데이터 압축 알고리즘을 이용한 빠른 가상 머신 저장 및 복원 기법

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

A Fast Virtual Machine Saving And Restoring Method Using Data Compression Algorithm

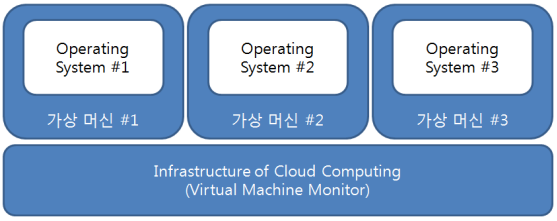
Bo Seob Kim*, Sung Chun Kim*
*Dept of Computer Science & Engineering, Sogang University

요 약

가상화 기술은 각종 장비들을 편리하게 이용하게끔 하고 있으며, 특히 데스크탑 가상화 기술의 경우 클라우드 컴퓨팅에 의한 유저의 요구가 많아져 상당한 연구가 진행되고 있다. 데스크탑 가상화의 경우 사용자에게도 큰 이점이 있지만 관리 측면에서 Cost를 줄이는 데 일조하고 있다. 데스크탑 가상화는 관리자 측면에서 보았을 때 사용자의 요청에 따라 가상 데스크탑을 빠르게 제공하여야 하는 의무가 있다. 관리자가 가상 데스크탑을 빠르게 제공하기 위해서 미리 구축해 둔 가상 머신의 상태를 저장하는 데, 본 논문에서는 가상 데스크탑의 상태를 저장하고 제공하는 데 있어서의 효율을 위해 압축 알고리즘을 사용하는 방법을 제안한다. 가상 데스크탑의 상태가 메모리 명령어의 집합이라는 특성에 따라 무손실 압축 알고리즘을 사용하고, 그들 중 빠르고 압축률이 높기로 알려진 네 가지 알고리즘을 찾아 비교하였다. 비교 결과 가장 효율이 좋았던 LZO Algorithm을 메모리 상태 저장/복원의 중간 과정에 삽입하였다. 실험 결과 기존의 압축하지 않았던 방법보다 최대 1.7배의 성능 향상을 보였으며 가상 머신의 상태를 저장했을 때의 평균 54%의 용량 절감이 있었다.

1. 서론

가상화 기술은 각종 장비들을 편리하게 이용하게끔 하고 있으며, 특히 데스크탑 가상화 기술의 경우 클라우드 컴퓨팅과 맞물려 연구가 상당부분 진행되고 있으며 이러한 부산물들이 제품화 및 상용화가 되어 많은 회사에서 이용 및 제공하고 있다.[1] 데스크탑 가상화는 Resource 관리 및 Cost 관리에 있어 많은 이점을 가지고 있으며, 이로 인하여 낮은 TCO(Total Cost of Ownership)을 가지고 있다. 이와 동시에 사용자 입장에서는 기존의 데스크탑을 원격으로 접속하는 것과 같이 사용되며 기존 데스크탑의 어플리케이션을 이용하는 데 있어 편리함을 가지게 되는 동시에 관리자 입장에서는 하나의 서버에서 여러개의 가상 머신(Virtual Machine, VM)을 구동할 수 있고, 그것을 통합적으로 관리할 수 있다는 것이다.[2] <그림 1>은 그것의 개요도를 보여주고 있다.



<그림 1> 데스크탑 가상화의 개요도.

데스크탑 가상화는 사용자에게도 큰 이점이 있지만 이 논문에서 주목하고자 하는 것은 관리자 측면에서의 이점을 최대한 살리는 것이다. 관리자 측면에서 가상화 기술의 장점이라고 한다면 서버 관리에 용이하다는 것인데 그 말을 자세히 기술한다면 VM에 이상이 생겼을 시 그것을 빠르게 복원이 가능하고, 새로운 사용자의 요청이 들어왔을 때 물리적 데스크탑을 세팅하는 것에 비해서 가상 데스크탑을 빠르게 제공 가능하다는 것이다. 이를 제공하는 기술이 바로 가상 머신 마이그레이션(Virtual Machine Migration)이다. 가상 머신 마이그레이션이란 VM의 상태를 저장하거나 복원하는 것을 말한다. 가상 머신 마이크레이션의 신뢰도가 높으면 높을수록 복원한 VM을 사용자에게 제공하였을 때 오류가 일어날 확률이 줄어들고 저장/복원 시간이 빠르면 빠를수록 사용자의 요청이 들어왔을 때 불편함이 최소화 된다. 이러한 신뢰도를 높이거나 저장 및 복원 시간을 줄이려는 노력으로 나온 기술이 있다. 라이브 마이그레이션(Live Migration)[3][4]이 바로 그것이다. VM이 구동되고 있는 상태에서 다른 새로운 VM으로써 복원하는 기술인 이것은 NFS(Network File System)과 맞물려 많이 이용되고 있는 기술이다. 그러나 NFS는 당초 네트워크를 사용한 파일 저장 방법이기 때문에 네트워크 속도에 의존적일 수밖에 없고, 따라서 상당한 병목현

상이 일어날 상황에 도출 될 것이다. 우리가 초점을 맞추려고 하는 것은 네트워크 속도보다 상대적으로 빠른 내부 I/O속도를 이용하여 VM의 상태를 효율적으로 저장 및 복원하는 방법이다. 본 논문에서는 그 방법을 압축 알고리즘의 사용이라고 보고 논지를 건개해 나갈 것이다.

본 논문의 구성은 다음과 같다. 1장에 서론에 이어 2장에서는 압축 시간 및 압축률, 전송 시간과의 관계, 압축 알고리즘의 선택 기준 및 선택을 할 것이며 3장에서는 이를 실제 가상 머신에 적용한 결과, 그리고 마지막 4장에서는 결과를 토대로한 결론을 도출해낼 것이다.

2. 제안 기법

2.1. 압축 시간 및 압축률, 전송 시간과의 관계

일반적으로 압축 시간은 압축파일을 메모리에 올릴 때 걸리는 시간, 메모리에 올라간 바이트 스트림을 또 다른 바이트 스트림으로 압축하는 시간, 압축된 바이트 스트림을 저장하는 시간의 세 가지 구성 요소로 이루어져 있다. 이것을 식으로 나타내면 다음과 같다.

$$T_{Comp} = IOTime_{input} + CompTime + IOTime_{Output} \quad (1)$$

여기서 IOTime(input)은 입력 파일에서 메모리에 올리는 전송 시간, CompTime은 압축 알고리즘 자체로 인한 압축 시간, IOTime(output)은 압축된 바이트 스트림이 파일로 써 써지는 Output 시간을 나타낸다. 여기서 첫 번째 인자인 IOTime(input)의 경우 메모리 저장의 특성상 0초가 된다. 이를 기준으로 식을 다시 쓰게 되면 다음과 같게 된다.

$$T_{Comp} = CompTime + IOTime_{Output} \quad (2)$$

압축 시간의 경우 압축 대상이 되는 용량에 비례하며 압축 전송률은 거의 모든 파일에 대해 비슷하다. 시간을 구하려면 용량을 시간대비 전송 용량으로 나누어 주면 된다. 또한 파일 시스템에 쓰는 Output 시간은 압축 된 용량을 전송률로 나눠주면 된다.(이때 저장 및 복원되는 용량은 MB급에서 GB급까지 용량이 충분히 크므로 탐색 시간과 디스크 회전 지연시간은 무시할 수 있다.) 이를 다시 한 번 식으로 써주면 다음과 같이 구성할 수 있다.

$$T_{Comp} = \frac{Filesize}{Speed_{Comp}} + \frac{Filesize * Rate_{Comp}}{Speed_{Transfer}} \quad (3)$$

Rate(Comp)는 파일 압축시의 압축률로, 이를 Filesize에 곱해준다면 압축된 실제 용량이 된다. Speed(Comp)는 압축 알고리즘 적용 시 전송률이고, Speed(Transfer)는 파일을 전송할 시의 전송률이다. 위의 식을 Filesize로 묶어낸다면 Filesize, 즉 파일 용량 이외의 통제 변인은 압축 알고리즘 적용 시 전송률, 파일을 전송할 시의 전송률, 그리고 압축률이 된다. 파일의 전송률은 NFS의 특성상 고정되므로 주요 변인은 압축 속도와 압축률이 된다. 압축을 이용한 저장이 유용하게 사용 되려면 다음과 같은 식이 성립해야만 한다.

$$\frac{Filesize}{Speed_{Transfer}} \geq Filesize\left(\frac{1}{Speed_{Comp}} + \frac{Rate_{Comp}}{Speed_{Transfer}}\right)$$

$$\frac{1}{Speed_{Transfer}} \geq \frac{1}{Speed_{Comp}} + \frac{Rate_{Comp}}{Speed_{Transfer}}$$

$$\frac{1 - Rate_{Comp}}{Speed_{Transfer}} \geq \frac{1}{Speed_{Comp}}$$

$$\frac{Speed_{Transfer}}{1 - Rate_{Comp}} \leq Speed_{Comp}$$

$$Speed_{Transfer} \leq Speed_{Comp} * (1 - Rate_{Comp}) \quad (3)$$

Rate(Comp)의 수치가 낮으면 낮을수록(압축률이 높아질수록), 압축 속도가 높으면 높을수록 상태 저장을 할 때 압축을 수행하였을 시의 효율이 증대됨을 알 수 있다. 압축 해제 시간의 경우에도 실제로 영향을 미치는 것은 압축 속도 대신 압축 해제 속도이므로 다음과 같은 식으로 해석할 수 있다.

$$Speed_{Transfer} \leq Speed_{Decomp} * (1 - Rate_{Comp}) \quad (4)$$

따라서 압축 알고리즘 선택 시 고려해야 하는 부분은 압축 속도, 압축 해제 속도, 그리고 압축률이다. 가령 100Mbps의 네트워크를 가정했을 때 Speed(Transfer)가 12.5MB/s이기 때문에 식 (3)의 우항 및 식 (4)의 우항, 즉 압축 시 기대할 수 있는 전송 속도 값은 12.5MB/s 이상이 나와야 하고, 1Gbps의 네트워크를 가정했을 때에도 전송 속도 값은 125MB/s 이상이 나와야 의미가 있다.

2.2. 압축 알고리즘 및 압축 알고리즘의 선택

가상화 및 OS 메모리 상에서의 압축은 명령어의 반복적인 sequence인 특성이 있다. 그렇기 때문에 압축률이 좋아지므로 어떠한 알고리즘을 쓰더라도 성능이 좋아질 것이라 예측할 수 있다.

압축 알고리즘은 메모리를 다루는 만큼 무손실 압축 알고리즘이 선택되어야 한다. 무손실 압축 알고리즘의 대표적인 예로는 Runlength Encoding(RLE), Huffman Coding[4], LZ 계열 알고리즘(대표적으로 LZO[5], LZW, LZ77 등) 등이 있다. 상태 저장 및 복원을 수행 시 압축 알고리즘을 수행하려는 시도가 이루어졌었다.[6] 우리는 이와는 다른 압축 알고리즘을 사용하면 더 나은 결과를 얻을 수 있을 거라 판단하여 그쪽으로 접근을 시도하였다.

우선 RLE의 경우에는 압축 전송률이 빠르고 시간복잡도가 낮다는 장점이 있지만 특성상 명령어 패턴이 연속적으로 일어난다고 가정했을 때에도 같은 바이트 스트림이 연속적으로 있을 확률이 매우 낮으므로 여기서는 논외로 한다. 여러 알고리즘 중 빠르고 standard로 알려진 4개의 알고리즘(quickLZ[7], LZO, huffman, deflate(huffman + LZ77))으로 비교 분석하여 가장 효율성이 좋은 알고리즘을 선택하고, 이를 기존의 Xen과 비교할 것이다. 대상 파일은 메모리의 가장 비슷한 성질을 가지는 실행 파일(9MByte)을 기준으로 파일을 메모리에 올린 시간을 제외한 나머지 시간에 대해서 이를 측정하였다. 측정된 결과는 <표 1>과 같다.

알고리즘	압축률	압축 속도	압축 해제 속도
quickLZ	45.71%	76.92MB/s	76.94MB/s
LZO	47.43%	105.76MB/s	141.02MB/s
huffman	61.62%	40.29MB/s	120.88MB/s
deflate	33.49%	13.22MB/s	120.88MB/s

<표 1> 각 압축 알고리즘의 비교

압축된 파일의 전송 속도가 일정하다고 가정한다면, 식(3)(4)에서 우항이 가장 높은 값을 가지는 경우 유용한 것이라 본다. 이는 압축시 기대할 수 있는 전송속도로, 식(3)(4)를 참고하여 전송속도 값을 구했을 경우 그 값은 <표 2>와 같다.

알고리즘	식 (3)의 우항 (압축 시 전송속도)	식(4)의 우항 (압축 해제시 전송속도)
quickLZ	41.76MB/s	41.77MB/s
LZO	55.60MB/s	74.13MB/s
huffman	15.46MB/s	46.39MB/s
deflate	8.79MB/s	80.40MB/s

<표 2> 기대할 수 있는 전송속도

deflate를 제외한 나머지 압축 알고리즘은 100Mbps 네트워크 기준의 12.5MB/s를 웃도는 값을 보였다. deflate의 경우 파일을 만들어 저장하는 데 시간이 상당히 오래 걸렸지만, 압축 해제 시간은 우수하였다. 저장 및 복원 시간을 보았을 때 위 네 가지 중 가장 우수한 것은 LZO이다. 따라서 LZO를 이용해 가상 머신을 저장하고 복원하는 데 사용하면 성능 향상이 있을 것이다.

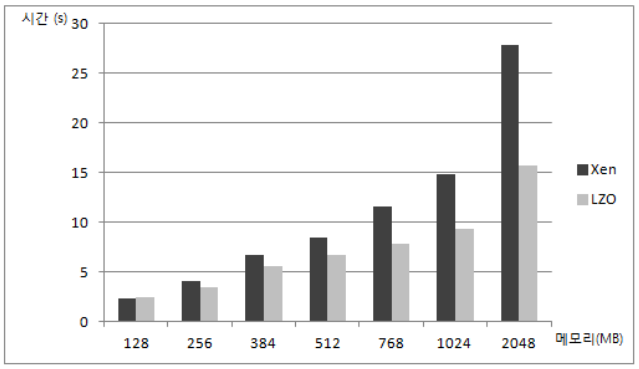
3. 시뮬레이션

시뮬레이션 환경은 <표 3>과 같이 설정하였다.

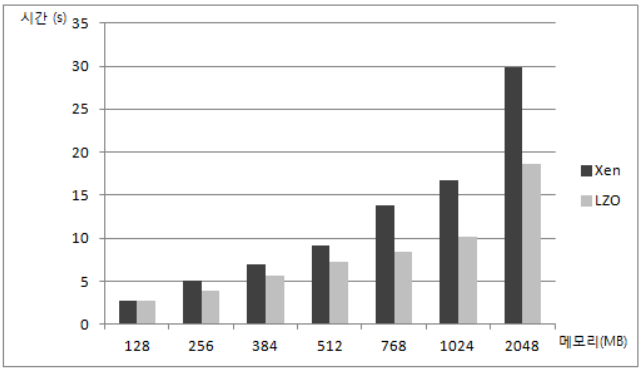
OS	CPU	Storage	RAM	network environment
Linux With Xen 3.1.0	Core i3 540	Samsung SSD 830 128GB	DDR3 8GB	Speed of 1Gbps

<표 3> 시뮬레이션 환경

Storage의 경우 NFS 3.0을 이용하여 SSD를 맞물려 NFS를 구성하였다. Xen에는 2개의 VCPU 및 다양한 크기의 RAM 크기를 가지도록 가상 머신을 설정하였다.(128MB, 256MB, 384MB, 512MB, 768MB, 1024MB, 2048MB) Xen의 save/restore 옵션과 저장 및 복원하는 과정에서 LZO 압축 알고리즘을 삽입한 Lsave/Lrestore 명령을 만들어 저장 속도 및 복원 속도를 비교하였다. 각 명령어를 다른 메모리크기 마다 10번씩 수행하게끔 하고, 시간을 재어 그 평균을 구하였다. 실험을 수행한 결과, 다음과 같이 실험 결과가 나왔다.



<그림 2> save 연산에 대한 연산 수행 결과



<그림 3> restore 연산에 대한 연산 수행 결과

전반적으로 상태를 저장하는 시간 보다 상대를 복원하는 시간이 더 높게 나타났다. 메모리 저장 용량이 128MB였을 경우 오히려 제안한 방법이 시간이 높았고, 점점 배수가 벌어져 2048MB의 경우 저장에서 약 1.78배, 복원에서 약 1.6배의 성능 향상이 있었다. 128MB의 경우에서 제안한 방법에 대해 압축 및 복원 시간이 높았던 이유는 메모리 용량이 낮았을 때에 I/O시간이 짧아지고 압축 I/O속도 이외의 CPU time의 영향이 커지기 때문이다. 이외에서는 Storage의 속도가 네트워크 속도인 1Gbps를 넘어가기 때문에 압축 및 저장하는 과정에서의 I/O 속도가 향상치를 보이고 압축했을 때의 줄어드는 용량 또한 커지기 때문에 저장하고 있는 메모리가 높을수록 더욱 효율을 보이게 된 것이다. 저장되는 용량 또한 평균적으로 54%의 압축률을 보여주었다. 이 측면은 상태를 조금 더 많이 저장할 수 있다는 면에서 용이하게 된다.

4. 결 론

시뮬레이션의 결과에서와 같이 압축 알고리즘을 사용함으로써 가상화 환경 상에서의 기존의 가상머신의 저장 및

복원보다 성능이 좋아진다는 것을 보였다. 효율 향상은 최대 1.6배의 복원 시간과 54%의 공간 절감율을 보여 명료했다고 볼 수 있다. 하지만 이 실험에서도 한계점은 있다. Local Storage의 속도가 네트워크 속도보다 더 빠른 경우에서만 시뮬레이션이 수행되었으며, 그 반대의 경우에서는 수행하지 못하였다. 또한 가상 머신의 상태 저장 및 복원에서 그 기능이 증명된 라이브 마이그레이션과의 성능 비교는 이루어지지 않았다. 라이브 마이그레이션은 상태를 미리 다른 저장공간에 저장할 필요가 없으므로 어느 측면에서는 성능이 더 좋을 수 있기 때문에, 이것과의 비교가 필요하다. 또한 압축 알고리즘도 저 네 가지가 끝이 아니라 다른 알고리즘도 많을 것이다. 지금은 압축을 하였을 시의 효율성을 보였기 때문에 가상화 환경, 즉 메모리의 명령어 sequence 상에서 가장 효율적인 압축 알고리즘이 무엇이 되겠는가에 대한 연구 또한 중요하다.

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] D. A. Huffman, “A method for the construction of minimum redundancy codes,” in *Proceedings of the Institute of Radio Engineers*, 1952, pp. 1098 - 1101.

[5] LZO real-time data compression library. [Online] Available : http://www.oberhumer.com/opensource/lzo/

[6] Li Deng, Hai Jin, Song Wu, Xuanhua Shi, Jiangfu Zhou, “Fast Saving and Restoring Virtual Machines with Page Compression” in *2011 International Conference on Cloud and Service Computing*; 2011, pp. 150 - 157

[7] QuickLZ real-time data compression library. [Online] Available : http://www.quicklz.com/index.php