# 계산 그리드를 위한 서비스 예측 기반의 작업 스케줄링 모델

장성호*, 이종식*

# Service Prediction-Based Job Scheduling Model for Computational Grid

Sung-Ho Jang, Jong-Sik Lee

**Abstract**

Grid computing is widely applicable to various fields of industry including process control and manufacturing, military command and control, transportation management, and so on. In a viewpoint of application area, grid computing can be classified to three aspects that are computational grid, data grid and access grid. This paper focuses on computational grid which handles complex and large-scale computing problems. Computational grid is characterized by system dynamics which handles a variety of processors and jobs on continuous time. To solve problems of system complexity and reliability due to complex system dynamics, computational grid needs scheduling policies that allocate various jobs to proper processors and decide processing orders of allocated jobs. This paper proposes the service prediction-based job scheduling model and present its algorithm that is applicable for computational grid. The service prediction-based job scheduling model can minimize overall system execution time since the model predicts a processing time of each processing component and distributes a job to processing component with minimum processing time. This paper implements the job scheduling model on the DEVSJAVA modeling and simulation environment and simulates with a case study to evaluate its efficiency and reliability. Empirical results, which are compared to the conventional scheduling policies such as the random scheduling and the round-robin scheduling, show the usefulness of service prediction-based job scheduling.

**Key Words:** Computational Grid, Job Scheduling, DEVS Modeling and Simulation

* 인하대학교 컴퓨터공학부

## 1. 서론

현재 인터넷의 보편화와 컴퓨터 네트워크의 발달로 인하여 클러스터를 대체하는 차세대 기술로서 그리드 컴퓨팅(Grid Computing)이 부각되고 있다[1]. 그리드란 지리적으로 분산된 컴퓨터, 대용량 저장장치 등의 고성능 자원들을 네트워크로 상호 연동하여 사용자가 단일 시스템처럼 모든 그리드 자원들을 사용할 수 있는 정보통신 인프라를 말한다. 그리드는 계산 그리드(Computational Grid), 데이터 그리드(Data Grid), 접근 그리드(Access Grid)로 분류된다[2]. 시간이 갈수록 기존의 클러스터 시스템이나 슈퍼컴퓨터로 해결할 수 없는 대용량의 복잡한 연산 집약적인 문제들이 늘어감에 따라 계산 그리드에 대한 연구와 개발이 한창 진행되고 있다. 계산 그리드는 현재 과학연구, 금융, 기계 항공 등의 산업 전 분야 걸쳐 적용되며 그리드 상의 자원들을 연결하여 고속, 고성능 연산을 요구하는 컴퓨팅을 위해 실행시간의 최소화를 목표로 한다.

계산 그리드의 핵심은 계산속도 향상과 처리량 분산을 위한 효과적 스케쥴링 기술이다[3]. 이는 전체 시스템 성능의 최적화 문제와 직결된다. 그리드에 존재하는 프로세서들의 수와 계산 처리를 필요로 하는 서비스의 수가 거의 무한대에 이르며 그리드 컴퓨팅 자원이 지리적, 조직적으로 독립적인 객체들로 구성되어 있고 개개의 관리 정책을 가지고 있으므로 그리드는 매우 동적인 환경 특성을 가진다. 이러한 환경 특성으로 인해 그리드의 자원 할당과 스케쥴링은 매우 어렵고 복잡하다.

본 논문에서는 계산 그리드에서 적용 가능한 서비스 예측 기반의 작업 스케쥴링 모델과 실행시간 예측 알고리즘을 제시한다. 우리는 DEVSJAVA 시뮬레이션 환경에서 시뮬레이션 모델을 설계하고 기존 스케쥴링 모델들과의 비교를 통해 성능 평가를 실시한다.

본 논문의 구성은 다음과 같다. 2장에서는 서비스 예측 기반의 작업 스케쥴링 모델을 제안한다. 3장에서는 시뮬레이션의 결과 분석을 통하여 모델의 효율성과 효과를 입증하고 마지막으로 4장에서는 결론을 맺는다.

## 2. 서비스 예측 기반의 작업 스케쥴링 모델

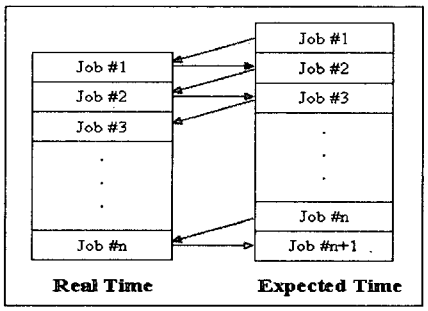
서비스 예측 기반의 작업 스케쥴링 모델은 네트워크 상에서 발생 가능한 작업 손실(job loss)과 전체 시스템 실행시간의 최소화하고 작업 처리량(throughput)을 최대화함으로서 시스템 내의 프로세서 활용도의 극대화를 목표로 한다. 시스템 실행시간은 시스템 용량, 프로세서의 활용도, 작업 크기 및 복잡도 등에 영향을 받는다. 이를 위해 모델은 실행시간 예측 알고리즘을 제공한다. 프로세서 수와 성능의 다양성은 실제 실행시간의 정확한 예측을 아주 어렵고 복잡하게 만든다. 작업크기의 다양성과 단일 서비스 타입과 멀티 서비스 타입으로 분류되는 서비스 복잡도 역시 실행시간 예측을 더욱 어렵게 만드는 요소이다. 작업 스케쥴링 모델에서는 예측 방해 요소를 간소화하기 위해 멀티 서비스를 제외한 단일 서비스의 실행시간 예측에 초점을 맞춘다. 모델은 일정 시간동안 시스템이 동일한 서비스를 실행하였던 경험을 데이터베이스화하여 실제 서비스가 시스템에서 실행되기 전 축적된 데이터베이스를 바탕으로 시스템의 다음 서비스 실행 시간을 계산하여 예측한다.

<그림 1>은 실행시간 예측 알고리즘을 의사코드어로 표현한 것이다. predict_T는 시스템 상에 실행되는 최초 서비스 작업의 예상 실행시간이다. T_expectedtime[]는 예상 실행시간을 축적하는 배열 요소이고 T_realtime[]는 실제 실행시간을 축적하는 배열 요소이다. 이 두 요소는 일정 시간동안 시스템이 동일 서비스를 실행한 경험의 데이터베이스를 나타낸다. 작업이 입력되기 전 최초 작업의 예상 실행시간인 predict_T가 계산된다. alpha는 학습비율이고 T_expectedtime[0], T_realtime[0]은 실험을 통해 얻을 수 있다. 계산을 통해 얻어진 값은 첫 번째 예상 실행시간인 배열 T_expectedtime[1]에 저장된다. 서비스 작업이 시스템 상의 프로세서에서 동작중일 때 동일한 서비스 작업이 입력되는 경우가 발생한다. 이 경우 첫 번째 예상 실행시간처럼 다음 작업의 예상 실행시간을 계산할 수는 있지만, 작업의 개수가 증가할수록 시스템부하의 영향을 받아 실제 시간과 예상 시간과의 오차가 증가하는

동적인 시스템부하로 인해 정확한 예상 실행시간을 계산하는 것은 어렵다. 그러나 T_expectedtime[i]/T_expectedtime[i-1]의 비율을 예측하는 것은 그리 어려운 일이 아니다. 우리는 예상 실행시간의 비율 expect[]와 실제 실행시간의 비율 real[]을 사용하여 반복 계산함으로써 n번째 예상 실행시간인 T_expectedtime[n]을 <그림 1>에서와 같이 예상할 수 있다.

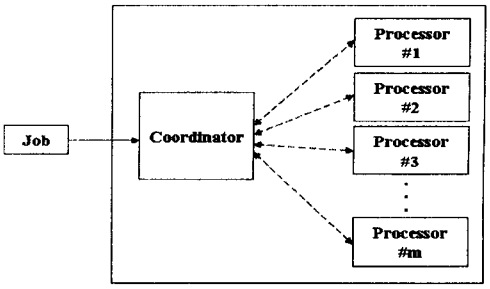
```
If (not exist T_expectedtime[1])
   predict_T = ((1-alpha) * T_expectedtime[0])+(alpha * T_realtime[0]);
   T_expectedtime[1] = predict_T;
Else if (exist T_expectedtime[1])
{
   for (i=1;i<=n-1 ;i++ )
   {
      expect[i] = T_expectedtime[i] / T_expectedtime[i-1];
      real[i] = T_realtime[i] / T_realtime[i-1];
      predict_T = (((1-alpha) * expect [i]) + (alpha * real [i]));
      total_pre_T * = predict_T;
   }
   T_expectedtime[n] = total_pre_T * T_expectedtime[1];
}
```

<그림 1> 실행시간 예측 알고리즘




<그림 2> 예상 실행시간의 계산 순서

<그림 2>는 예상 실행시간의 계산 순서를 도식화한 것이다. 모델에서는 n번째 작업이 도착하였을 때 n+1번째의 작업의 예상 실행시간을 계산하여 예측한다. 이렇게 계산된 예상시간은 그리드 시스템 상의 프로세서들의 작업 할당에 이용된다. 모델에서 그리드의 프로세서들을 <그림 3>과 같이 연결된다. 작업이 그리드 시스템에 입력되면 스케쥴링을 담당하는 코디네이터는 각 프로세서들의 예상시간을 비교하여 최저 예상 실행시간을 제공하는 프로세서에게 작업을 할당한다.



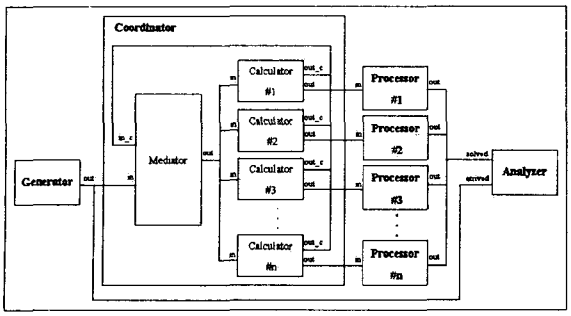

<그림 3> 작업 이동에 따른 프로세서 구성

## 3. 실험 및 결과 분석

본 논문에서는 서비스 예측 기반의 작업 스케쥴링 모델의 평가를 위해 <그림 4>와 같이 시뮬레이션 모델을 구성하였다. 시뮬레이션 환경은 DEVSJAVA 모델링&시뮬레이션[4]을 사용하였다. 우리는 작업 스케쥴링 모델 외에도 랜덤 스케쥴링(Random scheduling) 모델과 라운드로빈 스케쥴링(Round-robin scheduling) 모델과의 실험 비교를 통하여 작업 스케쥴링 모델의 효능을 입증한다.

### 3.1 시뮬레이션 모델 구성

서비스 예측 기반의 작업 스케쥴링 모델은 <그림 4>와 같이 크게 4개의 컴포넌트로 구성된다. Generator는 그리드에 입력되는 작업을 생성시키고 Coordinator와 Analyzer에게 작업을 전송한다. Coordinator는 Mediator와 Calculator로 구성된다. Calculator 는 3장에서 제안한 실행시간 예측 알고리즘을 사용하여 각 프로세서의 예상 실행시간을 계산한다. 시뮬레이션 시작과 동시에 Calculator는 계산된 각 프로세서의 예상 실행시간을 Mediator에게 보내고 Mediator는 예상 실행시간 비교 후 최저 예상 실행시간을 제공하는 프로세서에게 작업을 할당한다. 이때, 작업을 할당받은 프로세서의 Calculator는 다음 작업의 예상 실행시간을 예측하여 Mediator에게 전송함과 동시에 Processor에게 작업을 전달한다. Processor가 입력받은 작업을 처리하고 Analyzer에게 전달하면 Analyzer는 Generator와 Processor에게 전달받은 작업 메시지를 비교 분석한 후 작업 손실, 작업 처리량, 평균 처리 소요시간 등을 계산하여 시스템 성능을 평가한다.

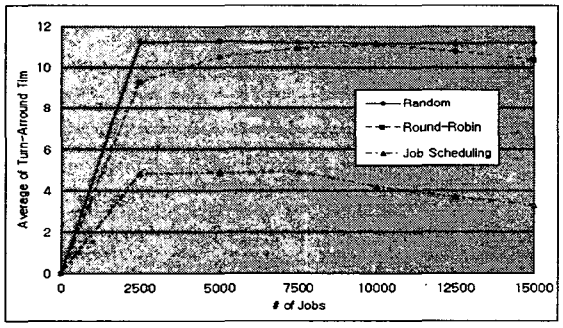



<그림 4> 시뮬레이션 모델 구성도

### 3.2 시뮬레이션 결과 분석

#### 3.2.1 실험 1: 작업 당 평균 처리시간 분석

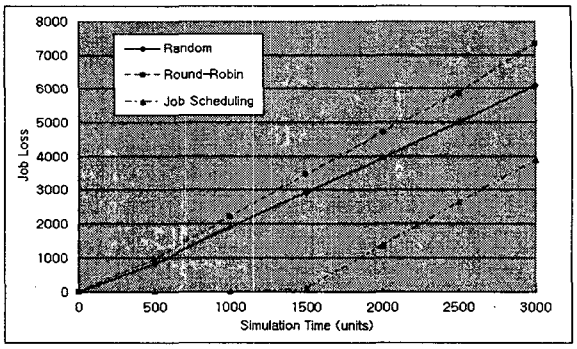
실험 1은 각 스케줄링 모델에 따른 작업 당 평균 처리 소요시간을 분석하기 위한 실험이다. <그림 5>는 각 스케줄링 모델의 작업 개수별 평균 처리 소요시간을 그래프화한 것이다. 작업의 개수가 많고 적음에 관계없이 작업 스케줄링이 랜덤 스케줄링, 라운드로빈 스케줄링과 비교하여 확연히 단축된 작업처리 소요시간을 제공한다. 또한, 작업 개수의 증가 즉 처리 요구량이 증가해도 작업처리 평균 소요시간이 증가하지 않는다는 것은 모델의 성능이 시간과 관계없이 안정적이라는 사실을 증명한다.




<그림 5> 작업 개수별 평균 처리 소요시간

#### 3.2.2 실험 2: 시간 당 작업손실 분석

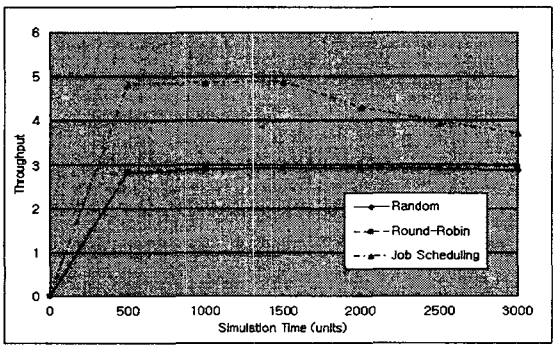
실험 2는 그리드 네트워크 상에서 발생하는 작업 손실을 측정하기 위한 실험이다. <그림 6>은 작업 스케줄링이 다른 모델들과 비교해 작업 손실이 적다는 것을 보여준다. 예를 들어 2000초에서 작업 스케줄링은 1375개의 작업 손실이 발생하여 13.75%의 손실률을 기록한 반면 랜덤 스케줄링과 라운드로빈 스케줄링은 3950개와 4700개의 작업 손실이 발생하여 각각 38.5%와 47%의 손실률을 기록했다. 이는 작업 스케줄링이 기존의 스케줄링 모델들과 비교해 볼 때 그리드 네트워크 상에서 신뢰성 있는 전송을 제공한다는 것을 증명한다.




<그림 6> 시간의 흐름에 따른 작업손실

#### 3.2.3 실험 3: 시간 당 작업 처리량 분석

실험 3은 시간에 따른 스케줄링 모델별 작업 처리량의 변화를 분석하기 위한 실험이다. <그림 7>은 작업 스케줄링, 랜덤 스케줄링, 라운드로빈 스케줄링의 시간당 평균 작업 처리량을 나타낸다. 그림을 보면 작업 스케줄링이 최대 작업 처리량을 랜덤 스케줄링이 최소 작업 처리량을 기록하였다. 예를 들어 1000초에서 시간 예측 스케줄링 방식은 초당 4.852개의 작업을 처리한 반면 랜덤 스케줄링은 초당 2.903개의 작업을 라운드로빈 스케줄링은 초당 2.925개의 작업을 처리하였다. 이는 작업 스케줄링 모델이 단위 시간당 가장 많은 작업을 처리한다는 것을 증명한다.




<그림 7> 시간의 흐름에 따른 작업 처리량

## 4. 결론 및 향후 연구

시간이 갈수록 계산 그리드에 대한 관심과 요구가 계속 증가함에 따라 그리드 컴퓨팅 기술은 발전하고 있으며 계산 그리드를 적용한 프로젝트도 한창 진행되고 있다. 이러한 계산 그리드의 발전에 맞춰 그리드 스케쥴링에 관한 중요도와 필요성 역시 증가하고 있다. 본 논문에서는 계산 그리드의 스케쥴링 문제를 해결하기 위해 서비스 예측 기반의 작업 스케쥴링 모델을 제안하였다.

작업 스케쥴링 모델에서는 효과적인 그리드 자원 할당과 스케쥴링을 위하여 각 프로세서들로부터 경험적인 데이터를 수집한다. 모델은 저장된 데이터들을 바탕으로 실행 시간 예측 알고리즘을 적용하여 프로세서의 실행 시간을 예측하고 적절한 프로세서에 작업을 할당하여 전체 시스템의 실행시간을 줄이고 시간당 작업 처리량을 최대화한다. 또한, 작업 손실을 최소화함으로서 높은 자원 활용도와 신뢰성을 보장하여 시스템 성능 개선을 기대하게 한다. 본 논문에서 제안한 모델은 예측 방해요소를 간소화하기 위해 단일 서비스 타입에 대한 시뮬레이션을 실시하였다. 실험 결과는 본 논문에서 제안한 작업 스케쥴링 모델이 기존의 스케쥴링 모델인 랜덤 스케쥴링, 라운드로빈 스케쥴링과 비교하여 뛰어난 작업 처리량을 제공하고 작업의 분배를 효율적으로 수행하는 것을 증명한다.

## 참고 문헌

[1] F.Berman, G. Fox and T. Hey, Grid computing :making the global infrastructure a reality, J. Wiley, New York, 2003.

[2] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," International Journal of High Performance Computing Application, Vol.15, No.3, pp.200-222, 2001.

[3] Klaus Krauter and Rajkumar Buyya, et al.,"A taxonomy and survey of grid resource management systems for distributed computing," Software Practice and Experience, pp.135 - 164, 2002.

[4]B.P. Zeigler, et al., "The DEVS Environment for High-Performance Modeling and Simulation", IEEE C S & E, Vol.4, No.3, pp.61-71, 1997.