가상화 환경에서 부하균형을 위한 가상머신 동적 재배치

(Dynamic Relocation of Virtual Machines for Load Balancing in Virtualization Environment)

사 성 일 † 하 창 수 †† 박 찬 익 †††
(Seong-il Sa) (Changsu Ha) (Chanik Park)

요 약 서버 가상화 기술에 의한 서버 통합은 효율적인 자원 사용에 따른 비용적인 장점이 있다. 그러나 하나의 물리적 장치에 여러 개의 서버가 가상머신으로 함께 동작함으로써 더욱 복잡한 부하특성을 가지게 되었다. 따라서 이를 해결하기 위한 효율적인 자원관리 방법이 요구된다. 이러한 문제에 대한 해결 방법으로 제안된 것이 가상머신 이동(live migration)[1,2]을 이용한 가상머신 동적 재배치 기법이다[3,4]. 본 논문은 가상머신 동적 재배치 알고리즘에 있어서 각 자원요소(CPU, network I/O, memory)들의 활용률을 다차원 공간 상에서 분석하여 조율함으로써 서버통합의 자원 효율성을 증가시키는 방법(Server consolidation optimizing algorithm)을 제안하고 있다. 실험을 위해서 여러 대의 통합서버와 수많은 서비스를 생성하여야 하는 어려움이 있기 때문에 본 논문에서는 기업환경에서의 서버 가상화 프로젝트 경험을 바탕으로 서버의 부하변화와 유사한 패턴의 모니터링 데이타들을 정의하여 수치적인 시뮬레이션을 통해 sandpiper[3]와 SCOA 알고리즘의 부하 균형에 대한 효율성을 비교하였다.

키워드 : 서버가상화, 서버통합, 가상머신 이동

Abstract Server consolidation by sever virtualization can make one physical machine(PM) to run several virtual machines simultaneously. Although It is attractive in cost, it has complex workload behaviors. For that reason, efficient resource management method is required. Dynamic relocation of virtual machine(VM)[3,4] by live migration[1,2] is one of resource management methods. We proposed SCOA(Server Consolidation Optimizing Algorithm) : a fine-grained load balancing mechanism worked on this dynamic relocation mechanism. We could obtain accurate resource distribution information through pointed physical machines on multi dimensional resource usage coordination, so we could maintain more balanced resource state. In this paper, we show the effectiveness of our algorithm by comparison of experimental results between SCOA and sandpiper[3] by software simulation.

Key words : Server virtualization, Server consolidation, Virtual machine migration

· 본 연구는 2008년도 두뇌한국 21사업과 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업(IITA-2008-C1090-0801-0045) 및 POSCO 산학연구(20078012)의 연구 결과로 수행되었음
· 이 논문은 2008 한국컴퓨터종합학술대회에서 '가상화 환경에서 부하균형을 위한 가상머신 동적 재배치'의 제목으로 발표된 논문을 확장한 것임

† 정 회 원 : 포항공과대학교 정보통신대학원
seongils@potech.ac.kr
†† 학생회원 : 포항공과대학교 정보통신대학원
csha@postech.ac.kr
††† 종신회원 : 포항공과대학교 컴퓨터공학과 교수
cipark@postech.ac.kr

논문접수 : 2008년 8월 25일
심사완료 : 2008년 10월 23일

정보과학회논문지: 시스템 및 이론 제35권 제12호(2008.12)

1. 서 론

최근 하드웨어 기술이 발전함에 따라 하나의 물리적 장치에 복수개의 서버를 구현할 수 있는 서버 가상화 기술[5,6]이 주목을 끌고 있다. 하나의 물리적 장치에서 복수개의 서버들이 자원을 공유함으로써 자원 활용에 대한 효율성이 증가할 뿐 아니라, 물리적 장치의 수가 줄어들게 됨에 따라 공간적인 효율성도 극대화할 수 있게 되었다. 이러한 비용적인 장점 때문에 서버 시스템의 교체나 증설을 계획중인 기업들에서 그 활용사례가 증가하고 있다[7].

서버 가상화는 자원을 효율적으로 활용할 수 있는 긍정적인 효과가 있는 반면에 가상화에 따른 부하발생[8,9]과 서버들 간에 자원활용에 있어서의 복잡성이 증가[10]하는 문제가 발생하기 때문에 이 복잡한 서버부하의 특성분석에 대한 연구와 더불어 자원 활용의 효율성을 증가시키기 위한 방법에 대한 연구가 시도되고 있다[3,4,11].

자원 활용의 효율성을 증가시키기 위한 접근법 중 Timothy Wood et al.에 의한 sandpiper[3]은 가상화 서버환경에서 통합서버 간의 실시간 자원 모니터링을 통해서 적절한 알고리즘을 이용하여 가상머신을 동적 재배치하여 자동으로 자원의 균형을 유지하는 기술이다. 본 논문은 이러한 가상머신 동적 재배치의 알고리즘에 있어서 어느 한 통합머신의 자원 사용량이 안전한 수준을 초과하는 현상이 발생할 때 이러한 자원의 편중을 해결하기 위하여 이동할 가상머신과 가상머신이 옮겨질 대상 통합서버를 선정하는 과정이 보다 효율적일 수 없는가에 주목하였다. sandpiper의 알고리즘에서는 가상머신과 통합서버가 사용중인 자원의 비중을 분석 하기 위해서 각 하드웨어 자원의 사용량을 종합하여 구한 하나의 대표 값을 사용한다. 결과적으로 이 대표 값을 사용하여 분석하고 결정함으로써 통합서버 간의 각 자원(CPU, network I/O, memory)들의 활용 분포가 균형을 이루는데 불리할 수 있음을 발견하였다.

본 논문에서 제안하는 알고리즘에서는 통합서버에서 사용되고 있는 각 자원들의 활용률을 바탕으로 다차원 좌표공간에 점으로 통합서버의 위치를 표현함으로써 어느 통합서버가 어떠한 자원에 편중되었고 어느 통합서버가 가장 많은 자원을 사용하고 있는지를 분석이 가능하며, 결국 이 다차원 좌표공간 상의 각 통합서버간의 거리를 좁히는 방법으로 가상머신을 이동함으로써 특정 자원에 대한 편중된 과부하 없이 균형적인 가상머신 이동이 가능하였다.

그림 1은 가상화 서버 기반구조에 대한 한 예를 보여주는 것이다. 여러 대의 물리적 장치들이 네트워크로 묶여 있으며, 모니터를 통해서 가상머신의 생성, 소멸, 이동, 상태점검 등의 관리 및 통제가 가능하도록 구성한 것이다. 본 논문에서 다루는 알고리즘들은 모니터로 수집된 각 장치들의 자원 사용량 정보를 토대로 가상머신의 이동을 결정하게 된다.

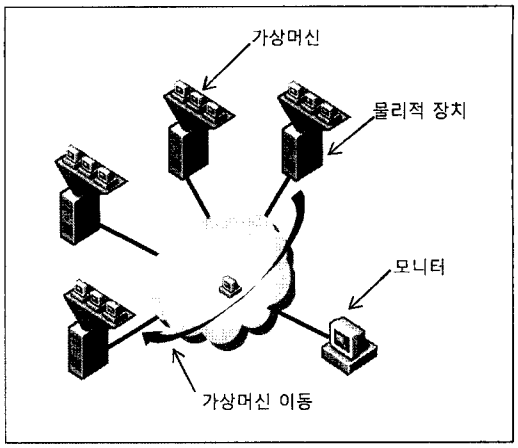

그림 1 가상화 서버 기반구조의 예

본 논문은 먼저 sandpiper의 알고리즘 방식과 개선된 알고리즘을 기술하고 있으며, 다음으로 자원들 간의 불균형 상황을 시뮬레이션 하여서 얻은 결과를 바탕으로 알고리즘의 차이점을 기술하고 있다.

2. 기존 연구

Timothy Wood et al.이 제안한 Sandpiper[3]는 가상머신들에 대한 실시간 자원활용 정보의 모니터링을 통해서 자원의 부족 상황(Hotspot)을 탐지하고, 이를 해소하기 위해 새로운 물리서버-가상머신 배치를 결정 한 후, 실시간으로 가상머신을 이동(live migration) 혹은 교체(Swap)시키는 기능을 구현하였다. Sandpiper의 주요 컴포넌트 중 이동관리자(Migration Manager)는 가상머신을 이동시켜야 할 통합서버(Source)와 이동 되어온 가상머신을 수용 하게 될 통합서버(Target)를 선정하는 기능을 하며, 이 과정을 통해서 결정된 결과에 따라 두 통합서버간의 가상머신 이동 및 교환이 수행되도록 하는 알고리즘을 제공하고 있다.

Sandpiper의 가상머신 배치 알고리즘은 다음과 같이 전개된다. 우선, 자원의 활용률 데이타를 기반으로 "Volume"이라는 다차원 하드웨어 자원의 활용률을 대표하는 값을 정의하고 이를 기준으로 통합서버 및 해당 가상머신을 정렬시킨다.

$$Volume = \frac{1}{1-cpu} * \frac{1}{1-mem} * \frac{1}{1-net}$$

(* 여기서 CPU, mem, net은 각 통합서버 및 가상머신 에 대한 각각의 해당 자원의 활용률을 의미한다.).

정렬이 끝나면, 정렬된 순서에 따라 가장 부하가 심한 ("Volume" 값이 가장 큰) 통합서버와 가장 부하가 적은 ("Volume" 값이 가장 적은) 통합서버 간에 가상머신의 이동 혹은 교환을 통해 부하 분산을 수행하게 된다.

이는 비교적 간단하게 통합서버의 부하를 경감 시키는 반면, 가상머신의 매핑정보 결정을 "Volume"이라는 단일 요소에 의존함으로써, 다중의 자원(CPU, 메모리, 네트워크 등) 각각에 대해 고려된 부하 분산이 이루어지지 못하는 문제가 발생할 수 있다.

Sandpiper외에 Norman Boboff et al.[4]의 연구 중 일부에 가상머신 동적 재배치 알고리즘이 제시되고 있으나, 이는 CPU라는 1차원적 자원에 대한 연구만 진행되었고, 다차원 자원의 고려 문제는 향후 계획 작업으로 미루어진 상태이며, 전통적인 최초적합 방식(First Fit Approximation)의 경험적 지식(Heuristic)을 사용하고 있다. 그 밖에 다중차원의 빈 패킹 문제의 측면으로 자원의 할당 문제를 고민한 연구들이 존재한다[11].

3. 제안하는 가상머신 동적 재배치 방법

본 논문이 제안하는 가상화 기반 서버통합 최적화 알고리즘(Server Consolidation Optimizing Algorithm 이하 SCOA)은, 최우선적으로 가상머신이 이동(live migration)이 가능하고 자원 활용률에 대한 변화의 예측을 단순화 할 수 있도록 하기 위해서 통합서버들 간의 물리적 특성과 용량이 모두 동일하다는 것을 전제하였다. 통합서버들의 자원 사용의 최적화는 다음과 같은 특성들을 바탕으로 정리될 수 있다.

첫째, 통합서버의 물리적 용량이 모두 동일한 환경일 경우 통합서버의 자원이 포화되지 않는 이상 통합서버 내에 존재하는 모든 가상머신의 자원 활용률들의 합은 일정하다. 다시 말해서, 가상머신이 어느 통합서버 상에 존재하더라도 동일 시간대에 이 가상머신의 자원 활용률은 동일한 값을 가진다는 것을 의미한다. 이는 가상머신의 자원 활용률이 다른 통합서버로 이동하더라도 유지됨으로써 가상머신의 이동으로 인한 통합서버들의 자원 활용률의 변화를 쉽게 예측할 수 있게 된다.

$$\sum u_1^n + \sum u_2^n + \ldots + \sum u_m^n = const$$

, where

PM_1 is servicing $VM_1^1 \ldots VM_1^n$ at time i

$\ldots PM_m$ is servicing $VM_m^1 \ldots VM_m^n$ at time i

u_m^n : Resource Utilization of VM_m^n at time i

둘째, 임의의 시간 개별 통합서버의 자원 사용률은 가상머신의 해당 자원 사용률의 합이다. 결국, 통합서버에서의 평균 자원 사용률 혹은 최악의 최대 자원 사용률(Worst Case Peak Utilization)은 통합서버가 가지는 가상머신들에 대한 평균 자원 사용률 혹은 최악의 최대 자원 사용률(Worst Case Peak Utilization)의 전체 합과 같다.

$$U_m = \sum u^n$$

, where

U_m : (Average/Worst Case Peak) Resource Utilization of PM_m,

u^n : (Average/Worst Case Peak) Resource Utilization of VM^n

셋째, 통합서버-가상머신의 매핑은 가상머신 이동(Migration)을 통해 어떠한 경우라도 재조합 가능하다.

이와 같은 3가지 특성을 전제로 통합서버간의 자원 활용의 균형을 이루기 위해서는 손쉽게 다음과 같은 정리에 도달할 수 있다.

정리 1. 개별 통합서버가 점으로 표시되는 다차원 자원 사용률(N-Dimensional Resource Utilization) 좌표상에서 임의의 시간에 개별 통합서버들 상호 간의 직선거리가 최소가 되는 위치에 놓이게 될 때 각 통합 서버는 자원 활용률 측면에서 가장 최적화된 상태에 놓이게 된다.

$$D = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$$

, where

D : Distance of 3 dimensional space where between consolidation servers,

$\Delta x, \Delta y, \Delta z$: Difference of each resource usage between consolidation servers

다차원 자원 사용률 공간에 개별 통합 서버를 표현할 경우 임의의 두 개별 통합서버간 직선거리의 의미는 자원 사용률에서 차이를 발생하고 있다는 것이고 차이가 발생하고 있는 통합서버간에 가상머신의 이동으로 자원의 사용률의 차이가 줄어든다면 다차원 자원 사용률 공간에서의 직선거리도 줄어들게 될 것이다. 즉, 각 통합서버들 간의 자원의 사용률이 고르게 분포된다면 다차원 공간에서 통합서버들간의 직선거리가 작게 분포될 것이고 이 상태가 자원 활용률 측면에서 가장 최적화된 것이다.

정리 1을 기반으로 SCOA는 다차원 자원의 사용률 좌표상에서 통합서버의 서로간의 직선거리가 최소화 되는 가상머신 조합을 최적화된 가상머신 조합으로 간주하고 이를 결과물로 산출하게 된다.

SCOA는 위에 설명한 방식으로 그림 2에서 보는 바와 같이 상대적으로 자원 활용률이 많은 통합서버에서

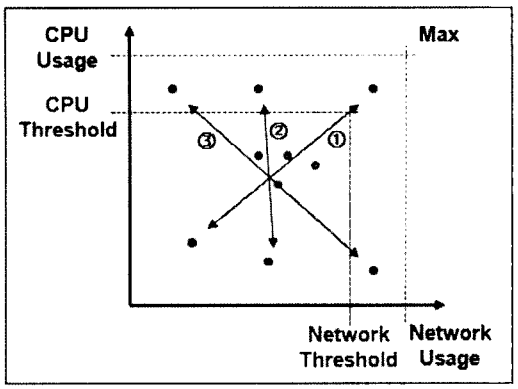

그림 2 서버통합 최적화 알고리즘의 개념

```
Sort_decending_PMs;
Sort_decending_VMs_in_PMs;
for i=1...N in VMs
  if overload(PMi)== true then
    for j=1...i+PER_PM_VM_CNT do
      PMk = find_most_longest_distance_PM(PMi)
      if migrate_fitness(Pmi,Vmi, PMk) != true then
        if most_loaded_resource(PMi)==most_loaded_resource(PMk) then
          swap(most_usage_VM(PMi), least_usage_VM(PMk));
        else
          swap(most_usage_VM(PMi), most_usage_VM(PMk);
        endif;
        sort_decending_PMs;
        sort_descending_VMs_in_PM;
        redo i;
      end if;
    end for;
  end if;
end for;
```

그림 3 알고리즘의 핵심 원리를 설명하는 모의코드

부터 ①→②→③순으로 통합서버와 가상머신의 매핑을 조정하게 된다. 세부적인 가상머신 매핑 수행 방법은 다음과 같다.

- 가상머신의 매핑 조정 대상이 될 소스(Source) 통합서버와 타겟(Target) 통합서버의 선정방식 : 자원 활용률이 가장 큰 통합서버(원점으로부터의 직선거리가 가장 큰) 순으로 과부하로 인식하는 통합서버를 소스 통합서버로 선정하고, 소스 통합서버와 다차원 좌표상 직선거리가 가장 먼 통합서버를 타겟으로 선정한다.
- 소스와 타겟 통합서버간의 가상머신 선정방식 : 두 개의 통합서버가 가장 많이 사용하고 있는 자원의 종류가 서로 다른 경우(예, 소스는 CPU, 타겟은 network I/O의 활용률이 타 자원의 활용률에 비해 비중이 큰 경우)는 소스 측 해당자원의 활용률이 가장 큰 가상머신을 타겟으로 이동시키고, 이동이 불가능한 경우(이동 후 타겟 통합서버의 부하가 소스 통합서버 보다 커지거나 사용자 정의 과부하 임계점을 넘어서는 경우)는 타겟 통합서버의 가상머신 중 비중이 높은 자원에 대해 활용률이 가장 큰 것과 교환 한다. 반대로, 두 개의 통합서버가 비중이 높은 자원의 종류가 같은 경우, 소스 통합서버의 해당 자원의 활용률이 가장 큰 가상머신을 이동시키거나, 이동이 불가능 할 경우 타겟의 가상머신 중 해당 자원의 활용률이 가장 작은 것과 교환한다.

그림 3은 위의 개념을 모의코드(Pseudo Code)로 표현한 것이다.

4. 실험 환경

실제환경에서 알고리즘에 따른 통합서버들간의 자원 변화를 관측하기 위해서는 여러 개의 서버머신 상에 다양한 서버들을 통합시키고 서비스들을 수행시켜 오랜 시간 동안 관찰하여야 한다. 알고리즘에 대한 비교 실험이 실제환경에서 여러 차례 비교 실험을 수행하는 것은 많은 시간과 비용을 요구한다. 그래서 본 논문에서는 가상적인 서버부하의 변화를 담고 있는 데이타를 만들고 알고리즘에 전달하여 그 결과를 반영하는 수치적인 시뮬레이션을 통해서 알고리즘의 차이를 분석하였다. 시뮬레이션을 정의하면서 가능한 실제환경과 유사한 환경이 되도록 서버 부하에 대한 데이타를 정의하였다. 시뮬레이션 환경에서 다양한 부하의 조합에서 여러 차례 실험을 수행하였으며, 대부분의 경우 평균적으로 유사한 결과를 보였지만, 그 중 알고리즘에 따른 결과의 차이를 가장 잘 보일 수 있는 실험 하나를 선택하여 본 논문에 기술하였다.

4.1 S/W 시뮬레이션

S/W 시뮬레이션에서는 대부분의 기능은 알고리즘의 주요부분인 수집된 정보의 분석과 분석에 따른 가상머신 이동의 대상을 선정하는 기능으로 구성되었다. 자원 활용률에 대한 정보 수집 과정은 이미 가정된 상황에 따라 가공된 데이타를 읽어 들이는 과정으로 대체하고 가상머신 이동의 동작은 알고리즘에 의해서 선택된 가상머신의 데이타에서 소속 통합서버 ID를 변경하는 것으로 대체하였다.

시뮬레이션은 시간단위로 통합서버의 자원활용률의 변화를 발생시키고 이에 따라 통합서버간의 자원 불균형이 발생되면 알고리즘에 의해서 서버 재배치가 일어나게 되는 과정을 보여주게 된다.

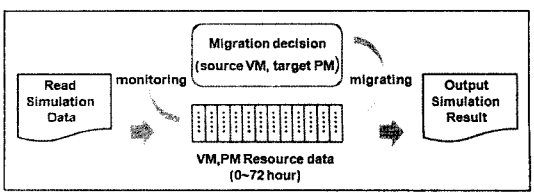

그림 4 S/W 시뮬레이션 구성도

4.2 서버 부하 정의

시뮬레이션을 위해서는 서버부하의 변화에 대한 데이타가 필요하다. 서버에서 사용하는 세부자원을 CPU, network I/O, memory로 대표하였다. 서버 데이타를 가공할 때 2가지 관점에서 고려하여 가공하였다. 첫째는 실제 서비스에서의 부하 변화를 고려하여 서버부하의 변화 형태를 정의하였고, 둘째는 알고리즘의 차이를 실험하기 위한 자원간의 활용률 차이를 발생시키는 것을 고려하였다.

먼저 서버부하 변화의 관점에서는 실제 기업환경의 서버가상화를 연구했던 경험을 바탕으로 그림 5와 같이 크게 3가지의 형태로 정의하였다. 업무 시간대를 기준으로 서버의 활용률이 점진적으로 증가하는 형태의 일반형(normal type)과 시간에 관계없이 일정한 안정형(steady type), 특정 시간대에 활용률이 급격히 증가하는 돌출형(ramp type)으로 정의하였다. 일반형과 안정형은 기업환경에서 경험한 형태이고, 돌출형은 일반적인 상황이 아닌 예외적인 상황으로 고려하였다. 대체로 일반형의 서버 비중을 크게 가지고 안정형과 돌출형의 서버 비중은 낮게 분포시켰다.

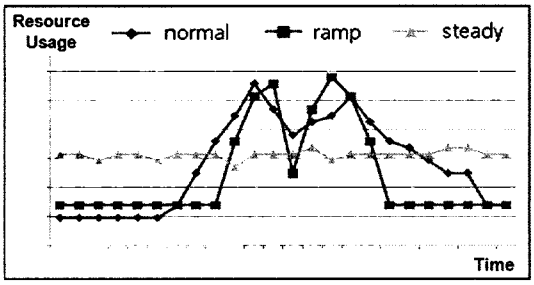

그림 5 시간에 따른 서버부하의 변화 패턴

다음으로 서버별 CPU, network I/O, memory의 자원 분배의 관점에서 3가지를 고루 사용하는 형태, CPU와 다른 한가지 자원을 균형 있게 사용하는 형태, 한 자원에 편중되어 사용하는 형태로 정의하였다. 알고리즘의 차이를 분석하기 위해서 자원의 편중이 잘 발생하도록 한 자원에 편중된 서버를 다수 배치하고 자원을 균형 있게 사용하는 서버의 비중은 낮추었다.

5. 성능 평가

6대의 통합서버(Physical Machine) 상에 32대의 가상머신(Virtual Machine)이 배치되어 운영 중인 서버 환경을 구성하였고, 각 통합서버들의 가상머신에서 서비스에 대한 요구가 증가하면서 통합서버간의 자원 불균형이 발생하고 있는 상황을 연출하였다.

표 1 32대 실험 가상머신들의 서버부하 특성

Main load	Workload type	Virtual Machine ID
CPU	normal	0,1,2, 18,19,20
	ramp	3,21
	steady	4,5,22,23
network I/O	normal	6,7,24,25
	ramp	8,26
	steady	9,27
memory	normal	10,11,28,29
	ramp	12,30
	steady	13,31
CPU, net, mem	Normal	14,15,32,33
CPU, net	normal	16,34
CPU, mem	normal	17,35

서버의 과부하를 인식하기 위한 각 자원들의 임계 값은 79%로 정의하였다. 서버의 과부하에 대한 임계값은 서버 운영자가 서버 부하에 의해서 서비스에 문제가 발생하기 전에 가상 머신 동적 재배치를 수행할 시점을 정의하는 것이다. 동적 재배치 알고리즘으로 해결해야 할 수준을 넘어 하드웨어 자원이 한계에 도달할 경우는 서버 증설과 같은 방법으로 대처하여야 할 것이다. 실험에서는 90% 자원 활용을 하드웨어 한계 시점으로 정의하여, 약10%의 추가적인 자원 여유(전체 20%) 내에서 가상머신 동적 재배치 알고리즘에 의해 자원 균형을 조율하고 있다고 가정하였다. 실제 운영의 관점에서는 더욱 높거나 낮은 임계값으로 운영될 수도 있다.

1시간 단위의 CPU, network I/O, memory의 자원 활용률을 72시간(3일간) 동안 모니터링하여 각 알고리즘별로 통합서버 단위의 자원사용이 어떻게 달라지는지를 결과로 출력하였다. 72시간 동안의 데이타를 실험하는 이유는 알고리즘에 의한 서버재배치가 통합서버간의 하루 주기의 부하 특성을 변화시키기 때문에 동적 재배치가 안정화 되기까지 부하 변화의 하루 주기가 2~3배를 측정한 것이다.

표 1은 실험에서 사용된 32대 가상머신의 부하 특성을 정리한 것이다. 그림 6은 각 통합서버별 CPU, network I/O, memory의 자원활용률의 시간적 변화를 보여준다.

5.1 실험 결과 분석

그림 6에서 보면, 통합서버0,3에서는 CPU에 의한 과부하가 발생하고 있고, 통합서버1에서는 network I/O에 의한 과부하가 발생하고 있음을 알 수 있다. 통합서버 2를 제외한 대부분의 통합서버에서 CPU의 활용률이 많은 것을 알 수 있으므로 현 상태에서 통합서버간의 자원 균형을 적절히 유지하기 위해서는 통합서버 1의 network 부하를 해결함과 동시에 통합서버1의 CPU 자원

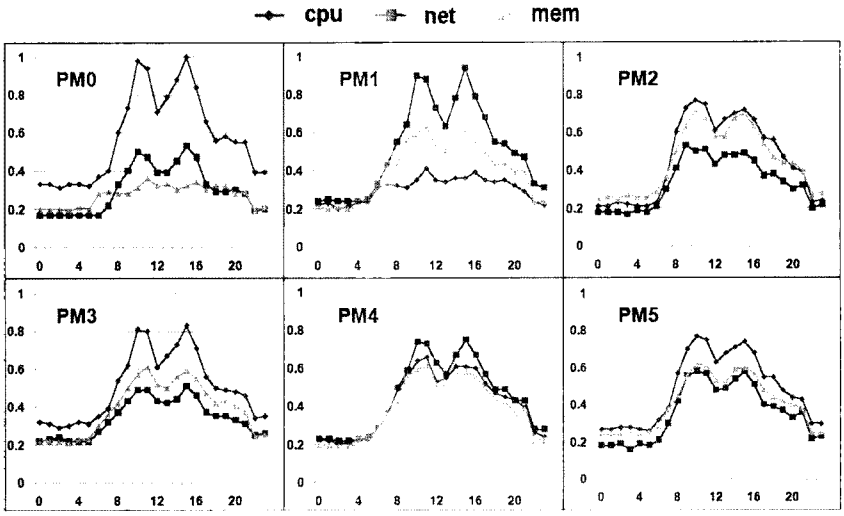

그림 6 통합서버별 시간에 따른 자원활용률 변화

여유를 잘 활용하여 전체적으로 과도한 CPU 활용률을 잘 분산할 수 있어야 할 것이다. 결국 이러한 상황은 각 자원별 활용률을 잘 분석할 수 있어야만 해결이 가능할 것이다.

표 2는 각 알고리즘에서 72시간 동안의 모니터링을 통해서 과부하에 대해서 어떻게 반응하였는지를 보여주는 결과이다. Sandpiper의 경우 과부하를 해결하지 못하는 경우가 2회 발생하고 있다.

그림 7에서 살펴보면 sandpiper 에서는 통합서버0 의 CPU 자원의 과부하가 해결되지 못하고 있는 것을 알 수 있다. 이것은 알고리즘에 따라 부하균형을 위한 대상 통합서버와 이동할 가상머신의 결정 과정의 차이를 보이고 있고 그에 따른 자원 균형의 효과가 다름을 알 수 있다.

표 3에서는 sandpiper에서 migration이나 swap을 수행하지 못하여 해결책을 찾을 수 없는 상태를 보여준다. 통합서버0의 CPU 과부하를 해결하기 위해서는 0.15 (=0.94-0.79)를 줄여야 하나 CPU 사용량이 제일 작은 통합서버3의 CPU 자원의 여유는 0.13(=0.79-0.66)으로 요구하는 CPU 자원에 비해서 여유가 부족하여 두 통합서버 간의 가상머신 이동으로는 문제를 해결할 수 없다.

표 2 72시간 알고리즘 동작 결과 (발생횟수)

	Overload	Migration	Swap	No solution
sandpiper	22	14	8	2
SCOA	23	17	6	0

Overload : 과부하 발생 횟수
Migration : 과부하를 해결을 위한 가상머신 이동 횟수
Swap : 과부하 해결을 위한 가상머신 교환 횟수
No solution : 알고리즘에 의한 해결 불가능 횟수

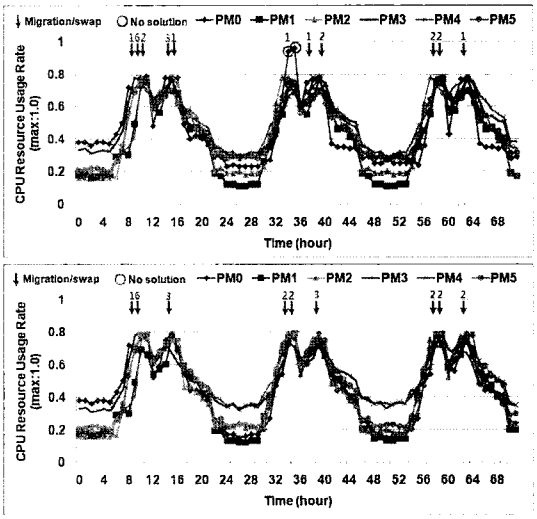

그림 7 sandpiper(위)와 SCOA(아래) 알고리즘에 의한 통합서버별 CPU 자원활용률 변화

이것은 이전의 발생한 sandpiper의 여러 차례 migration 동작이 통합서버0와 통합서버3 간의 CPU 자원의 편차를 직접적으로 줄여주지 못하여 swap으로도 해결할 수 없는 상황을 발생시킨 것이다. 이에 반해 SCOA에서는 CPU 자원이 적절히 균형을 이루고 있는 것을 알 수 있다.

그림 8은 초기 과부하를 발생하고 있는 상태의 전체 자원의 분산에 비해서 알고리즘 수행 후 자원 분산의 정도가 어떻게 변화하고 있는지를 보여주고 있다. SCOA에서는 CPU 자원 활용률이 최고에 이르는 시간대에서 CPU의 편차를 일관되게 유지하는 반면, sand-

표 3 Sandpiper의 No solution 상황에서의 비교

Time stamp	PM	cpu	net	memory
34 hour (sandpiper)	0	0.94	0.37	0.30
	1	0.75	0.77	0.61
	2	0.78	0.58	0.61
	3	0.66	0.54	0.37
	4	0.70	0.64	0.53
	5	0.73	0.73	0.69
34 hour (SCOA)	0	0.79	0.40	0.39
	1	0.76	0.65	0.51
	2	0.77	0.51	0.47
	3	0.72	0.65	0.51
	4	0.74	0.71	0.59
	5	0.78	0.70	0.66

piper에서는 CPU의 편차를 안정적으로 가져가지 못하고 급변하는 것을 알 수 있다. 이는 migration 동작에서는 CPU 자원의 요구량을 인식할 수 없기 때문에 편차를 커지게 만들기도 하고 다시 swap에 의해서 되찾는 과정의 결과이다.

실험결과에 의하면 Sandpiper는 migration에서는 자원요소 각각의 균형을 고려할 수 없고 swap에 전적으로 의존하게 된다. 따라서 각 자원 별로 통합서버간의 편중이 심하게 발생할 때에는 자원균형을 이루지 못하는 경우가 발생하게 되었다. 이에 비해 SCOA는 각 자원의 활용률의 변화에 반응하여 보다 많은 부하를 일으키고 있는 자원에 대해서는 집중적인 분산이 이루어지고 있듯이 자원 균형에 더욱 유연함을 보여주고 있음을 알 수 있다.

본 시뮬레이션에서 나온 결과를 확인해 보기 위해 실제 Xen환경에서 sandpiper의 가상머신 이동 관리자(migration manager)를 수정하여 본 논문에서 제안하는 가상머신 동적 재배치 방법을 적용하여 실험하였다. HTMLperf와 Web stress tool을 이용하여 가상머신에 CPU와 network I/O의 서버부하를 만들어 2개의 통합서버에 9개의 가상머신을 생성시킨 환경에서 실험하였다. 환경적인 제약으로 통합서버 2대 상의 실험밖에 할 수 없어서 시뮬레이션에서의 결과와 같은 분명한 차이를 확

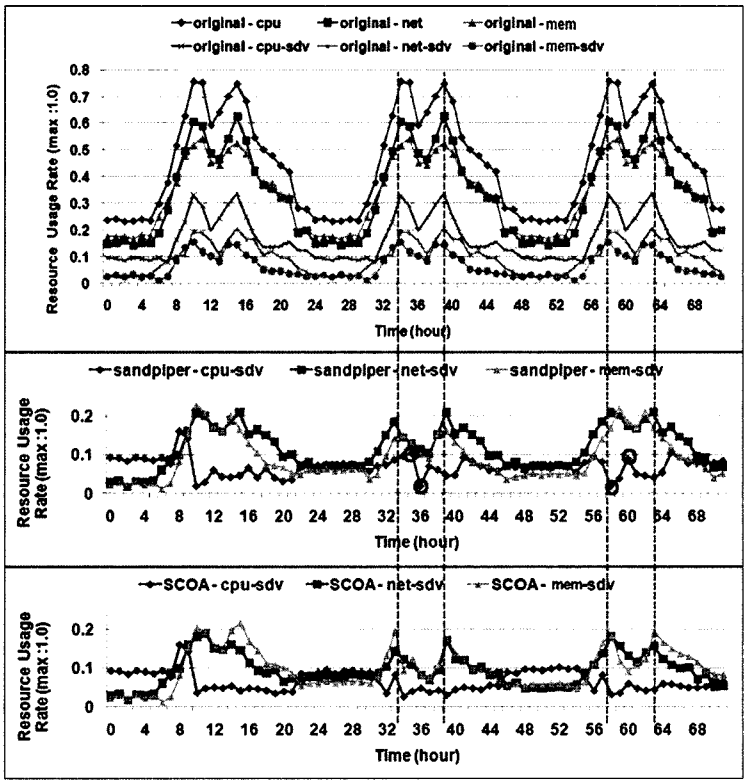

그림 8 시간에 따른 자원별 활용률의 표준편차(y축은 전체 가용한 자원을 1.0로 두었을 때 CPU, net, memory의 사용 비율을 표현한 값이다. Sandpiper는 최대사용량을 보이고 있는 시간대에서 상대적으로 자원 활용률이 높은 CPU 자원의 표준편차의 변화가 불안정하게 나타났다.)

인할 수는 없지만, 기존 Sandpiper에 비해서는 더 나은 결과를 보여주었다.

6. 결론 및 향후 연구

가상머신 동적 재배치 기술은 가상머신 이동을 통해서 통합 서버들간의 자원활용 효용성을 증가시키기 위한 방법이다.

본 논문에서 제안한 SCOA는 각 자원의 활용률로 구성된 다차원 공간상에 통합서버들을 점으로 표현하고 상대적인 거리 분석은 자원간의 복잡한 비교 문제를 단순화하면서 volume에 의한 자원 분배의 효율성보다 향상된 결과를 얻었다.

향후 연구에서는 알고리즘적인 개선보다는 가상머신 동적 재배치 기술의 유연성을 제공하기 위해서 동일 하드웨어 제원을 가진 서버머신 사이에서만 가능한 가상머신 이동의 제약사항을 해소하는 문제와, 가상머신 자원 활용률을 정확하게 예측하는 문제를 해결해야 한다.

참 고 문 헌

[1] J. Smith and R. Nair. "Virtual Machines: Versatile Platforms for Systems and Processes." Morgan Kaufmann, 2005.

[2] C.Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. "Live migration of virtual machines." USENIX NSDI '05, May 2005.

[3] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. "Black-box and Gray-box Strategies for Virtual Machine Migration," USENIX NSDI '07.

[4] Norman Boboff, Andrzej Kochut, Kirk Beaty. "Dynamic Placement of Virtual Machines for Managing SLA Violations," IEEE, 2007.

[5] 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 Proc. SOSP'03.

[6] http://www.xensource.com/ : Open source Xen.

[7] "미국 기업의 가상화 기술 도입과 활용 분석", 한국SW진흥원, 2007년 3월.

[8] L. Cherkasova and R. Gardner. "Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor," Proceedings of the USNIX Annual Technical Conference, April 2005.

[9] P.Apparao, S.Makineni, D. Newell. "Characterization of network processing overheads in Xen," 2nd International Workshop on Virtualization Technology in Distributed Computing (VTDC 2007).

[10] A. Menon et al. "Diagnosing Performance: Overheads in the Xen virtual Machine Environment," In First ACM/USENIX Conference on Virtual Execution Environments (VEE'05), June 2005.

[11] Guillermo A. Alvarez, Elizabeth Borowsky, Susie Go, Theodore H. Romer, Ralph Becker-Szendy, Richard Golding, Arif Merchant, Mirjana Spasojevic, Alistair Veitch, and John Wilkes. "Minerva: An automated resource provisioning tool for large scale storage systems," ACM Transactions on Computer Systems, 19(4):483-518, 2001.

사 성 일

1995년 성균관대학교 물리학과 학사. 2008년 포항공과대학교 정보통신대학원 석사 1995년~현재, ㈜포스데이타. 관심분야는 가상화, 유틸리티 컴퓨팅, 내장형 소프트웨어

하 창 수

2004년 경북대학교 전자전기공학부 학사 2002년~현재, ㈜휴원. 2007년~현재, 포항공과대학교 정보통신대학원 석사과정 관심분야는 가상화, 스토리지 시스템, 내장형 실시간 운영체제

박 찬 익

1983년 서울대학교 전자공학과 학사. 1985년 한국과학기술원 전자공학과(컴퓨터공학) 석사. 1988년 한국과학기술원 전자공학(컴퓨터공학)박사. 1988년~현재, 포항공과대학교 컴퓨터공학과 교수. 관심분야는 지능형 스토리지 시스템, 내장형 실시간 운영체제, 시스템 보안, 가상화