그리드 컴퓨팅을 위한 동적 자원 랭킹 및 그룹핑 알고리즘

(Dynamic Resource Ranking and Grouping Algorithm for Grid Computing)

이 진 성 † 박 기 진 †† 최 창 열 ††† 김 성 수 ††††
(Jinsung Yi) (Kiejin Park) (Changyeol Choi) (Sungsoo Kim)

요 약 고속 네트워크의 등장으로 관리 영역을 초월한 계산 자원의 공유가 가능한 그리드 컴퓨팅 개념이 등장하게 되었다. 그리드 컴퓨팅 환경에 포함된 각 자원들은 이질적이기 때문에 고성능을 얻기 위해서는 효과적인 자원 발견 및 자원 선택이 중요하다. 본 논문에서는 그리드 컴퓨팅에서 다양하고 이질적인 자원들을 각 응용 프로그램에서 효율적으로 이용할 수 있도록, 각 참여자들에게 순위를 부여하고 각 작업(Task)에 적절한 자원을 할당하여 전반적인 시스템 성능을 극대화시키는 메커니즘을 제안하였다. 각 참여자의 순위는 초기에 각 참여자별 자원의 시스템 사양(예:CPU 속도, RAM 크기)등 정적 요소를 고려하여 결정되며, 추후에는 각 참여자들이 작업을 마친 후 제공되는 반환 값의 정확도, 응답시간과 같은 요소를 고려하여 동적으로 수행될 수 있도록 하였다. 이러한 순위에 따른 작업 재분배 메커니즘은 전체적인 시스템 성능을 향상시키고 정확도를 높인다.

키워드 : 그리드 컴퓨팅, 신인도, 순위, 동적 정보

Abstract The high-speed network permits Grid computing to handle large problem of management areas and share various computational resources. As there are many resources and changes of them in Grid computing, the resources should be detected effectively and matched correctly with tasks to provide high performance. In this paper, we propose a mechanism that maximizes the performance of Grid computing systems. According to a priority, grade and size of heterogeneous resources, we assign tasks to those resources. Initially, a volunteer's priority and ranking are determined by static information like as CPU speed, RAM size, storage size and network bandwidth. And then, the rank of resources is decided by considering dynamic information such as correctness, response time, and error rate. We find that overall Grid system performance is improved and high correctness using resource reallocation mechanism is achieved.

Key words : Grid Computing, Credibility, Ranking, Dynamic Information

1. 서 론

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 분산된 컴퓨터를 연결해 주는 미들웨어가 개발됨으로써 그리드 컴퓨팅 기반 여러 응용 프로그램의 개발이 가능하게 되었다. 그리드 환경에서 초대형의 거대 문제들을 해결하기 위해서는 그리드로 연결된 유휴 자원들을 찾아내는 자원 검색 서비스, 할당된 작업들의 처리 순서를 결정하여 분산시키는 스케쥴링 서비스, 시스템 안정을 위한 그리드 보안 서비스, 컴퓨팅 자원들을 사용할 때 발생하는 비용 처리를 위한 사용자 계정 서비스 등이 필요하다. 이와 같이 그리드에 연결된 자원들을 필요로 하는 모든 사용자들에게 보편적인 서비스를 제공하여야 한다. 그리드 미들

· 본 연구는 한국과학재단 목적기초연구(R05-2003-000-10345-1) 지원으로 수행되었음
† 비 회 원 : LG전자 MC사업본부 연구원
jinslee@lge.com
†† 종신회원 : 아주대학교 산업정보시스템공학부 교수
kiejin@ajou.ac.kr
††† 비 회 원 : 아주대학교 정보통신전문대학원
clchoi@ajou.ac.kr
†††† 종신회원 : 아주대학교 정보통신대학 교수
sskim@ajou.ac.kr
논문접수 : 2003년 12월 30일
심사완료 : 2005년 3월 16일

웨어의 대표적인 소프트웨어가 글로버스 툴킷이다[1-3].

이질적인 그리드 컴퓨팅 환경에서는 동종의 구성요소들로 이루어졌던 기존의 시스템과 달리 자원 발견 및 자원 선택 문제가 중요하기 때문에, 응용 프로그램의 특성에 맞지 않는 자원들을 선택한다면, 성능이 떨어질 수 있으므로 효율적이지 못하다. 따라서 자원 선택기능은 미들웨어 상에서 중요한 기능이며 현재 Condor에서는 Set-extended ClassAds와 Set-matching알고리즘을 제공하여 병렬 프로그램을 위한 자원선택 기능을 제공한다. 하지만 자원에 대한 표현력이 부족하여 요구되는 자원과 존재하는 자원이 매칭되지 않는 경우가 자주 발생하는데, 이로 인해 요구하는 자원이 존재하는데도 없는 것으로 판단하여 선택에 있어서 오버헤드가 발생할 수 있다. 글로버스에서 한 사이트는 많은 수의 프로세서를 가진 SMP 머신으로 구성되어 있으며 SCEGrid는 한 프로세서 상에서 다양한 작업을 하도록 규정할 수 없다. 또한 서로 다른 표준의 멀티 사이트구조를 가지기 때문에 각 사이트의 최적 부하를 계산하기 어렵다는 단점과 스케줄링 방식은 라운드로빈 방법만이 가능하다는 단점을 가지고 있다[4,5].

따라서 본 논문에서는 그리드 컴퓨팅상의 다양한 참여자(Grid Volunteer)의 시스템 성능과 각 참여자들의 동적 정보를 토대로 신인도(Credibility)를 고려하여 참여자의 이력 정보(History Information)를 유지하는 알고리즘을 제안한다. 처음에 마스터 노드(Master)는 현재 접속되어 있는 참여자들의 정적 시스템 정보를 바탕으로 참여자를 그룹핑한 후, 그리드 컴퓨팅 요청 작업을 부여하며, 참여자 작업의 처리 결과에 따라 순위를 조정한다. 마스터는 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 응용 프로그램 성격에 따라 자원을 적절히 분배시켜 전반적인 그리드 컴퓨팅 기반 시스템의 성능을 향상 시킬 수 있다.

본 논문의 2장에서는 관련 연구를 언급하였고, 3장에서 그리드 컴퓨팅 시스템 구조 및 참여자 순위 기법을 제안하였으며 4장에서는 제안한 방법의 구조적 성능 평가를 수행하였고 5장에서 결론 및 향후 연구 방향에 대해 논하였다.

2. 관련연구

그리드 컴퓨팅 환경에서 마스터(Master)와 참여자(Worker) 모델을 따르는 응용 프로그램은 참여자들을 확보하고 메시지를 전달할 수 있는 객체지향 프레임워크를 제공한다[6-8]. 즉 마스터와 참여자 모델을 이용하여 자원에 대한 효율성과 성능을 최대화시키기 위한 애플리케이션의 행동에 관한 기록적 데이타를 사용하는 적응적이고 동적인 알고리즘을 제안하였다. 즉 각 태스크에 대한 평균 실행시간에 관한 통계자료를 이용하여 태스크를 적절한 프로세서에 할당하도록 한다. 마스터와 참여자 모델에서 제시하는 알고리즘은 적응적기법의 Random & Average의 방법으로 처음에는 무작위방법으로 태스크를 할당한 후 태스크의 실행시간에 관한 정보를 얻고 그 정보를 이용하여 다시 태스크를 할당하는 방법의 알고리즘을 제안하였으나 이는 단순히 작업에 대한 실행시간만을 고려하였다.

이질적인 자원으로 구성된 그리드 컴퓨팅 환경의 자원 가용상태는 다양하게 변하게 되는데, 이는 서비스 불능 및 실패 상태의 자원, 참여자로서 등록된 새로운 자원, 지속적으로 작업을 수행하고 있는 자원 등이 공존하고 있기 때문이다. 이와 같은 그리드 컴퓨팅 환경의 동적인 요소 때문에 스케줄링 방법은 분산 환경에 적합하고, 확장성을 가져야 하며, 다양한 에러에 효과적으로 대응할 수 있어야 한다[9-11]. 악의적인 사용자로부터 발생하는 문제를 피하기 위해서, [12-15]에서는 참여자들로부터 정확한 수행 결과값을 얻고, 결과의 신인도 향상을 위한 방법으로 투표(Voting)와 임의조사(Spot-Checking)를 이용하는 방식을 적용하였다. 기존의 결함허용(Fault-tolerance)기술은 패리티기술이나 혹은 체크섬을 이용하여 정확성을 증가시켰지만 이는 악의적인 참여자의 의도적인 공격에 대해서는 약할 수밖에 없기에 본 연구에서는 투표(Voting) 혹은 무작위선출(Spot-checking)과 같은 방법을 이용하였다. 결국 다양한 참여자에게 동일 작업을 부여해 결과값을 비교하거나, 무작위로 선출된 참여자의 결과값들을 조사하여, 악의적인 참여자를 선별하였으나, 이는 임무 수행에 대한 결과만을 고려한 것으로 이러한 결과를 만들어낸 다양한 요소에 대한 고려를 하지 않았다. [16,17]에서는 Ant알고리즘을 적용, 페로몬이라는 자원의 이력 정보를 이용하여 작업에 자원을 할당하는 기법을 제안하였다. Ant알고리즘에서 페로몬이라는 것은 이력정보로 표현이 되며 개미들이 페로몬을 따라 최적의 거리를 이용하여 먹이를 수집하고 운반하듯 태스크에 적절한 최적의 자원을 기존의 정보를 이용하여 매칭시키는 방법이다. 그러나 수행된 작업결과의 옳고 그름에 대한 판단 메커니즘이 적용되지 않았으며, 에러율이나 응답시간에 대한 요소를 제외하고 결과의 정확도 요소만을 고려하여 페로몬 정보를 갱신하는데 초점을 맞추었다. 이외에, 현재 그리드 컴퓨팅의 성능 향상을 위한 자원의 최적화된 스케줄링

연구와 자원의 그룹핑 및 순위를 위한 다양한 연구가 수행 중이다[18-22]. 본 논문에서는 마스터-참여자(Master-Worker)모델 기반에서 참여자(Volunteer or Grid-Client Node)의 신인도를 측정하기 위하여 투표와 임의조사를 이용하여 결과의 정확성을 측정하였다. 무엇보다도 작업을 배분하기 위해 다양한 사이즈의 크기의 태스크를 분류하는 방법을 연구하고 이렇게 분류된 다양한 사이즈의 태스크를 적절한 자원에 매칭시키기 위한 방법을 제시하였다. 일차적으로 하드웨어정보를 이용하여 다양한 그룹을 만들고 순위를 정한 후 다양한 태스크를 자원에 매칭시켜 수행의 결과를 이용하여 자원의 순위를 정하고 추후에 태스크를 부여하는 기준으로 삼았다. 한 그룹의 자원에 동일한 작업을 부여하고 반환된 값에 따라 각 자원의 정확성을 판단하고, 하트비트(Heartbeat)를 이용하여 시스템의 에러여부를 확인하였다. 결과를 얼마나 빠른 시간에 반환하는지 측정하였고, 이러한 결과들로부터 얻어진 값을 이용하여 작업을 전달하는 메커니즘을 개발하였다.

3. 그리드 컴퓨팅 구조

본 연구에서는 기존의 마스터와 참여자로 구성된 그리드 컴퓨팅 구조를 개선한 계층적 마스터-참여자(Hierarchical Master-Worker)구조를 사용하였다. 마스터는 각 참여자에게 태스크를 할당하고 태스크를 수행한 참여자들로부터 결과를 받아들인다. 각 태스크는 분할될 수 있는 작업들로 구성되어 있다고 가정하며 참여자역할을 하면서 마스터 역할을 할 수 있는 노드는 자신이 받은 작은 태스크를 분할하여 하위 참여자에게 전달한다. 임의의 참여자는 또 다른 참여자의 마스터가 될 수 있으며, 실제 시스템 환경에서 참여자들은 이질적인 단일 참여자 외에 단일 클러스터 시스템이 되는 경우가 많기 때문에 모든 참여자들을 모르더라도 효율적으로 관리할 수 있으며, 또한 다른 참여자들의 마스터 역할을 통하여 확장성을 높일 수 있다. 그림 1은 마스터와 참여자의 계층적 구조를 나타내고 있으며, 마스터는 자신의 일차적 하위 참여자만 인식한 후, 일차적 하위 참여자에게 작업을 분배하고, 각 작업이 끝난 참여자로부터 받은 결과를 통합한다.

3.1 참여자 순위 결정을 위한 시스템 구조

그리드 컴퓨팅에 참여하고자 하는 인터넷상의 참여자는 마스터에게 접속하여 작업을 부여 받기 위한 등록 과정을 거치는 것으로 시작한다.

참여자는 마스터에게 자신의 정적 정보를 등록해야 하며(그림 2의 ①), 마스터는 등록된 참여자 정보를 이용하여 참여자를 그룹핑한다(그림 2의 ②). 그룹을 기준으로 같은 작업을 부여하고 얻은 결과로부터 참여자의 신인도를 갱신한다(그림 2의 ③). 마지막으로, 마스터는 신인도에 따른 등급 및 순위를 기준으로 각 참여자에게 작업을 부여하게 된다(그림 2의 ④). 신인도에 따라 마스터가 작업을 부여할 때 같은 그룹의 참여자에게 유사한 일을 부여하므로써 유사한 성능을 가진 참여자들이 결과를 유사한 시간 내에 줄 확률이 높아지며, 이로 인해 자원의 전체 이용률을 향상시킬 수 있다.

그림 3은 그리드 컴퓨팅 환경에서 마스터 노드의 시스템 구조이다. 크게 작업 관리자(Task Manager)와 자원 관리자(Resource Manger) 및 자원부(Resource Pool)와 데이타베이스(Database)로 구성된다. 작업 관리자는

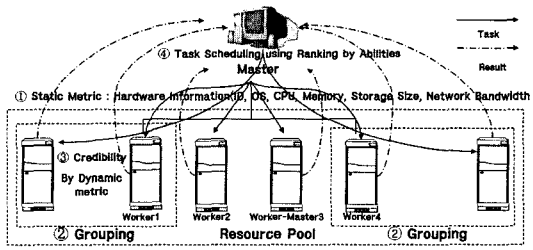


그림 2 그리드 컴퓨팅 환경에서 시스템 서열(Ranking)을 위한 시스템 구조도 및 등록 과정

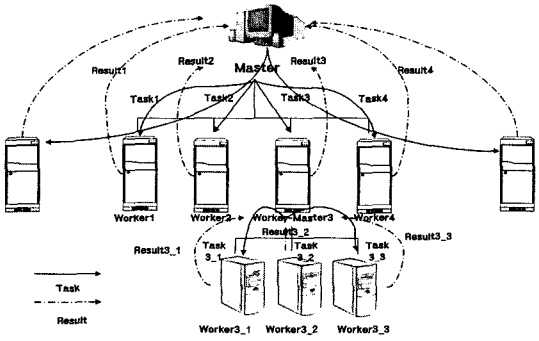


그림 1 확장성을 고려한 계층적 Master-Worker 구조

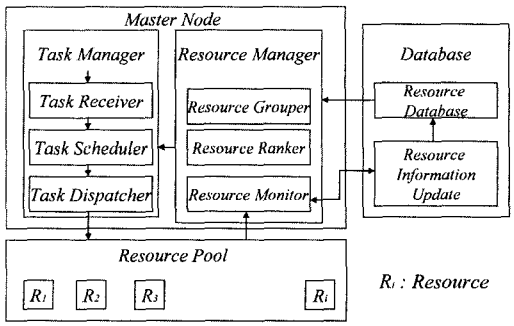


그림 3 그리드 컴퓨팅 환경에서 마스터 노드의 시스템 구조

작업 리시버(Task Receiver), 작업 스케줄러(Task Scheduler), 작업 분배기(Task Dispatcher)로 구성되며, 자원 관리자는 자원 그룹 분류기(Resource Grouper), 자원 순위 분류기(Resource Ranker), 자원 모니터(Resource Monitor)로 구성된다.

- 작업 리시버는 작업의 순위에 따라 혹은 우선순위를 고려하여 그리드 컴퓨팅 요청 작업을 큐에 저장한다.
- 작업 스케줄러는 자원 정적 정보 및 신인도를 고려하여 각 작업에 적절한 자원을 매칭시키는 역할을 한다.
- 작업 분배기는 자원과 매칭된 작업을 각 그룹의 참여자에게 전달하는 역할을 한다.
- 자원 그룹 분류기는 참여자들로부터의 정적 정보를 이용하여 일차적으로 참여자를 그룹핑하는 역할을 한다.
- 자원 순위 분류기는 각 참여자의 정적 정보와 작업 수행능력 요소를 고려하여 계속적으로 참여자를 평가한 후 순위를 부여한다.
- 자원 모니터는 참여자의 작업 수행과정을 모니터링하고 완성된 작업의 성공 여부를 평가하고, 악의적인 참여자를 선별한다.
- 자원 데이타베이스는 참여자의 정적 정보 및 자원의 완성 유·무 및 작업 정보를 지속적으로 저장하고 관·리한다.

3.2 참여자의 정적 정보(Worker's Static Information)

각 참여자들이 그리드 컴퓨팅 시스템에 등록을 원하고, 시스템을 등록할 경우 마스터에게 자신의 시스템 정보를 제공하게 된다. 이는 다양한 참여자의 이질적인 시스템을 최대한 고려하기 위한 목적으로, 처음에 각 참여자에게 작업을 부여할 때 정책적인 값으로 사용할 수 있다. 표 1은 참여자 시스템의 능력에 따라 등급을 부여하기 위한 정적 요소들로, 이 요소에 따라 참여자 그룹핑을 진행한다. 시스템 사양이 업그레이드 될 경우 정보는 갱신될 수 있다.

표 1 참여자 그룹핑을 위한 참여자 정적 정보

ID	각기 다양한 참여자들을 구별하기 위한 ID로 IP주소나 PSID(ProcessID)
운영체제	시스템의 운영체제 정보
C.P.U.	명령어 처리 장치로 계산속도 결정 정보
주 메모리	데이타의 접근 속도를 단축시키는 메모리 공간정보
저장공간	데이타 저장 공간 확보를 위한 장소
Network Bandwidth	각 자원(데이타)의 이동을 위한 통로

수행되어야 할 작업의 집합(T)는 식 (1)과 같으며, 각 작업은 크기에 따라, 혹은 시간 복잡도(Time Complexity)나 우선순위(Priority)에 따라서 정렬될 수 있다고 가정한다. 또한 i번째의 작업이 하위 작업으로 분리되는 경우 j를 이용하여 표현할 수 있다(식 (2) 참조).

$$T_i,\ i = 1,2,\ldots,\ n : \text{태스크 집합} \quad (1)$$

$$T_{i,j},\ j = 1,2,\ldots,\ n : \text{태스크 } i\text{의 } j\text{번째 서브 태스크} \quad (2)$$

자원의 집합을 R이라고 정의하며, R 역시 서브의 참여자로 구분할 수 있다(식 (3),(4)참조). i는 그룹을 j는 그룹 내에서 하나의 참여자를 나타낸다. 단일 참여자일 경우 i그룹의 j번째 참여자로 표현할 수 있으며, 클러스터 참여자인 경우는 k요소를 두어서 하위 참여자를 표현할 수 있다.

$$R_{i,\ j},\ i,\ j = 1,2,\ldots,\ n : i\text{번째 그룹의 } j\text{번째 참여노드} \quad (3)$$

$$R_{i,\ j,\ k},\ k = 1,2,\ldots n : k\text{번째 클러스터노드} \quad (4)$$

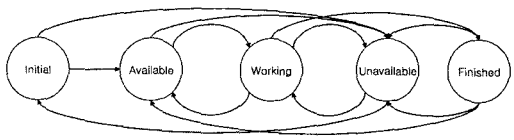


그림 4 자원의 상태 변화도

각 참여자의 상태는 위 그림 4와 같이 나타낼 수 있으며, 이들의 의미는 다음과 같다.

- 초기화(Initial) 상태 : 처음 참여자들이 등록한 상태이며 가용성의 유무를 파악하는 단계이다.
- 가용(Available) 상태 : 현재 자원이 이용 가능한 여부를 나타낸다.
- 불가용(Unavailable) 상태 : 업무 수행 결과의 성능이나 시스템 결함 때문에 현재 자원의 이용이 불가능한 경우를 나타낸다.
- 작업수행(Working) 상태 : 현재 작업을 수행 중이며, 현재 자원의 가용상태에 따라 가용상태나 완료 상태가 될 수 있다.
- 완료(Finished) 상태 : 주어진 업무를 완성한 상태이며 결과의 정확도에 따라 가용한 상태나 불가능한 상태로 변화될 수 있다.

가용한 상태인 자원의 정적 정보를 이용한 그룹핑 기법은 여러 참여자들에게 같은 작업을 부여하고 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 시스템(Max(*Turn Around Time*$_{i,j}$))에 의해 전체 그리드 컴퓨팅 시스템 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.

각 i번째 그룹, j번째 참여자의 CPU 정보($CPU_{i,j}$)와 주메모리($RAM_{i,j}$)및 저장 공간($DSA_{i,j}$:Data Storage Area) 정보, 그리고 네트워크 대역폭($NBD_{i,j}$:Network Bandwidth)은 아래와 같이 표현될 수 있다. 네트워크 대역폭은 자원과 참여자 사이의 대역폭을 의미한다.

$$CPU_{i,j} = \{CPU_{i,1}, CPU_{i,2}, CPU_{i,3}, ..., CPU_{i,j}\} \quad (5)$$

$$RAM_{i,j} = \{RAM_{i,1}, RAM_{i,2}, RAM_{i,3}, ..., RAM_{i,j}\} \quad (6)$$

$$DSA_{i,j} = \{DSA_{i,1}, DSA_{i,2}, DSA_{i,3}, ..., DSA_{i,j}\} \quad (7)$$

$$NBD_{i,j} = \{NBD_{i,1}, NBD_{i,2}, NBD_{i,3}, ..., NBD_{i,j}\} \quad (8)$$

또한 이러한 정적 정보는 응용 프로그램의 성격에 따라 가중치를 두어 시스템에 순위를 부여할 수 있다. 가령, 그리드 컴퓨팅 기반의 계산 컴퓨팅에서 작업을 분배하는 기준은 CPU의 속도에 더 많은 가중치를 부여하며, 데이타 그리드(Data Grid)에서는 저장장소의 크기에 더 많은 가중치를 부여할 수 있다. 또한, 데이타 저장 크기나 혹은 네트워크 대역폭을 초기 자원 정적 정보인 일차정보(Primary Resource Information)와 가중치 ($\alpha,\beta,\gamma,\delta$)를 고려하여 값을 산출할 수 있으며, 각 자원별 성능값은 모든 참여자의 정적 정보 중 가장 좋은 성능을 가진 참여자의 값을 나누어서 구할 수 있다.

$$PRI_{i,j} = \alpha \cdot CPU_{i,j} + \beta \cdot RAM_{i,j} + \gamma \cdot DSA_{i,j} + \delta \cdot NBD_{i,j}$$
$$(\alpha+\beta+\gamma+\delta=1) \quad (9)$$

3.3 참여자의 동적 정보(Worker's Dynamic Information)

참여자의 정적 요소와 아래에 설명되는 동적 요소를 고려하여 각 참여자의 순위 및 그룹을 나눌 수 있으며, 그림 3에 표시된 순서에 의해서 참여자의 신인도를 지속적이며 실시간으로 관리할 수 있다. 참여자의 동적 정보는 관련 연구에서 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 실패율(Failure Rate), 응답시간(*Turn Around Time*), 정확도(Correctness)의 세 요소를 이용한다.

각 참여자의 작업 수행 결과 동적 정보를 다루기 위해서, 실패율(FAR), 결과값 반환시간(TAT), 결과의 정확도(CRT)를 다음과 같이 정의한다.

$FAR_{i,j}$: i그룹의 j번째 참여자의
상대적 에러 발생 빈도율 ($i,j=1,2,...,n$) (10)

$TAT_{i,j}$: i그룹의 j번째 참여자의
상대적 응답시간율 ($i,j=1,2,...,n$) (11)

$CRT_{i,j}$: i그룹의 j번째 참여자의 정확도 ($i,j=1,2,...,n$) (12)

특정 그룹 i의 $ATAT$(Average Turn Around Time), 평균 결과값 반환시간은 식 (13)과 같으며, $ATAT$의 표준편차, 즉 그룹 i에 있어서 반환시간의 표준 편차 결과는 식 (14)를 통해 얻을 수 있다.

$$ATAT_i = \frac{\sum_{j=1}^{n} TAT_{i,j}}{n}, \quad n : i \text{ 그룹 내에 포함된 참여자 수} \quad (13)$$

$$SD_i = \sqrt{\frac{\sum_{j=1}^{n} (TAT_{i,j} - ATAT_i)^2}{n-1}} \quad (14)$$

즉, 비슷한 성능을 가진 참여자들이 결과를 유사한 시간 내에 줄 확률이 높은 참여자의 그룹을 선별하는 과정은 식 (15)에 의한 방법으로 확인할 수 있다. 다시 말해서 최소 표준편차를 가진 참여자 집합이 가장 적절하게 그룹핑이 이루어진 것으로 정의할 수 있다.

$$Min\{SD_1, SD_2, ..., SD_i\} \quad (15)$$

이는 같은 작업에 관해서 결과를 비교하기 위한 효율적인 방법이 될 수 있다. 작업을 할당 받은 참여자들은 에러, 계산 시간 및 결과 정확도 정보를 전달하고, 이 정보는 마스터 노드가 참여자의 신인도 계산을 위한 자

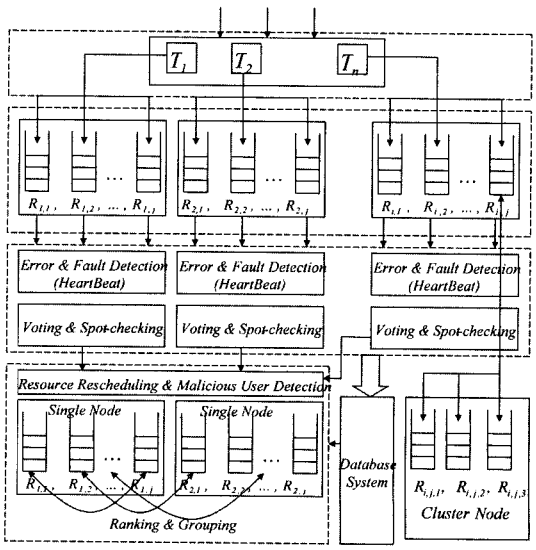


그림 5 자원 할당, 결과 비교 및 순위 부여를 위한 전체 시스템 구조도 (R :참여자, T : 작업)

료로 이용한다. 그림 2의 ①,②,③,④순서는 위에서 설명한 등록부터 순위를 부여하는 과정을 순차적으로 나타내고 있다.

그림 5는 작업 매니저의 작업 리시버가 작업을 큐에 삽입하는 과정을 시작으로 각 작업을 참여자에 부여하고 결과를 확인한 후 그 결과를 이용하여 자원의 순위를 결정하는 전체적 구조를 나타낸 그림이다.

3.3.1 실패율(Failure Rate)

실패율은 참여자들이 작업도중 에러를 발생하게 될 확률로 기존의 실패정보를 이용하여 결정한다. 각 참여자의 실패여부를 판별하기 위하여 개선된 하트비트를 이용하였으며 구조는 그림 6과 같다.

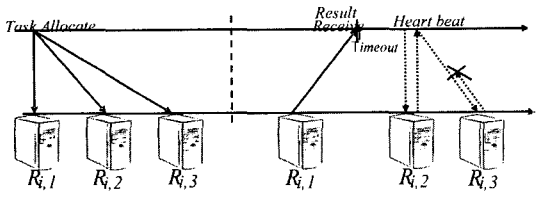


그림 6 그룹 i의 시스템 실패 여부 확인 모델

자원 관리기는 그룹의 각 자원에게 작업을 부여하고 결과를 기다린다. 결과는 참여자로부터 가장 처음 받은 시점을 기준으로 타임아웃(Timeout)을 적용하고 동일 그룹의 다른 참여자가 그 시간이 지나도록 결과를 반환하지 않는 경우 하트비트를 보낸다. 하트비트에 대한 응답을 하지 않을 경우 참여자 실패(Failure)로 간주하고 참여자의 실패율 정보를 갱신한다. 그림 6에서 $R_{i,3}$은 실패한 참여자로 간주된다. 또한 실패율은 식 (16)과 같이 그룹 i에서 최대 에러 발생수와 최소 에러 발생수, 참여자의 에러 발생수를 이용하여 구할 수 있다.

$$FAR_{i,j} = \frac{Max(ETC_{i,j}) - ETC_{i,j}}{Max(ETC_{i,j}) - Min(ETC_{i,j})}$$

$$0 \le FAR_{i,j} \le 1$$

$$If\ ^{\forall} ETC = 0; FAR = 1 \qquad (16)$$

$ETC_{i,j}$: j^{th} *Node Error Total Count in* i^{th} *Group*

3.3.2 응답시간(Turn Around Time)

응답시간은 참여자들이 작업을 받아 처리한 후 값을 반환하기까지 소요되는 시간으로 실질적인 처리시간(Processing) 정보를 구해야 하지만, 실제 네트워크 상황 정보를 포함하는 응답시간을 구하는 것이 구현하는데 더 효율적인 방안이다.

그림 7은 응답시간 모델을 보여주며 응답시간율(TATR)은 아래의 식 (17)을 적용하여 구할 수 있다.

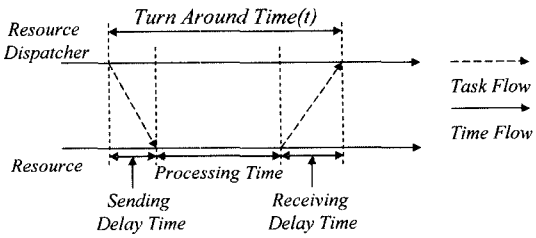


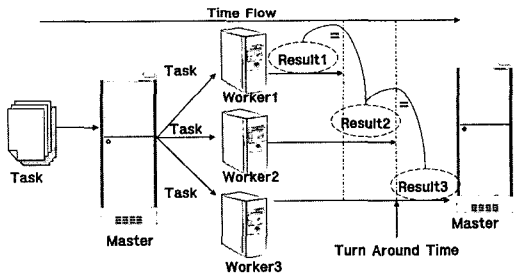
그림 7 결과의 응답시간 모델

$$TATR_{i,j} = \frac{Max(TATR_{i,j}) - TATR_{i,j}}{Max(TATR_{i,j}) - Min(TATR_{i,j})} \qquad (17)$$

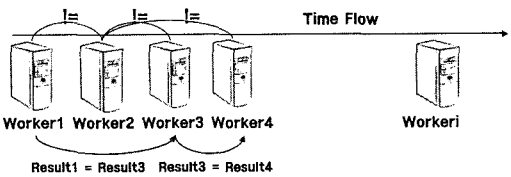
$TATR_{i,j}$: j^{th} *Node Turn Around Time Rate in* i^{th} *Group,* $0 \le TATR_{i,j} \le 1$

3.3.3 정확도(Correctness)

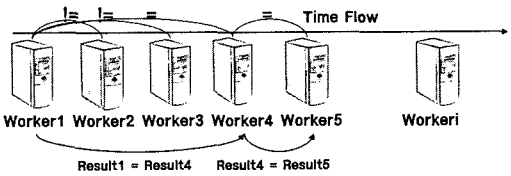
정확도는 참여자들이 작업을 마치고 난 후 반환된 값들을 비교하여 평가하며, 동일 그룹의 같은 작업을 부여받은 참여자로부터 결과를 비교한다. 동일 그룹 내에서 시간의 순서에 따라 과반수 이상이 동일한 결과 값을 반환한 경우 그 값이 정확한 값이라고 간주하고 나머지 참여자에 대해서는 정확한 값인지에 관한 확인 여부만



(a)



(b)



(c)

그림 8 결과의 정확성 측정 기법 모델 예

을 갖는다.

그림 8은 다양한 참여자와 함께 현재 동일 그룹의 참여자가 다섯이라고 가정하였을 경우를 예제로 나타내고 있다. 다섯 참여자 중에서 세 개의 참여자가 같은 결과를 반환한 경우 그 결과는 정확한 것으로 간주하고 시간상 그 후에 결과를 반환한 참여자에 대해서는 그 결과만을 비교하게 된다. 그림 8 (a)는 시간순서대로 세 참여자가 모두 정확한 값을 주었을 때를 보여주며, 그림 8 (b)는 첫 번째 참여자와 세 번째, 네 번째 참여자가 같은 값을 넘겨준 경우를 나타내며, 그림 8 (c)는 첫 번째, 네 번째, 다섯 번째 참여자가 같은 값을 넘겨준 경우를 보여주는 그림이다. 실제로 다양한 경우가 존재하지만, 결국, 세 참여자가 동시에 값을 넘겨준 경우 네트워크의 전송과 비교 판단에 따른 오버헤드가 줄어들어 전반적인 시스템 향상을 가져온다. 각 참여자의 정확도(Correctness)는 할당 받은 작업의 총 수와 정확한 값을 전달한 개수를 이용하여 아래와 같은 공식으로 구할 수 있다.

$$CRT_{i,j} = \frac{CCN_{i,j}}{TTN_{i,j}}, \quad 0 \le CRT_{i,j} \le 1 \tag{18}$$

$CCN_{i,j}$: j^{th} *Node Corret Count Number in* i^{th} *Group*

$TTN_{i,j}$: j^{th} *Node Total Task Number* i^{th} *Group*

각 참여자의 동적 정보(에러율, 응답시간, 정확도)를 바탕으로 i번째 그룹의 j번째 참여자의 동적 정보를 아래의 식과 같이 나타낼 수 있으며 에러율과 응답 시간율, 정확도의 합으로 계산되며 정적 정보와 마찬가지로 중요도에 따라 가중치(ε,η,κ)를 부여한다.

$$DINF_{i,j} = \varepsilon \cdot ERR_{i,j} + \eta \cdot CRT_{i,j} + \kappa \cdot TATR_{i,j} \quad (\varepsilon+\eta+\kappa=1) \tag{19}$$

3.4 참여자 신인도와 자원 정보 테이블

각 i번째 그룹의 j번째 참여자의 순위는 아래의 신인도를 나타내는 CRD값으로 결정이 되며, 가중치(τ,υ)를 적용한 참여자의 정적 정보와 동적 정보의 합으로 아래의 공식(20)을 이용한다.

$$CRD_{i,j} = \tau \cdot PRI_{i,j} + \upsilon \cdot DINF_{i,j} \quad (\tau+\upsilon=1) \tag{20}$$

표 2는 각 참여자의 시스템 정적 정보를 나타낸 것으로 그룹을 나누기 위한 정적 정보이다. 이 수치는 테스트 환경을 위하여 구성된 시스템 사양에 대한 정보이다. 9대에 대한 정적 정보를 나타냈으며, 총 9대중 사양별로 3대를 그룹 1로 정했으며, 또 다른 세대는 그룹 2로 정하였다. 나머지 노드는 그룹 i로 정하였으며, 각 그룹은 각 PRI값(식 (12) 참조)에 의하여 그룹이 결정된다. 여기서 가중치($\alpha_{0.25}, \beta_{0.25}, \gamma_{0.25}, \delta_{0.25}$)는 모두 같다고 정의하고 자원의 정적 정보를 구하였다. 실제 많은 참여자들이 계속 등록되고 나가는 과정을 생략 표시로 나타내었다. 표 3은 위의 정적 정보를 이용하여 파일 I/O(600MB R/W)로부터 발생한 동적 정보의 예를 보인 것이다.

표 2 정적 요소를 고려한 참여자 그룹 테이블 ($\alpha(0.25)+\beta(0.25)+\gamma(0.25)+\delta(0.25)=1$)

요소 / 구분	정적 정보(Static Information)					
	OS	*CPU*	*RAM*	*Storage Size*	*Network Bandwidth*	*PRI*
$R_{1,1}$	Win XP	933Mhz	256MB	40GB	100M	0.275
$R_{1,2}$	Win XP	933Mhz	256MB	40GB	100M	0.275
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$R_{1,3}$	Win NT	933Mhz	256MB	40GB	100M	0.275
$R_{2,1}$	Win NT	933Mhz	512MB	40GB	100M	0.587
$R_{2,2}$	Win NT	933Mhz	512MB	40GB	100M	0.587
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$R_{2,j}$	Win NT	933Mhz	512MB	40GB	100M	0.587
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$R_{i,1}$	Win NT	1.8Ghz	512MB	80GB	100M	0.791
$R_{i,j-1}$	Win NT	1.8Ghz	512MB	120GB	100M	0.875
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$R_{i,j}$	Win XP	1.533Ghz	1GB	120GB	100M	0.962

표 3 동적 요소를 고려한 참여자 그룹 테이블 ($\varepsilon(0.25)+\eta(0.5)+\kappa(0.25)=1$)

구분 \ 요소	동적 정보(Dynamic Information)			
	FAR	***TATR***	***CRT***	***DINF***
$R_{1,1}$	1	0.4	1	0.7
$R_{1,2}$	1	0.8	1	0.9
⋮	⋮	⋮	⋮	⋮
$R_{1,3}$	1	0.6	1	0.8
$R_{2,1}$	1	0.6	1	0.8
$R_{2,2}$	1	1	1	1.0
⋮	⋮	⋮	⋮	⋮
$R_{2,j}$	1	0.7	1	0.85
⋮	⋮	⋮	⋮	⋮
$R_{i,1}$	1	0.9	1	0.95
$R_{i,j-1}$	1	1	1	1.0
⋮	⋮	⋮	⋮	⋮
$R_{i,j}$	1	1	1	1.0

가중치는 그 합이 1이 되도록 가중치를 각각 $\varepsilon(0.25)$, $\eta(0.5)$, $\kappa(0.25)$로 두었는데 이는 파일 I/O에 있어서 성능을 좌우하는 것은 시스템의 정확도나 에러율보다는 네트워크 상황이나 서버의 오버헤드에 의해 큰 영향을 받으므로 의도적으로 응답시간을 두 배로 설정하였다. 표 3에서 시스템 사양이 같더라도 시스템 내부의 문제로 다양한 에러를 발생시킬 수 있으며, 네트워크의 부하에 따른 응답시간의 지연으로 다소 차이가 날 수 있음을 보여준다.

표 4는 시스템의 정적 요소와 동적 요소를 고려한 참여자의 신인도를 나타낸 것이다. 신인도는 동적 정보에 의하여 계속 갱신될 수 있으며, 자원의 순위를 결정하는 값이 된다. 자원의 일차적 정적 정보와 동적 정보를 이용하여 신인도를 계산하고, 신인도에 따라 그 자원의 순위를 결정하며 작업을 계속 수행함에 있어 변화량($\Delta\omega$)을 추가시켜 지속적으로 갱신한다(식 21 참조). 실제로 새로운 신인도는 기존의 신인도와 한 작업을 마치고 난 후 한작업에 대한 신인도를 이용하여 부여받은 태스크의 모든 개수로 나눈 것이다.

$$CRD_{i,j}^{new} = CRD_{i,j}^{old} + \Delta\omega \tag{21}$$

$$CRD_{i,j}^{new} = \frac{CRD_{i,j}^{old} \times (n-1) + (PRI_{i,j}perOneTask + DINF_{i,j}perOneTask)}{n}$$

$$n = The\ TotalNmberofAllocated\ Task\ into\ R_{i,j}$$

표 4 정적 요소와 동적 요소를 고려한 참여자 그룹 테이블 ($\tau(0.5)+\upsilon(0.5)=1$)

구분 \ 요소	***PRI***	***DINF***	***CRD***	***Ranking***
$R_{1,1}$	0.275	0.7	0.4875	9
$R_{1,2}$	0.275	0.9	0.5875	7
⋮	⋮	⋮	⋮	⋮
$R_{1,3}$	0.275	0.8	0.5375	8
$R_{2,1}$	0.587	0.8	0.6935	6
$R_{2,2}$	0.587	1.0	0.7935	4
⋮	⋮	⋮	⋮	⋮
$R_{2,j}$	0.587	0.85	0.7185	5
⋮	⋮	⋮	⋮	⋮
$R_{i,1}$	0.791	0.95	0.8955	3
$R_{i,j-1}$	0.875	1.0	0.9375	2
⋮	⋮	⋮	⋮	⋮
$R_{i,j}$	0.962	1.0	0.981	1

4. 성능 평가

그리드 컴퓨팅 시스템의 정적 성능을 평가하기 위해서 각 하드웨어 정적 정보의 가중치를 1이라고 가정하였다($\alpha+\beta+\gamma+\delta=1$). 우선, 시스템 정적 정보, 즉 시스템 사양(정적 정보)에 따른 파일 I/O의 결과를 살펴보았으며, 표 4에서 i그룹의 참여자는 같은 그룹이 아니기 때문에 실험의 대상에서 제외하였다. LAN상에서 동일 그룹의 참여자에게 작업을 부여하였을 경우는 거의 편차를 보이지 않았다. 그림 9는 신인도가 다른 다섯 개의 참여자를 이용하여 각 100-600MB의 다양한 크기의 파일을 읽어 오는 시간을 측정한 결과이다.

아래 그림 9는 LAN상에서 이루어진 실험의 결과이며, 파일 I/O 시간은 실제로 시스템 성능에 크게 영향을 받지 않았다. 오히려 성능이 더 낮은 참여자로의 파일 입출력 시간이 더 짧은 경우도 있었으나. 신인도에 따라 파일 I/O 시간이 조금씩 줄어드는 것을 확인할 수 있다. 즉, 시스템 성능보다 파일을 주고받는 시점의 네트워크 환경과 서버의 작업 진행 당시 상태에 크게 영향을 받는 것으로 간주된다. 비록 본 실험에서는 신인도에 따른 작업의 성과를 확연히 확인할 수 없었지만 동적정보에

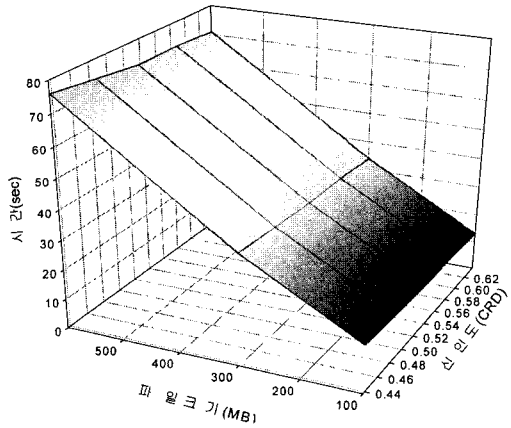


그림 9 정적 정보(PRI)와 파일 크기에 따른 파일 I/O 시간 관계

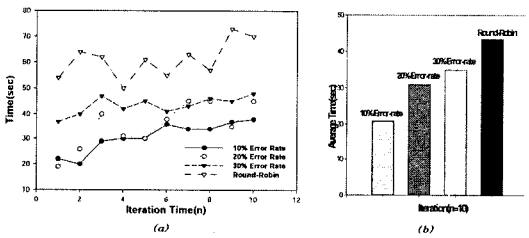


그림 10 이전 정보를 이용한 경우와 무작위 추출 시 정확도 비교

대한 가중치를 기존의 기록 정보를 이용하여 시스템 환경에 적절히 이용한다면 신인도에 따른 작업시간에 대한 관계를 확인할 수 있을 것이며 결국 신인도라는 것은 동적정보를 포함하고 있기 때문에 이러한 네트워크 상태의 변화에 따른 동적인 환경을 잘 반영할 수 있다.

그림 10은 그룹 내에서 자원 선택의 정확도 측면에서 성능을 살펴보았다. 그림 10 (a)는 반복 회수별로 자원 선택을 무작위로 하여 작업과 매칭시킬 경우와 오차율에 따른 연산시간을 보여주며 그림 10 (b)는 평균 시간을 보여준다. 본 연구에서 태스크에 적절한 자원의 매칭을 하기 위해 태스크의 사이즈와 자원의 성능을 고려하였다. 태스크의 사이즈는 데이타의 양이나 알고리즘에 대한 시간복잡도로 생각을 할 수 있으며 자원의 측면에서 살펴보면 성능에 따른 태스크처리에 대한 객관적 자료를 고려하기에는 어려움이 있으므로 가용한 자원에서의 상대적인 자원을 고려한 것이다. 태스크와 자원사이에 있어 객관적인 상관관계를 적용하기에는 어려움이 있으나 참여자의 자원중 cpu나 메모리사이즈가 큰 것이 더 낳은 성능을 가지고 있다고 가정한고 가장 큰 사이즈의 태스크를 가장 좋은 성능의 자원에게 매치시키는

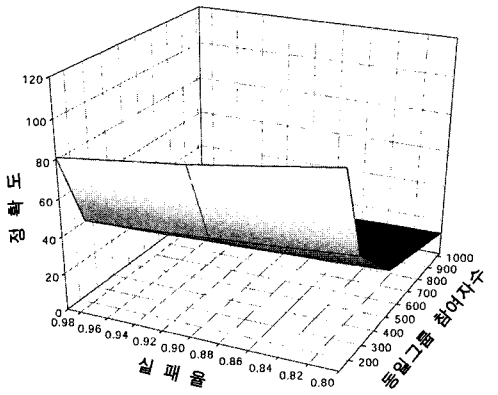


그림 11 신인도와 참여자수에 따른 연산 소요 시간 그래프

경우 이는 자원이 적절하게 매치가 이루어진 것이라 할 수 있다. 이러한 태스크에 대한 자원의 할당이 제대로 이루어지지 않은 경우를 오차라 가정하였다. 태스크의 사이즈는 시간복잡도나 공간복잡도와 같은 요소로 결정이 된다. 본 연구에서는 시간복잡도를 태스크의 크기로 가정하여 실험을 하였으며 태스크의 사이즈에 따라 적절한 자원이 할당되어야 한다. 태스크의 사이즈가 1인 경우에 신인도가 1인 자원이 할당되었을 때 가장 적절한 자원이 할당된 것이며 사이즈가 1인 태스크에 대해 0에 가까운 자원이 할당된다면 부적절한 매칭으로 효과를 극대화시킬 수 없다.

신인도는 0.1에서 1까지의 값 중 하나를 가진다고 가정하였으며, 작업의 사이즈도 0.1에서 1의 크기를 가진다고 가정하였다. 자원의 매칭에 대한 오차율을 10%, 20%, 30%로 설정하였을 경우, 총 작업 수행 연산 시간과 오차율을 적용하지 않은 라운드로빈 기법으로 자원을 스케줄링 하였을 경우 반복수에 따른 총 작업 수행 연산시간을 나타낸 결과이다. 여기서 오차율이란 태스크와 자원에 대한 매칭의 정확도를 나타낸 것으로 10%의 오차율이란 각 태스크에 따른 자원의 매칭에 대한 정확성이 90%라고 가정한 것이다. 신인도의 오차율이 30% 이하라고 가정한 경우 라운드로빈 기법으로 자원을 매칭하는 것보다 더 좋은 성능을 나타냈으며, 신인도의 오차율이 작을수록 더 짧은 연산시간이 필요하다. 라운드로빈 기법의 경우 자원의 이전정보 즉, 신인도를 고려하지 않은 경우 정확도가 떨어져 전체적인 연산시간이 길어지며 이전정보를 이용하는 경우에는 약 10%의 실패율(오차율)을 적용한다고 하더라도 최고 두배 이상의 성능 효과를 얻을 수 있다.

그림 11은 참여자 수와 신인도를 고려하여 연산시간을 산출하는 방식과 참여자수와 에러율에 따라 정확도

를 나타낸다. 위 그림 11에서 신인도는 연산시간과 비례한다고 가정하였을 때, 총 작업 시간을 측정한 것이다. 신인도와 실제 참여자수가 비례할 때, 총 연산시간을 보여주는 그래프로 신인도가 높아질수록 오버헤드가 줄어들고 참여자가 많아져 실제 연산시간에 있어서 많은 성능 향상을 보인다. 이는 실패율이 낮아짐에 따라 더 많은 참여자가 투표에 임하게 되고, 많은 참여자가 투표에 임할 때 정확한 값을 얻을 확률이 높아지기 때문이라 생각한다.

그림 12는 참여자와 정확도에 따라 실패율을 보여주는 그래프로 투표방법을 사용하여 같은 작업을 한 유사 그룹의 참여자들 중 과반수 이상의 참여자가 같은 값을 반환할 경우 그 값은 정확하다고 했을 때, 정확도를 측정한 것이다. 동일 그룹내 참여자들이 각각 10%(0.1)에서부터 40%(0.4)의 실패율, 즉 잘못된 값을 넘겨준다고 가정하였을 때 정확도를 구한 것이다. 한 예로, 40%(0.4)의 실패율을 가진 세 참여자가 60%(0.6)의 정확한 값을 넘겨줄 확률은 약 63%(0.63)정도라고 할 수 있다. 따라서, 참여자가 많고 실패율이 낮을수록 정확도는 증가한다는 것을 확인할 수 있다.

그림 13은 모니터링 주기에 따른 오버헤드의 관계를 보여주는 그래프로 M을 CPU, 메모리, 네트워크를 포함한 모니터링에 의한 오버헤드, p를 모니터링 주기동안 자원의 정보 변화가 일어날 가능성이라 하고 t를 모니터링 주기라고 가정하였을 때 최적 주기 시간을 나타내는 그래프이다. 오버헤드(M)가 같은 경우 자원의 정보가 변화할 가능성이 커질수록 변화가 잦은 시스템이므로 좀더 짧은 주기를 주어야 정확한 자원 정보를 얻을 수 있을 것이다. 또한 같은 자원의 정보가 변할 가능성을 가질 때 손실률이 클수록 주기가 짧아져야 정확한 자원의 정보를 얻을 수 있을 것이다.

본 연구에서는 성능 비교를 위하여 [23]에서 사용된 성능 평가 요소를 사용하였다. 5분에서 11분의 연산 시간을 가지는 태스크가 실험에서 동일하게 사용되었으며 다섯 종류의 서로 다른 우선순위를 부여하였다. 각 그룹

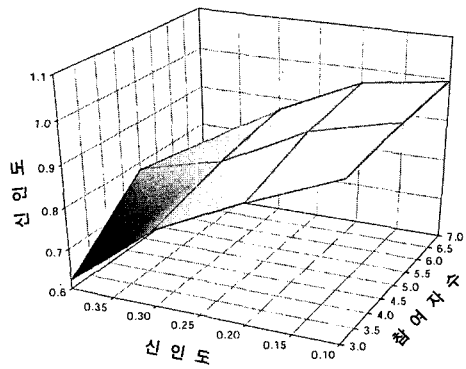
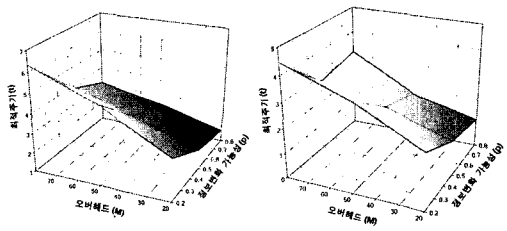


그림 12 에러율과 참여자수를 고려한 정확도 측정 그래프



a) 손실비용이 20%인 경우 b) 손실비용이 40%인 경우

그림 13 모니터링 주기 변동에 대한 오버헤드 변화

은 세 개의 노드로 구성되어 있으며 태스크 종류에 따라 31개의 그룹으로 나누었다. 연산 수행 시간은 그룹에서 가장 빠른 응답시간을 가진 것으로 측정하였다. 스케줄링 비교를 위한 자원과 태스크에 대한 유형은 표 5와 같다.

시스템 성능 평가를 위해 데드라인 정렬(Deadline Sort)기법과 선입선출 기법(First-come First-servered)을 이용하여 성능을 비교하였다. 그룹 및 신인도에 따른 스케줄링 기법은 이전 정보를 사용하기 때문에 각 태스크에 대한 정확한 매칭을 통하여 각 자원에 대한 유휴시간을 줄이고 이용률을 증가시키고 오버헤드를 감소시킴으로써 전반적인 성능을 향상시킨다.

표 5 성능 비교를 위한 태스크 및 자원별 유형

Factor / Task Type	Deadline (min: 5-31)	Priority (1-5)	Factor / Resource Type	Group
T_1	5	1	R_1	1
T_2	6	2	R_2	2
T_3	7	3	R_3	3
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
T_{31}	8	2	R_{93}	31

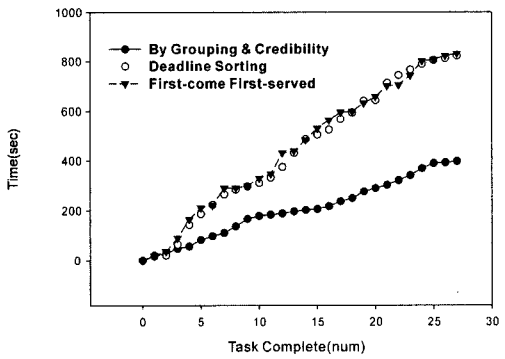


그림 14 스케줄링 알고리즘에 따른 연산수행시간(FCFS, DS, G&C)

5. 결론 및 향후 연구 방향

본 논문에서는 이질적인 자원을 가진 그리드 컴퓨팅 환경에서 각 응용 프로그램의 분할된 작업에 적합한 자원을 맵핑하기 위한 메커니즘을 제공하였으며, 정적 정보와 동적 정보를 이용하여 각 참여자의 순위를 부여하고 부여된 순위에 따라 작업을 제공할 때 전반적인 시스템 성능 향상을 가져왔다. 한편 그리드 컴퓨팅를 적용하는 대상과 참여자수, 태스크의 크기와 수에 따라 그룹핑의 기준은 크게 달라질 수 있으며, 다양한 환경을 고려한 기준 제시가 필요하다. 일차적으로 정확한 하드웨어 성능 정보에 따른 그룹핑 기준을 명확히 하는 연구가 요청된다. 또한 참여자 수와 시스템 오버헤드의 관계를 고려하여 시스템 성능을 향상시키는 방안 및 다양한 응용 프로그램과 각 가중치간의 의존성(Dependancy), 정적/동적 정보와의 관계, 다양한 환경에 따라 가중치등을 고려하여 신인도를 결정하는 알고리즘을 연구할 예정이다.

참 고 문 헌

[1] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," The International Journal of Supercomputer Applications and High Performance Computing, Vol. 11, No. 2, pp. 115-128, Oct. 1997.

[2] K. Krauter, R. Buyya, and M. Maheswaran, "A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing," Software Practice and Experience Journal, Vol. 32, No. 2, pp. 135-164, Feb. 2002.

[3] I. Foster, C. Kesselman, J. Nick, and S. Teucke, "Grid Services for Distributed System Integration," IEEE Computer, Vol. 35, No. 6, pp. 37-46, June 2002.

[4] K. Czajkowski, I. Foster, and C. Kesselman, "Resource Co-Allocation in Computational Grids," Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing, pp. 219-228, Aug. 1999.

[5] C. Liu, L. Yang, I. Foster, and D. 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.

[6] J.P. Goux, S. Kulkarni, M. Yoder, and J. Linderoth, "An Enabling Framework for Master-Worker Applications on the Computational Grid," Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing, pp. 214-217, Aug. 2000.

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

[8] L. Sarmenta, "Sabotage-Tolerance Mechanisms for Volunteer Computing Systems," Proceedings of the 1st ACM/IEEE International Symposium on Cluster Computing and the Grid, pp. 337-346, May 2001.

[9] R. Oldfield and D. Kotz, "Armada: a Parallel File System for Computational Grids," Proceedings of the 1st ACM/IEEE International Symposium on Cluster Computing and the Grid, pp. 194-201, May 2001.

[10] F. Truck, S. Vanhaste, B. Volckaert, and P. Demeester, "A Generic Middleware-based Platform for Scalable Cluster Computing," Future Generation Computer Systems, Vol. 18(4), pp. 549-560, 2002.

[11] Q. Snell, K. Tew, J. Ekstrom, and M. Clement, "An Enterprise-Based Grid Resource Management System," Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing. pp. 83-92, July 2002.

[12] L. Sarmenta, "Studying Sabotage-Tolerance Mechanisms through Web-based Parallel Parametric Analysis and Monte Carlo Simulation," Proceedings of the International Conference on Internet Computing, Vol. 2, pp. 557-563, June 2001.

[13] L. Sarmenta, S. Hirano, and S. Ward, "Towards Bayanihan: Building an Extensible Framework for Volunteer Computing Using Java," ACM Workshop on Java for High-Performance Network Computing in Concurrency: Practice and Experience, Vol. 10(11-13), pp. 1015-1019, 1998.

[14] L. Sarmenta, "An Adaptive, Fault-Tolerant Implementation of BSP for Java-based Volunteer Computing Systems," Proceedings of the 13th International Parallel Processing Symposium on Parallel and Distributed Computing, LNCS 1586,

pp. 763-780, Apr. 1999.
[15] L. Sarmenta, "Bayanihan: Web-Based Volunteer Computing Using Java," Proceedings of the 2nd International Conference on World-Wide Computing and its Applications, pp. 444-461, Mar. 1998.
[16] Z. Xu, X. Hou, and J. Sun, "Ant Algorithm-based Task Scheduling in Grid Computing," IEEE Canadian Conference on Electrical and Computer Engineering, Vol. 2, pp. 1107-1110, May 2003.
[17] C. Chu, J. Gu, X. Hou, and Q. Gu, "A Heuristic Ant Algorithm for Solving QoS Multicast Routing Problem," Proceedings of the 2002 Congress on Evolutionary Computation, Vol. 2, pp, 1630-1635, May 2002.
[18] K. Subramoniam, M. Maheswaran, and M. Toulouse, "Towards a Micro- Economic Model for Resource Allocation in Grid Computing Systems," IEEE Canadian Conference on Electrical and Computer Engineering, Vol. 2, pp. 782-785, May 2002.
[19] J. Cao, D.P. Spooner, S.A. Jarivs, S. Saini, and G.R. Nudd, "Agent-Based Grid Load Balancing using Performance-driven Task Scheduling," Proceedings of the 17th IEEE International Parallel and Distributed Processing Symposium, pp. 49-58, Apr. 2003.
[20] J. Cao, S.A. Jarvis, D.P. Spooner, J.D. Turner, D.J. Kerbyson, and G.R. Nudd, "Performance Prediction Technology for Agent-based Resource Management in Grid Environments," Proceedings of the 16th IEEE International Parallel and Distributed Processing Symposium, pp. 86-99, Apr. 2002.
[21] S. Corsava and V. Getov, "Agent-based Service Management in Large Datacenters and Grids," Proceedings of the 3rd ACM/IEEE International Symposium on Cluster Computing and the Grid, pp. 633-640, May 2003.
[22] J. Cao, D.J. Kerbyson, and G.R. Nudd, "Performance Evaluation of an Agent-based Resource Management Infrastructure for Grid Computing," Proceedings of the 1st ACM/IEEE International Symposium on Cluster Computing and the Grid, pp. 311-318, 2001.
[23] D.P. Spooner, S.A. Jarvis, J. Cao, S. Saini, and G.R. Nudd, "Local Grid Scheduling Techniques using Performance Prediction," IEEE Computer and Digital Techniques, Vol. 150, pp. 87-96, Mar. 2003.

이 진 성
2002년 8월 아주대학교 정보및컴퓨터공학부(공학사). 2004년 8월 아주대하교 정보통신전문대학원 정보통신공학과(공학석사). 2004년 8월~현재 LG전자 MC 사업본부 단말연구소 주임연구원. 관심분야는 클러스터컴퓨팅, Grid Computing, CDMA, Mobile Computing

박 기 진
1989년 한양대학교 산업공학과(공학사) 1991년 포항공과대학교 산업공학과(공학석사). 1991년~1996년 삼성종합기술원 기반기술연구소 전임연구원. 1996년~1997년 삼성전자(주) 소프트웨어센터 선임연구원. 1997~2001년 아주대학교 컴퓨터공학과(공학박사). 2001~2002년 한국전자통신연구원 네트워크장비시험센터 선임연구원. 2002년~2004년 안양대학교 컴퓨터학과 전임강사. 2004년~현재 아주대학교 공과대학 산업정보시스템공학부 조교수. 관심분야는 Embedded Computing, Intrusion Tolerance Systems, Dependable Computing, Grid Computing

최 창 열
1999년 아주대학교 정보및컴퓨터공학부(공학사). 2002년 아주대학교 정보통신전문대학원(공학석사). 2002년~현재 아주대학교 정보통신전문대학원 박사과정 관심분야는 유비쿼터스 컴퓨팅, Autonomic Computing, High Availability

김 성 수
정보과학회논문지 : 정보통신
제 32 권 제 1 호 참조