논문 15-40-01-10 The Journal of Korean Institute of Communications and Information Sciences '15-01 Vol.40 No.01
http://dx.doi.org/10.7840/kics.2015.40.1.61

클라우드 컴퓨팅 환경에서 사용패턴을 고려한 에너지 효율적인 가상머신 배치 기법

김 민 회◆, 박 민 호°

Energy-Aware Virtual Machine Deployment Method for Cloud Computing

Minhoe Kim◆, Minho Park°

요 약

가상 머신 기술을 통하여 가상머신은 적은 숫자의 물리적 서버에 통합 될 수 있다. 최대 전력 소비의 60% 이상이 idle한 물리적 서버에서 낭비되는데, 에너지 소모를 줄이는 가장 좋은 방법 중 하나는 컴퓨팅 자원에 대한 제약사항을 가진 배낭 문제 알고리즘을 사용하여 물리적 서버의 개수를 최소화 하는 것이다. 그러나 최적의 Consolidation을 통하여 물리적 서버의 개수를 최소화 하는 컴퓨팅 자원 기반의 가상 머신 배치 기법이 항상 효율적인 에너지 사용을 보장하지는 않는다. 본 논문에서는 동작 시간을 고려하여 에너지 사용을 최소화 할 수 있는 가상머신 Consolidation 알고리즘을 제안한다. 제안하는 알고리즘은 각 가상머신의 서비스 시간을 고려하여, 에너지 소비를 최소화하도록 물리적 서버에 가상머신을 배치한다. 다양한 시뮬레이션 결과를 통해서 제안하는 알고리즘이 30%이상의 에너지 절약 효과를 얻는 것을 확인할 수 있다.

Key Words : Cloud computing, Virtualization, Consolidation, Energy Saving

ABSTRACT

Through Virtual Machine technology(VM), VMs can be packed into much fewer number of physical servers than that of VMs. Since even an idle physical server wastes more than 60% of max power consumption, it has been considered as one of energy saving technologies to minimize the number of physical servers by using the knapsack problem solution based on the computing resources. However, this paper shows that this tightly packed consolidation may not achieve the efficient energy saving. Instead, a service pattern-based VM consolidation algorithm is proposed. The proposed algorithm takes the service time of each VM into account, and consolidates VMs to physical servers in the way to minimize energy consumption. The comprehensive simulation results show that the proposed algorithm gains more than 30% power saving.

Ⅰ. 서 론

데이터센터 아웃소싱 사업 규모가 큰 기업들과 금융, 의료, 유통 및 공공기관 등의 3차 산업에서의 데이터센터 비중은 기업들의 성장과 함께 커지고 있다. 데이터센터가 커지면서 데이터센터 에너지 사용의 경제적인 부분도 매우 커지고 있다. 예를 들어 Citigroup의 데이터센터의 경우, 인프라의 공간비중 1%에 불과하

※ 본 논문은 2014년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2013R1A1A1076105)

◆ First Author : Soongsil University Department of Telecommunication Engineering, minhoe@ssu.ac.kr, 학생회원
° Corresponding Author : Soongsil University Department of Telecommunication Engineering, mhp@ssu.ac.kr, 정회원

논문번호 : KICS2014-08-321, Received August 26, 2014; Revised November 4, 2014; Accepted December 30, 2014

지만 전력 사용량은 약 25%를 차지한다[1]. 따라서 전력사용비용이 매우 크기 때문에 에너지 효율적인 관리가 매우 중요하게 받아들여지고 있다. 효율적인 관리를 위해 데이터 센터 내 에너지 낭비 부분을 볼 필요가 있다. 데이터 센터 내 에너지 낭비는 idle한 시스템에서 발생된다. 전형적인 데이터 센터 안에서 고객의 서버 이용률은 30% 이하다. 그러나 이 중 idle한 서버로 인해 발생되는 전력 소모량은 최대 전력 소모량의 60%를 차지한다[2]. 그러므로 데이터센터 내 서버의 이용률을 늘리고 전력 소모량을 줄이는 가장 좋은 방법은 물리적 서버의 개수를 최소화 하는 것이다[3]. 물리적 서버의 개수를 줄이기 위한 방법으로 가상 머신 Consolidation이 있다. 가상 머신 Consolidation은 가상 머신 서버가 물리적 서버의 하드웨어 자원을 나눠서 사용하는 방법이다. 대부분 서버의 하드웨어 자원은 idle한 상태이다. 서버 하드웨어 자원 중 CPU의 평균 이용률을 조사한 연구조사에 따르면 서버 평균 CPU 사용률은 10~50% 정도로 낮은 값을 보인다[4]. 그러므로 각각의 하드웨어 자원의 사용률을 고려하여 물리적 서버 안에 다수의 가상 머신을 배치하면 물리적 서버의 개수를 줄일 수 있다. 이러한 가상 머신 Consolidation을 더 효과적으로 하기 위한 방법 중 하나가 배낭 문제 알고리즘을 이용하는 것이다. 배낭 문제 알고리즘은 무게가 정해진 가방에 최대가치를 얻을 수 있도록, 짐을 담는 방법을 찾는 고전적인 알고리즘 문제이다[5,6]. 배낭 문제 알고리즘을 이용하여 가상 머신을 물리적 서버에 Consolidation하면 물리적 서버의 개수는 최소가 된다. 그러나 배낭 문제를 이용한 가상 머신 Consolidation 방법에도 문제점이 있다. 물리적 서버의 개수는 최소가 되지만, 에너지 사용량은 최소가 아닌 경우가 있다. 간단한 예를 다음 2장에서 설명한다.

본 논문에서는 가상 머신의 사용패턴을 고려해서 에너지 효율적으로 배치 할 수 있는 알고리즘을 제안한다. 가상 머신의 사용패턴이 유사한 것들을 우선적으로 배치하여, 물리적 서버의 동작 상태를 최소화하여 전력사용을 최소화 한다. 시뮬레이션 결과를 통해서 30% 이상의 전력감소 효과가 보임을 확인하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 본 논문과 연관된 관련 연구에 대해서 소개한다. 3장에서는 배낭 문제 알고리즘에 대한 간략한 설명과 배낭 문제 알고리즘을 이용한 가상 머신 배치 예제를 설명하고 본 논문이 제안하는 알고리즘을 보인다. 4장에서는 제안하는 알고리즘의 성능을 검증하기 위해, C++를 통해 기존 배낭 문제 알고리즘과 제안하는 알고리즘의 프로그램을 구현한 시뮬레이션 결과를 비교하고, 5장에서 본 논문의 결론을 맺는다.

Ⅱ. 관련 연구

클라우드 컴퓨팅에서 얻는 이익을 최대화하기 위한 가상 환경 구성에 대해 가상 머신 배치의 다양한 연구[7-10]가 있었다. [8]은 클라우드 물리 용량을 줄이기 위해 동적 마이그레이션과 consolidation 알고리즘을 제안했다. [9]는 작업 부하의 모니터링 검사를 통해 마이그레이션을 자동화하는 시스템을 제안했다. [10]는 가상머신 마이그레이션으로 인해 향상되는 값들을 측정하고 가상머신 마이그레이션의 분석 모델을 제안했다. 최근에는 본 논문의 접근 방법과 유사하게 동작시간을 고려한 가상 머신 Consolidation 기법을 제안한 PAC[11]이 제안되었다. PAC은 가상 머신들의 signature 패턴을 기반으로 가상 머신을 호스트에 배치하는 방식이다. 하지만 PAC은 클라우드 시스템 안에서, 주기적인 어플리케이션 Consolidation을 통해 호스트들의 로드밸런싱을 돕고 오버러드 현상을 완화하는 효과를 준다. 본 논문은 가상 머신의 사용시간을 우선 고려하여 전력사용량을 최소화하는 가상 머신 Consolidation을 제안한다.

Ⅲ. Energy-Aware Consolidation

3.1 Knapsack Problem

배낭(Knapsack) 문제는 n개의 물건과 각 물건 i의 무게 wi와 가치 vi가 주어지고, 배낭의 용량이 C일 때, 배낭에 담을 수 있는 물건의 최대 가치를 찾는 문제이다.[1]

배낭 문제는 제한적인 입력에 대한 문제를 동적 계

Algorithm1. Knapsack problem algorithm

```
      Output : K[n,C]
1:    for i= 0 to n    K[i, 0]= 0
2:    for w= 0 to C    K[0, w]= 0

3:    for i= 1 to n {
4:       for w= 1 to C{
5:          if(wi > w)
6:             K[i, w]= K[i-1, w]
7:          else
8:             K[i, w]= max{K[i-1,w], K[i-1,w-wi]+vi}
9:          }
10:   }
11:   return K[n, C]
```

획 알고리즘으로 해결한다. 배낭 문제의 주어진 4가지 조건은 물건, 물건의 무게,

물건의 가치, 배낭의 용량이다. 배낭이 비어 있는 상태에서 물건을 하나씩 배낭에 담는 것과 담지 않는 것을 현재 배낭에 들어 있는 물건의 가치의 합에 근거하여 결정한다. 만약 담는 물건이 배낭 용량을 초과하면 담을 수 없다.

K[i, w] 는 물건 1~i까지만 고려하고, 임시 배낭 용량이 w일 때의 최대 가치를 구한 값이다. i= 1, 2, ⋯, n이고, w= 1, 2, 3, ⋯, C이다. 무게가 wi인 물건 i를 배낭에 담을 때, 임시 배낭 용량 w의 크기와 비교하여 배낭에 담을 수 있는지 고려한다. 물건 i를 배낭에 담지 않는 경우, K[i, w] = K[i-1, w]가 된다. 물건 i를 배낭에 담는 경우에는, 배낭에 담지 않는 경우인 K[i-1, w]와 배낭에 물건을 담은 경우인 K[i-1, w-wi]의 2가지 경우 중에서 큰 값이 K[i, w]가 된다. 물건 i의 개수와 임시 배낭 무게 w를 알고리즘에 따라 증가하면 최적 해는 K[n, C]가 나온다.

3.2 Naive Solution:배낭 문제를 이용한 가상머신 Consolidation

데이터 센터의 물리적 서버의 개수가 줄어들면 그만큼의 소비전력의 양은 적어진다. 배낭 문제를 이용하면 최소 물리적 서버의 개수를 얻을 수 있다.

데이터센터 내에 물리적 서버 수가 최소면 사용되어지는 에너지의 전력량도 최소라 가정할 수 있다. 가상 머신 Consolidation문제를 배낭 문제 알고리즘에 맞게 설정하면 다음의 표 1로 표현된다. 물리적 서버에 배치될 가상 머신이 i개가 있다고 가정한다. 각 i번째 가상 머신의 cpu 용량과 메모리 용량은 cpui, memi가 된다. vi는 i번째 가상 머신의 가치 값으로, 용량사용 측면에서 해당 가상 머신이 물리적 서버에서 얼마나 차지하고 있는지 표시하기 위해, cpui와 memi의 곱인 vi = cpui * memi으로 나타낸다. 아마존의 경우 고객이 사용하는 업로드 대역폭, 스토리지, 기타 필요로 하는 하드웨어 용량에 맞추어 비용을 요구한다. 그러므로 서버가 요구하는 하드웨어 자원량이 커지면 서버의 가치가 올라가는 사실에 근거하여 vi를 정한다.

표 1. 가상 머신 요소
Table 1. Virtual Machine Consolidation element

Notation	Description
i	The i'th virtual machine
cpui	Cpu capacity for the i'th virtual machine
memi	Memory capacity for the i'th virtual machine
vi	Value for the i'th virtual machine

알고리즘 2는 기존 배낭 문제 알고리즘을 이용하여 가상 머신의 물리적 서버의 배치에 맞게 내용을 수정하였다. i는 각 i번째 가상 머신을 의미하고, 기존 배낭 문제 알고리즘에서 물건의 무게인 w는 가상 머신의 자원인 cpu와 mem으로 대체한다. cpu는 가상 머신의 CPU 용량을 의미하고 mem은 메모리 용량을 나타낸다. i번째 가상 머신의 cpu와 mem 중 하나라도 서버의 용량을 초과하면, i번째 가상 머신은 서버에 배치될 수 없다. 그러므로 Line 7에 AND문을 이용해 cpu와 mem의 두 가지 조건을 모두 만족시켜야 한다. K[i, cpu, mem] 는 가상 머신을 1~i까지만 고려하고, 임시 CPU 사용량이 cpu이고, 임시 메모리 사용량이 mem일 때, 최대 가치를 구한 값이다. i= 1, 2, ⋯, n, cpu= 1, 2, 3, ⋯, C이고 mem= 1, ,2, 3, ⋯, M이다. 따라서 배낭 문제 알고리즘에 의해 알고리즘2의 최적해는 K[n, C, M]이다.

간단한 예제를 위해 가상 머신을 넣을 각 물리적 서버의 cpui(하드웨어 CPU자원의 총 용량)과 memi (하드웨어 메모리 자원의 총 용량)를 각각 600으로 값을 정하고 위 알고리즘 2를 진행한다. 표 2는 알고리즘 테스트를 위해 각 하드웨어 자원 요구량이 다른 14개의 가상 머신들을 임의로 만든 것이다. 표 2의 자원 요구량은 상대적인 크기가 중요한 요소이므로, 단위를

Algorithm2. VM deploy using the knapsack problem

```
      Output : K[n,C,M]
1:    for i= 0 to n     K[i, 0]= 0
2:    for cpu= 0 to C     K[0, cpu]= 0
3:    for mem= 0 to M     K[0, mem]= 0

4:    for i= 1 to n {
5:     for cpu= 1 to C{
6:       for mem= 1 to M{
7:         if(cpui > cpu && memi > mem)
8:           K[i, cpu, mem]= K[i-1, cpu, mem]
9:         else
10:          K[i, cpu, mem]= max{K[i-1, cpu,
11:   mem], K[i-1, cpu-cpui, mem-memi]+vi}
        }
      }
     }
12:   return K[n, C, M]
```

표 2. 가상 머신의 자원요구량
Table 2. Resource requirements of the Virtual Machine

i	cpu	memory
1	200	200
2	200	600
3	200	200
4	200	250
5	200	200
6	200	200
7	250	250
8	600	100
9	600	500
10	200	600
11	250	300
12	600	150
13	350	600
14	400	500

사용하지 않고 숫자로만 나타낸다.

그림 1의 사각형의 가로와 세로의 길이는 각각 cpu/mem의 용량을 의미한다. 그림 2는 알고리즘을 통해 얻은 Consolidation 결과를 도식화한 것이다. 하드웨어 자원 요구량에 따른 배낭 문제 알고리즘으로 위 가상 머신 14개를 물리적 서버 안에 배정하면 물리적 서버 4개 안에 14개의 가상 머신을 모두 배치할 수 있다.

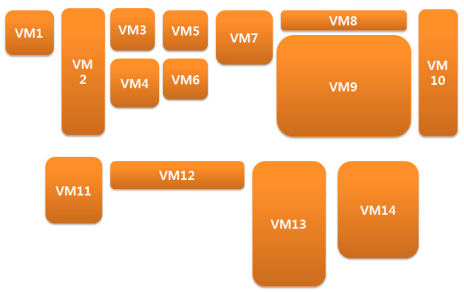

그림 1. VM_i의 도식화
Fig. 1. Diagram of VM_i

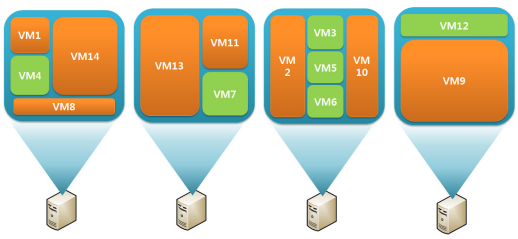

그림 2. Knapsack을 이용한 가상 머신 Consolidation
Fig. 2. Virtual Machine Consolidation with Knapsack

3.3 동기

그림 2의 배낭 문제를 이용한 가상 머신 Consolidation에는 한 가지 문제점이 있다. 물리적 서버의 개수는 최소화 되었지만, 에너지 절약 관점에서 이 방법으로 해결한 배치방법이 최적이 아닌 경우가 있다. 각 가상 머신이 하루 동안 동작하는 시간, 즉 동작 시간을 고려해 설계해보면, 배낭 문제를 이용한 가상 머신 Consolidation보다 에너지 소모가 더 적은 경우가 있다.

Consolidation에 앞서, 서버의 동작 패턴에 대해서 본다. 그림 3은 클라우드 서버의 동작패턴의 예를 보기 위해, Amazon EC2 cloud에서 호스트 서버의 CPU 사용률을 조사한 결과이다[12]. CPU 사용률을 보고, 해당 서버의 사용자 이용률이 시간에 따라 어떻게 변하는지 알 수 있다. 한 주간 측정된 이용률 수치에 관한 그래프를 보면, 7일간 한 서버 내 서버 이용률은 7~14시에 높은 값을 나타내고 나머지 시간대에는 낮은 값을 나타낸다. 이처럼, 서버의 이용률은 특정한 패턴을 보이며 이용률은 용도에 따라 서로 다른 시간대에 나타날 것이다. 이용률이 저조한 시간대에 서버를 절전 모드로 변환하거나 종료할 수 있다.

표 3은 사용패턴에 따른 가상 머신 Consolidation의 간단한 검증을 하기 위해, i번째 가상 머신의 사용시간인 Ti값을 표 2에서 추가한 내용이다. 서비스 패턴을 보면, 3, 4, 5, 6, 7, 12번 가상 머신은 24시간 동안 동작하고 나머지 가상 머신은 0시~16시 동안 동작한다고 가정한다.

기존의 배낭 문제를 이용한 가상머신 Consolidation은 그림 2를 보면 24시간 동작이 필요한 가상머신이

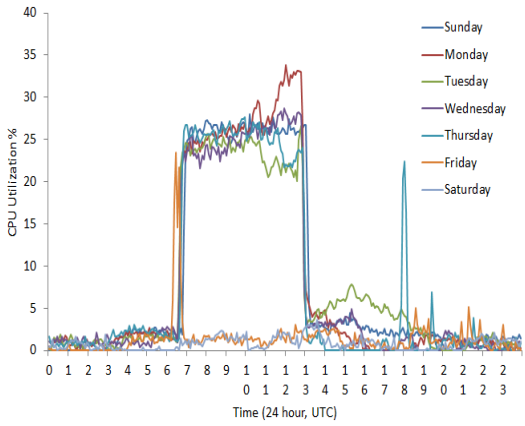

그림 3. 아마존 EC2 cloud의 호스트 서버 CPU 사용률
Fig. 3. Host server CPU utilization in Amazone EC2 cloud

표 3. 가상 머신$_i$의 자원요구량
Table 3. Working hours and resource requirements of the Virtual Machine$_i$

Virtual Machine$_i$	Ti	cpui	memi
1	16	200	200
2	16	200	600
3	24	200	200
4	24	200	250
5	24	200	200
6	24	200	200
7	24	250	250
8	16	600	100
9	16	600	500
10	16	200	600
11	16	250	300
12	24	600	150
13	16	350	600
14	16	400	500

각 물리적 서버마다 1개 이상 있다. 그러므로 물리적 서버 4대는 모두 24시간동안 동작한다. 물리적 서버의 각 시간당 전력 소비량을 Pwatt라고 가정한다. 물리적 서버의 전원 on/off가 전력사용량을 좌우하는 중요한 요소이므로, 전력량 계산을 on 상태와 off 상태일 때로 이분화하여 단순화시켰다. 전력 소비량이 Pwatt인 물리적 서버의 총 사용 전력은 4 * Pwatt * 24hours로 96PWh가 된다. 그림 4의 동작 시간을 파악하여 만든 가상 머신 Consolidation을 보면, 총 물리적 서버의 개수는 5대로 배낭 문제를 이용한 가상 머신 Consolidation 알고리즘보다 1대 늘어난다. 제안하는 방법은 24시간 동작하는 가상머신 3, 4, 5, 6, 7, 12를 한 대의 물리적 서버에 모아서 물리적 서버 한 대는 24시간 동작하고 나머지 서버는 16시간을 동작한다. 전력 소비량이 Pwatt인 물리적 서버의 총 사용 에너지를 계산하면 (1 * Pwatt * 24hours) + (4 * Pwatt * 16hours)로 88PWh가 된다. 에너지 절약 관점에서 보면 서버의 개수를 최소화하는 배낭 문제를 이용한 가상 머신 Consolidation과 비교해서 제안하는 알고리즘이 더 효율적인 것을 볼 수 있다. 예제의 결과를 바탕으로, 에너지 절약 관점에서 높은 효율을 얻기 위해서 각 물리적 서버의 동작 시간에 따른 가상 머신 Consolidation 과정이 필요한 것을 볼 수 있다.

3.4 제안하는 알고리즘

제안하는 알고리즘의 중심개념은 매우 간단하다.

사용시간이 비슷한 가상 머신들을 우선적으로 배치하는 것이다. 에너지 절약 관점에서 최대의 효율을 얻기 위해서 동작시간이 가장 긴 가상 머신 서버들을 한 개의 물리적 서버에 배치하고, 나머지 가상 머신 서버들은 사용시간에 맞추어 전원을 on/off함으로써 에너지 낭비를 줄일 수 있다. 그림 4는 테스트를 위해, 가상 머신 서버들의 시작 시간을 0시~16시간 또는 24시간동안 동작하게 정한다. 그러나 실제의 가상 머신 서버들은 시작 시간과 동작 시간이 다르기 때문에 알고리즘에 새로운 추가내용이 필요하다. 가상 머신 서버의 동작시간이 정해져 있다면 가상 머신 서버 중 동작시간이 가장 긴 가상 머신 서버를 물리적 서버에 우선 배치하고, 우선 배치된 가상 머신이 동작하는 시간대와 동작시간이 겹치는 값이 클수록 같은 물리적 서버에 배치하면 그림 4와 같은 가상 머신의 물리적 서버 배치를 기대할 수 있다.

가상 머신을 시간패턴에 따라 Consolidation하기 위해서는 배낭 문제 알고리즘을 이용한 에너지 인식 Consolidation이 필요하다. 그림 5는 제안하는 에너지 인식 Consolidation의 동작 순서를 표현한 플로우 차트이다. 플로우 차트의 시작은 가상 머신을 사용시간이 긴 순서대로 Sorting한다. 가장 사용시간이 긴 가상 머신을 물리적 서버에 우선 배치하고, 이 가상 머신과 중첩되는 시간이 길수록 해당 가상 머신의 Value를 높게 한다. 우선 배치된 가상 머신을 제외한 i-1개의 가상 머신에 Value값이 정해지면, 기존 배낭 문제 알고리즘을 통해 n개의 가상 머신를 물리적 서버에 배치한다. 물리적 서버에 배치되지 않은 가상 머신은 (i-n)개가 된다. 과정을 반복한 후 남은 가상 머신이 0대가 되면, Consolidation 과정을 끝낸다.

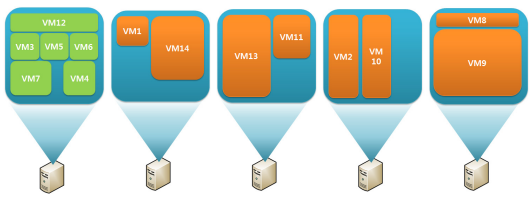

그림 4. 동작 시간에 따른 가상 머신 Consolidation
Fig. 4. Virtual Machine Consolidation according to the service pattern

Ⅳ. 시뮬레이션 결과

배낭 문제를 이용한 가상 머신 Consolidation방법과 동작 시간에 따른 가상 머신 Consolidation방법을 C++로 구현해 테스트를 한다. 테스트에 사용될 각 물리적 서버의 CPU 용량은 3GHZ, 메모리 용량은 8GB로 정한다. 각 가상 머신 서버의 CPU 요구용량은 100~800MHZ 사이의 값으로 정하고, 메모리 요구용량

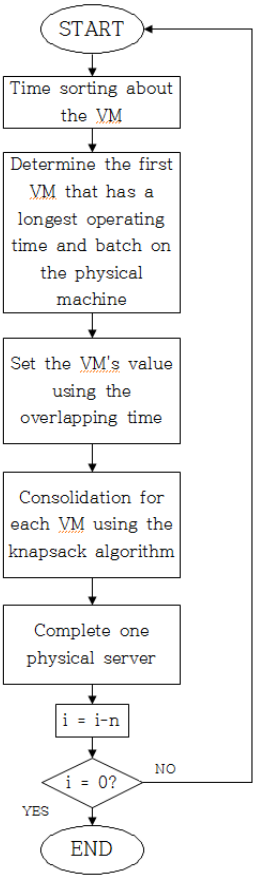

그림 5. 제안하는 알고리즘 동작 순서
Fig. 5. The proposed algorithm action sequence

은 100~800MByte 사이의 임의의 값으로 정한다. 가상 머신의 동작하는 시작시간은 0시~12시 사이의 값이고, 하루 동안 동작하는 시간은 5시~12시간으로 정한다.

비교 분석을 위해 가상 머신 Consolidation방법과 동작 시간에 따른 가상 머신 Consolidation방법으로 만들어지는 각 물리서버의 개수와 전력 사용량을 비교한다. 알고리즘 3은 제안하는 가상 머신 Consolidation에 알고리즘 2를 적용하여 코드로 구현한 것이다. 알고리즘 3의 외부변수 matrix[200][3000][8000]는 알고리즘2의 K[i, cpu, mem]와 같은 변수이며, 배낭 문제 알고리즘에서 Value값의 최대값을 산정할 때 사용된다. 코드에서 matrix 배열의 첫 번째 원소인 200은 가상 머신의 최대 개수인 nItems를 의미하고, 두 번째 원소인 3000은 CPU의 최대용량인 3GHZ값인 cpu, 세 번째 원소인 8000은 메모리의 최대용량인 8GB인 mem을 나타낸다. 코드는 알고리즘 2의 동작과 같은

Algorithm3. The proposed VM Consolidation Algorithm

```
1:  int matrix[200][3000][8000]

2:  for (i = 1; i <= nItems; i++){
3:   for (j = 1; j <= cpu; j++){
4:     for (k = 0; k <= mem; k++){
5:       if((vm[i].cpu_use > j) &&
6:       (vm[i].mem_use > k)){
7:         matrix[i][j][k] = matrix[i-1][j][k];
8:       }
9:       else{
           matrix[i][j][k] = max(matrix[i-1][j][k],
10:        vm[i-1].value + matrix[i-1][j-vm[i-1].cp
11:          u][k-vm[i-1].mem]);
         }
       }
     }
   }
12: return matrix[nItems][cpu][mem];
```

순서로 진행되며 마지막에 matrix[nItems][cpu][mem]을 리턴 값으로 받는다. 코드를 이용하여, 제안하는 가상 머신 Consolidation과 기존 배낭 문제를 이용한 가상 머신 Consolidation을 구현할 수 있다. 두 방법의 차이점은, 기존 배낭 문제를 이용한 가상 머신 Consolidation은 모든 가상 머신 서버에 대해 배낭 문제 알고리즘을 이용한 물리적 서버 배치를 하고, 제안하는 가상 머신 Consolidation 방법은 가장 사용시간이 긴 가상 머신 서버를 물리적 서버에 우선 배치한 뒤에 배낭 문제 알고리즘을 통해 서버에 가상 머신을 배치한다는 점이 다르다.

Value값을 정할 때, 기존 배낭 문제를 이용한 가상 머신 Consolidation과 제안하는 가상 머신 Consolidation은 차이가 있다. 기존 배낭 문제를 이용한 가상 머신 Consolidation의 i번째 가상 머신의 Value값은 vi = cpui * memi이 되고, 제안하는 가상 머신 Consolidation는 물리적 서버에 우선 배치된 가상 머신과 i번째 가상 머신의 겹치는 동작시간의 범위 값이 i번째 가상 머신의 Value값이 된다. 위의 알고리즘 4는 제안하는 가상 머신 Consolidation의 Value값을 정하는 코드 내용이다. first는 가상 머신 중에 서버에 우선 배치된 가상 머신 서버를 가리킨다. 우선 배치된 가상 머신인 first와 겹치는 시간을 구하기 위해 start, end의 변수를 만든다. first와 겹치는 시간 중에 처음

```
Algorithm4. Algorithm for the VM's value set
1:  first = VM that has a longest operating time
2:  start_time = start time of the VM
3:  end_time = end time of the VM
4:  i = for each VM
5:  for(i = 1~n)
6:    if(i.start_time < first.start_time)
7:      start = first.start_time
8:    else
9:      start = i.start_time
10:   if(i.end_time < first.end_time)
11:     end = i.end_time
12:   else
13:     end = first.end_time
14: Overlapping time = end - start
```

부분을 start로 정하고, 마지막 부분을 end로 정한다. end에서 start의 값을 빼면 first와 i번째 가상 머신의 겹치는 동작시간의 범위 값이 나온다. 이 값을 각 i번째 가상 머신의 Value값으로 정한다.

테스트는 가상 머신 20개의 Consolidation을 시작으로, 가상 머신의 개수를 20개씩 늘려가며 최대 160개까지의 가상 머신에 대해 기존 배낭 문제를 이용한 가상 머신 Consolidation 방법과 제안하는 가상 머신 Consolidation 방법으로 각 각 시뮬레이션한다.

그림 6은 가상 머신의 개수를 늘리면서 알고리즘을 통해 만들어지는 서버의 개수를 비교한 그림이다. 가상 머신 20개의 경우, 기존 방법을 통한 Consolidation 방법은 3개의 물리적 서버에 배치되고 제안하는 알고리즘은 4개의 물리적 서버에 배치된다. 가상 머신 40개의 경우, 기존 알고리즘은 6대, 제안하는 알고리즘은 7대에 배치되며 기존 배낭 문제를 이용한 가상머신 Consolidation 알고리즘이 물리적 서버 최소화 면에서 더 효율적인 것을 볼 수 있다. 가상 머신을 20개씩 늘리며 160대까지 테스트해 본 결과, 가상 머신이 120개가 될 때까지 제안하는 알고리즘으로 만든 물리적 서버의 개수가 1대 더 많았고 그 이후의 140개 이상부터는 만들어진 서버의 개수가 같았다.

그림 7은 전력 소모량에서 기존 배낭 문제를 이용한 가상 머신 Consolidation 방법과 제안하는 가상 머신 Consolidation방법을 비교한 그래프이다. 물리적 서버 1대의 시간당 전력 소모량은 50W로 정한다. 그림 6에서 배낭 문제를 이용한 가상 머신 Consolidation 방법이 제안하는 Consolidation보다 물리적 서버의 개수가 적거나 같다. 그러나 그림 7의 전력 소모량을 비교해보면, 제안하는 가상 머신 Consolidation 방법이 기존 알고리즘보다 서버의 개수가 많거나 같지만, 전력 소모량은 적은 것을 볼 수 있다. 가상 머신 20개의 경우, 기존 방법을 통한 Consolidation 방법은 총 전력소모량이 2900Wh이고 물리적 서버 1대당 평균 전력 소모량은 145Wh이다. 제안하는 알고리즘은 총 전력소모량은 1850Wh이고 물리적 서버 1대당 평균 전력 소모량은 92.5Wh이다. 가상 머신 20개에서 제안하는 알고리즘은 약 36%의 에너지 효율을 보인다. 가상 머신 40개의 경우, 기존 알고리즘은 총 전력소모량이 5650Wh, 물리적 서버 1대당 평균 전력 소모량이 141.25Wh였고 제안하는 알고리즘은 총 전력소모량이 3300Wh, 물리적 서버 1대당 전력 소모량은 82.5Wh이다. 가상 머신 40개에서 제안하는 알고리즘은 약 42%의 에너지 효율을 보인다. 가상 머신이 개수가 커질수록 기존 배낭 문제를 이용한 가상 머신 Consolidation 알고리즘과 제안하는 알고리즘의 전력 소모량 차이가 더 커진다. 평균 물리적 서버 1대당 전력 소모량도 제안하는 알고리즘이 기존 알고리즘 방법보다 더 낮은 전력 사용량을 유지한다. 제안하는 알고리즘의 에너지 효율은 최소값인 가상 머신 60개의

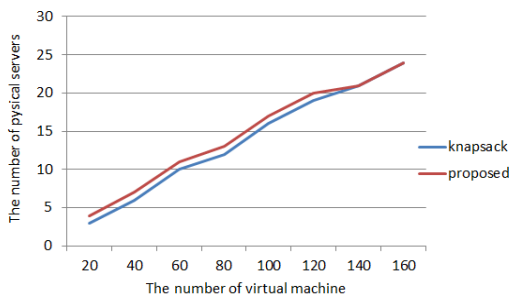

그림 6. 서버 개수 비교
Fig. 6. Compare the number of servers

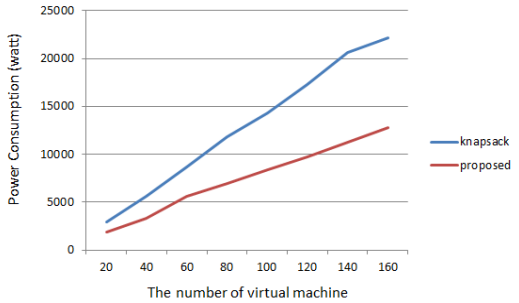

그림 7. 총 전력 소모량 비교
Fig. 7. Compare the total power consumption

35.6%부터 최대값인 가상 머신 140개의 45.5%까지 35.6%~45.5%로 기존 알고리즘에 비례하여 높은 효율을 나타낸다. 위의 결과를 통해, 제안하는 동작 시간에 따른 가상 머신 Consolidation 방법이 전력 소모량 면에서 기존 방법보다 우수한 면을 보이는 것을 입증한다.

V. 결 론

본 연구의 결과를 정리하면 다음과 같다. 기존 배낭 문제를 이용한 가상 머신 Consolidation은 각 가상 머신의 하드웨어 자원의 요구량에 따라 물리적 서버에 가상 머신을 Consolidation하고 만들어지는 물리적 서버의 개수는 최소가 된다. 이 방법은 서버의 개수는 최소지만 에너지 소모량은 최소량을 갖지 않는다. 제안하는 가상 머신 Consolidation은 가상 머신을 시간 동작 시간에 따라 동적으로 물리적 서버에 배치한다. 서버의 개수는 최소가 아니지만 에너지 효율에서 기존 알고리즘보다 큰 성능개선을 보인다.

본 연구의 결과에서 나타나는 바와 같이, 제안하는 알고리즘은 에너지 효율 면에서 데이터센터 와 그 외 가상 머신 서버를 다루는 여러 IT 분야에 기여할 수 있을 것이다.

References

[1] S. L. Sams, *Discovering hidden costs in your data centre-a CFO perspective(2011)*, Retrieved Jul. 22, 2014, from ibm.com/services/siteand facilities.

[2] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," *ASPLOS*, vol. 37, no. 1, pp. 205-216, Mar. 2009.

[3] M. Kim and M. Park, "VM consolidation based on dynamic programing knapsack algorithm," *KIPS*, Gyeonggi-Do, Korea, Apr. 2014.

[4] L. A. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer,* vol. 40, no. 12, pp. 33-37, Dec. 2010.

[5] Seongbong Yang, *Intuitive Algorithm(국문: 알기 쉬운 알고리즘)*, Saeng-Neung Publisher, PP. 169-178, 2013

[6] R. S. Camati, A. Calsavara, and L. Lima Jr., "Solving the virtual machine placement problem as a multiple multidimensional knapsack problem," *MMEDIA*, Nice, France, Feb. 2014.

[7] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," *INFOCOM*, pp. 1154-1162, NY, USA, Mar. 2010.

[8] N. Bobroff, A. Kochut, and K. A. Beaty, "Dynamic placement of virtual machines for managing SLA violations," *IFIP/IEEE Int. Symp. Integrated Netw. Management,* pp. 119-128, Munich, Germany, May 2007.

[9] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," *NSDI*, pp. 229-242, CA, USA, Apr. 2007

[10] A. Kochut and K. A. Beaty "On strategies for dynamic resource management in virtualized server environments," *Int. Symp. Modeling, Anal., Simulation of Comput. Telecommun. Syst. (MASCOTS '07)*, pp. 193-200, Istanbul, Turkey, Oct. 2007.

[11] Z. Gong and X. Gu, "PAC: Pattern-driven application consolidation for efficient cloud computing," *Int. Symp. Modeling, Anal., Simulation of Comput. Telecommun. Syst. (MASCOTS)*, pp. 24-33, FL, USA, Aug. 2010.

[12] H. Liu, *Host server CPU utilization in Amazon EC2 cloud(2012)*, Retrieved Nov. 1, 2014, from https://huanliu.wordpress.com/2012/02/17/host-server-cpu-utilization-in-amazon-ec2-cloud/

김 민 회 (Minhoe Kim)

2014년 2월 : 숭실대학교 정보통신전자공학부 졸업

2014년 3월~현재 : 숭실대학교 정보통신공학과 석사

<관심분야> 통신네트워크, 통신 소프트웨어, 네트워크 보안

박 민 호 (Minho Park)

2000년 2월 : 고려대학교 전자공학과 졸업
2002년 2월 : 고려대학교 전자공학과 석사
2010년 2월 : 서울대학교 전기컴퓨터공학과 박사

<관심분야> 통신네트워크, 통신소프트웨어, 네트워크 보안