가상화 환경에서 과부하 해소를 위한 가상 머신 재배치 알고리즘

최호근*, 박지수**, 한금주*, 손진곤*
*방송통신대학교 대학원 정보과학과
**고려대학교 대학원 컴퓨터교육과
hgchoi70@gmail.com

Relocation Algorithm of Virtual Machine for Alleviating Overloads in Virtualization Environment

HoGeun Choi*, JiSu Park**, Gum Ju Han*, Jin Gon Shon*
*Dept. of Computer Science, Graduate School, Korea National Open University
**Dept. of Computer Science Education, Korea University

요 약

가상화 환경은 한 대의 서버에 여러 개의 가상 머신을 생성하여 독립적인 운영체제를 실행시킨다. 이러한 서버는 가상 머신에 서버의 자원을 할당하여 작업을 처리하고, 작업의 증가로 인해 가상 머신의 자원이 부족하게 되면 서버의 자원을 재할당 한다. 그러나 서버 자원의 확장이 어려운 경우 과부하 상태가 되어 가상 머신의 자원 부족을 해결할 수 없다. 또한 가상 머신을 다른 서버에 재배치 시킴으로써 서버의 과부하 문제를 해결하고자 하였으나, 서버 선정 과정이 단순하여 유휴자원을 확보하기가 어렵다. 따라서 본 논문에서는 가상 머신을 재배치시키는 기법을 제안한다. 또한 기존 재배치 알고리즘과 제안한 알고리즘에 대한 성능평가를 한다.

1. 서론

서버 가상화의 등장으로 한 대의 서버는 여러 개의 가상 머신을 만들고 독립적인 운영체제를 실행할 수 있게 되었다. 가상 머신을 생성하기 위해 물리적인 서버들은 가상화 리소스 풀(pool)로 관리되고, 가상 머신에 작업을 할당함으로써 서버들을 운영한다[1]. 가상화 환경에서 제한된 서버 자원은 여러 가상 머신이 공유해서 사용하므로 가상 머신의 자원 사용률을 측정하여 서버의 자원을 할당하여야 한다[2]. 만약 특정 가상 머신의 작업이 증가하면, 서버의 유휴자원을 재할당해서 가상 머신의 자원을 늘려야 한다. 그러나 서버의 유휴 자원이 부족해서 더 이상 재할당을 하지 못한다면, 서버의 결함 혹은 고장으로 이어질 수 있다. 서버의 자원 재할당을 위한 방법으로 두 가지 연구가 있다. 첫 번째, 물리적인 서버의 자원을 늘려준다. 그러나 물리적인 자원 수용의 제약으로 자원을 늘리지 못하는 상황이 될 수 있다. 두 번째, 소스 서버의 특정 가상 머신을 서버 풀 내의 다른 타겟 서버로 재배치하여 소스 서버의 물리적 유휴자원을 확보하는 방법이 있다[3][4]. 그러나 두번째 방법은 타겟 서버에서 사용중인 자원의 사용률로 인해 가상 머신을 재배치 한 후에 타겟 서버의 과부하가 발생하는 문제점이 있다. 이런 문제점을 해결하기 위해서 과부화가 발생 하지 않는 서버를 타겟 서버로 선정하는 방법과 소스 서버와 타겟 서버의 가상 머신을 서로 교환하는 방법이 연구되었다[5][6]. 그러나 통합된 값의 자원 사용률을 기준으로 타겟 서버들을 선정하므로 가상 머신을 재배치 한 후에도 타겟 서버의 특정 자원의 과부화가 해소되지 않는 문제점을 가지고 있다.

본 논문에서는 이러한 문제점을 해결하기 위해 자원 사용률을 기준으로 가상 머신과 타겟 서버를 선정하는 방법을 제안한다. 그리고 제안한 방법은 실험을 통해 기존연구와 제안하는 알고리즘의 성능의 차이를 분석하고 평가한다.

2. 관련연구

2.1 Sandpiper

Sandpiper 는 주기적으로 서버와 가상 머신의 자원 사용률을 수집하여 서버 과부하를 해소하기 위한 가상 머신 재배치 알고리즘을 제시한다[5]. 서버의 부하는 CPU, 네트워크, 메모리의 사용률을 통합한 Volume(이하 V)값으로 식 (1)을 적용하여 자원 사용률의 통합 값인 V 값을 계산한다.

Sandpiper 는 미리 정의된 임계값(Threshold)보다 수집된 서버의 V 값이 초과한 서버를 소스 서버로 선정한다[3][5]. 서버의 과부화를 해소하기 위해 소스 서버에서 V 값이 가장 큰 가상 머신을 부하가 가장

적은 타겟 서버로 재배치한다. 만약 재배치 후 타겟 서버의 V 값이 임계값을 초과한다면, 타겟 서버에서 부하가 가장 적은 가상 머신을 소스 서버로 재배치한다.

$$V = \frac{1}{1-c} * \frac{1}{1-n} * \frac{1}{1-m} \quad (1)$$

여기서 ,

c : CPU 사용률 , $0 \le c < 1$

n : $Network$ 사용률 , $0 \le n < 1$

m : $Memory$ 사용률 , $0 \le m < 1$

Sandpiper 는 V 값을 기준으로 재배치 대상을 선정하기 때문에, 서버의 자원 사용률을 비교하는 방법이 단순한 장점이 있다. 그러나 과부하를 결정하는 자원이 어떤 것인지 고려하지 않기 때문에, 타겟 서버의 수용 능력에 따라 가상 머신 재배치 이후 타겟 서버의 특정 자원이 과부화가 되는 경우가 된다.

2.2 SCOA

Sandpiper 의 문제점을 해결하기 위한 알고리즘으로 가상화 기반 서버통합 최적화 알고리즘(server consolidation optimizing algorithm 이하 SCOA)이 있다[6]. SCOA 는 모든 서버의 CPU, 네트워크, 메모리 자원의 사용률을 기반으로 3 차원 좌표에 점으로 표시하고 원점과의 거리가 가장 긴 점을 소스 서버로 선정한다. 타겟 서버는 소스 서버의 좌표점과 가장 먼 거리에 있는 서버를 선정한다. 서버 사이의 거리는 식 (2)로 구한다.

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

여기서,

D: 서버를 나타내는 좌표점간의거리

소스 서버에서 과부화 자원을 기준으로 재배치를 할 대상의 가상 머신을 선정하고, 가상 머신 교환이 예상되면 타겟 서버의 과부화 자원을 고려하여서 교환한다.

과부하 자원을 기준으로 재배치를 할 대상의 가상 머신을 선정하여 Sandpiper 의 문제점을 해결한다. 그러나 Sandpiper 와 같이 3 가지 자원 사용률의 통합된 값을 기준으로 타겟 서버를 선정하기 때문에 여전히 특정 자원의 과부하를 유발하는 문제점을 가지고 있다.

본 논문에서 제안하는 알고리즘은 타겟 서버 선정 방법에서 개별적인 자원의 사용률을 기준으로 하여 기존연구의 문제점을 개선하고자 한다.

3. 가상 머신 재배치 알고리즘

제안한 알고리즘은 소스 서버에서 임계값을 초과한 자원을 기준으로 재배치를 할 가상 머신과 타겟 서버를 선별한다. 가상 머신은 타겟 서버에서 사용률이 높은 자원을 기준으로 선정한다. 가상 머신이 재배치된 후에 타겟 서버의 과부하가 예측된다면, 타겟 서버를 다른 서버로 변경해서 가상 머신의 재배치를 시도한다.

3.1 가상 머신 선별

소스 서버의 자원 과부하를 해소하려면, 적어도 임계값의 초과 수치(Diff) 만큼 자원 사용률이 줄어들어야 한다. 즉, 초과 수치 이상인 가상 머신 중에 하나를 타겟 서버로 재배치하고 확보한 소스 서버의 유휴자원을 재할당하여 과부하를 해소할 수 있다. 전체 가상 머신(VMlist) 중 자원 사용률이 Diff 값 이상인 가상 머신(FilterVMlist)을 선별한다. FilterVMlist 의 가상 머신 중 재배치 대상 가상 머신을 선정해야 하는데, 이 때 재배치 후에 타겟 서버의 과부하가 예측 된다면, 다른 서버를 타겟 서버로 선정하여 재배치 대상 가상 머신을 선정한다.

알고리즘 1 가상 머신 선별 알고리즘

```
1  Diff = OverloadResourceUtil - THRESHOLD;
2  j = 0;
3  for i = 0..n-1 in VMlist
4    if VMlist[i].OverloadResourceUtil > Diff
5    then
6      j = j++;
7      FilterVMlist[j] = VMlist[i];
8    end if;
9  end for;
```

3.2 타겟 서버 선별

가상 머신을 재배치 한 후 타겟 서버의 과부하를 방지 하려면, 자원 사용률이 임계값에서 Diff 를 뺀 값 보다 작은 서버를 타겟 서버로 선정해야 한다. 소스 서버의 과부하 자원을 기준으로 서버(PMlist)의 자원 사용률이 임계값에서 Diff 값을 뺀 수치 미만의 서버들을 타겟 서버 목록(PMtemplist)에 포함시킨다. 이 때 자원 사용률이 가장 적은 서버를 타겟 서버 후보로 선정하고 재배치를 시도한다. 이렇게 하기 위해서 PMtemplist 를 오름차순으로 정렬한다.

알고리즘 2 타겟 서버 선별 알고리즘

```
1  j = 0;
2  for i = 0..m-1 in PMlist
3    if PMlist[i].OverloadResourceUtil < THRESHOLD - Diff
4      then
5      j = j++;
6      PMtemplist[j] = PMlist[i];
7    end if;
8  end for;
9  FilterPMlist = sortIncreasing(PMtemplist);
```

3.3 가상 머신 재배치

선정된 타겟 서버에서 사용률이 가장 큰 자원은 가상 머신을 재배치 한 후 과부화가 될 가능성이 있다. 가상 머신 중 관련 자원의 사용률이 가장 적은 가상 머신을 타겟 서버로 재배치 한다면, 과부하를 방지할 수 있다. 타겟 서버에서 사용률이 가장 큰 자원(HighestloadResource)을 과부하 예상 자원(P)으로 정의 하고 가상 머신 중 P 자원의 사용률이 가장 적

은 것을 재배치 대상으로 선정한다.

알고리즘 3 가상 머신 재배치 알고리즘

```
1   for i = 0..M-1 in FilterPMlist
2      P = HighestloadResource(FilterPMlist[i]);
3      VM = LowestloadResourceUtilVM(FilterVMlist.P);
4      Migration(VM, FilterPMlist[i]);
5      if FilterPMlist[i].HighestResourceUtil < THREHSOLD
6      then
7            break;
8      end if;
9   end for;
```

Sandpiper 재배치 대상 가상 머신과 타겟 서버를 선정하기 위해서, 최악의 경우 가상 머신과 서버 전부를 비교해야 한다. 제안한 가상 머신 재배치 알고리즘은 과부하 자원의 사용률을 기준으로 가상 머신과 타겟 서버를 재배치 대상 후보로 선별하는 것으로써 재배치 가능성을 확인하기 위한 비교대상을 줄인다. 그리고 타겟 서버의 자원 사용률을 기준으로 가상 머신을 선정 하므로, 가상 머신을 재배치 한 후 타겟 서버의 자원 과부하를 피할 수 있다.

4. 성능 평가

제안하는 알고리즘은 Sandpiper 의 자원 사용률 감지 방법을 기반으로 서버와 가상 머신 각각의 자원 사용률을 주기적으로 검출한다[5]. 검출된 서버의 자원 중 특정 자원의 사용률에 대해서 연속된 5 개의 값을 확인하고, 이 때 3 개 이상이 임계값을 초과했다면 서버 자원의 과부하 상태로 인지하고, 이 서버를 소스 서버로 선정한다[5].

<표 1>과 같이 제안한 가상 머신 재배치 알고리즘의 성능평가를 위하여 17 개의 가상 머신을 5 개의 서버에 배치하고, 가상 머신의 자원사용률을 계산하여 서버의 자원사용률을 나타낸다.

<표 1> 실험 환경

서버	PM1	PM2	PM3	PM4	PM5
가상 머신 수	4 개	4 개	3 개	3 개	3 개

서버의 자원사용률이 임계값(70%)을 초과하는 과부하 상태인지 점검하고, 가상 머신 재배치를 실시한다. 가상 머신의 CPU, 네트워크, 메모리 사용률은 0 에서 1 이하의 난수 값을 이용하고, 소수점 2 자리까지 표시한다. 각각의 서버 자원 사용률은 가상 머신 자원 사용률의 합을 가상 머신의 수로 나눈 값으로 정의한다. 여기서 가상 머신의 수는 재배치 및 교환이 발생할 때 재배치된 가상 머신의 수와 관계없이, 초기에 배치된 가상 머신의 수로 지정한다. 2 개의 서버에는 4 개씩 가상 머신을 배치하고, 3 개의 서버에는 3 개씩 가상 머신을 배치한다. 자원 과부하를 나타내기 위한 방법으로는 5 개 서버의 자원 중 하나의 자원 사용률만 임계값을 초과하도록 난수를 만든다. 이와 동일한 방법으로 30 개의 서로 다른 실험 데이터를 만들고, Sandpiper, SCOA 와 제안 알고리즘을 적용하여 가상 머신 재배치를 실시한다. 각각의 실험 데이터는 서버의 순간 자원사용률을 나타내므로, 가상 머신 재배치를 1 회만 수행한다. 제안 알고리즘에는 가상 머신 교환이 없기 때문에 가상 머신 교환이 필요할 경우에는 Sandpiper 와 SCOA 알고리즘에서만 적용한다.

실험 데이터에 가상 머신의 재배치 이후 타겟 서버의 과부하가 예상되면 다른 서버를 타겟 서버로 선정하고 재배치 가능성을 측정한다. 이때 서버를 찾는 횟수를 비교횟수로 정의한다. 만약 타겟 서버를 선정하지 못할 경우 소스 서버와 타겟 서버의 가상 머신 교환을 실행한다. 재배치 후 임계값을 초과한 서버의 수를 측정한다. <표 2>은 30 개의 실험 데이터에 3 가

<표 2> 가상머신 재배치 실험 결과

테스트 데이터	재배치 결과(S:성공, F:실패)			재배치 후 임계값 초과 PM 수			비교횟수		
	Sandpiper	SCOA	제안알고리즘	Sandpiper	SCOA	제안알고리즘	Sandpiper	SCOA	제안알고리즘
1	S	F	S	0	제외	0	1	2	2
2	S	F	F	1	제외	제외	1	2	0
3	S	S(교환)	S	0	0	0	1	2	2
4	S	S(교환)	F	2	0	제외	1	2	2
5	S	S(교환)	S	0	0	0	1	2	1
6	S	S	S	0	0	0	1	1	1
7	S	S	S	0	0	0	1	1	1
8	S	F	S	1	제외	0	2	2	2
9	S	F	F	1	제외	제외	1	2	2
10	S	S	S	0	0	0	1	1	3
:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
21	S	S	S	0	0	0	1	1	1
22	S	S	S	0	0	0	1	1	1
23	S	F	F	2	제외	제외	1	2	1
24	S	S(교환)	S	2	0	0	3	2	1
25	S	S(교환)	S	1	0	0	1	2	2
26	S	S(교환)	S	1	0	0	1	2	1
27	S	F	F	1	제외	제외	1	2	3
28	S	S(교환)	S	1	0	0	2	2	1
29	S	F	F	1	제외	제외	1	2	1
30	S	S	S	0	0	0	1	1	1

지 알고리즘을 적용한 결과이다.

<표 3> 가상 머신 재배치 결과 통계치

	Sandpiper	SCOA	제안알고리즘
재배치 성공률	100.00%	56.67%	76.67%
교환 발생률	0.00%	47.06%	0.00%
임계값 초과 PM 잔여률	73.33%	0.00%	0.00%
비교횟수	1.2 회	1.7 회	1.3 회

<표 3>는 실험 결과 값을 통계로 나타낸 것이다. Sandpiper 경우 30 개의 실험 데이터 모두에 대해서 재배치를 성공했다. 그러나 남아있는 서버 중 임계값을 초과한 서버는 73.33% 가 되었다. SCOA 는 56.67% 성공률 나타냈고 가상 머신 교환은 46.06%를 나타냈다. 재배치 이후 임계값을 초과한 서버는 없었다. 제안 알고리즘의 재배치 성공률은 76.67%이고 재배치 이후 임계값을 초과한 서버는 없었다.

제안 알고리즘은 Sandpiper 보다 재배치 성공률은 낮았으나 SCOA 보다 높았고 재배치 이후에 서버 과부하를 모두 해소하는 결과를 보였다.

5. 결론

제안 알고리즘은 3 가지 자원의 사용률을 모두 비교하여 재배치 대상 범위에 포함되는 가상 머신과 타겟 서버를 선별한다. 그리고 타겟 서버의 자원 사용률을 참고하여 재배치 대상 가상 머신을 선정하므로 재배치 이후 타겟 서버가 과부화 상태가 되는 것을 방지한다. 과부화 상태인 자원을 식별하여 가상 머신을 재배치 하므로 재할당이 필요한 자원을 정확히 선정하여 유휴 자원을 확보한다.

특정 조건에 포함되는 서버들만을 비교하므로 재배치 대상을 선정하기 위한 비교횟수가 줄어들 것으로 예상했으나 실험 결과는 Sandpiper 가 가장 적은 것으로 나타났다. 향후 과제는 실험 횟수를 늘려서 보다 신뢰성이 있는 실험 결과를 표출하고 실제 서버환경에서 시뮬레이션 하여 알고리즘을 최적화 하는 연구가 필요하다.

참 고 문 헌

[1] http://www.vmware.com/kr/virtualization/virtualization-basics/virtual-infrastructure.html (최종방문일 : 2012.10. 24)

[2] 양은지, 최현식, 한세영, 박성용. “Xen 환경에서 스케줄링 지연을 고려한 가상머신 우선순위 할당 기법”, 정보과학회 논문지 시스템 및 이론, 제 37 권 제 4 호, 2010.

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

[4] Anton Beloglazov. "Energy-aware resource allocation heuristics for efficient management of data center for Cloud computing", Future Generation Computer Systems 28, pp.755-768, 2012.

[5] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. “Sandpiper:Black-box and gray-box resource management for virtual machines”, Computer Networks Volume 53, Issue 17, pp.2923-2938, 2009.

[6] 사성일, 하창수, 박찬익, “가상화 환경에서 부하균형을 위한 가상 머신 동적 재배치,” 정보과학회 논문지 시스템 및 이론, 제 35 권 제 12 호, 2008.