JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 18, No. 11 : 2670~2677 Nov. 2014

웹 서버 구성을 통한 가상머신과 컨테이너 방식 비교 분석

배유미[1] · 정성재[2] · 소우영[3*]

Comparative Analysis of the Virtual Machine and Containers Methods through the Web Server Configuration

Yu-mi Bae[1] · Sung-jae Jung[2] · Woo-young Soh[3*]

[1]Research Institute, Sky Computing C&S, Inc., Daejeon 302-845, Korea
[2]Research Institute, Sky Computing C&S, Inc., Daejeon 302-845, Korea
[3*]Department of Computer Engineering, Hannam University, Daejeon 306-791, Korea

요 약

Xen, KVM 등과 같은 하이퍼바이저를 이용한 가상 머신 생성 기술이 클라우드 인프라 구성에 주로 사용되고 있다. 이 기술은 기존의 운영 방식에 비해 자원을 할당하고 관리하는 측면에서는 효율적이다. 그러나, 이 기술은 가상 머신 생성시에 높은 자원의 사용량이 요구되고 할당된 자원을 사용하지 않는 경우에는 또 다른 자원의 낭비를 초래한다. 이러한 문제점을 해결하기 위한 방법이 컨테이너 기반의 Docker이다. 본 논문은 가상 머신 방식과 컨테이너 방식을 비교하여 웹 서버 구축 기술로 Docker와 같은 컨테이너 방식이 효율적임을 보여준다. 특히, 웹 서버나 프로그램 개발 환경과 같이 데이터를 데이터베이스나 스토리지 등에 저장하는 경우에는 유용한 것으로 분석되었다. 앞으로의 클라우드 환경에서 Docker와 같은 컨테이너 방식이 자원의 효율성과 관리의 편의성을 더욱 높일 수 있을 것으로 기대된다.

ABSTRACT

The technique of virtual machine construction using hypervisor such as Xen and KVM is mainly used for implementation of cloud computing infrastructure. This technique is efficient in allocating and managing resources compared to the existing operation methods. However it requires high resource usage when constructing virtual machines and results in wasting of resources when not using the allocated resources. Docker is a technique based on the container method to resolve such a problem. This paper shows the container method such as Docker is efficient as a web construction technique by comparing virtual machine method to container method. It is shown to be especially useful when storing data into DB or storage devices in such environments of web server or program development. In the upcoming cloud computing environment the container method such as Docker is expected to improve the resource efficiency and the convenience of management.

키워드 : 컨테이너, 도커, 케이브이엠, 리눅스, 가상머신

Key word : Containers, Docker, KVM, Linux, Virtual Machine

접수일자 : 2014. 07. 29 심사완료일자 : 2014. 10. 16 게재확정일자 : 2014. 10. 29

* **Corresponding Author** Woo-young Soh(E-mail:wsoh@hnu.kr, Tel:+82-42-629-7657)
Department of Computer Engineering, Hannam University, Daejeon 306-791, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2014.18.11.2670 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

IT 인프라 구축시에 클라우드 컴퓨팅(Cloud Computing) 및 빅데이터(Big Data) 기술의 활용이 보편화되고 있다. 이 중 클라우드 컴퓨팅 기술은 IT 자원을 서로 공유하고 유휴 자원을 효율적으로 이용할 수 있다는 장점으로 인해 서버나 시스템 구성시 필수적으로 이용되고 있다. 클라우드 컴퓨팅의 핵심 기반 기술로 가상화(Virtualization) 기술을 꼽을 수 있는데, 서버 분야에서 많이 사용되는 공개형 서버 가상화 기술에는 하이퍼바이저(Hypervisor)기반의 Xen, KVM, Virtualbox 등이 있다. 하이퍼바이저 기반의 서버 가상화 기술은 물리적 서버 위에 운영체제(이하 호스트 OS)를 설치하고, 그 위에 하이퍼바이저를 기반으로 자원을 분할하여 가상 머신을 생성한 뒤에 또 다시 운영체제(이하 게스트 OS)를 설치하고 응용 프로그램을 구동하는 방식이다. 이러한 방식은 하나의 물리적 시스템에 독자적으로 운영 가능한 다수의 서버들을 운영할 수 있다는 장점은 있으나, 호스트 OS와 게스트 OS가 동일한 운영체제로 작동중인 경우에는 자원의 낭비가 크다. 하이퍼바이저 방식의 비효율성을 극복하기 위해 등장한 방법이 리눅스 컨테이너(Linux Containers) 방식의 도커(Docker)이다[1]. 도커는 물리적 서버 위에 호스트 OS를 설치하고 하이퍼바이저없이 응용프로그램, 바이너리 및 라이브러리 파일로만 구성된 도커 엔진 컨테이너라는 이름으로 구동된다. 호스트 OS에서는 사용자 공간에 격리된 하나의 프로세스로 구동되기 때문에 가상 머신의 이점을 누리면서 가볍고, 빠르고, 이동성도 뛰어난 하나의 시스템을 구성할 수 있다.

본 논문에서는 리눅스 기반 오픈 소스 서버 가상화 기술 중에 가장 효율성이 좋은 KVM(Kernel based Virtual Machine)[2,3]와 컨테이너 방식의 도커(Docker)를 이용하여 웹 서버를 구성하고, 이 두 가지 방식에 대해 비교 분석하고자 한다.

Ⅱ. 관련연구

현재 서버 분야에 많이 사용되는 가상화 기술은 가상화 대상 개체에 따라 크게 하드웨어 레벨(Hardware-level) 가상화와 운영체제 레벨(OS-level) 가상화로 나눌 수 있다[4].

2.1. 하드웨어 레벨 가상화

하드웨어 레벨 가상화는 가상화를 제공하는 가상화 계층(Virtualization Layer)이 물리적인 하드웨어 또는 호스트 운영체제 위에 위치하여 가상화 계층에 의해 생성된 가상 머신에 게스트 운영체제를 생성한다. 이 기술의 가장 큰 특징은 가상화 계층 내의 주요 구성 요소인 Virtual Machine Monitor(이하 VMM)이다. VMM은 물리적인 하드웨어를 가상머신의 하드웨어에 매핑시키고 게스트 운영체제의 자원 활용을 관리하는 업무를 수행한다. 하드웨어 레벨의 가상화 기술은 VMM의 구성 위치에 따라 Bare-Metal/Hypervisor (전가상화), Para-Virtualization(반가상화), 호스트기반 가상화로 나눈다.

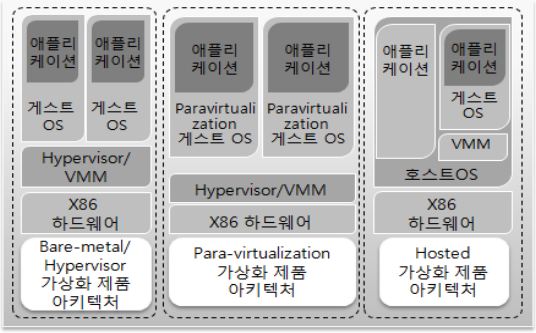

그림 1. 하드웨어 레벨 가상화
Fig. 1 Hardware-level Virtualization

2.1.1. 전가상화

전가상화 및 반가상화는 가상 머신의 하드웨어를 에뮬레이팅 하는 대신에 하이퍼바이저(hypervisor)라는 제어 프로그램을 통해 호스트의 하드웨어 자원을 가상 머신의 여러 운영체제가 나누어 사용하도록 하는 기법이다. 그 중 전가상화는 가상화 소프트웨어가 수정하지 않은 게스트 운영체제를 100% 완전 가상화 환경 위에서 제공하는 기술로 범용적인 x86 가상화 하드웨어(CPU, 메모리, 디스크, I/O 장치 등)를 VMM위에 구성한다. 게스트 운영체제 측면에서 별도의 수정 및 간섭 없이 완벽하게 물리적인 하드웨어 자원의 접근 및 이용이 가능하다. Bare-Metal/Hypervisor 기법을 이용한 제품에는 HP Integrity VM, IBM zSerises z/VM, VMware ESX Server, Xen(with HW assist) 등이 있다.

2.1.2. 반가상화

반가상화의 기본적인 구성 방법은 전가상화 방법인 Bare-Metal /Hypervisor 동일하나 게스트 운영체제 측면에서 VMM과의 원활한 통신을 위해 게스트 운영체제의 커널 일부분을 수정하여 적용한다. 이 방법은 일부 가상화 하드웨어 자원에 대해 필요에 따라 직접적으로 물리적 하드웨어 접근을 일부 허용한다. 반가상화는 전가상화에 비해 기존 시스템에 가까운 성능을 나타내기 위해 지원하지만, 게스트 운영체제의 커널 일부분을 수정해야 한다는 부분이 단점으로 지적된다. 반가상화 기법을 이용한 제품으로는 오픈 소스인 Xen, Citrix사의 XenServer 등이 있다.

2.1.3. 호스트 기반 가상화

VMM이 호스트 운영체제 위에 설치되어 가상머신을 생성하는 구조로 게스트 운영체제는 호스트 운영체제가 인식한 하드웨어를 가상 하드웨어로 재구성하여 사용하는 방법이다. 이 방법은 하드웨어 레벨 가상화 기법중에서 호스트 운영체제가 설치된 환경 위에 구동되어 성능은 물론 자원관리 능력 측면에서 제약사항이 많은 편이나 일반인들이 사용하기에는 쉬운 장점이 있다. 이 기술이 적용된 제품으로는 Microsoft의 Virtual Server 및 Virtual PC, VMware Server, VMware Workstation, VMware Player 등이 있다.

2.2. 운영체제 레벨 가상화

운영체제 레벨 가상화는 하나의 CPU에 하나의 운영체제만 수행되고, 가상화 계층(Virtualization Layer)이 호스트 운영체제 윗부분에 존재하여 가상화 환경을 제공하는 방법을 말한다. 단일 호스트 운영체제 위에 다중 애플리케이션 전용 컨테이너를 생성하여 애플리케이션 단위의 가상화 환경을 제공한다. 각각의 애플리케이션 컨테이너 안에는 가상 운영체제와 네트워크, 프로세서 등을 생성한다. 이 방법은 크게 두 가지 방식으로 분리하는 데 하나는 Containers방식이고 또 다른 하나는 Hardware Emulator방식이다.

Container방식은 가상화 계층이 호스트 운영체제 내에 임베디드 형태로 구성되고, 각각의 애플리케이션별로 추가적인 컨테이너를 생성하는 방식이다. 각각의 컨테이너안에 생성된 가상 운영체제와 호스트 운영체제에 있는 공통 커널을 공유하고 사용한다.

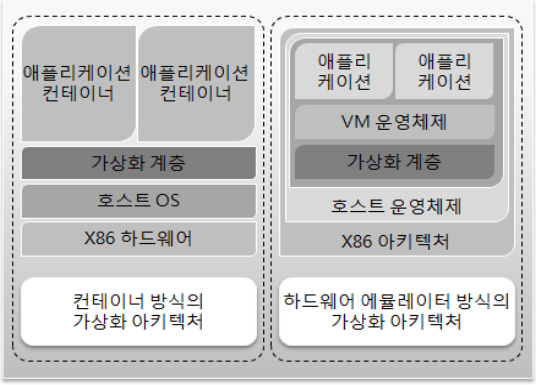

그림 2. 운영체제 레벨 가상화
Fig. 2 OS-level Virtualization

대표적인 제품이 오픈 소스인 OpenVZ, Parrells사(구 SWSoft)의 Virtuozzo, FreeBSD Jail, HP의 Secure Resource Partitions, Sun의 Solaris Containers Zones, User-Mode Linux, Docker 등이 있다. 또 다른 방법인 하드웨어 에뮬레이터방식은 애플리케이션 중심의 가상 운영체제를 생성하는 과정에서 소프트웨어적으로 하드웨어를 가상 에뮬레이팅하는 방식이다. 대표적인 제품으로는 마이크로소프트사의 Virtual PC for Mac, QEMU, Bochs 등이 있다. 운영체제 레벨의 가상화 기법은 기본적으로 단일 호스트 운영체제 위에 구성되는 애플리케이션 중심의 가상 운영체제를 생성한다. 각각의 가상 운영체제는 호스트 운영체제와 커널 구성 측면의 공유로 운영체제를 사용함으로써, 다음에 열거된 하드웨어 레벨의 전가상화 제품에 비해 성능적으로 많은 제약사항을 가지고 있다. 또한 지원 가능 애플리케이션이나 다양한 가상 운영체제 구성 역시 한계성을 가지고 있다.

Ⅲ. KVM 및 Docker를 이용한 웹 서버 구축

3.1. 구성 방안

한 대의 물리적 서버에 비용 부담이 없는 공개 소프트웨어 리눅스인 CentOS 7 버전을 설치하고, 가장 최소한의 자원 할당을 통해 웹 서버를 구축하여 테스트한다[5]. 가상머신 및 컨테이너 안에 실행되는 운영체제의 기준은 현재 가장 많아 사용되고 있는 CentOS 6.5로 구

성하고, 웹 서버 프로그램은 아파치 2.2 버전으로 구성한다. KVM 기반의 가상 머신은 CentOS 6.5를 설치하기 위해 최소한의 하드웨어 자원만을 할당하는데, 권장사항을 기준으로 메모리는 1GB를 할당하고 하드디스크 용량은 10GB를 할당한다. 간단한 웹 페이지를 작성하고 웹 서버를 동작시키는 과정을 통해 설치방법, 운영 방법, 자원 사용량 등에 대해 분석한다.

하드웨어 구성 환경은 x86_64 계열의 하드웨어 이용하고 세부적인 사양은 표 1과 같다.

표 1. 시스템 하드웨어 구성
Table. 1 The system Hardware configuration

CPU	Intel Xeon 5130 CPU 2.0GHz 64bit Proccessor
Main Board	Intel S5000VSA-SCSI
Memory	4GB(DDR2 667Mhz ECC)
HDD	Samsung 300GB Serial-ATA(Model: HD300LJ)
NIC	Intel 82563EB Dual Gigabit Ethernet
VGA	ATI ESI1000 SVGA PCI

3.2. KVM을 이용한 웹 서버 구성

3.2.1. 설치 개요

CentOS 7 버전에서 KVM을 사용하기 위해서는 처음 리눅스 설치시에 나타나는 소프트웨어 선택(Software Selection) 메뉴에서 가상 호스트(Virtualization Host)를 선택하면 손쉽게 이용 가능하다. 만약, 다른 메뉴를 선택했다면 리눅스 설치 후에 yum 명령을 이용하여 qemu-kvm, virt-manager 등을 설치하면 이용 가능하다. 설치 후에 가상 머신 관리자(virt-manager)를 이용하여 가상 머신을 생성한 뒤에 CentOS 6.5를 설치하고, yum 명령을 이용하여 아파치 웹 서버인 httpd를 설치한다.

3.2.2. 자원 할당

KVM은 가상머신을 이용하는 하이퍼바이저형이므로 반드시 CPU, RAM, HDD 등의 자원 할당이 필요하다. 게스트 OS로 사용할 CentOS 6.5 버전을 웹서버로 운영하기 위해서는 최소 512MB ~ 1GB의 RAM 용량이 필요하고, 8~10GB 정도의 HDD 용량이 요구된다. 가상머신에 할당되는 CPU의 개수는 물리적 CPU의 개수나 코어(Core)수에 따라 좌우된다. 초기의 가상 머신 관리자는 물리적인 CPU나 CPU의 코어수에 상관없이 가상머신을 생성할 수 있었으나, 최근에는 최적의 성능을 발휘할 목적으로 물리적인 CPU나 CPU의 코어수에 따라 제한되고 있다.

3.2.3. 네트워크 및 기타 설정

가상머신을 서버로 사용하기 위해서는 먼저 가상 이더넷 관련 장치를 설정하고, IP(Internet Protocol) 주소를 할당해야 한다. 공인 IP 주소를 할당해도 되고, 공인 IP 주소가 없다면 사설(Private) IP 주소를 할당해도 된다. 사설 IP 주소를 할당하는 절차를 끝냈다면 추가적으로 호스트 OS의 80번 포트로 들어오는 웹 서비스 요청을 게스트 OS의 80번 포트로 포워딩(Forwarding)하는 절차가 요구된다. 이 때 iptables라는 패킷필터링(Packet Filtering) 및 방화벽(Firewall) 역할을 해주는 프로그램을 이용해서 NAT(Network Address Translation) 설정해줘야 한다[6]. 만약 호스트 OS의 공인 IP주소가 203.247.40.245번이고, 게스트 OS의 사설 IP주소가 192.168.0.10번이라면 다음의 표 2 및 표 3과 같이 DNAT(Destination NAT) 및 SNAT(Source NAT)명령을 내려야 한다[7].

표 2. iptables를 이용한 웹 포트 포워딩 설정
Table. 2 Web Port Forward using iptables

```
# iptables -t nat -A PREROUTING -p tcp -d 203.247.40.245
--dport 80 -j DNAT --to 192.168.0.10
```

표 3. iptables를 이용한 SNAT 설정
Table. 3 SNAT setup using iptables

```
# iptables -t nat -A POSTROUTING -j SNAT --to
203.247.40.245
```

3.3. Docker를 이용한 웹 서버 구성

3.3.1. 설치 개요

CentOS 7 버전에서 Docker를 사용하려면 'yum install docker*'라는 명령만 수행하면 손쉽게 설치가능하다. Docker는 CentOS 6.5 기반의 웹 서버를 운영하기 위해 KVM처럼 별도의 운영체제 설치가 필요하지 않다. Docker 사이트에서 제공되는 Docker Container를 검색하여 설치만 해주면 된다[8]. 컨테이너 이미지 검색을 통해 최소화된 CentOS 6.5의 컨테이너 이미지를 설치할 수도 있고, 다른 사용자가 공유한 컨테이너 이미지를 선택하여 설치할 수도 있다. 다음의 표 4와 같은

단계를 거쳐 CentOS 6.5용 이미지 컨테이너를 설치할 수 있다.

표 4. Docker 컨테이너 설치 과정
Table. 4 The installation process Docker Container

과정	명령 예
이미지 검색	# docker search centos
이미지 설치	# docker pull posein/centos
이미지 확인	# docker images

제공되는 컨테이너 이미지는 서버 운영에 필요한 최소한의 응용프로그램, 바이너리 및 라이브러리로만 구성되어 있으므로 해당 컨테이너 이미지에 접속해서 사용할 웹 서버 프로그램인 아파치를 설치해야 한다. 관련 프로그램 설치과정은 다음의 표 5와 같다.

표 5. Docker 컨테이너안에 프로그램 설치 과정
Table. 5 Program installation in Docker Container

과정	명령 예
이미지에 접속	# docker run –i –t d592b9a589a6 /bin/bash
프로그램 설치	# yum install httpd

3.3.2. 자원 할당

Docker는 KVM과 다르게 별도의 자원 할당이 필요 없다. Docker에서 운영하는 Container는 기본적인 동작 방식이 가상머신처럼 보이지만, 호스트 OS에서는 프로세스(Process)로 관리된다. 따라서, 다른 응용프로그램들과 같이 실행시에 CPU 및 RAM 등의 하드웨어 자원을 할당하고 종료시에 해당 자원을 회수한다.

3.3.3. 네트워크 및 기타 설정

Container 운영중인 경우에는 어떠한 네트워크 설정이나 변경이 필요하지 않다. 즉, Container의 네트워크에 대한 어떠한 정보도 알지 못해도 상관없다. 표 6과 같이 단지 호스트 OS의 80번 포트로 들어오는 웹 서비스 요청을 해당 컨테이너의 80번 포트로 보내는 docker 명령만 실행하면 된다.

표 6. 웹 포트 포워딩하는 Docker 명령
Table. 6 Docker's Web Port forwarding command

docker run –p 80:80 --rm centos_web /usr/sbin/httpd -DFOREGROUND

Ⅳ. 비교 분석

4.1. 설치 관련 분석

하이퍼바이저인 KVM 기반으로 운영하기 위해서는 qemu-kvm이라는 패키지 이외에 가상 머신을 관리하기 위한 가상머신 관리자라는 프로그램을 설치해야 한다. 또한, 가상머신 생성을 위해 메모리나 디스크의 용량을 할당해야 하고, 설치하려는 운영체제인 CentOS 6.5 설치 DVD나 ISO 이미지 파일을 별도로 준비해야 한다. Docker는 해당 프로그램만 설치하면 되고, 웹 서버로 운영하고자 하는 CentOS 6.5는 Docker 저장소에서 제공하고 있으므로 최적화된 해당 컨테이너 이미지만 가져다 사용하면 된다. 따라서, Docker는 인터넷 기반으로 최적화된 운영체제 이미지를 제공받는 형태라 설치가 매우 쉽고 빠르다.

4.2. 자원 분석

4.2.1. CPU 자원 분석

CPU의 자원 분석은 sysstat 패키지의 mpstat 명령을 이용하여 그림 3과 같이 호스트 OS의 남는 자원값인 idle값을 분석한다[9]. KVM은 가상 머신 관리자인 virt-manger를 실행했을 때와 가상 머신을 실행하여 웹 서버를 가동했을 때의 자원 사용량을 분석하고, Docker는 가상 머신 관리자가 필요 없으므로 Container를 실행했을 때만 측정한다.

```
[root@www ~]# mpstat 2 10
Linux 3.10.0-123.el7.x86_64 (www)       07/17/14        _x86_64_        (2 CPU)

17:52:17     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
17:52:19     all   12.03    0.00    1.00    0.00    0.00    0.00    0.00    0.00    0.00   86.97
17:52:21     all   12.00    0.00    1.00    0.00    0.00    0.00    0.00    0.00    0.00   87.00
17:52:23     all   12.00    0.00    1.25    0.00    0.00    0.00    0.00    0.00    0.00   86.75
17:52:25     all    9.75    0.00    1.00    0.00    0.00    0.00    0.00    0.00    0.00   89.25
17:52:27     all   12.31    0.00    0.50    0.00    0.00    0.00    0.00    0.00    0.00   87.19
17:52:29     all    9.48    0.00    1.00    0.00    0.00    0.00    0.00    0.00    0.00   89.53
17:52:31     all   12.03    0.00    1.25    0.25    0.00    0.00    0.00    0.00    0.00   86.47
17:52:33     all   11.78    0.00    1.25    0.00    0.00    0.00    0.00    0.00    0.00   86.97
17:52:35     all   12.00    0.00    1.00    0.00    0.00    0.00    0.00    0.00    0.00   87.00
17:52:37     all   11.78    0.00    1.25    0.00    0.00    0.00    0.00    0.00    0.00   86.97
Average:     all   11.51    0.00    1.05    0.03    0.00    0.00    0.00    0.00    0.00   87.41
[root@www ~]#
```

그림 3. mpstat를 이용한 분석
Fig. 3 Analysis using mpstat

CPU의 idle 값을 분석해보면 KVM 기반의 가상머신 방법에서는 가상 머신 관리자 실행시에 평균 13~14%의 CPU의 자원을 사용하였고, 가상 머신의 실행에 따른 CPU 자원의 증가는 미비하였다. Docker 방식인 경우에는 웹 서버가 동작하는 컨테이너를 실행하여도 평균 2~3% 정도의 CPU 자원만을 사용하였다. 따라서,

Docker를 사용하면 KVM 방식에 비해 훨씬 많은 수의 가상 서버를 구현할 수 있다.

표 7. CPU의 idle 값 비교
Table. 7 Comparison of CPU idle Value

	KVM	Docker
Booting 시	88.13%	88.13%
가상머신 관리자 실행	75.25%	88.13%
가상머신 또는 컨테이너 실행	73.44%	85.67%

4.2.2. 메모리 사용량 분석

메모리 사용량 분석은 그림 4와 같이 “free –m” 명령을 이용하여 버퍼(Buffer)와 캐쉬(Cache)로 사용중인 메모리를 제외한 실제 사용가능한 메모리양을 확인한다[10]. KVM 환경에서는 먼저 부팅 직후의 메모리 상태를 분석하고, 가상 메모리 관리자를 실행했을 때와 가상머신 실행후 웹 서버 가동했을 때를 분석한다. Docker 환경에서는 부팅 직후와 웹 서버가 설치된 Container 실행시를 분석한다.

```
[root@www ~]# free -m
             total       used       free     shared    buffers     cached
Mem:          3786       1836       1949         11          1        681
-/+ buffers/cache:       1154       2632
Swap:         7999          0       7999
```

그림 4. free 명령을 이용한 분석
Fig. 4 Analysis using free command

그림 4와 같은 방법으로 50회 이상 측정하여 메모리 사용량을 분석해보면 KVM 방식은 가상머신 관리자 실행시에 평균 50MB 내외의 메모리를 사용하였고, 가상머신 실행시에는 가상머신에 할당한 용량인 1GB에 육박하는 700MB 내외의 메모리를 사용하였다.

표 8. 메모리 사용량 비교
Table. 8 Comparison of Memory Usage

	KVM			Docker		
	Total	Used	Free	Total	Used	Free
Booting 시	3786	977	2809	3786	977	2809
가상머신 관리자 실행시	3786	1026	2764	3786	977	2809
가상머신 또는 컨테이너 실행	3786	1698	2088	3786	1019	2767

표 8의 결과를 보면 Docker 방식은 웹 서버 역할을 하는 컨테이너 실행을 했을 경우에 대략적으로 40MB 정도의 메모리만 사용하였다. Docker는 메모리가 충분하지 않은 시스템에서도 효율적으로 사용가능한 방식이라고 할 수 있다.

특히, 그림 5와 같이 Docker 기반으로 생성한 2개의 웹 서버 컨테이너를 top 명령어를 이용하여 실제 물리적 메모리 사용량인 RES 항목을 분석한 결과 4~6MB 정도만 사용하였다.

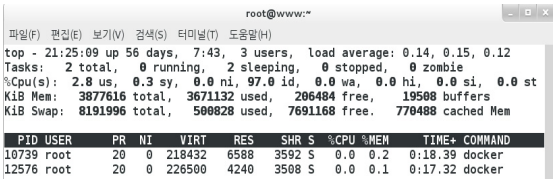

그림 5. Docker 컨테이너의 메모리 사용량 분석
Fig. 5 Docker Container’s Memory usage analysis

4.2.3. 디스크 사용량 분석

KVM 방식의 디스크 사용량은 가상 머신 생성시에 할당하는 용량이라고 볼 수 있다. 즉, 가상머신의 디스크 용량에 10GB를 할당하면 호스트 OS에 10GB 짜리 관련 파일이 생성된다. 그러나, 최근 가상머신 관리자에서는 디스크 사용량의 효율성을 높이기 위해서 VirtualBox의 가상머신 관리자가 사용하는 디스크 동적 할당 방법을 지원하고 있다. 가상머신 생성시에 10GB를 할당하였다고 하더라도 실제 가상머신에 설치된 게스트 OS의 용량이 5GB라면 디스크 사용을 5GB만 차지한다. 다음의 그림 6을 통해 가상머신관련 파일을 용량을 ‘ls –l’ 했을 경우와 ‘du –sh’했을 경우에 차이를 확인할 수 있다. ‘ls –l’했을 때 나타나는 약 10GB에 해당하는 용량이 가상머신 생성시에 할당된 디스크 공간이고, ‘du –sh’했을 때 나타내는 5.5GB라는 수치가 실제 설치된 용량이라고 볼 수 있다.

Docker의 경우에는 다음의 그림 7와 같이 /var/lib/docker/devicemapper/devicemapper라는 디렉터리안에 있는 data 및 metadata 파일의 용량으로 디스크 사용량을 확인할 수 있다.. 보통 버전마다 차이가 있지만 대략적으로 data는 1.2GB 정도이고 metadata는 보통 몇 MB이다. 또한 컨테이너를 여러 개를 생성해도 가상머신 방식처럼 크게 증가되지 않으므로 가상머신 방식에 비

해 디스크 사용량은 현저히 낮다고 볼 수 있다.

```
[root@www ~]# ls -l /var/lib/libvirt/images/www.img
-rw-------. 1 root root 10737418240 Jul 18 13:40 /var/lib/libvirt/images/www.img
[root@www ~]# du -sh /var/lib/libvirt/images/www.img
5.5G    /var/lib/libvirt/images/www.img
```

그림 6. KVM의 가상머신 디스크 사용량 분석
Fig. 6 KVM's Virtual Machine disk usage analysis

```
[root@www devicemapper]# pwd
/var/lib/docker/devicemapper/devicemapper
[root@www devicemapper]# du -sh *
1.2G    data
3.6M    metadata
```

그림 7. Docker의 디스크 사용량 분석
Fig. 7 Docker's disk usage analysis

4.3. 네트워크 및 기타 설정 분석

KVM 기반의 가상 머신에 웹 서버를 운영하기 위해서는 가상 머신의 네트워크 설정에 대해 명확히 이해해야 한다. 공인 IP 및 사설 IP에 대한 개념, SNAT 및 DNAT 등에 대해 명확히 알아야 한다. 또한, 패킷 필터링 역할을 하는 iptables에 대해서도 알아야 하므로 설정하기가 쉽지 않다. 그러나, Docker를 사용하는 경우에는 docker 관련 명령 하나만 알면 손쉽게 운영이 가능하다.

4.4. 웹 서버 운영 분석

KVM 기반의 가상 머신으로 웹 서버 운영할 때 데이터의 저장이나 변동되었을 경우에는 할당된 별도의 디스크 공간이 존재하므로 큰 문제가 없다. 그러나, docker의 컨테이너는 변경된 내용이 저장되지 않는다. 변경된 사항을 저장하기 위해서는 커밋(commit)이라는 작업이 요구되고, 커밋된 이미지는 Docker 저장소에 보관하여 개인적으로 사용하거나 다른 사용자에게 공개할 수도 있다. 또한, 웹 서버의 경우에는 대부분의 데이터가 데이터베이스(Database)에 저장되고, html이나 이미지 파일 등은 네트워크 파일시스템(Network File System:이하 NFS) 이나 스토리지에 저장되는 경우가 많다. 따라서, 데이터베이스를 사용하고, NFS 및 스토리지를 이용하는 웹 서버 환경에서는 Docker로 운영해도 큰 문제가 없다.

Ⅴ. 결 론

현재 클라우드 인프라 구성에 많이 사용되는 Xen, KVM, VMware, Virtualbox 등을 살펴보면 대부분 하이퍼바이저 방식으로 호스트 OS위에 가상머신을 생성하여 게스트 OS를 설치하고 다시 응용프로그램을 구동하는 방식이다. 자원을 명확히 할당받아 사용하므로 안정적인 서비스를 받을 수 있다는 장점을 있으나, 하드웨어 자원의 한계로 인해 많은 수의 서비스를 운영하기 어렵고 자원을 할당한 만큼 사용하지 않는다면 또 다른 자원의 낭비를 초래할 수 있다. 그러나, Docker가 사용하는 컨테이너 방식은 가상 머신처럼 운영이 가능하면서, 하드웨어 자원의 사용률은 일반적으로 동작하는 응용프로그램의 수준 정도이다. 따라서, 하드웨어의 자원 사용률은 현저히 낮으면서 내부적으로 프로세스로 처리되어 관리의 효율성도 높다. 물론 가상머신 환경의 변화가 많거나 데이터 저장이 빈번하면 컨테이너 저장 단계가 필요하지만, 웹 서버나 프로그램 개발 환경과 같이 특정 데이터를 데이터베이스, 스토리지 등에 저장하는 경우에는 매우 유용하게 사용될 수 있다. 앞으로의 클라우드 환경은 Docker와 같은 컨테이너 방식으로 자원의 효율성과 관리의 편의성을 더욱 높이리라 사료된다.

감사의 글

이 논문은 2013년도 한남대학교 학술연구조성비 지원에 의하여 연구되었음

REFERENCES

[1] Docker, http://www.docker.com

[2] KVM, http://www.linux-kvm.org

[3] S. J. Jung, K. Sung, and Y. M. Bae, "Comparison and Analysis of Resource Usage of Open Source Server Virtualization Techniques," Journal of The *Korea Knowledge Information Technology Society,* vol. 7, no. 5, pp. 43-49, Apr. 2011.

[4] S. J. Jung, “Implementation and Utilization of Linux Virtualization for Cloud Computing Infrastructure Development”, Ph. D. dissertation, Hannam University, Daejeon, pp. 19-24, Feb. 2011.
[5] CentOS Project, http://www.centos.org
[6] Netfilter, http://www.netfilter.org
[7] Y. M. Bae, S. J. Jung, and W. S. Soh, “Design and Implementaion of Web Clustering System Based on Linux Virtualization,” The Journal of The *Korea Institute of Information Technology,* vol. 8, no. 11, pp. 251-260, Nov. 2010.
[8] Docker Registry, https://registry.hub.docker.com/
[9] Sysstat, http://sebastien.godard.pagesperso-orange.fr/
[10] Procps-ng, https://sourceforge.net/projects/procps-ng/

배유미(Yu-Mi Bae)

2005년 2월 한남대학교 컴퓨터멀티미디어학과 공학사
2007년 8월 한남대학교 정보기술학과 공학석사
2013년 2월 한남대학교 컴퓨터공학과 공학박사
2013년 ~ 현재 ㈜스컴씨엔에스 기업부설연구소 선임연구원
※관심분야 : 리눅스, 정보보호, 클라우드 컴퓨팅, 서버 가상화

정성재(Sung-Jae Jung)

1998년 2월 한남대학교 컴퓨터공학과 공학사
2003년 8월 한남대학교 컴퓨터공학과 공학석사
2011년 2월 한남대학교 컴퓨터공학과 공학박사
2005년 3월 ~ 2010년 2월 한남대학교 국제IT교육센터 전임강사
2012년 9월 ~ 현재 ㈜스컴씨엔에스 기업부설연구소 연구소장
※관심분야 : 리눅스, 정보보호, 클라우드 컴퓨팅, 서버 가상화

소우영(Woo-young Soh)

1979년 중앙대학교 전자계산학과 학사
1981년 서울대학교 전자계산학과 석사
1991년 메릴랜드대학교 전자계산학과 공학박사
1991년 ~ 현재 한남대학교 컴퓨터공학과 교수
※관심분야 : 정보보호, 암호학