한국시뮬레이션학회 논문지
Vol. 21, No. 2, pp. 59-67 (2012. 6)

그리드 컴퓨팅에서 유효자원 동적 재배치 기반 작업 스케줄링 모델

김재권[1] · 이종식[1†]

Dynamic Available-Resource Reallocation based Job Scheduling Model in Grid Computing

Jaekwon Kim · JongSik Lee

ABSTRACT

A grid computing consists of the physical resources for processing one of the large-scale jobs. However, due to the recent trends of rapid growing data, the grid computing needs a parallel processing method to process the job. In general, each physical resource divides a requested large-scale task. And a processing time of the task varies with an efficiency and a distance of each resource. Even if some resource completes a job, the resource is standing by until every divided job is finished. When every resource finishes a processing, each resource starts a next job. Therefore, this paper proposes a dynamic resource reallocation scheduling model (DDRSM). DDRSM finds a waiting resource and reallocates an unfinished job with an efficiency and a distance of the resource. DDRSM is an efficient method for processing multiple large-scale jobs.

Key words : Grid Computing, Dynamic Reallocation, Grid Scheduling, DRRSM

요 약

그리드 컴퓨팅은 하나의 대용량 작업을 처리하도록 물리 자원을 구성하고 있지만 최근에는 데이터의 급속한 증가로 인해서 복수개의 작업을 처리하는 방법이 필요하다. 일반적으로 대용량 작업을 요청하면 각 물리 자원들이 작업을 분할하게 되며, 자원의 성능과 거리에 따라 처리 시간이 다르다. 성능에 따라 먼저 완료된 유효자원은 어떠한 작업도 하지 않으며, 모든 작업이 끝났을 경우에 다음 작업을 처리한다. 이에 본 논문에서는 먼저 처리가 완료된 자원을 다른 작업에 할당할 수 있는 동적 자원 재배치 스케줄링 모델(DRRSM: Dynamic Resource Reallocation Scheduling Model)을 제안한다. DRRSM은 먼저 처리가 완료된 자원을 다른 작업에 자원의 성능과 거리에 따라 작업을 재배치시키는 방법이다. DRRSM은 여러 개의 대용량 작업을 처리하는데 효과적이다.

주요어 : 그리드 컴퓨팅, 동적 재배치, 그리드 스케줄링, DRRSM

1. 서 론

그리드 컴퓨팅은 분산 병렬 컴퓨팅의 한 분야로서, 원거리 통신망으로 연결된 서로 다른 복수개의 물리 자원을 하나의 가상 서버로 구성하는 대용량 컴퓨팅 환경(Foster 등 2001a, 1999b)으로, 작은 소규모 컴퓨터로 작업을 분산시켜 수행하는 점에서 클러스터 컴퓨팅의 확장된 개념이라고 볼 수 있으나, 원거리 통신망에서 서로 다른 기종의 물리 컴퓨터를 연결하는 점으로 인해 여러 가지 표준 규약이 필요하며 글로버스(Globus)(Foster 등 1997)를 중심으로 그리드 컴퓨팅의 표준이 정립되고 있다.

이기종의 자원들로 구성된 그리드 컴퓨팅 환경에서 자원 활용과 대용량의 데이터를 처리하기 위해서는 자원의 상태 변화를 고려하지 않은 정적 작업 스케줄링 모델로는 한계가 있다. 이기종 자원을 어떻게 배치하고 작업 스케줄링 할 것인지는 NP-Complete Problem(Bokhari 1987)

*이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구 사업 지원을 받아 수행된 것임(2012002751).
접수일(2012년 3월 2일), 심사일(1차 : 2012년 5월 3일), 게재 확정일(2012년 5월 3일)
[1] 인하대학교 정보공학과
주 저 자 : 김재권
교신저자 : 이종식
E-mail; jslee@inha.ac.kr

으로 알려져 있으며 동적인 환경에서 작업 스케줄링하기 위한 방법으로 자원의 상태를 평가하거나 휴리스틱(heuristic)에 의한 방법에 대한 많은 연구가 진행되어졌다(Baghban 등 2008; Munir 등 2008). 즉, 그리드 컴퓨팅은 대용량 작업을 요청하게 되면 각 자원들의 구성에 따라 작업을 분할하게 되는데 자원들의 성능과 거리에 따라 처리 시간이 다르다.

기존의 연구 내용은 복수개의 대용량 작업을 요청하게 되면 작업의 크기나 우선순위에 따라 그리드 환경으로 구성된 자원들을 배치시키는데 연구가 진행되고 있다. 하지만 성능에 따라 먼저 완료된 유효자원은 아무런 작업을 하지 않으며 모든 작업이 끝났을 경우에 다음 작업을 처리한다. 따라서 먼저 완료된 유효자원의 활용에 대한 연구는 현재까지 미비하다. 대용량 작업에서 다른 자원보다 먼저 처리가 완료된 자원을 또 다른 대용량 작업에 재배치시키게 되면 작업 처리 속도를 높일 수 있으며 자원의 활용도를 향상 시킬 수 있다.

자원의 재배치는 그리드의 가상화 환경(Jon 외 2002)에서 네트워크에 따라 자원을 재배치시키는 방법으로서 그리드 컴퓨팅의 도입 이전에 시뮬레이션을 통한 검증이 필요하다.

이에 본 논문에서는 그리드 환경에서 먼저 처리가 완료된 유효자원을 다른 작업에 할당할 수 있는 동적 자원 재배치 스케줄링 모델(DRRSM: Dynamic Resource Reallocation Scheduling Model)을 제안한다. 제안하는 DRRSM은 먼저 처리가 완료된 유효자원을 다른 작업으로 재배치시키는 방법이다. 다른 작업에 재배치시키기 위해 작업의 예상 처리시간과 자원의 성능, 네트워크의 거리에 따라 재배치시킨다. 이에 자원을 동적 재배치를 위한 DRRSM의 구성도와 알고리즘을 제공하며, DRRSM의 성능을 측정하기 위해 DEVS 형식론(Zeigler 외 1996)을 적용한 그리드 컴퓨팅 환경을 구성하고 실험 하여 제안하는 모델의 효율성을 입증한다.

본 논문의 구성은 2장 관련연구를 통해 그리드 컴퓨팅의 작업 처리 방법에 대해 설명을 하며, 3장에서는 제안하는 DRRSM에 대해 기술한다. 4장에서는 DRRSM의 검증을 위한 실험 방법과 결과를 기술하며, 5장에서는 결론을 기술한다.

2. 관련연구

2.1 가상화(Virtualization)

가상화란 물리적으로 분리된 자원을 하나의 논리적으로 통합하거나 혹은 반대로 하나의 자원을 논리적으로 분할하여 자원을 효율적으로 사용하는 기술이다. 따라서 가상화 기술은 그리드 컴퓨팅과 밀접한 관련을 가지고 있다. 가상화 기술은 현재 그리드 컴퓨팅, 유틸리티 컴퓨팅, 클라우드 컴퓨팅과 함께 분산 환경의 핵심 기술로 주목받고 있으며 많은 연구가 진행되고 있다.

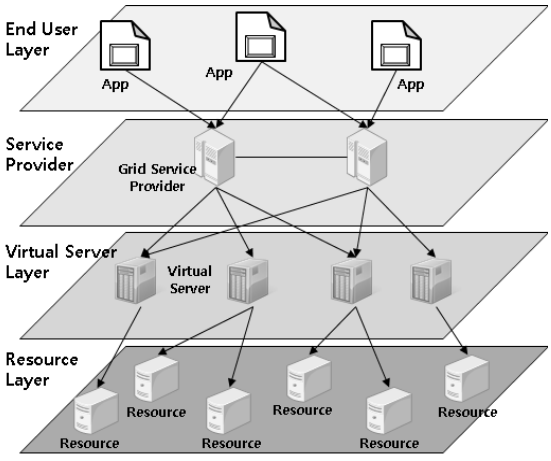

그림 1. 그리드 컴퓨팅의 가상화 환경

가상화는 여러 물리 자원을 하나의 큰 서버로 통합하는 방법으로 이기종의 컴퓨팅, 스토리지 및 네트워크 자원을 통합하여 자원을 효율적으로 사용할 수 있다. 그림 1은 그리드 컴퓨팅의 가상화 환경이다.

사용자는 그리드 서비스를 이용하며, 서비스 프로바이더는 서비스 요청에 대해 가용성이 높은 가상 서버를 선택한다. 그리고 가상서버는 물리 자원 리소스를 가지고 있으며 물리 자원을 이용하여 그리드 작업을 처리한다. 가상화된 시스템은 자원을 필요로 하는 워크로드에 적절한 시간에 우선순위에 따라 동적으로 자원을 할당함으로서 워크로드 자원의 집중을 방지하며 자원 낭비를 줄이며 서비스 최적화를 실현한다(장성호 2006). 따라서 가상화 기술을 이용하여 그리드 환경에서 여러 자원을 결합하고 관리할 수 있으며, 사용자에게 효율적으로 자원을 제공할 수 있다.

본 논문에서는 그리드 컴퓨팅의 가상화를 이용하여 다수의 가상화 환경과 다수의 물리 자원을 구성하여 동적 재배치 모델에 대한 시뮬레이션을 한다.

2.2 동적 자원 재배치

가상화 환경에서는 가상머신의 자원 부족상황(Hotspot)을 탐지하고 이를 해결하기 위한 연구가 진행 중이다.

Sandpipier(Wood 등 2007)는 가상머신의 배치를 위해 Volume이라는 CPU와 RAM, 네트워크 상태를 이용하여 이 값을 기준으로 가상머신을 이동 및 교체를 시킨다. Volume은 CPU라는 1차원 적인 연구에 집중이 되어 있기 때문에 지리적으로 분산되어있는 그리드 환경에 적합하지 않다. Real Time Resource Reallocation for Better Resource Utilisation(Sijin 등 2011)은 가상머신의 재배치를 위해서 실시간으로 메모리와 CPU의 사용량을 계산하여 가상머신의 부하량을 측정한 후, 실시간 재배치를 한다. 이는 여러 자원에 대한 부하량을 줄일 수 있지만, 실시간 재배치로 인해 네트워크의 트래픽이 상당히 높아지기 때문에, 거리적으로 분산되어 있는 그리드 환경에 적합하지 않다. 휴리스틱 기반의 재배치 방법(Yves 등 2010)은 휴리스틱을 이용하여 작업의 크기와 자원의 상태, 예상 처리시간을 통해 자원의 재배치를 적절히 하지만, 이는 특정 데이터를 처리해야하는 상황에 의존하기 때문에 예상치 못한 작업이 발생할 경우 부하가 높아진다.

위와 같은 가상머신 이주 방법은 하나의 물리 자원에서 분산된 여러 가상머신에서 적합하기 때문에, 그리드 컴퓨팅에서는 고려되기 힘들다.

그리드 컴퓨팅의 자원 관리 스케줄링은 동적 스케줄링 방식인 Global 스케줄링을 이용하며 동적으로 자원을 할당하게 된다. 하지만, 이러한 Global 스케줄링은 자원의 할당을 고려할 뿐, 작업 처리 이후에 대해서는 고려하지 않는다. 따라서 작업처리 이후의 자원을 관리하기 위한 동적 자원의 재배치가 필요하다.

동적 자원 재배치를 위해서는 하나의 물리서버에 여러 가상머신으로 구성된 상태에서의 가상머신의 이동이 아닌, 여러 개의 물리서버가 하나의 가상머신을 이용하는 환경(장성호 2006)에서, 가상머신을 구성하는 물리 자원의 네트워크상의 유동적인 이동이 필요하다.

따라서 본 논문에서는 기존의 가상머신의 재배치 방법을 고려하여 하나의 가상머신에서 여러 대의 유효자원을 재배치시키기 위한 새로운 방법과 패러다임이 필요하다.

3. 동적 자원 재배치 스케줄링 모델

본 논문에서는 그리드 컴퓨팅에서 복수개의 작업을 효과적으로 처리할 수 있는 동적 자원 재배치 스케줄링 모델(DRRSM: Dynamic Resource Reallocation Scheduling Model)을 제안한다.

DRRSM을 제안하기 위해 첫 번째로, 가상머신으로 구성된 그리드 컴퓨팅 환경을 기술하며, 두 번째로, DRRSM의 구조에 대해 기술한다. 마지막으로 동적 재배치 알고리즘에 대해 기술 한다.

3.1 그리드 환경의 동적 자원 재배치

그리드 환경의 동적 자원 재배치는 하나의 가상머신이 여러 대의 물리적 자원을 소유하고 있으며, 각 가상머신마다 하나의 대량의 작업을 처리한다. 그리드 환경의 작업 처리 방법은 그림 2와 같다.

가상머신의 작업 관리는 기본적으로 사용자 요청에 대한 작업을 작업 Queue에 저장하게 되며 Queue에 있는 작업을 단위작업으로 구성한다. 단위작업은 Map-Reduce 방식으로 대량의 작업을 분할하여 구성한다. 단위작업 관리 모듈은 이러한 분할 작업들을 가상서버에 구성된 각 물리 자원으로 전송한 후, 물리자원은 단위 작업을 처리한다. 단위 작업의 처리가 완료되면 다시 가상서버로 단위 작업 처리 정보를 전송하며 단위 작업 관리 모듈에서는 단위 작업 처리 정보를 합병 한다.

가상머신에서 단위 작업 처리 정보를 합병하여 작업을 처리를 완료를 하게 되는 방식으로서, 여기서 처리가 안된 단위 작업이 2개가 있으며, 현재의 물리 자원은 3개가

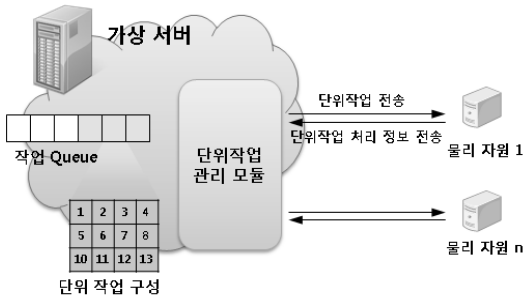

그림 2. 가상머신의 단위작업 관리자의 작업할당

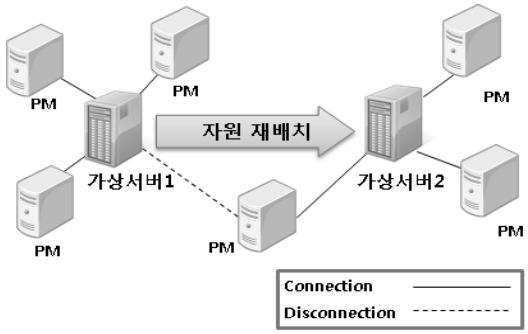

그림 3. 동적 자원 재배치 방법

휴식 상태면, 단위 작업 관리 모듈은 물리자원 2개에게 작업을 전송하게 되며, 작업을 받지 못한 유효 자원에 대해서는 동적 자원 재배치를 한다. 유효 자원이 탐색이 되면 이를 다른 가상 서버에 등록을 하게 되는데 동적 자원 재배치의 방법은 그림 3과 같다.

가상머신 1에서는 현재의 작업 태스크가 2개가 남은 상태이며 2개의 자원에 할당하여 1개의 자원은 유효 상태이다. 그리고 가상머신 2는 현재의 작업 태스크가 아직 남아있는 상태이다. 여기서 동적 자원 재배치 방법은 가상머신 2의 유효 자원을 가상머신 1에 할당하는 방법이다. 가상머신 2의 작업 처리에는 지장이 없으며, 유효 상태의 자원을 태스크가 많은 자원에게 전달함으로서, 복수개의 작업에 대해 빠른 처리가 가능하다.

3.2 DRRSM 구조

앞서 그리드 환경의 동적 자원 재배치 방법에 대해 기술하였으며 이를 위한 DRRSM의 시스템 구조에 대해 설명한다. DRRSM의 시스템 구조는 그림 4와 같다.

DRRSM은 서비스 사용자가 이용하는 Grid Application을 통해 접속을 하면, 사용자의 요구사항에 따라 작업을 생성하는 Job Generator, 가상서버에 대용량 작업 처리를 요구하는 Coordinator 그리고 유효자원을 탐지하는 Resource Detector와 가상서버와 물리 자원의 상태를 식별하는 Monitor와 물리 자원을 재배치시키는 Resource Reallocation으로 구성되어 있다. 또한 Grid Site는 가상서버와 물리서버로 구성되어 각 가상서버에는 복수개의 물리서버를 포함한다.

- **Job Generator**

데이터 추출모듈로부터 구성된 대용량 데이터를 Job Generation Module에서는 일개의 작업 단위로 구성하여 그리드 자원이 처리할 수 있는 순서대로 작업을 비치한다.

- **Coordinator**

중계부 모듈인 Coordinator는 Job Generator로부터 생성된 작업들을 정적 혹은 동적 작업 스케줄링 모델을 이용하여 가상 서버에 대용량 Job을 지정하게 된다. 또한 가상서버에서 작업이 완료되었을 경우, 이를 다시 받아 사용자에게 정보를 제공하는 역할을 한다.

- **Resource Detector**

유효 자원 검색 모듈인 Resource Detector는 가상 서버로부터 유효자원이 탐지 되면, 다른 가상 서버에 현재의 작업 크기 및 위치정보, 구성되어있는 물리 자원의 상태에 대해 요청을 한다.

- **Monitor**

Monitor는 현재의 가상서버와 물리 자원의 상태를 감지하는 모듈로서, Resource Detector로부터 요청 되어진 자원들의 데이터들을 전달 받고, Resource Reallocation이 처리가 가능한 자료구조를 구성한 후 전달한다.

- **Resource Reallocation**

Monitor로부터 유효자원의 정보와 가상서버 및 물리 자원의 상태에 대해 데이터를 받게 되면 Resource Reallocation에서는 가상 서버에 따른 물리 자원을 알고리즘을 통하여 재배치시킨다. 여기서 Reallocation Algorithm은 현재의 가상머신의 Job Size와 물리 자원들의 위치 및 거리, 그리고 물리 자원의 성능을 고려하여 동적으로 자원을 재배치한다.

3.3 Dynamic Resource Reallocation 알고리즘

위와 같이 DRRSM의 구조에 대해 설명 했으며, 본 장

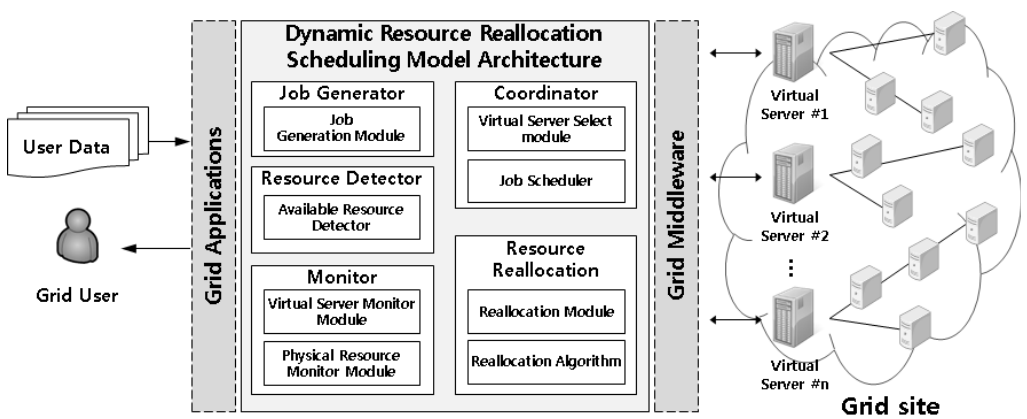

그림 4. DRRSM 구조

표 1. 동적 자원 재배치 알고리즘의 매개변수

Property Name	Description	Value Type	Value Range
VS	가상 서버	Integer	1 ~ VS #N
PM	물리 자원	Integer	1 ~ PM #N
x	서버와 자원의 X좌표	double	km
y	서버와 자원의 Y좌표	double	km
p	물리 자원의 작업 처리 속도	double	0 ~ Processing Time
job_size	가상 서버의 남은 작업량	integer	0 ~ Job_Size
Avail_Score	자원 가용 점수	double	-

에서는 동적 자원을 재배치시키기 위한 알고리즘을 제안한다. 그리드 컴퓨팅의 특성은 지리적으로 분산되어있는 특징을 가지고 있기 때문에, 물리 자원의 성능뿐만 아니라 물리 서버와 가상서버 간의 네트워크상의 거리에 대해 처리 효율성이 바뀌게 된다. 따라서 작업이 완료되는 시점과 거리에 대한 판단이 필요하므로, 본 논문에서 제안하는 알고리즘은 각 가상서버의 재의 작업 처리 정보와 가상서버와 물리자원 사이의 네트워크 거리 그리고 물리자원의 성능을 이용하여 동적 자원 재배치 알고리즘을 설계한다.

동적 자원 재배치 알고리즘의 매개변수는 표 1과 같다.

위와 같이 매개변수를 정의하였으며, Avail_Score은 각 가상 서버마다의 동적 자원을 제공하기 위해서 어느 가상서버로 전송할지에 대한 결과 수치이다. 즉 Avail_Score을 이용하여 자원 재배치를 하게 될 가상 서버를 지정한다. 각 가상서버의 Avail_Score을 구하기 위한 공식은 식 (1)과 같다.

$$AvailScore = VS_{Queuesize} + (VS_{Jobsize} * 0.01) + (1/(Distancce(VS_X, PM_X, VS_Y, PM_Y) * 0.1)) \quad (1)$$

$$Distancce(VS_X, PM_X, VS_Y, PM_Y) = \sqrt{(VS_X - PM_X)^2 + (VS_Y - PM_Y)^2}$$

Avail_Score은 현재의 가상서버의 작업 크기(VS_{job_size}), 현재 가상서버의 위치와 유효 자원의 위치 사이의 거리(Distance(VS, PM) 그리고 유효 자원의 성능(PM_P)에 따라 계산을 한다. 가상 서버와 유효 자원의 거리를 구하기 위해서 유클리드 거리(Euclidean Distance)(Pang 등 2007)를 사용한다.

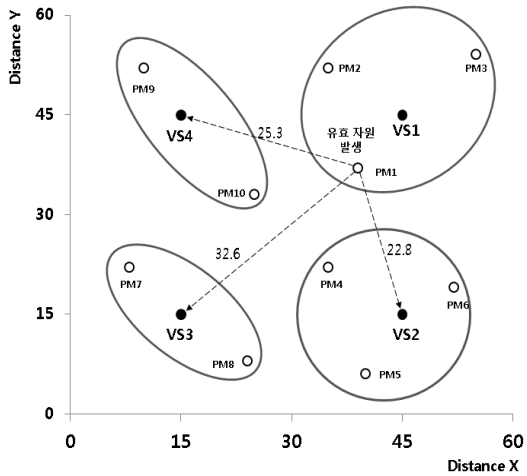

그림 5. 자원 재배치 거리 계산

가상서버의 작업 크기는 높을수록 유효 자원의 재배치의 확률이 높아지게 되며, 거리와 처리시간은 낮을수록 자원 재배치의 확률이 높아지게 된다. 이는 처리시간은 거리에 따른 처리시간이 다르기 때문에 이를 같이 고려해야 한다. 이와 같이 가장 높은 Avail_Score에게 유효 자원을 할당한다.

가상 서버와 유효 자원간의 거리를 구하게 되며, 이는 유효자원을 다른 가상서버로 이동시에 가장 효과적으로 스케줄링을 하기 위해서다.

예를 들어 그림 5와 같이 가로와 세로가 각각 60km이라고 가정을 하고 4개의 가상서버와 각 가상서버를 구성하는 물리자원이 있고, 가상서버1의 물리자원1이 유효 자원이 발생을 하면 물리자원1과 VS1을 제외한 모든 가상서버와의 거리를 계산한다. 여기서 가장 가까운 VS2에 유효 자원이 재배치된다.

위와 같이 거리에 따르는 자원 재배치 방법과 Avail_Score을 구하는 공식을 설명 했으며, 다음으로 이를 처리하기 위한 알고리즘은 표 2와 같다.

각 가상서버는 좌표 x, y인 위치 값과 현재의 작업 크기 정보를 가지게 된다. 그리고 유효자원은 좌표 x, y와 유효 자원의 작업 처리 속도를 가진다.

작업 처리 중 유효자원이 발생할 경우, 각 가상서버는 Avail_Score 값을 구하게 된다. Avail_Score은 현재의 작업 사이즈와 유효자원과 가상머신과의 거리 그리고 유효자원의 처리 시간을 통해 계산을 한다. 유효자원과 가상머신과의 거리는 유클리안 거리를 사용하여 구하게 된다. 각 가상서버마다의 Avail_Score값이 구해지면 이 중에서

표 2. 동적 자원 재배치 알고리즘

```
Initial State
VS(distanceX, distanceY, Queue_Size, Job_Size)[i] ← 1 to Number of Virtual Server
PM(distanceX, distanceY, Processing_time) ← Available Physical Machine

Function Resource_Reallocation_Algorithm(VS, PM) return the Max Virtual Server of i
if occurrence of PM
  for each VS in 1 to i do
    Avail_Score[i] ← VS(Queue_Size)[i] + (VS(Job_Size)[i]*0.01) +
                     (Euclidean(VS(distanceX, distanceY)[i], PM(distanceX, distanceY))* 0.1)
  Max ← Max(Volume 1 to i)
    for 1 to i do
      if Max equal Volume[i] then
      return Max[i]
return Max[i]

Function Euclidean(VS(distanceX, distanceY)[i], PM(distanceX, distanceY)[j]) return distance of between VS[i] and PM
  Distance ← Sqrt(((VS(distanceX)[i] - PM(distanceX))^2)+((VS(distanceY)[i] - PM(distanceY))^2))
return Distance

Function Max(1 to i) return the max of Volume[i]
m ← -∞
for 1 to i do
  m ← Max(u, i)
  if m ≥ i then return m
return m
```

가장 높은 Avail_Score 값을 구한다. 가장 높은 가상서버의 Avail_Score값이 구해지면 이에 해당하는 가상서버에 유효자원을 할당하게 된다.

4. 실험 및 결과

본 논문에서 제안하는 DRRSM의 효과를 입증하기 위해서 가상화 그리드 환경을 구성하고, 성능을 측정하기 위해 서비스 처리 완료 시간(Execution Time), 자원 활용율(Utilization)을 측정한다.

4.1 실험 환경

가상화 그리드 환경의 구성을 위해 DEVS 형식론을 이용하여 그림 6과 같이 구성하였다. 총 6개의 가상머신과 15개의 물리 자원이 있으며, 각 자원들의 위치에 대해서는 그림 6과 같다. DEVS의 Atomic 모델 중 Generator는 작업을 생성하는 모듈이다. Coordinator는 해당 작업에 대해 VS(가상 서버)에 작업을 할당하는 모듈이다. 그리고 VS는 각각의 PM(물리 자원)을 가지고 있으며, PM은 실제 작업을 처리하는 모듈이다. 초기의 VS와 PM의 위치는 그림 7과 같은 위치에서 시작을 한다. Detector는 유효자원이 생겼는지 확인 하는 모듈로서, 현재의 PM의 상태를 Detector가 감지를 한다. Monitoring은 감지가 되었을 경우, 모든 PM의 상태를 체크를 하게 되며, 해당 정보를 Reallocation에 전달한다. Reallocation에서는 모든 자원을 재배치 알고리즘을 통해 자원의 재배치가 필요한

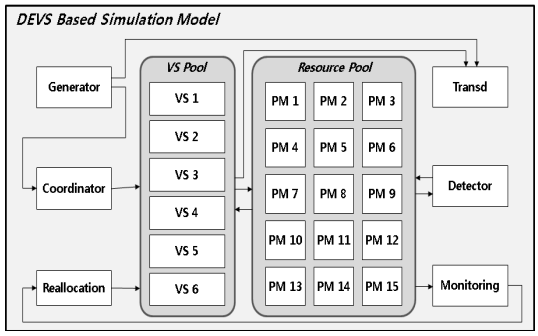

그림 6. 실험 환경

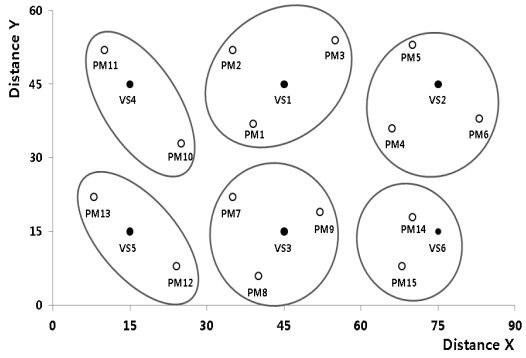

그림 7. 자원 위치 초기 값

VS에 정보를 전달하게 된다. 이후 VS는 재배치를 하게 된다. 마지막으로 Transd는 작업 처리율에 대해 계산을 한다.

표 3. 실험 환경

PM#	P_time	PM#	P_time	PM#	P_time
1 (VS1)	10	6 (VS2)	20	11 (VS4)	10
2 (VS1)	20	7 (VS3)	10	12 (VS5)	20
3 (VS1)	30	8 (VS3)	20	13 (VS5)	30
4 (VS2)	30	9 (VS3)	20	14 (VS6)	10
5 (VS2)	10	10 (VS4)	30	15 (VS6)	20

실험을 위한 작업의 개수 60개를 구성하였으며 15개의 물리 자원 처리시간은 표 3과 같이 구성하였다.

시뮬레이션을 위한 물리자원 처리시간은 P_time + (가상서버 거리-물리서버 거리)이다. 또한, 초기 값으로 가상서버에 등록된 물리 자원은 괄호로 표시했다.

실험을 위한 스케줄링 모델은 총 2가지로 작업의 실행 순서 등을 결정하는 방법인 라운드 로빈 토너먼트 스케줄링(Scheduling Round Robin Tournaments : SRRT)(Rasmus외, 2008)과 실행시간과 대기시간에 따른 스케줄링 기법인 위한 반응시간 최적화된 자원 관리 알고리즘(Response time Optimization-based Resource Management algorithm: RORM)(Assuncao 외 2009)으로 구성하였고, 각 모델에 본 논문에서 제안하는 DRRSM을 추가하였을 경우와 그렇지 않았을 경우를 비교한다.

4.2 서비스 처리 완료 시간

서비스 처리 완료 시간(Execution Time)은 그리드 작업 요청별 처리 시간을 의미한다. 그림 8의 X축은 서비스 처리 완료 시간이며, Y축은 시뮬레이션 완료 시간이다.

직업처리 완료시간은 사용자가 요청한 서비스에 대해 처리를 하는 시간을 의미하므로 사용자에게 QoS를 제공하기 위해서는 짧은 시간 내에 많은 작업을 처리해야한다.

SRRT는 60개의 서비스에 대한 최종 처리 시간은 6193이며, RORM은 5255이다. 그리고 SRRT With DRRSM은 3690, RORM With DRRSM은 3731이다.

작업 처리 개수가 40개 까지는 모든 모델이 다 비슷한 서비스 처리를 진행하지만 DRRSM이 도입되지 않은 2가지 실험군은 시간이 지날수록 처리 시간이 늘어난다. 기존의 모델들은 서비스처리에 중점을 두지만, DRRSM은 서비스 처리 이후의 유효 자원에 대해 처리하기 때문에 기존의 방식에서 좀 더 효과적으로 서비스를 처리할 수 있다. 따라서 DRRSM이 기존의 스케줄링 방식에 도움을 줄 수 있는 것을 나타내고 있다.

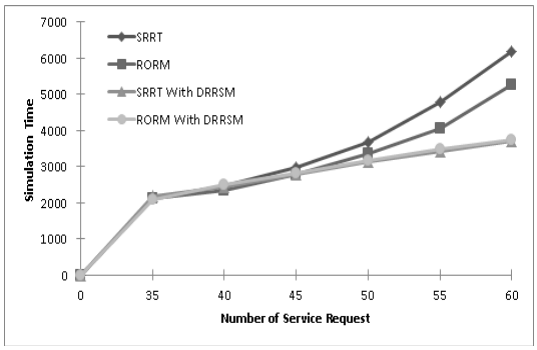

그림 8. 서비스 처리 완료 시간

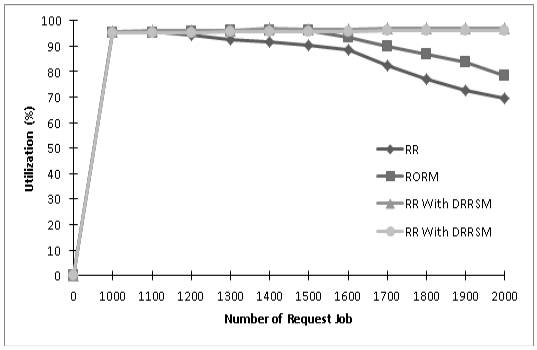

그림 9. 자원 활용율

4.3 자원 활용율

자원 활용율(Utilization)은 단위 작업이 완료된 시점에서 물리 자원의 활용율을 나타내는 척도이다. 자원 활용율의 식은 식 (2)와 같다.

$$Utilization(\%) = \frac{((\sum_{i=1}^{n} R_n * JS_n) * PT_n) / ST}{n} * 100 \qquad (2)$$

n은 총 자원의 개수를 의미하며, Rn은 n번째 자원을 의미하며 JSn은 n번째 자원의 작업처리 개수를 나타낸다. 그리고 PTn은 n번째 자원의 처리 시간을 의미하며 ST는 전체 시뮬레이션 시간을 의미한다. 즉, 전체 시뮬레이션 시간동안 각 자원들이 작업을 처리하는 총 처리 시간의 비율을 백분율로 나타낸 값이 Utilization이며 측정결과는 그림 9와 같다.

모든 대조군들은 1200개의 단위작업을 처리하기 전까지는 비슷한 활용율을 나타내지만, 이후에 SRRT와 RORM은 현저히 활용 율이 떨어지고 있다. DRRSM을

도입한 대조군은 활용율의 감소가 줄지 않는다. 이는 작업 요청이 끝나갈 무렵 SRRT와 RORM은 유효자원에 대한 처리를 하지 않기 때문에 자원에 대한 활용율이 현저히 떨어지며, DRRSM은 유효자원에 대한 관리를 하기 때문에, 활용율의 감소가 일어나지 않는다. 따라서 DRRSM은 그리드 컴퓨팅의 대용량 작업 처리 및 관리에 효과적인 것으로 나타난다.

5. 결 론

대용량의 데이터의 증가와 이를 분석하고 처리하기위해서 많은 연구 중에 있으며 그 중에 그리드 컴퓨팅은 분산 처리 환경에서 대용량의 데이터의 분석 및 처리에 주목을 받고 있다.

기존의 그리드 컴퓨팅은 대용량의 작업을 분산하여 물리 자원에 할당하는 연구만 진행되어 왔으며, 물리 자원에서 작업을 처리 한 후, 그 이후의 스케줄링 방법에 대한 연구는 진무했다. 이에 작업을 처리 한 후의 스케줄링 방법이 필요하며 이에 본 논문은 그리드 환경에서 먼저 처리가 완료된 자원을 다른 작업에 할당할 수 있는 동적 자원 재배치 스케줄링 모델(DRRSM: Dynamic Resource Reallocation Scheduling Model)을 제안한다.

제안하는 DRRSM의 구조적인 설명과 유효 자원의 관리를 위한 알고리즘에 대해 기술 하였다. DRRSM의 성능을 검증하기 위해 그리드 컴퓨팅 서비스의 모델링과 시뮬레이션을 했다. 실험결과, 기존의 자원할당 스케줄링 모델에 DRRSM을 적용하게 되면, 유효 자원에 대한 처리가 가능하기 때문에, 적용을 하지 않은 모델보다 높은 활용 율을 보여준다. 따라서 DRRSM은 그리그 컴퓨팅으 도입에 우수한 성능을 나타내는 것을 확인 했으며, 그리드 컴퓨팅 서비스제공에 적합하다고 할 수 있다.

향후 연구로는 동적 재배치의 횟수를 줄일 수 있는, 히스토리 기반의 스케줄링 모델을 통해 자원 활용도를 높이는 방법을 연구할 계획이다.

참 고 문 헌

1. 장성호 (2006), 가상화를 이용한 위탁형 그리드 서비스 거래망 모델, 석사학위논문, 인하대학교, pp. 3-4.
2. Assuncao, M. D., Costanzo, A. (2009), “Evaluating the Cost-Benefit of Using Cloud Computing to Extend the Capacity of Clusters.”, *In: 18th ACM International Symposium on High Performance Distributed Computing*, New York, pp. 141-150.
3. Baghban, H., Rahmani, A.M. (2008), “A Heuristic on Job Scheduling in Grid Computing Environment”, *Proceedings of the 2008 Seventh International Conference on Grid and Cooperative Computing, Shenzhen*, pp. 141-146.
4. B.P. Zeigler, et al. (1996), DEVS Framework for Modeling, Simulation, Analysis and Design of Hybrid Systems in Hybrid II, Leture Notes in CS, Springer-Verlag, Berlin, pp. 529-551.
5. Foster, I. C. Kesselman, S. Tuecke. (2001), “The Anatomy of the Gird: Enabling Scalable Virtual Organizations”, *International Journal of High Performance Computing Applications,* Vol. 15, No. 3, pp. 200-222.
6. Foster, I. and C. Kesselman. (1999), “The Grid: Blueprint for a New Computing Infrastructure“, Morgan Kaufmann Publishers.
7. Foster, I. and C. Kesselan, (1997), “ Globus: A Metacomputing Infrastructure Toolkit”, *International Journal of High Performance Computing Applications*, Vol. 11, No. 2, pp. 115-128
8. Jon B. Weissman and Byoung-Dai Lee(2002), “The Virtual Service Grid: An Architecture for Delivering High-End Network Services”, *Concurrency: Practice and Experience*, Vol. 14, No. 4, pp. 287-319.
9. Maheswaran, M., Braun, T.D., Siegel, H.J. (1999), “Heterogeneous Distributed Computing”, *Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster, editor, John Wiley & Sons, Vol. 8, pp. 679-690.
10. Munir, E.U., Li, J., Shi, S., Zou, Z., Yang, D. (2008), “MaxStd: A Task Scheduling Heuristic for Heterogeneous Computing Environment”, *Information Technology*, Vol. 7, pp. 679-683.
11. Pang Ning Tan., Michel Steinbach and Vinpin Kumar (2007), “Introducton to Data Mining”, *Addison Wesley*, pp.66-69.
12. Rasmus V. Rasmussen and Michael A. Trick, “Round robin scheduling-a survey”, *European Journal of Operational Research*, pp. 617-636, 2008.
13. S. H. Bokhari. (1987), Assignment Problems in Parallel and Distributed Computing, Kluwer Academic Publisher.
14. Wood, T., Shenoy, P., Venkataramani, A., and Yousif, M.(2007), “Black-box and gray-box strategies for virtual machine migration”, *Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation*, pp. 229-242.
15. Sijin, H., Li, P., and Yike, G.(2011), “Real Time Elastic Cloud Management for Limited Resource”, *Cloud 2011*

IEEE International Conference on Computing(CLOUD), Washington DC, pp. 622-629.
16. Yves, C., Ghislain, C. and Frederic, D.(2011), "Evaluation for Reallocation Heuristics for Moldable Tasks in Computational Grids", *In 9th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2011)*, Australia, pp. 1-10.

김 재 권 (jaekwonkorea@naver.com)

2011 가천의과학대학교 정보처리과 학사
2011～현재 인하대학교 컴퓨터정보공학과 석사과정

관심분야 : 클라우드 컴퓨팅, 그리드 컴퓨팅, 분산처리, 모델링&시뮬레이션

이 종 식 (jslee@inha.ac.kr)

1993 인하대학교 전자공학과 학사
1995 인하대학교 전자공학과 석사
2001 애리조나대 전기・컴퓨터공학과 박사
2001～2002 캘리포니아 주립대학교 전기・컴퓨터공학과 전임강사
2002～2003 클리블랜드 주립대학교 전기・컴퓨터공학과 조교수
2003～현재 인하대학교 컴퓨터정보공학부 교수

관심분야 : 소프트웨어공학, 모델링&시뮬레이션