서버 기반 컴퓨팅을 이용한 가상 OS 활용 및 구현

사공현, 신장원, 곽종욱
영남대학교 컴퓨터공학과
e-mail:midsight@naver.com

Implementation of Virtual OS Application using Server Based Computing

Hyeon Sagong, Jang Won Shin, Jong Wook Kwak
Dept. of Computer Engineering, Yeungnam University

요 약

서버 기반 컴퓨팅(Server Based Computing)은 데이터와 작업 처리가 서버에서 이루어지기 때문에 데이터를 효과적으로 통합하고 관리를 할 수 있다. 본 논문에서는 서버 기반 컴퓨팅을 이용하여 사용자에게 본인만의 데스크톱 환경을 제공하고, 언제 어디서나 필요한 정보와 애플리케이션을 실행할 수 있는 방법을 제안한다. 이러한 환경 하에서 최대한 서버의 활용률을 높이고 낭비하는 자원을 줄이기 위해 서버 가상화 기법(Server Virtualization)과 가상 OS 메모리 할당 알고리즘을 도입하였다. 서버와 사용자의 수에 따른 메모리 할당 방식을 hard handoff 라고 명하고, 사용자에게 메모리를 적절히 할당할 수 있도록 하였다. 또한 기존 사용자에 대한 메모리 재할당의 경우, Immutable OS와 별도의 사용자 데이터 공간으로 나누어 관리하여 가상 OS의 재접속 시간을 단축시킬 수 있었다.

1. 서론

최근 들어 여러 기업과 공공기관 사이에서 서버 기반 컴퓨팅(Server Based Computing)의 수요가 증가하고 중요성이 높아지고 있다. 서버 기반 컴퓨팅이 주목받는 배경으로는 효율적인 업무 환경과 보안성 향상 및 관리 비용 절감의 장점이 있기 때문이다.

이러한 수요가 증가함에 따라 서버를 효율적으로 사용할 수 있는 기술 또한 이슈가 되고 있다. 서버 가상화 기법(Server Virtualization)은 서버의 통합 및 분리를 통해 시스템의 자원을 보다 효율적으로 활용할 수 있는 환경을 제공해 준다. 통계적으로 봤을 때 활용도가 낮은 많은 수의 서버가 존재하여 시스템 자원이 낭비가 되고 있기 때문에 물리적으로 서버를 통합하여 시스템의 성능을 높이고, 논리적으로 분할을 하여 높은 활용률과 시스템 비용의 절감 효과를 얻게 된다[1].

대부분의 기업과 공공기관의 IT 환경은 많은 사용자가 공용으로 사용하기 때문에 OS 환경이 점점 악화된다. 이로 인해 시스템 속도 저하와 악성 코드 등 보안성에 문제가 발생하게 되고, 정기적으로 많은 수의 컴퓨터를 관리하는데 적지 않은 시간과 노력이 소모된다. 따라서 본 논문에서는 사용자에게 가상의 데스크톱 환경을 지원하기 위해 서버 가상화 기법 도입과 서버 기반 컴퓨팅 환경을 이용할 수 있는 방법을 소개한다. 이를 통해 사용자는 데이터의 통합 관리가 가능하고, 관리자는 서버를 통해 비교적 간단한 조작만으로 유지 보수 및 관리를 할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구와 본 논문에서 사용된 주요 배경 기술들을 살펴보고, 3장에서는 전체 시스템 구성과 세부 동작에 대하여 설명한다. 4장에서는 모의 실험 환경을 소개하고 제한된 시스템의 성능에 대한 분석과 토의를 한다. 마지막으로 5장에서는 본 시스템에 대한 결론에 대해서 논의한다.

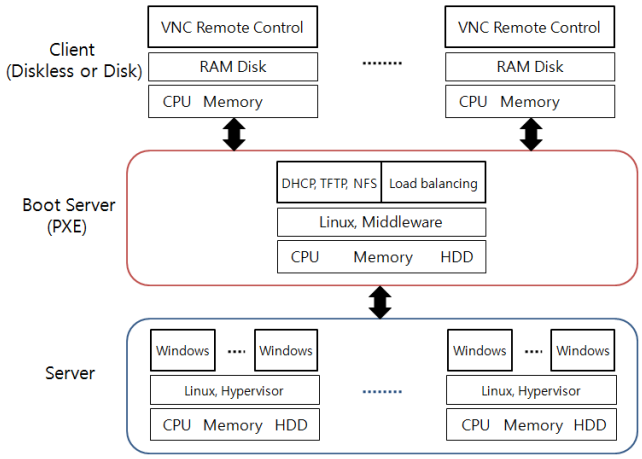
2. 관련 연구 및 배경 지식

2.1. 관련 연구

일반적으로 원격 접속을 하기 위해 로컬 컴퓨터의 운영체제 상에서 원격 제어 프로그램을 실행하게 된다. 가상화를 이용하여 시스템을 구축한 “Xen 기반 관리구조[2]” 역시, 서버 기반에서 동작하는 애플리케이션을 이용하게 되지만 로컬 하드디스크가 사용된다는 점에서 추후 별도의 유지 보수와 관리가 필요하게 된다.

또한 워크스테이션급의 시스템이 아니라 일반 데스크톱 PC로 서버를 구축하기 때문에 시스템의 성능과 자원을 최대한 활용할 수 있어야 한다. 그 중에서 다른 자원에 비해 다소 부족한 메모리 이용률을 향상시킬 필요가 있다. 관련 연구로 가상화 시스템에서 메모리 할당량을 동적으로 조절하여 시스템 자원을 효율적으로 관리하고 활용하는 방법을 설명하고 있다[3].

본 논문에서는 대기 중인 메모리 자원을 최소화하기 위해 메모리 할당 알고리즘을 도입하고, 서버 기반에서 동작하는 가상 데스크톱 환경을 구축하는 방법을 제안한다.



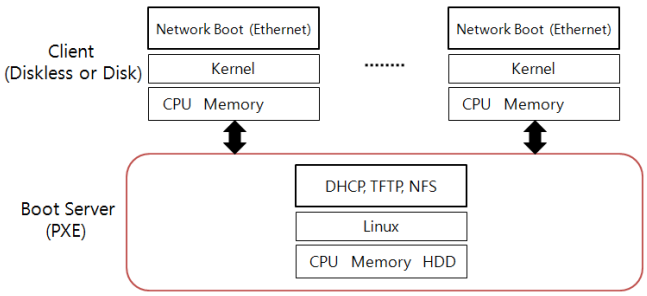
(그림 1) 시스템 구조

2.2. 배경 기술

본 논문에서 제안된 시스템에서 활용하는 기술들에 대해 소개한다. PXE(Pre-Execution Environment)[4]는, 네트워크 인터페이스를 통해 부트 이미지 파일 정보를 전송하여 컴퓨터를 부팅할 수 있게 해준다. NAT(Network Address Translation)[5]는 사설 IP 주소를 공인 IP 주소로 변환시키는 통신망의 주소 변환기이고, IP Masquerade(IP MASQ)[6]는 NAT의 한 형태로 리눅스 서버에 연결된 하나의 공인 IP 주소를 통해서 등록된 IP 주소가 없거나 사설 IP 주소를 부여 받은 내부의 컴퓨터들이 인터넷을 이용 가능하도록 하는 기능이다. 그리고 Port forwarding[7]은 특별히 설정된 포트로 패킷을 전달하는데 NAT 또는 IP MASQ와 함께 쓰인다.

SDL(Simple Directmedia Layer)[8]은 멀티미디어 및 입력 부분을 처리하는 API로 구성된 2D 그래픽 라이브러리이다. SDL 라이브러리를 바탕으로 제작된 VNC(Virtual Network Computing)[9] 툴은 원격 접속을 통해 서버의 가상 OS를 제어하고 클라이언트의 텍스트 모드(커맨드 입력) 상에서 그래픽 화면을 출력하게 된다.

Hypervisor(또는 Virtual Machine Monitor)[10]는, 다수의 OS를 동시에 실행하기 위한 가상 플랫폼 계층이며, Middleware[11]는 클라이언트가 서버에 어떠한 처리를 요구하고, 서버가 그 처리한 결과를 클라이언트에게 돌려주는 과정을 효율적으로 수행하도록 도와주는 역할을 한다.



(그림 2) 네트워크 부팅 과정

그리고 Java RMI(Java Remote Method Invocation)[12]은 로컬 컴퓨터에서 원격지의 메소드를 호출할 수 있어서 분산 처리 작업을 수행 할 수 있다.

3. 서버 기반 컴퓨팅을 이용한 가상 OS 활용 및 구현

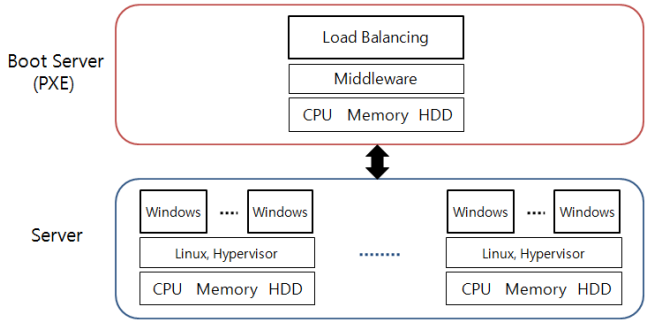
3.1. 기본 시스템 구성

서버 기반 컴퓨팅을 이용한 가상 OS 활용 및 구현은 그림 1과 같이 크게 세 부분으로 나누어져 있다. 클라이언트는 사용자가 서버로 접속할 PC이고, 서버는 클라이언트로 부트 이미지 정보를 전송할 부트 서버와 가상 OS를 지원하는 서버로 나뉜다.

클라이언트는 하드디스크가 없어도 부팅[13]을 할 수 있고 기본적인 하드웨어 제어를 위한 커널과 리눅스 기반의 파일시스템이 램디스크 상에서 동작하게 된다. 그 이후의 모든 작업은 서버에서 이루어지며 그 결과만을 화면에 출력한다. 부트 서버는 PXE[4] 서버라고 불리며, 네트워크 부팅에 필요한 장치로써 IP 주소를 관리하고 부팅에 필요한 파일을 전송하는 역할을 한다. 그리고 서버에 탑재된 미들웨어[11]로 적절한 분산 처리를 수행하면서 NAT[5]를 통한 네트워크 연결의 중간 통로가 된다. 최종적으로 클라이언트는 이 부트 서버를 거쳐서 가상 OS로 원격 접속을 하게 된다. 마지막으로 하위에 가상 OS를 지원하는 서버가 있다. 리눅스가 호스트 OS로 동작함과 동시에 하이퍼바이저[10]가 다수의 운영체제를 관리하게 된다.

3.2. 세부 동작 과정

전체 시스템 운용상에서는 두 부분으로 나눌 수 있다. 먼저 그림 2는 네트워크 부팅을 수행하는 과정을 나타내고 있다. 클라이언트 PC 전원이 켜지면 부트 서버로 IP 주소를 요청한 다음, 커널 이미지와 시스템 초기화 및 시스템 구성을 위한 초기 루트 디스크 파일[14]을 TFTP 프로토콜로 전송받는다. 초기화 과정이 끝나면 루트 파일시스템을 램 디스크 상에서 사용하게 되는데, 이 부분은 전송하기에 무리가 되는 용량이 큰 파일로 구성되어 있어서 NFS로 설치한다. 그리고 사용자 접속 모듈이 자동적으로 실행되면서 그림 3에서 부트 서버에 탑재된 미들웨어로 부팅이 완료되었다는 신호를 보낸다.



(그림 3) 부트 서버 및 가상화 서버 동작 과정

중앙에서 통제 역할을 하는 미들웨어가 부팅 완료 신호를 받으면, 가상 OS를 구동하는 서버 전체를 모니터링하면서 적절한 분산 처리를 하게 된다. 그리고 운영체제(가령 Windows)를 대기 중인 특정 서버로 실행 할 것을 명령하면 서버에서는 가상 머신 상에서 운영체제를 부팅한다. 부팅이 완료되면 사설 IP를 부여받은 클라이언트가 가상 OS로 원격 접속을 바로 할 수 없기 때문에, 부트 서버에서 미리 설정해 둔 IP 마스커레이딩[6]과 포트포워딩[7]을 통해서 외부 네트워크와 연결이 이루어진다.

클라이언트는 그래픽 작업 환경이 아닌 텍스트 모드 환경에서 원격 접속 화면을 출력 할 수 있어야 한다. 이것은 GUI 요소를 바탕으로 한 일반적인 VNC[9] 애플리케이션으로는 불가능하고, SDL[8] 라이브러리를 바탕으로 수정한 VNC 애플리케이션으로 원격 접속을 할 수 있다.

마지막으로 서버의 대기 중인 자원을 최소화하고 시스템 활용률을 높이기 위해서 메모리 관리를 지속적으로 하게 되는데 다음 알고리즘으로 자세히 설명하도록 한다.

3.3. 메모리 할당 알고리즘

사용자가 적을 경우 서버의 대기 자원을 충분히 활용한다면 좀 더 좋은 성능에서 작업을 할 수 있다. 서버에서 실행되는 가상 OS를 이용하기 때문에 서버의 메모리를 효율적으로 관리할 수 있는 방법이 필요하다.

시스템의 서버 수와 할당할 메모리 용량 단계를 적용하면 사용자 수 U를 기준으로 메모리 할당량 식 (1)을 정의할 수 있다.

$$U = 2^m \cdot S - 1 \quad (1)$$

(m = 메모리 할당 단계, S = 서버 수(V))

예를 들어, 서버 하나의 최대 메모리 용량이 2GB이고 메모리 할당량을 2GB, 1GB, 512MB 세 단계로 나누면 m은 차례대로 0, 1, 2 이다. 서버 수가 총 4대 1(B)+3(V)[1] 일 때, 식 (2)와 같은 사례를 확인 할 수 있다.

$$\begin{aligned} U_0 &= 2^0 \cdot 3 - 1 = 2\text{명} \\ U_1 &= 2^1 \cdot 3 - 1 = 5\text{명} \\ U_2 &= 2^2 \cdot 3 - 1 = 11\text{명} \end{aligned} \quad (2)$$

$(0 \le m \le 2,\ S = 3(V))$

접속 인원 2명까지는 최대 메모리 용량인 2GB를 할당해주고, 3명에서 5명까지는 1GB씩 할당하면서 2GB 사용자를 점차 1GB로 용량을 줄이고 재할당한다. 6명 이후로는 최소 용량인 512MB로 할당하고, 1GB 사용자를 점차 512MB로 재할당한다.

표 1은 식 (1)과 식 (2)를 바탕으로 본 논문에서 제안된 메모리 할당 알고리즘이다. 부트 서버의 미들웨어가 메모리 할당 알고리즘을 사용하여 사용자 수에 따라 가상 OS 메모리 할당하는 기준을 정한다.

1) B은 부트 서버, V는 가상화 서버로 구분한다.

<표 1> 가상 OS 메모리 할당 알고리즘

```
initialize m to max(m)
initialize S
initialize U as array
for index from zero to m by increase index one
    U[index] = 2^index · S - 1

check userList
initialize currentUsers to size(userList)
call isRealloc(currentUsers, U)
if isRealloc is equal to true
    reallocate memory to last connection user

if the currentUsers is less than or equal to U[zero]
    allocate maximum memory
else if it is between U[zero] and U[one]
    allocate half of maximum memory
    ...
else
    send memory allocation error
```

4. 성능 평가

4.1. 모의 실험 환경

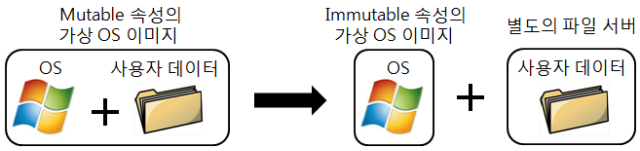
<표 2> 실험 환경

부트 서버	CPU	single-core 2.8GHz
	Memory	DDR 512MB
	HDD	80GB
가상화 서버	CPU	dual-core 2.66GHz
	Memory	DDR2 2GB
	HDD	240GB
	CPU	dual-core 2.90GHz
	Memory	DDR3 2GB
	HDD	320GB
	CPU	dual-core 2.93GHz
	Memory	DDR3 2GB
	HDD	500GB

표 2는 모의 실험 환경에서 사용할 서버로 역할에 따라 부트 서버와 가상화 서버로 나눈다. 부트 서버는 네트워크 및 부트 이미지 전송 작업과 미들웨어만 실행하기 때문에 비교적 높은 성능을 요구하지 않는다. 반면에 가상화 서버는 가상 OS를 지원하기 위한 서버로써 다수의 운영체제가 동시에 실행되는 환경이기 때문에 부트 서버에 비해 빠른 작업이 이루어져야 한다. 또한 대규모의 데이터를 저장하고 처리하기 위해서 메모리와 하드디스크의 용량이 비교적 크다.

<표 3> 접속 인원에 대한 메모리 할당량

접속 인원(명)	메모리 할당량(GB)											남은 용량(GB)
1	2	idle	idle									4
2	2	2	idle									2
3	1	2	2									1
4	1	1	1	2								1
5	1	1	1	1	1							1
6	0.5	1	1	1	1	1						0.5
7	0.5	0.5	0.5	1	1	1	1					0.5
8	0.5	0.5	0.5	0.5	0.5	1	1	1				0.5
9	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1	1			0.5
10	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1		0.5
11	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
12	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0



(그림 4) 메모리 재할당을 위한 데이터 관리법

4.2. 시스템 성능 평가 및 분석

접속 인원에 대한 메모리 할당량의 실험 결과가 표 3에 소개되어 있다. 서버 수가 총 4대 1(B)+3(V)이고 최대 접속 인원이 12명일 때 메모리 할당량이다. 인원에 따라 할당량이 점차 낮아지고 기존 사용자에 대한 메모리 용량 또한 재할당이 이루어진다.

기존 사용자에 대한 재접속은 미리 재할당된 가상 OS를 부팅시키고 접속 경로를 바꿔준다. 그 후에 기존의 가상 OS는 종료를 시키고 다음 사용자에 대한 메모리를 확보한다. 이는 결과적으로 사용자가 재할당된 가상 OS로 교체하는 시간을 단축시키기 위한 작업이다. 실제로 가상 OS 부팅 시간이 단축되는 것은 아니고, 사용자 입장에서 교체 대기 시간을 최대한 줄이도록 해야 한다. 가상 OS를 교체해도 사용자마다 본인만의 데이터를 유지하는 이유는 그림 4의 방법을 도입했기 때문이다. 가상 OS가 Immutable의 속성을 가지고 있고 별도의 데이터 공간을 이용한다. 그리고 hard handoff 방식으로 메모리를 재할당하여 기존 사용자의 재접속 시간을 단축하였다.

5. 결론

본 논문에서는 기업과 공공기관의 IT 환경을 좀 더 효율적으로 관리하고자 서버 기반 컴퓨팅을 연구하여 데이터 및 운영체제를 서버에서 관리하고 동작하도록 설계하였다. 가상 OS를 제공하여 네트워크가 연결된 어느 곳에서든 자신의 데스크톱 환경으로 접속하여 작업을 수행할 수 있다. 원격 접속 환경이기 때문에 물리적인 장치 사용이나 멀티미디어 성능상 제한적이지만, 시스템 관리와 서비스 사용 측면에서 많은 장점을 가지고 있다.

서버 가상화의 경우, 특히 메모리 자원을 효율적으로 관리하기 위해 본 시스템에서는 hard handoff 방식으로 각 사용자에게 메모리를 할당하도록 하였다. 사용자가 많을 때, 다음 사용자의 메모리 확보를 위해서 기존 사용자의 OS가 재부팅하는 시간이 필요하지만, 데이터 관리 방식을 바꾸고 미리 재할당한 OS를 부팅시킴으로써 재접속 시간을 단축시킬 수 있었다.

참고문헌

[1] 한국IBM 시스템 테크놀로지 그룹, “가상화 기술의 새로운 패러다임”, 2007
[2] 유은하, 주홍택, 김태영, “Xen 기반의 컴퓨터실습실 관리 구조”, KNOM Review, Vol. 12, No. 2, Dec. 2009, pp. 49-58
[3] 이권용, 박성용, “Xen에서 메모리 이용률 향상을 위한 동적 할당 기법”, 정보과학회논문지 시스템 및 이론 제 37권 제 3호, 2010
[4] PXE, http://syslinux.zytor.com/wiki/index.php/PXELINUX
[5] 임경훈, “vmware 워크스테이션”, 2007
[6] IP MASQ, http://www.e-infomax.com/ipmasq/
[7] Port forwarding, http://portforward.com/help/portforwarding.htm
[8] SDL, http://www.libsdl.org/
[9] VNC, http://www.tightvnc.com/
[10] Hypervisor, http://www.xen.org/products/xenhyp.html
[11] Middleware, http://docs.sun.com/app/docs/doc/820-0532/6nc919fai?l=ko&a=view
[12] Java RMI, http://download.oracle.com/javase/tutorial/rmi/index.html
[13] 신창균, 혜지원, “(리눅스&윈도우) 클러스터로 도전하는 슈퍼컴퓨터 : 구축과 활용”, 2006
[14] 이토 나오야, “서버/인프라를 지탱하는 기술”, 2009