그리드 컴퓨팅 성능 향상을 위한 자원의 순위 및 그룹화 메커니즘

이진성* 박기진** 최창열* 김성수*
아주대학교 정보통신전문대학원* 안양대학교 문리과학대학 컴퓨터학과**
{cslee77,clchoi,sskim}@ajou.ac.kr* kiejin@aycc.anyang.ac.kr**

Resource Ranking and Grouping Mechanism for System Performance Improvement in Grid Computing

Jinsung Yi* Kiejin Park** Changyeol Choi* Sungsoo Kim*
Graduate School of Information and Communication, Ajou University*
Department of Computer Engineering, Anyang University**

요 약

고속 네트워크의 등장으로 관리 영역을 초월한 계산 자원의 공유가 가능하게 되었고, 그리드 컴퓨팅이 등장하게 되었다. 그리드 환경에 포함된 각 자원들은 이질적이기 때문에 고성능을 얻기 위해서는 효과적인 자원 발견 및 자원 선택이 중요하다. 본 논문에서는 그리드 컴퓨팅에서 다양하고 이질적인 자원들을 각 응용프로그램에서 효율적으로 이용할 수 있도록, 각 참여 노드들에게 순위를 부여하고 각 작업(Task)에 적절한 자원을 할당하여 전반적인 시스템 성능을 극대화시키는 메커니즘을 제안하였다. 각 노드의 순위는 처음 각 노드별 자원의 시스템 사양을 토대로 그룹화하여 작업을 배분하고 각 노드들이 작업을 마친 후 반환 값의 정확도와 응답시간과 같은 동적 요소를 고려하여 참여자의 순위를 부여하였고, 이러한 순위에 따른 작업 재분배 메커니즘은 전체적인 시스템 성능을 향상시키고 정확도를 높인다.

1. 서론

최근 들어 대용량의 계산 및 고성능을 필요로 하는 응용 프로그램들이 개발되고 있으며, 이와 같은 프로그램들을 실행하기 위해서는 지리적으로 분산되어 있는 여러 대의 슈퍼 컴퓨터를 동시에 사용할 필요성이 있다. 고속 네트워크를 통해 지역적으로 분산되어 있는 자원들을 하나로 연결하여 전체적인 시스템 성능을 향상시키는 기법을 그리드(Grid) 컴퓨팅이라 한다. 다양한 그리드 응용 프로그램을 구현하기 위해서는 이질적인 자원들을 투명하게 활용할 수 있게 하는 미들웨어가 필요하며, 글로버스(Globus)와 같은 미들웨어가 개발됨으로써 그리드 기반의 여러 응용프로그램의 개발이 가능하게 되었다[1]. 이질적인 그리드 환경에서는 동종의 구성요소들로 이루어졌던 기존의 시스템과 달리 자원 발견 및 자원 선택 문제가 중요하기 때문에, 응용 프로그램의 특성에 맞지 않는 자원들을 선택하였을 경우, 성능이 떨어질 수 있으며 효율적이지 못할 수 있다. 따라서 자원 선택기능은 미들웨어 상에서 중요한 기능이며 현재 글로버스에서는 set-extended ClasAd 와 set-matching 알고리즘을 제공하여 병렬 프로그램을 위한 자원선택 기능을 제공하나 표현력의 부족과 요구되는 자원과 존재하는 자원이 매칭되지 않았을 때 선택에 있어서 오버헤드가 발생할 수 있다[2]. 따라서, 본 논문에서는 그리드 컴퓨터상의 다양한 참여자(Grid Volunteer)의 시스템 성능과 각 참여자들의 참여의 동적 정보를 토대로 신인도(Credibility)를 고려하여 이력정보를 유지한다. 처음에 마스터 노드(Master)는 현재 접속되어 있는 참여자들의 정적 시스템 정보를 바탕으로 그룹화 한 후 작업을 부여하며, 작업의 처리 결과에 따라 순위를 부여한다. 마스터는 작업의 성격에 따라 각 참여자의 순위 혹은 작업의 성격을 고려하여 작업을 분배하며, 참여자의 작업 수행여부와 반환된 결과나 작업의 양과 같은 동적 요소를 고려하여 계속적으로 참여자들에게 순위를 부여한 후 다시 작업 분배정보로 사용한다. 결국, 각 응용프로그램 성격에 따라 자원을 적절히 분배시켜 전반적인 그리드 기반의 시스템 성능을 향상 시킬 수 있다.

본 논문의 2장에서는 관련 연구를 언급하고, 3장에서 그리드 시스템 참여자 순위 기법을 제안하며 4장에서는 제안한 방법의 구조적 성능 평가를 수행하고 5장에서 결론을 내린다.

본 연구는 한국 과학재단 목적 기초 연구(R05-2003-000-10345-0) 지원으로 수행되었음.
본 연구는 2003 년도 두뇌한국 21 사업에 의하여 지원되었음.

2. 관련 연구

그리드 컴퓨터 환경에서 마스터(Master)와 참여자(Worker) 모델을 따르는 응용 프로그램은 노드들을 확보하고 메시지를 전달할 수 있는 객체지향 프레임 워크를 제공한다[3][4][5]. [6]에서는 참여자들로부터 정확한 값을 얻고 결과의 신인도 향상을 위하여 투표(Voting)와 임의조사 (Spot-Checking)를 이용하는 방법으로 다양한 참여자로부터 같은 태스크를 주어 값을 비교하거나 무작위로 선출된 참여자의 값들을 비교하여 악의적인 참여자를 선별하였으며, 현재 글로버스에서 set-extended ClasAd 와 set-matching 알고리즘을 제공하여 병렬 프로그램을 위한 자원선택 기능을 제공하고 있다.

그리드 컴퓨팅의 성능 향상을 위한 자원의 순위 및 그룹화를 위하여 다양한 연구가 수행 중이며, 본 논문에서는 마스터-참여자(Master-Worker)모델 참여자(Grid-Client Node)의 신인도(Credibility)를 측정하기 위하여 투표(Voting)와 임의조사(Spot-Checking)를 이용하여 결과의 정확성을 측정하였다.

3. 그리드 컴퓨팅 구조

본 연구에서는 기존의 마스터와 참여자로 구성된 그리드 컴퓨팅 구조를 개선한 계층적 마스터-참여자(Hierarchical Master-Worker)구조를 사용하며, 임의의 참여자는 또 다른 참여자의 마스터가 될 수 있다. 실제 시스템 환경에서 참여자들은 이질적인 단일 참여자 외에 단일 클러스터 시스템이 되는 경우가 많기 때문에 모든 참여자들을 모르더라도 효율적으로 관리할 수 있으며 또 다른 참여자들의 마스터 역할을 통하여 확장성을 높일 수 있다. 그림 1 은 마스터와 참여자의 계층적 구조를 나타내고 있으며, 자신의 일차적 하위 참여자만 알고 작업을 전달하는 모습을 나타낸다.

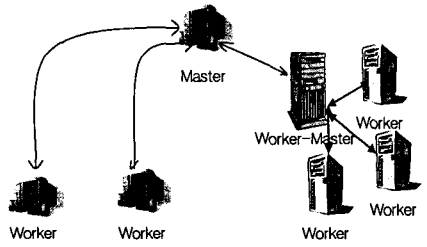

그림 1. 확장성을 고려한 계층적 Master-Worker 구조

3.1 참여자 순위 기법

그리드 컴퓨팅에 참여하고자 하는 인터넷상의 참여자는 마스터(Master)에게 접속하여 작업을 부여 받기 위한 등록 과정을 거치는 것으로 시작한다.

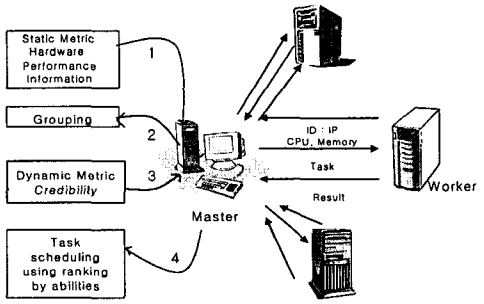

그림 2. 그리드 환경에서 시스템 서열(Ranking)을 위한 시스템 구조도

참여자는 마스터(Master)에게 자신의 자원 정보를 등록해야 하며, 마스터는 등록된 참여자 정보를 이용하여 참여자를 그룹화한다. 마스터가 작업을 부여할 때 같은 그룹의 참여자에게 같은 일을 부여하므로써, 비슷한 성능을 가진 참여자들이 결과를 유사한 시간 내에 줄 확률이 높아진다. 이는 같은 작업에 관해서 결과를 비교하기 위한 효율적인 방법이 될 수 있다. 작업을 할당 받은 참여자들은 결과, 계산시간 및 에러정보를 전달하고 이 정보는 참여자의 신인도를 위한 자료로 이용된다. 그림 2 는 앞에서 설명한 등록부터 순위를 부여하는 과정을 보여준다.

3.2 참여자의 정적 정보(Worker's Static Information)

각 참여자들은 시스템을 등록할 경우 마스터에게 자신의 시스템 정보를 제공한다. 이는 다양한 참여자에 의한 이질적인 시스템을 최대한 이용하기 위한 목적으로 처음에 각 참여자에게 작업(Task)을 부여할 때 정책적인 값으로 사용할 수 있다. 표 1 은 시스템의 능력에 따라 등급을 부여하기 위한 정적 요소들로 이 요소에 따라 그룹화를 진행한다. 시스템 사양이 업그레이드 될 경우 정보는 갱신될 수 있다.

표 1. 참여자 그룹화를 위한 정적 정보

ID	각기 다양한 참여자들을 구별하기 위한 ID 로 IP 를 이용
운영체제	시스템의 운영체제 정보
C.P.U.	명령어 처리 장치로 계산 속도를 결정 정보
주 메모리	데이터의 접근 속도를 단축시키는 메모리 공간정보
저장공간	데이터 저장 공간 확보를 위한 장소
Network Bandwidth	각 자원(데이터)의 이동을 위한 통로

이러한 그룹화는 여러 참여자들에게 같은 태스크를 제공하고 정확도를 측정하는 방법(Voting & Spot-checking)을 사용할 때 다양한 참여자로부터 계산 결과를 동시에 얻기 위한 방법이다. 투표(Voting)와 임의 추출(Spot-checking)과 같은 방법에서 각 참여자의 결과로부터 결과 값을 비교하기 위해서는 가장 늦게 값을 반환하는 시스템에 의해 전체 성능이 좌우되기 때문에, 정적 정보에 의하여 유사한 성능을 가진 참여자들에게 같은 작업을 분배하는 것이 유사한 시간에 결과를 받을 수 있는 효율적 방안이라 할 수 있다. 또한 이러한 정보는 응용프로그램의 성격에 따라 가중치를 두어 시스템에 순위를 부여할 수 있다. 가령, 그리드 기반의 계산컴퓨팅에서 작업을 분배하는 기준은 CPU 의 속도에 더 많은 가중치를 부여하며, 데이터 그리드(Data Grid)에서는 저장장소의 크기에 더 많은 가중치를 부여할 수 있다.

3.3 참여자의 신인도(Worker's Credibility)

참여자의 정적 요소와 아래에 설명되는 동적 요소를 고려하여 각 참여자의 순위 및 그룹을 나눌 수 있으며, 그림 3 의 알고리즘을 이용하여 참여자의 신인도를 지속적이며 실시간으로 관리를 할 수 있다. 참여자의 동적인 정보는 관련연구에서 소개한 투표(Voting)나 임의 추출(Spot-checking)을 이용하여 결과의 정확성을 측정하며, 정확성 외에 결과의 반환시간과 작업도중 발생하는 에러나 혹은 각 참여자의 시스템 내적 요소 혹은 외적 요소로 발생된 실패의 이력정보를 순위를 위한 동적 요소에 포함시킬 수 있다. 따라서 동적 요소는 크게 정확도(Correctness), 응답시간(Turn Around Time), 에러율(Error Rate)의 세 요소를 이용한다.

정확도는 참여자들이 작업을 마치고 난 후 반환된 값들을 비교하여 평가하며, 응답시간은 참여자들이 작업을 받아 처리한 후 값을 반환하기까지 소요되는 시간으로 실제 네트워크 상황 정보도 포함된다. 에러율은 참여자들이 작업도중 에러를 발생하게 될 확률로 기존의 에러정보를 이용하여 결정한다. 아래 알고리즘은 각 참여자의 동적 정보(에러율, 응답시간, 정확도)를 바탕으로 신인도를 계산하는 과정을 나타낸다.

```
For( i=0; i < VolunteerNumber; i++ )
 {
   If ( Worker[i].ErrorFlag == TRUE ) // 에러 발생시
     {Worker[i].ErrorTotalCount =     // 참여자의 에러발생 총 횟수
      Worker[i].ErrorTotalCount + ErrorCount(에러 발생 횟수) ;}
   Worker[i].TurnAroundTime =         //참여자의 계산시간 정보
      Worker[i].ValueReturnTime - Worker[i].SavePassTime;
   If (TruthValue == Worker[i].ReturnValue) //결과 비교 후 참인 경우
     {Worker[i][j].Correctness =      //참여자의 정확도
      Worker[i].Correctness + CorrectCount(정확한 값을 반환한
횟수);}
  }
  .....
 Worker[i].Credibility = //동적 정보 이용하여 신인도 계산
 Worker[i].Credibility - ( Worker[i].ErrorTotalRate +
```

```
Worker[i].CorrectnessRate -
Worker[i].TurnAroundTimeRate )
```

그림 3. 동적 정보를 이용한 신인도 계산 알고리즘

3.4 참여자 자원 정보 테이블

각 참여자의 순위는 아래의 공식을 바탕으로 계산된다.

$$\text{정확도 } (c) = \frac{\text{총 작업중 정확한 값을 리턴한 수}}{\text{부여된 총 작업의 수}},\ 0 \le c \le 1$$

$$\text{응답시간 } (t) = \frac{\text{응답시간}}{\text{같은 작업을 받은 참여자로부터 응답시간의 총합}},\ 0 \le t \le 1$$

$$\text{에러율 } (e) = \frac{\text{에러 발생 횟수}}{\text{같은 작업을 받은 참여자로부터 에러의 총합}},\ 0 \le e \le 1$$

위 공식을 기초로 참여자의 정확도를 더한 후 응답시간과 에러율을 감하여 신인도를 계산하여 순위를 부여한 예를 들면 표 2 와 같으며 시스템 사양이 같더라도 네트워크의 부하에 따른 응답시간의 지연으로 차이가 날 수 있다.

표 2. 정적 요소와 동적 요소를 고려한 참여자 순위 계산 예

요소 / 구분	정적 요소(Static Factors)				동적 요소(Dynamic Factors)				총합(Total)
ID(IP)	OS	CPU	RAM	Network Bandwidth	Correctness	Time (Sec)	Error Rate	Credibility	Ranking Grouping
1	WinNT	937	512	10 Mbps	0.98	7	0.09	0.478	1
2	WinNT	937	512	10 Mbps	0.95	10	0.1	0.261	2

4. 성능 평가

시스템 성능은 참여자 수와 신인도를 고려하여 연산시간을 산출하는 방식과 참여자 수와 에러율에 따라 정확도 측면에서 살펴보았다. 그림 4 는 신인도와 실제 참여자수가 비례할 때, 총 연산시간을 보여주는 그래프로 신인도가 높아질수록 오버헤드가 줄어들고 참여자가 많아져 실제 연산시간에 있어서 많은 성능 향상을 보인다.

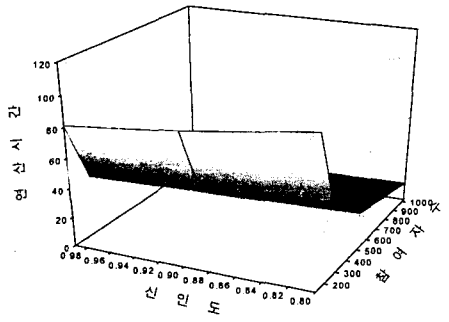

그림 4. 신인도와 참여자수에 따른 연산 소요 시간 그래프

그림 5 는 참여자와 정확도에 따라 에러율을 보여주는 그래프로 투표(Voting)방법을 사용하여 같은 작업을 한 유사 그룹내의 참여자들 중 과반수 이상의 참여자가 같은 값을 반환할 경우 그 값은 정확하다고 했을 때, 정확도를 측정한 것이다. 참여자가 많고 에러율이 낮을수록 정확도는 증가하게 된다.

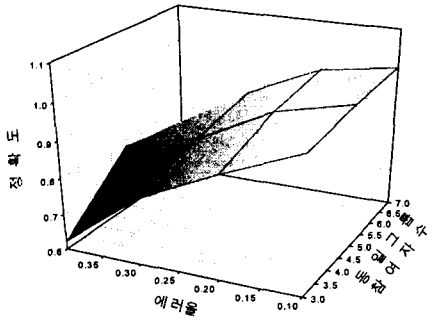

그림 5. 에러율과 참여자수를 고려한 정확도 측정 그래프

5. 결론 및 향후 방향

본 논문에서는 이질적인 자원을 가진 그리드 환경에서 각 응용 프로그램의 분할된 태스크에 적합한 자원을 맵핑하기 위한 메커니즘을 제공하였으며, 정적 정보와 동적 정보를 이용하여 각 참여자의 순위를 부여하고 부여된 순위에 따라 태스크를 제공할 때 전반적인 시스템 성능 향상을 가져왔다.
향후 과제로 참여자 수와 시스템 오버헤드의 관계를 고려하여 시스템 성능을 향상시키는 방안을 연구할 예정이다.

6. 참고문헌

[1] I. Foster, and K. Kesselman, "Globus: A metacomputing infrastructure toolkit," The International Journal of Supercomputer Applications and High Performance Computing, Vol. 11, No. 2, pp. 115-128, 1997.

[2] Chuang Liu, Lingyun Yang, Ian Foster, and Dave Angulo, "Design and Evaluation of a Resource Selection Framework for Grid Applications," Proceedings of the 11th IEEE Symposium on High-Performance Distributed Computing, pp. 63-72, July 2002.

[3] Jean-Pierre Goux, Sanjeev Kulkarni, Michael Yoder, and Jeff Linderoth, "An Enabling Framework for Master-Worker Applications on the Computational Grid," 9th IEEE International Symposium on High Performance Distributed Computing (HPDC'00), pp. 214-217, Aug. 2000.

[4] Elisa Heymann, Miquel A. Senar, Emilio Luque, and Miron Livny "Evaluation of an Adaptive Scheduling Strategy for Master-Worker Applications on Clusters of Workstations," Proceedings of 7th International Conference on High Performance Computing, pp. 310-319, 2000.

[5] Luis F. G. Sarmenta. "Sabotage-Tolerance Mechanisms for Volunteer Computing Systems," ACM/IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01), pp. 337-346, May 2001.

[6] Luis F. G. Sarmenta. "Studying Sabotage-Tolerance Mechanisms through Web-based Parallel Parametric Analysis and Monte Carlo Simulation," Proceedings of the International Conference on Internet Computing, pp. 557-563, June 2001.