그리드 시스템을 위한 포인트 기반 스케줄링 알고리즘

(A Point-based Scheduling Algorithm for GRID Environment)

오 영 은 † 김 진 석 ††
(Young-Eun Oh) (Jin Suk Kim)

요 약 과학 기술이 발전함에 따라 대량의 정보를 분석하고 처리하기 위해 대두된 그리드 시스템은 이질적인 시스템 위에 구축된 환경이므로, 사용자 작업을 효율적으로 할당하기 위한 스케줄링 알고리즘이 필요하다. 현재까지 여러 스케줄링 알고리즘이 연구되어 왔지만, 대부분 자원 사이의 네트웍 대역폭을 고려하지 않는 문제점을 가지고 있다. 본 논문에서는 이러한 문제점을 해결하기 위하여 글로벌 그리드 시스템에서의 스케줄링 알고리즘을 제안하였다. 또한 글로벌 그리드 시스템에서 사용되는 여러 알고리즘의 점수를 각각 계산하여 가장 점수가 낮은 알고리즘을 선택하는 포인트 기반 스케줄링 알고리즘을 제안하였으며, 시뮬레이션을 통하여 다른 스케줄링 알고리즘보다 성능이 뛰어남을 보였다.

키워드 : 그리드, 병렬 알고리즘

Abstract GRID environments have been developed in distributed heterogeneous computing infrastructure for advanced science and engineering. Therefore efficient scheduling algorithms for allocating user job to resources in the GRID environment are required. Many scheduling algorithms have been proposed, but these algorithms are not suitable for the GRID environment. That is the previous scheduling algorithm does not consider network bandwidth between multiple resources. In this paper, we propose a new scheduling algorithm for Global GRID environment and show that our algorithm has better performance than other scheduling algorithms through extensive simulation.

Key words : GRID, Parallel Algorithm

1. 서 론

최근 과학 기술이 발전함에 따라 여러 복잡한 문제를 해결하기 위하여 고성능의 계산 자원이 필요하게 되었다. 이러한 요구를 충족하기 위하여 최근 지역적으로 분산되어 있는 이질적인 고성능 컴퓨팅 자원을 하나로 묶어 거대한 시스템을 구성하는 그리드(GRID)[1]에 대한 연구가 활발하게 이루어지고 있다. 그리드는 지리적으로 분산되어 있는 고성능의 컴퓨터, 대용량 저장장치, 첨단 과학 설비들을 초고속 네트워크로 연결하여 병렬 분산 처리 환경을 구성하기 위하여 연구되고 있는 프로젝트이다. 그리드는 이질적인 시스템 위에 구축된 환경이므로, 사용자 작업을 실행시키기 위하여 효율적인 자원 선택이 필요하다. 이러한 스케줄링 문제는 NP-Complete로 알려져 있으며[4], 현재까지 여러 스케줄링 알고리즘이 연구되어 왔다[2,3,5]. 그러나 대부분 작업의 계산량을 중심으로 다루어 지역적으로 분산된 그리드 시스템에 적합하지 않는 문제점을 가지고 있다. 본 논문에서 다루고자 하는 작업들은 작업 간의 의존성이 존재하지 않는 독립적인 작업들이며, 사용자 작업이 실행되는 자원들을 선택하기 위하여 계산 시간과 네트웍 시간을 구분하였다. 그리고 여러 알고리즘을 평가하기 위한 포인트 기반 알고리즘을 제안하여 기존의 스케줄링 방식에 비해 성능이 향상되었음을 보였다. 본 논문의 순서는 다음과 같다. 제 2장에서는 알고리즘을 기술하고 평가하는데 필요한 용어를 설명하고 기존의 스케줄링 방법에 대하여 설명하였다. 제 3장에서는 글로벌 그리드 시스템에서의 스케줄링 알고리즘을 제안하였으며, 제 4장에서는 포인트 기반 스케줄링 알고리즘을 제안하여 이를 시뮬레이션을 통하여 성능을 평가하였다.

· 이 논문은 2000년도 서울시립대학교 학술연구용 첨단장비사업 지원에 의하여 연구되었으며, 이에 감사드립니다.
† 학생회원 : 로커스테크놀로지스
iceize@naver.com
†† 종신회원 : 서울시립대학교 컴퓨터과학부 교수
jskim@venus.uos.ac.kr
(Corresponding author임)
논문접수 : 2004년 9월 23일
심사완료 : 2005년 8월 19일

2. 관련용어 및 연구

그리드 시스템은 여러 이질적인 시스템으로 이루어진 환경이다. 이질적인 시스템(heterogeneous system)은 자원의 특성과 성능이 다른 자원들로 구성된 시스템을 말한다. 본 논문에서는 알고리즘의 성능을 평가하기 위하여 완료 시간과 makespan[7]을 사용하였는데, makespan은 시스템에 주어진 작업 집합 $T = \{T_1, T_2, ..., T_n\}$에서 T_n이 완료되는 시간을 말한다. 또한 작업 집합을 구성하기 위하여 유휴 시간(idle time)을 이용하였는데, 유휴 시간이란 작업 집합 T에서 T_{n-1} 작업이 제출되고 난 후 T_n 작업이 제출될 때까지의 시간을 나타낸다.

이질적인 환경에서 독립적인 사용자 작업을 수행하기 위한 스케줄링 알고리즘은 크게 정적 스케줄링 알고리즘과 동적 스케줄링 알고리즘으로 나뉘어진다[8]. 정적 스케줄링 알고리즘은 작업의 개수와 선후 관계 및 자원의 상태 등이 고정되어 있을 때 사용되는 방법이며, 동적 스케줄링 알고리즘은 자원의 상태가 유동적일 때 사용되는 방법이다. 또한 동적 스케줄링 알고리즘은 스케줄링을 언제 수행하는지에 따라 온라인 방식과 배치 방식의 알고리즘으로 나눌 수 있다. 온라인 방식의 스케줄링 알고리즘이란 사용자 작업을 큐에 저장하지 않고 작업이 도착하는 즉시 스케줄링 하는 방식을 말한다. 대표적인 온라인 방식의 스케줄링 알고리즘으로는 MET (Minimum Execution Time), MCT(Minimum Completion Time), SA(Switching Algorithm) 등이 있다[5]. MET는 작업의 실행시간의 기대 값이 가장 작은 자원을 찾아 그 자원에 작업을 할당한다. 즉, T_i를 스케줄링할 때, 자원 R에서 실행시간 e_{ij}가 가장 작은 자원을 찾아 T_i에 할당한다. MET는 실행시간이 가장 작은 자원을 작업에 할당하기 때문에 자원사용의 불균형을 초래할 수 있다. MCT는 해당 작업에 대하여 완료시간이 가장 작은 자원을 작업에 할당한다. 즉, T_i을 스케줄링할 때 그 작업에 대하여 R에서 완료시간 ct_{ij}가 가장 작은 자원을 T_i에게 할당한다. MCT는 간단하면서도 좋은 성능을 보여주며 이 분야에 관한 연구에서 새로운 알고리즘을 제안할 때 비교하는 대상으로 많이 이용되고 있다. SA 알고리즘은 최대준비시간과 최소준비시간의 비율에 대한 한계 값을 적용하여 MET와 MCT를 교대로 적용하는 방법이다. 그러나 위의 알고리즘에서는 사용자 작업마다의 기대 실행시간과 기대 완료시간을 알고 있다고 가정하였으며, 네트워을 이용하는 병렬처리 환경이 아니므로 그리드 시스템에 적합하지 않다.

그리드 시스템에서의 스케줄링 알고리즘으로는 자원의 상태를 고려한 K-Windows 알고리즘과[6], 사용자 작업에 대한 비용 최적화 및 시간 최적화 알고리즘이 있다[7]. K-Windows 알고리즘은 그리드 시스템에 있는 모든 자원으로부터 1부터 k개까지의 시스템 정보를 가져온 후, CPU 부하량, 사용 가능한 메모리, 실행되고 있는 프로세스의 개수 순으로 우선 순위를 정하여 스케줄링하는 방식이다. 비용 최적화 및 시간 최적화 알고리즘은 자원의 빌링 시스템을 이용하여 사용자가 작업을 수행하고자 할 때 비용적인 측면과 시간을 고려한 스케줄링 방법이다. 그러나 K-Windows 알고리즘의 경우 네트웍에 대한 고려를 하지 않았으며, 비용 최적화 및 시간 최적화 알고리즘의 경우 자원에 대한 빌링 및 자원의 위치(지역 간의 시차)를 고려한 알고리즘이므로 관점이 다르다.

3. 글로벌 그리드 시스템에서의 스케줄링 알고리즘

3.1 스케줄링 알고리즘의 요구 조건

글로벌 그리드 시스템은 캠퍼스 그리드 시스템과 달리 WAN 환경으로 이루어져 있다. 또한 대부분의 MPI 응용 프로그램은 네트웍의 대역폭에 민감하게 반응하므로[10], 네트웍에 대한 고려가 반드시 필요하다. 본 논문에서는 글로벌 그리드 시스템에서의 스케줄링 알고리즘을 제안하기 위하여 다음과 같은 가정을 세웠다.

- 클러스터는 동일 아키텍처로 구성되어 있음
- 자원 간의 네트웍 대역폭을 알고 있으며, 값은 실시간으로 변하지 않는다고 가정함
- 사용자 작업 파일 및 실행 결과의 전송 시간을 고려하지 않음
- 기대 실행 시간에 대한 프로파일이 제공되지 않음
- 각 클러스터의 지역 작업 관리자는 단일 큐를 사용함

또한 자원의 상태는 다음 파라미터를 갖는다고 가정하였다.

- 사용자 작업: 요구 CPU 시간, 요구 노드 수, 요구 네트웍 대역폭(메시지 전송량), 분산 작업에 대한 동기화 여부
- 자원: CPU 속도, 전체 노드 수, 사용 가능한 노드 수
- 네트웍: 두 자원 간의 네트웍 대역폭

3.2 자원의 선택 방법

N개의 그리드 자원에서 사용 중인 m개의 자원이 있을 때, k개의 자원을 선택하는 경우의 수는 다음과 같이 표현할 수 있다.

$$\frac{(N-m)!}{k!\,(N-m-k)!}$$

만약 전체 100개의 자원 중 10개의 자원이 사용하고 있고 5개의 자원을 새로 선택한다고 하였을 때, 자원 선택 방법은 총 14,649,756가지가 된다. 그러므로 이와 같은 문제는 모든 경우의 수를 고려하여 해결하지 않고,

일반적으로 휴리스틱 알고리즘을 이용하여 이 문제를 해결하고 있다. 본 논문에서는 그리드 시스템에서 자원을 선택하는 방법으로 다음과 같은 여섯 가지 알고리즘을 고려하였다. FASTCPU 알고리즘과 FASTER 알고리즘은 자원의 계산 속도를 고려한 알고리즘이고, BESTFIT 알고리즘과 WORSTFIT 알고리즘은 자원간의 통신량을 고려한 알고리즘이다. 또한 FIRSTFIT 알고리즘은 라운드 로빈(round robin)과 비슷한 방법으로 수행되는 알고리즘이며, FASTNET 알고리즘은 주로 대용량의 메시지 전송이나 파일 전송을 고려한 알고리즘이다.

3.2.1 FASTCPU 알고리즘

FASTCPU 알고리즘은 그리드 자원 중 계산 속도만을 고려하여 자원을 선택하는 방법이다. 즉 계산 속도가 가장 빠른 자원부터 할당하는 방법이다.

3.2.2 FASTER 알고리즘

FASTER 알고리즘은 FASTCPU 알고리즘과 비슷하나, 기준 값보다 노드의 개수가 많은 자원 중 계산 속도가 빠른 자원을 선택하는 알고리즘이다.

3.2.3 FIRSTFIT 알고리즘

FIRSTFIT 알고리즘은 라운드 로빈과 같은 방법으로, 그리드 자원에 index를 부여하고 사용자 작업이 자원을 요청할 때마다 순서대로 자원을 할당하는 방법이다.

3.2.4 BESTFIT 알고리즘

BESTFIT 알고리즘은 사용자 작업이 요구하는 노드 수와 가장 근접한 자원을 할당하는 방법이다. 즉, 수식 (1)에서 N_{min}을 갖는 자원을 선택하는 방법이다. 수식 (1)에서 J_k(nodecount)는 k번째 사용자 작업이 요구하는 노드 수를 나타내며, R_i(available)은 i번째 자원의 사용 가능한 노드 수를 나타낸다.

$$N = J_k(\text{nodecount}) - R_i(\text{available})) \geq 0 \quad (1)$$

3.2.5 WORSTFIT 알고리즘

WORSTFIT 알고리즘은 BESTFIT 알고리즘과 비슷하지만, 노드 수가 가장 많은 자원을 우선으로 할당한다는 점이 다르다. 즉 수식 (1)에서 N_{max}인 자원을 선택한다.

3.2.6 FASTNET 알고리즘

FASTNET 알고리즘은 사용자 작업을 가장 계산 속도가 빠른 자원과의 네트워 대역폭이 큰 자원 순으로 할당하는 방법이다.

4. 포인트 기반 스케줄링 알고리즘

4.1 알고리즘

본 논문에서 제안한 포인트 기반 스케줄링 알고리즘은 자원의 계산 능력과 네트워의 대역폭을 모두 고려한 알고리즘이다. 제3장에서 제안되었던 스케줄링 알고리즘은 나름대로의 장점을 가지고 있으나, 어떠한 알고리즘이 좋은 성능을 나타내는지 측정하기 어렵다. 그러므로 스케줄링 알고리즘의 성능을 평가하기 위한 방법으로 자원의 계산량과 네트워의 대역폭을 포인트와 가중치를 사용하여 가장 높은 점수를 취하는 알고리즘을 선택하도록 하였다.

수식 (2)는 그리드 자원을 평가하기 위한 모델링이며 [6], P는 성능(performance), PF는 성능 인자(performance factor), α_i는 i번째 성능 인자에 대한 가중치를 나타낸다.

$$P = \alpha_1 PF_1 + \alpha_2 PF_2 + ... + \alpha_3 PF_3 \quad (2)$$

본 논문에서 제안한 포인트 기반 스케줄링 알고리즘은 계산 속도와 네트워 대역폭을 고려하여 점수를 계산하는 방법이며, 수식으로 나타내면 다음과 같다. 이는 자원의 계산량에서 네트워에 대한 부하량을 감산한 것이다.

$$P = \sum nRP - k\sum \frac{m}{NP} \quad (3)$$

수식 (3)에서 P(point)는 점수, RP(resource point)는 자원 점수, NP(network point)는 두 자원 사이의 네트워 점수를 의미한다. n과 m은 알고리즘에 의해 선택된 자원의 수이며, k는 네트워에 대한 가중치이다. 수식 (3)에서 사용자 작업이 요구하는 노드 수가 같다 하더라도 여러 자원에 분산하여 처리하는 것이 좋지 않은 성능을 보인다는 것을 확인할 수 있다. 예를 들어, 사용자 작업이 요구하는 노드의 개수가 z개라고 할 때, x개의 자원을 선택하는 방법을 A 알고리즘, y개의 자원을 선택하는 방법을 B 알고리즘이라고 하자. 사용자 작업이 요구하는 노드의 개수가 z이므로, 각 알고리즘은 다음과 같은 수식이 성립한다.

$$\begin{aligned} a_1 + a_2 + a_3 + ... + a_x &= b_1 + b_2 + b_3 + ... + b_y \\ &= z \quad \text{for } x < y \end{aligned} \quad (4)$$

수식 (4)에서 각 자원을 연결하는 네트워의 대역폭이 동일하다고 가정하면, A 알고리즘은 수식 (5), B 알고리즘은 수식 (6)와 같이 표현할 수 있다.

A 알고리즘:

$$\sum \frac{m}{NP} = \frac{(x-1)(a_1 + a_2 + a_3 + ... + a_x)}{NP} \quad (5)$$

B 알고리즘:

$$\sum \frac{m}{NP} = \frac{(y-1)(b_1 + b_2 + b_3 + ... + b_y)}{NP} \quad (6)$$

수식 (4), (5), (6)에 의해, 사용자 작업이 여러 자원에 분산되어 처리될수록 P값은 작아지게 된다. RP 값을 계산하는 방법은 수식 (7)과 같이 표현할 수 있다.

$$RP = \frac{\text{CPU speed}}{\text{System Load} + 1} \quad (7)$$

CPU 속도를 시스템의 부하량으로 나누는 이유는 사용자 작업의 실행 시간이 같은 CPU 속도를 갖는 자원에서 실행되더라도 시스템의 부하량에 가장 큰 영향을 받기 때문이다[9]. 만약 RP 값이 같은 자원이 있는 경우 전체 노드의 수와 사용 가능한 노드의 수가 큰 자원을 선택한다. 또한 NP는 두 자원 사이의 네트웍 대역폭으로 정의한다. 만약 두 개의 노드가 하나의 자원에 속해 있다면 NP 값은 0으로 한다. 가중치 k는 작업의 특성을 반영하기 위한 수치로서, k값이 0이라면 계산 위주의 작업에 적합하며, k값을 크게 잡을수록 네트웍 위주의 작업에 적합하다. 포인트 기반 스케줄링 알고리즘은 위와 같은 계산식을 이용하여 3장에서 제시되었던 알고리즘을 평가한다. 그리고 각각의 알고리즘에서 선택된 자원에 대하여 점수를 계산한 후 가장 높은 점수를 갖는 알고리즘을 선택한다.

그림 1은 그리드 시스템에서 각각의 알고리즘의 점수를 어떻게 평가하는지 알아보기 위한 것이다. 여기에서는 사용자 작업이 요구하는 노드의 수가 12이고, k가 1000, 다음 자원을 가르키는 index는 B인 경우라고 가정한다.

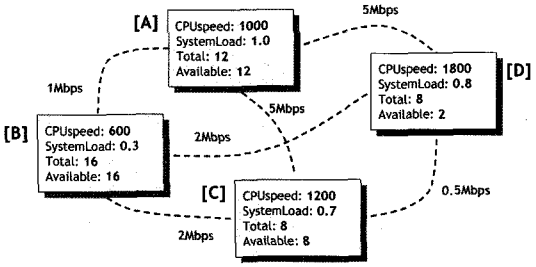

그림 1 4개의 자원으로 구성된 그리드 시스템

그림 1에서 각 자원의 RP 값은 각각 다음과 같다. RP(X)는 자원 X에 대한 RP 값을 나타낸다.

$$RP(A) = \frac{1000}{1+1} = 500$$

$$RP(B) = \frac{600}{0.3+1} = 462$$

$$RP(C) = \frac{1200}{0.7+1} = 706$$

$$RP(D) = \frac{1800}{0.8+1} = 1000$$

FASTCPU 알고리즘은 자원의 계산 속도를 기준으로 한 스케줄링 알고리즘이므로 자원을 RP 값으로 정렬하여 순서대로 선택하면, D(2), C(8), A(2)가 된다. 이렇게 선택된 알고리즘의 점수를 평가하면 다음과 같다.

$\sum nRP = 1000\times2 + 706\times8 + 500\times2 = 24648$

$\sum m/NP = (2+2)/5 + (2+8)/5 + (2+8)/0.5 = 22.8$

$P = 24648 - 1000\times22.8 = 1848$

FASTER 알고리즘은 기준 값보다 높은 자원 중 사용 가능한 노드의 개수를 갖는 자원을 RP값 순으로 정렬하여 선택하는 알고리즘이므로, C(8), A(4)가 된다. 여기에서는 기준 값을 (사용 가능한 평균 노드 수 / 전체 자원의 수)로 정의하였다.

$\sum nRP = 706\times8 + 500\times4 = 7648,$

$\sum m/NP = (8+4)/5 = 2.4$

$P = 7648 - 1000\times2.4 = 5248$

FIRSTFIT 알고리즘은 index를 기준으로 사용자 작업을 자원에 순서대로 할당하는 알고리즘이므로, B(12)가 된다.

$\sum nRP = 462\times12 = 5544,\ \sum m/NP = 0$

$P = 5544 - 1000\times0 = 5544$

BESTFIT 알고리즘은 사용자 작업이 요구하는 노드 수와 가장 근접한 노드를 갖는 자원을 할당하는 알고리즘이므로, A(12)가 된다.

$\sum nRP = 500\times12 = 6000,\ \sum m/NP = 0$

$P = 6000 - 1000\times0 = 6000$

FASTNET 알고리즘은 가장 속도가 빠른 자원과의 네트워 대역폭이 큰 자원을 할당하는 알고리즘이므로, 자원 D에서 2개의 노드를 선택하고, 자원 D와 네트웍 대역폭이 가장 큰 자원 A에 10개의 노드가 할당된다.

$\sum nRP = 1000\times2 + 500\times10 = 7000,$

$\sum m/NP = (2+10)/5 = 2.4$

$P = 7000 - 1000\times2.4 = 4600$

그러므로, 그림 1의 경우에는 포인트 기반 스케줄링 알고리즘은 P_{max} 값을 갖는 BESTFIT 알고리즘을 선택한다.

4.2 시뮬레이션 환경

본 논문에서 사용한 글로벌 그리드 시스템은 온라인 모드의 스케줄링을 수행하며, 그리드 자원은 Consistent 모델[3,5]을 따르고 있다고 가정하였다. 시뮬레이션 환경은 JDK 1.4.1로 개발되었고, RedHat Linux 7.3 운영체제를 사용하였다. 그림 2는 그리드 시스템을 시뮬레이션하기 위하여 사용한 구조이다.

그림 2에서 GenRandom 클래스를 이용하여 시뮬레이션에서 사용하는 파라미터 값을 초기화하고, 이를 JobGenerator 클래스와 ResGenerator 클래스를 이용하여 각각 userjob.xml 파일과 resource.xml 파일을 생성한다. 이렇게 생성된 XML 파일을 이용하여 스케줄러는 XMLParser 클래스를 이용하여 그리드 자원을 생성하며 사용자 작업에 대하여 여러 스케줄링 알고리즘을 적

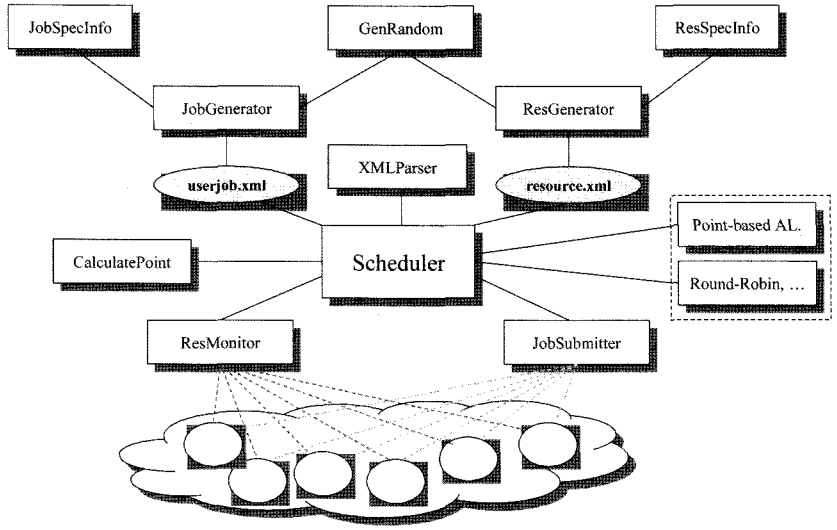

그림 2 그리드 시뮬레이터의 구조

용할 수 있다. 사용자 작업은 JobSubmitter 클래스를 이용하여 자원에 할당할 수 있으며, 자원의 상태는 ResMonitor 클래스를 이용하여 모니터링 할 수 있다. 시뮬레이션에 사용되는 사용자 작업 및 그리드 자원에 대한 확률 분포는 다음과 같다.

표 1 사용자 작업에 대한 확률 분포

엘리먼트 이름	의미	분포
ID	작업의 일련 번호	-
CPUTIME	요구 CPU 시간	N (15000, 1)
NODECOUNT	요구 노드 수	N (16, 1)
BANDWIDTH	요구 네트웍 대역폭	N (1, 1)
IDLETIME	유휴 시간	P (2)
SYNC	동기화 여부	True or False

표 2 그리드 자원에 대한 확률 분포

엘리먼트 이름	의미	분포
NAME	자원의 이름	-
CPUSPEED	CPU 속도	U {600, 800, 1000, 1200, 1500, 1800, 2000}
NODECOUNT	전체 노드 수	U {2, 4, 8, 16, 32, 64}
SYSTEMLOAD	시스템 부하량	초기값: 0.0
BANDWIDTH	다른 자원 간의 네트웍 대역폭	N (50, 1)

4.3 시뮬레이션 결과 및 분석

본 논문에서는 작업의 크기, 작업의 개수, 자원의 개수 등을 변경시켜 가면서 제 3장에서 제안되었던 여러 스케줄링 알고리즘과 포인트 기반 스케줄링 알고리즘을

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE joblist SYSTEM "userjob.dtd">
<joblist>
  <jobspec id="0">
    <cputime>13139</cputime>
    <nodecount>18</nodecount>
    <bandwidth>366</bandwidth>
    <idletime>22</idletime>
    <sync>false</sync>
  </jobspec>
</joblist>
```

그림 3 userjob.xml 파일의 예

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resourcelist SYSTEM "resource.dtd">
<resourcelist>
  <resource name="0">
    <cpuspeed>800</cpuspeed>
    <nodecount>2</nodecount>
    <systemload>0.0</systemload>
    <bandwidth to="0">0</bandwidth>
    <bandwidth to="1">39</bandwidth>
    <bandwidth to="2">69</bandwidth>
    <bandwidth to="3">77</bandwidth>
    <bandwidth to="4">47</bandwidth>
    <bandwidth to="5">78</bandwidth>
  </resource>
</resourcelist>
```

그림 4 resource.xml 파일의 예

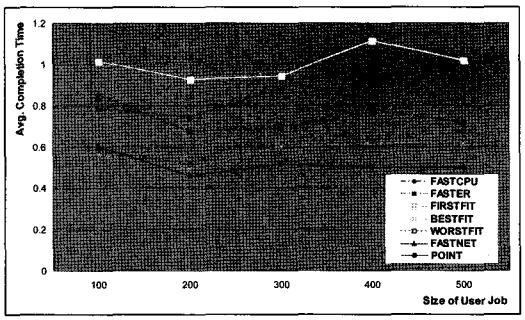

그림 5 작업의 크기를 변경한 시뮬레이션 결과

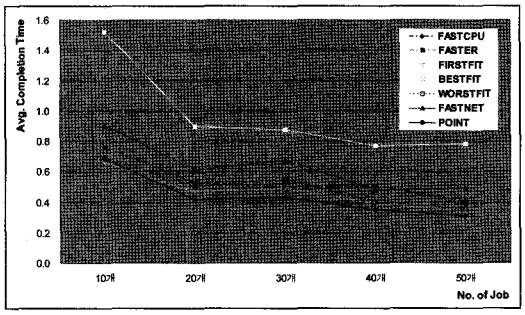

그림 6 작업의 개수를 변경한 시뮬레이션 결과

비교하였다. 평가 기준으로 각 사용자 작업의 완료 시간으로 하였으며, 각각의 시뮬레이션마다 100번을 실험하여 평균값을 취했다. 그림 5는 작업의 크기를 변경한 결과 그래프이며, 임의로 자원을 선택하는 RANDOM 알고리즘과의 상대 값으로 표현하였다. 그림 5에서 볼 수 있듯이, 포인트 기반 스케줄링 알고리즘은 BESTFIT 알고리즘과 비슷한 양상을 띄고 있다는 것을 알 수 있다. 즉 사용자 작업을 할당할 때 대부분의 경우 $-k\sum\frac{m}{NP}$ 값이 0인 BEST FIT 알고리즘을 따르고 있기 때문이다.

그림 6은 자원의 개수와 작업의 크기는 고정하고 작업의 개수를 변경시켰을 때의 결과 그래프이다. 이 그래프에서 볼 수 있듯이, 작업의 개수를 변경하더라도 포인트 기반 스케줄링 알고리즘이 다른 알고리즘보다 더 좋은 성능을 보인다는 것을 알 수 있다.

그림 7은 작업의 개수와 작업의 크기를 고정하고 자원의 개수를 변경시켰을 때의 결과 그래프이다. 다른 그

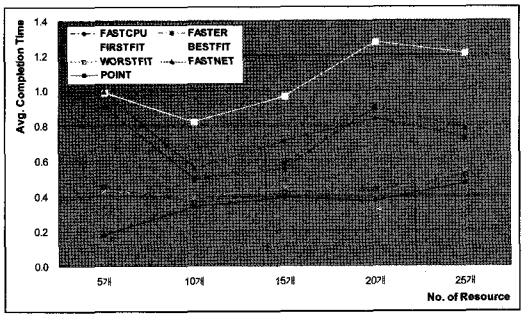

그림 7 자원의 개수를 변경한 시뮬레이션 결과

래프와는 달리 알고리즘의 특성이 두드러지게 나타나는데, 이는 제3장에서 제안되었던 여러 알고리즘이 사용자 작업의 특성보다는 자원의 특성에 더 민감하게 반응하기 때문이라고 볼 수 있다.

지금까지의 그래프를 분석하면, 포인트 기반 스케줄링 알고리즘은 여유 자원이 충분할 때 BESTFIT 알고리즘을 선호한다는 것을 알 수 있다. 그러나 BESTFIT 알고리즘보다 더 좋은 성능을 보이는 이유는 그리드 시스템에 있는 모든 자원이 사용자 작업의 요구 노드 수보다 작은 경우에 다른 알고리즘을 선택하여 자원을 할당하기 때문이다. 즉 사용자 작업의 준비 시간이 짧아지므로 작업의 완료 시간이 줄어들었다고 볼 수 있다. 또한 대부분의 MPI 응용 프로그램이 가장 느린 CPU 속도와 네트워에 맞추어 동기화되므로, 사용자 작업을 여러 노드에 나누어 할당할 때 좋지 않는 성능을 보일 수 있다. 표 3은 사용자 작업을 포인트 기반 스케줄링 알고리즘으로 선택한 자원에 할당하여 구한 준비 시간, 실행 시간, 완료 시간, makespan 값을 각각 1.00으로 보았을 때, 제3장에서 제안한 알고리즘을 비교한 상대 값을 보여주고 있다.

5. 결 론

과학 기술이 발전함에 따라 대량의 정보를 분석하고 처리하기 위해 대두된 그리드 시스템은 이질적인 시스템 위에 구축된 환경이므로, 사용자 작업을 효율적으로 할당하기 위한 스케줄링 알고리즘이 필요하다. 현재까지 여러 알고리즘이 연구되어 왔지만, 대부분 여러 자원 사이의 네트워 대역폭을 고려해야 하는 그리드 시스템에

표 3 포인트 기반 스케줄링 알고리즘의 성능 비교

	FASTCPU	FASTER	FIRSTFIT	BESTFIT	WORSTFIT	FASTNET	RANDOM
ready	6.943	6.880	18.348	22.848	69.605	8.483	229.744
execution	2.114	2.097	2.987	1.401	1.391	2.218	2.723
completion	1.598	1.582	2.381	1.156	1.201	1.723	2.426
makespan	1.375	1.379	1.401	1.139	1.149	1.364	1.555

적합하지 않는 문제점을 가지고 있다. 본 논문에서는 그리드 시스템에서 사용자 작업이 실행되는 자원들을 선택하기 위한 포인트 기반 스케줄링 알고리즘을 제안하였다. 포인트 기반 스케줄링 알고리즘은 글로벌 그리드 시스템에서 사용되는 여러 알고리즘의 점수를 각각 계산하여 가장 점수가 낮은 알고리즘을 선택한다. 그리하여 사용자 작업의 준비 시간과 메시지 전송에 걸리는 시간을 단축할 수 있음을 시뮬레이션을 통하여 보였다. 또한 글로벌 그리드 시스템에서는 자원 간의 네트웍 대역폭과 사용자 작업이 할당된 자원의 수에 영향을 받는다는 것을 보였다.

참 고 문 헌

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Journal of the International Supercomputer Applications*, vol. 15, no. 3, pp. 200-222, 2001.

[2] R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel, "Scheduling Resources in Multi-User Heterogeneous, Computing Environments with SmartNet," *Proc. of the 7th IEEE Heterogeneous Computing Workshop*, pp. 184-199, March, 1998.

[3] T. D. Braun, H. J. Siegel, and N. Beck, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems," *Journal of Parallel and Distributed Computing*, vol. 61, pp. 810-837, 2001.

[4] O. H. Ibarra and C. E. Kim, "Heuristic Algorithm for Scheduling Independent Tasks on Nonidentical Processors," *Journal of the ACM*, vol. 24, no. 2, pp. 280-289, April, 1977.

[5] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems," *Proc. of the 8th Heterogeneous Computing Workshop*, pp. 30-44, April, 1999.

[6] Srisan E and Uthayopas P, "Heuristic Scheduling with Partial Knowledge under Grid Environment," *Proc. of the 2nd International Symposium on Communications and Information Technology*, p. 4, October, 2002.

[7] M. Pinedo, Scheduling: *Theory, Algorithms, and Systems*, Prentice Hall, NJ, 1995.

[8] M. Maheswaran, T. D. Braun, and H. J. Siegel, "Heterogeneous Distributed Computing," Encyclopedia of Electrical and Electronics Engineering, J. G. Wdbster, editor, John Wiley and Sons, vol. 8, pp. 679-690, 1999.

[9] S. Venkataramaiah and J. Subhlok, "Performance Estimation for Scheduling on Shared Networks," *Proc. of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 148-165, June 2003.

[10] B. R. de Supinski and N. T. Karonis, "Accurately Measuring MPI Broadcasts in a Computational Grid," *Proc. of the 8th IEEE International Symposium on High Performance Distributed Computing*, pp. 29-37, August, 1999.

오 영 은

2001년 서울시립대학교 전산통계학과 졸업(학사). 2004년 서울시립대학교 컴퓨터통계학과 졸업(석사). 2004년~현재 로커스테크놀로지스 근무. 관심분야는 병렬처리, 리눅스, 그리드 컴퓨팅

김 진 석

1990년 과학기술대학 전산학과 졸업(학사). 1992년 KAIST 전산학과 졸업(석사). 1997년 KAIST 전산학과 졸업(박사). 1997년~1999년 KAIST 인공지능연구 센터 Postdoc 연구원. 1997년~1998년 미국 M.I.T. Laboratory for Computer Science Postdoc Fellow. 1998년~1999년 전자통신연구원(ETRI) 슈퍼컴퓨터센터 초빙 연구원. 1999년~현재 서울시립대학교 컴퓨터과학부 부교수. 2005년~현재 서울시립대학교 고성능컴퓨팅 센터(HPCRC) 센터장. 관심분야는 그리드 컴퓨팅, 계산 금융, 병렬 알고리즘