WAN 환경에서의 가상 머신 마이그레이션 성능 향상을 위한 가상 머신간 파일 블록 공유에 관한 연구*

신은환, 김정한, 엄영익
성균관대학교 정보통신공학부
{comsky, gtgkjh, yieom}@ece.skku.ac.kr

A Study on Sharing the File Blocks Between Virtual Machines for Efficient Virtual Machine Migration in Wide Area Networks

Eun Hwan Shin, Jung Han Kim, Young Ik Eom
School of Information and Communication Engineering
Sungkyunkwan University

요 약

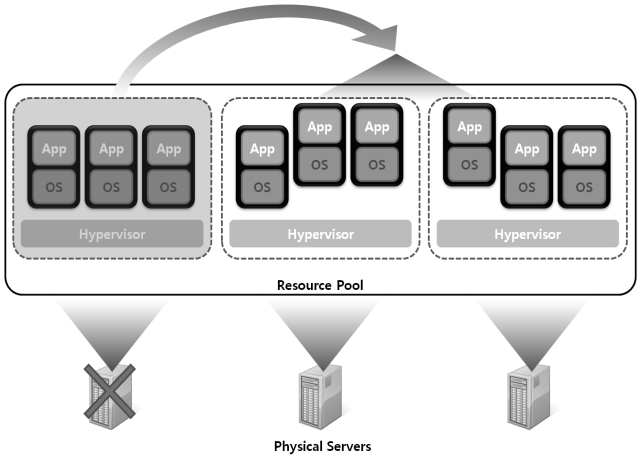
가상화는 하나의 물리적 시스템을 논리적으로 분할해 자원 공유의 효율성을 극대화하는 기술이다. 하지만 가상화에 필수적인 시스템의 추상화는 자원 관리의 복잡도를 증가시켜 하드웨어 장치의 로드 발생을 야기한다는 문제점이 있다. 가상 머신 마이그레이션(migration)은 로드를 발생시키는 가상 머신을 자원적 여유가 있는 시스템으로 이동시키는 기술이며, 기존의 기법들은 메모리상의 실행 컨텍스트의 이동 비용을 최소화하기 위한 방법에 초점을 맞추고 있다. 하지만 네트워크를 통한 스토리지의 공유가 어려운 WAN(wide area network) 환경에서는 실행 컨텍스트뿐만 아니라 가상 머신 이미지 자체의 이동이 불가피하며, 이는 많은 비용을 발생시킨다. 따라서 본 연구에서는 가상 머신간 파일 블록 공유를 통한 WAN 환경에서의 효율적인 가상 머신 마이그레이션 기법을 제안하고, 실험을 통해 제안 기법에 대한 분석 및 평가를 실시하였다.

1. 서론

가상화(virtualization) 기술은 하나의 물리적 장치를 복수개의 가상 머신으로 나누어 자원 공유의 효율성을 극대화하는 기술이다. 이러한 자원 공유의 장점에 의해 시스템 하드웨어 구입 및 유지보수에 소모되는 비용이 절감되며, 이는 물리적 장치수의 감소에 따른 것이다.

하지만 하나의 시스템을 논리적으로 분할하거나, 물리적으로 다른 시스템을 논리적으로 통합하기 위해 가상화 환경에서는 시스템 추상화(abstraction)의 도입이 필수적이며, 이는 자원 관리의 복잡도를 증가시켜 하드웨어 자원의 로드 발생을 유발한다는 문제점이 있다.

가상 머신 마이그레이션(migration)은 그림 1과 같이 과도한 로드를 발생시키는 가상 머신을 자원의 여유가 있는 다른 물리적 장치로 이동함으로써 특정한 장치로의 로드 집중을 방지하고 컴퓨팅 자원의 균형 분배를 목적으로 하는 기술이다. 기존의 연구들은 가상 머신 이동시 메모리상에서 실행 중이던 컨텍스트를 최소한의 비용으로 이동시



(그림 1). 가상 머신 마이그레이션(migration)의 예

키는데 그 초점을 맞추고 있다. 이와 같이 기존 연구들에서는 마이그레이션시 가상 머신 이미지 파일 자체의 이동에 대한 고려는 이루어지지 않고 있으며, 이는 상대적으로 근접한 네트워크 환경에서 스토리지의 공유가 가능하다는 것을 전제로 하고 있다. 하지만 최근 주목 받고 있는 클라우드 컴퓨팅과 같은 가상화 관련 기술들은 기본적으로 데이터 센터 간을 연결하는 대규모 네트워크 환경을 구성하

* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (NIPA-2010-(C1090-1021-0008))

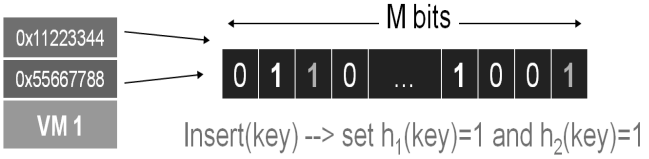
고 있다. 이러한 환경에서도 데이터 센터 내부에서의 하드웨어 로드 발생으로 인한 데이터 센터간의 가상 머신 마이그레이션이 발생할 수 있지만, 먼 거리로 인해 기존 환경과 같은 스토리지 공유가 사실상 어렵다. 따라서 메모리 상의 실행 컨텍스트뿐만 아니라 가상 머신 이미지 파일 자체에 대한 이동이 필요하며, 이를 효율적으로 수행하기 위한 방법이 요구된다.

이에 본 논문에서는 WAN(wide area network) 환경에서의 가상 머신 마이그레이션 성능 개선을 위해 가상 머신간 파일 블록 공유에 관한 아이디어를 제안한다. 본 논문의 2장에서는 관련 연구 및 기술 현황에 대해서 다루고, 3장에서 제안 아이디어에 대한 소개를, 그리고 4장에서 실험 및 평가를 실시한다. 마지막 5장에서 본 논문을 마무리한다[1][2].

2. 관련 연구 및 기술 현황

2.1 Memory Buddies

Memory Buddies는 가상 머신간 메모리 공유 가능성을 평가하고 그 결과를 토대로 로드를 발생시킨 가상 머신의 재배치를 수행하기위한 시스템이며, Massachusetts 대학의 Timothy Wood에 의해 제안되었다. 이 시스템은 가상 머신간의 공유 가능성을 효과적으로 평가하기 위해 memory fingerprinting이라는 시스템을 포함하고 있으며, 이를 토대로 가상 머신을 재배치하기위한 최적의 위치를 선정한다. 여기서 memory fingerprint란 유사도를 비교하고자하는 각 4Kbyte 메모리 페이지에 대한 해시값을 의미한다. 해시값의 리스트 크기는 시스템 물리 메모리 크기에 대비하여 증가하며, Memory Buddies에서는 비교시 속도 개선을 위해 Bloom 필터라는 자료 구조를 사용한다. Bloom 필터는 그림 2와 같이 각 해시값을 0, 1과 이루어진 M비트 벡터 값으로 저장하는 방식이며, Bloom 필터를 적용한 비교시 일반적인 해시값 비교 대비 최소 10배에서 100배의 속도 향상이 있었다.

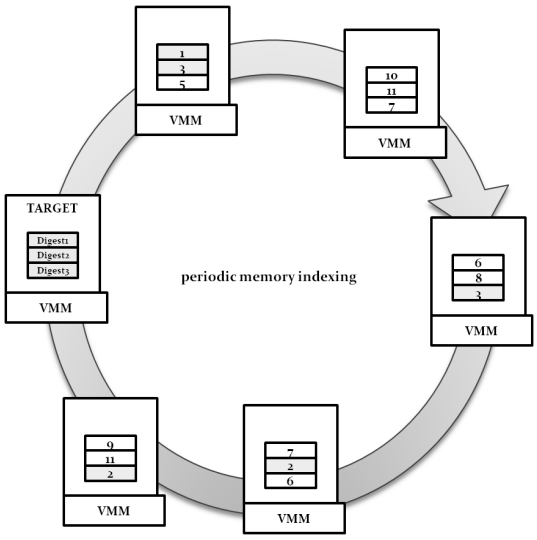


(그림 2) Bloom 필터

이와 같이 memory fingerprinting 시스템을 적용해 가상 머신간 메모리 공유 가능성을 평가한 실험 결과 Linux와 Windows 시스템간의 조합의 경우 최대 36.8%, 동일한 Linux 배포판 간의 조합의 경우 40%의 메모리 공유가 가능함을 보여준다. 이러한 실험 결과를 토대로 저자는 메모리 공유도를 고려해 물리적 서버상에 최대한 많은 수의 가상 머신을 배치함으로써 자원 활용의 극대화가 가능하다고 주장하고 있다[3].

2.2 Shrinker

Shrinker는 WAN 환경에서의 효과적인 실시간 가상 머신 마이그레이션을 수행하기 위한 시스템이다. 저자인 Rennes 1 대학의 Pierre Riteau은 기존의 기법들이 LAN(local area network) 환경에 한정적이라는 것에 대해 문제를 제기하고, WAN 상에 존재하는 대규모 클러스터 단위 가상 머신들에 대한 마이그레이션 기법을 제안하였다. Shrinker는 메모리 페이지의 해시값 비교를 통해 마이그레이션 대상 가상 머신에서 요구되는 메모리 페이지들이 해당 네트워크상에 존재하는지를 확인한다. Shrinker는 그림 3과 같이 주기적인 메모리 인덱싱을 통해 해당 네트워크 내부에 존재하는 노드들이 지닌 메모리 페이지들의 정보를 수집한다. WAN으로부터 마이그레이션 요청이 있을시 Shrinker는 수집한 정보를 근거로 네트워크에 분산되어 있는 메모리 페이지들을 찾아 마이그레이션 대상 가상 머신으로 복사한다. 요구하는 메모리 페이지가 네트워크 내부에 존재하지 않을 경우, 요청이 들어온 네트워크 노드로 돌아가 동일한 작업을 수행한다. Shrinker는 KVM 하이퍼바이저 내부에 구현되었으며, 실험 결과 WAN에서의 네트워크 전송량이 약 30~40%, 총 마이그레이션 수행 시간이 약 20% 감소하는 것을 보여주고 있다[4].



(그림 3) periodic memory indexing

2.3 KVM

KVM(Kernel-based Virtual Machine)은 Qumranet사가 2006년에 개발한 가상화 지원 플랫폼이며, 가상화(Intel VT, AMD-V) 확장 명령어 셋을 지원하는 x86 기반의 리눅스 환경에서의 전 가상화를 지원한다. 이는 가상화 핵심 인프라를 제공하는 커널 모듈인 kvm.ko와 프로세서에 특화된 모듈인 kvm-intel.ko와 kvm-amd.ko로 구성되어 있다. KVM을 이용한 가상화 환경에서는 복수대의 Linux 혹은 Windows 가상 머신 이미지를 수정 없이 구동 가능하며, 각 가상 머신은 개별적으로 가상화된 하드웨어를 가진다[5][6].

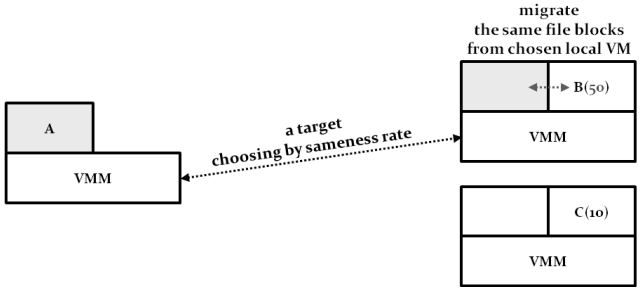
2.4 MD5

MD5(Message-Digest algorithm 5)는 128비트 암호화 해시 함수이다. 주로 프로그램이나 파일이 원본과 일치하는지 여부를 확인하는 무결성 검사 등에 사용된다. 1991년에 미국의 암호학자인 Ronald Lorin Rivest에 의해 고안되었다. 8비트에 최적화된 초기 버전인 MD2를 거쳐 32비트 컴퓨터에 최적화된 MD4가 개발되었으며, 최신 버전인 MD5는 MD4에 비해 속도가 느리지만 데이터 보안에 있어 더 많은 확신을 제공한다. MD5는 임의의 길이를 가진 메시지(variable-length message)를 입력받아, 128비트 길이의 고정적인 값을 출력한다. 입력된 메시지는 512비트의 블록들로 분할되며, 그 과정은 다음과 같다. 입력 받은 메시지의 끝부분에 단일 비트인 1을 추가한 뒤, 512의 배수 길이보다 64비트가 적은 곳까지 0으로 채운다. 남은 64비트는 최초 메시지의 길이를 나타내는 64비트 정수(integer)값으로 채워진다. 메인 MD5 알고리즘은 A, B, C, D라고 이름이 붙여진 32비트 워드 4개로 구성된 하나의 128비트 스테이트(state)에 대해 동작한다. A, B, C, D는 정해진 상수 값으로 초기화되며, 메인 알고리즘은 각 512비트 크기의 입력 메시지에 블록에 대해 차례로 동작한다. 각 512비트 입력 메시지 블록의 처리가 끝난 뒤 128비트 스테이트(state)의 값이 변하게 된다. 본 연구에서는 가상 머신 이미지 파일간의 일치도를 비교하기 위하여 512바이트 단위 파일 블록들에 MD5 변환을 적용하였다[7].

3. 가상 머신간 파일 블록 공유

3.1 기본 아이디어

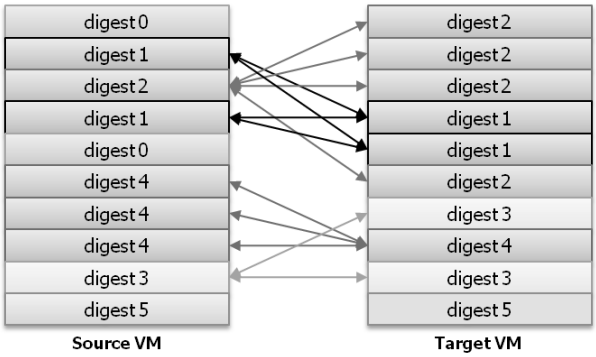
본 연구에서는 WAN 환경에서의 가상 머신 마이그레이션 비용을 최소화하기 위해 네트워크의 각 노드 상에 존재하는 가상 머신들의 이미지 파일과 마이그레이션 대상 가상 머신 이미지 파일간의 일치도 평가를 실시한다. 평가를 통해 얻어진 결과를 기반으로 우선 파일 블록(가상 머신 이미지 파일) 간의 일치도가 가장 높은 가상 머신이 존재하는 노드를 마이그레이션 대상으로 선정한다. 이를 통해 최대한 많은 양의 데이터를 원격지에 존재하는 가상 머신이 아닌 로컬 가상 머신으로부터 공유함으로써 네트워크를 통한 데이터 전송 비용을 최소화하는 것이다. 그림 4는 마이그레이션 대상 가상 머신 A와 다른 가상 머신간 일치도 평가를 통해 10%의 일치도를 지니는 가상 머신 C 대신 50%의 일치도를 지니는 가상 머신 B를 선택하고, 50%의 파일 블록을 로컬 가상 머신 B로부터 공유하는 모습을 보여주고 있다.



(그림 4) 기본 아이디어

3.2 파일 블록 일치도 비교 과정

본 연구에서는 보다 정확한 일치도 평가를 위해 모든 가상 머신 이미지의 크기를 생성시 3.2GB로 고정시켰으며, 파일 시스템 종류에 따른 영향을 최소화 할 수 있도록 이미지를 디스크 섹터 크기인 512바이트(디스크 섹터 크기) 단위로 나누었다. 이후 각 블록들에 대한 MD5 해시 값을 추출하여 가상 머신간 일치도 비교를 실시하였다.



(그림 5) 파일 블록 일치도 비교 과정

그림 5는 가상 머신간의 일치도를 전수 조사(brute-force)하는 과정을 도식화하고 있으며, 아래 표 1은 그림 5의 비교 결과에 따른 일치도 계산 과정을 나타내고 있다. 중복 되는 블록을 포함할 경우 70%의 파일 블록 일치도를 보여주고 있으며, 이는 70%의 파일 블록을 로컬 가상 머신으로부터 공유 가능함을 나타낸다.

<표 1> 파일 블록 일치도 계산 과정

	블록 개수	비고
전체 블록	10	0, 1, 2, 1, 0, 4, 4, 4, 3, 5
동일 블록 (중복 포함)	7	1, 2, 1, 4, 4, 4, 3
동일 블록 (중복 제거)	4	1, 2, 4, 3
블록 일치도 (중복 포함)	-	7/10*100=70%, 전송 필요 30%
블록 일치도 (중복 제거)	-	4/10*100=40%, 전송 필요 60%

4. 실험 및 평가

4.1 실험 환경

실험에는 인텔 코어 2 쿼드 CPU, 2GB 메모리, 500GB 하드디스크를 탑재한 데스크탑 PC가 사용되었으며, 일치도 평가를 위한 프로그램은 Ubuntu 10.04(Linux) 환경에서 작성 및 수행되었다. 실험은 표 2에 나타낸 것과 같이

2개의 그룹으로 나뉘어 수행되었으며, 그룹 1은 Ubuntu 9.04를 기준으로, 그룹 2는 Windows XP를 기준으로 일치도 평가를 수행하였다.

<표 2> 비교 그룹

그룹 1(Linux)		그룹 2(Windows, Linux)	
Ubuntu 9.04	Ubuntu 9.10	Windows XP	Ubuntu 9.04
	Ubuntu 10.04		Ubuntu 9.10
	Fedora 13		Fedora 13
	Centos 5.5		Centos 5.5

4.2 가상 머신간 파일 블록 일치도 평가 결과

표 3은 그룹 1의 파일 블록 일치도 평가 결과를 나타내고 있다. 같은 종류의 배포판이며 버전의 차이가 비교적 적은 Ubuntu 9.04 버전과 Ubuntu 9.10 버전이 22.85%의 가장 높은 일치도를 보여주고 있음을 볼 수 있다. 이는 기준 가상 머신 이미지(Ubuntu 9.04) 내부의 파일 블록 중복을 고려하지 않은 수치이며, 괄호 안에 나타낸 수치는 내부 중복을 포함한 일치 비율이다.

<표 3> 가상 머신간 파일 블록 일치도 (그룹1)

항목 / 대상	전체 블록수	동일 블록수	일치 비율(%)
9.04-9.10	3,819,652	872,923	22.85(32.37)
9.04-10.04		612,738	16.04(26.36)
13-9.04	4,031,102	226,514	5.62(15.47)
5.5-9.04	3,241,598	159,639	4.92(20.74)

9.04: ubuntu 9.04, 9.10: ubuntu 9.10, 10.04: ubuntu 10.04, 13: fedora 13, 5.5: centos 5.5

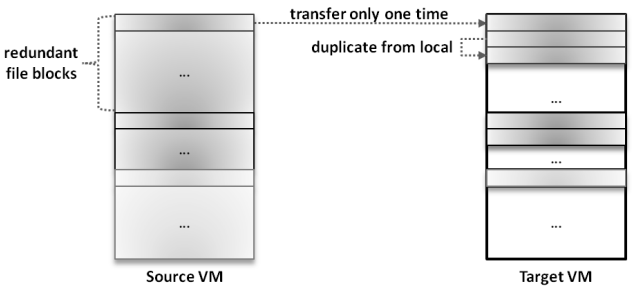
표 4는 Windows XP와 Linux 배포판간의 일치도 평가 결과를 보여주고 있으며, 모든 비교 대상에 대해 0에 가까운 일치도를 나타내고 있다. 하지만 내부 중복을 포함할 경우 상이한 종류의 운영체제 사이에서도 23.05%라는 비교적 의미 있는 수치의 일치도를 보이고 있다.

<표 4> 가상 머신간 파일 블록 일치도 (그룹2)

항목 / 대상	전체 블록수	동일 블록수	일치 비율(%)
xp-9.04	2,619,922	4,029	0.15(23.05)
xp-9.10		4,067	0.16(23.05)
xp-13		3,650	0.14(23.05)
xp-5.5		6,113	0.23(23.05)

xp: windows xp, 9.04: ubuntu 9.04, 9.10: ubuntu 9.10, 13: fedora 13, 5.5: centos 5.5

내부 중복 비율이 높다는 것은 그림 6과 같이 데이터 전송시 중복되는 데이터를 한번만 전송한 후 타겟 내에서 해당 데이터를 복제할 수 있는 비율이 그만큼 높다는 의미가 된다. 이를 통해 불필요한 중복 전송 블록 수를 줄임으로써 네트워크 비용을 절약할 수 있다.



(그림 6) 내부 중복에 의한 전송량 감소

5. 결 론

본 연구에서는 WAN 환경에서의 가상 머신 마이그레이션 성능 향상을 위한 아이디어를 제안하였다. 가상 머신간 파일 블록 일치도를 기반으로 로컬 가상 머신으로부터의 데이터 전송량을 최대화함으로써 네트워크를 통한 전송량을 최소화하고자 하였다. 실험 결과 비교 대상에 따라 최소 5%에서 최대 32%까지 파일 블록 공유가 가능하다는 것을 알 수 있었고, 이는 실험 대상 가상 머신들이 초기 설치 상태인 점을 감안할 때 앞으로 설치될 애플리케이션의 종류에 따라 더욱 많은 양의 데이터의 공유가 가능함을 보여주고 있다. 향후 본 연구의 아이디어를 실제 WAN 환경에서의 마이그레이션 프로세스에 적용한 플랫폼 개발을 진행해나갈 것이다.

참고문헌

[1] IBM Systems: Virtualization Version 2 Release 1, 2005.

[2] Michael Miller, “Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online,” Landmark Ltd.

[3] T. Wood, G. Tarasuk-Levin, P.Shenoy, P. Desnoyers, E. Cecchet, and M. Corner, "Memory Buddies: Exploiting Page Sharing for Smart Colocation in Virtualized Data Centers," In Proceedings of the 5th ACM Intl. Conference on Virtual Execution Environments, 2009.

[4] P Riteau, C Morin, T Priol, "Shrinker: Efficient Wide-Area Live Virtual Machine Migration using Distributed Content-Based Addressing," hal.inria.fr, 2010.

[5] Avi Kivity, Yaniv Kamay, Dor Labor, Uri Lublin, Anthony Liguori, "kvm: the Linux Virtual Machine Monitor," Linux Symposium, 2007.

[6] Inhyuk Kim, Taehyoung Kim, Young Ik Eom, "NHVM: Design and Implementation of Linux Server Virtual Machine using Hybrid Virtualization Technology," In Proc. of ICCSA, 2010.

[7] MD5, Wikipedia, http://ko.wikipedia.org/wiki/MD5.