가상 머신을 위한 인-메모리 페이지 스왑 기법의 구현

이치영 ○, 장영준, 권재환, 유혁

고려대학교 컴퓨터전파통신공학과

{cylee, yjjang, jhkwon, hxy}@os.korea.ac.kr

Implementation Of In-Memory Page Swapping For The Virtual Machine

Chiyoung Lee○, YoungJun Jang, Jae-Hwan Kwon, Chuck Yoo

Department of Computer and Radio Communications Engineering, Korea University

요 약: 가상화 환경에서는 다수의 가상 머신들이 메모리를 분할하여 사용하기 때문에 메모리 부족 문제가 빈번하게 발생한다. 기존에 사용하던 페이지 스왑 기법은 낮은 디스크 접근 속도와 가상화 환경의 분할 드라이버 모델로 인해 가상 머신의 성능을 크게 저하시킨다. 따라서 성능 저하를 유발하지 않으면서 메모리 부족 문제를 해결할 수 있는 기법이 필요하다. 이를 위해 본 논문은 가상 머신을 위한 인-메모리 페이지 스왑 기법을 제안하고 구현한다. 이는 페이지 스왑의 성능 향상을 위해 메모리 안에 스왑 공간을 위치시킨다. 또한, 더 많은 여유 공간을 얻기 위해서 다른 가상 머신으로의 스왑 공간 이전과 페이지 압축을 제공한다. 이 기법을 통해 본 논문은 기존 인-메모리 스왑 기법보다 평균 32% 더 많은 메모리 공간을 얻을 수 있다.

1. 서 론

가상화 환경은 소프트웨어적인 자원 분배를 통해 제한된 물리 시스템에서 다양한 컴퓨팅 환경을 제공할 수 있다. 그러나 가상 머신은 컴퓨팅 자원을 다른 가상 머신과 공유하기 때문에 가상 머신의 수가 증가할수록 성능이 떨어진다. 특히, 메모리는 각 가상 머신이 분할하여 사용하고 서로 간의 접근을 막기 때문에 자원 부족 문제가 심각하다.

메모리 부족 문제를 해결하기 위해 기존에는 페이지 스왑(page swapping)이 사용되었다. 페이지 스왑은 프로세스가 사용하지 않는 페이지를 메모리에서 내보내고, 프로세스가 요구한 새로운 페이지를 메모리에 적재한다. 그러나 페이지 스왑은 디스크에 사용하지 않는 페이지를 저장하기 때문에 실행 속도가 느리다. 게다가 각 가상 머신은 분할된 작은 크기의 메모리를 이용하기 때문에 발생 빈도가 잦다. 이는 가상 머신의 성능을 크게 저하시키는 요인이 된다. 따라서 성능에 영향을 주지 않으면서 메모리의 여유 공간을 효과적으로 늘릴 수 있는 기법이 필요하다.

이를 위해 본 논문은 가상 머신을 위한 인-메모리 페이지 스왑(In-Memory Page Swapping) 기법을 제안한다. 이는 스왑 공간(swap space)으로 디스크 대신 메모리를 이용함으로써, 페이지 스왑으로 인한 성능 저하를 막을 수 있다. 또한, 스왑 공간의 메모리 사용을 최소화하기 위해 저장할 페이지를 압축하고, 관리 가상 머신으로 스왑 공간을 이전한다. 일반적인 페이지 스왑 기법은 디스크를 스왑 공간으로 사용하지만, 몇몇 기법은 디스크 접근으로 인한 성능 저하를 줄이기 위해서 디스크 대신에 메모리를 스왑 공간으로 사용한다[1][2][3]. 하지만 가상 머신에서의 인-메모리 페이지 스왑은 해당 가상 머신에게 스왑 공간으로 인한 추가적인 메모리 압박을 준다. 따라서 본 논문은 이 압박을 줄이기 위해 두 가지의 방법을 사용한다. 첫번째는 스왑 공간 자체의 크기를 작게 유지하기 위해 압축 기법을 사용한다. 압축은 스왑 공간에 저장할 페이지의 크기를 줄임으로써, 스왑 공간의 크기를 줄인다. 예를 들어, 하나의 4KB 페이지를 위한 메모리가 필요한 경우 4KB의 스왑 공간이 필요하지만, 압축률이 50%인 압축 기법을 사용하면 2KB만 필요하다. 두번째로, 메모리에 충분한 여유가 있는 관리 가상 머신으로 스왑 공간을 이전한다. 이는 메모리 부족을 겪고 있는 가상 머신의 메모리에서 스왑 공간을 제거했기 때문에 기존의 인-메모리 페이지 스왑에 비해 더 많은 여유 공간을 얻을 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 인-메모리 페이지 스왑 기법의 특징을 설명하고, 3장에서는 기법의 구현 환경에 대한 설명과 실험 결과를 보인다. 마지막으로, 4장에서는 향후 연구 방향에 대해 논하면 결론을 맺는다.

2. 스왑 공간 이전과 인-메모리 페이지 스왑의 구조

이 장에서는 인-메모리 페이지 스와핑의 구조를 설명한다. 본 논문은 이 기법을 통해 가상 머신 내에서 페이지 스왑의 성능 향상과 메모리 사용량 감소를

"이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2010-0029180)."

달성하고자 한다. 각 가상 머신은 물리적인 메모리를 분할하여 사용하기 때문에 가상 머신의 수가 증가하면 각 가상 머신이 갖는 메모리의 크기가 줄어든다. 이는 잦은 페이지 스왑을 발생시켜 각 가상 머신의 프로세싱 능력을 감소시킨다.

제안하는 인-메모리 페이지 스왑 기법은 "인-메모리 스왑 공간"과 "스왑 공간 이전", "압축"의 세 축으로 구분할 수 있다. 인-메모리 스왑 공간은 메모리에 스왑 공간을 위치시키는 방법이다. 기존의 페이지 스왑은 디스크를 스왑 공간으로 활용하였다. 그러나 디스크는 메모리에 비해 접근 속도가 매우 느린 단점이 있다. 비가상화 환경에서는 이 성능 저하를 해결하기 위해 인-메모리 스왑 공간을 연구해 왔다[1][2][3]. 게다가 가상화 환경은 장치를 여러 가상 머신 간 공유 지원과 간섭 최소화를 위해 분리 드라이버 구조(split driver model)로 되어 있다. 이는 구조적으로 디스크에 대한 직접적인 접근을 막아 악의적인 가상 머신의 공격으로부터 시스템을 보호하기 위함이다. 하지만 이 구조로 인해 가상 머신에서의 페이지 스왑 성능이 더 크게 저하된다. 따라서 본 논문도 페이지 스왑의 성능 저하를 막기 위해서 디스크 대신 메인 메모리를 스왑 공간으로 사용한다.

하지만 가상화 환경에서는 각 가상 머신이 이용가능한 메모리의 크기가 작기 때문에 스왑 공간에 의해 추가적인 메모리 압박이 생길 수 있다. 비가상화 환경에서는 하나의 물리 메모리를 하나의 시스템이 사용하기 때문에 충분한 메모리가 제공된다. 따라서 자신의 메모리 내에 스왑 공간을 생성해도 압축만으로 메모리 압박을 크게 줄일 수 있다. 그러나 가상화 환경은 많은 가상 머신이 물리 메모리를 분할하여 사용하기 때문에, 각 가상 머신이 가진 메모리의 크기가 작다. 따라서 본 논문은 인-메모리 스왑의 성능을 그대로 유지하는 동시에 스왑 공간으로 인한 메모리 압박을 제거하기 위해 스왑 공간을 다른 가상 머신으로 이전한다. 이전 연구를 통해 우리는 인-메모리 페이지 스와핑을 위한 스왑 공간 설계를 제안하고, 그 효율성을 입증하기 위한 사전 실험 결과를 보였다[4]. 스왑 공간을 대상 가상 머신의 외부로 이동시키기 때문에 페이지 공유처럼 페이지의 크기에 해당하는 여유 공간을 얻을 수 있다. 또한, 이 저장소를 모든 가상 머신이 공유하게 함으로써, 물리 메모리 내의 스왑 공간의 비중을 가상 머신의 수와 관계없이 일정하게 유지할 수 있다.

그림 1은 다른 가상 머신으로 이전한 스왑 공간을 나타낸다. VM1과 VM2는 각각 VM0의 메모리 내에 스왑 공간을 가지고 있다. 이를 위해 페이지 스왑이 필요한 가상 머신(VM1, VM2)은 여유 메모리를 갖고 있는 다른 가상 머신(VM0)에게 스왑 공간을 위한 메모리 공간을 요청하여 빌려온다. 이 그림에서 VM0는 관리 가상 머신으로 다른 가상 머신들을 관리하는 역할을 한다. 따라서 불필요한 응용프로그램이 동작하지 않는다. 또한, 관리 가상 머신은 커널 로드에 관련된 하이퍼바이저 코드와 라이브러리, 도구들을 실행해야 하기 때문에 요구 메모리량이 많지만, 이들 대부분이 부팅이 끝난 후에는 사용되지 않기 때문에 메모리에 충분한 여유 공간이 있다. 따라서 우리는 이 여유 공간을 스왑 공간으로 활용한다. 이를 통해 VM0, 1은 다른 VM0로 스왑 공간을 이동시켜 스왑 공간 유지로 인한 메모리 사용량을 제거할 수 있다.

또한, 압축을 통해 VM0가 내어준 공간의 크기도 최소화할 수 있다. 압축은 사용하지 않는 페이지의 크기를 줄여 스왑 공간의 크기를 줄이는 역할을 한다. 또한, 스왑 공간에 의한 메모리 낭비를 막기 위해 압축을 사용하여 스왑 공간의 크기를 최소화한다.

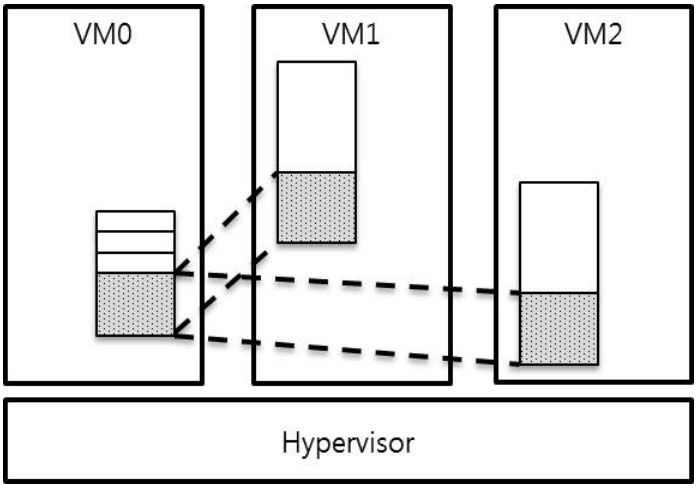

그림 1 가상 머신을 위한 인-메모리 스왑 공간

3. 구현

본 논문은 가상화된 데스크 탑 환경을 가정하고, 스왑 공간을 구현하였다. 가상화 환경을 구축하기 위해 Xen-4.0.1[5][6]을 하이퍼바이저(hypervisor)로 사용하고, 가상 머신은 Linux-2.6.32.27과 2.6.37.1을 사용한다.

인-메모리 페이지 스왑은 4가지 알고리즘으로 나누어 구현하였다. 인-메모리 스왑 공간 알고리즘은 메모리 내에 리스트 형태로 갖고 있는 스왑 공간을 유지한다. 이 리스트로부터 페이지 단위로 공간을 할당받아 페이지를 저장한다. 스왑 공간 이전 알고리즘은 인-메모리 스왑 공간을 관리 가상 머신에게 이전하기 위한 것이다. 이를 위해, 각 가상 머신은 사전에 관리 가상 머신으로부터 스왑 공간을 생성하기 위한 메모리 영역을 받아온다. 이를 위해 xen 하이퍼바이저가 지원하는 그런트(Grant) 페이징 기법[7]을 적용한다. 관리 가상 머신으로부터 받은 메모리 영역은 인-메모리 스왑 공간 알고리즘에게 전달되어 스왑 공간을 생성하고 유지하게 된다. 페이지 선택 알고리즘은 여유 페이지(free page)를 생성하기 위해 메모리에서 스왑 공간으로 이동시킬 페이지를 선택한다. 내보낼 페이지는

LRU 페이지 교체 알고리즘에 의해 선택된다. 선택된 페이지는 페이지 압축 알고리즘에게 전달되어 페이지 압축을 실행한 후, 스왑 공간으로 이동된다. 페이지 압축 알고리즘은 페이지의 크기를 줄이기 위한 알고리즘이다. 이를 통해 스왑 공간으로 보낼 페이지를 압축(compress)하고, 메인 메모리로 복구할 페이지를 복원(decompress)한다. 압축 알고리즘은 LZO 알고리즘[8]을 사용한다. 압축과 복원에 걸리는 시간은 디스크 쓰기에 비해 약 70% 더 빠르기 때문에 압축과 복원에 걸리는 시간은 오버헤드가 되지 않는다[4].

4. 실험 및 평가

이 장에서는 구현한 인-메모리 스왑을 실험을 통해 메모리 여유 공간 확장의 효율을 검증한다. 실험 환경은 2.4GHz 듀얼 코어 CPU와 4GB의 메모리를 가진 시스템에서 두 개의 가상 머신을 동작시켰다. 두 가상 머신 모두 1GB씩 메모리를 할당하였고, 메모리 부족을 야기하기 위해 한쪽 가상 머신에서 백그라운드 프로세스(background process)로 메모리 점유를 발생시키는 간단한 어플리케이션을 실행시켰다. 그 후, 커널 컴파일을 실행하여 커널 컴파일 도중에 페이지 스왑이 발생하도록 하였다.

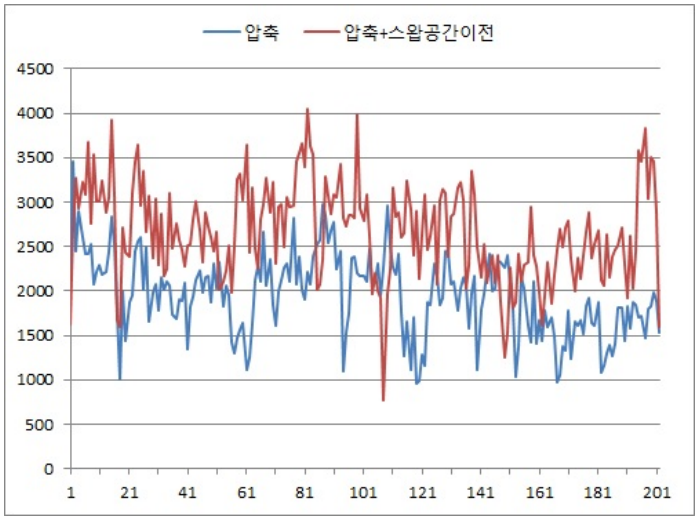

그림 2 시간 당 평균 여유 페이지 수의 변화

그림 2는 시간에 따른 여유 페이지 수의 변화를 나타낸다. "압축"은 가상 머신이 자신의 메모리 안에 스왑 공간을 두는 경우이고, "압축+스왑공간이전"은 관리 가상 머신으로 스왑 공간을 이동시킨 경우를 의미한다. 그래프에서 보이는 바와 같이, 스왑 공간을 다른 가상 머신으로 이전한 경우에 더 많은 여유 페이지를 갖는다. 여유 페이지가 많다는 것은 더 적은 페이지를 사용한다는 의미이다. 따라서 스왑 공간을 이전하는 경우가 메모리 사용량이 더 적다. 이는 스왑 공간이 차지하는 부분이 다른 가상 머신으로 이동했기 때문에 나타난다. 즉, 관리 가상 머신과 같이 충분한 여유 메모리를 가진 가상 머신이 함께 동작하는 환경에서는 더 많은 여유 메모리를 얻을 수 있고, 페이지 스왑의 성능을 향상시킬 수 있다.

5. 결론 및 향후 연구 방향

본 논문은 가상화 환경에서의 인-메모리 페이지 스왑 기법을 제안하고 구현하였다. 이 기법은 메모리를 스왑 공간으로 활용하기 때문에 디스크를 이용하는 기존의 페이지 스왑에 비해 빠르다. 또한, 다른 가상 머신으로의 스왑 공간 이전과 페이지 압축의 과정을 통해 다른 인-메모리 페이지 스왑에 비해 더 많은 여유 공간을 확보할 수 있다.

향후에는 다양한 환경과 어플리케이션에 대해 성능을 보이고, 다른 메모리 문제 해결책들과의 비교를 통해 효과를 검증할 계획이다.

참고문헌

[1] Irina Chihaia Tuduce and Thomas Gross, "Adaptive Main Memory Compression," USNIX Annual Technical Conference, 2005.

[2] L. Yang, R. P. Dick, H. Lekatsas, and S. Chakradhar, "Online Memory Compression for Embedded Systems," ACM Transactions o Embedded Computing Systems, Vol. 9, No. 3, Article 27, 2010.

[3] Google's open project, CompCache, http://code.google.com/p/compcache/

[4] 이치영, 이정환, 유혁, "가상화 환경에서 효율적인 메모리 사용을 위한 페이지 압축과 저장소 공유," 한국정보과학회 추계학술발표대회, 2011.

[5] Xen Hypervisor, http://www.xen.org/

[6] 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(SOSP), 2003.

[7] A Rough Introduction to Using Grant Tables, http://xenbits.xen.org/docs/unstable/misc/grant-tables.txt

[8] LZO Compression Algorithm, http://www.oberhumer.com/opensource/lzo/