Co-allocation 환경의 그리드 시스템에서 통신비용에 따른 스케줄링 알고리즘의 성능 분석

강 오 한[†] · 강 상 성[††] · 김 진 석[†††]

요 약

지역적으로 분산되어 있는 이기종의 시스템들을 하나로 묶어 사용하는 그리드 컴퓨팅이 차세대 병렬·분산 연산을 위한 새로운 패러다임으로 관심을 끌고 있다. 고속 네트워크로 연결된 다수의 컴퓨터 시스템이 사용자에게 통합된 가상의 컴퓨팅 서비스를 제공하는 그리드 시스템은 통신비용에 대한 중요성이 매우 크다. 따라서 그리드 환경에서 스케줄링 알고리즘은 작업의 실행시간을 단축하기 위하여 자원들의 연산능력과 함께 통신에 대한 비용을 고려하여야 한다. 그러나 현재까지 발표된 대부분의 스케줄링 알고리즘들은 작업이 한 클러스터에서 처리되는 것을 가정함으로써 통신비용을 무시하였으며, 작업이 다수의 클러스터에 분산되어 처리되는 경우에도 통신비용에 관한 오버헤드를 고려하지 않았다. 본 논문에서는 그리드 시스템에 적합한 기존 스케줄링 알고리즘들의 성능을 분석하였으며, 작업이 다수의 클러스터에 분산되어 수행되는 co-allocation 환경에서 통신비용을 고려하여 알고리즘들의 성능을 비교하고 분석하였다.

키워드 : 그리드 컴퓨팅, 스케줄링 알고리즘, 통신 비용, 시뮬레이션, 클러스터

Performance Evaluation of Scheduling Algorithms according to Communication Cost in the Grid System of Co-allocation Environment

Oh-Han Kang[†] · Sang-Seong Kang[††] · Jin-suk Kim[†††]

ABSTRACT

Grid computing, a mechanism which uses heterogeneous systems that are geographically distributed, draws attention as a new paradigm for the next-generation operation of parallel and distributed computing. The importance of grid computing concerning communication cost is very huge because grid computing furnishes users with integrated virtual computing service, in which a number of computer systems are connected by a high-speed network. Therefore, to reduce the execution time, the scheduling algorithm in grid environment should take communication cost into consideration as well as computing ability of resources. However, most scheduling algorithms have not only ignored the communication cost by assuming that all tasks were dealt in one cluster, but also did not consider the overhead of communication cost when the tasks were processed in a number of clusters. In this paper, the functions of original scheduling algorithms are analyzed. More importantly, the functions of algorithms are compared and analyzed with consideration of communication cost within the co-allocation environment, in which a task is performed separately in many clusters.

Key Words : Grid computing, Scheduling algorithm, Communication cost, Simulation, Cluster

1. 서 론

그리드(Grid) 환경에서는 상이한 성능을 갖는 다양한 자원들이 지역적으로 분산되어 있으며 대용량의 연산능력이 요구되는 응용프로그램들이 처리된다[1, 2]. 그리드 환경에서 시스템의 성능을 극대화하기 위해서는 사용자의 요구와 사용 가능한 자원의 성능을 파악하여 응용프로그램을 효과적으로 처리하도록 하는 스케줄러가 필요하다. 응용프로그램의 다양성과 자원의 이질성 등의 특성으로 인하여 그리드 환경에서는 응용프로그램에 대한 실행시간과 완료시간의 예측이 매우 어렵다. 이러한 그리드 시스템의 특성으로 인하여 그리드 시스템의 성능을 향상시키기 위해서는 그리드 연산 환경의 특성을 반영한 스케줄링 기법이 사용되어야 한다.

이러한 환경에서 스케줄링 문제는 NP-Complete[3]이며, 현재까지 그리드 환경에서 사용될 수 있는 다양한 형태의

※ 본 연구는 정보통신부의 2006년 IT정책개발지원사업(No. B1220-0601-0018) 지원사업에 의하여 수행되었음.
†종신회원 : 안동대학교 컴퓨터교육과 교수
††정 회 원 : 안동대학교 컴퓨터교육과 강사
†††정 회 원 : 서울시립대학교 컴퓨터과학부 교수
논문접수 : 2006년 9월 26일, 심사완료 : 2007년 2월 8일

스케줄링 알고리즘이 국·내외에서 연구되었다[4-14]. 그리드 환경에서 사용되는 스케줄링 알고리즘을 분류하면 응용프로그램에 대한 비용과 시간의 최적화를 고려한 기법[4], 계층구조를 갖는 전역 스케줄링 기법[5], 휴리스틱에 기반한 스케줄링 기법[6, 7] 등이 있다. 국내의 경우 그리드 시스템에서 스케줄링에 관한 연구는 현재 초기단계 수준이며, 최근에 스케줄링 알고리즘에 관한 연구결과들이 발표되었다[11-14].

그리드 환경에서 처리되는 응용프로그램은 연산량, 사용자 요구, 통신 유형, 입출력 비율 측면에서 서로 다른 다양한 특성들을 가지고 있다. 이러한 상이한 특성을 갖는 응용프로그램은 서로 다른 스케줄링 알고리즘의 적용이 요구된다. 그러나 현재까지 발표된 대부분의 스케줄링 알고리즘들은 작업이 한 클러스터에서 처리되는 환경에 적합한 것이었으며, 작업이 다수의 클러스터에 분산되어 처리되는 경우에도 통신비용에 관한 오버헤드를 고려하지 않았다. 본 논문에서는 기존 스케줄링 알고리즘들의 성능을 분석하였으며, 작업이 다수의 클러스터에 분산되어 수행되는 co-allocation 환경에서 통신비용을 고려하여 알고리즘들의 성능을 비교하고 분석하였다

2. 스케줄링 알고리즘의 종류

2.1 관련 용어

이질성(heterogeneous)을 갖는 시스템은 성능이나 노드의 수 등이 다른 특성을 가진 자원들로 구성된 시스템을 말한다. 그리드 시스템과 클러스터 시스템은 여러 시스템이 네트워크로 연결됨으로서 구성된다. 클러스터 시스템을 구성하는 자원들은 짧은 거리의 고속 네트워크로 연결되는 반면 그리드 시스템은 비교적 먼 거리에 위치한 자원들로 연결되어 있다. 그리드 시스템을 구성하는 자원들은 비교적 먼 거리의 네트워크로 연결된 단일 노드 시스템 혹은 클러스터 시스템으로 구성된다. 그리드 시스템을 구성하는 클러스터는 동질의 자원으로 구성된 시스템과는 달리 작업의 요구사항에 따라 그 성능의 차이를 보이는 자원들로 구성되어 있다.

자원의 준비시간(ready time)은 작업이 자원에서 처리되기 위해 대기하는 시간이다. 즉, 특정자원에서 자원에 주어진 모든 작업의 수행을 마친 후 이용 가능한 시간을 말한다. n개의 작업으로 구성된 작업집합을 $T = \{T_1, T_2, T_3, ..., T_n\}$, m개의 자원으로 이루어진 자원집합을 $R = \{R_1, R_2, R_3, ..., R_m\}$으로 정의한다. 여기서 단일 노드는 자원집합의 원소 당 하나의 노드로 이루어져 있으며, 클러스터는 여러 개의 노드로 구성되어 있다. 임의의 수 i와 j가 있을 때 작업 T_i의 자원 R_j에서의 완료시간(completion time)은 T_i를 R_j에서 실행했을 때 작업이 완료되는 시간이다. 즉, 완료시간은 R_j의 준비시간(b_j)에 T_i의 R_j에서의 실행시간(e_{ij})을 합한 값이며, 이것은 식 (1)로 표현 할 수 있다[15].

$$ct_{ij} = b_j + e_{ij} \quad (1)$$

실행시간은 작업의 정보를 이용하여 계산한 기대 실행시간(expected execution time)과 작업을 실제 실행했을 때 소요되는 실제 실행시간(actual execution time)으로 분류된다. 작업의 실행시간은 자원에 따라 서로 다를 수 있으며, 자원의 이질성은 하나의 작업을 여러 자원에서 실행했을 때 나타나는 차이이다. 작업의 이질성이란 여러 작업을 하나의 자원에서 실행하였을 때 나타나는 차이로 자원의 이질성과 마찬가지로 상대적인 값을 가진다. 본 논문에서 스케줄링 알고리즘의 성능을 평가하기 위해 makespan[16]을 사용하였다. 성능평가를 위한 makespan은 전체 작업중에서 마지막 작업이 스케줄링 되었을 때 자원들의 완료시간 중 가장 큰 값을 나타낸다.

2.2 스케줄링 알고리즘

자원에 작업을 할당하는 방법은 크게 정적 스케줄링 방식과 동적 스케줄링 방식으로 분류된다[17]. 정적 스케줄링 방식은 작업의 선후관계와 자원의 변동이 없는 환경에서 사용되는 방법이며, 동적 스케줄링 방식은 작업들 간의 선후관계가 불분명할 때 사용된다. 동적 스케줄링 방식은 스케줄링 시기에 따라 작업이 도착하는 즉시 스케줄링하는 온라인 방식과 간격을 두고 그 간격에 도착한 작업들 모두 스케줄링하는 배치방식으로 나눌 수 있다. 본 논문에서는 온라인 방식의 알고리즘에 대해서 성능분석과 통신비용에 대한 효율성을 시뮬레이션 하였다.

알고리즘의 종류로는 MET(Minimum Execution Time), MCT(Minimum Completion Time), KPB(K-Percent Best)[16], MECT(Minimum Execution and Completion Time)[13, 14] 등을 사용하였다. 이들 스케줄링 방식은 단일 클러스터를 위한 알고리즘이며, 다중 클러스터를 위한 알고리즘으로 MECTwC[12]와 MECTwC의 다중배정 알고리즘을 MET와 MCT에 적용한 METwC, MCTwC를 사용하였다.

MET는 작업 실행시간의 기대 값이 가장 작은 자원을 찾아서 그 자원에 작업을 할당한다. 즉, 작업을 스케줄링할 때, 자원집합에서 완료시간이 가장 작은 자원을 찾아 작업을 할당한다. MET는 실행시간이 가장 작은 자원을 작업에 할당하기 때문에 자원사용의 불균형을 초래할 수 있다.

MCT는 해당 작업에 대하여 완료시간이 가장 작은 자원에 작업을 할당한다. MCT는 준비시간과 실행시간의 합이 가장 짧은 자원에 작업을 할당하는 것이다. MCT는 간단하면서도 좋은 성능을 보여준다. 이 분야에 관한 연구에서 새로운 알고리즘을 제안할 때 비교하는 대상으로 많이 사용되고 있다.

KPB는 자원 중 실행시간이 작은 순서로 $km/100$개의 자원을 선택한다. 그 중에서 완료시간이 가장 작은 자원에 할당하게 된다. 여기서 k는 스케줄링 전에 미리 주어지는 상수 값이고 m은 자원의 개수이다. 만약, k가 100이라면 KPB는 모든 자원을 대상으로 완료시간이 가장 작은 자원을 찾게 됨으로 MCT와 같아진다. 그리고 k가 $100/m$ 이라면 모든 자원에서 실행시간이 가장 작은 것을 찾기 때문에 MET

와 같게 된다. 따라서 k 값의 설정에 따라 KPB의 성능은 변하게 되며, 일반적으로 k의 값이 20일 때 가장 좋은 성능을 나타낸다.

MECT는 동적 스케줄링의 온라인 방식에 속하며 독립적인 작업들을 그 대상으로 하는 알고리즘이다. 자원의 이질성이 존재한다는 것은 특정 작업의 실행시간이 각 자원의 특성에 따라 다르다는 것을 의미한다. 작업이 요구하는 자원의 특성을 고려하여 작업이 할당된다면 전체적인 준비시간을 단축시킬 수 있다. 이것은 이질적인 자원으로 구성된 시스템에서 작업의 실행시간이 성능을 좌우하는데 큰 역할을 할 수 있다는 것을 의미한다. MECT 알고리즘은 이런 가정에서 출발한다. MECT는 완료시간을 고려한다는 점이 MCT와 유사하며, 실행시간을 고려한다는 점에서는 MCT와 다르다. 먼저 준비시간 중 가장 큰 값을 찾고 그 값보다 작은 완료시간을 갖는 자원들을 찾는다. 그러한 자원이 존재할 경우에는 그 자원 중 실행시간이 가장 작은 자원에 작업을 할당하고, 존재하지 않으면 MCT와 같은 방법으로 전체 자원을 대상으로 완료시간이 가장 작은 자원에 작업을 할당한다. 가장 긴 준비시간(이전 작업의 완료시간이 될 수 있음)을 고려해서 현재 자원들 중에서 가장 작은 완료시간을 찾고 존재한다면 실행시간을 고려한다.

(1) MECTcW 알고리즘

MET, MCT, KPB, MECT 알고리즘은 모두 단일 클러스터에 작업을 분배하기 위한 알고리즘들이다. MECTcW는 다중 클러스터에 작업을 분배하여 완료시간을 단축하기 위한 스케줄링 알고리즘이다. 단일 클러스터에 작업을 분배하는 알고리즘은 단순히 현재 자원의 노드 수에만 관련되어 작업이 수행된다. MECTcW 알고리즘을 사용하기 위해서는 단일 클러스터 환경보다 더 많은 데이터가 필요하다. 각 자원의 조합에 의해 작업이 배정될 수 있기 때문에 각 자원의 조합과 그 조합에 대한 노드수를 알아야 한다. 그리고 작업을 분할하기 때문에 그에 따른 통신비용도 감안을 해서 할당해야 한다. <표 1>에서 <표 3>까지는 단일 클러스터에 사용되는 데이터와 다중 클러스터에서 사용되는 데이터를 비교한 예를 나타낸 것이다.

〈표 1〉 단일 클러스터에서 사용되는 데이터 행렬

	R_1(20)	R_2(60)	R_3(40)
T_1(16)	12	24	24
T_2(32)	20	5	10
T_3(60)	30	25	45

〈표 2〉 다중 클러스터에서 사용되는 확장된 데이터 행렬

	R_1(20)	R_2(60)	R_3(40)	r_1,r_2,80	r_1,r_3,60	r_2,r_3,100
T_1(16)	12	24	24	24	24	24
T_2(32)	20	5	10	20	20	10
T_3(60)	30	25	45	30	45	45

〈표 3〉 통신비용 테이블

	R_1	R_2	R_3
R_1	0	0.11	0.68
R_2	0.11	0	0.05
R_3	0.68	0.05	0

〈표 4〉 통신비용이 적용된 확장된 데이터 행렬

	R_1(20)	R_2(60)	R_3(40)	r_1,r_2,80	r_1,r_3,60	r_2,r_3,100
T_1(16)	12	24	24	27	75	25
T_2(32)	20	5	10	22	63	11
T_3(60)	30	25	45	34	141	47

<표 1>은 단일 클러스터에서 사용하는 일반적인 데이터 행렬이다. <표 1>에서 작업 T_1이 R_1에 할당되어 실행될 때 소요되는 실행시간은 12임을 알 수 있다. <표 2>에서 <r_1, r_2, 80>은 작업 T_1이 자원 R_1과 R_2의 두 자원에 분할되어서 할당될 수 있으며, 사용 가능한 노드의 수는 80이라 것을 의미한다. 노드의 수가 80이 되는 것은 R_1과 R_2에서 사용 가능한 노드 수를 합한 것이다. <표 3>는 통신비용 테이블을 나타낸 것이며, 표에서 0은 통신비용이 부가되지 않는 것으로 자신의 클러스터에서 작업이 처리되는 경우이다. <표 3>에서 클러스터 자원 R_1과 R_2사이의 연산정보 교환시 들어가는 통신비용이 0.11임을 알 수 있다. <표 2>의 확장된 데이터 행렬에 <표 3>의 통신비용을 적용하면 <표 4>를 만들 수 있다. 실행시간이 et이고 통신비용이 c라면 통신비용을 반영하여 실행시간을 구하는 것은 수식 (2)와 같이 나타낼 수 있다[12]. 만약 통신비용(c)이 0.5인 두 클러스터에 동시에 작업을 배정하면 단일 클러스터에 배정한 경우보다 실행시간은 2배로 늘어나게 된다.

Input : T_i , $\{e_{i1}, e_{i2}, e_{i3}, ..., e_{im}\}$, $m = |R|$
Output : k (resource index)

① find $b_{\max} = MAX_{R_j \in R}\ b_j$
② look for cluster combination
③ sort by ready time of each node on the each machine
④ find a set of resources R that have completion time smaller than b_{max}
if ($|R| > 0$)
⑤ find k, such that $e_{ik} = MIN_{R_j \in R}\ e_{ij}$
else
⑥ find k, such that $ct_{ik} = MIN_{R_j \in R}\ ct_{ij}$
end if
⑦ return k

(그림 1) MECTcW 알고리즘의 Pseudo 코드

$$\text{통신비용을 반영한 실행시간} = \frac{et}{1-c} \quad (2)$$

예를 들면, <표 2>의 T_1과 $<r_1, r_2, 80>$에 <표 3>을 반영하여 수식 (2)의 값을 구하는 방법은 아래와 같다. (그림 1)은 MECTcW 알고리즘을 Pseudo 코드로 나타낸 것이다.

$$\frac{(1, <r_1, r_2, 80>)}{1-(1,2)} = \frac{24}{1-0.11} = 26.9$$

3. 알고리즘 시뮬레이션

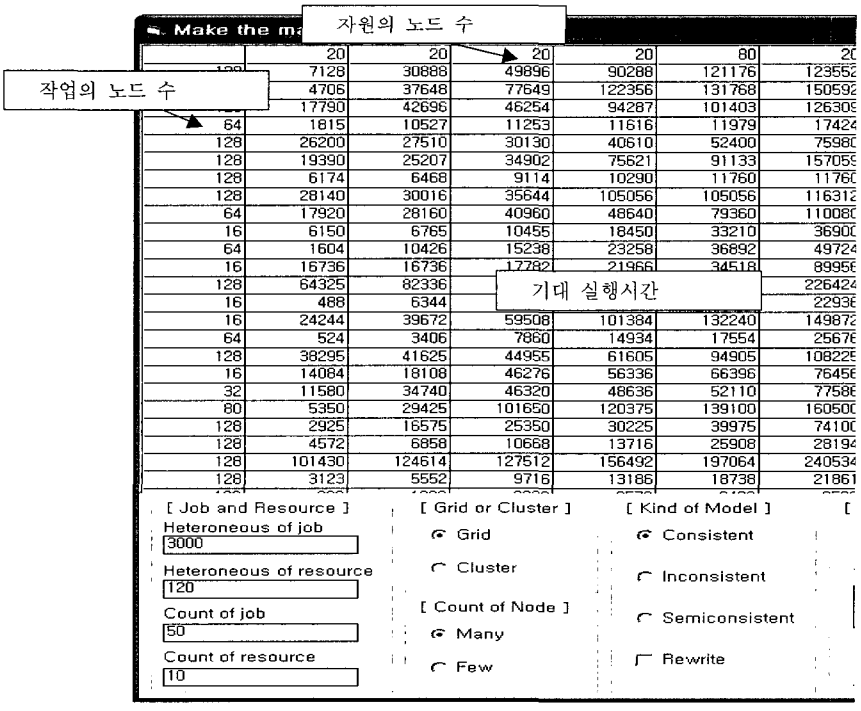
본 논문에서는 단일 클러스터에 작업을 배정하는 알고리즘과 다중 클러스터에 작업을 배정하는 알고리즘의 성능을 비교 분석한다. 다중 클러스터 환경의 co-allocation을 지원하는 알고리즘의 성능을 분석하기 위하여 단일 클러스터 환경에 기초한 알고리즘에 몇 가지의 추가적인 가정이 필요하다. 모든 작업은 실행에 필요한 노드 수를 명시해야 하고, 모든 자원은 허용된 노드의 수를 명시해야 한다. 즉, 몇 개의 노드에서 해당 작업을 수행할 것인지를 명시하여야 한다. 모든 작업은 작업 단위로 클러스터에서 수행되므로 작업이 필요로 하는 노드의 수보다 적은 노드로 구성된 클러스터에는 작업을 할당하지 않는다.

스케줄링 알고리즘의 시뮬레이션을 위해서 기대 실행시간, 준비시간, 통신비용 등의 데이터가 필요하다. 이것들은 작업의 개수, 자원의 개수, 작업의 이질성, 자원의 이질성을 이용하여 만들어진다. 각 작업이 필요로 하는 노드와 자원의 노드 개수는 이질성에 따라 정해진 범위 내에서 랜덤(random)하게 생성한다. 기대 실행시간은 작업 이질성과 자원 이질성의 범위에서 각각 랜덤 값을 추출한 후 이를 곱하여 생성한다. (그림 2)는 작업의 노드 수, 자원의 노드 수, 기대 실행시간을 생성한 예를 나타낸 것이다.

그리드 환경에서의 자원간 통신비용은 각 자원이 네트워크에 연결된 대역폭과 두 자원간의 자료전송 소요시간을 종합하여 경험적으로 얻을 수 있다. 이러한 통신비용은 자원의 상황에 따라 다양하게 나타날 수 있으므로 실험에서는 랜덤 값을 추출하여 사용하였다. 자원의 수가 x이고 작업의 수가 y라면 x와 y를 곱한 만큼의 수가 만들어진다. 이 값들은 전체적으로 랜덤으로 하거나 통신비용이 적게 소요되는 부분과 많이 소요되는 부분으로 분류하여 생성할 수 있다. 또한 통신비용에 따른 분할 알고리즘의 효율성을 확인하기 위해 통신비용을 단위별로 분리하여 만들 수도 있다.

준비시간에 대한 행렬을 만들어야 하며 행렬에서 준비시간의 수는 작업의 수와 같다. 준비시간은 작업이 자원에서 실행되기 전에 대기하는 시간을 나타내는 것이며, 이미 작업이 배정되어 있는 자원이라면 완료시간이 준비시간이 될 수 있고 그렇지 않으면 준비시간 리스트의 값을 생성하여 사용한다. 이 값들은 도착율에 대한 랜덤 값으로 만들며, 도착율이 100 이상이면 가우시안(Gaussian) 분포를 사용하고 100 이하면 포아송(Poisson) 분포를 사용한다.

본 논문에서는 위와 같은 방법[12,13,16]으로 생성한 데이터들을 사용하여 그리드 시스템에 적용 가능한 MET, MCT, MECT, KPB, MECTcW, MCTcW, METcW 알고리즘을 시뮬레이션 하였으며, 자원의 이질성에 따라 각 알고리즘의 성능에 어떤 차이가 있는지 분석하였다.



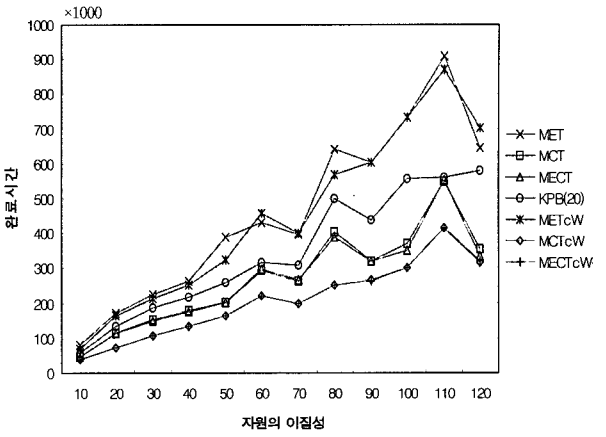
(그림 2) 기대 실행시간, 자원의 노드 수, 작업의 노드 수

3.1 자원의 이질성에 따른 알고리즘 성능분석

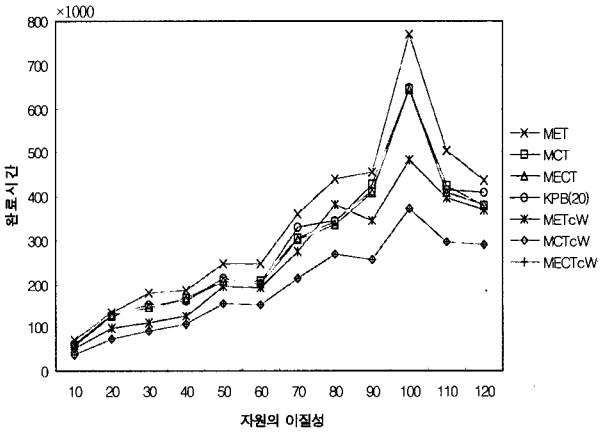
작업의 이질성은 3,000으로 고정하고, 자원의 이질성을 10에서 120의 범위로 10씩 증가시켰다. 각 자원의 이질성마다 10개의 데이터를 준비하여 그리드 시스템에서 어떤 성능을 보이는지 알아보았다. 결과 값은 각 이질성마다 10개의 데이터에 대한 완료시간의 평균을 이용하여 나타내었으며, 일관성(consistent), 반일관성(semi-consistent), 비일관성(inconsistent)의 특성을 가지는 데이터의 결과 값으로 분류하였다. 일관성 모델은 어떤 자원이 다른 자원과 비교해서 특정 작업을 더 빠르게 처리한다면 다른 작업에 대해서도 더 빠르게 처리하는 경우이며, 작업·자원 행렬의 기대 실행시간들이 오름차순으로 정렬되어 있다. 비일관성 모델은 이와 같은 일관성이 보장되지 않은 경우로 행렬의 기대 실행시간들은 무작위로 구성되어 있다. 반일관성은 일부 자원에 대하여만 일관성이 보장되는 경우로 홀수의 열들만 오름차순으로 정렬되어 있는 데이터이다. 현실세계에서는 빠른 입출력 속도를 요구하거나 빠른 연산을 요구하는 경우와 같이 자원의 특성의 종류가 제한적이어서 반일관성이 가장 적합한 모델이라고 볼 수 있다[16]. (그림 3)에서 (그림 5)까지는 각 특성의 데이터들을 적용하여 실험한 결과를 나타낸 것이다.

그림에서 MET는 자원사용의 불균형으로 인하여 성능이 가장 낮음을 알 수 있다. 그리고 자원의 이질성이 높아질수록 그 차이도 증가하는 것을 볼 수 있다. KPB는 일관성 모델에서 알 수 있듯이 MCT와 MECT보다 성능이 조금 낮지만 MET보다 우수한 성능을 가진 것을 알 수 있다. 그러나 KPB도 특정자원에 편중되는 스케줄링을 하기 때문에 알고리즘의 특성상 MCT와 MECT보다 성능이 낮다. 이것은 k의 값을 어떻게 설정하는가에 따라 달라지는데 20으로 했을 때 가장 성능이 좋다는 것을 실험 결과에서 알 수 있다. 100에 근접할수록 MCT와 같아지고 1에 근접할수록 MET와 같아지는데 20이라는 수에서 최적의 성능이 나타난다는 것은 MET와 비슷한 특성을 지니고 있다고 볼 수 있다. MET가 특히 일관성 모델에서 다른 알고리즘들과 성능 차이가 큰 것은 작업 할당시 실행시간만 고려하기 때문이며 작업들을 한 자원에 편중되게 스케줄링 함으로써 나타나는 결과이다. MCT와 MECT는 비슷한 성능을 보여주고 있는데 MECT가 MCT보다 다소 좋은 성능을 보여주고 있다.

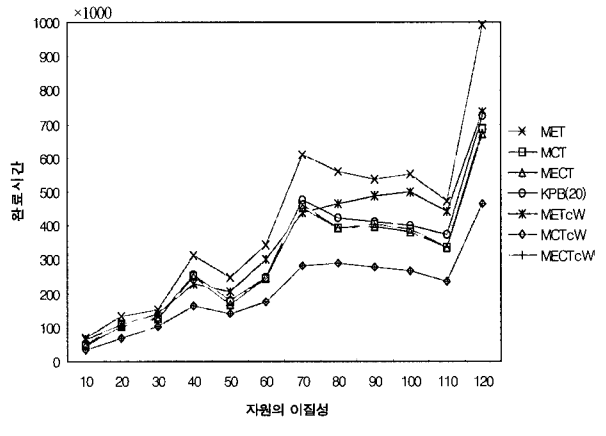
다중 클러스터에 작업을 할당하는 METcW, MCTcW, MECTcW 알고리즘은 단일 클러스터에 작업을 할당하는 알고리즘보다 성능이 우수하다는 것을 알 수 있다. METcW는 MET와 마찬가지로 편중된 스케줄링으로 인하여 낮은 성능을 나타내고 있다. 그러나 단일 노드에 작업을 할당하는 MET보다 다중 클러스터에 작업을 할당하는 METcW가 성능이 우수함을 알 수 있다. MCTcW와 MECTcW는 역시 비슷한 성능을 보여주고 있는데 다중 클러스터에서 MCTcW와 MECTcW는 단일 클러스터에 작업을 할당하는 것보다 약간 우수하다는 것을 알 수 있다.



(그림 3) 일관성 모델에서 알고리즘 성능비교



(그림 4) 비일관성 모델에서 알고리즘 성능비교

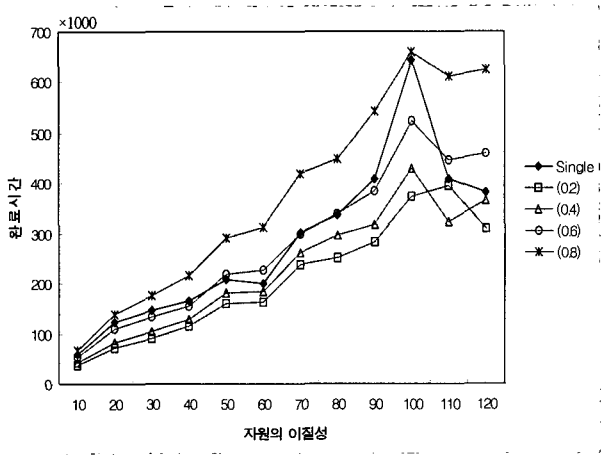


(그림 5) 반일관성 모델에서 알고리즘 성능비교

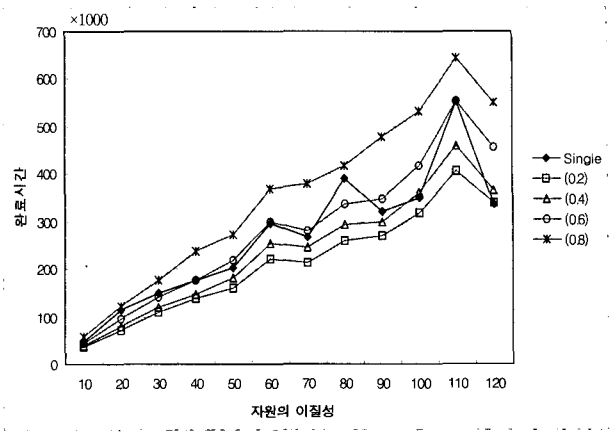
3.2 통신비용에 따른 알고리즘의 효율성 분석

METcW, MCTcW, MECTcW 알고리즘에서 완료시간은 통신비용까지 고려하여 실험한 결과이다. 각 알고리즘에 통신비용이 추가되었지만 그 비용을 감수하고도 단일 클러스터에 작업을 할당하는 것보다 좋은 성능을 나타내었다. 통신비용이 증가함에 따라 다중 클러스터에 작업을 할당하는 것보다 단일 클러스터에 작업을 할당하는 것이 더 좋은 성능을 보이는 한계점이 있을 것이다. 그 한계점을 찾기 위해 시뮬레이션을 수행하여 확인하였다. 통신비용이 0이라면 자신의 클러스터를 가리키며 통신비용이 필요하지 않는 것을 의미한다. 통신비용에서 숫자가 커지면 통신비용이 증가하는 것을 의미하며, 본 논문에서는 0.2, 0.4, 0.6, 0.8로 통신비용을 설정하여 실험하였다. 본 논문에서는 작업을 일관성, 반일관성, 비일관성으로 분류하였으며 단일 클러스터에서 가장 좋은 성능을 보인 MECT를 이용하여 단일 클러스터에 할당하는 경우를 실험하였다. 또한 다중 클러스터에서 가장 좋은 성능을 보인 MECTcW 알고리즘을 사용하여 통신비용이 0.2, 0.4, 0.6, 0.8인 경우에 다중 클러스터 할당을 실험하였다. MECT는 실험 결과는 (그림 6), (그림 7), (그림 8)에 나타나있다.

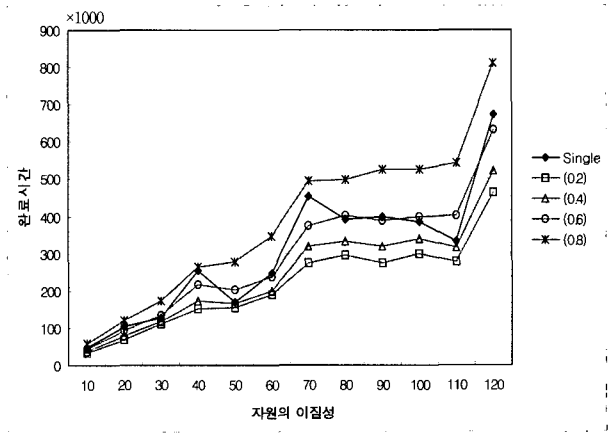
(그림 6), (그림 7), (그림 8)에서 알 수 있듯이 통신비용이 0.5 이상으로 증가하면 co-allocation한 것이 단일 클러스터보다 낮은 성능을 보여주고 있다. 단일 클러스터는 배정될 수 있는 자원의 종류가 한정되어 있기 때문에 완료시간의 폭이 크고, 다중 클러스터는 단일 클러스터보다 배정할 수 있는 자원의 종류가 많아지기 때문에 자원의 이질성이 증가해도 변화의 폭이 적음을 볼 수 있다. 위의 실험결과로 통신비용이 0.5 이상이면 다중 클러스터에 작업을 배정하는 것보다 단일 클러스터에 작업을 배정하는 것이 효율적이라는 것을 알 수 있다. 이것은 기대 실행시간이 약 2배로 올라가면 다중 클러스터에 작업을 배정하는 것이 의미가 없어진다는 것을 나타낸다.



(그림 6) 비일관성 모델에서 통신비용에 따른 성능 비교



(그림 7) 일관성 모델에서 통신비용에 따른 성능 비교



(그림 8) 반일관성 모델에서 통신비용에 따른 성능 비교

4. 결론

본 논문에서는 그리드 시스템에서 기존의 스케줄링 알고리즘을 단일 클러스터와 다중 클러스터에 맞게 수정하여 성능을 비교하였다. 그리드 환경에서 시스템의 특성과 성능이 모두 다르다는 점을 고려하였고, 다중 클러스터에서 작업을 할당할 때는 통신비용을 반영하여 실험하였다. 실험에서는 일관성, 반일관성, 비일관성 모델의 작업과 자원 행렬을 사용하여 자원의 이질성에 따른 각 알고리즘의 성능 분석과 다중 클러스터에서 통신비용에 따른 작업 할당의 효율성을 확인하였다.

MET는 가장 낮은 성능을 보였고, 다중 클러스터에서 사용된 METcW 역시 단일 클러스터보다 낮은 성능을 보여주었다. 이것은 자원사용의 불균형에 기인한 것이다. KPB는 MET보다 우수하고, MCT보다 낮은 성능을 보이는데 그 이유도 MET와 마찬가지로 자원을 편중하여 사용하는 경향이 있기 때문이다. MCT와 MECT는 비슷한 성능을 보였으며,

MECT가 다소 좋은 성능을 보여주었다.

다중 클러스터에 사용된 METcW, MCTcW, MECTcW는 METcW를 제외하고 단일 클러스터에서 사용된 알고리즘보다 매우 우수한 성능을 나타내었다. METcW도 MET보다 좋은 성능을 나타내었는데 이것은 단일 클러스터보다 다중 클러스터로 작업을 할당하여 처리하는 것이 처리시간을 단축할 수 있다는 것을 입증한다. MCTcW와 MECTcW는 비슷한 성능을 보이는데 약간의 차이로 MECTcW의 성능이 좋다는 것을 확인하였다.

다중 클러스터 환경의 co-allocation에서 통신비용을 0.2, 0.4, 0.6, 0.8로 설정하고 MECTcW에 적용시켜 단일 클러스터에 사용되는 MECT보다 성능이 낮게 나오는 한계점을 찾는 실험을 하였다. 통신비용이 약 0.5(통신비용이 2배) 이상이면 작업을 다중 클러스터에 할당하는 것보다 단일 클러스터에서 작업하는 것이 좋은 것으로 나타났다. 이것은 통신비용이 0.5 이상으로 증가하면 다중 클러스터에 작업을 분배하는 것이 단일 클러스터에서 처리하는 것보다 효율 측면에서 장점이 없다는 것을 의미한다.

참 고 문 헌

[1] 특집 그리드 컴퓨팅, 한국정보과학회, 정보과학회지, 2002. 2

[2] Foster and C. Kesselman, "The Grid: Blueprint for a Future Computing Infrastructure," Morgan Kaufmann Publishers, USA, 1999.

[3] 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, 1977.

[4] Rajkumar Buyya, "Economic-based Distributed Resource Management and Scheduling for Grid Computing," Ph. D. Thesis, Monash University, Melbourne, Australia, 2002.

[5] I. Foster, C. Kesselman, "Globus: A metacomputing infrastructure toolkit," International Journal of Supercomputer Applications, Vol.11, No.2, pp. 115-128, 1997.

[6] Srisan E. et al, "Heuristic Scheduling with Partial Knowledge under Grid Environment," Proc. of the 2nd International Symposium on Communications and Information Technology, 2002.

[7] Michael Walker. A Framework for Scheduling Data-Parallel Applications in Grid Systems. MS Thesis, University of Virginia, 2001.

[8] Bruno Volckaert, et. al., "Evaluation of Grid Scheduling Strategies through a Network-aware Grid Simulator," Proc. of PDPTA 2003, June 2003.

[9] Volker Hamscher, et. al., "Evaluation of Job Scheduling Strategies for Grid Computing," Proc. of the 7th International Conference on High Performance Computing (HiPC 2000), pp. 191-203, 2000.

[10] Carsten Ernemann, et.al., "On Advantage of Grid Computing for Parallel Job Scheduling," Proc. of the 2nd IEEE International Symposium on Cluster Computing and the Grid(CCGRID 2002), pp. 39-47, 2002.

[11] 박미선, 박기진, "그리드(Grid)기반 웹 서비스(Web Service)의 응답 시간 향상을 위한 스케줄링 기법," 정보과학회 2003년 추계학술대회, Vol. 30 No. 2-1, pp. 424~426, 2003.

[12] Jeong Woo Jo, Jin Suk Kim, "A Scheduling Algorithm with Co-allocation Scheme for Grid Computing Systems.", GCC 2004, pp. 983-986, 2004

[13] Hak Du Kim, Jin Suk Kim, "An Online Scheduling Algorithm for Grid Computing Systems.", GCC 2003, pp. 34-39, 2003

[14] 김학두, 김진석, 박형우 "GRID 시스템을 위한 온라인 스케줄링 알고리즘," 정보과학회논문지: 시스템 및 이론, 제31권 제2호, 2004.

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

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

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

강 오 한

e-mail: ohkang@andong.ac.kr

1982년 경북대학교 전자계열 전산모듈(학사)

1984년 한국과학기술원 전산학과(공학석사)

1992년 한국과학기술원 전산학과(공학박사)

1984년~1994년 (주)큐닉스컴퓨터 선임/책임연구원

1994년~현재 안동대학교 컴퓨터교육과 교수

관심분야: 그리드 컴퓨팅, 태스크 스케줄링, OVPN 등

강 상 성

e-mail: edukang@andong.ac.kr
1998년 안동대학교 컴퓨터교육과(학사)
2001년 안동대학교 교육대학원
교육공학전공(이학석사)
2001년~현재 안동대학교 컴퓨터교육과 강사
관심분야: 그리드 컴퓨팅, 스케줄링 등

김 진 석

e-mail: jskim@venus.uos.ac.kr
1990년 과학기술대학 전산학과(학사)
1992년 KAIST 전산학과(공학석사)
1997년 KAIST 전산학과(공학박사)
1997년~1999년 KAIST 인공지능연구센터
Postdoc 연구원
1997년~1998년 미국 MIT Lab. for Computer Science Postdoc Fellow
1998년~1999년 ETRI 슈퍼컴퓨터센터 초빙 연구원
1999년~현재 서울시립대학교 컴퓨터과학부 교수
관심분야: 그리드 컴퓨팅, 병렬 알고리즘 등