그리드에서 워크플로우 기반 어플리케이션을 위한 서비스 스케줄링

진성호[†] · 이종혁[†] · 유헌창[††]

요 약

최근 들어, 그리드 컴퓨팅 환경은 웹 서비스와 통합된 서비스 기반 그리드 컴퓨팅 환경으로 변화하고 있다. 이러한 구조에서의 그리드 어플리케이션은 워크플로우에 따라 그리드 서비스가 수행되는 서비스 기반 그리드 어플리케이션이 되게 된다. 이것은 그리드 스케줄링 기법에서 그리드 서비스의 특징을 고려해야 한다는 것을 의미한다. 따라서 본 논문에서는 그리드 서비스의 가용성, 접근성, 신뢰성, 처리율, 감속속도에 바탕을 둔 그리드 서비스 성능 모델을 제안하고 서비스 기반 그리드 어플리케이션의 성능향상을 위해서 서비스 스케줄링 기법을 개발한다.

키워드 : 서비스 스케줄링, 그리드 스케줄링, 그리드 서비스, 워크플로우 어플리케이션

Service Scheduling for Workflow Based Applications in Grids

SungHo Chin[†] · JongHyuk Lee[†] · HeonChang Yu[†]

ABSTRACT

Recently, grid computing environment has being changed to the service-oriented grid integrated with Web Services. In this architecture, there is the service-oriented grid application in which Grid Services are executed according to the workflow of the grid application. This implies that the grid scheduling method needs to consider grid service characteristics. Therefore, we propose the grid service performance model based on factors such as service availability, accessibility, reliability, throughput, and slowdown, and develop service scheduling method to increase performance of the service-oriented grid application.

Keywords : Service Scheduling, Grid Scheduling, Grid Service, Workflow Application

1. 서 론※

그리드 컴퓨팅은 여러 기관들의 자원 공유를 통해 대용량 어플리케이션들이 효율적으로 수행할 수 있는 가상 컴퓨팅 환경이다[1]. 그리드 컴퓨팅은 우수한 어플리케이션 수행 능력으로 인하여 과학 어플리케이션, 기업 어플리케이션 등 다양한 분야에서 이용되고 있다. 그리드 응용분야에서 사용하는 어플리케이션들은 크게 PSA(Parmeter Sweep Application)[2]과 WBA(Workflow Based Application)[3]으로 구별할 수 있다. 이 두 가지의 가장 큰 차이점은 개별 어플리케이션

† 종신회원: 고려대학교 컴퓨터교육학과 박사과정
†† 종신회원: 고려대학교 컴퓨터교육과 교수(교신저자)
논문접수: 2006년 10월 17일, 심사완료: 2006년 11월 18일
※ 이 논문은 2005년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2005-041-D00624)

을 구성하는 부분 작업들 사이의 의존성(dependency)과 이질성(heterogeneity) 존재 여부이다. 즉, PSA는 같은 성질을 가지는 부분작업들로 구성되고 부분작업들 사이에 의존성이 존재하지 않는다. 반면, WBA는 서로 다른 특징을 가지는 부분작업들로 구성되고 이 부분작업들 사이에 의존성이 존재한다. 부분작업들 사이의 의존성은 작업수행 순서를 결정짓게 되는데 WBA에서는 이러한 순서를 워크플로우(workflow)로 표현한다.

대용량 어플리케이션이 그리드 컴퓨팅 환경에서 효과적으로 수행되기 위해서는 부분작업들과 이 작업들이 수행되는 그리드 자원들을 매핑하는 문제를 해결해야 한다. 이러한 문제를 그리드 스케줄링이라 하고 관련된 많은 연구들이 진행되었다. 특히, WBA가 그리드 스케줄링의 대상이 되게 되면 각 부분작업들 사이의 의존성을 유지하기 위해서 부분작업들의 수행 순서를 지켜야 하고 개별 부분작업의 성능보다는 그리드 어플리케이션 전체의 성능을 보장할 수 있는 방법을 개발하여야 한다.

위와 같이 그리드 어플리케이션 형태의 변화와 함께 본 논문에서 중점적으로 다루는 것은 그리드 컴퓨팅 환경의 변화이다. 즉, 그리드 컴퓨팅 환경이 웹 서비스 플랫폼과의 통합을 통한 서비스 기반 그리드 컴퓨팅 환경으로의 변화를 의미한다[5]. 그러나 서비스 기반 그리드 환경은 HTTP, XML 등과 같은 기본적으로 처리 속도가 떨어지는 웹서비스 표준을 따르고 있기 때문에 성능적인 측면에서 단점을 가지고 있다[4]. 이것은 서비스 기반 그리드 환경에서 고성능의 어플리케이션 수행을 보장하기 위해서는 그리드 스케줄링이 더욱 중요하게 다루어져야 함을 의미한다.

따라서 서비스 기반 그리드에서 스케줄링은 그리드 서비스 성능 요소를 고려하여 스케줄링이 이루어져야 한다. 즉, 기존의 전통적인 그리드 스케줄링에서는 CPU 사용률, 가용 메모리 량 등과 같은 하부의 하드웨어 정보를 이용한 것과 달리 서비스 기반 환경에서는 서비스 가용성, 처리율, 신뢰성 등과 같은 상위 정보를 이용하여 스케줄링 해야 한다. 그러므로 서비스 기반 그리드 컴퓨팅 환경에서 그리드 어플리케이션의 효과적인 수행을 위해서는 새로운 스케줄링 기법인 서비스 스케줄링이 필요하다. 이 스케줄링 기법은 워크플로우에 나타나는 각 부분작업을 수행할 수 있는 적절한 그리드 서비스를 선택하고 매핑하여 서비스 기반 그리드 어플리케이션을 구성하는 역할을 한다.

이를 위해, 본 논문에서는 그리드 서비스의 성능에 영향을 줄 수 있는 요소들을 모델링한다. 그리고 모델링된 성능 요소들을 이용하여 서비스 기반 그리드 환경에서 워크플로우 어플리케이션의 효율적인 수행을 지원하는 서비스 스케줄링 기법을 개발한다.

2. 시스템 모델

2.1 서비스 기반 그리드 환경

2.1.1 그리드 시스템

그리드 시스템은 n 개의 그리드 사이트를 포함하고 있다. 각 그리드 사이트들 사이에는 그리드 서비스를 수행하기 위해 필요한 데이터를 전송하기 위한 통신 채널이 존재한다.

- 그리드 시스템 G 는 그리드 사이트들의 집합이다. 즉, $G=\{g_1,g_2,\mathrm{L},g_n\}$, g_i는 그리드 사이트.
- $\forall g_k \in G$에 대해서 g_i와 g_k의 통신비용은 $C_i=\{c_i^k,c_i^{k+1},\mathrm{L},c_i^{k+n-1}\}$ 이다. 만약, g_i와 g_k 사이에 통신 채널이 존재하지 않으면 통신비용은 $c_i^k=\infty$ 이고 $i=k$ 이면, 인트라사이트 통신이 되어서 통신비용은 $c_i^k=0$ 이다.

2.1.2 그리드 어플리케이션

본 논문에서 관심을 가지고 있는 그리드 어플리케이션은 워크플로우 기반 어플리케이션이다. 이 어플리케이션은 서로 다른 수행 목적을 가지는 여러 개의 부분 작업으로 구성된다. 본 논문에서 가정하는 그리드 어플리케이션을 나타내기 위해서 DAG(Directed Acyclic Graph)를 이용한다. DAG는 일반적으로 선후관계를 가지는 부분작업들로 구성되는 워크플로우 어플리케이션을

나타내기 위해 주로 사용되는 모델이다. DAG에 의해 나타나는 그리드 어플리케이션은 단지 수행 계획만을 나타낸다. 즉, 그래프에 나타나는 부분 작업이 서비스 스케줄링에 의해 선택된 그리드 서비스들로 매핑된 상태가 실제 수행을 나타낸다. 이렇게 그리드 서비스들로 구성된 그래프를 서비스 기반 그리드 어플리케이션으로 정의한다.

- 그리드 어플리케이션 A는 DAG $A=(T,D)$로 표현된다.
- $T=\{t_1,t_2,\mathrm{L},t_n\}$ 는 그리드 어플리케이션의 부분작업 집합이다.
- D는 부분 작업들 간의 의존성을 나타내는 집합이다. 만약, t_i와 t_j 사이에 의존성이 존재하면 DAG상에 t_i에서 t_j로 간선이 존재하고 $t_i \xrightarrow{depended} t_j$로 표시한다.

2.1.3 그리드 서비스

본 논문에서 다루는 그리드 서비스는 그리드 사용자에 의해 작업을 수행하기 위해 구현된 그리드 서비스인 사용자 정의 서비스를 의미한다. 그리드 서비스 집합인 S_i은 개별적인 그리드 사이트 g_i에 설치되어 수행되게 된다. 그리고 그리드 서비스 s_i^k는 팩토리 방식(factory fashion)에 의해 수행된다.

- S_i는 그리드 사이트 g_i에 설치된 그리드 서비스들의 집합이다. 즉, $S_i=\{s_i^k,s_i^l,\mathrm{L}\ s_i^m\}$. 그리드 서비스 s_i^k는 그리드 어플리케이션에 포함되는 부분 작업 t_k의 수행 목적을 달성할 수 있다.

2.1.4 서비스 스케줄링

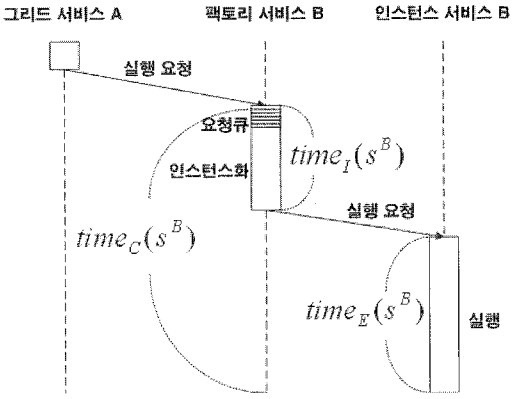
서비스 스케줄링은 그리드 어플리케이션의 DAG, 그리드 사이트와 그리드 서비스들의 성능에 관련된 정보 등을 입력으로 받아들여 정해진 스케줄링 기법을 이용하여 서비스 스케줄 SS_A를 생성한다. 서비스 스케줄 SS_A가 생성되면, 그리드 어플리케이션 A의 서비스 기반 그리드 어플리케이션인 A_S가 생성되고 이것을 구성하는 그리드 서비스들이 워크플로우를 따라 수행된다.

- SS_A는 그리드 어플리케이션 A를 수행하기 위한 그리드 서비스들의 집합이다.
- A_S는 서비스 스케줄링에 의해 생성되는 그리드 어플리케이션 A의 서비스 기반 그리드 어플리케이션이다.

2.2 성능 모델

이 절에서는 그리드 서비스의 성능을 나타낼 수 있는 요소들을 정의하고 그리드 어플리케이션의 성능을 구별할 수 있는 기준을 제시한다.

2.2.1 그리드 서비스 성능 모델



<그림 1> 그리드 서비스의 수행과정

그리드 사이트에 설치된 그리드 서비스 수행에 관한 요청이 전달되면, 이 요청은 해당 그리드 서비스의 팩토리 서비스에게 전달한다. 이 때, 팩토리 서비스는 전달된 메시지를 인스턴스 서비스를 생성하기 위한 자원이 확보될 때 까지 큐잉한다. 팩토리 서비스는 기존의 큐에 있던 요청들이 처리되고 필요한 자원이 확보되면 인스턴스 서비스를 생성하고 이 인스턴스 서비스를 통해서 그리드 작업을 수행하게 된다. <그림 1>은 이 과정을 보여주고 있다.

따라서 그리드 서비스의 수행완료 시간은 식-(1)과 같이 팩토리 서비스에서의 인스턴스화 시간과 인스턴스 서비스의 수행 시간의 합으로 나타낼 수 있다.

$$time_C(s)=time_I(s)+time_E(s) \qquad (1)$$

이와 같은 수행 시간 기반의 그리드 서비스 수행 모델에서 다음과 같은 몇 개의 요소들을 정의

할 수 있다.

▫ 가용성(Availability) : 그리드 서비스의 가용성은 해당 팩토리 서비스가 그리드 서비스의 인스턴스를 생성하기 위한 자원을 확보할 확률로 정의될 수 있다. 만약, 팩토리 서비스가 인스턴스를 생성하기 위한 CPU 사이클과 메모리를 획득할 확률이 $P(require\_cpu(s_i^j))$, $P(require\_mem(s_i^j))$ 이라면, 그리드 서비스 s_i^j 의 가용성은 식-(2) 와 같이 나타낼 수 있다.

$$avai(s_i^j) = P(require\_cpu(s_i^j)) \cdot P(require\_mem(s_i^j)) \cdot avai(g_i) \quad (2)$$

▫ 접근성(Accessibility) : 그리드 서비스의 접근성은 그리드 서비스 즉시 수행 정도로 정의될 수 있다. 따라서 인스턴스화 시간인 $time_I(s_i^j)$와 관련이 있다. 그리드 서비스 s_i^j 의 접근성은 식-(3)과 같다.

$$acce(s_i^j) = \frac{1}{time_I(s_i^j)} \quad (3)$$

▫ 신뢰성(Reliability) : 그리드 서비스의 신뢰성은 해당 그리드 서비스의 수행 요청 수에 대한 그리드 사이트의 결함이나 재스케줄링 등으로 인한 수행 중단 수의 비율로 나타낸다. 그리드 서비스 s_i^j 의 수행 요청 수와 수행 중단 수를 각각 N_r, N_f 라고 하면 신뢰성은 식-(4) 와 같다.

$$reli(s_i^j) = 1 - \frac{N_f}{N_r} \quad (4)$$

▫ 처리율(Throughput) : 그리드 서비스의 처리율은 수행 시간과 관련이 있다. 그리드 서비스 s_i^j 의 처리율은 식-(5)와 같이 나타낼 수 있다.

$$thro(s_i^j) = \frac{1}{time_E(s_i^j)} \quad (5)$$

▫ 감속비율(Slowdown Ratio) : 그리드 서비스의 감속비율은 평균 완료 시간과 평균 인스턴스화 시간과 관련이 있다. 이것은 서비스가 수행될 때 팩토리 서비스에서 받을 수 있는 혼잡 비용을 나타낸다. 그리드 서비스 s_i^j 의 감속비율은 식-(6)과 같다.

$$slow(s_i^j) = \frac{\sum_{n=1}^{N_r} time_C^n(s_i^j)/N_r}{\sum_{n=1}^{N_r} time_E^n(s_i^j)/N_r} \quad (6)$$

2.2.2 그리드 어플리케이션 성능 모델

그리드 사이트에서 수행되는 개별적인 부분 작업에 대한 기대 수행 시간에 대한 예측치는 미리 알고 있으며 ETC(Expected Time to Compute) 테이블에 저장되어 있다. 이러한 가정은 이질적인 컴퓨팅 환경에서 작업의 매핑이나 스케줄링에 관한 연구[6,7,8]를 할 때 일반적으로 받아들여지는 가정이다. 따라서 본 연구에서도 부분 작업에 해당하는 그리드 서비스가 수행되는 수행 시간은 미리 예측치를 알고 있다는 가정을 가진다. 그리고 두 개의 그리드 서비스가 동일한 그리드 사이트에서 동시에 수행될 수 없음을 가정한다. 본 논문에서는 이러한 가정들 하에서 전체 그리드 어플리케이션의 성능을 측정하기 위해서 다음과 같은 몇 개의 값들을 정의한다.

▫ 그리드 사이트 g_j 에서 부분 작업 t_i 의 계산 비용은 그리드 서비스 s_j^i 의 완료 시간과 같다. 이것은 식-(7)과 같이 계산된다.

$$time_C(s_j^i) = time_E(s_j^i) \cdot slow(s_j^i) \quad (7)$$

▫ 부분 작업 t_i 와 t_j 사이의 통신비용은 두 부분 작업에 대해 각각 그리드 서비스 s_m^i 와 s_n^j 가 스케줄 되었다고 한다면 식-(8)과 같이 계산된다.

$$time_M(s_m^i, s_n^j) = d_i^j \cdot c_m^n \quad (8)$$

여기서 d_i^j 는 부분 작업 t_i 에서 t_j 로 전송되는 데이터의 양을 나타낸다.

▫ 그리드 사이트 g_n 에서 부분 작업 t_i 의 최초 시작시간(Earliest Start Time)은 식-(9)와 같다.

$$EST(t_i, g_n) = \max\{ready(s_n^i), \max_{s_m^h \in depend(s_n^i)}(time_C(s_m^h) + time_M(s_m^h, s_n^i))\} \quad (9)$$

여기서 $ready(s_n^i)$ 는 그리드 서비스 s_n^i 의 인스턴

스가 생성되고 수행이 준비되었을 때의 시간을 나타난다. $depend(s_n^i)$은 그리드 서비스 s_n^i와 의존성을 가지고 있는 그리드 서비스들의 집합을 나타낸다. 즉, $depend(s_n^i)=\{s_m^h \mid t_h \xrightarrow{depended} t_i \wedge c_m^n \neq \infty\}$

▫ 그리드 사이트 g_n에서 부분 작업 t_i의 최초 종료시간(Earliest Finish Time)은 식-(10)과 같다.

$$EFT(t_i,g_n)=EST(t_i,g_n)+time_c(s_i^n) \quad (10)$$

▫ 만약 그리드 어플리케이션의 DAG 상에서 말단 부분작업 t_z이 그리드 사이트 g_n에서 수행된다면, 그리드 어플리케이션 A의 전체 성능은 식-(11)과 같이 계산된다.

$$OP(A)=EFT(t_z,g_n) \quad (11)$$

3. 그리드 서비스를 위한 리스트 스케줄링

서비스 스케줄링은 그리드 어플리케이션의 부분작업들을 수행할 수 있는 그리드 서비스를 선택하고 각각의 그리드 사이트에서 선택된 그리드 서비스들의 적당한 수행 순서를 정하는 과정이다. 이러한 서비스 스케줄링의 목적은 그리드 어플리케이션의 전체의 성능 향상이다. 예를 들면, 그리드 어플리케이션의 말단(terminal) 부분 작업이 끝나는 시간을 최소화하는 것을 목적으로 할 수 있다. 일반적으로, 워크플로우 기반 어플리케이션에 대한 스케줄링 문제는 *NP-complete* 으로 알려져 있기 때문에 본 논문에서는 휴리스틱을 이용하여 서비스 스케줄링을 시행한다. 본 논문에서 사용하는 휴리스틱은 리스트 스케줄링 휴리스틱이다. 이 휴리스틱은 워크플로우 기반 어플리케이션의 스케줄링에서 사용되는 다른 휴리스틱에 비해서 뛰어난 성능을 발휘한다[9]. 본 논문에서 제시하는 그리드 서비스를 위한 리스트 스케줄링(LSS(List Scheduling for Service))※은 가중치 설정 단계, 랭킹 단계, 선택 단계로 구성된다. 각 단계에 대한 자세한 내용은 다음의 각 소절에서 설명한다.

※ 이하 LSS 로 칭함

3.1 가중치 설정 단계

이 단계에서는 스케줄링 하게 되는 그리드 어플리케이션의 DAG에 나타나는 부분 작업들과 부분작업들 사이의 간선들에 대해서 가중치를 설정한다. 부분 작업에 대한 가중치는 부분 작업의 수행 시간을 이용하여 계산한다. 실제적으로 각 부분 작업에 대해 올바른 가중치 적용은 전체 스케줄링 알고리즘의 성능에 영향을 줄 수 있기 때문에 매우 중요하다. 본 논문에서는 서비스 기반 그리드 컴퓨팅의 개념을 반영하기 위해서 그리드 서비스의 가용성과 신뢰성을 이용하여 그리드 서비스의 가중치를 설정한다. 또한 부분 작업의 계산비용을 이용한다. 부분 작업의 가중치는 식-(12)와 같이 계산된다.

$$w(t_n)=\frac{\sum_{g_i \in G^n} time_C(s_i^n)/|G^n|}{\sum_{g_i \in G^n} avai(s_i^n)\cdot reli(s_i^n)/|G^n|} \quad (12)$$

G^n은 그리드 서비스 s_i^n가 설치된 그리드 사이트를 의미하고 s_i^n은 부분 작업 t_n의 실행 목적을 달성할 수 있는 그리드 서비스를 의미한다. 즉, $G^n=\{g_i \mid s_i^n \in S_i\}$ 이다.

각 간선들에 대한 가중치는 식-(13)과 같이 계산된다.

$$w(t_i,t_j)=\frac{\sum_{(g_k,g_l)\in d(t_i,t_j)} time_M(s_k^i,s_l^j)}{|d(t_i,t_j)|} \quad (13)$$

$d(t_i,t_j)$는 $d(t_i,t_j)=\{(g_k,g_l) \mid g_k \in G^i \wedge g_l \in G^j \wedge C_k^l \neq \infty\}$을 의미한다. <그림 2>는 LSS 알고리즘의 가중치 설정 단계를 나타낸다.

Algorithm 1. LSS algorithm-Weighting Phase		
1	*for* each $t_n \in T$ *do*	
2		calculate $w(t_n)$
3	*end*	
4	*for* each $(t_i,t_j) \in D$ *do*	
5		calculate $w(t_i,t_j)$
6	*end*	

<그림 2> 가중치 설정 알고리즘

3.2 랭킹 단계

서비스 스케줄링의 두 번째 단계인 랭킹 단계는 특정 랭킹값에 따라서 DAG의 부분 작업들을 내림차순으로 정렬하게 된다. 여기서 랭킹값이 높은 부분 작업은 랭킹값이 낮은 부분 작업에 비해 먼저 스케줄링을 위해 다루어지게 된다. 만약 같은 랭킹값을 갖는 부분 작업들이 생기게 되면 임의적으로 순서를 정한다. 이렇게 생성된 리스트는 DAG에 나타나는 부분 작업들 사이의 의존성을 유지하고 있다. 본 논문에서는 부분작업의 랭킹값을 구하기 위해서 *b-level*[10]을 이용한다. 부분작업의 *b-level*은 해당 부분작업에서 말단 부분작업까지의 최대 경로값을 의미한다. 따라서 *b-level*값은 DAG 그래프의 임계경로에 의해 제한된다. 본 논문에서 *b-level*을 랭킹값으로 이용하기 때문에 랭킹 단계에서는 워크플로우 그래프를 말단 부분 작업에서부터 위쪽으로 탐색하면서 이루어진다. 랭킹값은 식-(14)와 같다.

$$RV(t_n) = w(t_n) + \max_{t_m \in depneded(t_n)} \{w(t_n, t_m) + RV(t_m)\} \quad (14)$$

여기서 $depended(t_n)$는 부분 작업 t_n에 후행해서 수행되고 t_n과 의존성을 가지는 부분 작업들의 집합을 나타낸다. <그림 3> 은 LSS 알고리즘의 랭킹 단계를 나타낸다.

Algorithm 2. LSS algorithm-Ranking Phase

```
7    T_new = T
8    while T_new ≠ {} do
9        T_tmn = {t_n | depended(t_n) = φ}
10       for each t_n ∈ T_tmn do
11           calculate RV(t_n)
12       end
13       T_new = T_new − T_tmn
14   end
15   for all t_n ∈ T sort T_list in decreasing
     order of RV(t_n)
```

<그림 3> 랭킹 단계 알고리즘

3.3 선택 단계

LSS 알고리즘의 마지막 단계는 선택 단계이다. 이 단계에서는 랭킹 단계에서 형성된 리스트에 나타나는 연속적인 부분작업에 대해서 적절한 그리드 서비스를 선택한다. 본 논문에서는 전체적인 그리드 어플리케이션의 성능을 보장하기 위해서 선택값(selection value)을 정의하고 사용한다. 특정 그리드 서비스가 선택되게 되면, 그 서비스의 자손 서비스들의 최초시작시간에 영향을 미칠 수 있으므로 선택값은 선택되는 그리드 서비스와 의존성을 가지고 있는 자손 서비스들 사이의 통신비용을 고려하여야 한다. 그리고 이들 자손 서비스들의 계산 비용 또한 고려되어야 한다. 따라서 본 논문에서의 선택값은 식-(15)와 같이 계산된다.

$$SV(t_i, g_n) = EFT(t_i, g_n) + \max_{t_j \in depend(t_i)} \{\min_{g_m \in G^j} \{time_M(s_n^i, s_m^j) + time_C(s_m^j)\}\} \quad (15)$$

그리드 어플리케이션을 구성하는 각각의 부분작업에 대해서 가장 작은 선택값을 가지는 그리드 서비스가 선택된다. 만약 말단 부분작업에 대해 그리드 서비스가 선택되면 서비스 스케줄링에 의해 개별 부분 작업의 역할을 수행할 수 있는 그리드 서비스들로 구성된 서비스 기반 그리드 어플리케이션이 생성된다. 이 서비스 기반 그리드 어플리케이션은 전체 그리드 어플리케이션의 수행 성능을 보장하게 된다. <그림 4>는 LSS 알고리즘의 선택 단계를 나타낸다.

Algorithm 3. LSS algorithm-Selection Phase

```
16   SS_A = φ
17   for each t_k ∈ T_list do
18       Find g_n ∈ G^k having Minimum
         SV(t_i, g_n)
19       SS_A = SS_A Y {s_n^k}
20   end
```

<그림 4> 선택 단계 알고리즘

4. 모의실험 및 결과

이 장에서는 본 논문에서 LSS 알고리즘을 다른 스케줄링 기법들과 비교하여 평가하게 된다. 이를 위해 본 논문에서는 실험 결과에 영향을 미칠 수 있는 다양한 파라미터들을 정의하고 이 값

들을 변경하면서 실험하여 실험의 신뢰성을 높였다.

4.1 모의실험 설계

LSS에 대해서 다양한 그리드 어플리케이션의 DAG를 바탕으로 모의 실험하기 위해서 DAG 그래프의 구조에 영향을 미치는 파라미터 값을 변화시키면서 다양한 DAG 그래프를 생성한다. 이러한 파라미터들은 다음과 같다.

- *TaskNumber* : DAG 그래프에 포함되는 부분작업의 개수를 의미한다.
- *HostNumber* : 그리드 시스템을 구성하고 있는 그리드 사이트의 개수를 의미한다.
- *HeightRatio* : DAG 그래프의 높이를 정의하는 값이다. 즉, n개의 부분작업을 가지는 그래프의 높이는 $\sqrt{n} \cdot heightRatio$ 와 같이 계산된다.
- *MaxEdge* : 개별 부분 작업이 가질 수 있는 간선의 최고수를 나타낸다.
- *CommRatio* : 그리드 어플리케이션의 평균 계산 시간과 평균 통신 시간의 비율을 나타낸다. 만약, 특정 그리드 어플리케이션의 평균 계산 시간이 평균 통신 시간에 비해 크다면 그 어플리케이션은 계산 집약적인 특징을 가지고 반대의 경우라면 데이터집약적인 특징을 가진다.

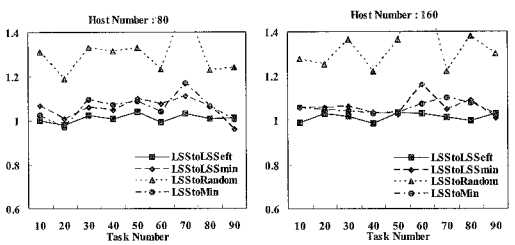
위에서 정의한 파라미터들을 통해 생성된 그리드 어플리케이션의 DAG에 대해서 본 연구에서 제시한 스케줄링 기법의 성능을 비교하기 위해 다음과 같은 스케줄링 기법을 이용한다.

- LSSeft : 이 기법은 본 논문의 기법과 유사하나 선택 단계에서 본 논문에서 제시하는 선택값이 아닌 최초종료시간을 이용하게 된다.
- LSSmin : 이 기법은 본 논문의 기법과 유사하나 적절한 그리드 서비스를 선택하기 위해서 MinMin 휴리스틱을 이용하게 된다.
- Min : 이 기법은 리스트 스케줄링 기법이 아니다. 따라서 가중치 설정단계와 랭킹 단계가 없이 적절한 그리드 서비스를 그리디(greedy)한 방법으로 선택하게 된다.
- Random : 본 논문에서 제시하는 성능 모델을 사용하지 않고 임의로 그리드 서비스를 선택하는 기법이다.

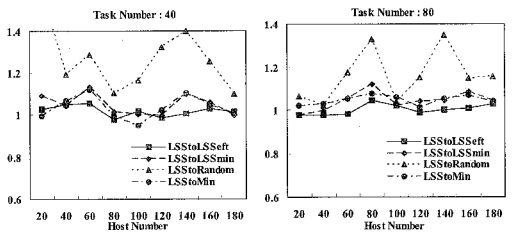
본 논문에서는 실험의 신뢰성을 향상시키기 위해서 각 파라미터 튜플(tuple)에 대해서 1000번의 실험을 하고 그 결과값들의 평균을 이용하여 성능 비교를 하게 된다. 그리고 각 기법들과의 성능 차이를 나타내기 위해서 제시하는 스케줄링 기법과 비교 기법들 사이의 비율을 이용하게 된다. 만약 이 비율이 '1' 보다 크게 되면, 제시하는 스케줄링 알고리즘이 좋은 성능을 발휘하는 것이고 '1'보다 작으면 그 반대를 의미한다.

4.2 실험 결과

부분 작업의 개수에 대해 제시한 알고리즘의 성능을 측정하기 위해서 *TaskNumber* 파라미터를 변화시키면서 모의 실험하였다. 이 모의 시험의 결과값은 <그림 5>와 같다. <그림 5>에서 LSS 알고리즘은 부분작업의 개수에 상관없이 우수한 성능을 나타낸다.

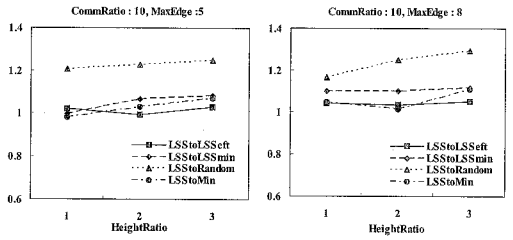


<그림 5> *TaskNumber*의 변화에 따른 실험결과

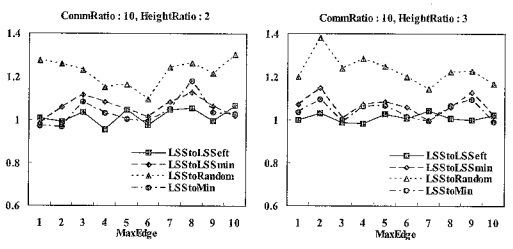


<그림 6> *HostNumber*의 변화에 따른 실험결과

<그림 6>은 그리드 사이트의 개수에 상관없이 LSS 알고리즘의 성능이 우수하다는 것을 보여주고 있다.

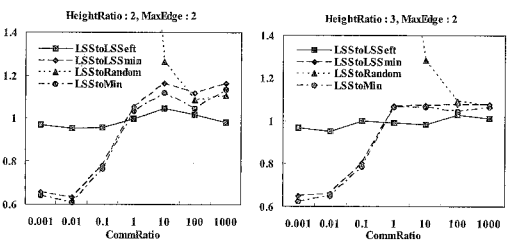


<그림 7> *HeightRatio* 변화에 따른 실험결과



<그림 8> *MaxEdge* 변화에 따른 실험결과

부분 작업의 개수나 호스트의 개수뿐 아니라 그리드 어플리케이션의 DAG 그래프의 구조가 스케줄링 기법에 영향을 미치기 때문에 그래프의 구조에 영향을 미칠 수 있는 파라미터인 *HeightRatio*, *MaxEdge* 를 변화시키면서 모의 실험하였다. <그림 7>과 <그림 8>은 이러한 실험의 결과값을 보여주고 있다. <그림 7>에서 LSS 알고리즘은 *HeightRatio* 가 높을수록 우수한 성능을 발휘하고, <그림 8>에서는 전체 *MaxEdge* 값에 대해서 일관성 있는 성능을 나타내고 있다. 이것은 제시하는 스케줄링 기법이 DAG 그래프의 복잡도와 상관없이 우수한 성능을 발휘할 수 있음을 의미한다.



<그림 9> *CommRatio* 변화에 따른 실험결과

마지막 실험은 그리드 어플리케이션의 특성에 따른 LSS 알고리즘의 성능을 측정하였다. 이를 위해 *CommRatio* 값을 변화시키면서 모의 실험하였다. <그림 9>는 그 결과값을 보여주고 있다. <그림 9>에서 알 수 있듯이 제시하는 스케줄링 기법은 그리드 어플리케이션의 특징이 데이터 집약적일수록 우수한 성능을 발휘함을 알 수 있다.

전체적으로 볼 때, LSS 알고리즘이 LSSeft에 비해 우수한 성능을 나타내었다. 이것은 본 논문의 선택값이 선택된 그리드 서비스와 나머지 그리드 서비스들 사이의 의존성을 잘 반영하고 있음을 의미한다. 그리고 LSSmin에 대해 우수한 성능을 나타내는 것은 본 논문에서 제시하는 선택 단계가 MinMin 선택 기법 등에서 발생할 수 있는 복잡한 선택 기법을 거치지 않고 효율적으로 그리드 서비스를 선택하고 있음을 의미한다. 나머지 두 가지 기법 즉, Random 과 Min 기법에 대해서 우수한 성능을 나타내는 것은 본 논문의 랭킹 단계가 올바르게 설계되었음을 의미한다.

5. 관련 연구

그리드 어플리케이션의 부분작업들을 여러 개의 이질적인 그리드 사이트에 매핑하는 문제는 매우 어려운 문제이다. 실제적으로 최적 멀티프로세서 스케줄링 문제가 NP-complete이기 때문에 위와 같은 문제는 NP-complete하다[11]. 그래서 대부분의 그리드 스케줄링에 관한 연구는 우수한 휴리스틱을 다루고 있다. 지난 몇 년간은 독립적인 부분 작업들을 멀티프로세서에 매핑하는 많은 휴리스틱들이 연구되었다[7,8]. 그리고 이와 함께 PSA를 위해서 데이터의 이동성을 고려하는 휴리스틱 또한 제안되었다[2]. 그러나 이 기법은 본 연구에 그대로 적용되지 않는다. 왜냐하면 본 연구는 WBA에 초점을 맞추고 있기 때문이다. PSA와 마찬가지로 WBA에서의 휴리스틱 스케줄링 기법 또한 연구되었다[6,10]. 그러나 이러한 기법들은 동질 시스템을 바탕으로 연구가 진행되어서 이질의 그리드 시스템을 바탕으로 하고 있는 본 연구와는 차별성을 가진다. 본 연구와 비슷한 연구로는 이질 시스템에서 WBA에 대한 리스트 스케줄링 기법에 관한 연구가 있다[12]. 이 연구에서 소개하는 HEFT(Heterogeneo-

us Earliest Finish Time) 알고리즘은 다른 알고리즘들에 비해 우수한 성능을 발휘하지만 전체적인 그리드 어플리케이션의 성능을 보장하지 못하는 경우가 발생할 수 있다[13].

위와 같은 연구들과 함께 그리드 환경에서 WBA에 관한 두 가지 비슷한 스케줄링 기법에 관한 연구가 진행되었다[3,9]. 그러나 이 연구들은 서비스 기반 그리드 환경을 고려하고 있지 않다. 일반적으로, 서비스 기반 그리드 환경은 성능적인 측면에서 단점을 가지게 되는데[4], 이를 극복하기 위해서는 그리드 서비스 성능 요소[14]를 고려하여야 한다. 비록, 서비스 기반 그리드 개념을 이용한 스케줄링 기법이 제시되기는 하였으나 이 기법은 WBA를 고려하지 않고 있다[15]. 따라서 본 연구에서는 서비스 기반 그리드 환경에서 WBA를 고려하는 새로운 그리드 스케줄링 기법을 제시하였다. 제시한 LSS 기법은 그리드 서비스의 성능 요소를 고려하였고 그리드 어플리케이션 전체에 대해서 우수한 성능을 보장할 수 있다.

6. 결 론

최근 들어, 그리드 컴퓨팅 환경이 서비스 기반 컴퓨팅 환경으로 변화하고 워크플로우 그리드 어플리케이션에 대한 관심이 증대되고 있다. 이러한 상황에서, 그리드 어플리케이션 전체의 성능 향상을 위해서 그리드 어플리케이션의 부분작업을 적절한 그리드 서비스에 매핑하는 문제는 매우 중요한 문제이다.

따라서 본 논문에서는 이러한 문제를 해결하기 위해서 그리드 서비스의 성능 요소 즉, 그리드 서비스 가용성, 접근성, 신뢰성, 처리율, 감속비율 등을 모델링 하였다. 그리고 그리드 서비스를 위한 새로운 스케줄링 알고리즘 기법을 개발하였다. 이 알고리즘은 리스트 스케줄링 기법과 그리드 서비스 성능 요소에 바탕을 두어 워크플로우 기반의 그리드 어플리케이션을 효율적으로 수행할 수 있는 서비스 기반 그리드 어플리케이션을 생성하게 된다. 제시한 알고리즘의 성능 평가를 위해서 다양한 모의 실험을 실시하였다. 모의 실험 결과에서 알 수 있듯이, LSS 알고리즘은 서비스 기반 그리드 환경을 잘 반영하고 다른 스케줄링 기법에 비해 우수한 성능을 발휘하였다.

향후 연구 과제로는 제시한 알고리즘에 대한 좀 더 다양한 실험과 성능개선 방안에 대한 연구가 진행될 것이다. 그리고 서비스 스케줄러의 개발을 통해 실제적인 워크플로우 기반 그리드 어플리케이션에 적용해 볼 계획이다.

참 고 문 헌

[1] I. Foster, C. Kesselman and S. Tuecke(2001), The Anatomy of the Grid : Enabling Scalable Vir-tual Organizations, *International J. Supercomputer Applications*.

[2] Casanova, H., Legrand, A., Zagorodnov, D., Berman, F.(2000), Heuristics for Scheduling Parameter Sweep Applications in Grid environments. *9th Heterogeneous Computing Systems Workshop*,

[3] Blythe, J. Jain, S.; Deelman, E.; Gil, Y.; Vahi, K.; Mandal, A. Kennedy, K.(2005), Task scheduling strategies for workflow-based applications in grids, *IEEE International Symposium on Cluster Computing and the Grid 2005(CCGrid 2005), Vol. 2*, May 759- 767.

[4] Taiani, F.; Hiltunen, M.; Schlichting, R.(2005), The impact of Web service integration on grid performance, *IEEE International Symposium on High Performance Distributed Computing, 2005.(HPDC-14)*, July 14- 23.

[5] Foster, I., Kesselman, C., Nick, J., Tuecke, S.(2002), The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.globus.org/research/papers.html .

[6] T. D. Braun, H. J. Siegel, and A. A. Maciejewski(2002), Static mapping heuristics for tasks with dependencies, priorities, deadlines, and multiple versions in heterogeneous environments, *16th International*

Parallel and Distributed Processing Symposium, Apr. .

[7] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, R. F. Freund(1999), Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems, *Eight Heterogeneous Computing Workshop, IEEE Computer Society Press*, 30-44.

[8] T. D. Braun, H. J. Siegel et al.(2001), A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, *J. Parallel and Distributed Computing, Vol. 61*, 810-837.

[9] Anirban Mandal; Kennedy, K.; Koelbel, C.; Marin, G.; Mellor-Crummey, J.; Liu, B.; Johnsson, L(2005), Scheduling strategies for mapping application workflows onto the grid, *14th IEEE International Symposium on High Performance Distributed Computing, 2005(HPDC-14)*, July 125-134.

[10] Y.-K. Kwok and I. Ahmad(1999). Benchmarking and comparison of the task graph scheduling algorithms. *J. Parallel and Distributed Computing, Vol. 59, No. 3,* Dec. 381-422.

[11] Garey, M. and Johnson, D.(1979), Computers and Intractability: A Guide to the Theory of NP-Completeness, *W. H. Freeman and Company*, New York.

[12] H. Topcuoglu, S. Hariri, M.Y. Wu(2002), Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing, *IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 3*, Mar. 260-274.

[13] Zhiao Shi and Jack J. Dongarra(2006), Scheduling workflow applications on processors with different capabilities, *Future Generation Computer Systems, Vol. 22, Issue 6*, May 665-675.

[14] Chuan He, Bin Du, Sanli Li (2003), Grid Services Performance Tuning in OGSA, *Lecture Notes in Computer Science, Volume 2834*, Sep 373 - 381.

[15] Jingbo Ding; WeiqinTong(2003), A service scheduling system in service grid environment, *IEEE International Conference on Industrial Informatics(INDIN 2003)*, Aug. 413-418.

진 성 호

2002 고려대학교 컴퓨터교육과 (이학사)
2004 고려대학교 컴퓨터교육과 (교육학석사)
2004~현재 고려대학교 컴퓨터교육학과 박사과정
관심분야: 그리드 컴퓨팅, 분산 시스템, SOA
E-Mail: wingtop@comedu.korea.ac.kr

이 중 혁

2004 고려대학교 컴퓨터교육과 (이학사)
2006 고려대학교 컴퓨터교육과 (이학석사)
2006~현재 고려대학교 컴퓨터교육학과 박사과정
관심분야: 그리드 컴퓨팅, 분산 시스템
E-Mail: spurt@comedu.korea.ac.kr

유 헌 창

1989 고려대학교 전산과학과 (이학사)
1991 고려대학교 전산과학과 (이학석사)
1994 고려대학교 전산과학과(이학박사)
1996~현재 고려대학교 컴퓨터교육과 교수
관심분야: 그리드 컴퓨팅, 분산 시스템, 결함포용, 에이전트 시스템, 모바일 컴퓨팅, 컴퓨터교육
E-Mail: yuhc@comedu.korea.ac.kr