Xen에서 메모리 이용률 향상을 위한 동적 할당 기법

(A Dynamic Allocation Scheme for Improving Memory Utilization in Xen)

이 권 용 † 박 성 용 ††
(Kwonyong Lee) (Sungyong Park)

요 약 최근 서버의 통합을 통해 시스템 자원의 효율적인 활용을 제공할 수 있는 시스템 가상화가 많은 주목을 받고 있다. 이 시스템 가상화 기술을 통하여 보다 효과적으로 시스템 자원을 활용하고 가상화 소프트웨어의 성능을 향상시킬 수 있는 방안이 다양하게 연구되고 있다. 이러한 연구들은 CPU 측면에서 동적으로 가상머신에 할당된 양을 조절하거나 마이그레이션 기능을 활용하여 머신 간 자원 관리 등의 다양한 측면에서 활발하게 진행되고 있으나 메모리 측면에서는 그 연구가 매우 부족한 실정이다. 따라서 서버 통합에서의 메모리 자원의 이용은 가상머신 탑재 시에 정적으로 할당된 메모리를 사용하는 수준에서 머물고 있다. 하지만 본 논문의 성능 비교 환경인 Xen 가상화에서 가상머신에 정적으로 메모리를 할당하는 방식은 유휴메모리를 다량 발생시켜 메모리 이용률을 낮추게 된다. 메모리 이용률을 높이기 위하여 가상머신에 할당하는 메모리양을 줄일 경우 다른 시스템 자원에도 영향을 미치게 되며 가상머신에서 운영되는 서비스의 성능 저하를 유발하게 된다. 본 논문에서는 가상머신 사이의 메모리 할당량을 조절하여 가상머신의 서비스에 성능저하가 없으면서 이용률을 향상시킬 수 있는 메모리의 동적 할당을 제안한다. 메모리 사용량 예측을 위한 AR 모델과 메모리 이용률 최적화를 위한 개미 군집 알고리즘을 사용하여 구현한 메모리의 동적 할당 시스템을 통하여 정적 할당의 경우에 비하여 더 많은 수의 가상머신을 운영할 수 있게 되고 서버로 운영되는 가상머신의 서비스 성능 저하 없이 약 1.4배의 이용률 향상을 얻을 수 있었다.

키워드 : 가상화, Xen, 메모리, 할당, 이용률

Abstract The system virtualization shows interest in the consolidation of servers for the efficient utilization of system resources. There are many various researches to utilize a server machine more efficiently through the system virtualization technique, and improve performance of the virtualization software. These researches have studied with the activity to control the resource allocation of virtual machines dynamically focused on CPU, or to manage resources in the cross-machine using the migration. However, the researches of the memory management have been wholly lacking. In this respect, the use of memory is limited to allocate the memory statically to virtual machine in server consolidation. Unfortunately, the static allocation of the memory causes a great quantity of the idle memory and decreases the memory utilization. The underutilization of the memory makes other side effects such as the load of other system resources or the performance degradation of services in virtual machines. In this paper, we suggest the dynamic allocation of the memory in Xen to control the memory allocation of virtual machines for the utilization without the performance degradation. Using

· 본 연구는 지식경제부 및 한국산업기술평가관리원의 IT산업원천기술개발사업의 일환으로 수행하였음[2010-KI002090, 신뢰성 컴퓨팅(Trustworthy Computing) 기반 기술 개발]

† 학생회원 : 서강대학교 컴퓨터공학과
dlrnjsdyd@sogang.ac.kr
†† 종신회원 : 서강대학교 컴퓨터공학과 교수
parksy@sogang.ac.kr

논문접수 : 2009년 5월 2일
심사완료 : 2010년 3월 26일

Copyright©2010 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.

정보과학회논문지: 시스템 및 이론 제37권 제3호(2010.6)

AR model for the prediction of the memory usage and ACO (Ant Colony Optimization) algorithm for optimizing the memory utilization, the system operates more virtual machines without the performance degradation of servers. Accordingly, we have obtained 1.4 times better utilization than the static allocation.

Key words : Virtualization, Xen, Memory, Allocation, Utilization

1. 서 론

최근 UCC 등 미디어 서버의 증가와 관련 인프라의 투자가 촉진됨에 따라 고용량 파일을 저장하고 신속하게 처리할 수 있는 각종 서버의 수요가 더욱 증가하고 있다. 이러한 서버의 수요 증가는 앞으로도 수많은 분야에서 필요로 하는 서버의 역할이 커짐에 따라 계속될 것으로 전망되고 있다. 하지만 서버의 증가에 따라 사용되는 에너지 증가, 서버 유지 및 관리 비용, 필요 공간 증가, 그리고 다양한 운영체제에 따른 소프트웨어 의존성 등이 문제가 되고 있다. 이에 따라 효율적인 자원 활용을 통해 최소 비용으로 서버를 효과적으로 사용할 수 있는 서버 통합 기술에 관심이 집중되고 있다. 실제로 시스템의 하드웨어 성능이 향상되면서 시스템 자원의 이용률은 평균 15~20% 정도에 불과하기 때문에 서버 통합 기술을 통하여 낮은 자원 이용률을 향상시켜 시스템 비용 절감 효과를 얻을 수 있다. 시스템 가상화 기술은 서버 통합에 널리 활용되고 있는 기술이며 하드웨어 성능 향상에 따라 더 많은 서버의 통합을 가능하게 해주고 있다.

현재 시스템 가상화 기술을 통하여 보다 효과적으로 시스템 자원을 활용하고 가상화 소프트웨어의 성능을 향상시킬 수 있는 방안이 많이 연구되고 있다. 이러한 연구들은 CPU 측면에서 동적으로 가상머신에 할당된 양을 조절하거나 마이그레이션 기능을 활용하여 머신 간 자원 관리 등의 다양한 측면에서 활발하게 진행되고 있으나 메모리 측면에서는 그 연구가 매우 부족한 실정이다. 따라서 서버 통합에서의 메모리 자원의 이용은 가상머신 탑재 시에 정적으로 할당된 메모리를 사용하는 수준에서 머물고 있다. 본 논문의 대상인 Xen은 한 가상머신에 할당된 메모리 자원을 다른 가상머신이 접근하지 못하도록 완전히 분리된 가상화 환경을 제공한다[1,2]. 완전히 분리된 환경을 제공하기 때문에 정적으로 메모리를 할당하는 방식은 다량의 유휴메모리가 발생시키고 이는 시스템 전체의 메모리 이용률을 낮추게 된다. 메모리 이용률을 높이기 위하여 가상머신에 할당되는 메모리양을 줄일 경우 다른 시스템 자원인 CPU 등의 동작에 영향을 미치게 되며 가상머신에서 동작하는 서비스의 성능 저하도 발생시키게 된다.

본 논문에서는 메모리 역시 시스템의 이용률에 중요한 요소라는 것에 초점을 맞추고 메모리의 동적 할당을 제안하고자 한다. 동적 할당에는 다음 메모리 사용량 예측을 위한 AR 모델[3]과 메모리 이용률을 최적화하기 위한 개미 군집 알고리즘[4,5]이 사용되었다. 제안하는 메모리의 동적 할당의 필요성과 타당성을 검증하기 위하여 응용프로그램의 서비스 지연에 영향을 미치는 메모리 요구량 대비 실제 제공된 메모리양과 시스템 전체의 메모리 이용률 등을 기존의 정적메모리 할당 방식 및 간단한 동적 할당 방식과 비교하였다. 실험 결과 본 논문에서 제안하는 동적 할당 방식이 정적 할당의 경우에 비하여 더 많은 수의 가상머신을 성능 저하 없이 운영할 수 있게 되어 약 1.4배의 이용률 향상을 얻을 수 있었고 간단한 동적 할당과 비교해서도 더 높은 이용률을 얻을 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련 있는 연구를 살펴보고, 3장에서는 본 연구에서 제안하는 메모리의 동적 할당과 이를 적용한 시스템에서 필요로 하는 시스템의 각 부분에 대하여 설명한다. 4장에서는 메모리의 동적 할당 시스템의 성능 측정 및 다른 시스템과의 비교 분석을 수행하여 성능에 대한 평가를 한다. 마지막으로 5장에서는 본 논문의 결론과 동적 할당 시스템이 갖는 오버헤드 및 한계를 논의한다.

2. 관련 연구

시스템 가상화 분야의 메모리 자원 이용에 대한 연구는 CPU 자원에 비해 동적으로 시스템 자원을 관리할 수 있는 연구가 많이 부족한 실정이다. 가상화 시스템 소프트웨어 중 VMware측에서는 "Memory Resource Management in VMware ESX Server"[6]와 같은 연구가 있으나 오픈소스인 Xen에 대한 메모리 관리 연구는 찾아보기 힘들다. [6]에서는 벌루닝(Ballooning)을 통한 메모리 관리와 공유 기반의 메모리 관리, 그리고 빈번한 I/O에 대한 페이지 리맵핑(Hot I/O page remapping) 기술을 언급하고 있다. 그 중 벌루닝을 통한 메모리 관리는 벌루닝 기술 자체를 기술하고 있어 동적으로 가상머신에 메모리를 할당해주는 범위까지는 다뤄지지 않고 있다. 공유 기반의 메모리 관리는 컨텐츠 기반의 페이지 공유를 의미하므로 범용적으로 사용할 수는 없는 기술이다. 마지막으로 빈번한 I/O에 대한 페이지 리맵핑은 대용량 메모리 시스템에서 빈번하게 발생하여 생기는 I/O 카피 오버헤드를 줄이고자 하는 기술이다.

VMware에 반해 Xen 기반의 시스템 자원 관리 연구들 대부분은 메모리에 대해서는 언급을 최소화하고 특정 머신에 가상머신을 탑재하기에 메모리가 부족한 경우 메모리가 충분하게 남아있는 다른 머신으로 가상머신을 마이그레이션하는 방법을 사용하거나 메모리가 부족한 머신에서 운영되고 있는 가상머신과 메모리가 충분히 남아있는 머신에서 운영되는 가상머신을 스왑 하는 방식[7]으로 메모리 관리 방법을 제안하는 정도에 그치고 있다. 이러한 방식은 다수의 가상화 노드가 연동하여 동작하는 환경에서 가능하기 때문에 한 가상화 노드에서의 가상머신들에 할당된 메모리를 관리하는 방법을 제시하고 있는 본 논문과는 차이가 있다. 최근 Xen의 메모리 관리를 연구한 [8]에서는 가상머신의 메모리 이용률에 기반을 두고 메모리를 동적으로 할당하는 방안에 대하여 제안하고 있다. 이 논문에서는 이용률을 임계값으로 하여 가상머신이 SLO(Service Level Objective)를 지킬 수 있도록 가상머신의 메모리 할당량을 조절한다. 사용하고 있는 메모리와 현재 할당된 메모리의 비율이 설정한 메모리 이용률 임계값에 다다르는 경우 가상머신에 할당된 메모리를 증가시켜주고 반대의 경우 메모리 할당을 줄인다. 하지만 메모리 이용률 임계값을 기준으로 동작하기에 반드시 추가 할당된 메모리양이 반드시 존재하게 되고 이는 메모리의 이용률을 감소시키는 요인이 된다. 그리고 현재의 메모리 할당량과 이용량에 기반을 두어 메모리를 조절하기 때문에 메모리가 적게 할당된 상황에서 SLO를 지켜주기 위해 추가 할당되는 메모리양과 메모리가 많이 할당된 상황에서의 추가 할당되는 메모리양이 다르기 때문에 메모리가 적게 할당되어 있는 상황에서 큰 폭의 메모리 요구량 변화에 대한 추가 할당에서는 매우 취약한 모습을 나타낼 수밖에 없다. 또한 할당하고자 하는 메모리양이 전체 메모리양을 넘어서는 경우에 대한 고려가 되어있지 않다.

따라서 본 논문에서는 불규칙적이고 변화의 폭이 큰 메모리 사용에 대해서도 메모리의 낭비를 최소화하여 이용률을 극대화하고 전체 메모리가 부족한 경우에도 최적의 할당을 통해 가상머신 상의 서비스 성능에 적은 영향만을 미치도록 하는 메모리의 동적 할당을 제안하고자 한다.

3. 메모리의 동적 할당

본 장에서는 메모리 자원 이용률을 극대화하고 특정 가상머신의 희생 없이 최적의 할당을 수행할 수 있는 메모리의 동적 할당을 제안한다. 우선 제안하는 기법의 목적을 살펴보고 그 목적을 달성하기 위한 메모리의 동적 할당 시스템 전체 구조와 각 구성에 대하여 자세히 설명한다.

3.1 동적 할당의 목적

본 논문에서 제안하고자하는 기법은 CPU 자원에 비해 비교적 중요도가 떨어진다고 판단되어지는 메모리 자원 역시 중요한 자원임을 보이고 기존의 Xen이 제공하는 가상머신의 낮은 메모리 자원 이용률을 개선하기 위한 정책이다. 기존의 Xen이 제공하는 가상머신의 메모리 자원 이용률은 발생하는 유휴메모리로 인하여 낮은 수치일 수밖에 없고 비용이나 메모리 슬롯 등의 문제로 인해 메모리를 무한정 늘릴 수 없는 상황에서 CPU와 같은 다른 자원이 충분히 제공될 수 있다고 하더라도 가상머신을 제한된 수만 운영할 수 있어서 가상화 시스템 환경에서 하나의 걸림돌이 되고 있다. 또한 가상머신이 서비스를 제공하는데 있어서 메모리 자원은 FTP 파일 서버와 같은 몇몇 서비스를 제외하고는 비교적 중요도가 떨어지는 자원으로 다루어지지만 QoS를 요구하는 시스템에서는 서비스 지연과 같은 문제가 발생하기 때문에 메모리가 중요하게 여겨지는 경우가 존재한다. 그리고 가상머신에 메모리 자원이 충분히 제공되지 않는 경우 발생하는 스왑으로 인해 발생하는 오버헤드를 고려한다면 메모리도 가상머신에 제공되는 시스템 자원에서 중요한 부분을 차지한다는 것을 알 수 있다. 그림 1은 가상머신 환경에서 운영체제가 메모리 부족을 해소하기 위해 스왑을 수행하는 경우 소모되는 CPU 사용량과 스왑 수행에 소모되는 시간을 측정하여 표시한 그래프이다. 스왑에 의하여 메모리가 확보되는 동안 가상머신에서 운영되던 서비스는 충분한 CPU 자원을 사용할 수 없어져서 정상적인 서비스를 제공할 수 없게 된다. 그림 1의 실험 결과에 따르면 가상머신에서 운영되던 특정 서비스가 CPU 50%의 이용률을 지속적으로

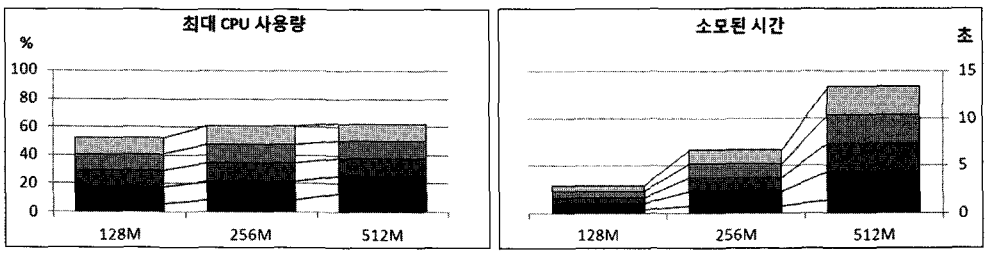

그림 1 메모리 스왑에 사용된 최대 CPU 사용량과 스왑 수행 시간

필요한 경우 256MB의 메모리가 부족해 스왑이 수행되는 경우 서비스는 운영체제의 스왑 작업과 CPU를 나눠 쓰게 되므로 온전한 서비스를 제공할 수 없게 된다.

하이퍼바이저 기반의 가상화 기법인 Xen은 가상머신에 할당될 메모리 자원을 먼저 확보한 후 가상머신을 탑재한다. 그리고 가상머신에 할당된 메모리 자원은 사용되지 않더라도 다른 가상머신에서 사용할 수 없고 필연적으로 유휴메모리를 발생시키게 된다. 그림 2는 Xen 가상화 환경에서 두 가상머신에 요청된 메모리 요구량과 실제 각 가상머신에서 사용된 메모리양을 나타낸 그래프이다. 각 가상머신의 비사용 메모리, 즉 유휴메모리는 더 많은 메모리를 필요로 하는 다른 가상머신에서 사용할 수 없기 때문에 그림 3과 같이 낮은 이용률을 갖게 된다. 따라서 한 가상머신에서 발생한 유휴메모리를 다른 가상머신이 사용할 수 있도록 할당해줌으로써 메모리 자원의 이용률 향상을 얻을 수 있다.

따라서 메모리의 동적 할당에서는 각 Xen 가상머신에서 발생할 유휴메모리를 예측하여 미리 확보하고 메모리를 추가로 필요하게 될 다른 가상머신에 할당해주어 이를 사용할 수 있도록 한다. 이때 메모리 필요량을 모두 확보하지 못하는 경우가 발생할 수 있으므로 이 경우에는 최적화를 통하여 메모리 이용률이 최대가 될 수 있는 할당량 조합을 찾아서 각 가상머신에 할당하여 특정 가상머신의 희생을 방지하면서 가상머신들이 정상적으로 서비스를 할 수 있도록 한다. 결과적으로 향상된 메모리 이용률을 얻을 수 있고 동적 할당을 통해 추가적으로 확보된 메모리를 이용하여 더 많은 수의 가상머신이 서비스를 정상적으로 운영할 수 있도록 한다.

3.2 동적 할당의 흐름

그림 4는 본 논문에서 제안하는 동적 할당을 순서도로 나타낸 것이다. 메모리의 동적 할당에서는 우선 탑재된 가상머신들이 앞으로 사용하게 될 메모리양을 구하기 위해 각 가상머신의 메모리 사용량을 모니터링하고 그 값을 바탕으로 다음 메모리 사용량을 예측해야 한다. 예측된 값에 따라 더 적은 메모리를 사용하게 될 가상

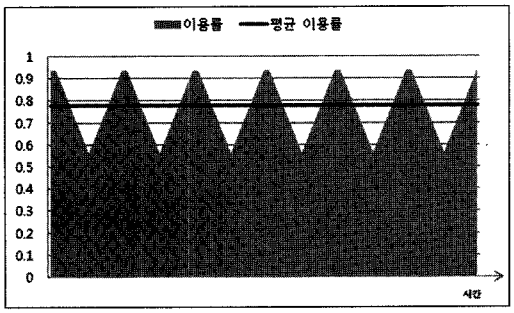

그림 3 두 Xen 가상머신의 메모리 이용률

머신들로부터 메모리 자원을 회수한 다음 더 많은 메모리를 사용하게 될 가상머신에게 할당해주는 과정을 수행한다. 이때 이전 시점에서 모두 할당받지 못한 가상머신이 존재할 경우 일련의 계산을 통해 부족했던 부분을 할당해주는 지연된 할당을 수행한다. 이러한 지연된 할당을 통해 가상머신의 서비스 성능 저하를 최소한으로 줄일 수 있게 된다. 미리 계산해둔 다음 메모리 사용량 예측을 통해 추가 할당해주어야 할 메모리양 계산되면 그 총합이 확보된 메모리양보다 많은 경우 최적화 알고리즘을 통해 특정 가상머신의 희생이 없으면서 시스템 전체의 메모리 이용률을 최대가 되도록 하는 메모리 할당량 조합을 찾아 메모리 할당 배분을 조정해주어야 한다. 마지막으로 예측된 값 또는 최적화된 할당량 조합에 따라 각 가상머신에 메모리 할당을 수행한다. 이러한 일

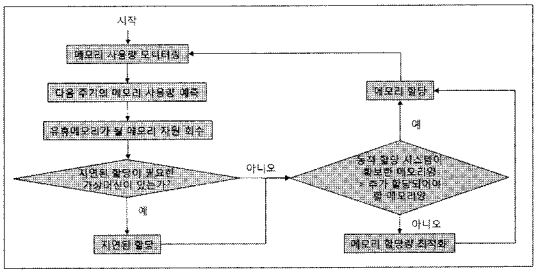

그림 4 동적 할당의 순서도

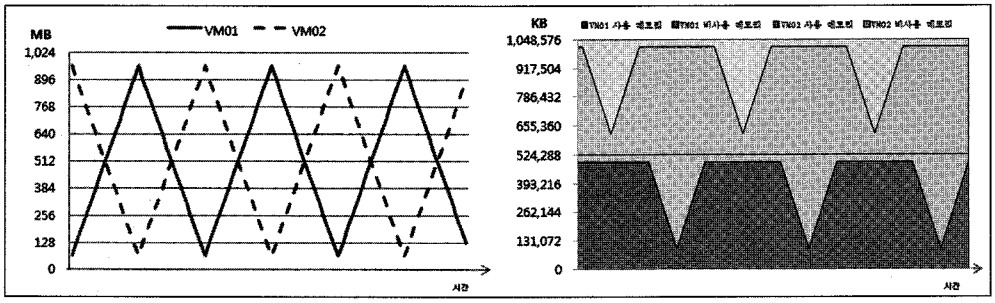

그림 2 두 Xen 가상머신의 메모리 요구량과 실제 사용량

련의 과정을 통해 동적 할당 시스템은 동작을 하게 되며 이 과정을 주기적으로 반복하게 된다. 한 주기를 마치고 다음 주기를 수행하는데 필요한 시간은 1초가 되도록 구성하였으며 이를 위해 한 주기를 마치고 남은 시간은 대기하게 된다.

3.3 메모리 사용량 예측

메모리의 동적 할당은 관찰된 메모리 사용량을 바탕으로 앞으로 사용하게 될 메모리양을 예측하고 가상머신에 추가로 할당해주거나 할당량을 줄여주어야 한다. 메모리 사용량 모니터링 데이터는 일정 시간 간격으로 배치된 데이터들의 수열을 의미하는 시계열 데이터라고 할 수 있다. 그리고 시계열 예측은 주어진 시계열을 보고 수학적인 모델을 만들어서 미래에 일어날 것들을 예측하는 것을 뜻한다. 시계열 데이터를 분석하고 예측하는 수학적 모델은 여러 가지가 있을 수 있으나 실제 응용에서 가장 많이 쓰이는 세 가지 범용 모델은 선형 종속적인 AutoRegressive(AR) 모델[3], Integrated(I) 모델, Moving Average(MA) 모델 등이 있다[9,10]. 본 논문에서는 이러한 시계열 데이터 예측 모델 중 간단하면서도 입증된 정확도를 갖고 있어서 컴퓨터 시스템 자원 예측에 널리 사용되는 AR 모델을 빠르고 비교적 정확한 다음 주기의 메모리 사용량 예측을 위해 사용하였다.

AR 모델에는 AR 차수나 예측에 사용될 데이터 개수와 같이 지정해 주어야하는 파라미터가 존재한다. 본 논문에서는 이러한 파라미터의 값을 지정하기 위해서 실제 메모리 사용량 데이터를 바탕으로 가장 정확한 예측을 수행하는 파라미터를 측정하여 사용하였다.

AR 예측은 차수가 높아질수록 더 정확한 예측 결과를 얻을 수 있지만 차수가 커질수록 예측에 필요한 시간과 데이터양이 증가하는 특성을 갖고 있다. 본 논문의 기법을 활용하는 시스템에서는 실시간으로 메모리의 동적 할당을 수행해야하므로 예측에 소모되는 시간을 최소화 시켜야 한다. 따라서 AR 차수를 결정하기 위하여 충분한 개수의 데이터를 사용하여 차수에 따른 정확도를 측정하였다. 그림 5는 차수에 따른 오차를 제곱 평균

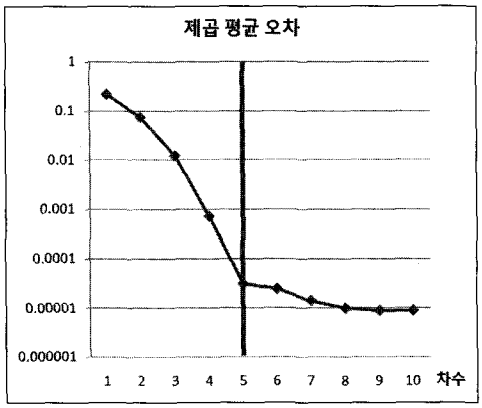

그림 5 AR 차수에 따른 제곱 평균 오차

으로 표시한 그래프이다. 이 그래프를 통해 AR 차수 5 이후로 오차의 감소폭이 줄어들고 비교적 적은 오차를 갖게 된다는 것을 알 수 있었으므로 본 논문의 AR 예측에 차수 5를 사용하였다.

다음으로 결정된 AR 차수에 대하여 예측에 필요한 메모리 사용량 데이터양을 결정하여야 한다. AR 예측에 필요한 최소한의 데이터 개수는 차수보다 하나 큰 수이다. 하지만 최소한의 데이터로는 예측의 정확도가 떨어지고 예측 자체가 실패하는 경우도 발생하게 된다. 따라서 예측에 사용될 메모리 사용량 데이터 개수를 결정하기 위해 실제 메모리 사용량 데이터를 바탕으로 실험하였다. 그림 6의 그래프는 AR 예측 평균 오차와 AR 예측 실패 횟수 그래프이다. 메모리의 동적 할당에서 수행되는 다음 메모리 사용량 예측은 예측이 실패하는 경우가 없으면서 최소한의 오차를 가져야하므로 예측에 사용할 메모리 사용량 데이터는 12개로 결정하였다.

본 논문에서 제안하는 동적 할당 시스템은 위의 두 실험에서 결정한 AR 모델의 파라미터를 사용하여 최소한의 시간 내에 비교적 정확한 결과를 도출하여 다음 주기의 메모리 사용량을 예측할 수 있게 된다. 더 정확한 예측을 위해서는 더 높은 차수와 더 많은 데이터를

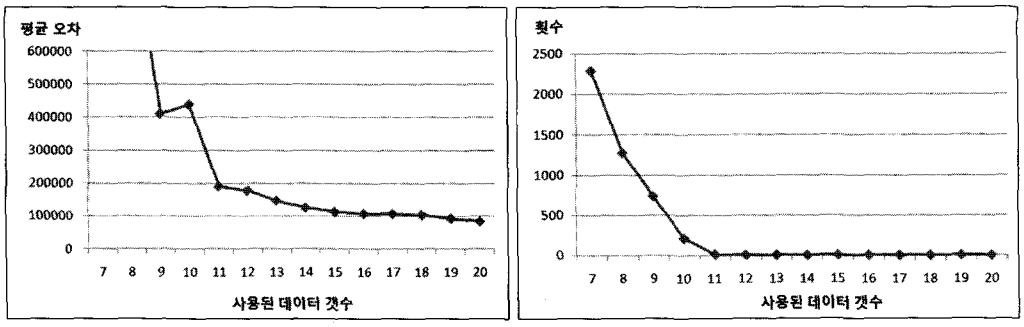

그림 6 AR 예측에 필요한 데이터양 실험

사용하여 계산하면 되지만 본 논문의 시스템은 메모리의 동적 할당을 실시간으로 수행하기 위하여 최소한의 시간에 비교적 정확한 예측 결과를 도출할 수 있는 이 파라미터 값들을 사용하였다.

3.4 메모리 할당량 최적화 및 할당

본 논문에서 제안하는 기법의 시스템은 Xen에서 제공하는 메모리 벌룬 드라이버를 사용하여 탑재된 가상머신의 메모리 할당량을 조절한다[1,2]. 메모리 벌룬 드라이버를 통하여 가상머신에 할당된 메모리를 증가시키기 위해서는 어떤 가상머신에도 할당되지 않은 메모리가 존재하여야 한다. 따라서 다른 가상머신에 현재 할당된 메모리를 감소시키는 메모리 벌룬 드라이버 동작을 통해 메모리를 확보한 후 추가 메모리가 필요한 가상머신의 메모리 할당량을 확장시켜주어야 한다. 하지만 추가 요청된 메모리양의 총합이 확보된 메모리양보다 큰 경우 모든 추가 메모리 할당 요구를 들어줄 수가 없게 된다. 단순히 메모리 이용률만을 고려하는 경우 탐욕적인(Greedy) 알고리즘을 통해 문제 해결이 가능하지만 본 논문의 목표는 특정 가상머신의 희생없이 각 가상머신에서 운영되는 서버의 성능을 최대한 유지하면서 시스템 전체의 메모리 이용률을 최대화시키는 것이므로 이러한 조건들을 만족시킬 수 있는 각각의 가상머신에 대한 할당량을 최적화 알고리즘을 통해 결정한다.

추가 할당해주어야 하는 메모리가 부족한 경우의 메모리 할당 문제는 잘 알려진 분할가능 배낭 문제(Fractional knapsack problem)와 유사한 형태를 갖는다. NP-Hard 문제로 알려진 배낭 문제는 GA(Genetic Algorithm)나 ACO(Ant Colony Algorithm), SA(Simulated Annealing) 등의 최적화 알고리즘을 통해 해결될 수 있다. 본 논문에서는 이러한 최적화 알고리즘 중 개미 군집 알고리즘(ACO)을 메모리 할당량 최적화 방법으로 선택하였다[4,5]. 표 1은 배낭 문제에 대응되는 메모리 할당 문제의 파라미터들을 나타낸 것이다.

개미 군집 최적화 알고리즘을 사용하기 위해서는 우선 적합도 함수가 정의되어야 한다. 본 논문은 메모리 자원의 이용률을 높이면서 특정 가상머신의 희생이 없도록 하는 것을 목표로 하고 있으므로 각각의 가상머신이 요구하는 메모리양과 실제 확보한 메모리양의 비율, 메모리 이용률에 의하여 적합도를 계산하였다. 이때 특정 가상머신의 희생이 없도록 하기 위한 가상머신의 메모리양에 대한 비율과 메모리 이용률에 대한 비중은 동일하도록 구성하였다. 전체 메모리양을 M, 탑재된 가상머신의 개수를 n, 가상머신 i에 할당될 메모리양을 A_i, 가상머신 i에서 사용할 예측된 메모리양을 P_i라고 할 때, 가상머신 i의 메모리 사용량 X_i는 식 (1)과 같다.

$$X_i = \begin{cases} P_i & , P_i \le A_i \\ A_i & , otherwise \end{cases} \tag{1}$$

계산된 메모리 사용량 X_i를 이용하여 메모리 이용률 식 (2)와 가상머신의 메모리양에 대한 비율 식 (3)을 정의할 수 있다.

$$\text{메모리 이용률} = \left(\sum_{i=1}^{n} X_i\right)/M \tag{2}$$

$$\text{가상머신의 메모리양에 대한 비율} = \left(\sum_{i=1}^{n}\left(|(A_i/P_i)|-1\right)\right)/n \tag{3}$$

위의 식 (2)와 식 (3)을 동일한 비중으로 적합도를 계산하기 위해서 각각 0.5로 정규화(Normalization)를 시켜준다. 아래의 식 (4)로 적합도를 구할 수 있고 구해진 적합도는 페로몬 값을 반영하는 척도로 사용된다.

$$\text{적합도} = \begin{cases} \left(\left(\sum_{i=1}^{n} X_i\right)/M\right)\times 0.5 + \left(\left(\sum_{i=0}^{n}\left(|A_i/P_i|-1\right)\right)/n\right)\times 0.5 & , \left(\sum_{i=1}^{n} A_i\right) \le M \\ 0 & , otherwise \end{cases} \tag{4}$$

다음으로 개미 군집 알고리즘의 한 구간에서 사용할 개미의 수를 결정하여야 한다. AR 모델의 파라미터와 마찬가지로 더 많은 수의 개미를 사용할수록 더 최적화된 결과를 얻을 수 있지만 개미의 수가 늘어날수록 계산에 필요한 시간은 증가하게 된다. 따라서 본 논문의 동적 할당 시스템에서 사용할 개미의 수를 결정하기 위해 최적 값을 발견하기까지의 구간 반복 횟수를 측정하였다. 그림 7의 그래프는 실제 메모리 사용량 데이터를 바탕으로 식 (4)의 적합도가 가장 높게 나타날 때까지의 구간 반복 횟수를 1000번 측정하여 평균을 표시한 그래프이다. 실험 결과에 따르면 적합한 개미의 수는 그래프의 기울기 감소폭이 매우 적어지는 70마리 이상이라는 것을 알 수 있었고 따라서 동적 할당 시스템에서의 최적화에서는 적절한 결과를 얻을 수 있는 최소한의 개미 마릿수인 70마리를 사용하였다.

표 1 배낭 문제와 메모리 할당 문제의 파라미터 대조

배낭 문제	메모리 할당 문제
배낭	메모리
담을 수 있는 총 무게	할당되지 않은 메모리 총량
물건	메모리 추가 할당이 필요한 가상머신
물건의 값어치	추가 할당된 메모리로 인한 메모리 이용률 향상
물건의 무게	가상머신에 할당될 메모리

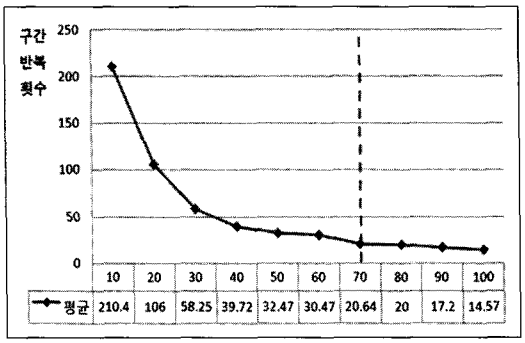

그림 7 개미 수에 따른 최적 값 발견에 필요한 반복 횟수

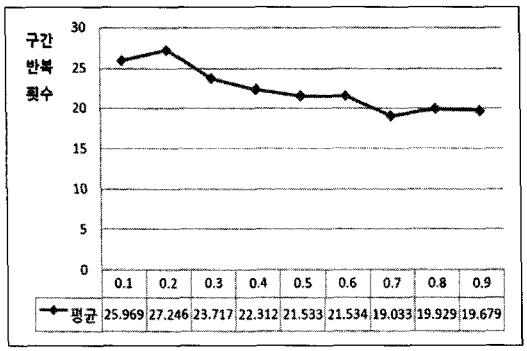

그림 8 페로몬 증발율에 따른 최적 값 발견에 필요한 반복 횟수

마지막으로 개미 군집 알고리즘의 중요한 요소인 페로몬 증발율을 결정하기 위해 마찬가지로 최적 값을 발견하기까지의 필요한 구간 반복 횟수를 페로몬 증발율 변화에 따라 실험하였다. 이 페로몬 증발율은 개미 군집 알고리즘의 최적화 수준을 좌우하는 중요한 파라미터이다. 이 파라미터는 최적화에 사용되는 데이터에 따라 결과가 달라지기 때문에 실제 메모리 사용량에 기반을 두어 1000번의 실험을 수행하였다. 앞서 개미 마릿수를 결정한 실험과 마찬가지로 적합도가 가장 높게 나타날 때까지의 구간 반복 횟수를 1000번 측정하여 평균을 내어서 그림 8에 표시하였다. 이 그래프를 통해 페로몬 증발율이 0.7일 때 가장 좋은 결과를 갖는다는 것을 알 수 있었고 본 논문의 동적 할당 시스템에서는 개미 군집 알고리즘의 페로몬 증발율을 0.7로 사용하였다.

3.5 지연된 할당

이 절에서 설명할 지연된 할당은 3.4절의 메모리 할당량 최적화에서 사용하게 될 모든 메모리양을 할당받지 못한 가상머신에게 이후 메모리가 확보되면 할당해주는 것이다. 이는 3.1절에서 설명한 가상머신에 할당된 메모리가 부족한 경우 스왑으로 인하여 발생하는 CPU 소모가 가상머신의 서비스에 미치게 될 영향에 대해 최대한 빠르게 대처해주기 위한 것이다. 본 논문의 동적 할당 시스템을 대상으로 설명하자면 이전 주기에서 요구한 메모리양을 할당받지 못한 가상머신이 스왑 동작을 수행하게 됨으로써 발생될 서비스의 성능저하를 최소화시키기 위하여 유휴메모리가 발생하면 아직 스왑이 수행되지 않은 메모리양만큼을 할당해주는 것을 지연된 할당이라고 한다.

지난주기에 부족했던 메모리양에 대해서는 이미 스왑이 발생한 상태가 되지만 아직 스왑이 진행 중인 메모리양에 대해서는 가상머신의 메모리 할당을 늘려줌으로써 스왑 동작을 중지시킬 수 있다. 따라서 본 논문에서는 그림 1 그래프의 스왑 수행 시간 수치를 활용하여 이전 주기에 부족했던 가상머신이 현 시점에서 아직 확보하지 못한 메모리양을 계산할 수 있다. 3.2절에서 설명한 것처럼 동적 할당 시스템의 주기를 1초로 구성하였으므로 1초 동안 가상머신이 자체적으로 스왑을 통해 확보할 수 있는 메모리양이 계산되어야 한다. 그림 1에 따르면 스왑을 수행해서 128M의 메모리를 확보하는데 필요한 시간은 약 2.5초이다. 따라서 동적 할당 시스템의 주기인 1초 동안 스왑을 통해 확보할 수 있는 메모리양은 식 (5)에 의해 약 50M라는 계산을 할 수 있다.

$$128 : 2.5 = x : 1$$
$$x = 128/2.5 \quad (5)$$

이 값을 바탕으로 지연된 할당에서는 부족한 메모리양만을 할당받은 가상머신이 발생한 경우 그로부터 메모리를 확보할 때까지 지연된 할당이 필요한 메모리양을 1초에 50M씩 감소시킨다. 예를 들어 시점 t에 할당받지 못한 메모리양이 128MB이었다고 한다면 시점 $t+1$에서는 128-50M의 메모리가 부족한 상태가 되고 이 시점에서 다시 할당받지 못한 메모리 32MB이 발생한다면 시점 $t+1$에서의 지연된 할당이 필요한 메모리양은 128-50+32MB이 된다. 계속해서 이후 시점 $t+3$에서는 128-50+32-50-50MB의 지연된 할당이 필요한 메모리가 남게 된다. 부족한 메모리를 할당받은 시점을 T, 특정 가상머신 i가 부족한 메모리를 할당받은 시점을 T라고 할 때, 가상머신 i가 시점 T에 예측했던 메모리 사용량을 $P_{i,T}$, 가상머신 i가 시점 T에 할당받기로 계산된 메모리양을 $A_{i,T}$라고 표기한다면 현재의 시점 t에서 지연된 할당이 필요한 메모리양은 식 (6)으로 나타낼 수 있다.

$$L_{i,t} = \begin{cases} (P_{i,T} - A_{i,T}) - 50(T-t), & P_{i,T} - A_{i,T} > 50(T-t) \\ 0 & , \quad otherwise \end{cases} \quad (6)$$

식 (6)에 의해 계산되고 유지되는 $L_{i,t}$ 값은 동적할당 시스템이 확보한 메모리가 존재할 때 가장 먼저 확인되고 지연된 할당이 필요한 가상머신을 판단하고 필요한 양만큼 할당해준다.

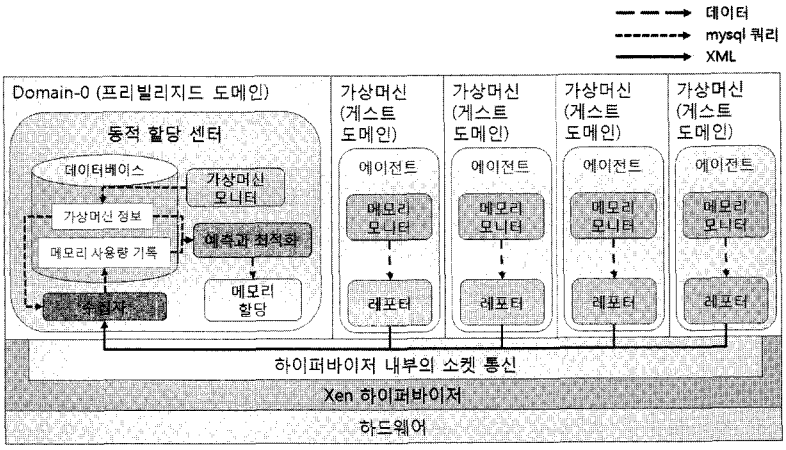

그림 9 동적 할당 시스템의 구성

4. 성능 평가

본 장에서는 메모리 정적 할당 시스템 및 간단한 규칙에 의한 동적 할당 시스템의 성능을 측정하고 본 논문에서 제안하는 동적 할당 시스템의 성능 측정 결과와 비교 분석한다. 가상화 환경으로는 Xen 3.2.1 버전을 사용하였다. 성능 평가의 환경은 두 개의 Intel Xeon 2.33GHz Quad-core CPU, 4GB 메모리를 갖고 있는 기계이다. 본 논문의 실험을 위하여 가상머신들은 총 2GB의 메모리를 사용할 수 있고 각각 하나의 CPU를 할당받도록 제한하였다. 또한 한 가상머신에는 최대 1GB의 메모리가 할당될 수 있도록 설정하였다.

4.1 성능 평가 실험을 위한 시스템 구성

그림 9는 동적 할당 시스템의 전체적인 구성을 보여준다. 우선 메모리 사용량을 측정하기 위해 각 가상머신에서 자신의 메모리 사용량을 측정한다. 이때 메모리 관찰이 각 가상머신에서 수행되는 것은 Xen이 고수준의 메모리 자원 분리를 제공하기 때문에 해당 가상머신 외의 다른 가상머신에서는 메모리 사용량을 알 수 없기 때문이다. 이러한 특징은 Xen 시스템을 관리하는 관리(Privileged) 도메인 Domain-0에서도 마찬가지이다. 관찰된 메모리 사용량은 레포터에게 전달되고 동적 할당 센터의 수집자에게 다시 전달된다. 동적 할당 센터의 수집자는 모든 가상머신의 레포터로부터 주기적으로 각 가상머신의 메모리 사용량을 수집하고 데이터베이스에 저장한다. 이 데이터베이스는 mysql 14.12[11]를 사용하여 구성하였고 모든 데이터는 데이터베이스를 통해 저장되거나 참조된다. 수집자로부터 데이터베이스에 기록된 데이터를 바탕으로 동적 할당 센터에서는 가상머신의 다음 메모리 사용량을 예측하고 예측된 값을 바탕으로 가상머신의 메모리 할당량을 조절한다. 만약 할당이 불가능한 경우 메모리 이용률을 극대화시킬 수 있는 할당 조합을 찾아 할당한다.

4.2 시스템 성능 측정 및 분석

메모리의 동적 할당 시스템은 탑재된 각각의 가상머신에서 서버가 정상적으로 동작하도록 자동으로 메모리의 할당량을 조절해주는 시스템이다. 성능 측정에서 사용된 가상머신의 메모리 워크로드는 그림 10의 좌측 그래프에 나타나 있는 eLuna 웹서버의 실제 메모리 워크로드 데이터[12]를 이용하였다. 그리고 메모리 요구량을 실제로 할당하여

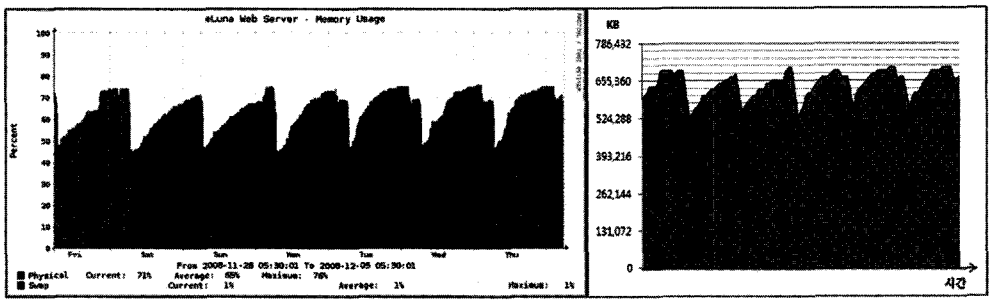

그림 10 eLuna 웹서버의 메모리 사용량 워크로드와 실험에 사용한 메모리 워크로드 수치

사용하는 간단한 서버/클라이언트 프로그램을 벤치마크 프로그램으로 사용하였다. 그림 10의 우측 그래프는 좌측 그래프에서 추출한 수치를 바탕으로 실험에 사용할 메모리 워크로드를 나타낸 그래프이다.

성능 측정의 척도인 메모리 이용률은 관리 도메인을 제외한 모든 가상머신에 대하여 측정한다. 따라서 총 메모리는 2GB가 되고 각 가상머신이 사용한 사용량들의 총합을 통해 메모리 이용률을 구한다.

4.2.1 메모리의 정적 할당

메모리의 정적 할당이란 가상머신을 탑재하는 시점에 설정된 메모리를 고정시켜두고 사용하는 것을 의미한다. 정적 할당의 성능 측정 실험에서는 각 가상머신에 1GB 메모리를 고정 할당해주고 메모리 워크로드를 적용하여 가상머신들의 메모리 이용률을 구하였다. 정적 할당의 경우 두 개의 가상머신을 초과하여 탑재하는 것은 불가능하다. 이 두 가상머신 VM01과 VM02에 각각 그림 11의 메모리 워크로드를 요구하였다.

그림 12의 그래프는 그림 11의 메모리 워크로드에 대하여 각 가상머신들의 메모리 사용량과 총합을 표시한 그래프이고 그림 13은 시스템 전체의 메모리 이용률을 표시한 그래프이다. 메모리 이용률 그래프에 따르면 메모리 정적 할당에서는 유휴메모리가 많이 발생하여 그 이용률이 평균 약 0.68에 지나지 않는다는 것을 알 수 있다.

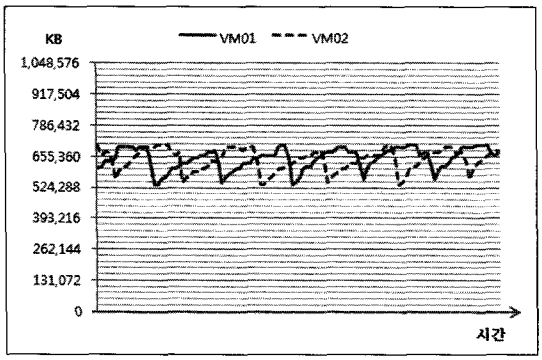

그림 11 메모리 정적 할당 방식의 두 가상머신에 대한 메모리 워크로드

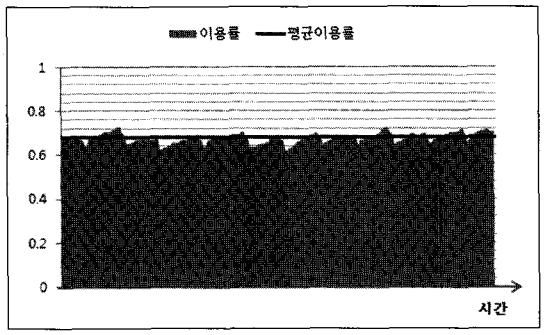

그림 13 메모리 정적 할당의 메모리 이용률

4.2.2 메모리의 초과 할당

본 논문에서 제안하는 메모리의 동적 할당과 비교하기 위하여 동적으로 메모리를 할당하지만 현재 메모리 사용량에 기반을 두는 간단한 초과 할당 시스템을 구성하여 실험하였다. 초과 할당도 동적 할당 방식의 일종이므로 정적 할당 방식보다 많은 수의 가상머신을 운영할 수 있게 되었다. 그림 14의 메모리 워크로드를 이용하여 이전 주기의 메모리 사용량에 대한 4MB 초과 할당을 실험하였다.

그림 15는 4MB 초과 할당 실험에서 측정한 각 가상머신들의 메모리 실제 사용량과 그림 14의 메모리 워크로드를 표시한 그래프이다. 이 그래프를 통하여 탑재된 모든 가상머신들이 검은 실선으로 표시된 요청 메모리 워크로드보다 더 적은 어두운 면의 메모리만을 사용했음을 알 수 있다. 부족한 메모리양은 메모리 워크로드와 밝은 면의 실제 메모리 사용량 차이를 통해 알 수 있고 가상머신의 운영체제에 의하여 스왑이 수행되는 결과를 가져오게 된다. 그림 15의 그래프에서 메모리 사용량 그래프가 지연되지 않은 것은 메모리 벤치마크에서 요구한 메모리 워크로드보다 적은 메모리만을 사용할 수 있을 경우 남은 메모리만을 사용하여 스왑의 발생을 최소

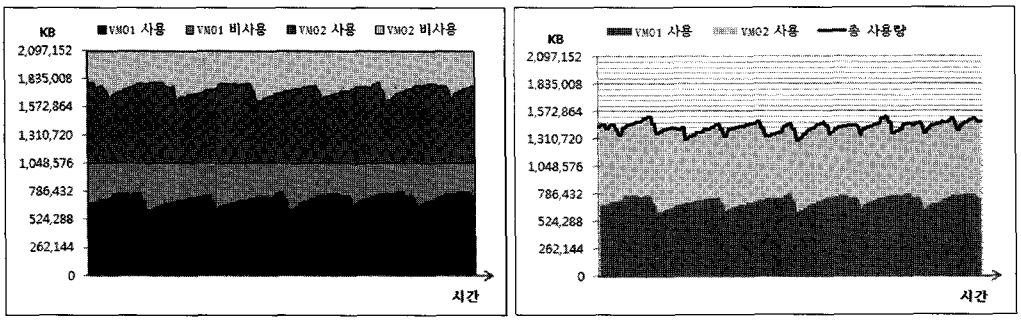

그림 12 메모리 정적 할당의 두 가상머신이 사용한 실제 메모리 사용과 총 사용량

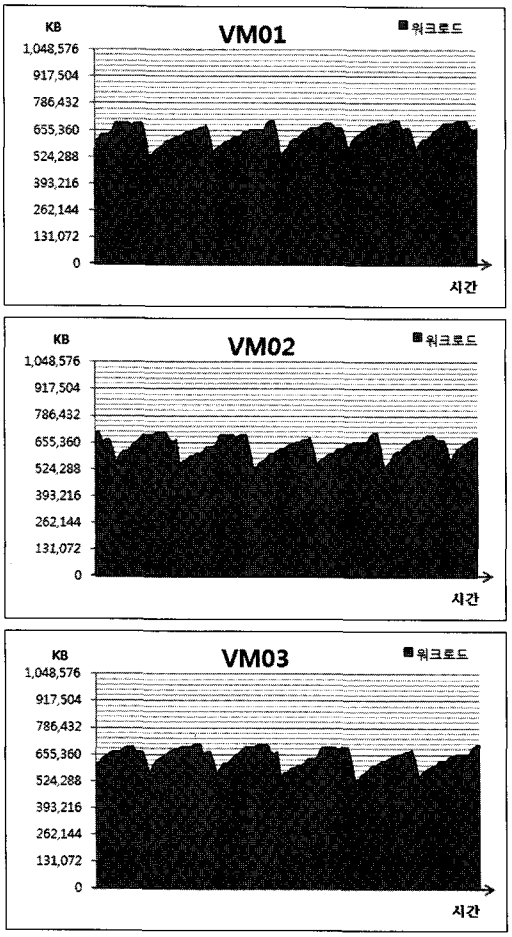

그림 14 세 가상머신의 메모리 워크로드

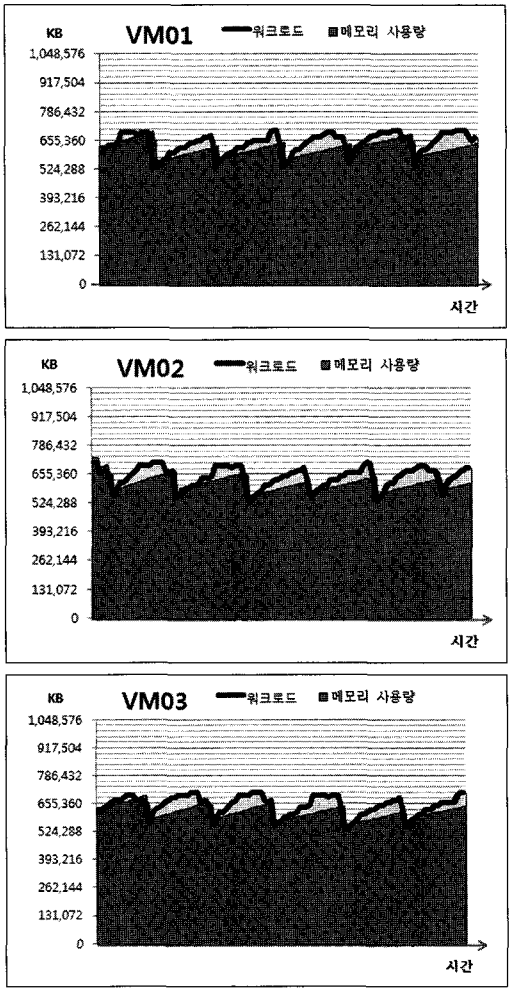

그림 15 4MB 초과 할당의 메모리 사용량과 워크로드

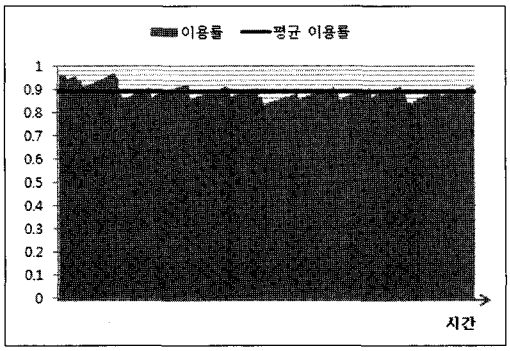

그림 16 4MB 초과 할당의 메모리 이용률

화시켜 실험을 수행하였기 때문이다. 실제로 서비스가 수행되는 경우에 메모리 스왑이 발생한다면 상당한 서비스 지연이 일어나게 된다.

4MB 초과 할당의 메모리 이용률은 그림 16에서 나타난 것과 같이 약 0.89로 측정되었다. 이는 앞서 정적 메모리 할당의 메모리 이용률 0.68보다 높은 수치로 메모리 자원을 더욱 효과적으로 활용하였다고 결론지을 수 있다. 하지만 4MB 초과 할당의 경우 가상머신의 서비스가 정상적으로 동작하지 못한다는 것을 알 수 있었고 다음으로 더 많은 양을 초과 할당하는 8MB 초과 할당에 대하여 실험을 수행하였다.

8MB 초과 할당 실험에서도 그림 14의 메모리 워크로드를 각 가상머신에 요청하여 실험 결과를 측정하였다. 그림 17은 8MB 초과 할당에 대해서 측정한 가상머신별 실제 메모리 사용량 그래프이다. 이 그래프에서도 4MB 초과 할당과 마찬가지로 메모리 요구량보다 적은 메모리를 할당받아 사용하였기 때문에 모든 가상머신들이 정상적인 서비스를 제공하지 못하였음을 알 수 있다.

8MB 초과 할당의 메모리 이용률은 그림 18에 나타난 것과 같이 약 0.92로 측정되었다. 이 이용률의 향상을 통

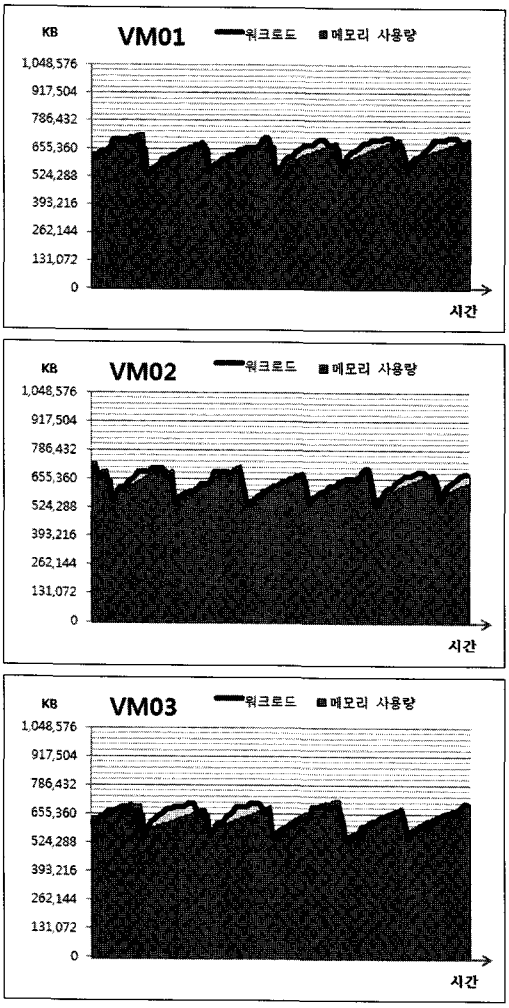

그림 17 8MB 초과 할당의 메모리 사용량과 워크로드

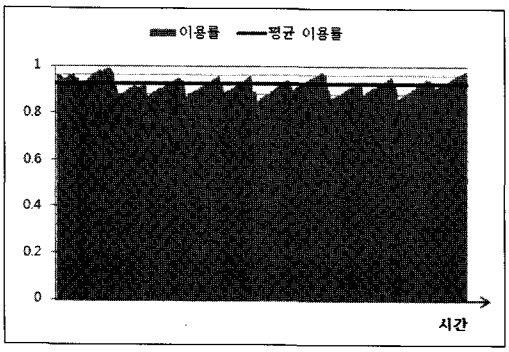

그림 18 8MB 초과 할당의 메모리 이용률

해 4MB 초과 할당보다 8MB 초과 할당의 경우가 서비스 요구를 잘 충족시켜주었다고 결론을 내릴 수 있다.

메모리 초과 할당의 메모리 사용량 그래프 그림 15와 그림 17을 통해 메모리 워크로드 증가량이 초과 할당량보다 많은 경우 서버가 사용하고자 하는 메모리를 충분히 제공하지 못한다는 것을 알 수 있다. 하지만 큰 폭의 메모리 워크로드 증가량에 대비하여 초과 할당량을 늘여주는 것은 메모리 낭비를 발생시켜 이용률을 떨어뜨리는 결과를 가져온다. 충분히 제공받지 못해 발생하는 메모리 부족은 가상머신 서버의 서비스 성능 저하를 가져오기 때문에 이러한 방식을 실제 서버 통합 환경에 적용하기에는 무리가 있다. 그림 19와 그림 20은 각각 4MB와 8MB 초과 할당된 메모리 및 실제 사용량을 표시한 그래프이다. 이 그래프들을 살펴보면 초과 할당된 메모리가 여러 가상머신에 걸쳐 할당되므로 발생하는 유휴메모리양이 지속적으로 존재하여 메모리 이용률에 영향을 미치게 된다는 것을 알 수 있다.

4MB와 8MB 초과 할당 실험에서 초과 할당으로 인

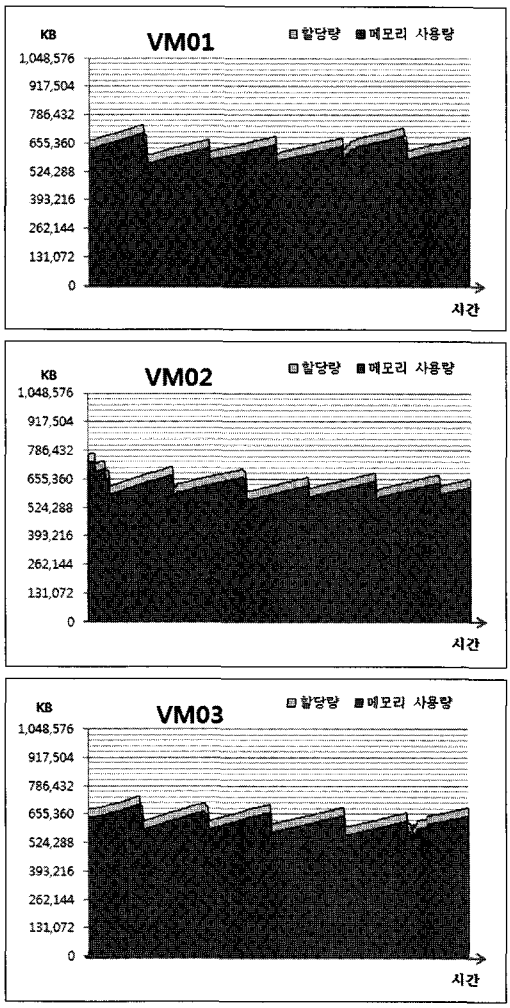

그림 19 4MB 초과 할당의 메모리 할당량과 실제 사용량

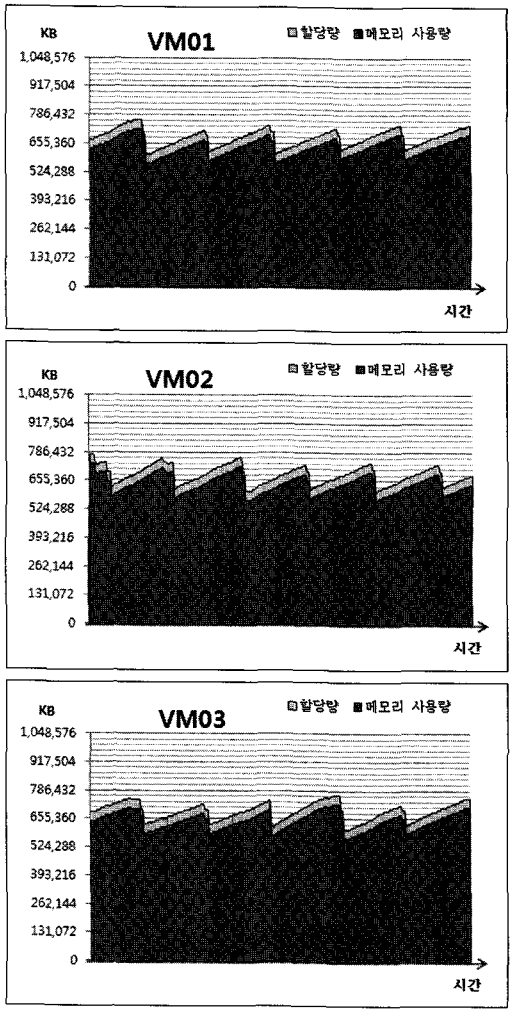

그림 20 8MB 초과 할당의 메모리 할당량과 실제 사용량

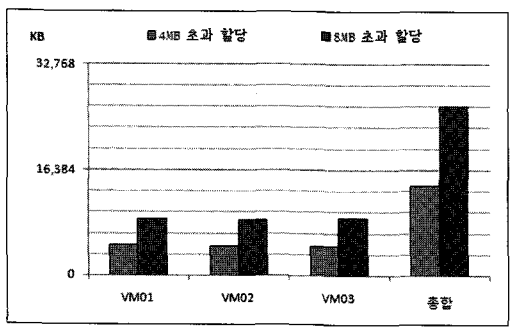

그림 21 초과 할당의 메모리 낭비

하여 낭비된 메모리양, 즉 할당량과 실제 사용량 차이의 평균을 그림 21의 그래프에 표시하였다. 4MB 초과 할당의 경우 세 가상머신에서 총 14MB 가량의 메모리 낭비가 있었고 8MB 초과 할당에서는 총 26MB 가량의 메모리 낭비가 있었다. 이러한 메모리 낭비는 가상머신에 할당은 되었으나 사용하지 않는 유휴메모리이고 결과적으로 메모리 이용률을 떨어뜨리는 요인이 된다.

4.2.3 메모리의 동적 할당

이 절에서는 본 논문이 제안하는 동적 할당을 적용한 시스템에 대한 실험 결과를 보인다. 본 논문의 동적 할당 성능을 측정을 위하여 초과 할당 실험에서 사용한 그림 14의 메모리 워크로드를 사용하였다. 그림 22는 메모리 워크로드에 대하여 각 가상머신이 사용한 실제 메모리양을 측정한 그래프이다.

그림 22의 동적 할당 실험 그래프에서는 그림 15이나 그림 17에서 볼 수 있었던 밝은 면을 거의 찾아볼 수 없고 이는 서버로 운영되는 가상머신이 요구한 메모리를 비교적 정확하게 제공하였음을 의미한다. 따라서 동적 할당 시스템에서는 서버의 성능 저하가 발생하지 않았다고 할 수 있다. 동적 할당에서의 메모리 이용률은

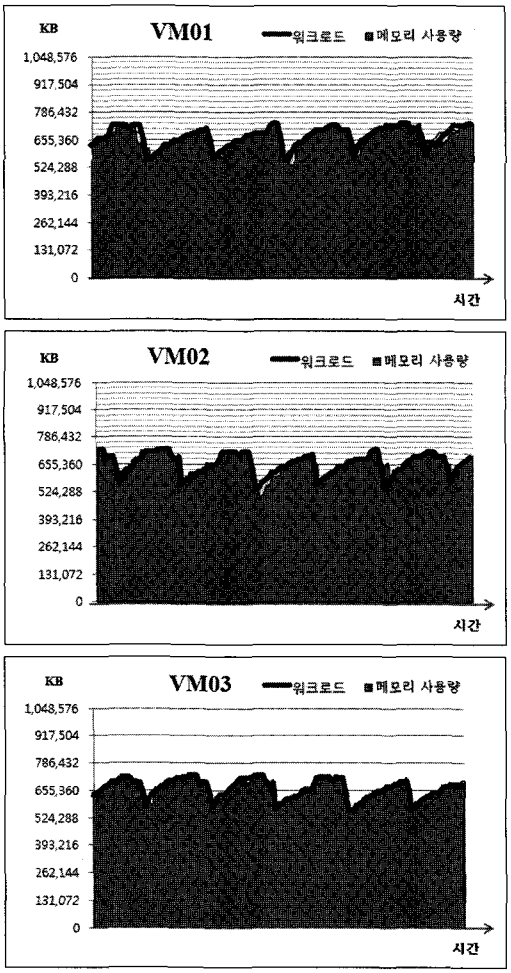

그림 22 동적 할당의 메모리 사용량과 워크로드

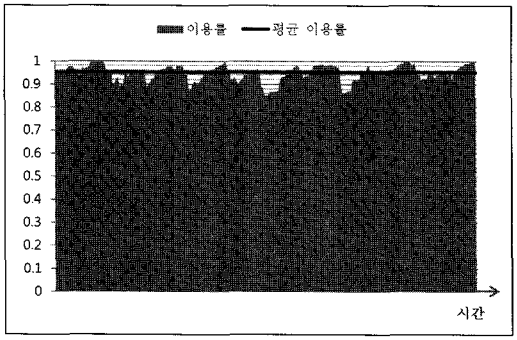

그림 23 동적 할당의 메모리 이용률

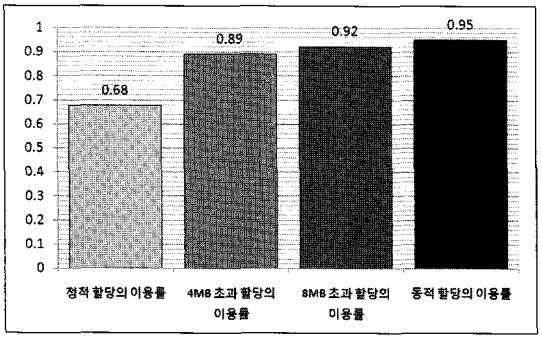

그림 24 메모리 할당 방식에 따른 이용률 비교

그림 23에서 볼 수 있듯이 약 0.95로 높게 나타났다. 이 수치는 정적 할당의 메모리 이용률과 비교하여 약 1.4배의 수치이고 상당한 메모리 이용률 향상을 얻을 수 있다는 것을 알 수 있게 해준다. 그림 24는 각 실험에서 얻은 메모리 이용률을 표시한 그래프이다.

5. 결론 및 한계

기존의 시스템 가상화를 통한 시스템 자원의 활용에 대한 연구는 CPU 자원에 대해서는 많은 연구들이 진행되어 왔으나 메모리 자원의 효율적인 사용에 대한 연구는 매우 부족한 실정이었다. 하지만 메모리 자원 역시 시스템 자원에서 중요한 자원이며 메모리를 충분히 확보하지 못하고 스왑이 발생하게 되는 경우 CPU와 같은 다른 자원의 사용에 영향을 미칠 수 있다. 따라서 본 논문에서는 메모리 자원에 초점을 맞추어 가상머신에서 발생할 수 있는 스왑 동작을 최소화시키면서 메모리의 이용률을 최대화시킬 수 있는 시스템을 제시하였다. 메모리의 동적 할당 시스템을 위하여 예측 기법인 AR 모델을 사용하였으며 추가 메모리 요청량 보다 적은 메모리양을 사용할 수 있는 경우, 즉 할당해주어야 할 메모리가 부족한 경우 메모리의 할당량을 최적화하여 할당하기 위해 개미 군집 알고리즘을 사용하였다. AR과 개미 군집 알고리즘의 파라미터는 실제 메모리 사용량 데이터에 기반을 두어 가장 적합한 결과를 도출할 수 있는 값을 실험을 통해 얻어서 사용하였다. 본 논문에서 제안한 성능을 검증하기 위해 기존에 사용하던 정적 할당 방식과 간단한 동적 할당 방식에 대한 성능을 측정하고 결과를 분석하여 각각의 문제점들을 살펴보았다. 그 결과 본 논문에서 제안한 메모리의 동적 할당 방식이 더 많은 수의 가상머신을 서비스의 성능 저하 없이 운용할 수 있으며 메모리 이용률을 향상시킬 수 있음을 확인할 수 있었다.

하지만 본 논문에서 구현된 동적 할당 시스템은 메모리 할당량 조절에서 발생하는 오버헤드에 대한 대책이 마련되어 있지 않다. 동적 할당 시스템이 1초 간격으로 주기적인 동작을 하는데 있어서 모니터링 된 메모리 사용량을 가상머신으로부터 수집하고 계산하는 과정에서 필요한 CPU의 소모는 Domain-0에서 약 17%이고 각각의 가상머신에서 1% 미만으로 무시해도 될 수치이다. 동적 할당 시스템이 가상머신의 메모리 할당량을 조절하기 위해서 벌루닝을 수행하는 경우 미리 확보된 메모리로부터 1024MB의 메모리를 벌루닝 하여 가상머신에 할당 또는 가상머신으로부터 회수하는 경우 필요한 시간은 평균 1초 미만이다. 또한 이 벌루닝에 사용되는 CPU 소모는 128MB 메모리의 경우 각 가상머신은 평균 약 5%, Domain-0는 평균 약 16.3%이다. 가상머신에서 동적 할당 시스템에 의해 메모리가 부족하지 않도록 할당되어 스왑이 발생하지 않고 동작한다고 가정하고 가상머신이 128MB씩 메모리 사용량이 변한다고 가정하였을 때, 결과적으로 실제 이 시스템이 동작하는 과정에서 발생하는 CPU 소모를 측정해본 결과 가상머신마다 평균적으로 약 16%, Domain-0에서 평균 약 31.5%가 사용되는 것을 알 수 있었다. 이러한 CPU 소모가 본 논문에서 제안하는 동적 할당 시스템을 1초 주기로 운영하는데 필요한 오버헤드라고 할 수 있지만 가상머신에서 스왑이 발생하게 되는 경우와 비교하였을 때 더욱 안정적으로 서비스를 운영할 수 있을 것으로 판단된다. 또한 동적 할당 시스템의 주기를 더욱 길게 잡고 AR 모델에서 더 정확한 예측을 할 수 있도록 차수와 데이터를 늘려 더 장기적인 예측을 함으로써 할당을 최적화하여 할당 빈도를 줄여 이 오버헤드도 더욱 줄일 수 있을 것이다. 이 주기는 시스템 파라미터로써 시스템을 엄격하게 할당을 적용할 것인지 아니면 느슨하게 할 것인지에 따라 오버헤드가 트레이드-오프(Trade-off) 관계를 갖게 된다. 하지만 1초를 주기로 하였을 경우의 오버헤드를 측정해본 결과를 미루어볼 때 가상머신이 사용할 수 있는 최대의 메모리를 할당해주는 정적 할당의 방식인 경우 동적 할당에 비해 적은 수의 가상머신

과 낮은 메모리 이용률을 갖게 되지만 가상머신 내에서 스왑이 발생하지 않기 때문에 추가적인 CPU 소모가 없다는 점과 비교하였을 때 동적 할당으로 인한 CPU 소모 오버헤드가 발생한다고 할 수 있다. 결국 정적 할당 방식과 동적 할당 방식의 효율적인 메모리 이용 및 더 많은 가상머신 운영 능력과 대비해서 CPU 오버헤드에 대한 트레이드-오프 관계를 갖고 있다고 할 수 있다.

다음으로 본 논문의 동적 할당 시스템은 메모리 프래그멘테이션(Fragmentation) 문제가 언급될 수 있다. 이 시스템에서 벌루닝으로 다른 가상머신에 할당되어 있던 메모리를 회수한 다음에 필요한 가상머신에게 할당하기 때문에 가상머신에 할당된 메모리가 연속적으로 존재하기는 불가능하다. 하지만 실제 Xen의 메모리 가상화 방식은 쉐도우 페이지 테이블(Shadow page table)[1]을 사용하기 때문에 이 시스템에서의 메모리 조각화 문제는 가상화로 인한 오버헤드와 마찬가지라고 할 수 있다.

본 논문에서 제시한 메모리의 동적 할당 시스템은 Xen 가상화 환경을 이용하는 서버 환경에서 메모리를 효율적으로 관리하여 더욱 많은 가상머신을 탑재할 수 있게 한다. 기존의 정적인 가상머신 메모리 할당 방식에서 탑재하고 운영할 수 있던 가상머신에 비해 더욱 많은 수의 가상머신을 탑재하고 가상머신의 내부에서 운영되는 서버가 서비스를 제공하는데 있어서 성능에 지장이 없도록 한다. 특히 가상머신 내에서 파일 서버나 방송 컨텐츠 같은 멀티미디어 서버처럼 다량의 메모리를 필요로 하는 서비스의 경우 본 논문의 메모리 동적 할당 시스템이 큰 도움을 줄 수 있을 것으로 생각된다.

참 고 문 헌

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebuer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," Nineteenth ACM Symposium on Operating Systems Principles (SOSP), Oct. 2003.

[2] David Chisnall, "The Definitive Guide to the Xen Hypervisor," pp.75-96, 2007.

[3] AR, http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/ar/

[4] Hanxiao Shi, "Solution to 0/1 Knapsack Problem Based on Improved Ant Colony Algorithm," *2006 IEEE International Conference on Information Acquisition*, pp.1062-1066, Aug. 2006.

[5] Peiyi Zhao, Peixin Zhao, Xin Zhang, "A New Ant Colony Optimization for the Knapsack Problem," *2006 IEEE 7th International Conference on Computer-Aided Industrial Design and Conceptual Design*, pp.1-3, Nov. 2006.

[6] C. Waldspurger, "Memory Resource Managemet in VMware ESX Server," OSDI, Boston, 2002.

[7] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration," *4th USENIX Symposium on Networked Systems Design & Implementation*, pp.229-242, April 2007.

[8] Jin Heo, Xiaoyun Zhu, Pradeep Padala, and Zhikui Wang, "Memory Overbooking and Dynamic Control of Xen Virtual Machines in Consolidated Environments," IFIP/IEEE Symposium on Integrated Management (IM'09) mini-conference, Jun. 2009.

[9] 시계열, http://en.wikipedia.org/wiki/Time_series

[10] Terence C. Mills, "Time Series Techniques for Economists," Cambridge University Press, 1990.

[11] mysql, http://www.mysql.com/

[12] 메모리 워크로드, http://graphs.eluna.org/index.pl?rrd=03_mem

이 권 용

2007년 서강대학교 컴퓨터공학과 졸업(공학사). 2009년 서강대학교 컴퓨터공학과 대학원 석사과정 졸업(공학석사). 2008년~현재 서강대학교 컴퓨터공학과 대학원 박사과정. 관심분야는 시스템 가상화, 임베디드 시스템

박 성 용

1987년 서강대학교 컴퓨터학과(공학사) 1994년 미국 Syracuse University 대학원(공학석사). 1998년 미국 Syracuse University(공학박사). 1998년~1999년 미국 Bell Communication Research 연구원 1999년~2008년 서강대학교 컴퓨터학과 부교수. 2008년~현재 서강대학교 컴퓨터학과 정교수. 관심분야는 Autonomic Computing, Peer to Peer Computing, High Performance Cluster Computing and System