라이브 가상 머신 마이그레이션에서 예측을 통한 효율적인 시간 절감 방법

김보섭*, 김성천*
*서강대학교 컴퓨터공학과
e-mail : kbsonic@sogang.ac.kr

An Efficient Method for Reducing Times using Prediction in Live Migration of Virtual Machine

Bo Seob Kim*, Sung Chun Kim*
*Dept of Computer Science & Engineering, Sogang University

요 약

클라우드 컴퓨팅 기술은 이용자로 하여금 자원을 편리하게 이용할 수 있게 하며 낮은 TCO(Total Cost of Ownership)를 보장하는 기술로 널리 사용되고 있다. 특히 IaaS에서 많은 기대를 할 수 있는데, 이 기술 중에 하나가 서버 가상화 기술이다. 가상화된 서버를 관리하기 위해서 쓰이는 기술 중 하나가 바로 라이브 가상 머신 마이그레이션으로 이는 서비스를 제공하고 있는 서버의 구동 중지 시간을 줄여 이용자의 불편함을 최소로 하는 기술이다. 그러나 라이브 가상 머신 마이그레이션은 기존의 방법보다 마이그레이션 시간이 길어 네트워크 자원을 오래 쓰는 단점이 있다. 이런 단점을 보완하기 위해 본 논문에서는 pre-copy 예측을 통한 마이그레이션 시간 감축 기법을 제안한다. pre-copy를 예측하는 기법을 소개하고 이를 이용한 통제 알고리즘을 보이며 이를 Xen에 적용했을 때의 결과를 보인다. 적용해본 결과 기존의 방법보다 마이그레이션 시간이 최대 17% 향상됨을 볼 수 있었다.

1. 서론

클라우드 컴퓨팅 기술은 이용자로 하여금 편리하게 자원을 이용할 수 있으며, 관리자 측면에서는 낮은 TCO (Total Cost of Ownership)를 보장하는 기술로 많은 사용을 하고 있다. 특히 IaaS(Infrastructure-as-a-Service)는 클라우드 컴퓨팅에서 서버 및 각종 호스팅 서비스를 편리하게 이용 및 관리하는 데에 중요한 기술로 작용하고 있다. 이러한 IaaS에서 중요한 기술이 바로 서버 가상화이다. 서버 가상화를 이용할 경우 이용자뿐만이 아니라 관리자 측면에서도 이득을 볼 수 있는데, 자원 및 비용 관리 그리고 고장 방지 시스템의 측면에서 상당한 이점을 보이고 있다. 관리자는 하나의 서버에서 여러 종류의 가상 머신(Virtual machine, VM)을 구동할 수 있고 이를 통합적으로 관리할 수 있다.[1][2] 가상 머신을 사용하여 이용자에게 유용성을 주기 위해서는 가상 머신을 이용자의 요구에 따라 관리할 수 있어야 한다. 이러한 가상 머신을 관리할 수 있는 기술 중 하나가 바로 가상 머신 마이그레이션(VM Migration)이다. 가상 머신 마이그레이션이란 하나의 물리 서버 위에 있는 가상 머신 관리자(Virtual Machine Monitor)에서 구동 되고 있는 가상 머신의 상태를 저장하여 이를 다른 가상 머신 관리자로 옮기는 것을 말한다. 가상 머신 마이그레이션은 신뢰도가 높게 설계되어 있어 복원한 가상 머신을 이용자가 이용하였을 때 오류가 일어날 확률이 줄어들고 이와 동시에 이전에 있던 가상 머신 관리자의 부담을 줄일 수 있다.

하지만 가상 머신 마이그레이션의 단점은 옮기는 과정에서 나타난다. 가상 머신을 다른 가상 머신 관리자로 옮기는 동안 필연적으로 가상 머신의 구동을 중지해야 하므로 그 시간 동안 이용자는 가상화된 서버의 응답을 받을 수 없는 불편함이 있다. 이용자가 사용함에 불편함을 최소화하기 위해 구동 중지 시간(downtime)을 최소화 하려는 기술이 바로 라이브 가상 머신 마이그레이션(Live VM Migration, 이하 라이브 마이그레이션)이다.[3] 라이브 가상 머신 마이그레이션은 보내질 대상이 되는 가상화된 서버가 이용자의 요청에 대한 응답을 주는 동안 받는 대상에서는 바로 사용할 수 있는 된 가상 머신을 준비하여 구동 중지 시간을 낮춘다. 라이브 마이그레이션은 이러한 강점 때문에 Xen[4]이나 VMWare[5] 상에서도 채용되고 있다. 라이브 마이그레이션 기술은 기존의 방법에 비해 응답 속도 면에서는 상당한 개선이 있는 방법이지만 방법의 특성상 기존의 방법보다 더 많은 전송량을 필요로 하며 전송 시간의 증가는 곧 네트워크 상에서의 통신량 증대로 이어진다. 늘어난 통신량은 가상화된 서버가 이용자의 요청에 응답하고 서버의 내용을 전송하는 데에 있어 상당한 방해 요소가 될 수 있다. 따라서 본 논문의 목표는 라이브 마이그레이션에서의 목표는 구동 중지 시간을 유지하거나

줄이면서 동시에 마이그레이션을 완료하는 데에 걸리는 시간(이하 마이그레이션 시간)을 줄이는 것이 된다. 본 논문에서는 라이브 마이그레이션의 단계에서의 주요 변인을 분석하고 그 변인을 예측하는 방법을 제안한다. 그리고 그 예측한 변인으로 라이브 마이그레이션을 통제하는 알고리즘 또한 기술한다.

본 논문의 구성은 다음과 같다. 1장에 서론에 이어 2장에서는 기존 라이브 마이그레이션의 실행 단계를 설명한 뒤 마이그레이션 시간에 영향을 주는 단계가 iterative pre-copy phase(반복적인 사전 복사 단계)임을 설명한다. 3장에서는 iterative pre-copy phase의 반복 횟수를 예측하는 방법 및 예측한 수치를 통한 반복 횟수를 통제하는 알고리즘을 서술할 것이며 4장에서는 이를 Xen에 적용한 시뮬레이션 결과를 보인다. 마지막으로 4장에서는 시뮬레이션을 토대로 한 결론을 도출해낼 것이다.

2. 라이브 마이그레이션의 실행 단계

라이브 마이그레이션은 여러 번의 메모리 복사, 특히 수정된 페이지(dirty page)의 복사로 구동 중지 시간을 줄이는 것이 특징이다. 라이브 마이그레이션의 아이디어의 핵심은 바로 그 반복성으로, 받는 대상에서 페이지를 받는 동안 보내는 대상에서 이용자의 요청으로 수정된 페이지를 가지고 있다가 이를 반복적으로 전송하여 나중에는 수정된 페이지의 양이 줄어들었을 때 가상 머신의 구동을 중지시키고 남은 페이지를 전송하는 단계를 거친다. 조금 더 자세히 말하자면, 라이브 마이그레이션의 실행 단계는 총 6단계로 나타낼 수 있다.[3]

1) Initialization : 보내질 대상이 되는 가상 머신이 선택되고 이를 받을 가상 머신 관리자에서는 이를 받아들일 새로운 가상 머신을 만든다.
2) Reservation: 받는 쪽의 가상 머신 관리자에서 새로 만든 가상 머신에 자원을 할당해준다.
3) Iterative pre-copy: 가상 머신이 보내지는 동안 셰도우 페이징이 활성화되어 가상 머신이 계속 이용자의 요청을 처리하는 동안 페이지는 대상 가상 머신 관리자에게 전송된다. 이 단계는 여러 번 수행되며 첫 번째 동작에서는 모든 메모리가 전송된다.
4) Stop-and-copy: 가상 머신을 구동 중지하고 남은 페이지를 전송한다.
5) Commitment: 받는 가상 머신 관리자에서 모든 전송이 끝났다는 응답을 준다.
6) Activation: 디스크 자원 등을 전송이 완료된 새로운 가상 머신에 이어준다.

1, 2, 5, 6단계에서는 내부 I/O를 요구하거나 약간의 메시지 밖에는 전송하지 않는다. 그러나 3, 4단계에서는 메모리의 내용을 메가바이트 단위, 많게는 기가바이트 단위로 전송하는 것이므로 마이그레이션 시간에 영향을 주는 주요 단계가 된다. 모든 메모리가 전송되는 첫 번째 pre-copy는 네트워크 속도를 향상시키지 않는 한 성능을 향상시키기 어렵다. 따라서 마이그레이션 시간을 줄이기 위해서는 두 번째 이후의 pre-copy 및 stop-and-copy 단계를 줄이는 방법을 찾아야 한다. stop-and-copy 단계에서의 시간도 pre-copy 단계의 영향을 받는 데, 이는 pre-copy의 반복을 어떻게 하느냐에 따라 남은 수정된 페이지의 수가 결정되기 때문이다.

3. 제안 기법

pre-copy가 마이그레이션 시간에 영향을 준다는 것을 설명하였다. 2. 의 과정을 거쳤을 때 '전송해야 하는 페이지의 수가 충분히 stop-and-copy 단계로 넘어갈 만큼 적다'면 다음 단계로 넘어간다. 우리기 그 회수를 예측할 수 있는 기준을 두어 pre-copy의 실행수를 줄일 것이다.

3.1. pre-copy의 반복 회수의 예측 기법

pre-copy의 반복 회수를 예측하기 위한 기법의 이해를 돕기 위해서 우선 iterative pre-copy가 어떤 방식으로 구동 중지 시 페이지 전송을 줄이는 지 조금 더 자세히 기술한다. 우선 네트워크의 전송 속도보다 페이지 수정 속도가 더 낮다고 가정한다. 또한 페이지 수정 속도가 고정 수치라고 가정한다. 가정대로라면 첫 번째 과정에서 모든 메모리의 내용을 복사하는 데 그 복사하는 동안 수정되는 페이지의 수는 모든 메모리의 용량보다는 낮을 것이다. 두 번째 과정에서 수정된 메모리의 내용을 복사하는 시간은 첫 번째 과정에서 보다 시간이 적게 걸리므로 두 번째 과정을 수행하는 동안의 수정된 페이지의 수는 첫 번째보다 더 적을 것이다. 이와 같은 과정을 반복하면 언젠가는 페이지 수가 다음 단계로 넘어갈 만큼 적어지게 된다. 위의 가정을 따랐을 때 다음 단계에서 보내야할 수정된 페이지의 개수를 식으로는 다음과 같이 나타낼 수 있다.(단, n은 0보다 큰 정수이다.)

$$D_{n+1} = D_n \times \frac{R \times C_{page}}{S_{Transfer}}$$

Dn은 n번째 pre-copy에서 보내져야 할 수정된 페이지의 수이고 R은 단위 시간당 수정이 일어나는 페이지의 개수, Cpage는 페이지의 용량이며 위의 식에서 분자를 곱하게 되면 단위 시간당 수정되는 메모리의 용량이 나오게 된다. Stransfer는 단위 시간당 전송 속도로 이것으로 나누어주게 되면 n번째 pre-copy에서 감소되어 전송해야 되는 메모리의 비율이 나온다. R이 고정값이라고 가정하였으므로 위의 점화식을 일반식으로 나타낼 수 있다. 이를 일반식으로 나타내면 다음과 같다.

$$D_n = D_1 \times (\frac{R \times C_{page}}{S_{Transfer}})^{n-1}$$

D_1은 initial value로, 첫 번째 단계에서 보내야하는 메모리의 양, 즉 메모리의 전체 용량으로 따졌을 때의 페이지 수이다. 충분히 적은 수의 페이지의 개수를 T라고 하자. 그렇다면 다음의 부등식이 성립하였을 때 stop-and-copy phase로 넘어갈 수 있게 된다.

$$T \geq D_1 \times (\frac{R \times C_{page}}{S_{Transfer}})^{n-1}$$

우리가 구하고 싶은 것은 위의 부등식이 성립하였을 때의 n이다. 이를 구하기 위해 식을 전개하면 다음과 같다.

$$\log(\frac{T}{D_1}) \geq (n-1) \times \log(\frac{R \times C_{page}}{S_{Transfer}})$$

이때 T < D_1 이므로(만약 T가 더 커지게 된다면 메모리를 첫 번째 단계에서 모두 복사하게 되므로 pre-copy 자체가 실행이 되지 않는다.) 이로 나누어 주게 되면 부등식이 반대가 된다. 이를 다시 전개하여 식으로 나타내면 예측 회수를 구할 수 있다.

$$n \geq \log_{\frac{R \times C_{page}}{S_{Transfer}}} (\frac{T}{D_1}) + 1$$

n은 0보다 큰 정수이므로 충분히 적은 페이지에서 다음 단계로 넘어가기 위한 최소 반복수 N은 다음과 같이 예측할 수 있다.

$$N = \left\lceil \log_{\frac{R \times C_{page}}{S_{Transfer}}} (\frac{T}{D_1}) + 1 \right\rceil$$

위에서 구한 N을 이용하면 우리는 언제 pre-copy를 멈춰야 할지 기준을 구할 수 있다.

3.2. 예측한 회수를 이용한 pre-copy 통제 알고리즘

위의 가정대로라면 N을 정확히 구하여 예측할 수 있다. 문제는 언제 이용자의 요청이 많고 적은지 알 수 없다는 것이다. 이는 일률적인 단위 시간당 수정된 페이지의 수를 얻을 수 없다는 뜻이 된다. 그래도 우리는 이전의 반복에서의 수정된 페이지의 수와 각 반복에서의 전송 시간을 얻을 수 있다. 이를 이용해서 우리는 평균을 구하여 N을 유추하는 방법을 사용한다. 단순하게, R의 값은 (각 단계마다 수정된 페이지의 합)/(각 단계마다 걸린 전송시간)으로 사용한다. 이용자의 요청이 갑자기 많아지는 경우 R의 평균값이 매우 증가하여 이로써 예측되는 N의 값도 커질 것이다. 이용자의 요청이 많아지면 앞으로 요청이 계속될 가능성도 많으므로 그 전에 pre-copy를 중단하고 다음 단계로 넘어가게 해주면 약간의 구동 중지 시간의 증가로 상당한 마이그레이션 시간의 절약이 있을 것이다.

또한 우리는 N이 0보다 작아질 수도 있는 경우에 대해서도 주의를 해야 한다. 수정되는 페이지의 수가 너무나도 많아 전송 속도를 뛰어 넘는 경우 log의 전체는 음수 값이 되어 버린다. 이 경우에는 바로 pre-copy를 중지하고 stop-and-copy phase로 넘어가는 것이 마이그레이션 시간 및 구동 중지 시간을 절감하는 데 도움이 된다. 이를 토대로 알고리즘을 구성하면 다음과 같이 나타낼 수 있다.

```
T<-Threshold of going next phase;
C<-size of page;
D1<-(memory size)/C;
TP<-log(T/D1); // 계산 속도의 절감을 위함
N<-Limitation of iteration; // 정의해야함
PRC<-1; // PRC는 행해진 반복의 횟수
transfer whole memory;
S<-(memory size)/(transfer time);
P[1]<-(number of dirtied pages in first iteration);
T[1]<-(transfer time);
while(PRC < N)
{
    RR<-0;
    TT<-0;
    for(i=1;i<=PRC;i++)
    {
        RR+=P[i];
        TT+=T[i];
    }
    RR/=TT;
    temp<-TP/log((RR*C)/S)+1;
    if(temp > N || temp<=0)
        break;
    else
    {
        N=temp;
        PRC++;
        transfer dirtied pages; // next iteration
        P[PRC]=(number of dirtied pages for PRC-th iteration);
        T[PRC]=(transfer time of PRC-th iteration);
    }
}
```

<표 1> 회수 예측을 이용한 pre-copy 통제 알고리즘

사전의 설정 값은 다음 단계로 넘어가게 하기 위한 최소 페이지 수, 반복의 최대 수 (적당히 큰 수면 된다.)이며 나머지는 pre-copy 단계를 통하여 얻을 수 있는 수치이다.

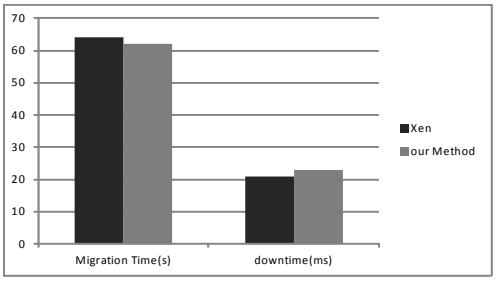
4. 시뮬레이션

시뮬레이션 환경은 <표 2>와 같이 설정하였다.

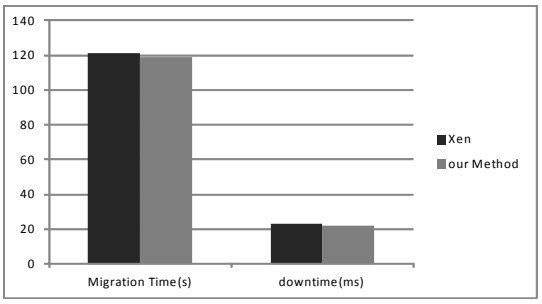
OS	CPU	Storage RAM	network environment
Xen	Core2duo E6300	DDR3 8GB	Speed of 100Mbps

<표 2> 시뮬레이션 환경

Storage의 경우 NFS 3.0을 이용하여 SSD를 맞물려 NFS를 구성하여 메모리의 복사시간만을 사용하게끔 하였다. 컴퓨터는 같은 LAN상에 존재하게 하였고 RAM 크기를 512MB, 1024MB로 하여 측정하였다. I/O가 자주 일어나는 경우와 자주 일어나지 않는 경우를 비교하기 위해 환경을 커널 컴파일 상태 및 유휴 상태로 놓고 기존의 Xen에서 쓰는 방법과 비교하였다.

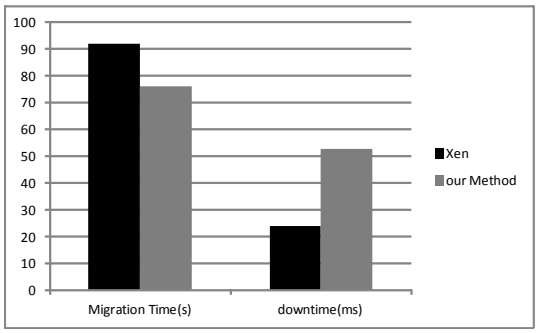


<그림 1> 512MB VM 유휴 상태에서의 결과

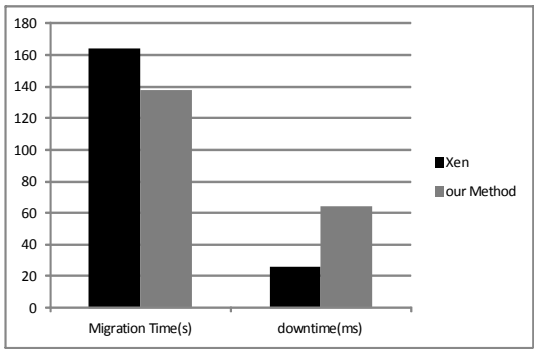


<그림 2> 1024MB VM 유휴 상태에서의 결과

유휴 상태에서는 전반적으로 마이그레이션 시간의 큰 하락 폭이 없는 데 그 이유는 Xen에서 pre-copy를 종료하는 조건이 충족되는 반복 단계와 예측한 단계가 같았기 때문이다. 512MB VM에서는 오히려 downtime이 늘어난 것을 볼 수 있다. 이는 오차범위 이내이다.



<그림 3> 512MB VM 커널 컴파일 상태에서의 결과



<그림 4> 1024MB VM 커널 컴파일 상태에서의 결과

반면 커널 컴파일상에서 예측 기법을 적용한 결과는 마이그레이션 시간에 있어 효율적으로 나타났다. 적용된 방법은 마이그레이션 시간에 대해 512MB VM에서는 17%, 1024MB VM에서는 14%의 성능 향상이 있었다. 그러나 커널 컴파일 과정은 stage가 줄면서 구동 중지 시간 측면에서는 늘어났다. 커널 컴파일의 경우 꾸준한 파일 I/O가 일어나면서 페이지가 수정되므로 N의 값을 가지고 있는 도중에 I/O가 늘어나는 데에서 stage의 중지가 일어났다. 기존 Xen에서 수행해야 했던 반복 과정 동안에서의 마이그레이션 시간이 없어진 대신 구동 중지 시간이 늘어난 것이다.

5. 결론 및 향후 연구

시뮬레이션의 결과는 I/O가 많이 일어나는 환경에서 향상된 마이그레이션 시간을 가져왔다. 그러나 이는 구동 중지 시간의 향상을 가져왔다. 이는 인자가 적어 생기는 요소라 판단한다. 본 논문에서는 다음 단계로 넘어가게 하기 위한 최소의 페이지 수를 고정으로 놓고 실험했었으나 이를 네트워크 속도에 따라 유연하게 조정할 수 있게끔 방법을 수정하면 구동 중지 시간이 많은 폭으로 늘어나지 않으면서 마이그레이션 시간을 절약할 수 있을 것이다. 아울러 pre-copy 회수의 정확도는 I/O 이벤트에 대한 자료가 충분하다면 더 증가할 것이다. 실제로도 I/O 이벤트를 기록하여 성능을 향상한 논문이 있다.[6] 마지막으로 위의 실험은 LAN 상에서의 실험으로 WAN으로 넘어갔을 경우 스토리지도 고려해주어야 한다. 이는 향후 진행되어야 할 것이다.

ACKNOWLEDGMENTS

이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2012R1A1A2009558)

참고문헌

[1] Virtual Desktop Infrastructure. [Online] Available : http://www.vmware.com/pdf/virtual_desktop_infrastructure_wp.pdf

[2] The Virtualization Gurus. [Online] Available : http://thevirtualizationgurus.com/

[3] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, “Live migration of virtual machines,” in *Proceedings of the Second Symposium on Networked Systems Design and Implementation (NSDI’05)*, 2005, pp. 273 - 286.

[4] 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",in *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. ACM Press*, 2003, pp. 164-177

[5] VMWare, vmware inc, [Online]. Available :http://www.vmware.com/

[6] H Liu, H Jin, X Liao, L Hu and C Yu, “Live migration of virtual machine based on full system trace and replay” in *Proceedings of the 18th ACM international symposium on High performance distributed computing,* 2009, pp 101-110