중첩 가상화 환경에서 메모리 오버커밋을 하는 하이퍼바이저 VM과 중첩 VM의 성능 평가*

유태묵[1], 임종범[1], 정광식[2], 서태원[3], 유헌창[1]**
[1]고려대학교 대학원 분산클라우드컴퓨팅 연구실
[2]방송통신대학교 컴퓨터과학과
[3]고려대학교 대학원 컴퓨터시스템 연구실
e-mail: lyoo12@korea.ac.kr, jblim@korea.ac.kr, kchung0825@knou.ac.kr, suhtw@korea.ac.kr, yuhc@korea.ac.kr

Performance Evaluation of Hypervisor VMs and Nested VMs Overcommitting Memory in Nested Virtualization Environments

Taemuk Lyoo[1], JongBeom Lim[1], Kwang-Sik Chung[2], Teaweon Suh[3] and Heonchang Yu[1]
[1]Distributed and Cloud Computing Lab., Korea University
[2]Computer Science, Korea National Open University
[3]Computer Systems Lab., Korea University

요 약

가상화는 가상의 자원이 물리적 자원에 접근할 수 있게 해주는 기술이며 VM(가상머신)을 다수 설치하여 VM의 수만큼 운영체제들을 이용할 수 있다. 이러한 가상화는 자원의 낭비를 막고 관리비용을 줄이기 위해 사용한다. 가상화 기술은 CPU, 메모리, I/O 가상화로 구분 지을 수 있으며 이 중 메모리 가상화 기술은 메모리 자원의 효율적인 사용을 가능하게 해준다. 여러 VM들이 실제 머신의 메모리보다 많은 메모리를 할당받아 사용하는 것이 가능한데 이것을 오버커밋 상태라고 한다. 중첩 가상화는 VM에 하드웨어 가상화 기법의 사용을 허용하게 하여 VM 위에 또 다른 VM이 동작할 수 있는 환경을 제공해준다. 이와 같은 (중첩) 가상화 환경에서의 메모리 접근은 일반적으로 하드웨어 지원을 통한 중첩 페이징 기법을 이용하여 메모리의 접근이 이루어진다. 본 논문에서는 오버커밋 발생 시 중첩 VM과 하이퍼바이저 VM의 성능 차이를 실험을 통하여 보여주고자 한다.

1. 서론

가상화는 물리적 서버나 자원을 하드웨어의 지원을 받아 VM의 vCPU와 같은 가상의 하드웨어를 호스트 하드웨어와 연결해주는 기술이다. VM을 다수 설치한다는 것은 한 대의 물리적 시스템에서 VM의 수만큼 OS(운영체제)를 운용할 수 있다는 것이다. 이러한 가상화 기술은 넓은 인프라를 사용하고 있는 대기업이나 업무의 특성상 오류 복구나 업무 연속성, 보안이 요구되는 분야에서 이용되고 있다.

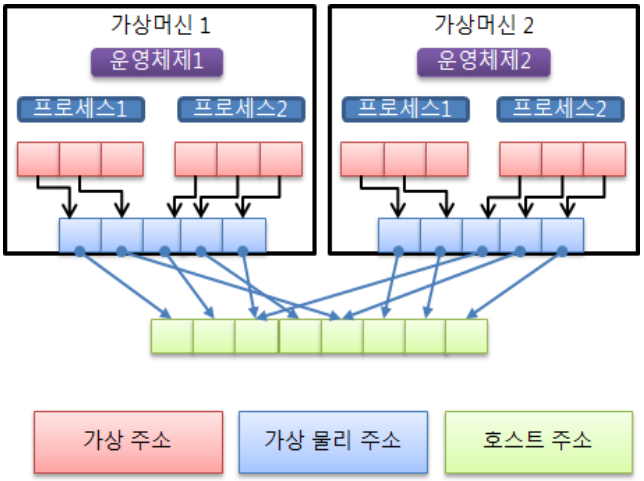
현재 가상화 기술은 많은 공공기관에 도입하여 사용하고 있다. 적은 자원으로 작업이 가능하게 함으로써, 가상화 기술이 없던 환경에 비해서 불필요한 자원의 낭비를 막고 그에 수반되는 시스템 자원에 대한 관리 비용을 절감하고 있다. 또한 몇몇 기업에서도 이러한 관리 비용을 절감하기 위하여 가상화 기술을 업무에 적용시키고 있다.

오늘날 가상화 기술은 크게 CPU 가상화, I/O 가상화, 메모리 가상화로 나뉜다. 이 중 메모리 가상화는 가상의 메모리가 호스트 메모리에 접근이 가능하게 하는 것이다. 이를 위해 가상화 페이징 기법을 이용하여 메모리 가상화를 구현한다. 이 때 사용되는 가상 메모리는 CPU가 내보내는 가상 주소를 메모리 관리 유닛(MMU)를 통하여 호스트 물리주소로 변환하는 것이다. 이러한 기본적인 가상메모리의 개념을 통하여 메모리 가상화가 이루어진다.

메모리 가상화는 VM의 가상주소와 가상 물리주소, 호스트 머신주소로 구분 되며, 가상 주소가 메모리를 읽으려할 때 가상 주소에서 가상 물리주소로 접근을 한다. 가상 물리주소는 가상 페이징 테이블을 통하여 호스트 머신주소로 접근을 하여 가상 주소가 호스트 머신 주소에 접근한다. 이와 같이 가상 주소가 물리주소로 매핑되는 과정은 (그림 1)에서 보여 준다.

(그림 1)에서 호스트 머신 메모리 크기가 8GB라고 가정하고 두 개의 VM들의 메모리 크기가 5GB라고

* 이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2013056346)
** 교신저자



(그림 1) VM 메모리 주소 구조

가정한다면 VM들의 메모리 크기만을 고려했을 때 호스트 머신의 물리적 메모리 공간은 10GB를 필요로 하게 된다. 하지만 호스트 머신의 물리적 메모리 공간이 8GB임에도 두 개의 VM들을 사용할 수 있다. 이러한 상황일 때 오버커밋이 발생하였다라고 할 수 있다. 즉, 메모리 오버커밋은 호스트 머신의 메모리보다 할당된 VM들의 메모리 크기가 클 때를 말한다[1].

메모리 오버커밋 상태에서 메모리를 관리하는 페이징 기법은 크게 두 가지가 있다. 하나는 섀도우 페이징 기법이다. 이 기법은 VM의 가상 주소와 호스트 머신 주소에 대한 페이지 테이블을 하이퍼바이저에서 관리를 해준다. 하지만 VM에서 페이지 테이블을 접근하려고 할 때마다 하이퍼바이저로 트랩과 에뮬레이션이 발생하고, 섀도우 페이지 테이블과 VM의 페이지 테이블의 동기화를 맞추어주기 위한 오버헤드가 크다는 단점이 있다[3].

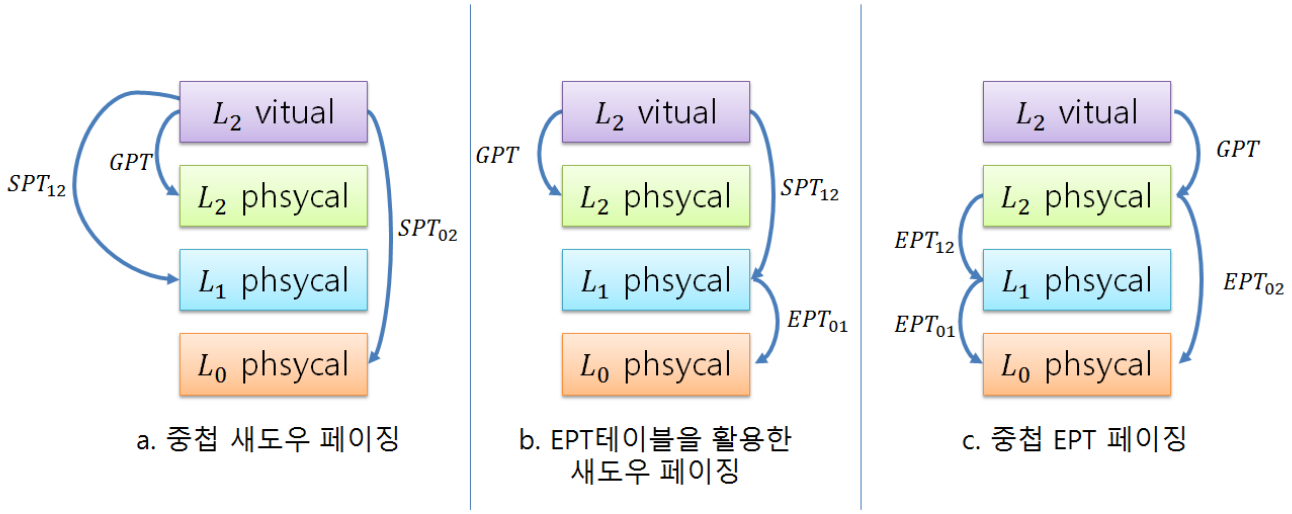
두 번째로 중첩 페이징 기법이다. 중첩 페이징 기법은 VM의 가상주소를 가상 물리주소로 변환하려고 할 때 이를 하드웨어 페이징을 이용하여 가상 물리주소를 호스트 물리주소로 변환하는 과정을 통해 페이지 테이블 접근에 대한 변환이 중첩의 형태로 이루어진다. 이 기법을 통하여 섀도우 페이징 기법이 가져야 하는 페이지 테이블 동기화 부담을 줄이고 메모리 접근을 위한 하이퍼바이저로의 트랩이 발생하지 않게 되었다. 하지만 문맥교환이 자주 일어나는 경우, TLB 플러시(flush)가 자주 발생하게 되면 오히려 오버헤드가 더 발생한다는 단점을 가지고 있다.

본 논문의 나머지 구성은 다음과 같다. 2장에서는 중첩 가상화가 어떤 기법을 사용하는지를 살펴본다. 3장 성능 평가에서는 중첩 가상화를 통하여 오버커밋 발생으로 인한 메모리 크기 별 성능 평가와 중첩 VM과 하이퍼바이저 VM의 성능차이를 보여준다. 4장 결론 및 향후연구에서는 실험을 통하여 중첩 가상화 시에 오버커밋이 발생할 경우의 메모리의 효율적 사용에 관해 논의하고 결론을 맺는다.

2. 중첩 가상화 환경에서 메모리 페이징 기법

중첩 가상화에서 사용하는 하드웨어 지원 기반의 페이징 기법은 중첩 페이지 테이블이다. 이 중첩 페이지 테이블은 일반적으로 섀도우 페이지 테이블의 단점, 즉 프로세스에 페이지 테이블을 중복해서 유지해야만 하는 점을 보완한 기술이라고 할 수 있다. 이는 오늘날 하드웨어의 페이지 테이블 가상화 지원으로 구현이 가능해졌다[2].

(그림 2)는 중첩 가상화 환경에서 메모리 페이징 접근 기법들이다. 여기서 L_0, L_1, L_2는 각각 호스트 머신, 하이퍼바이저 VM, 중첩 VM이다. (그림 2)(a)는 하드웨어 페이지 테이블 가상화를 지원하지 않을 때 섀도우 페이징을 이중으로 사용한 기법이다. L_0와 L_1사이에 하나의 섀도우 페이지를 추가하여 호스트 머신과 하이퍼바이저 VM과의 페이지를 매핑시킨다. 그리고 L_1과



(그림 2) 중첩 가상화에서의 메모리 페이징 기법

L_2 VM 사이에 또한 섀도우 페이징을 만들어 매핑을 시키는 방법이다. 이 때 L_2의 VM이 바로 L_0머신에 접근을 하려면 두 개의 섀도우 페이징을 사용한다. 이를 효율적으로 하기 위해 L_2과 L_0를 압축하여 직접 연결하는 섀도우 페이지로 매핑이 된다.

(그림 2)(b)는 현재 대부분의 중첩 VM에서 사용하는 방법으로 프로세서에서 중첩 가상화를 지원하여 EPT(확장 페이지 테이블)를 사용한 기법이다. L_0와 L_1 사이에 EPT로 페이지를 매핑 시켜 메모리 접근을 빠르게 한다. L_1과 L_2는 섀도우 페이지로 연결을 하여 메모리를 매핑 시킨다.

(그림 2)(c)는 최신 리눅스커널에서 사용 할 수 있는 방법으로 중첩으로 EPT를 사용하여 메모리 접근을 빠르게 하는 방법으로 L_0, L_1, L_2 사이사이에 EPT를 두어 접근이 가능하게 하였으며 이 때 L_2에서 L_0에 메모리에 접근하려 할 때 두 개의 EPT를 거칠 때 생기는 성능 감소를 줄이고자 두 EPT를 압축하여 바로 접근이 가능하도록 한다.

본 연구 실험 환경에서는 중첩 EPT 페이징 기법을 적용할 수 없어 (그림 2)(b)의 페이징 기법을 이용하여 실험을 하였다.

3. 성능 평가

중첩 VM과 하이퍼바이저 VM의 성능 비교를 위해 SPEC 2006 벤치마크 프로그램을 사용하였으며 오버커밋이 발생하는 환경을 만들기 위해 각각의 VM들의 메모리 크기를 증가시키며 성능을 측정하였다.

3.1 실험 환경

본 논문에서는 중첩 VM과 하이퍼바이저 VM의 성능 비교를 위해 호스트 머신에 하이퍼바이저의 중첩 가상화를 지원하는 리눅스 커널을 사용하여 VM들을 구성하였다. 실험을 위해 사용한 시스템의 사양은 <표 1>과 같으며, 각 VM의 사양은 <표 2>와 같다.

<표 1> 호스트 실험 환경

분류	사양
CPU	Intel(R) Core(TM) i5-2500 CPU @ 3.30GHz, quad core
RAM	8GB
HDD	SAMSUNG HD502HM 500GB
OS(Kernel)	Linux (x86_64, kernel version: 3.9.6-200)
Hypervisor	KVM

<표 2> VM 실험 환경

분류	Hypervisor virtual machine	Nested virtual machine
VCPU	4 cores	4 cores
RAM	8GB	5GB, 6GB, 7GB, 8GB. 9GB
HDD	100GB	20GB
Hypervisor	KVM	-

각 VM들의 성능 비교를 위한 실험은 SPEC 2006 벤치마크 프로그램을 중첩 VM과 하이퍼바이저 VM에서 수행하여 각각의 VM들의 성능을 여덟 번 측정 후 평균시간을 계산하여 성능을 비교하며 진행하였다.

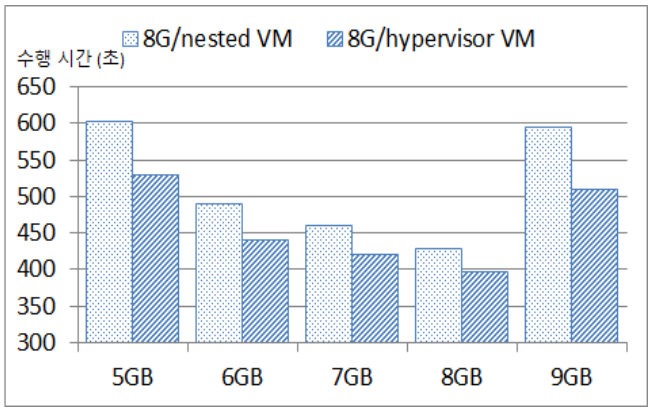
오버커밋 발생 시 중첩 VM과 하이퍼바이저 VM의 성능차이를 알기 위해서 호스트 머신에서 두 개의 하이퍼바이저 VM을 구성한 뒤 벤치마크 프로그램을 동시에 수행하여 실험을 하였으며, 하나의 하이퍼바이저 VM에서 두 개의 중첩 VM을 구성한 뒤 벤치마크 프로그램을 동시에 수행하여 실험하였다.

이 때 사용한 벤치마크 프로그램은 SPEC 2006 벤치마크 프로그램 중 mcf 프로그램이다. 이 mcf는 대중교통과 같은 차량에 대한 스케줄링을 다루는 벤치마크 프로그램으로 최적 스케줄을 찾는 성능을 벤치마크하는 프로그램이다[4].

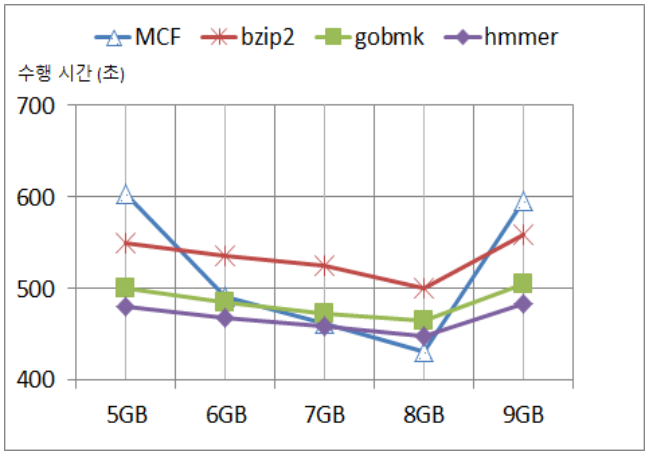
또한 오버커밋 시 가장 효율적으로 사용할 수 있는 메모리의 크기를 찾기 위한 실험은 두 개의 중첩 VM에서 메모리의 크기만을 변경하여 두 개의 VM에서 각각 벤치마크 프로그램을 동시에 수행하여 결과를 비교하는 것으로 진행하였다.

3.2 실험결과

(그림 3)은 메모리 오버커밋 발생 시 하이퍼바이저 VM과 중첩 VM의 성능 차이를 보여준다. 이 때 가로는



(그림 3) mcf의 중첩 VM와 하이퍼바이저 VM의 성능비교



(그림 4) 중첩 VM에서의 메모리 크기별 벤치마크 성능분석

메모리 크기의 증가를, 세로는 벤치마크 프로그램의 완료 시간을 나타낸다.

중첩 VM과 하이퍼바이저 VM의 평균 성능차이는 11.9%로 나타났으며 특히 호스트 머신의 메모리 크기와 동일해 졌을 때 성능차이가 가장 적게 나타났다(8.04%). 이는 하나의 하이퍼바이저 VM에서 두개의 VM이 실행되었다는 점을 감안하면 평균 11.9%의 성능 감소로 두 배의 작업을 수행하는 것을 알 수 있다.

(그림 4)는 각 벤치마크(mcf, bzip2, gobmk, hmmer))마다 나타나는 성능 시간을 비교하였다. 이때의 가로는 메모리의 크기, 세로는 벤치마크 프로그램의 완료시간을 의미한다.

5GB부터 나타낸 것은 하이퍼바이저 머신의 메모리가 8GB이며 동시적으로 작업을 수행하는 중첩 VM은 두개이므로 오버커밋이 발생하는 상황을 위해 5GB부터 실험하였다.

(그림 4)를 보게 되면 모든 벤치마크 프로그램이 8GB에서 최상의 성능을 보여주고 있고 8GB에서 5GB로 메모리의 크기가 줄어들수록 평균적으로 11.62% 성능저하를 보여주며 최대 13.86%의 성능저하가 나타났다. 수치적으로 오버커밋은 5GB부터 일어나지만 VM이 할당된 모든 메모리를 다 사용하지 않기 때문에 중첩 VM이 두 개가 실행되어도 무리 없이 실행되는 것으로 보이며, VM의 메모리 크기가 호스트 머신의 메모리 크기와 동일할 때(8GB) 최고의 성능을 나타내었다. (그림 4)의 성능평가에서 알 수 있듯이 9GB로 VM 메모리를 증가 시켰을 때 급격하게 성능이 저하된다. 이는 하나의 VM이 호스트 머신 메모리보다 큰 메모리를 할당받아 작업을 한다면 페이지 부재가 더 많이 발생하게 되어 성능이 저하되며 페이지 테이블의 동기화를 맞추기 위해 오버헤드가 발생한 것으로 여겨진다.

4. 결론 및 향후연구

본 논문에서는 중첩 가상화 환경에서 중첩 VM과 하이퍼바이저 VM의 성능을 비교하기 위해 두 VM 모델들 각각을 두개씩 실행을 하고 동시에 SPEC 2006 벤치마크 프로그램을 수행하였다. 또한 오버커밋이 발생하였을 때 중첩 VM을 통하여 메모리의 크기를 일정하게 증가 시키며 벤치마크 프로그램을 실행하여 성능을 비교하였다.

오버커밋 발생 시 SPEC 2006 벤치마크 프로그램으로 하이퍼바이저 VM과 중첩 VM의 성능 차이를 측정한 결과 평균 11.9%의 성능 차이가 났다. 이는 하나의 하이퍼바이저 VM에서 두 개의 VM이 수행된 점을 감안하면 11.9%의 차이로 두 배의 일을 수행한 것과 같다.

그리고 중첩 VM의 메모리 크기별로 오버커밋 발생 시 SPEC 2006 벤치마크 프로그램을 수행한 결과 각 중첩 VM의 메모리 크기가 호스트 머신의 메모리 크기와 가까워질수록 성능이 좋아지며, VM의 메모리 크기가 호스트 머신의 메모리 크기보다 크게 할당 되었을 경우에는 VM의 메모리 크기가 호스트 머신의 메모리크기와 동일하게 할당되었을 경우보다 성능이 떨어지는 결과를 보였다.

향후 성능 측정 시 하나의 하이퍼바이저 VM에서 세 개 이상의 머신에서 수행 하여 메모리 오버커밋을 하는 모든 중첩 VM에 대하여 최적의 성능을 나타내는 VM의 수와 메모리의 크기와의 상관관계를 측정할 계획이다.

참고문헌

[1] Michael R. Hines, Abel Gordon, Marcio Silva, Dilma da Silva, Kyung Dong Ryu, Muli Ben-Yehuda "Applications Know Best: Performance-Driven Memory Overcommit with Ginkgo", Cloud Engineering (IC2E), 2011 IEEE International Conference on, On page(s): 103 - 137, November 2011.

[2] M. Ben-Yehuda, M. D. Day, Z. Dubitzky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour. The Turtles Project: Design and Implementation of Nested Virtualization. In *USENIX OSDI'10*, October 2010.

[3] AMD-VTM Nested Paging White Paper: http://developer.amd.com/wordpress/media/2012/10/NPT-WP-1%201-final-TM.pdf

[4] SPEC CPU 2006: http://www.spec.org/cpu2006/CINT2006 (2008.08.24)