-out 을 위한 victim 을 선정하게 된다. 하지만 이 기법은 게스트 운영체제의 페이지 정보를 가상머신모니터와 공유하기 위한 추가적인 오버헤드가 들고 게스트 운영체제에 수정이 필요하므로 소스 코드가 공개되어 있지 않은 운영체제에는 적용할 수 없는 단점이 있다.

본 논문에서는 가상화지원 스왑 장치를 이용하여 게스트 운영체제의 수정없이 이중 페이징 문제를 해결하여 효율적인 페이징이 가능한 방법을 제안한다.

3. 가상화 지원 스왑 장치

이중 페이징 문제는 가상머신모니터에서 swap-out 된 페이지를 게스트 운영체제에서 swap-out 하려고 할 때, swap-out 을 위해서 해당 페이지를 접근하므로 가상머신모니터에서 다시 swap-in 이 발생하는 현상이다. 이러한 문제점을 해결하기 위해서 본 논문에서는 가상머신모니터와 가상머신의 게스트 운영체제가 같은 스왑 장치(swap device)를 공유하는 구조를 제안한다.

그림 1 은 본 논문에서 제안하는 가상화 지원 스왑 장치의 구조이다. 제안 시스템은 게스트 운영체제의 스왑 프론트엔드 드라이버(swap front-end driver)와 QEMU 의 스왑 백엔드 드라이버(swap back-end driver) 그리고 페이징이 가능한 가상머신모니터로 구성된다. 디바이스 드라이버 에뮬레이션(emulation)을 담당하는 QEMU 에 스왑 백엔드 드라이버가 존재하며 이는 게스트 운영체제로부터의 스왑 요청과 가상머신모니터로부터 스왑 요청을 모두 처리한다. 게스트 운영체제에는 스왑 프론트엔드 드라이버가 존재한다. 스왑 프론트엔드 드라이버는 게스트 운영체제에게는 하드 디스크와 같은 일반 블록 디바이스로 보이나 실제로는 게스트 운영체제의 스왑 요청을 스왑 백엔드 드라이버에 전달하는 역할을 한다. 게스트 운영체제의 스왑 요청은 *<GFN(Guest Physical Page Frame Number), GPSN(Guest Page Slot Number), direction>*의 쌍으로 구성된다. 둘 사이의 통신은 효율적인 장치 가상화 모델인 virtio 를 이용한다 [7]. 가상머신모니터의 스왑 요청도 마찬가지로 QEMU 스왑 백엔드 드라이버에 전달되어 처리된다. 이 경우 스왑 요청은 *<GFN(Guest Physical Page Frame*



(그림 3) 게스트 페이지 상태 천이도

Number), direction>의 쌍으로 구성되며, 둘 사이의 통신은 공유 큐(shared queue) 구조를 이용한다.

QEMU 백엔드 드라이버는 하나의 스왑 디바이스에 게스트 운영체제의 스왑 요청과 가상머신모니터의 스왑 요청을 모두 처리해야 한다. 게스트 운영체제의 경우 운영체제에서 스왑 장치에 대한 페이지 슬롯을 관리 및 할당하므로 스왑 요청에 GFN 과 direction 은 물론 게스트 운영체제에서 할당한 페이지 슬롯 번호인 GPSN 이 포함되게 된다. 반면에 가상머신모니터의 경우, 페이지 슬롯 할당을 QEMU 백엔드 드라이버에 맡기므로 스왑 요청은 GFN 과 direction 으로만 구성되게 된다. 백엔드 드라이버는 이와 같이 상이한 두개의 스왑 요청을 처리하기 위해서 그림 2 에서와 같이 두개의 매핑 테이블(mapping table)과 페이지 슬롯 할당 비트맵(page allocation status bit map)을 관리한다. 먼저 게스트 운영체제가 할당한 게스트 페이지 슬롯(GPSN)과 백엔드 드라이버가 할당한 실제 페이지 슬롯(PPSN:Physical Page Slot Number)간의 매핑 정보를 저장하기 위해서 백엔드 드라이버는 GSRT(Guest Swap Radix Tree)를 관리한다. GSRT 는 GPSN 을 검색 키(key)로 하고 PPSN 을 값(value)로 하는 기수트리(radix tree)로써, 백엔드 드라이버는 해당 GPSN 에 대응하는 PPSN 을 할당할 때마다 GRST 에 매핑 정보를 추가하게 된다. GFNHT(Guest Physical Page Frame Number Hash Table)는 가상머신모니터의 스왑 요청에 대한 매핑 정보를 저장하기 위한 해쉬 테이블(hash table)로 GFN 와 PPSN 간의 매핑 정보를 저장하고 있다. 페이지 슬롯 할당 비트맵은 비트맵 형태로 각 PPSN 이 할당되었는지 여부를 나나태며, 백엔드 드라이버는 이를 참조하여 할당되지 않은 페이지 슬롯을 할당하고 비트맵을 갱신한다. 페이지 슬롯에 할당과 해제는 다음과 같은 경우에 이루어 진다. 게스트 운영체제의 스왑 요청에 대해서 요청된 GPSN 에 대한 PPSN 이 없으면 새로운 페이지 슬롯을 할당한다. 게스트 운영체제에서는 swap-out 된 이후에 변경되지 않은 페이지에 대해서는 추후에 swap-out 없이 swap-in 할 수 있으므로 게스트 운영체제의 스왑 요청에 의해서 할당된 PPSN 은 해제하지 않고 계속 사용한다. 반면에 가상머신모니터의 스왑 요청에 대해서는 페이지 슬롯의 사용 기간이 swap in/out 에 의해서 분명히 판단될 수 있으므로 swap-out 시에 PPSN 을 할당하고 swap-in 시에 할당을 해제한다.

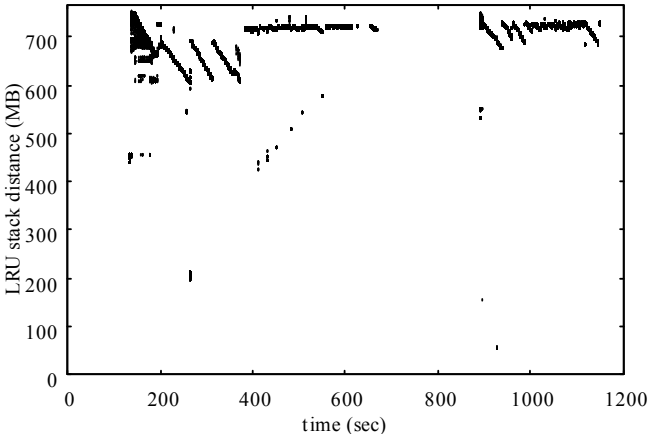
가상머신모니터가 스왑 요청을 하는 경우 다음과 같은 과정을 거쳐서 처리되게 된다. 스왑 요청은 <*GFN, direction*>의 쌍으로 공유 큐를 통해서 백엔드 드라이버에 전달되게 된다. 백엔드 드라이버는 direction 이 swap-out 이 경우, 비트맵을 참조하여 새로운 페이지 슬롯을 할당하고 이에 대한 매핑 관계를 GFNHT 에 저장한다. 백엔드 드라이버는 GFN 의 페이지에 대해서 할당된 PPSN 에 쓰기 작업을 수행하여 스왑 요청에 대한 처리를 마치게 된다. 반대로 direction 이 swap-in 인 경우, GFNHT 에서 GFN 에 할당된 PPSN 을 찾는다. PPSN 은 swap-in 을 통해서 사용이 종료되므로 비트맵과 GFNHT 에서 할당 정보와 매핑 정보를 삭제하게 한다. 백엔드 드라이버는 PPSN 의 기록된 데이터를 해당 GFN 에 읽어 들임으로써 스왑 요청에 대한 처리를 마치게 된다.

게스트 운영체제가 스왑 요청을 하는 경우 다음과 같은 과정을 거쳐서 처리되게 된다. 스왑 요청은 <*GFN, GPSN, direction*>의 쌍으로 virtio 인터페이스를 통하여 전달되게 된다. Direction 이 swap-out 인 경우, 해당 페이지가 가상머신모니터에 의해서 이미 swap-out 되어 있을 수 있는지 여부를 GFNHT 를 검색하여 확인한다. 만약 해당 GFN 이 GFNHT 에 있는 경우, 해당 페이지가 이미 가상머신모니터에 의해서 swap-out 된 것이므로 GFNHT 에서 <GFN, PPSN>의 매핑을 삭제하고 GSRT 에 <GPSN, PPSN>의 매핑을 추가하여 GFNHT 에 할당된 PPSN 을 GSRT 로 옮기게 된다. 이 경우, 가상머신모니터에서 swap-out 된 페이지를 swap-in 하지 않고 매핑 테이블 정보만 갱신함으로서 게스트 운영체제의 swap-out 을 처리할 수 있으므로 이중 페이징 문제를 해결할 수 있다. 반면에 게스트 운영체제가 swap-out 하려는 GFN 이 GFNHT 에 없는 경우는 가상머신모니터에 의해서 swap-out 되지 않은 것이다. 이 경우 GSRT 를 검색하여 이미 할당된 PPSN 이 있는지 확인하고 없는 경우 새로운 PPSN 을 비트맵을 참조하여 할당하여 GSRT 에 추가한다. 이 경우 실제 I/O 가 필요하므로 해당 GFN 을 할당 받은 PPSN 에 쓰기연산을 수행한다.

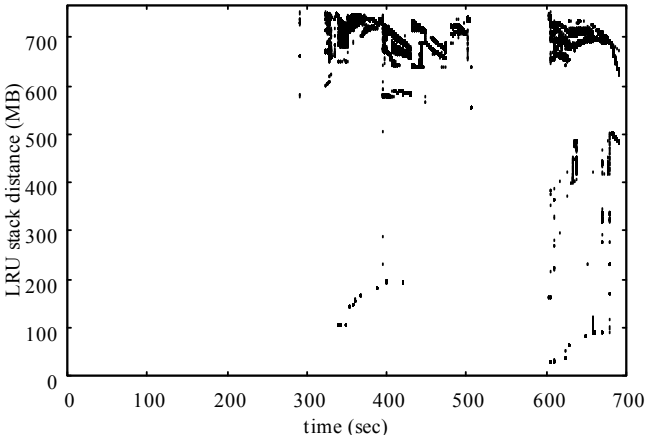
게스트 운영체제가 swap-in 을 요청한 경우는 다음과 같은 특징이 있다. 게스트 운영체제는 swap-out 하지 않은 페이지에 대해서 swap-in 을 요청하지 않으므로 요청 GFN 에 대한 PPSN 할당은 이미 되어 있다. 또한 가상머신모니터에서 swap-out 이 발생한 경우에도 게스트 운영체제의 swap-out 에 의해서 매핑이 GFNHT 에서 GSRT 로 옮겨져 있으므로 매핑 관계는 항상 GSRT 에 존재하게 된다. 따라서 swap-in 의 경우, GSRT 에서 GFN 에 해당되는 PPSN 을 찾아서 PPSN 의 페이지 슬롯을 GFN 으로 읽어들이면 해당 요청에 대한 처리가 완료되게 된다.

4. 실험

본 논문에서 제안한 기법은 오픈소스 가상머신인 KVM[5]을 확장하여 구현되었다. 장치 가상화를 위해서는 QEMU[6]를 사용하였다.



(그림 4) 401.bzip2 의 게스트 운영체제 swap-out



(그림 5) 403.gcc 의 게스트 운영체제 swap-out

실험에 사용된 게스트 운영체제는 리눅스 커널 2.6.32 버전이었으며, 최대 768MB 까지 메모리를 할당 받을 수 있도록 설정하여 실험하였다. 모든 실험은 2.67 GHz Intel Core i5 쿼드코어 프로세서를 장착한 PC 에서 수행되었다. 가상머신모니터의 페이징을 위한 페이지 교환 정책은 LRU(Least Recently Used)로 구현하였다. LRU 리스트는 게스트 물리페이지의 접근을 페이지 권한 비트를 off 함으로써 모니터링하여 구성하였다. 실험은 실제 환경에서 이중 페이징 문제가 얼마나 발생할 수 있는지 확인하기 위하여 게스트 운영체제에 특정 워크로드를 수행시킨 후에, 게스트 운영체제에서 swap-out 시키는 GFN 에 대하여 가상머신모니터상의 LRU 페이지 리스트상의 거리(LRU page distance, LRU stack distance)가 얼마인지를 측정하였다. 게스트 운영체제에서 swap-out 할 것으로 선정한 GFN 의 LRU 거리가 크다면 가상머신모니터에서 필요한 양보다 적은 메모리를 할당하고 이때 swap-out 할 victim 을 LRU 방법에 의해서 선정하는 경우 이미 가상머신모니터에서 swap-out 된 페이지를 게스트 운영체제에서 swap-out 하여 이중 페이지 문제가 발생할 확률이 높아지게 된다.

실험에 사용된 워크로드는 SPEC CPU 2006 [8] 중 일부를 사용하였다. SPEC CPU 2006 은 CPU 중심의 응용프로그램 수준 벤치마크로써 12 개의 정수 계산 벤치마크와 17 개의 부동소수점 계산 벤치마크로 구성되어 있다. 본 논문에 실험에서는 그 중에서 메모리 사용량이 많아서 본 실험조건에서 페이징을 유발시키는 벤치마크인 401.bzip2 와 403.gcc 에 대해서 수행하였다. 그림 3 과 그림 4 는 401.bzip2 와 403.gcc 에 대한 실험 결과를 보여준다. 그래프에서 x 축은 초(seconds)로서 게스트 운영체제 부팅이후의 시간을 나타내며, y 축은 게스트 운영체제에서 swap-out 이 발생했을 때 가상머신모니터의 LRU 에서 거리를 MB 단위로 환산하여 보여준다. 게스트 운영체제에서 사용할 수 있는 최대 메모리 크기를 768MB 로 설정하였으므로 y 축의 최대값은 768MB 가 되며 값이 클 수록 가상머신모니터의 LRU 에서 꼬리 쪽에 있음을 의미한다. 그림 3, 4 의 실험결과는 게스트에서 swap-out 되는 페이지의 LRU 거리가 상당히 커서 가상머신모니터가 LRU 방식으로 victim 을 선정하면 이중 페이지 문제가 발생할 가능성이 높음을 보여 주고 있다.

5. 결론

본 논문에서 가상화 환경에서 서버 통합의 병목문제를 해결하기 위한 여러가지 기법에서 공통적으로 사용하고 있는 가상머신모니터 페이징에서 발생하는 이중 페이지 문제를 해결하기 위한 방법을 제시한다. 본 논문에서 제시하는 방법은 게스트 운영체제와 가상머신모니터가 스왑 장치를 공유하고 스왑장치에서 게스트 swap-out 시 해당 페이지가 가상머신모니터에서 swap-out 되었는지 여부를 검사하여 해결하도록 하였다. 또한 실험을 통하여 가상머신모니터가 페이지 교환 정책으로 LRU 를 사용할 경우, 이중 페이지 문제가 상당한 빈도로 발생할 수 있음을 보였다.

참고문헌

[1] C. A.Waldspurger. Memory resource management in VMware ESX server. SIGOPS Oper. Syst. Rev., 36(SI):181–194, 2002

[2] Diwaker Gupta, Sangmin Lee, Michael Vrable, Stefan Savage, Alex C. Snoeren, George Varghese, Geoffrey M. Voelker, and Amin Vahdat. Difference engine: Harnessing memory redundancy in virtual machines. In Usenix OSDI, December 2008.

[3] M. Schwidefsky, H. Franke, R. Mansell, H. Raj, D. Osisek, and J. Choi. Collaborative Memory Management in Hosted Linux Environments. In Proceedings of the 2006 Ottawa Linux Symposium, 2006.

[4] R. Goldberg and R. Hassinger. The double paging anomaly. In Proc. 1974 National Computer Conference, pages 195–199, May 1974.

[5] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. kvm: the Linux virtual machine monitor. In Proceedings of the 2007 Ottawa Linux Symposium, pages 225–230, July 2007.

[6] QEMU. URL http://wiki.qemu.org/

[7] R. Russell. Virtio: towards a de-facto standard for virtual I/O devices, ACM SIGOPS Operating Systems Review, Vol.42, No.95-103 (2008).

[8] SPEC CPU2006. URL http://www.spec.org/cpu2006/