2015년 1월 전자공학회 논문지 제 52권 제 1 호
Journal of The Institute of Electronics and Information Engineers Vol. 52, NO. 1, January 2015 http://dx.doi.org/10.5573/ieie.2015.52.1.069

논문 2015-52-1-9

KVM 기반의 가상머신 자원 사용량 분석을 위한 VDI 실시간 모니터링 시스템 설계 및 구현

(VDI Real-Time Monitoring System for KVM-Based Virtual Machine Resource Usage Analysis)

김 태 훈*, 김 현 지*, 노 재 춘**

(Taehoon Kim, Hyeunjee Kim, and Jaechun No©)

요 약

차세대 단말기 보급과 초고속 네트워크의 발전으로 클라우드 컴퓨팅이 등장하면서 VDI(Virtual Desktop Infrastructure)가 핵심 기술로서 많은 주목을 받고 있다. VDI는 하나의 물리적인 컴퓨터에 하이퍼바이저를 설치하여 다수의 가상머신을 운영하는 기술이다. 현재 VDI 분야의 핵심 이슈는 가상머신과 물리적머신 간의 성능 차이를 줄이는 것이며, 이와 관련된 여러 연구들이 꾸준히 진행되고 있다. 본 논문에서는 서버와 가상머신의 실시간 자원 사용량과 정보를 수집하는 SETMOV(Supervised Real Time Monitoring on VDI)를 설계하고 구현하였으며, 이를 바탕으로 VDI 성능 최적화를 위해 가상머신의 자원 사용량을 분석하여 서버의 자원을 재분배하는 방법을 제시한다.

Abstract

Recently, due to the development of next-generation computing devices and high-performance network, VDI (Virtual Desktop Infrastructure) is receiving a great deal of attention from IT market as an essential part of cloud computing. VDI enables to host multiple, individual virtual machines that are provisioned from servers located at the data center by using hypervisor. One of the critical issues related to VDI is to reduce the performance difference between virtual machines and physical ones. In this paper, we present a real-time VM monitoring system, called SETMOV, that is able to collect the real-time resource usage information. We also present the performance results using iozone to verify SETMOV.

Keywords : VDI, 데스크톱 가상화, KVM, 가상화 모니터링 시스템, SETMOV

Ⅰ. 서 론

인터넷의 발전으로 새로운 패러다임인 클라우드 컴퓨팅(Cloud Computing)[1,2]이 등장하였으며, 가상화 기술 중 하나인 VDI(Virtual Desktop Infrastructure)[3]가 클라우드의 핵심기술로 주목을 받고 있다.

VDI는 물리적 서버 위에 다수의 가상머신을 관리하는 시스템으로, 하이퍼바이저를 이용하여 CPU와 메모리 등을 포함한 서버 자원들을 공유한다. 이를 바탕으

* 학생회원, ** 정회원, 세종대학교 컴퓨터공학과
(Dept. of Computer Engineering, Sejong University)
© Corresponding Author (E-mail: jano@sejong.ac.kr)
※ 이 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2014R1A2A2A01002614). 또한, 산업융합원천기술개발사업(No.10047118)의 지원을 받았음.
접수일자: 2014년07월10일, 수정일자: 2014년11월24일
게재확정: 2014년12월01일

로 서버에서 발생하는 하드웨어의 유휴상태를 낮추고 시스템의 활용도를 높일 수 있는 장점을 가지고 있다.

반면에, 가상머신에 할당되어 있는 서버의 자원량은 초기에 설정된 크기로 고정되어 있기 때문에, 사용 용도에 따라 가상머신의 자원 유휴상태가 발생할 수 있다. 예를 들어, 가상머신에서 CPU 활용도가 높은 애플리케이션이 실행될 경우, 초기에 설정된 자원량보다 높게 할당해 준다면 변경 전보다 빠른 실행이 가능할 수 있다.

본 논문은 VDI 성능 최적화를 위해 가상머신의 자원 사용량을 분석하여 서버의 자원을 재분배 하는 방법을 제시한다. 또한, 가상머신의 자원 사용량을 분석하기 위해 서버와 가상머신의 리소스 정보를 수집하는 실시간 모니터링 시스템인 SETMOV(Supervised Real-Time Monitoring on VDI)를 소개한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 VDI와 KVM(Kernel-Based Virtual Machine), Xen 하이퍼바이저에 대해 설명한다. 또한, 서버와 가상머신의 데이터 후킹(Hooking)에 필요한 proc 파일시스템과 libvirt에 대해 설명하고, 시스템 모니터링 툴을 소개한다. Ⅲ장에서는 VDI 실시간 모니터링 시스템인 SETMOV에 대해 설명한다. 마지막으로 Ⅳ장에서는 결론을 맺고 향후 연구 방향을 제시한다.

Ⅱ. 관련 연구

2.1 데스크톱 가상화

가상화는 컴퓨터 속에 컴퓨터를 뜻한다. 사용자는 가상화를 이용하여 하나의 컴퓨터에서 다양한 시스템들을 접할 수 있고 손쉽게 관리할 수 있다. 데스크톱 가상화는 하나의 서버에 다수의 가상머신을 실행하여 사용자에게 서비스를 제공하는 기술이다. 대표적인 데스크톱 가상화 기반의 오픈소스 솔루션으로 시트릭스의 Xen과 레드햇의 KVM이 있다. Xen과 KVM은 베어메탈(Bare Metal) 종류의 하이퍼바이저로 분류되며, Xen은 반가상화(Para-Virtualization) 기법을, KVM은 전가상화(Full-Virtualization) 기법을 사용하고 있다.

가상화의 핵심 기능인 하이퍼바이저는 서버의 자원을 다수의 가상머신이 공유할 수 있도록 관리하고 제어하는 역할을 수행한다. 하이퍼바이저의 종류는 [그림 1]과 같이 두 가지 타입으로 분류된다[4].

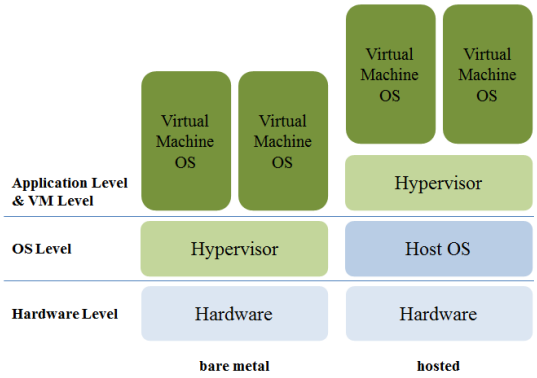


그림 1. 하이퍼바이저 아키텍처
Fig. 1. Hypervisor Architecture.

우선, 베어메탈 타입은 호스트 서버에 의존하지 않고 하이퍼바이저가 직접 하드웨어를 관리하는 구조로 되어 있으며, 그 결과 운영체제와 하이퍼바이저 간에 발생하는 오버헤드가 감소하여 고속의 시스템 성능을 산출할 수 있다. 베어메탈 타입은 크게 전가상화와 반가상화 기법으로 나뉜다. 전가상화는 하드웨어 가상화 기법인 VT(Virtualization Technology)[5]를 사용하여 가상머신이 자신의 범위 외에 자원을 침범할 수 없도록 제한한다. 반대로 반가상화 기법은 가상머신 운영체제 소스코드를 수정하여 하이퍼바이저의 성능을 높이도록 구성되어 있다[6]. 반가상화 기법에서 사용되는 하이퍼콜은 가상머신과 하이퍼바이저의 통신 API로서 리눅스의 유저모드와 커널모드를 연결해주는 시스템콜과 유사한 역할을 담당한다.

두 번째 타입인 호스티드 하이퍼바이저는 기존 운영체제의 애플리케이션과 동일한 레벨에서 실행되며, 가상머신은 그 다음 레벨에서 동작한다. 이러한 방식은 서버의 애플리케이션 레벨에서 하드웨어를 에뮬레이션 하기 때문에 오버헤드가 큰 단점이 있지만, 가상머신의 운영체제에 큰 제약이 없어서 다양한 시스템들을 운영할 수 있다.

2.2 KVM

KVM은 대표적인 오픈소스로서 전가상화 기법을 사용하는 하이퍼바이저이다. 레드햇은 리눅스에서 가상머신을 운영하기 위해 KVM을 도입하였으며, 현재 레드햇 오픈소스 프로젝트[7]로 연구가 진행되고 있다.

KVM은 [그림 2]와 같이 리눅스 커널 모듈에서 동작한다. 가상머신은 하드웨어 오픈소스 에뮬레이션 프로

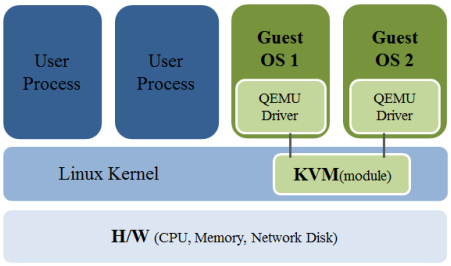


그림 2. KVM 구조도
Fig. 2. KVM Structure.

그램인 QEMU[8]를 일부 수정한 qemu-kvm을 통해 하드웨어를 공유한다. 이를 바탕으로 하이퍼바이저는 가상머신에게 다양한 프로세서와 I/O 하드웨어들을 지원하며, 가상머신은 에뮬레이션 된 하드웨어를 실제 하드웨어처럼 인식하여 사용한다[9~10]. KVM은 QEMU 이외에 디스크와 네트워크 입출력을 위한 virtio[11]등, 다양한 기술들을 활용하고 있다.

2.3 Xen

Xen은 대표적인 오픈소스 반가상화 하이퍼바이저로 시트릭스의 XenServer, XenDesktop 등 가상화 솔루션에서 사용하고 있다. Xen은 아마존 웹 서비스[12]가 구축될 때 채택된 만큼 안정성도 검증된 하이퍼바이저이며, 시트릭스는 XenCenter, XenMobile, XenApp 등 XenServer를 도와주는 패키지 소프트웨어도 공개하였다.

Xen은 [그림 3]과 같이 별도의 운영체제가 존재하지 않고 도메인 0이라는 특수 인터페이스를 통해 하이퍼바이저를 관리한다. 도메인 0에는 하드웨어와 통신하는 리눅스 드라이버가 존재하는데, 이는 가상머신과 하이퍼바이저의 통신 API인 하이퍼콜을 받아 하드웨어를 동작시키는 역할을 수행한다[13].

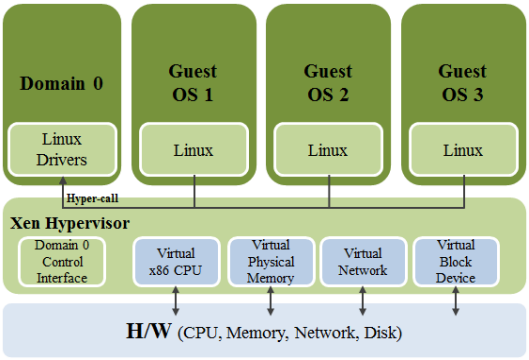


그림 3. Xen 구조도
Fig. 3. Xen Strucuture.

2.4 시스템 모니터링 툴

Virt-top[14]은 libvirt를 통해 가상머신의 실시간 자원 사용량을 측정할 수 있는 가상화 모니터링 툴이며, KVM 설치 시 패키지로 제공된다. 또한, 시스템 기반의 모니터링 툴 중 하나인 top은 리눅스에서 사용되는 대표적인 프로그램으로, CPU와 메모리 등 하드웨어의 실시간 사용량을 측정하는데 사용된다. 본 논문에서 구현한 SETMOV는 신뢰성을 높이기 위해 top과 virt-top에서 사용하는 자원 사용량 측정 공식을 참고하였다.

2.5 Proc 파일시스템

Proc 파일시스템은 운영체제의 각종 정보를 커널 모드가 아닌 유저모드에서 쉽게 접근할 수 있도록 지원하는 리눅스 기반의 가상 파일시스템이다. Proc 파일시스템은 기존의 파일시스템과 다르게 운영체제가 부팅되었을 때 메모리 안에서 자동 생성되며, 이에 대한 정보는 디스크에 저장되지 않는다[15]. SETMOV는 proc 파일시스템을 통해 서버의 정보를 수집한다.

2.6 Libvirt

Libvirt는 가상화를 도와주는 툴 킷이며, QEMU가 에뮬레이션 한 가상 디바이스를 관리할 수 있다[16]. 또한, C와 C++을 포함한 다양한 스크립터 언어들을 지원하며, 가상머신의 실시간 상태를 관리하는 virt-top과 virt-Manager는 libvirt API를 이용하여 구현된 프로그램이다. Libvirt는 GNU의 GPL 라이센스를 따라 오픈소스이며, KVM 뿐만 아니라 Xen, VMWare[17], Hyper-V[18], PowerVM[19]등 다양한 하이퍼바이저 들을 지원한다. SETMOV는 libvirt를 통해 가상머신의 정보를 수집한다.

Ⅲ. SETMOV

본 절에서는 SETMOV에 대한 전체 구조와 모듈간의 관계를 설명한다. 또한, 자원 사용량 측정을 위해 사용된 수식과 데이터베이스의 테이블 관계를 설명한다.

3.1 SETMOV 구조

[그림 4]는 SETMOV 구조를 나타내며, 구성 모듈들 간의 흐름도를 보여준다. SETMOV는 서버와 가상머신의 정보를 수집하는 리소스 수집 모듈과 수집된 데이터

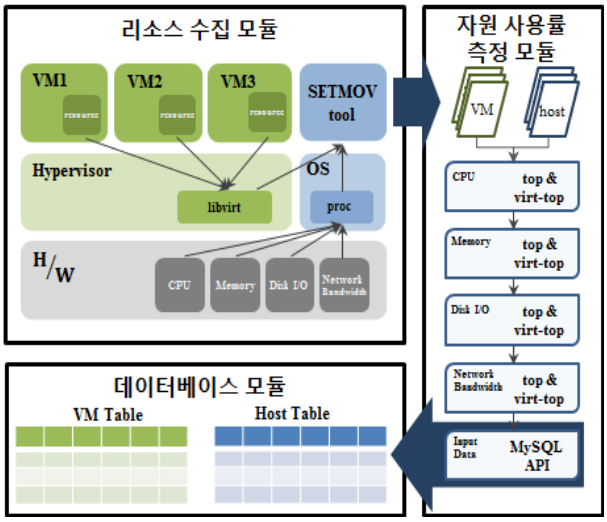


그림 4. SETMOV 구조도
Fig. 4. SETMOV Structure.

를 바탕으로 실시간 사용량을 측정.하는 사용량 측정 모듈, 마지막으로 분석을 위한 데이터베이스 모듈로 나뉘어진다. 각 모듈의 설명은 다음과 같다.

- 리소스 수집 모듈

  리소스 수집 모듈은 가상머신이 아닌 하이퍼바이저 레벨에서 동작하며, proc 파일시스템과 libvirt를 통해 서버와 가상머신의 시스템 정보를 수집한다.

- 자원 사용량 측정 모듈

  사용량 측정 모듈은 리소스 수집 모듈에서 읽어온 서버와 가상머신의 CPU, 메모리 등 하드웨어 정보들을 자원 사용량으로 가공한다. 또한, top과 virt-top의 공식을 참조하여, 높은 신뢰성을 가지도록 하였다.

- 데이터베이스 모듈

  데이터베이스 모듈은 가상머신 자원 사용량 분석을 위해 수집한 정보를 데이터베이스에 저장한다. 서버와 가상머신에 대한 정보와 각각의 자원 사용량을 나타내기 위해서 네 개의 테이블로 나누어 구성하였다.

3.2 시스템 정보 수집

SETMOV의 리소스 수집 모듈은 VDI 서버와 가상머신의 시스템 정보를 수집한다.

Host Server Info

Attribute	Structure	Source
Hostname	S_VdiData	gethostname()
IP Address	S_VdiData	getmyip()
CPU Usage	S_ProcCpuInfo	/proc/stat
Memory Usage	S_ProcMemInfo	/proc/meminfo
Disk Usage	S_ProcDiskInfo	/proc/diskstats
Network Usage	S_ProcNetInfo	/proc/net/dev

그림 5. 서버의 수집 정보
Fig. 5. Server Information.

Virtual Machine Info

Attribute	Source
VM Name	virDomainGetName()
VM UUID	virDomainGetUUIDString()
CPU Usage	virNodeGetInfo()
Memory Usage	virDomainMemoryStats()
Disk Usage	virDomainBlockStats()
Network Usage	virDomainInterfaceStats()

그림 6. 가상머신의 수집 정보
Fig. 6. Virtual Machine Information.

[그림 5]와 [그림 6]은 서버와 가상머신의 데이터 수집을 위해 설계한 함수와 구조체, 그리고 데이터를 정리한 표이다. SETMOV는 proc 파일시스템과 libvirt를 이용하여 CPU, 메모리, 디스크, 네트워크 정보를 실시간으로 수집한다. 리소스 수집 모듈은 서버에 대한 호스트 이름, IP 주소, CPU와 메모리 등 하드웨어 사용량을 수집하며, 가상머신에 대한 정보로 가상머신 이름, UUID, 하드웨어 사용량을 수집하여 저장한다.

3.3 실시간 시스템 자원 사용량 측정

SETMOV는 신뢰성있는 결과를 위해 모니터링 툴인 top과 virt-top의 공식을 참고하였다. 자원사용량 데이터는 서버와 가상머신 별로 각각 CPU, 메모리, 디스크, 네트워크로 나뉘어 측정된다.

우선, 서버의 CPU 사용량은 /proc/stat에 있는 CPU

상태 정보로 구할 수 있다.

```
struct S_ProcCpuInfo {
        __u128 user, nice, system, idle, iowait
        __u128 irq, softirq, steal, guest
}
```

S_ProcCpuInfo 구조체는 /proc/stat에서 수집한 서버의 CPU 상태 정보를 저장하며, 각 변수의 설명은 다음과 같다. User는 유저모드에서 기본 순위보다 높은 우선순위로 실행되는 시간을 뜻하며, nice는 유저모드에서 기본 순위보다 낮은 우선순위로 실행되는 시간을 뜻한다. System은 커널모드에서 실행되는 시간을 뜻하며, idle은 프로세서가 동작하지 않는 상태를 의미한다. Iowait는 디스크 I/O 요청으로 대기하는 시간이며, irq와 softirq는 인터럽트 핸들에 사용되는 시간을 뜻한다. Steal은 가상 CPU로 인해 비자발적 대기시간을 뜻하며, guest는 가상화에 사용된 시간을 뜻한다. 실시간 CPU 사용량은 다음 수식을 이용한다.

$$CPU\% = 100 - \frac{100}{total} \times (idle_{now} - idle_t) \quad (1)$$

수식(1)은 top에서 사용하는 실시간 CPU 사용량을 보여준다. Total은 CPU의 모든 상태 값을 더한 변수이며, now와 t는 idle의 현재 값과 t 초 전 값을 나타낸다.

가상머신의 CPU 사용량은 libvirt API에서 지원하는 virNodeGetInfo() 함수와 virDomainGetInfo() 함수를 이용하여 측정한다. 각각의 함수는 하이퍼바이저의 정보를 저장하는 virNodeInfo 구조체와 가상머신의 정보를 저장하는 virDomainInfo 구조체를 초기화한다.

$$CPU\% = 100 \times \frac{cpuTime_{now} - cpuTime_t}{t \times cores \times 10^9} \quad (2)$$

수식(2)는 virt-top에서 사용하는 실시간 CPU 사용량의 측정 공식이다. CpuTime은 virDomainInfo 구조체에 정의되어 있으며 가상머신의 CPU 사용 값을 나타낸다. 또한, cores는 virtNodeInfo 구조체에 정의되어 있으며 하이퍼바이저의 CPU 개수를 뜻한다. Now와 t는 현재 값과 t 초 전 값을 뜻한다.

두 번째로 서버의 메모리 사용량은 /proc/meminfo에 있는 메모리 상태 값으로 구할 수 있다.

```
struct S_ProcMemInfo {
        __u128 total, free
        __u128 buffers, cached
}
```

S_ProcMemInfo 구조체는 /proc/meminfo에서 수집한 서버의 메모리 상태 정보를 의미하며, 각 속성의 설명은 다음과 같다. Total은 가용 메모리의 전체 크기이며, free는 비 할당 공간을 뜻한다. Buffers와 cached는 커널에서 사용하기 위해 할당한 임시 공간이다. 실시간 메모리 사용량은 수식(3)을 이용한다.

$$Memory = 100 \times \frac{total - free}{total} \quad (3)$$

사용자는 S_ProcMemInfo의 값과 수식(3)을 통해 메모리의 실시간 사용량을 알 수 있다. 우선, total과 free를 이용하여 메모리 사용량을 구한 후, 전체 값에 백분율을 적용하여 측정한다.

가상머신의 메모리 사용량도 libvirt API에서 지원하는 virDomainMemoryStatStruct 구조체와 수식(3)으로 구할 수 있다. 그러나 애플리케이션에 따라 실시간으로 변경되기 때문에 초기에 할당된 전체 메모리 용량만을 알 수 있다.

세 번째로 서버의 디스크 사용량은 /proc/diskstats에 있는 디스크 상태 값으로 구할 수 있다.

```
struct S_ProcDiskInfo {
        char disk_name[20]
        __u128 r_compl, r_merge, r_sectr, r_milsc
        __u128 w_compl, w_merge
        __u128 w_sectr, w_milsc
        __u128 io_c_prc, io_milsc, io_w_milsc
}
```

S_ProcDiskInfo 구조체는 /proc/diskstats에서 수집한 서버의 디스크 상태 정보를 저장한다. Disk_name은 파티션의 정보가 담겨있는 /proc/partitions에서 블록 디바이스 드라이버의 이름을 추출하여 저장한다. 구조체의 각 변수 앞에 있는 r과 w는 read/write를 나타내며, compl와 merge는 완료된 요청과 병합된 요청의 수를 뜻한다. Sectr와 milsc은 I/O 작업이 완료된 섹터의 수와 밀리 세컨드 단위의 시간을 뜻한다. Io_c_prc 변수는 현재 I/O 진행 값을 나타내며, io_milsc는 io_c_prc의 소

요된 시간을 뜻한다. 마지막으로 Io_w_milsc는 다른 변수의 값들도 추가되어 io_milsc보다 정확한 시간을 가지고 있다.

$$Disk\,\% = (\mathrm{sec}tr_{now} - \mathrm{sec}tr_{t}) \times 512 \quad (4)$$

수식(4)는 서버와 가상머신의 실시간 디스크 사용량 측정 공식이다. 우선, I/O에 사용된 섹터의 개수를 가지고 있는 r_sectr와 w_sectr를 통해 t 초 전 사용한 sectr의 값을 구한다. 또한, byte 단위로 변경을 하기 위해 섹터의 크기인 512 Byte를 곱한다.

가상머신의 실시간 디스크 사용량도 같은 방법으로 구할 수 있다. Libvirt API의 virDomainBlockStats() 함수는 virDomainBlockStatsStruct 구조체를 초기화한다. 이 구조체를 구성하는 변수 중, 디스크의 I/O 값을 가지고 있는 rd_bytes와 wr_bytes 변수를 수식(4)에 대입하여 가상머신의 디스크 사용량을 구한다.

마지막으로 실시간 네트워크 사용량은 PPS(Packets Per Second)와 BPS(Bit Per Second)로 나뉘어 구할 수 있다. PPS는 초당 패킷 처리율을 뜻하며, BPS는 초당 크기를 나타낸다.

```
struct S_ProcNetInfo {
        char net_name[20]
        char net_hwaddress[20]
        __u128 r_byte, r_pack, r_err, r_drop
        __u128 r_fifo, r_frm, r_cmp, r_mult
        __u128 t_byte, t_pack, t_err, t_drop
        __u128 t_fifo, t_col, t_car, t_cmp
}
```

S_ProcNetInfo 구조체는 /proc/net/dev에서 수집한 서버의 네트워크 정보를 저장한다. Net_name과 net_hw-address 변수는 네트워크 인터페이스의 이름과 MAC 주소를 나타낸다. 각 변수의 r과 t는 receive/transmit를 뜻한다. Byte와 pack 변수는 해당하는 인터페이스에 통과된 byte 값과 pack의 개수를 저장하며, err와 drop은 에러와 드랍으로 처리된 패킷의 개수를 나타낸다.

Fifo와 frm 변수는 FIFO(First-in, First-out)의 에러 개수와 framing 에러를 뜻한다. Col과 cmp 변수는 인터페이스 충돌에러 개수와 receive/transmit의 압축 개수를 저장한다. 마지막으로 car과 mult 변수는 캐리어 분실 개수와 receive/transmit의 멀티캐스트 프레임 개수를 나타낸다. 네트워크 측정에 사용된 수식은 다음과 같다.

$$PPS = packet_{now} - packet_{t} \quad (5)$$

$$Packet\,Size = \frac{(byte_{now} - byte_{t}) \times 8}{PPS} + 12 + 7 + 1 \quad (6)$$

$$BPS = PPS \times Packet\,Size \quad (7)$$

수식(5)는 PPS를 구하는 공식으로 pack의 현재 값과 t 초 전 값을 뺄셈하여 구할 수 있다. 수식(6)은 패킷의 평균 크기를 구하는 공식으로 초당 byte의 값을 구한 후 bit 단위로 변환하여 PPS로 나눈다. 수식(6)의 분수를 계산한 결과 값은 OSI 7Layer에서 네트워크 계층의 패킷 크기이므로, 데이터링크의 캡슐화 과정에서 추가되는 헤더의 크기를 더해주어야 한다. 데이터링크에서 추가되는 헤더는 preamble(7bit), SFD(1bit), interpacket-gap(12bit)으로[20], 각 헤더의 크기만큼 추가로 더한다.

수식(7)은 BPS를 구하는 공식으로, 수식(5)와 수식(6)에서 구한 PPS와 패킷의 평균 크기를 곱하여 구할 수 있다.

가상머신의 네트워크 사용량은 libvirt에서 지원하는 virDomainInterfaceStatsStruct 구조체에 저장되며, virDomainInterfaceStats() 함수를 통해 구할 수 있다.

가상머신 네트워크 사용량은 서버의 네트워크 사용량 공식을 이용하여 구할 수 있다. 수식(5)의 pack과 수식(6)의 byte 값은 libvirt 구조체의 rx_byte, rx_packet, tx_byte, tx_packet 변수를 사용한다.

3.4 자원 사용량 저장을 위한 데이터베이스 테이블

SETMOV는 가상머신의 자원 사용량을 분석하기 위해 VDI의 리소스 정보를 데이터베이스에 저장한다. 본 연구는 오라클의 MySQL[21]를 사용하였으며, 테이블은 4개의 구성으로 나뉜다.

테이블은 서버와 가상머신 각각 두 개의 테이블로 구성하였다. [그림 7]의 테이블 T1과 T2는 서버의 정보와 실시간 자원 사용량을 저장한다. 테이블 T1은 HostIP를 기본키로 가지며, HostName과 서버에서 운영 중인 가상머신 개수를 저장하는 VMC(Virtual Machine Count)로 구성되어있다. 테이블 T2는 테이블 T1의 HostIP를 외래키로 가지며 CPU, 메모리, 디스크, 네트

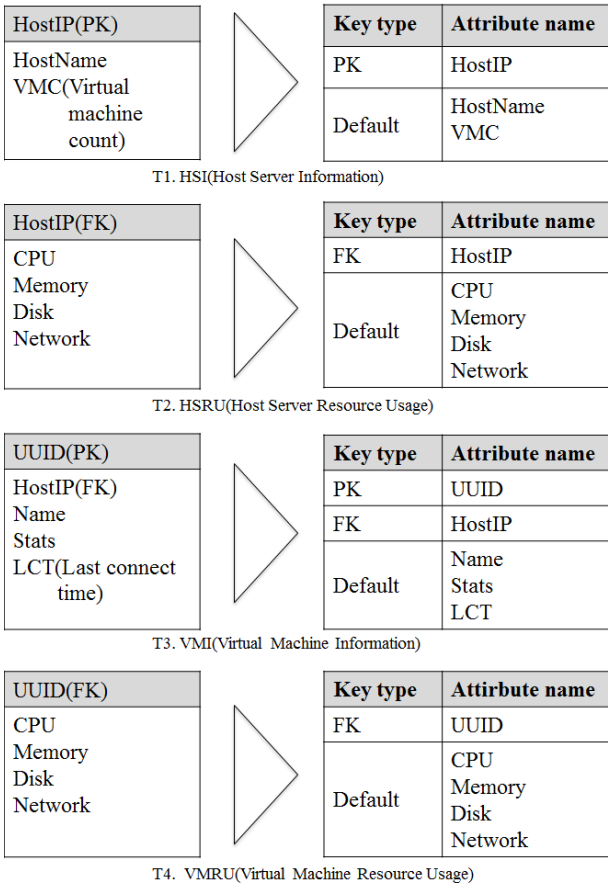


그림 7. 호스트 서버와 가상머신의 테이블 구성도
Fig. 7. Table Structure.

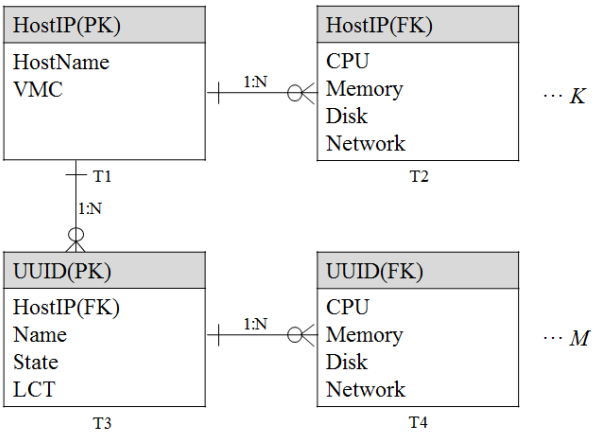


그림 8. 테이블 관계도
Fig. 8. Table Relationship.

워크 속성으로 구성되어 있다.

[그림 7]의 테이블 T3와 T4는 가상머신의 정보와 실시간 자원 사용량을 저장한다. 테이블 T3는 가상머신의 UUID를 기본키로 가지며, 테이블 T1의 HostIP를 외래키로 가진다. 그리고 가상머신의 이름, 상태, 마지막 연결 시간을 나타내는 LCT(Last Connect Time)로 구성되어있다. 테이블 T4는 테이블 T3의 UUID를 외래키로 가지며 CPU, 메모리, 디스크, 네트워크 속성으로 구성되어있다. 다음으로 각 테이블 사이의 관계를 설명한다.

[그림 7]에서 보여준 네 개의 테이블은 [그림 8]과 같은 관계를 가진다. 테이블 T1과 T2는 HostIP 속성을 외래키로 설정하여 1:N의 관계를 나타내며, 테이블 T2는 테이블 T1의 튜플 개수만큼 K개의 테이블로 연결되어 있다. 또한, 테이블 T1은 테이블 T3와 HostIP 속성을 통해 1:N의 관계를 가진다. 마지막으로 가상머신의 정보를 저장하는 테이블도, 테이블 T3의 UUID 속성을 외래키로 설정하여, 테이블 T1과 T2의 관계처럼 M개의 테이블로 연결되도록 구성하였다.

3.5 가상머신 자원 사용량 분석

본 절에서는 SETMOV를 실행하여 수집한 가상머신의 데이터를 그래프로 설명한다. 실험 환경은 [그림 9]와 같이 구성하였으며, 파일 압축과 다운로드를 통해 CPU, 디스크, 네트워크 사용량을 측정한다.

Host Server Info

Device Name	Performance
CPU	24core
Memory	32GB
Disk	14k rpm SAS 300G x2
Network	1GiB Ethernet
OS Version	CentOS 6.5 64bit
Software Version	Libvirt-0.10.2 qemu-kvm-0.12.1

Virtual Machine Info

Device Name	Performance
CPU	2core
Memory	4GB
Disk	qcow2 40GB
Network	vnet NAT
OS Version	CentOS 6.4
Benchmark	iozone-3.408

그림 9. 서버와 가상머신의 구성환경
Fig. 9. Test Environment.

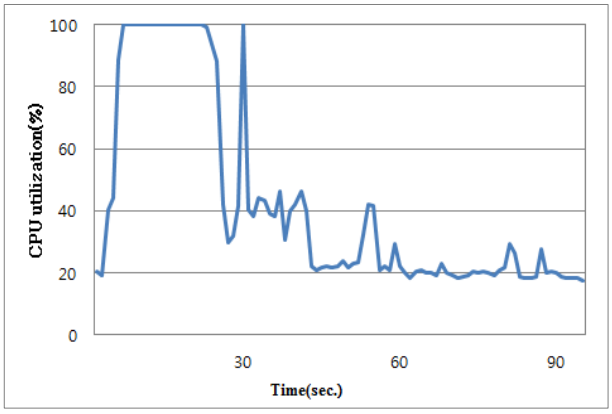


그림 10. CPU 사용량 그래프
Fig. 10. CPU Usage Graph.

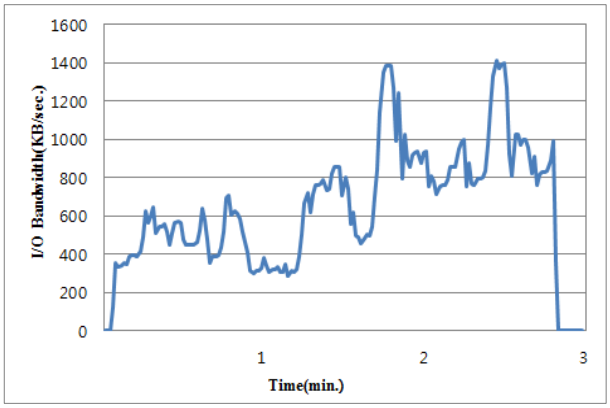


그림 11. 네트워크 사용량 그래프
Fig. 11. Network Usage Graph.

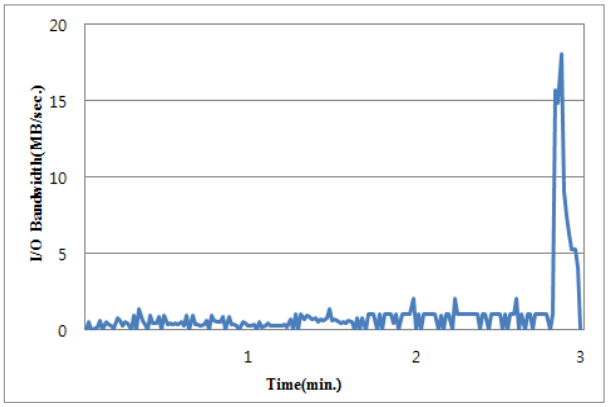


그림 12. 디스크 사용량 그래프
Fig. 12. Disk Usage Graph.

첫 번째는 CPU의 사용량을 측정하기 위해 100MB 크기의 파일을 압축하였다. [그림 10]의 가로축은 압축에 소요된 시간을 초 단위로 나타내며, 세로축은 CPU의 사용량을 %로 표기하였다. 파일 압축은 약 1분 35초의 시간이 걸렸으며, 약 28초까지는 CPU 사용량이 100%라는 것을 알 수 있다. 더불어, 45초 이후 평균 20%대를 유지하는 것을 알 수 있다.

두 번째는 인터넷에서 100MB 크기의 데이터를 다운받아 네트워크 사용량을 측정하였다. [그림 11]의 가로축은 다운로드에 소요된 시간을 분 단위로 나타내며, 세로축은 네트워크 속도를 뜻한다. 그래프를 보면 데이터를 다운받는데 걸린 시간은 약 3분가량 소요되었다는 것을 알 수 있으며, 최대 1.4MB/sec. 속도를 확인할 수 있다. 또한, 최저 350KB/sec., 평균 624KB/sec. 속도를 나타내는 것을 알 수 있다.

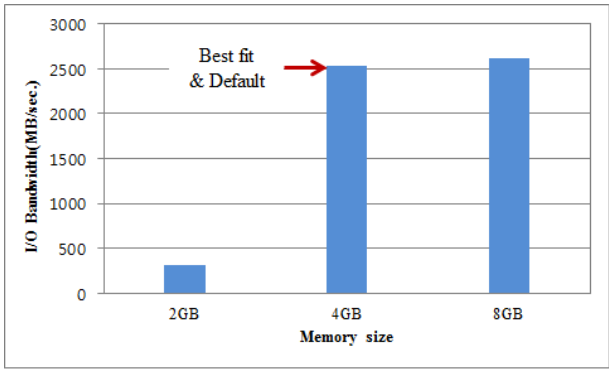
세 번째로 파일을 저장할 때 디스크의 사용량을 측정하였다. [그림 12]의 가로축은 데이터 저장에 소요된 시간을 나타내며, 세로축은 디스크 속도를 MB/sec.로 표기하였다. 디스크 사용량은 인터넷에서 다운받은 파일을 저장할 때 측정하였다. 다운받는 동안은 2MB/sec. 미만의 속도가 나타났으며, 다운로드가 끝난 시점부터 디스크의 사용량이 높아진 것을 알 수 있었다. 또한, 디스크의 최대 속도는 18MB/sec.이며, 데이터 저장에 소요된 시간은 약 8초임을 알 수 있다.

3.6 자원 사용량 기반의 가상머신 최적화

본 논문은 VDI 환경에서 성능 최적화를 위해 앞서 소개한 SETMOV를 이용하여 가상머신의 자원 사용량을 수집하고 데이터를 분석하였다. 연구에 사용된 호스트 서버와 가상머신의 구성 환경은 [그림 9]와 같으며, 메모리 크기에 따른 디스크 사용량을 분석하였다.

[그림 13]은 가상머신의 메모리 크기별로 I/O 벤치마크 툴인 iozone을 이용하여 4KB부터 2GB까지의 읽기 성능을 측정한 결과이다. 디스크 사용량 분석을 위해 메모리 크기를 변경한 이유는, 메모리에 저장되는 버퍼의 크기가 디스크 성능에 영향을 미치기 때문이다. 위 실험에서는 2GB, 4GB, 8GB의 메모리 크기로 디스크 사용량을 측정하였다.

우선, 4GB 메모리를 사용한 읽기 성능은 2GB 데이터 기준으로 2473.9 MB/sec.가 나왔으며, SETMOV가 수집한 전체 평균은 772.9 MB/sec.로 집계되었다. 또한, 메모리 크기를 2GB와 8GB로 변경하여 테스트한 결과, 각각 107.1 MB/sec.와 2548.4 MB/sec.로 큰 차이를 보였다. 그러나 4GB와 8GB를 비교했을 때 큰 차이는 보이지 않았다.



Memory	MB/sec.	Disk Usage
2GB	107.1	674.9 MB/sec.
4GB	2473.9	772.9 MB/sec.
8GB	2548.4	829.7 MB/sec.

그림 13. 디스크 사용량 분석
Fig. 13. Disk Usage Analysis.

위 실험을 통해 초기 설정한 4GB의 메모리 크기가 최적화된 상태임을 알 수 있으며, 메모리 크기를 높여도 성능 차이에 비해 자원의 유휴상태가 많다는 것을 알 수 있다.

Ⅳ. 결 론

본 논문은 VDI 환경에서 가상머신의 자원 사용량을 분석하여, 가상머신의 성능을 최적화하는 방법을 제시하였다. 또한, 서버와 가상머신의 정보를 수집하는 실시간 모니터링 시스템인 SETMOV를 소개하였고, 데이터를 수집하여 가상머신의 자원 사용량을 분석하였다.

그 결과, 메모리 크기에 따른 디스크의 성능 실험을 보았을 때, 일정크기가 넘어서는 경우 메모리의 크기가 디스크의 성능에 큰 영향을 주지 않는 것을 확인할 수 있었다. 따라서, 서버의 자원을 소수의 가상머신에 많이 할당하는 것보다, 사용 용도를 분석하여 최적화된 환경으로 구축하는 것이 효율적이라는 것을 알 수 있었다.

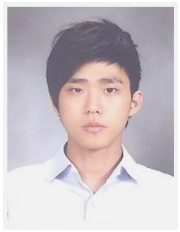
향후 연구로는 SETMOV를 이용하여 사용자 운영 패턴 기반의 실시간 마이그레이션(Migration) 연구와 데이터 I/O 모니터링을 통한 가상머신 캐시 연구를 진행할 것이며, 더 나아가 SETMOV를 통해 가상화 성능에 영향을 주는 변수를 찾아 높은 시스템 성능을 나타내도록 할 것이다.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, “A view of cloud computing.” Communications of the ACM, Vol. 53, no. 4, pp. 50-58, April 2010.
[2] Cho. Yoohee, Seo. Youngil and Kim. Yihan, “Consideration of Network as a Service.” Conference on Electronics and Information Communications(CEIC), pp. 157-160, October 2010.
[3] Wongyu Hong, Jihyeong Song, Minseon Kim, Juhui Kim, Gyeongho Lee, Gwangil Kang, Hyeonseok Shim, Chunho Son and Seokhyeong Cho, “Future Direction for Virtual Desktop Infrastructure (VDI).” OSIA Standards & Technology Review, Vol. 25, no. 2, pp. 8-21, July 2012.
[4] Wikipedia, “Hypervisor” http://en.wikipedia.org/wiki/Hypervisor
[5] D. Abramson, “Intel Virtualization Technology for Directed I/O.” Intel Technology Journal, Vol. 10, no. 3, 2006.
[6] Inc NI, “Virtualization Technology Under the Hood.” http://www.ni.com/white-paper/8709/ko
[7] A Redhat Emerging Technology, http://et.redhat.com
[8] F. Bellard, “QEMU, a Fast and Portable Dynamic Translator.” USENIX Annual Technical Conference, FREENIX Track, pp. 41-46, April 2005.
[9] A. Kivity, Y. Kamay, D. Laor, U. Lublin, A. Liguori, “kvm: the Linux virtual machine monitor.” Proceedings of the Linux Symposium, Vol. 1, pp. 225-230, July 2007.
[10] I. Habib, “Virtualization with kvm.” Linux Journal, Vol. 166, no. 8, 2008.
[11] R. Russell, “virtio: towards a de-facto standard for virtual I/O devices.” ACM SIGOPS Operating Systems Review, Vol. 42, no. 5, pp. 95-103, July 2008.
[12] Amazon Web Service, http://aws.amazon.com
[13] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, “Xen and the art of virtualization.” ACM SIGOPS Operating Systems Review, Vol. 37, no. 5, pp. 164-177, December 2003.
[14] virt-top, http://people.redhat.com/~rjones/virt-top
[15] T. Bowden, B. Bauer, J. Nerin, S. Feng and S. Seibold, “The/proc filesystem.” Linux Kernel

Documentation, 2000.
[16] Libvirt, http://www.libvirt.org
[17] M. Rosenblum, “VMware’s virtual platform ™.” In Proceedings of hot chips, Vol. 1999, pp. 185-196, August 1999,.
[18] A. Velte and T. Velte, “Microsoft virtualization with Hyper-V” Inc McGraw-Hill, 2009.
[19] Powervm, http://www.ibm.com/systems/kr/power/software/virtualization/
[20] Wikipedia, “Ehternet Frame” available at http://en.wikipedia.org/wiki/Ethernet_frame
[21] MYSQL, A. B. “MySQL” 2001.

저 자 소 개



김 태 훈(학생회원)
2012년 국가평생교육진흥원 컴퓨터공학과 학사 졸업.
2014년 세종대학교 컴퓨터공학과 석사 졸업.
2014년～현재 세종대학교 컴퓨터공학과 박사과정
<주관심분야 : 리눅스, 파일시스템, 분산처리, 가상화, 빅데이터>

김 현 지(학생회원)
2013년 세종대학교 컴퓨터공학과 학사 졸업.
2014년～현재 세종대학교 컴퓨터공학과 석사과정
<주관심분야 : 리눅스, 디바이스 드라이버, 가상화>

노 재 춘(정회원)-교신저자
1985년 이화여자대학교 전산과 학사 졸업.
1993년 Western Illinois Univ. 전산과 석사 졸업.
1999년 Syracuse Univ. 전산과 박사 졸업.
1999년～2001년 Argonne National Lab. 연구원
2001년～2003년 Hewlett Packard HPDC Lab. 연구원
2003년～현재 세종대학교 컴퓨터공학과 정교수
<주관심분야 : 파일시스템, 저장장치 시스템, 가상화, 빅데이터, 분산처리, 모바일네트워크>