韓國 컴퓨터情報學會 論文誌
第12卷 第5號, 2007. 11.

2007-12-5-1-6

계산 그리드를 위한 퍼지로직 기반의 그리드 작업 스케줄링 모델

박 량 재*, 장 성 호*, 조 규 철*, 이 종 식*

Fuzzy Logic-based Grid Job Scheduling Model for omputational Grid

Park Yang Jae*, Jang Sung Ho*, Cho Kyu Cheol*, Lee Jong Sik*

요 약

계산 그리드 컴퓨팅은 수많은 컴퓨팅 자원들을 이용하여, 슈퍼 컴퓨팅이나 이전의 분산 컴퓨팅으로 해결 할 수 없는 대용량의 연산 문제를 해결한다. 계산 그리드 컴퓨팅 환경에서의 자원은 이 기종으로 구성되어, 효율적인 작업 처리를 위해서는 스케줄링 기법이 필요하다. 본 논문에서는 계산 그리드에서 효율적인 작업 스케줄링을 위하여, 퍼지로직 기반의 그리드 작업 스케줄링 모델을 제안한다. 퍼지로직 기반의 그리드 작업 스케줄링 모델은 퍼지로직을 이용하여 자원의 효율성을 평가하며, 평가된 기반으로 그룹을 구성하여 작업을 할당하는 모델이다. 우리는 DEVS 모델링 & 시뮬레이션 환경에서 시뮬레이션 모델을 구성하고 Random 스케줄링과 MCT 스케줄링 모델과의 비교 실험을 통하여 제안된 퍼지로직 기반의 그리드 작업 스케줄링 모델이 작업완료시간, 작업손실, 통신량을 개선함으로써 더욱 더 안정적이고 빠른 작업 처리 서비스를 그리드 사용자에게 제공할 수 있다는 사실을 증명하였다.

Abstract

This paper deals with grid job allocation and grid resource scheduling to provide a stable and quicker job processing service to grid users. In this paper, we proposed a fuzzy logic-based grid job scheduling model for an effective job scheduling in computational grid environment. The fuzzy logic-based grid job scheduling model measures resource efficiency of all grid resources by a fuzzy logic system based on diverse input parameters like CPU speed and network latency and divides resources into several groups by resource efficiency. And, the model allocates jobs to resources of a group with the highest resource efficiency. For performance evaluation, we implemented the fuzzy logic-based grid job scheduling model on the DEVS modeling and simulation environment and measured reduction rates of turnaround time, job loss, and communication messages in comparison with existing job scheduling models such as the random scheduling model and the MCT(Minimum Completion time) model. Experiment results that the proposed model is useful to improve the QoS of the grid job processing service.

▸ Keyword : 계산 그리드(Computational Grid), 작업 스케줄링(Job Scheduling), 퍼지로직(Fuzzy Logic)

• 제1저자 : 박량재
• 접수일 : 2007. 9.10, 심사일 : 2007. 9.25, 심사완료일 : 2007. 10.10.
* 인하대학교 정보공학과

Ⅰ. 서 론

최근 기존의 슈퍼컴퓨팅이나 배타적인 도메인 내 에서의 클러스터 컴퓨팅으로 해결 할 수 없는 대용량의 연산문제들이 증가함에 따라 네트워크상에 분산된 자원들을 모아 가상 컴퓨터를 구현하는 그리드 컴퓨팅이 주목받고 있다. 네트워크 통신 장비 및 기술의 발전으로 인해 그리드 컴퓨팅은 지리적으로 분산되어 네트워크로 연결된 대규모의 이기종 자원들을 동적으로 관리하고 활용할 수 있게 되었다 즉, 그리드는 능동적으로 자원을 결합하여, 대규모의 자원이 소요되는 분산 응용 프로그램을 수행 할 수 있는 가상 시스템을 그리드 사용자들에게 제공한다[1]. 그리드 컴퓨팅은 응용분야에 따라 많은 자원을 연결하여 대용량 연산 문제를 해결할 수 있는 계산 그리드(Computational Grid)[2], 분산처리를 필요로 하는 어플리케이션을 위한 액세스 그리드(AccessGrid), 원격지의 분산된 자료들을 통합하여 분석 할 수 있게 해주는 데이터 그리드(Data Grid)로 분류된다. 이 중 계산 그리드는 다양한 연구진 및 기업체에 의해 가장 많은 관련 프로젝트가 진행 중이다.

계산 그리드의 가장 큰 목적은 그리드 사용자로부터 요청받은 대용량의 데이터 및 응용 프로그램의 처리이다. 따라서 효과적인 안정적인 작업 처리를 위해서는 이 기종이라는 그리드 자원의 특징을 고려하여 자원 평가를 기반으로 한 작업 스케줄링 기법이 필요하다[3]. 일반적인 멀티프로세서 환경이나 배타적인 도메인 내의 클러스터 컴퓨팅 환경에서의 효율적인 작업 처리를 위한 스케줄링 기법은 오래 전부터 많은 연구가 이루어져 왔고 많은 기법들이 제시되었다. 그러나 이러한 기법들은 동기종의 정적인 자원에 대해 최적화되어 있어, 이기종의 동적인 자원으로 이루어진 그리드 컴퓨팅의 스케줄링과는 차이가 있다. 또한, 기존의 그리드 스케줄링 모델은 작업의 실행시간 단축이나 실행비용 감축에 초점을 맞추어 작업을 할당함으로써 작업 손실 및 지연이 발생한 가능성이 존재함으로써 안정적인 작업 처리를 제공하기 어렵다.

본 논문에서는 계산 그리드 상의 효율적이고 안정적인 작업 처리와 자원 사용을 목적으로 하는 퍼지로직 기반의 그리드 작업 스케줄링 모델을 구성하였다. 이 모델은 그리드 자원의 CPU 속도, 네트워크 지연시간, 작업 가용 큐 등과 같은 파라미터를 입력으로 한 퍼지로직을 이용하여 출력된 자원효율성을 토대로 그룹을 구성하며, 가장 높은 효율성을 제공하는 그룹 내의 자원들에게 작업 크기 및 자원 성능을 고려하여 작업을 할당하는 스케줄링 모델이다. 또한 성능 측정을 위하여 DEVS (Discrete Event System Specification) 모델링 & 시뮬레이션[4]을 이용하여 그리드 환경의 퍼지로직 기반 그리드 작업 스케줄링 모델을 설계하고 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 그리드 스케줄링 모델과 퍼지로직에 대해 소개하고, 3장에서는 퍼지로직 기반의 그리드 작업 스케줄링 모델에 대하여 설명한다. 그리고 4장에서는 DEVS 모델링 & 시뮬레이션 환경[4]에서 다른 기존 스케줄링 모델과의 비교 실험 결과를 통해 본 논문에서 제안된 퍼지로직 기반의 그리드 작업 스케줄링 모델의 효율성 및 효과를 증명하고 마지막 5장에서는 결론을 맺는다.

Ⅱ. 관련연구

2.1 그리드 스케줄링

그리드 컴퓨팅에서는 이기종의 그리드 자원을 공유하여 연산문제를 처리하므로 각 자원을 평가하고 평가 내용을 토대로, 어느 자원에게 얼마만큼의 작업을 할당할 것인가에 대해 결정하는 것이 매우 중요하다. 이러한 자원 평가 기준과 스케줄링 방식은 효율적인 작업 처리에 큰 영향을 미친다[5].

자원을 평가하여 스케줄링 하는 대표적인 방식으로는 옥션 모델[6], MCT(Minimum Completion Time), MET (Minimum Execution Time) [7] 등이 있다. 경제이론을 도입하여 작업을 할당하는 옥션 모델은 경매를 바탕으로 자원이 작업을 처리하는데 사용되는 비용을 줄이는 것에 초점을 맞추고 있다. MCT 모델[7]은 해당 작업에 대하여 가장 작은 완료시간을 가지는 자원에게 작업을 할당 하는 방식이며, MET모델은[7] 실행시간이 가장 작은 자원에게 작업을 할당 하는 방식이다. 하지만 MCT, MET 모델[7]은 실제로 기대 예측 시간을 구하기 위해서는 많은 커뮤니케이션 메시지와 프로세서 성능을 필요로 하고 패킷 손실 및 지연 가능성이 존재하며, 옥션 모델[6]은 자원의 상태 데이터에 따른 작업 처리의 비용을 줄이고 사용자 만족도의 향상에 목적을 두고 있으므로 다른 스케줄링 모델들에 비해 빠른 작업 처리가 어렵다는 단점이 있다.

본 논문에서 제안하는 위한 스케줄링 모델은 빠르고 안정적인 작업 처리를 위해 처리 비용, 실행 시간 등과 같은 단순한 측정 기준이 아닌 자원의 처리 능력, 네트워크 상태 등을 고려한 자원 효율성을 기준으로 작업을 할당함으로써 작업의 실행 시간 단축만이 아닌 안정적인 작업 처리를 가능케 한다.

2.2 퍼지이론

퍼지이론은 시스템의 복잡성을 설명하며, 모델의 복잡성을 단순화 하는 방법을 제공한다. 퍼지이론은 수학적으로 복잡한

모델링 평가문제에 있어서 단순화 하여 모델링을 더욱 쉽게 하며 현재 인공 지능 신경망, 로봇제작 등과 같은 다양한 하이테크 분야에 이용되고 있다. 퍼지이론은 표현하기 어려운 애매한 측정값을 구조적으로 정의하여 시스템에서 사용할 수 있는 공학적으로 유용한 문장을 구성하고 퍼지 규칙을 작성하여 이 규칙에 따라 출력을 결정한다. 퍼지 규칙은 여러 개의 조건 항과 결론 절을 가지고, 해당 규칙에 대한 결과 값을 가지고 있다. [8]

퍼지로직은 입력자료를 퍼지화한 퍼지추론과 일반적인 IF-Then 형식을 이용하여 결과를 출력하며, 이 결과 값을 여러 역퍼지화 방법을 통해 실제 출력값을 도출한다. 본 논문에서는 퍼지 규칙의 추론 방법으로는 맘다니(Mandani)의 Min-Max 합성법[9]을 이용하고, 실제 실수 값을 출력을 위한 역퍼지화(Defuzzification)에는 무게중심법(Center Of Graity Method)을 이용하였다.

III. 퍼지로직 기반의 그리드 작업 스케줄링 모델

3.1 퍼지로직 기반의 그리드 작업 스케줄링 모델의 구성

수많은 자원들을 포함하는 그리드에서 하나의 관리 서버를 이용하여 모든 자원들의 상태를 모니터링하고 관리하는 것은 그리드 미들웨어에 과부하로 이어져 자원관리와 스케줄링에 어려움이 발생할 수 있다. 이러한 문제를 해결하기 위해 퍼지로직 기반의 그리드 작업 스케줄링 모델에서는 분산구조를 이용하여, 자원들을 관리하고 작업을 할당하여 그리드 환경에서의 효율적인 자원관리와 스케줄링을 실현한다.

퍼지로직 기반의 그리드 작업 스케줄링 모델은 크게 4가지 컴포넌트로 구성되며 <그림 1>과 같은 구조로 이루어진다.

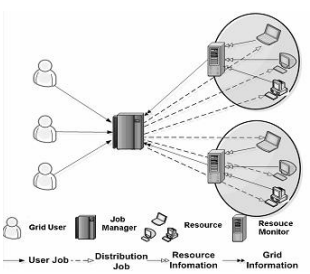


그림 1. 퍼지로직 기반의 그리드 작업 스케줄링 모델의 구성도
Fig. 1. Component Composition of Fuzzy Logic-based Grid Job Scheduling Model

각 컴포넌트의 기능을 설명하자면 다음과 같다.

- 그리드 사용자(Grid User) : 그리드 컴퓨팅 환경에서 작업을 처리를 요구하는 사용자
- 자원(Resource) : 자원 스케줄러로부터 받은 작업을 처리한다. 각 자원들은 서로 다른 프로세서 능력, 네트워크 상태, 작업 가용 큐를 가지고 있다. 그리고 자신의 상태가 변경 될 때 마다 자원모니터에게 상태를 전달한다.
- 자원모니터(Resource Monitor) : 각 자원들의 상태를 관찰 및 측정하는 자원모니터가 존재한다. 각 자원모니터는 자원들의 상태변화를 측정하고 이를 바탕으로 퍼지로직을 적용하여 각 자원의 자원 효율성을 측정한 뒤 그 정보를 작업매니저에게 전달한다.
- 작업매니저(Job manager): 작업매니저는 그리드 사용에게 받은 작업을 분할하여 각각의 자원의 효율성을 토대로 그룹을 구성하여 작업을 할당한다.

퍼지로직 기반의 그리드 작업 스케줄링 모델에서 이용한 분산구조는 작업매니저에 걸리는 작업부하의 집중을 분산시키고 광대역으로 분산된 그리드 자원들을 결합하고 관리하는데 유리하다. 그리드 사용자는 자원요구와 자원소모를 담당한다. 자원모니터의 역할은 크게 두 가지로 나뉜다. 첫 번째로는 자원의 작업 가용 큐가 꽉 차 있는 자원들을 1차 필터링을 하며 일정 이하 프로세서 능력, 일정이상의 네트워크 지연시간을 가진 자원들을 2차 필터링한다. 무한대에 가까운 자원을 가진 그리드 컴퓨팅에서는 모든 자원을 관리하는 것은 그리드 미들웨어의 과부하를 유발하기 때문에, 이러한 필터링을 통해 그리드 미들웨어의 과부하를 줄일 수 있으며, 그리드 컴퓨팅에서 광대역으로 분산된 광역자원들을 결합하고 관리하는데 유리하다. 두 번째 역할은 자원의 발견과 자원의 프로세서의 성능, 네트워크 지연시간, 작업 가용성을 측정하며, 측정된 값을 퍼지로직을 이용하여 자원의 효율성을 계산한다. 자원의 효율성을 측정이 완료되면, 측정된 효율성을 작업 매니저에 전달한다. 작업매니저는 전달받은 효율성을 이용하여 자원 그룹을 구성하며, 구성된 그룹에게 작업을 할당하는 역할을 한다.

3.2 자원효율성 측정을 위한 퍼지 이론의 적용

우리는 자원의 효율성을 측정하기 위하여 그리드 컴퓨팅 환경에서의 그리드 자원의 성능 지표를 이용하였다. 그리드 컴퓨팅 자원은 서로 다른 성능을 가진 자원들로 구성되어 있다. 우리는 그리드 성능에 가장 큰 영향을 미치는 자원의 CPU 속도, 네트워크 지연시간, 작업 가용 큐 크기를 퍼지 입력 변수 X, Y, Z로 지정하고 자원 효율성을 퍼지 출력

변수 E로 지정하였다. CPU 속도는 각 자원이 소유한 CPU의 클럭 속도 레벨을 나타내고 네트워크 지연시간은 각 자원이 하나의 패킷을 자원 모니터에게 전달하는데 걸리는 시간 레벨을 나타내며 작업 가용 큐 크기는 각 자원이 작업을 저장하기 위해 사용 가능한 큐의 크기 레벨을 나타낸다. 입, 출력변수의 구성은 다음과 같다.

X (자원의 CPU 속도)={VERY SLOW, SLOW, MEDIUM, FAST, VERY FAST}
Y (자원의 네트워크 지연시간)={VERY SLOW, SLOW, MEDIUM, FAST, VERY FAST}
Z (자원의 작업 가용 큐 크기)={VERY SMALL, SMALL, MEDIUM, LARGE, VERY LARGE}
E (자원 효율성)={VERY POOR, POOR MEDIUM, EFFICIENCY, VERY EFFICIENCY}

퍼지로직에서는 각 입,출력 변수에 대한 멤버십 함수를 가지고 있다. 〈그림 2〉는 입력 변수인 자원의 CPU 속도에 대한 멤버십 함수를 나타내고, 〈그림 3〉은 입력 변수인 자원의 네트워크 지연시간에 대한 멤버십 함수를, 〈그림 4〉는 입력 변수인 자원의 작업 가용 큐에 대한 멤버십 함수를 나타낸다. 〈그림 5〉는 출력 변수인 자원효율성에 대한 멤버십 함수이다.

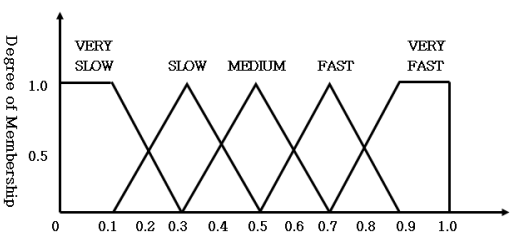


그림 2. CPU 속도 (입력 변수)
Fig. 2. CPU Speed

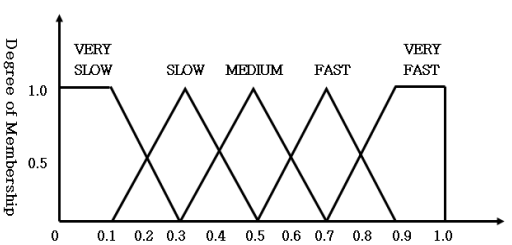


그림 3. 네트워크 지연시간 (입력 변수)
Fig. 3. Network Latency

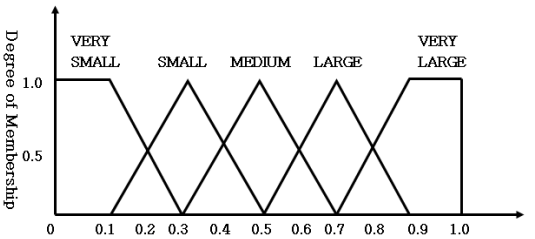


그림 4. 작업 가용 큐의 크기 (입력 변수)
Fig. 4. Size of Available Queue

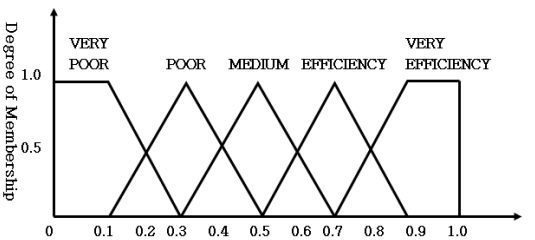


그림 5. 자원 효율성 (출력 변수)
Fig. 5. Resource Efficiency

본 논문에서는 출력값 결정을 위해 5개의 상태를 가진 3개의 입력변수 X, Y, Z를 이용하여 〈그림 6〉과 같이 125개의 퍼지규칙을 생성한다. 이 퍼지 규칙들을 이용하여 각 자원의 효율상태를 추론할 수 있다. 예를 들어 CPU 속도와 네트워크 상태가 매우 느리고 가용 큐의 크기가 매우 작을 경우 그 자원의 자원 효율성은 매우 좋지 않다고 추론할 수 있다.

```
RULE 0:IF(X IS VERY SLOW) AND (Y IS VERY SLOW) AND (Z IS VERY SMALL) THEN (E IS VERY POOR);
RULE 1:IF(X IS VERY SLOW) AND (Y IS VERY SLOW) AND (Z IS SMALL) THEN (E IS POOR);
RULE 2:IF(X IS VERY SLOW) AND (Y IS VERY SLOW) AND (Z IS MEDIUM) THEN (E IS POOR);
RULE 3:IF(X IS VERY SLOW) AND (Y IS VERY SLOW) AND (Z IS LARGE) THEN (E IS MEDIUM);
RULE 4:IF(X IS VERY SLOW) AND (Y IS VERY SLOW) AND (Z IS VERY LARGE) THEN (E IS MEDIUM);
.
.
RULE 124:IF(X IS VERY FAST) AND (Y IS VERY FAST) AND (Z IS VERY LARGE) THEN (E IS VERY EFFICIENCY);
```

그림 6. 자원 효율성 측정을 위한 퍼지 규칙
Fig. 6. Fuzzy Rules for Resource Efficiency Measurement

또한, 퍼지 출력값을 실제로 사용하기 위해서는 등가의 정확한 수치로 변환하는 과정인 추론과 역퍼지화(Defuzzification)의 과정이 필요하다. 따라서 본 논문에서는 정확한 출력값 추론을 위해서 맘다니(Mandani)의 Min-Max 합성법[9]을 이용하였고 실제 실수 출력을 생성하기 위한 역퍼지화에는 무게 중심법을 이용하여 각 자원의 정확한 자원 효율성을 도출하였다.

3.3 작업 스케줄링

그리드 컴퓨팅 환경에서 사용하는 중앙 스케줄링 방식을 이용하여 광대역으로 분산된 자원을 관리 할 경우 발생하는 수많은 통신 메시지와 방대한 량의 자원 정보데이터는 그리드 미들웨어 상에서 오버헤드를 발생시킬 수 있다. 따라서 본 논문에서는 그리드 자원들을 여러 개의 그룹으로 나누고, 그룹별로 자원을 관리한다. 이를 위해 본 논문에서는 TwoThreshold알고리즘[10]을 이용하여 두 개의 임계값을 이용하여 동적으로 그룹의 수를 조정한다.

그룹을 구성 할 때 너무 많은 그룹의 수는 오히려 자원의 선택의 어려움과 작업 처리 성능의 저하를 가져 올 수 있다. 적절한 그룹의 수를 유지하기 위해 본 논문에서는 Two Threshold 알고리즘 [10]을 적용하여 자원 그룹을 생성 및 재구축하였다. <그림 7>은 Job매니저에서 그룹 구성에 사용되는 Two Threshold 알고리즘 [10]을 표현한 의사코드이다.

```
For m = 1 to GroupNum
  For i = 1 to ResourceNum
    ldt = LD(R_i, G_m)
    IF(ldt < T_1)
      GroupInclude(G_m, R_i)
    ElseIF(T_1 < ldt < T_2)
      LaterGroupInclude(R_i)
    ElseIF(ldt > T_2)
      NewGroup(R_i)
      GroupNum + 1
```

그림 7. Two Threshold 알고리즘을 위한 의사 코드
Fig. 7. Two Threshold Algorithm [10]

<그림 7>에서 T_1은 기존의 그룹에 포함 될 수 있는 최대 임계값을 나타내고 T_2는 새로운 그룹을 만들 수 있는 최소 임계값을 나타내어준다. R_i는 자원의 효율성을 나타낸다. LD 함수는 R_i와 C_m의 효율성 차이를 구하는 함수 이다. GroupInclude 함수는 기존의 그룹에 해당 자원을 포함시키는 함수이며 NewGroup 함수는 새로운 그룹을 구성하는 함수이다.

분산 작업 처리 시스템에서는 하나의 Job이 여러 Sub-Job으로 나누어지는데 각 Sub-Job의 크기가 서로 다르다. 일반적인 분산 작업 처리에 소요되는 시간 T는 다음 수식(1)과 같다.

$$T = Max\left(\sum_{j=0}^{m} JC_j\right) \quad \cdots\cdots (1)$$

T는 분산 작업 처리에 소요되는 총 시간을 나타내고 m는 작업을 처리하는 자원의 수를 나타내며 JC는 각 자원이 분산 작업을 처리하는 시간을 나타내준다. 위의 수식(3.1)에서 분산 처리 작업은 가장 늦게 수행된 자원에 전체적인 처리 시간을 영향을 받게 된다. 즉 그룹 내의 모든 자원이 최대한 같은 시간에 작업이 처리될 수 있도록 작업이 할당 되어야 한다. 우리는 이러한 문제점을 해결하기 위해 아래와 같은 작업 할당 알고리즘을 실행한다.

(1) 작업 매니저는 그리드 사용자에 받은 Job을 Sub-Job으로 분할한다.
(2) 작업 매니저는 자원들의 효율성을 자원모니터로 부터 요청한다.
(3) 자원들의 효율성을 높은 순서대로 정렬한다.
(4) 각 Sub-Job의 크기를 큰 순서대로 정렬한다.
(5) 작업매니저는 가장 큰 Sub-Job을 가장 높은 효율성을 지닌 자원 에 할당한다.
(6) 각 자원에 할당된 Sub-Job을 수행한다.
(7) 처리된 결과를 취합하여 결과 데이터를 생성한다.

Ⅳ. 실험 결과

우리는 본 논문에서 제시한 퍼지로직 기반의 그리드 작업 스케줄링 모델의 성능 평가를 위해 DEVS 모델링 & 시뮬레이션[4] 환경에서 시뮬레이션을 실시하였다. 그리고 퍼지로직 기반의 그리드 작업 스케줄링 모델의 효과 및 효율성을 증명하기 위해 기존의 스케줄링 모델인 Random 스케줄링 방식과 MCT 스케줄링 모델[7]을 추가로 구현하여 실험 결과를 비교하였다. 세 스케줄링 모델 모두 동일한 조건에서 실험을 하였다. 그리드 사용자는 일정 간격으로 작업

을 발생시키며, 생성되는 작업은 작업의 크기가 크며 비연속적인 작업(Large Job)과 작업의 크기가 작으며 연속적인 작업(Small Job)으로 분류된다. 또한 각 작업은 각각 다른 Sub-Job을 가지고 있으며, 모든 Sub-Job이 완료된 경우에만 작업 완료로 가정하였으며, 모든 Sub-Job이 완료되는 시간을 작업 완료시간으로 계산 하였다. 그리드 자원의 특징을 표현하기 위해 CPU 속도와 네트워크 지연 시간, 작업 가용 큐의 크기를 서로 다르게 설정된 다양한 형태의 500개의 자원들로 구성하여 실험을 실시하였다.

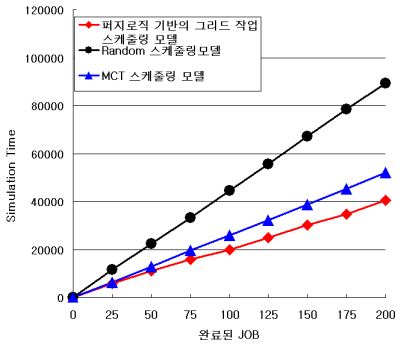


그림 8. Small Job에서의 평균 작업 처리 소요시간
Fig. 8. Turnaround Time(Small Job)

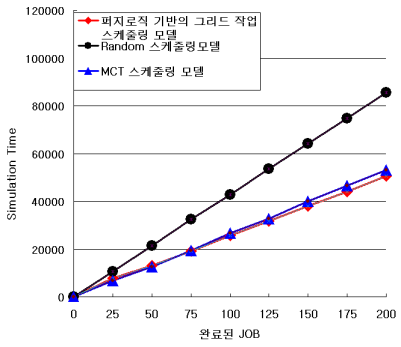


그림 9. Large Job에서의 평균 작업 처리 소요시간
Fig. 9. Turnaround Time(Large Job)

〈그림 8〉, 〈그림 9〉는 각 모델의 시간 별 평균 작업 처리 소요시간을 보여준다. Small Job에서 퍼지로직 기반의 그리드 작업 스케줄링 모델은 평균적으로 작업을 처리하는데 Random 스케줄링 모델보다 약 54.7% 정도 감소된 소요시간을 기록하였고 MCT 스케줄링 모델[7]보다 약 22.3% 정도 감소된 소요시간을 기록하였다. Large Job에서는 Random 스케줄링 모델보다 약 69.5% 정도 감소된 소요시간을 기록하였고, MCT 스케줄링 모델[7]과 비슷한 작업 처리 소요시간을 기록하였다.

두 번째 실험은 각 스케줄링 모델의 작업 손실을 측정하였고, 그 결과는 〈그림 10〉, 〈그림 11〉에서 보여준다. 손실된 작업의 수가 적을수록 안정적인 작업 처리를 보장해 준다고 할 수 있다.

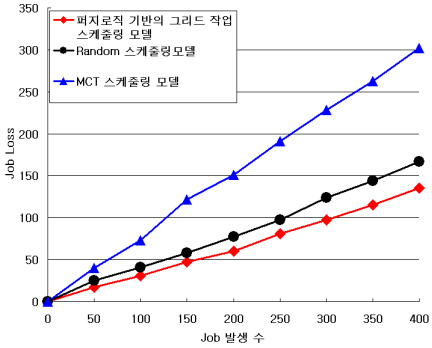


그림 10. Small Job에서의 손실된 작업의 수
Fig. 10. Job Loss(Small Job)

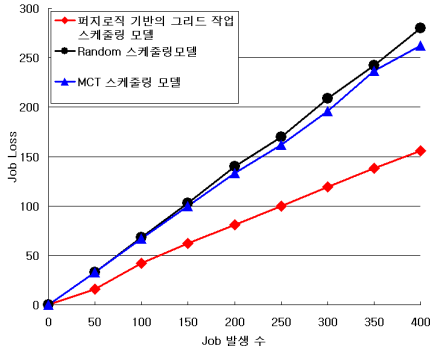


그림 11. Large Job에서의 손실된 작업의 수
Fig. 11. Job Loss(Large Job)

〈그림 10〉, 〈그림 11〉에서 알 수 있듯이 MCT 모델[7]이 가장 높은 작업 손실률을 기록하였다. 이와 같은 결과는 MCT 스케줄링 모델[7]이 자원의 작업 가용성을 고려하지 않고 빠른 프로세서 성능과 네트워크 성능을 가진 자원에게 작업을 연속적으로 할당하기 때문이다. Small Job에서는 퍼지로직 기반의 그리드 작업 스케줄링 모델이 MCT 스케줄링 모델[7]과 Random 스케줄링 모델보다 55.3%와 19.2%의 낮은 작업 손실률을 기록한 반면 Large Job 모

델에서는 Random 모델과 비슷한 작업 손실률을 나타낸다. 하지만 위의 작업완료시간 측정 실험의 데이터와 연관 지어 보면, Random 스케줄링 모델은 여러 자원에게 동일하게 작업을 분배하여 낮은 작업 손실률을 기록한 반면 자원의 프로세서 성능이나 네트워크 성능을 고려하지 않은 작업 할당에 의해 퍼지로직 기반의 그리드 작업 스케줄링 모델보다 많은 작업처리시간이 소요됨을 알 수 있다.

세 번째 실험은 통신량을 측정 하였다. 통신량은 각 자원이 작업을 처리하는 동안 스케줄러와 통신 하는 메시지의 총 수를 측정한 것이다. 그 결과는 〈그림 12〉,〈그림 13〉과 같다.

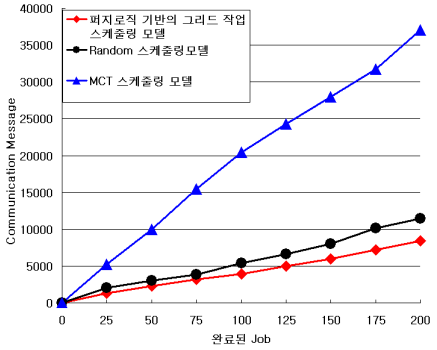


그림 12. Small Job에서의 통신량
Fig. 12. Communication Message(Small Job)

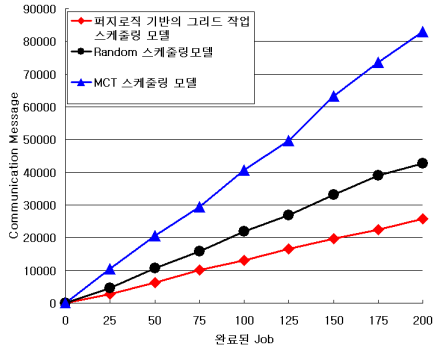


그림 13. Large Job에서의 통신량
Fig. 13. Communication Message (Large Job)

〈그림 12〉은 Small Job에서 퍼지로직 기반의 그리드 작업 스케줄링 모델이 Random 스케줄링 모델과 MCT 스케줄링 모델[7]과 비교하여 각각 26.1%, 76.3% 정도의 통신량을 감소시킨다는 사실을 보여준다. 또한 〈그림 13〉에서 보여주듯이 퍼지로직 기반의 그리드 작업 스케줄링 모델은 Large Job에서도 Random 스케줄링 모델과 MCT 스케줄링 모델[7]보다 약 29.6%, 69.9% 감소된 통신량을 기록하였다.

위의 실험 결과들은 퍼지로직 기반의 그리드 작업 스케줄링 모델이 계산 그리드 환경에서 빠른 작업 처리와 작업 안정성, 적은 통신량을 제공함으로써, 효율적이고 빠른 작업 처리를 보장해줄 수 있음을 증명하고 있다. 이를 통해 우리는 본 논문에서 제안한 퍼지로직 기반의 그리드 작업 스케줄링 모델이 계산 그리드 컴퓨팅 환경에서의 그리드 자원 관리 및 작업 스케줄링에 효과적이라는 사실을 알 수 있다.

VI. 결론 및 향후 과제

시간이 갈수록 그리드 컴퓨팅에 대한 요구가 계속 증가함에 따라 다양한 이기종의 자원들로 구성된 그리드 컴퓨팅에서 효율적이고 안정적인 작업 처리를 위한 새로운 스케줄링 기법의 필요성이 대두되고 있다. 본 논문에서는 분산된 이기종의 자원들의 성능을 측정하여 작업을 할당하는 하기 위해 퍼지로직 기반의 그리드 작업 스케줄링 모델을 제안하였다. 퍼지로직 기반의 그리드 작업 스케줄링 모델은 다양한 자원 정보를 바탕으로 퍼지로직을 이용하여 각 자원의 자원효율성을 측정하고 이를 바탕으로 자원들을 그룹화 하여 작업을 할당한다.

본 논문에서는 퍼지로직 기반의 그리드 작업 스케줄링 모델의 성능을 평가하기 위해 DEVS 모델링 환경에서, 기존 스케줄링 모델(Random 스케줄링 모델, MCT 모델[7]) 들과 비교하여 두 가지 작업 유형에 대한 각 모델별 작업처리완료시간, 작업 손실, 통신량을 측정하였다. 이 비교 실험으로 통해 우리는 퍼지로직 기반의 그리드 작업 스케줄링 모델이 기존 스케줄링 모델들에 비해 계산 그리드 상에서 빠르고 안정적으로 작업을 처리하고 통신량을 감소시키실 수 있음을 증명하였다. 이것은 곧 퍼지로직 기반의 그리드 작업 스케줄링 모델이 계산그리드 컴퓨팅 환경에서 작업을 할당하고 처리하는 것에 매우 효과적이라는 것을 증명해주는 결과이다.

참고문헌

[1] F. Berman, G. Fox and T. Hey, (2003), "The Grid: past, present, future, Grid Computing - Making the Global Infrastructure a Reality", Wiley and Sons.

〔2〕 Xiu-Chuan Wu, Linag Hu and Jiu-Bin Ju, (2003), "the active computational grid framework. Machine Learning and Cybernetics", ACGWPRS, 2003 International Conference on Vol 2, pp.994-999.
〔3〕 R. Buyya, D. Abramson, J. Giddy and H. Stockinger,(2002), "Economic Models For Re-source Management and Scheduling in Grid Computing. Journal of Concurrency and Computation" Practice and Experience (CCPE) Wiley Press.
〔4〕 Zeigler, B.P., et al. (1997), "The DEVS Environment for High Performance Modeling and Simulation" IEEE CS & E, Vol. 4, No3, pp 61-71.
〔5〕 Junzhou Luo, Peng Ji, Xiaozhi Wang, Ye Zhu, Feng Li, Teng Ma and Xiaopeng Wang, (2004), "Resource management and task scheduling in grid computing", Computer Supported Cooperative Work in Design, Proceedings. The 8th International Conference on Vol 2, pp. 431-436.
〔6〕 R. Buyya, (2002) "Economic based Distributed Resource Management and Scheduling for Grid Computing" , Available at http://www.buyya.com/thesis/ .
〔7〕 M.Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, (1997), "Dynamic Matching and Scheduling Independent Tasks on Nonidentical Processors," Journal of the ACM, vol. 23, no. 2, pp.280-289.
〔8〕 Zadeh, L. A., (1965), "Fuzzy sets", Information and control, vol. 8, pp.338-353.
〔9〕 J. S. R. Jang, C. T. Sun, E. Mizutani,(1997) "Neuro-Fuzzy and Soft Computing", Prentice-Hall International.
〔10〕 Sergios Theodoridis and Konstantinos Koutriumba, (1999), "Pattern recognition" Academic Press, pp.385-394.

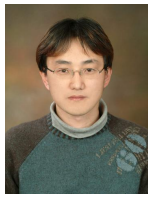
저 자 소 개

박 량 재

2006년 : 천안대학교 컴퓨터공학부 학사
2006년 ~ 현재 :인하대학교 정보공학과 석사과정
관심분야: 시스템 모델링 및 시뮬레이션, 그리드 컴퓨팅

장 성 호

2004년 : 용인대학교 컴퓨터공학부 학사
2006년 : 인하대학교 정보공학과 석사
2006년~ 현재 : 인하대학교 정보공학과 박사과정
관심분야 : 그리드 컴퓨팅, 모바일 컴퓨팅, RFID 응용프로그램



조 규 철

2005년: 인하대학교 컴퓨터공학부 학사
2007년: 인하대학교 정보공학과 석사
2007년~현재: 인하대학교 정보공학과 박사과정
관심분야 : 그리드 컴퓨팅, 소프트웨어 공학, 패턴인식

이 종 식

1993년 : 인하대학교 전자공학과 학사
1995년 : 인하대학교 전자공학과 석사
2001년 : 미국 애리조나대 전기 ·컴퓨터공학과 박사
2001~2002년 : 미국 캘리포니아 주립대학교 전기·컴퓨터공학과 전임강사
2002~2003년 : 미국 클리블랜드 주립대학교 전기·컴퓨터공학과 조교수
2003~2006년 : 인하대학교 컴퓨터공학부 조교수
2006년 9월~현재 : 인하대학교 컴퓨터공학부 부교수
관심분야: 시스템 모델링 및 시뮬레이션, 그리드 컴퓨팅