가상화 환경에서 동적 임계치 기반 가상 머신 이주 기법

최호근† · 박지수†† · 손진곤†††

요 약

가상화 환경에서는 물리적 자원을 여러 가상 머신이 같이 사용한다. 그러나 특정 가상 머신이 컴퓨팅 자원을 많이 쓰면 다른 가상 머신들이 동작하지 못하게 된다. 이러한 문제를 해결하기 위한 다양한 방법이 있다. 이 중 대표적인 것이 특정의 가상 머신을 다른 서버(이 서버를 타겟 서버라고 함)에 이주시키는 방법이다. 이는 가상 머신을 타겟 서버에 이주시키면서 서버의 과부하가 전이되는 현상이 있고, 가상 머신을 다시 다른 서버로 이주시켜야 하는 문제점이 있다. 본 논문에서는 이러한 문제점을 해결하기 위해 동적으로 임계치를 적용하여 이주 대상을 결정하는 알고리즘을 제안한다. 동적 임계치를 적용한 이주 알고리즘은 다음과 같은 특징을 가진다. 첫째, 서버의 CPU, 네트워크, 그리고 메모리 등의 자원 사용률의 변화에 따라 동적으로 임계치를 적용한다. 둘째, 서버에서 임계치를 초과한 자원을 기준으로 가상 머신 집합과 타겟 서버를 결정한다. 셋째, 타겟 서버의 자원 사용률을 기준으로 가상 머신을 결정한다.

주제어 : 가상화 환경, 동적 임계치, 가상 머신 이주

A Migration Method of Virtual Machines based Dynamic Threshold in Virtualization Environments

Hogun Choi† · JiSu Park†† · Jin Gon Shon†††

ABSTRACT

In an virtualization environment, several virtual machines use physical resources together. If a specific virtual machine uses to much of the computing resources, other machines may not be working properly. There are various method to solve this problem. Most representative study is to migrate a specified virtual machines to a different server, a target server. In this study, server load can be transferred to a target server by the remigrate of the load imposed on virtual machine. It is still problematic that virtual machine has to remigrate to a different server. This thesis has proposed the algorithm determining the remigration targets by applying dynamic thresholds to solve those problems. The migration algorithm applies dynamic thresholds according to the following criteria. Firstly, the usage of CPU, network and memory; secondly, decide the set of artificial machine and the target server based on the resources surpassed thresholds; thirdly, determine artificial machines based on the resource usage in the target server.

Keywords : Virtualization Environments, Dynamic Threshold, Virtual Machine, Migration

† 정 회 원: 한국방송통신대학교 석사
†† 정 회 원: 고려대학교 정보창의교육연구소 연구교수 (교신저자)
††† 종신회원 :한국방송통신대학교 교수
논문접수: 2015년 2월 10일 심사완료: 2015년 2월 24일 게재확정: 2015년 3월 26일
* 본 논문은 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원으로 수행되었음(No. 2014R1A1A2058888)

1. 서론

서버 가상화는 물리적 서버 위에 여러 개의 논리적인 가상 머신을 배치시켜 각 가상 머신에서 독립적인 운영체제를 운영하는 것이다. 서버 가상화 방법들은 모두 서버의 물리적인 자원을 가상화하여 가상 머신의 작업에 할당하고 서버들을 가상화 자원 풀(pool)로 관리한다[1]. 즉 서버에서 실행되는 여러 가상 머신이 자원을 공유해서 사용한다. 따라서 서버는 가상 머신들의 자원 사용률을 측정하여 서버의 자원을 할당하여야 한다[2].

특히 특정 가상 머신의 작업이 증가하면, 서버는 유휴자원을 가상 머신에 재 할당한다. 그러나 서버가 물리적인 자원 수용의 제약으로 가상 머신에 더 이상의 자원을 할당하지 못하게 되는 서버 과부하 상황이 되면 이는 가상 머신의 결함 혹은 고장이 발생된다. 이와 같은 문제를 해결하기 위해 서버는 특정 가상 머신을 자원 풀 안의 다른 서버(타겟 서버)로 이주하여 과부하 상태인 서버(소스 서버)의 물리적 자원을 확보한다[3][4]. 가상 머신을 이주하는 방법에서 고려해야 할 요인은 이주 기준이 되는 계산 자원, 이주 시점과 응용프로그램의 서비스 가용성이다[3][4][5][6].

본 논문에서는 이주 횟수를 줄이기 위해 자원(CPU, 네트워크, 그리고 메모리) 사용률의 변화를 기반으로 동적 임계치를 정하고, 서버 과부하를 진단하여 이주할 가상 머신과 타겟 서버를 선정한다. 그리고 이주 시점을 결정하여 가상 머신을 이주 시킨다.

2. 관련 연구

서버 가상화는 기존의 물리적 서버에 가상 머신을 생성하여, 하나의 물리적 서버에서 여러 가상 머신이 통합되어 운영된다. 이에 기존 물리적 서버의 성능을 유지하기 위한 효율적인 서버 통합에 대한 연구가 시작되었다.

서버통합은 정적 통합(Static Consolidation)과 동적 통합(Dynamic Consolidation)이 있다. 정적 통합은 가상 머신이 종료될 때까지 동일한 서버에서 수행 할 수 있도록 통합 전에 가상 머신들의 사용량을 분석하여 통합하는 방식이다. 그러나 서버의 수용력 이하의 부하만 고려했기 때문에, 특정 가상 머신의 갑작스런 과부하로 서버의 자원 수용력이 한계를 넘는다. 이로 인해 서버 또한 과부하 상태가 되어 결국 작업 지연이나 고장이 발생한다.

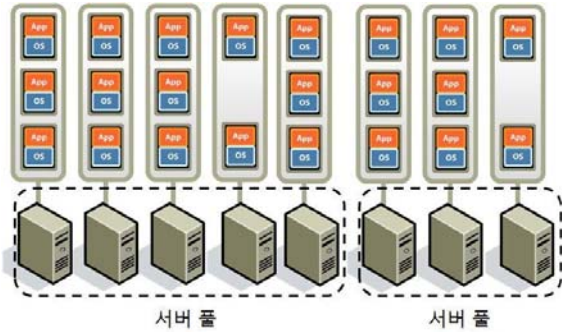
동적 통합은 가상 머신의 작업량이 서버의 자원 수용력을 초과하면, 소스 서버의 특정 가상 머신을 타겟 서버로 이주 시켜 서버의 과부하를 해소하는 방법이다. 동적 통합 방법 중의 하나로 Sandpiper가 있다. Sandpiper는 10초를 주기로 서버와 가상 머신의 자원(CPU, 네트워크, 메모리) 사용률을 수집하여 서버의 부하를 모니터링 한다. 그리고 모니터링된 자원을 이용하여 정의된 한계치를 넘는 서버 과부하가 예측되면, 타겟 서버로 가상 머신을 이주하여 과부하를 방지한다[4]. 이 방법은 가상 머신의 자원 사용률을 통합한 Volume값을 기준으로 이주될 서버를 선정하기 때문에, 서버의 자원 사용률을 비교하는 방법이 단순하다는 장점이 있다. 그러나 과부하를 결정하는 자원이 어떤 것인지 고려하지 않기 때문에, 타겟 서버의 수용 능력에 따라 가상 머신을 이주한 이후 타겟 서버의 특정 자원이 과부하가 되는 경우가 발생한다.

또한 서버의 부하를 모니터링하기 위해 CPU 자원 사용률을 고려한 연구로 학습 모듈을 사용한다[5]. 이는 서버의 표준편차가 가상 머신을 타겟 서버로 이주한 후의 값이 이주 전의 값보다 적다고 예상하면 가상 머신을 이주한다. 이 연구는 표준편차의 값을 줄이는 목적으로 서버의 부하 균형에 중점을 두고 있다. 이는 CPU의 사용률에만 임계치를 적용하였으므로 다른 자원들의 부하가 높은 응용프로그램 환경에서는 적합하지 않다. 따라서 본 논문에서는 CPU, 네트워크, 그리고 메모리의 자원 사용률을 기반으로 가상 머신과 타겟 서버를 결정하고 이주를 수행하는 알고리즘을 제안한다.

3. 동적 임계치 기반 가상머신 이주

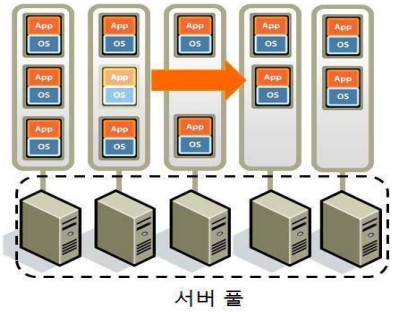
3.1 시스템 환경

데이터센터의 서버들은 특정 회사에 임대되고 그 회사의 작업을 수행한다. 임대된 서버들은 특정한 목적에 따라 <그림 1>처럼 가상화 자원 풀(이하 서버 풀)로 관리되고 서버 각각은 하나 또는 여러 가상 머신을 실행한다.



<그림 1> 데이터센터 내의 서버 풀

가상 머신의 응용프로그램 작업량이 증가하면서 서버의 자원 사용량도 증가한다. 이러한 상황은 동일한 서버에서 실행되는 모든 가상 머신의 작업에 영향을 준다. 특정 가상 머신의 작업량이 증가하면 서버는 유휴 자원을 그 가상 머신에 재 할당하여 작업을 처리한다. 그러나 서버의 유휴 자원이 없으면 가상 머신은 더 이상 서버의 자원을 재 할당 받지 못하게 된다. 이로 인해 서버 과부하 상황이 발생한다. 이것은 작업량이 많아진 가상 머신의 응용프로그램 가용성을 떨어뜨린다. 그리고 이때 다른 가상 머신의 작업량이 늘어나면 서버 자원이 재 할당되지 않기 때문에 모든 가상 머신에 영향을 준다. 이와 같이 서버 과부하는 서버 전체의 성능에 영향을 주어 서버와 가상 머신의 가용성이 낮아지고 응용프로그램 클라이언트의 요청을 처리하지 못하는 서비스 결함 상황이 된다.



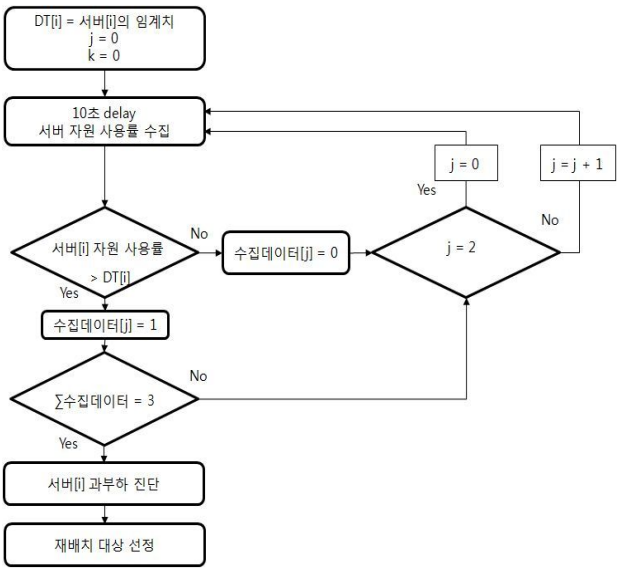
<그림 2> 가상 머신 이주

이러한 문제를 해결하기 위해 서버는 <그림 2>와 같이 특정한 가상 머신을 서버 풀에 있는 타겟 서버로 이주한다.

3.2 서버 과부하 진단 기법

서버 과부하를 방지하기 위해서 주기적으로 서버의 자원 사용률을 점검한다. 자원 사용률은 응용프로그램 서비스에 따라 높아진다. 이를 모두 서버 과부하로 진단하면 가상 머신 이주가 자주 발생한다. 이를 방지하기 위해서 임계치를 초과한 횟수를 기반으로 서버 과부하를 진단한다. 관련 연구에서는 10초를 주기로 자원 사용률을 수집하고 연속된 값의 3개가 임계치를 초과하면 서버 과부하로 진단한다. 이와 같은 방식은 MDT(Migration of Virtual Machine based Dynamic Threshold) 알고리즘에 적용하여 서버 과부하를 진단한다.

본 연구에서 서버 과부하를 진단하는 방법은 <그림 3>과 같이 10초를 주기로 자원 사용률을 수집하고 임계치를 초과할 경우 배열 변수에 1을 저장하고 그렇지 않을 경우 0을 저장한다. 배열 변수에 저장된 값의 합이 3이면 서버 과부하로 진단하고 이 서버를 소스 서버로 결정한다.



<그림 3> 서버 과부하 진단 순서도

3.3 동적 임계치 적용

서버의 자원 사용률은 가상 머신의 응용프로그램 작업에 따라 일정하게 유지되기도 하지만 증가하거나 예기치 않게 증가하는 경우가 있다. 자원 사용률이 증가하는 서버를 타겟 서버로 정의하면, 가상 머신을 이주 한 후 가까운 미래에 타겟 서버의 과부하가 판단되어 특정 가상 머신을 다른 서버로 이주해야 한다. 이런 경우 작업이 증가하는 서버의 임계치를 이전 보다 낮게 설정한다면, 이 서버는 타겟 서버로 선정되지 않는다. 반대로 작업이 감소하는 서버의 임계치를 이전 값보다 높게 설정 하고 이 서버를 타겟 서버로 선정한다면, 가상 머신이 이주된 이후에도 타겟 서버는 과부하로 판정되지 않는다. 이와 같은 변화하는 임계치(th_n)는 시간(t)의 흐름에 따라 자원 사용률의 변화량(Δu)을 기반으로 다음과 같이 구한다. 이 때 u는 CPU, 네트워크, 메모리 각각의 자원 사용률을 나타낸다.

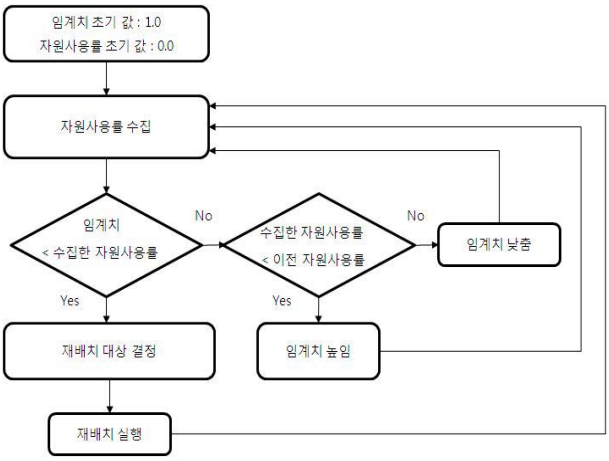
$$\Delta u = u_n - u_{n-1}$$

$$th_{n+1} = \begin{cases} th_n + (u_n - th_n)*\Delta u, & \Delta u \geq 0 \text{ 일 때} \\ th_n + (th_n - 1)*\Delta u, & \Delta u < 0 \text{ 일 때} \end{cases} \quad (1)$$

u_n: $t = n$일 때의 자원사용률, $0 \leq u_n \leq 1$
th_n: $t = n$일 때의 임계치, $0 \leq th_n \leq 1$
Δu: 자원사용률 변화량

식 (1)에서 자원 사용률이 서버와 가상 머신이 실행하기 전에는 사용되지 않기 때문에 초기 값(u_0)은 0.0으로 한다. 그리고 임계치(th_1)는 자원 사용률의 최댓값인 1.0으로 한다. $t = n$일 때 자원 사용률 u_n이 임계치 th_n보다 크면 서버 과부하로 진단하고 가상 머신을 이주한 다음 임계치 th_{n+1}을 th_n과 같게 한다.

이와 같은 수식은 <그림 4>와 같이 서버 사용률의 변화에 따라 임계치를 높이고 낮출 때 사용한다.



<그림 4> 동적 임계치 적용 순서도

3.4 가상 머신 이주 기법

가상 머신 이주는 물리서버에 있는 가상 머신들 중 조건에 맞는 하나의 가상 머신을 다른 물리서버로 이동시킨다. 이주 방법은 다음과 같은 순서에 따른다. 첫 번째, 이주가 필요한 가상 머신들의 소그룹을 결정한다. 두 번째, 타겟 서버를 결정한다. 세 번째, 첫 번째 단계에서 결정한 가상 머신들의 소그룹에서 두 번째 결정한 타겟 서버로 이주할 가상 머신을 결정한다. <그림 5> 동적 임계치 기반의 가상 머신 이주를 위한 알고리즘이다.

소스 서버의 자원 과부하를 해소하려면, 적어도 자원 사용률(u)이 임계치(th)를 초과한 수치 $(u - th)$ 만큼 자원 사용률을 줄여야 한다. 즉, 초과 수치 이상인 가상 머신 중에 하나를 이주 대상으로 결정하여 타겟 서버로 이주하면 소스 서버에서는 확보한 유휴자원을 가상 머신에 재할당하여 서버 과부하를 해소할 수 있다. 이와 같이 전체 가상 머신(VMlist) 중 임계치를 초과한 자원 사용률(VMlist[j].u[r])을 지닌 가상 머신의 집합(VMSet)을 결정한다. 이주 대상인 가상 머신은 타겟 서버를 결정한 후 VMSet의 가상 머신 중에서 결정한다. 이 때 이주 후에 타겟 서버의 과부하가 예측 되면, 다른 서버를 타겟 서버로 선정하여 이주 대상 가상 머신을 결정한다.

가상 머신을 이주 한 뒤에 타겟 서버의 과부하를 방지 하려면, 자원 사용률이 임계값에서 초과 수치를 뺀 값 보다 작은 서버를 타겟 서버로

결정해야 한다. 이는 임계치 초과 수치만큼의 사용률을 지닌 가상 머신을 타겟 서버로 이주할 때 임계치를 넘지 않게 하기 위함이다. MDT 알고리즘은 소스 서버의 과부하 자원(r)을 기준으로 타겟 서버의 자원 사용률(PMlist[j].u[r])이 계산된 범위$(u-(u-b))$내에 포함되면 타겟 서버 목록(PMSet)에 포함시킨다. 이 때 자원 사용률이 가장 적은 서버를 타겟 서버 후보로 선정하고 이주를 시도한다. 이를 위해 PMSet의 서버들을 오름차순으로 정렬한다.

가상 머신으로 결정된 타겟 서버(PMSet)에서 사용률이 가장 큰 자원(HighR(pm))은 가상 머신을 이주 한 후 과부하가 될 가능성이 있다. 그러므로 가상 머신 중 관련 자원(r)의 사용률이 가장 적은 가상 머신을 타겟 서버로 이주하여 과부하를 방지한다. MDT 알고리즘은 가상 머신을 결정하기 위해 r을 기준으로 가상머신을 오름차순으로 정렬하고 순서대로 타겟 서버에 이주를 시도한다. 가상 머신의 이주 시도는 가상 머신을 타겟 서버로 이주했을 때 타겟 서버의 자원 사용률(pm.u[k])이 임계치를 초과하는 서버 과부하 상태인지를 확인한다.

타겟 서버의 자원 사용률은 가상 머신의 자원 사용률(vm.u[k])의 합을 초기 가상 머신 수(pm_INIT_NUM)로 나눈 것이다. 이것은 vm.u[k]을 pm_INIT_NUM으로 나눈 값과 pm.u[k]의 합과 같다. 이렇게 구한 타겟 서버의 자원 사용률 중에 어느 하나라도 임계치를 초과하면 1을 반환해서 다른 가상 머신을 이주 시도하고 그렇지 않으면 0을 반환해서 이주를 수행한다.

```
입력: 서버 전체 목록 PMlist, 과부하 서버내의 가상 머신
      목록 VMlist, 과부하 원인 자원 r,
      서버의 자원 사용률 u, 임계치 th
출력: 가상 머신 이주 성공 0, 이주 실패 1
migrationVM(PMlist[], VMlist[], r, u, th) {
  n = count(VMlist[]);
  i = 0;
  for (j = 0; j < n; j++) {
    if (VMlist[j].u[r] > u - th) {
      VMSet[i] = VMlist[j];
      i++;
  } }
  n = count(PMlist[]);
  i = 0;
  for (j = 0; j < n; j++) {
    if (PMlist[j].u[r] < th - (u - th)) {
      PMSet[i] = PMlist[j];
    i++;
    } }
  PMSet = SortIncreasing(PMSet[], r);
  n = count(PMSet[]);
  m = count(VMSet[]);
  for (i = 0; i <n; i++) {
    pm = PMSet[i];
    r = HighR(pm);
    VMSet = SortIncreasing(VMSet[], r);
    for (j = 0; j <m; j++) {
      vm = VMSet[j];
      init_num = pm_INIT_NUM
      for (k = 0; k < 3; k++) {
        pm.u[k] = pm.u[k] + (vm.u[k] / init_num);
        if (pm.u[k] < THRESHOLD) {
          migrate(vm, pm);
          return (0);
        } } } }
  return (1);
}
```

<그림 5> MDT 알고리즘

4. 실험 및 성능평가

4.1 실험환경

실험 환경은 <표 1>과 같이 8개의 가상 머신을 세 개의 서버에 적용하여 구성한다. 자원의 사용률은 Linux Shell의 RANDOM 변수를 이용하여 24개의 랜덤 값을 가상 머신의 자원 사용률로 한다.

서버의 자원 사용률은 식 (2)와 같이 가상 머신 자원 사용률의 합을 초기 가상 머신의 수로 나눈 값으로 한다. 여기서 초기 가상 머신 수는 서버가 수용할 수 있는 최적의 가상 머신 수이다.

$$u = \frac{\sum vm.u[k]}{pm\_INIT\_NUM} \quad (2)$$

제안한 MDT 알고리즘에서 서버의 자원 사용률은 $t=n-1$, n, $n+1$의 시간으로 구분한다. 각

시간에 따라 RANDOM 함수로 구한 72개의 값을 가상 머신의 자원 사용률로 사용한다. 이렇게 구한 가상 머신의 사용률은 식 (2)에 적용하여 서버의 자원 사용률을 구한다. 서버의 자원 사용률은 시간의 흐름에 따라 변하고 $t=n-1$, n 시간의 사용률을 기반으로 $t=n+1$ 시간의 임계치를 결정한다. 서버 자원 사용률은 서버 과부하 상황을 만들기 위해 세 개의 서버 중 한 서버와 세 가지 자원 중 하나의 자원 사용률이 임계치를 초과하도록 한다.

<표 1> 실험 환경

서버	PM1	PM2	PM3
가상머신 수	3	3	2

이와 같은 방법이 500개의 실험 데이터 개수를 만들어 이를 이용하여 실험한다. 실험 평가 항목으로 이주 횟수를 측정하며, 임계치는 초기에 설정한 수치를 기준으로 한다.

4.2 성능평가

성능평가에서는 정적 임계치를 사용하는 방법인 Sandpiper와 동적 임계치를 사용하는 방법으로 Beloglazov의 알고리즘(ATA), 그리고 제안하는 MDT 알고리즘의 성능을 평가한다. 실험 데이터의 자원 사용률은 0.0부터 1.0까지의 RANDOM 변수 값으로 구한다.

단, 정적 임계치를 적용하는 방법은 동적 임계치를 적용하는 방법 보다 자원 사용률의 증가와 감소에 대해 대응하지 못한다. 따라서 정적 임계치 기반의 연구에 적합하도록 자원 사용률의 10%를 변동률로 추가하였다. <표 2>는 초기 임계치와 실험데이터 수에 따른 가상 머신 이주의 실험결과이다.

Sandpiper와 MDT 비교에서 초기 임계치가 0.8일 때의 실험결과를 보면, MDT 알고리즘의 재배치 횟수는 Sandpiper 보다 60% 이상 적었고, 0.9일 때는 45% 적었다. 따라서 초기 임계치가 작을 때, MDT 알고리즘의 재배치 정확도가 높고 성능이 우수함을 보였다.

또한 ATA와 MDT 비교에서 초기 임계치를 0.8로 했을 때, MDT 알고리즘의 재배치 횟수는 ATA 알고리즘 보다 평균 14% 적었고 초기 임계치를 0.9로 했을 때는 평균 16% 적었다. 초기 임계치가 클 때, MDT 알고리즘의 재배치 정확도가 높고 성능이 우수함을 보였다.

<표 2> 가상 머신 이주 결과

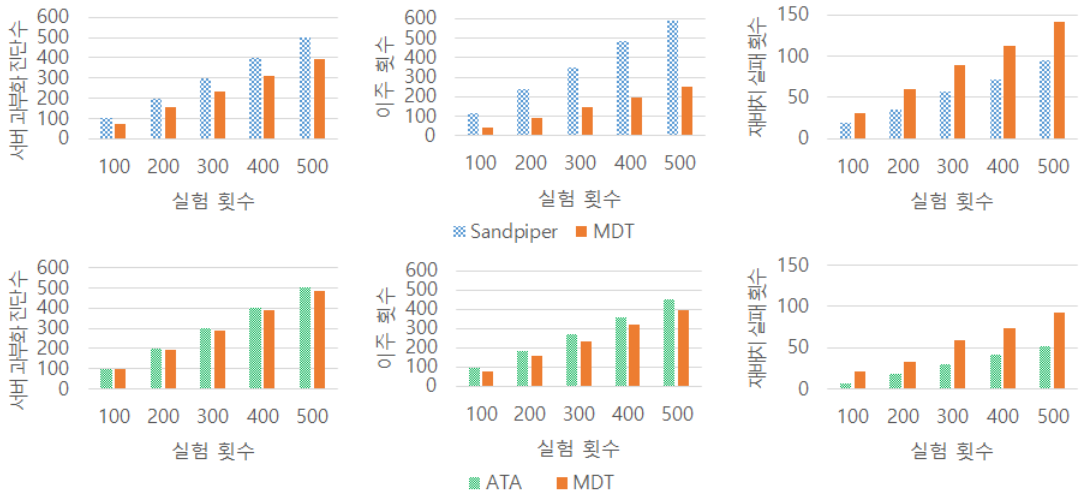
임계치	실험 데이터 수 (개)	서버 과부하 진단 수		이주 횟수		재배치 실패 횟수	
		Sandpiper	MDT	Sandpiper	MDT	Sandpiper	MDT
.8	100	100	73	119	42	19	31
	200	200	153	238	93	36	60
	300	300	236	352	147	57	89
	400	400	312	486	199	71	113
	500	500	391	592	250	95	141
.9	100	100	86	112	60	13	26
	200	200	169	231	124	20	45
	300	300	260	338	196	32	64
	400	400	344	442	263	45	81
	500	500	430	545	328	56	102

임계치	실험 데이터 수 (개)	서버 과부하 진단 수		이주 횟수		재배치 실패 횟수	
		ATA	MDT	ATA	MDT	ATA	MDT
.8	100	100	97	93	76	7	21
	200	200	193	182	160	18	33
	300	300	290	270	231	30	59
	400	400	390	358	317	42	73
	500	500	486	448	394	52	92
.9	100	100	79	83	69	17	10
	200	200	162	170	141	30	21
	300	300	245	255	211	45	34
	400	400	330	341	288	59	42
	500	500	416	428	361	72	55

또한 실험에서는 0.6과 0.7의 임계치로 하였으나, 0.8보다 재배치 횟수가 적어지므로 생략하였다. 다음 <그림 6>과 <그림 7>은 <표 2>를 그래픽화 한 것이다

5. 결론

본 논문에서 제안하는 알고리즘은 CPU, 네트워



<그림 6> 가상 머신 이주 결과 (임계치 0.8)

<그림 7> 가상 머신 이주 결과 (임계치 0.9)

크, 그리고 메모리 세 가지 자원의 사용률을 모두 고려하여 가상 머신의 재배치를 수행한다. 임계치는 자원 사용률의 변화를 기반으로 동적으로 변한다. 서버 과부하가 발생하면 임계치와 임계치를 초과한 사용률을 기준으로 가상 머신 집합과 타겟 서버를 결정한다. 그리고 재배치할 가상 머신은 타겟 서버의 자원 사용률을 기준으로 결정하고 타겟 서버로 재배치 시뮬레이션 한다.

시뮬레이션 결과에서 타겟 서버의 과부하가 예상되면 다른 서버를 타겟 서버로 결정하고 가상 머신을 재배치한다. 실험결과는 첫째, 정적 임계치를 적용한 관련 연구와 비교한 실험에서 MDT 알고리즘의 재배치 횟수는 모든 경우에 Sandpiper 알고리즘 보다 적었다. 둘째, 동적 임계치를 적용한 관련 연구와 비교한 실험에서 MDT 알고리즘의 재배치 횟수는 모든 경우에 ATA 알고리즘 보다 적었다.

결론적으로 제안 알고리즘은 초기 임계치 0.8과 0.9사이일 때, 기존 연구보다 향상된 성능을 보였다.

Sandpiper 및 ATA 알고리즘의 경우 가상 머신이 이주 된 후 타켓 서버의 과부하로 인해 다시 이주해야 하는 경우가 발생한다. 반면 MDT 알고리즘은 CPU, 네트워크, 메모리 사용률의 변화에 따라 자원 각각의 임계치를 변동하여 특정 자원의 사용률을 예측하므로 가상 머신의 이주 횟수가 적게 된다.

향후 과제는 실험 횟수를 늘려서 서버의 자원

사용률의 변화에 기반한 초기 임계치 설정에 관한 연구와 동적 임계치를 적용한 가상머신 재배치 방법에 관한 연구의 신뢰성을 높이고, 알고리즘의 시간 복잡도가 $O(n^3)$인 것을 보완하여 실제 서버환경에서 시뮬레이션 통하여 동적 임계치를 적용한 가상머신 재배치 알고리즘을 최적화하는 연구를 수행하고자 한다.

참 고 문 헌

[1] VMware Inc. (2012). 가상화(가상화 기술)의 기본 개념에 대해서, Available: http://www.vmware.com/kr/virtualization/virtualization-basics/virtualinfrastructure.html (last visit : 2014.12.23.)

[2] 양은지, 최현식, 한세영, & 박성용. (2010). Xen 환경에서 스케줄링 지연을 고려한 가상머신 우선순위 할당 기법. *정보과학회논문지: 시스템 및 이론, 37*(4), 246-255.

[3] Wood, T., Shenoy, P., Venkataramani, A., & Yousif, M. (2009). Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks, 53*(17), 2923-2938.

[4] Ferreto, T. C., Netto, M. A., Calheiros, R. N., & De Rose, C. A. (2011). Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems*, *27*(8), 1027-1034.

[5] Choi, H. W., Kwak, H., Sohn, A., & Chung, K. (2008, June). Autonomous learning for efficient resource utilization of dynamic vm migration. In*Proceedings of the 22nd annual international conference on Supercomputing*(pp. 185-194). ACM.

[6] Beloglazov, A., & Buyya, R. (2010, November). Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science* (p. 4). ACM.

[7] Voorsluys, W., Broberg, J., Venugopal, S., & Buyya, R. (2009). Cost of virtual machine live migration in clouds: A performance evaluation. In *Cloud Computing* (pp. 254-265). Springer Berlin Heidelberg.

최 호 근

1997 수원대학교 전자계산학과(이학사)
2013 한국방송통신대학교 컴퓨터과학과(이학석사)
관심분야: 서버 가상화
E-Mail: hgchoi70@gmail

박 지 수

2013 고려대학교 컴퓨터교육과(이학박사)
2013～현재 고려대학교 정보창의교육연구소 연구교수
관심분야: 분산 시스템, 모바일 클라우드 & 클라우드 컴퓨팅, e-Learning
E-Mail: bluejisu@korea.ac.kr

손 진 곤

1991 고려대학교 전산학전공(이학박사)
2013～현재 한국방송통신대학교 컴퓨터과학과 교수
1997～1998 State University of New York (Stony Brook) Visiting Professor
2000～현재 ISO/IEC JTC1/SC36 Korea Delegate
2010년 한국정보처리학회 부회장
2009～현재 이러닝학회 부회장
관심분야 : 컴퓨터통신망, 분산시스템, 그리드 컴퓨팅 e-Learning, 정보기술 표준화
E-Mail: jgshon@knou.ac.kr