클라우드 컴퓨팅에서 CPU 사용률을 고려한 가상머신 할당 기법

배준성* · 이봉환**

A Virtual Machine Allocation Scheme based on CPU Utilization in Cloud Computing

Jun-Sung Bae* · Bong-Hwan Lee**

이 논문은 2010년도 교육과학기술부와 한국연구재단의 지역혁신인력양성사업 및 일반연구자 지원사업으로 수행된 연구 결과임 (No. 2010-0016574)

요 약

클라우드 컴퓨팅 환경에서 가상 머신의 할당 기법인 매치메이킹, 라운드 로빈은 노드의 CPU 사용률을 고려하지 않고 CPU, RAM, HDD 등의 하드웨어 사양만으로 가상 머신을 할당한다. 이렇게 노드의 자원 할당 가능 여부만을 고려한 노드 선택 과정은 자원의 균등한 분배 측면에서 효율적이라 할 수 있지만 자원의 사용률 측면에서는 효율적이지 못하다. 따라서 본 논문에서는 사용자의 과거 가상 머신 사용률을 기반으로 노드의 CPU 사용률을 계산해 가장 작은 CPU 사용률을 갖는 노드를 선택하여 가상 머신을 할당하는 기법을 제안한다. 제안한 할당 기법의 효율성을 검증하기 위해 매치메이킹, 라운드로빈 할당 기법과 비교 실험을 진행하였고, 실험 결과 제안하는 할당 기법을 적용한 노드들이 타 기법을 적용한 노드들보다 전체적인 CPU 사용률에 있어 고른 분포를 보여 제안하는 할당 기법이 노드의 부하 분산에 효율적임을 입증하였다.

ABSTRACT

The two most popular virtual machine allocation schemes, both match making and round robin, do consider hardware specifications such as CPU, RAM, and HDD, but not CPU usage, which results in balanced resource distribution, but not in balanced resource usage. Thus, in this paper a new virtual machine allocation scheme considering current CPU usage rate is proposed while retaining even distribution of node resources. In order to evaluate the performance of the proposed scheme, a cloud computing platform composed of three cloud nodes and one front end is implemented. The proposed allocation scheme was compared with both match making and round robin schemes. Experimental results show that the proposed scheme performs better in even distribution of overall CPU usage, which results in efficient load balancing.

키워드

클라우드 컴퓨팅, 가상머신, 임대 서비스, OpenNebula, Xen

Key word

Cloud Computing, Virtual Machine, OpenNebula, Xen

* 준회원 : 대전대학교 정보통신공학과 석사 졸업
** 종신회원 : 대전대학교 정보통신공학과 교수 (교신저자, blee@dju.kr)

접수일자 : 2011. 01. 18
심사완료일자 : 2011. 02. 17

Ⅰ. 서 론

클라우드 컴퓨팅[1,2]은 인터넷상의 서로 다른 물리적인 위치에 존재하는 각종 컴퓨팅 자원들을 가상화[3,4] 기술로 통합하여 사용자에게 언제 어디서나 필요한 양만큼 편리하고 저렴하게 사용할 수 있는 IT 환경을 제공하는 컴퓨팅 패러다임이다.

클라우드 컴퓨팅 구축을 위한 플랫폼으로는 오픈니블라(OpenNebula)[5,6], 님버스(Nimbus)[7], 유칼립투스(Eucalyptus)[8] 등이 있다. 각 플랫폼별로 자원의 효율적인 관리를 위한 가상머신 스케줄러가 포함되어 있다. 대표적인 가상 머신 스케줄러로는 오픈니블라의 매치메이킹(Matchmaking)[9], Haizea, 유칼립투스의 라운드 로빈(Round- Robin)[10] 등이 있다.

이 스케줄러들은 가상 머신 생성에 필요한 사용자의 요구사항으로 CPU, 메모리, 하드디스크 등의 물리적 자원량만을 파라미터로 사용한다. 이렇게 노드의 자원 할당 가능 여부만을 고려한 노드 선택 과정은 자원 분배 측면에서 효율적이라 할 수 있지만 자원의 사용률 측면에서 보면 효율적이지 못하다. 이유는 가상 머신 사용자들의 사용 패턴에 따라 자원 사용률이 다르고 그에 따른 부하가 달라지기 때문이다. 즉, 특정 노드는 과부하 상태가 되고 다른 노드들은 유휴 자원이 많은 상태가 될 것이다.

이런 노드의 부하 분산을 위해 각 스케줄러들은 동적 마이그레이션[11] 기능을 제공한다. 하지만 마이그레이션 중에 생기는 오버헤드로 인해 가상 머신의 성능이 일시적으로 저하되는 문제점이 있다[12]. 이 문제점을 해결하기 위해 가상 머신 할당 시의 노드의 현재 CPU 사용률을 고려할 필요성이 있다.

본 논문에서는 가상머신 생성 과정에서 노드의 CPU 사용률의 고른 분포와 부하 분산을 위한 가상 머신 할당 기법을 제안한다. 제안하는 할당 기법은 가상 머신 할당 과정에서 현재 노드들의 일정 기간 동안 전체 CPU 사용률을 기반으로 타 노드에 비해 평균적으로 CPU 사용률이 떨어지는 노드에게 가상 머신을 할당함으로써 타 할당 기법들이 갖는 노드 간 CPU 사용률의 불균형 문제를 완화시켰다. 즉, 전체 노드들의 CPU 사용률을 고르게 분포시켰고, 이를 통해 효율적인 부하 분산을 가능케 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 기술하며, 3장은 본 논문에서 제안한 가상 머신 할당 기법 설계 및 동작과정과 성능 측정을 위한 클라우드 컴퓨팅 테스트베드의 구축에 대해 설명한다. 4장에서는 제안한 가상 머신 할당 기법과 오픈니블라의 매치메이킹, 유칼립투스의 라운드 로빈 스케줄러의 가상머신 할당 기법과의 차이점을 설명하고 CPU 사용률과 부하 분산도를 측정하여 제안한 가상 머신 할당기법의 효율성을 검증한다.

Ⅱ. 관련연구

매치메이킹은 기본적으로 FCPU의 잔량 순위를 매긴 뒤 가상머신을 할당한다. FCPU는 노드에서 사용할 수 있는 CPU의 양으로 할당 여유분을 나타내는 ACPU와는 다른 의미를 갖는다. 즉, FCPU는 가상머신들의 CPU 사용 여부에 따라서 수시로 달라지는 값이며, ACPU는 가상머신들에 할당되고 남은 양을 의미하므로 고정적인 값이다. 따라서 수시로 계속 변하는 값을 참조하여 가상머신을 할당하므로 실제적으로 노드의 부하 분산에는 효과가 없다. 만약 순위가 같을 경우에는 오픈니블라에 등록되어진 노드의 순서에 따라 할당이 된다. 매치메이킹은 그림 1과 같이 물리적 자원의 각 구성 요소들에 대해 속성을 정의하고 수치화를 시킨다.

```
[
  Type            = "Machine";
  Activity        = "Idle";
  DayTime         = 36107 // current time
    // in seconds since midnight
  KeyboardIdle    = 1432;     // seconds
  Disk            = 323496;   // kbytes
  Memory          = 64;       // megabytes
  State           = "Unclaimed";
  LoadAvg         = 0.042969;
  Mips            = 104;
  Arch            = "INTEL";
  OpSys           = "SOLARIS251";
  KFlops          = 21893;
  Name            = "leonardo.cs.wisc.edu";
  ResearchGroup   = { "raman", "miron",
                      "solomon", "jbasney" };
  Friends         = { "tannenba", "wright" };
  Untrusted       = { "rival", "riffraff" };
  Rank            =
    member(other.Owner, ResearchGroup) * 10
    + member(other.Owner, Friends);
  Constraint      =
    !member(other.Owner, Untrusted)
       && Rank >= 10
          ? true
    : Rank > 0
          ? LoadAvg<0.3 && KeyboardIdle>15*60
    : DayTime < 8*60*60
          || DayTime > 18*60*60;
]
```

그림 1. 매치메이킹 기법의 정규 표현식
Fig. 1. The regular expression of the match making scheme

라운드 로빈은 작업의 실행 순서 등을 결정하는 방법으로 CPU 스케줄링, 네트워크 DNS 스케줄링, 가상 머신 스케줄링 등 여러 자원들의 선택 또는 공유 문제 해결에 많이 사용되고 있다.

라운드 로빈은 가상 머신 할당 스케줄러에도 적합한 구조를 갖고 있다. 순차적으로 순환하며 자원을 할당 하는 방식은 서버 자원의 균등한 자원 분배를 가능케 한다. 즉, 존재하는 모든 노드에 차례대로 가상 머신을 할당함으로써 자원 활용률을 고르게 할 수 있다. 하지만 자원의 균등 배분 측면에선 좋으나 노드의 CPU 사용률을 고려하지 않는 단점이 있다.

Ⅲ. CPU 사용률을 고려한 가상머신 할당 기법

매치메이킹, 라운드 로빈은 노드의 현재 CPU 사용률을 고려하지 않으므로 이후 노드들의 부하를 예측할 수 없다. 이는 같은 CPU를 갖는 가상머신이라 할지라도 사용자의 사용 패턴에 따라 노드간 CPU 사용률이 달라지기 때문이다. 따라서 부하 분산 측면에서 약점을 보인다. 본 장에서는 이러한 문제를 해결하기 위해 가상머신 할당 요구에 의한 노드 선택 시에 물리적 자원의 할당 여부만을 보는 것이 아니라, 자원의 활용 상태까지 같이 고려하는 가상머신 할당 기법을 소개한다.

3.1 NSSL(Node Selector Service Layer)

그림 2의 NSSL(Node Selector Service Layer)는 본 논문에서 제안하는 할당 기법을 추상화하여 표현한 그림이다. NSSL은 NIM(Node Information Module), NSM(Node Selector Module), 및 데이터베이스로 구성이 된다. NSSL은 사용자 계층과 OpenNebula 계층 사이에 위치하며 할당 기법의 실제 클라우드 컴퓨팅 적용을 위해 JAVA로 모듈을 작성하였고 노드들로부터 수신된 데이터를 저장하기 위해 MySQL을 사용하여 데이터베이스를 구현하였다.

NIM은 데이터베이스에 가상머신들의 최근 CPU 사용률을 저장 또는 불러오는 역할을 수행하며, NSM은 NIM으로부터 받은 정보를 바탕으로 최적의 노드를 찾는 역할을 수행한다.

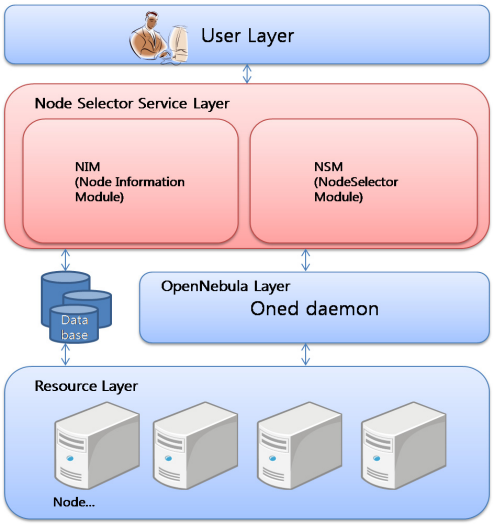

그림 2. 제안하는 할당 기법의 계층도
Fig. 2. Hierarchy of the Proposed Allocation Scheme

3.2 전체 수행 과정

그림 3은 제안하는 할당 기법의 전체 수행 순서도이다.

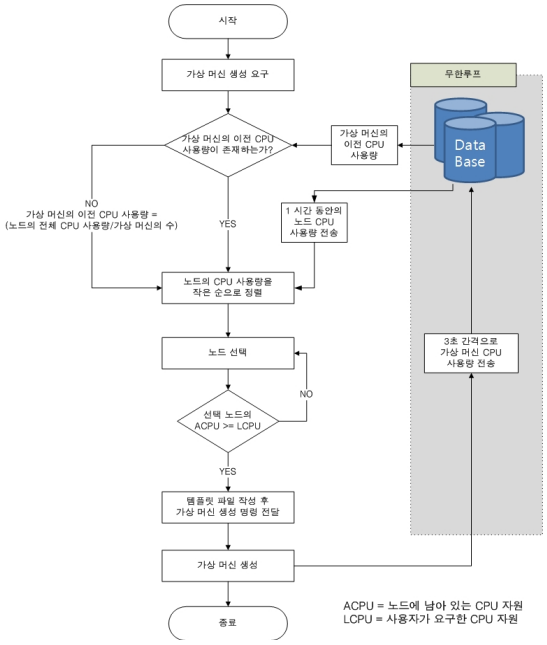

그림 3. 제안하는 할당 기법의 전체 수행 과정
Fig. 3. Flowchart of the Proposed Allocation Scheme

가상 머신이 생성되는 과정을 단계별로 설명하면 다음과 같다.

- 가상머신의 생성 요구를 받는다.
- 데이터베이스에 해당 가상 머신의 이전 사용 정보가 있는지 확인한 후, 없으면 현재 노드의 전체 CPU 사용률/노드에 생성되어 있는 가상머신의 수로 나누어 대입한다.
- 노드의 전체 CPU 사용률을 서로 비교하여 가장 낮은 CPU 사용률을 보이는 노드를 선택한다.
- 선택된 노드에 여유 CPU가 있는지 확인한다.
- 오픈니블라의 ONE 데몬에 전달할 가상머신 템플릿 파일을 작성한다.
- 가상머신이 생성되면 3초의 간격으로 데이터베이스에 CPU 사용량을 기록한다.

3.3 클라우드 컴퓨팅 테스트베드 구축

제안하는 가상 머신 할당 기법의 효율성을 검증하기 위해 클라우드 컴퓨팅 테스트베드를 구축했다. 테스트베드의 전체적인 구축도는 그림 4와 같다.

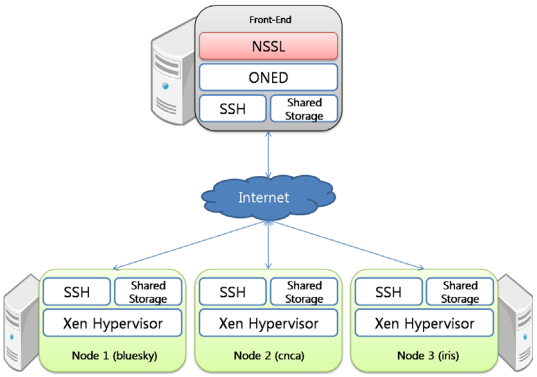

그림 4. 클라우드 컴퓨팅 테스트베드
Fig. 4. Cloud Computing Test Bed

Front-End에는 오픈니블라와 제안하는 가상머신 할당 기법을 프로그램 모듈화 시킨 NSSL이 설치되어 있다. Front-End에는 사용자의 가상 머신 생성 요구를 받고 그에 따른 할당 작업을 수행한다. Node 1~3에는 가상화 플랫폼 Xen이 설치되어 있다. Node에 가상머신이 생성되어 구동되게 된다. 테스트베드의 사양은 표 1과 같다.

표 1. 클라우드 컴퓨팅 테스트베드 사양
Table 1. Cloud Computing Test Bed Specification

	CPU	RAM	HDD	OS
Front-End	Dual Core 2.13GHz	2GB	230GB	ubuntu9.04 i686
Node 1	Core 2 Duo 2.80GHz	8GB	460GB	ubuntu8.04 i686
Node 2	Core 2 Duo 2.80GHz	8GB	460GB	ubuntu8.04 i686
Node 3	Core 2 Duo 2.80GHz	8GB	460GB	ubuntu8.04 i686

3.4 고려사항

3.4.1 Xen의 스케줄러 특성 고려

Xen은 가상 머신의 생성 및 관리를 위해 Credit 스케줄러를 사용한다. Credit 스케줄러는 가상 머신별로 cap이라 불리는 일정한 가중치를 부여하고 부여된 가중치만큼 남는 유휴 CPU 자원을 더 할당해 주는 방식이다. cap의 관리는 IDD(Isolated Driver Domain)라는 계층에서 수행한다. 처음 Xen은 서버 환경에서의 가상화를 위해 만들어졌기 때문에 cap을 분배하는 정책은 서버에 최적화되었다. IDD는 디스크 I/O 작업 요구량이 많은 가상 머신에게 cap을 우선적으로 주게 되는데 서버에서 발생하는 대부분의 작업은 디스크 I/O 또는 네트워크 문제지만 다수의 가상 머신을 구동시키는 클라이언트 환경에서는 다양한 종류의 부하가 있을 수 있기 때문에 작업량이 많더라도 디스크 I/O가 적은 가상 머신은 cap의 분배에서 소외될 수 있고 이는 성능에도 영향을 줄 수 있다. 본 논문에서는 다수의 가상 머신이 생성되기 때문에 가상 머신들의 정확한 CPU 사용량을 측정하기 위해 Credit 스케줄러의 cap을 고정시킨다.

3.4.2 가상 머신의 CPU 사용률

가상 머신은 노드 안에서 구동되는 것이므로 가상 머신의 CPU 사용량을 모두 더한 값이 해당 노드의 CPU 사용량이라 할 수 있다. 하지만 본 논문에서 제안하는 할당 기법에 필요한 파라미터에는 가상 머신의 과거 정보가 필요하므로 노드의 전체 CPU 사용률이 아닌 가상 머신

의 CPU 사용률을 노드 선택에 필요한 파라미터로 사용한다. 여기서 가상 머신의 과거 정보란 해당 가상 머신의 이전 사용 내역을 데이터베이스에 저장해 놓은 것을 의미한다.

3.4.3 노드 할당의 기준

본 논문에서는 할당 받은 노드의 전체 CPU 사용률이 타 노드와 균등한지를 보는 것이므로 노드의 전체 CPU 사용률만을 노드 선택의 기준으로 삼는다.

Ⅳ. 제안하는 가상 머신 할당 기법의 성능 평가

본 장에서는 제안하는 가상 머신 할당 기법의 성능을 테스트하기 위하여 실험을 수행하였다. 성능 비교 대상은 오픈니블라의 매치메이킹 스케줄러와 유칼립투스의 라운드 로빈 스케줄러이다. 총 3대의 노드에 21대의 가상 머신을 생성시키며 가상 머신별로 임의 크기의 부하를 내도록 만들어졌다. 부하를 만들어내기 위해 SuperPI, 리눅스의 스케줄러인 Crontab과 쉘 스크립트를 사용하였다.

4.1 실험 시나리오

가상 머신을 이용하는 목적이나 범위에 따라 가상 머신의 자원 요구 사항도 달라질 것이다. 예를 들어 PC방, 학교 실습실, 백화점 단말기 등과 같이 정적인 환경이라면 가상 머신에 필요한 하드웨어 자원은 동일할 가능성이 매우 높다. 반대로 웹을 통해 불특정 다수의 사용자들에게 가상 머신을 서비스하는 경우처럼 매우 동적인 환경이라면 가상 머신에 필요한 하드웨어 자원은 불규칙할 것이다. 따라서 본 논문에서 제안하는 가상 머신 할당 기법이 위에서 설명한 두 가지 경우의 가상 머신 할당 과정에서 CPU의 사용률과 부하 분산 측면에 어떤 장점이 있는지를 실험을 통해 확인한다.

CPU의 사용률은 상황에 따라 예측하기 힘들고 같은 파라미터를 가지고 실험을 하여도 그 값은 오차가 존재할 수 있다. 따라서 가상 머신 별로 최대 CPU 사용량만을 지정한 상태로 임의의 부하를 유도할 경우 결과 데이터별로 값이 각각 다르게 나타날 것이다. 이는 성능 평가의 신뢰도에 영향을 끼치므로 CPU 사용량에 따라 등급별로 나누고 Xen에서의 cap 값 고정을 통해 가상 머신별로 사용할 수 있는 CPU의 클록을 제한하였다. 즉, CPU의 초기 할당량은 똑같지만 다시 그 안에서 CPU의 클록에 제한을 두는 것이다. 그런 다음 최대 부하를 지속적으로 발생시키면 오차가 거의 없는 CPU 사용량을 얻을 수 있다. 가장 높은 등급을 갖는 가상 머신의 최대 CPU 사용량은 실제로 할당 받은 CPU를 100% 사용했을 때의 사용량이다.

표 2와 3은 가상 머신의 사양이 같을 때의 자원 할당과 가상 머신들이 가질 수 있는 최대 CPU 사용량과 등급별 할당 대수를 나타낸다.

표 2. 가상 머신 사양(CPU 사양은 동일)
Table 2. Virtual Machine Specification (the same CPU specification)

CPU	560MHz
Memory	512MB
HDD	1GB
Network	10Mbps

표 3. 가상 머신 분류(CPU 사양은 동일)
Table 3. Types of Virtual Machines (the same CPU specification)

가상 머신	CPU 사용량	대수
낮은 CPU 사용률	10	7
중간 CPU 사용률	15	7
높은 CPU 사용률	20	7

표 4와 5는 가상 머신의 사양이 다를 때의 자원 할당과 가상 머신들이 가질 수 있는 최대 CPU 사용량과 등급별 할당 대수를 나타낸다.

표 4. 가상 머신 사양(CPU 사양 다름)
Table 4. Virtual Machine Specification(Different CPU Specification)

CPU	280MHz, 560MHz, 840MHz
Memory	512MB
HDD	1GB
Network	10Mbps

표 5. 가상 머신 분류(CPU 사양 다름)
Table 5. Types of Virtual Machines(Different CPU Specification)

가상 머신	CPU 사용량	대수
낮은 CPU 사용률	10	7
중간 CPU 사용률	20	7
높은 CPU 사용률	30	7

4.2 실험 결과

4.2.1 가상 머신의 사양이 동일한 경우

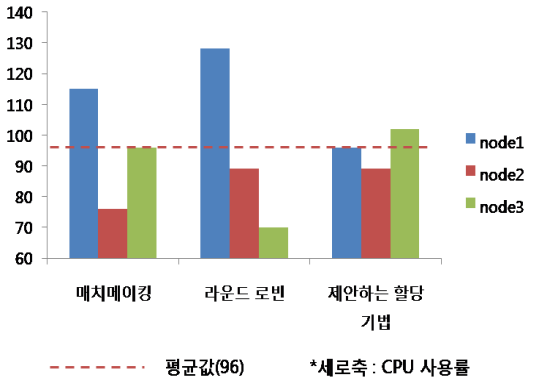

그림 5. 노드의 CPU 사용률 비교(CPU 사양 동일)
Fig. 5. Comparison of CPU Utilization of a Node(the same CPU specification)

그림 5는 스케줄링 기법별 노드 CPU 사용률을 비교한 그래프이다. 기준 값이 의미하는 것은 각 할당 기법을 적용한 노드들의 전체 CPU 사용률에 대한 평균값으로 1대의 노드가 가질 수 있는 가장 이상적인 CPU 사용률을 의미하며, 노드들의 CPU 사용률이 기준 값에 근접하는지 아닌지의 여부를 통해 CPU의 사용률이 고르게 분포하는지의 여부를 확인할 수 있다. 먼저 매치메이킹의 경우 node 3을 제외한 타 노드들의 CPU 사용률이 기준 값인 96과 큰 차이를 보이며 떨어져 있다. 라운드 로빈의 경우 node 1의 CPU 사용률은 약 130인 반면 node 3은 70 근처에 머물고 있어 노드간 CPU 사용률이 고르지 못하다는 것을 알 수 있다.

반면 제안하는 할당 기법을 적용한 노드들의 경우 기준 값 96에 거의 일치하는 CPU 사용률을 보여 특정 노드의 CPU 사용률이 떨어지거나 반대로 높은 경우를 보이는 타 할당 기법들에 비해 CPU 사용률이 고른 장점을 확인할 수 있다. 그림 6은 5회 반복 실험한 결과의 그래프이다.

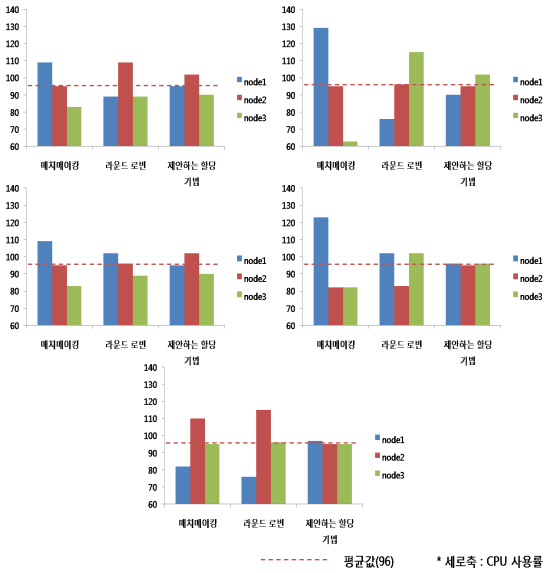

그림 6. 노드의 CPU 사용률 비교(CPU 사양 동일)
Fig. 6. Comparison of CPU Utilization of a Node (the same CPU specification)

그림 6처럼 5회 반복 실험한 결과에서도 제안하는 할당 기법이 타 가상 머신 할당 기법과 비교하였을 때, 평균값에 더욱 근접하여 CPU의 활용과 부하 분산이 잘되고 있음을 확인할 수 있다.

그림 7은 매치메이킹, 라운드 로빈 및 제안하는 할당 기법을 통해 얻은 노드별 CPU 사용률의 표준 편차 그래프이다. 매치메이킹과 라운드 로빈의 경우 표준 편차 값은 약 6~33 사이에 위치하고 변화 폭이 상당히 큰 편이며 평균적으로 각각 19 및 16의 표준 편차를 갖는다. 반면 제안하는 할당 기법을 적용한 경우는 약 1~6 사이의 낮은 표준 편차 값을 가지며 평균적으로 약 4의 표준 편차를 가져 노드간 CPU 사용률이 고르게 분포되고 있음을 확인할 수 있다.

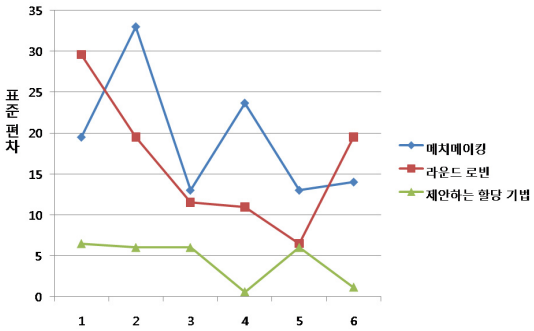

그림 7. 노드의 CPU 사용률 표준 편차 비교 (CPU 사양 동일)
Fig. 7. Standard Deviation Comparison of CPU Usage of a Node (the same CPU specification)

4.2.2 가상 머신의 사양이 다른 경우(실험 결과)

그림 8은 각 스케줄링 기법별 가상 머신 평균 CPU 사용률의 전체 합을 비교한 그래프이다.

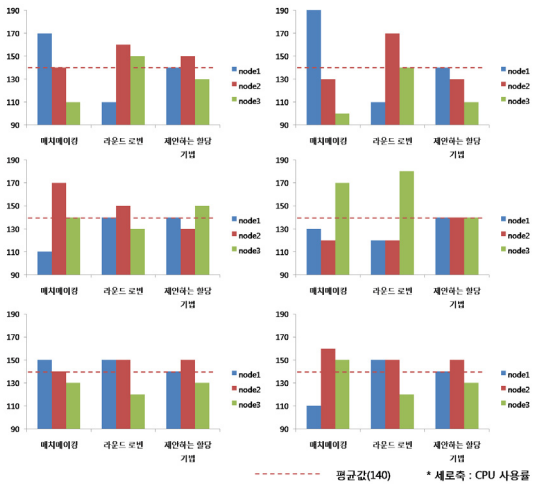

그림 8. 노드의 CPU 사용률 비교(CPU 사양 다름)
Fig. 8. CPU Utilization Comparison of a Node (Different CPU Specification)

그림 6과 마찬가지로 매치메이킹과 라운드 로빈은 전체적으로 CPU 사용률의 분포가 고르지 못하다. 또한 가상 머신의 CPU 할당량에 따라 사용량의 크기도 변하므로 가상 머신의 CPU 사양이 동일한 경우보다 CPU 사용률의 편차가 더 크다. 반면에 제안하는 할당 기법을 적용한 노드들의 CPU 사용률은 대부분 평균값에 근접하고 있어 CPU 사용률이 고르게 분포하고 있음을 알 수 있다. 이는 CPU의 고른 사용률과 그에 따른 부하 분산 측면에서 봤을 때, 다양한 사양을 갖는 가상 머신 환경에서도 제안하는 할당 기법이 효율적임을 나타낸다.

그림 9는 노드별 CPU 사용률의 표준 편차 그래프이다. 매치메이킹과 라운드 로빈의 경우 그림 7과 마찬가지로 표준 편차의 변화 폭이 상당히 크며, 10~46사이의 표준 편차를 갖고, 각각 28과 19의 높은 평균 표준편차를 갖는다. 반대로 제안하는 할당 기법의 경우, 0~15 사이의 표준 편차를 나타내며 평균 표준편차값은 9이다. 이처럼 제안하는 할당 기법의 표준 편차가 타 기법에 비해 상대적으로 낮은 이유는 CPU 사용률이 적은 노드에 우선적으로 가상 머신을 할당하기 때문이다. 이런 특성으로 인해 노드에 할당되는 가상 머신의 대수가 비대칭해져 노드 간 할당 가능한 CPU의 양 차이가 발생할 수 있으나, CPU 사용률의 고른 분포와 부하 분산 측면에서 볼 때 제안하는 할당 기법이 효과적이라 할 수 있다.

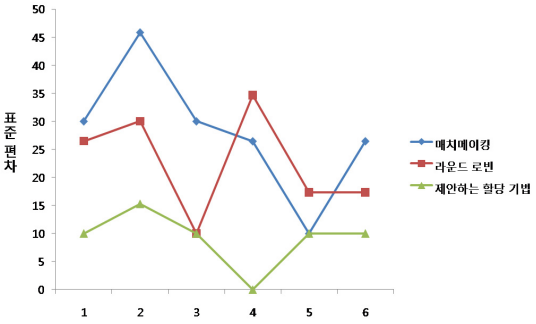

그림 9. 노드의 CPU 사용률 표준 편차 비교 (CPU 사양 다름)
Fig. 9. Standard Deviation Comparison of CPU Usage of a Node (different CPU specification)

V. 결 론

본 논문에서는 가상 머신의 할당 과정에 전체 노드의 일정 기간 동안 평균 CPU 사용률을 비교하여 가장 적은

CPU 사용률을 갖는 노드를 선택하도록 하였다. 평균 CPU 사용률은 CPU의 사용 여부에 따라 수시로 변하는 노드의 실시간 CPU 사용률 보다 변화의 폭이 완만하므로 가상 머신 할당에 필요한 노드 선택 시 보다 정확한 비교가 가능하였다. 이런 노드 선택 과정을 통해 타 가상 머신 할당 기법들이 갖는 특정 노드에만 CPU 사용률이 많거나, 적은 상태, 즉 노드 간 CPU 사용률의 불균형 및 부하 분산 문제를 해결하였다.

실험은 전체 가상 머신의 CPU 사양이 같은 경우와 다른 경우의 두 가지 시나리오로 진행하였다. 가상 머신들의 CPU 사양이 같은 경우는 CPU 사용률의 폭이 일정하지만, CPU 사양이 다양해지면 CPU 사용률의 폭 차이가 크게 발생한다. 두 경우에 매치메이킹과 라운드 로빈을 적용하여 가상 머신을 할당 하였을 때, 특정 노드의 CPU 사용률이 떨어지거나 반대로 높은 경우가 빈번하게 발생한 반면, 제안하는 할당 기법을 적용했을 때는 대부분의 노드에서 CPU 사용률이 고른 것을 확인했으며, 이를 통해 효율적인 부하 분산이 가능하였다.

향후 연구 내용으로는 좀 더 대규모의 테스트베드에서의 다양한 실험이 필요하며 사용자의 가상 머신 사용 패턴을 분석하고 예측하는 보다 정밀한 알고리즘의 연구가 필요하다. 또한, 가상 머신이 할당되고 난 후에 일어나는 노드의 과부하를 막기 위한 부하 분산 정책에 대한 연구가 필요하다.

참고문헌

[1] Mladen A. Vouk, "Cloud Computing - Issues, Research and Implementations," Journal of Computing and Information Technology, Vol. 16, pp. 235-246, 2008.

[2] 민옥기, 김학영, 남궁한, "클라우드 컴퓨팅 기술 동향," 전자통신동향분석, Vol. 24, No. 4, pp. 1-13, 2009. 8.

[3] Virtualization, http://en.wikipedia.org/wiki/Virtualization

[4] Xen, http://xen.org

[5] OpenNebula, http://www.opennebula.org

[6] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Capacity Leasing in Cloud Systems using the OpenNebula Engine," Workshop on Cloud Computing and its Applications 2008 (CCA08), Chicago, 2008. 10.

[7] Nimbus, http://www.nimbusproject.org

[8] Eucalyptus, http://www.eucalyptus.com

[9] Rajesh Raman, Miron Livny, and Marvin Solomon, "Matchmaking: Distributed Resource Management for High Throughput Computing," High Performance Distributed Computing 1998, The Seventh International Symposium, pp. 140~146, Chicago, 1998.

[10] Rasmus V. Rasmussen and Michael A. Trick, "Round robin scheduling-a survey," European Journal of Operational Research, pp. 617-636, 2008.

[11] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield, "Live migration of virtual machines," In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2, pp. 273~286, 2005.

[12] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," Lecture Notes In Computer Science, Vol. 5931, pp. 254~265, 2009.

배준성(Jun-Sung Bae)

2009년 대전대학교 정보통신공학과 졸업(학사)
2011년 대전대학교 정보통신공학과 졸업(석사)

※관심분야 : 클라우드컴퓨팅, 네트워크보안 등

이봉환(Bong-Hwan Lee)

1985년 서강대학교 전자공학과 졸업(학사)
1987년 연세대학교 대학원 전자공학과 졸업(석사)
1993년 Texas A&M 대학교 대학원 전기 및 컴퓨터 공학과 졸업(박사)
현재 대전대학교 정보통신공학과 교수
※관심분야 : 클라우드컴퓨팅, 유비쿼터스헬스케어, 네트워크보안 등