분산형 P2P 그리드 컴퓨팅 환경에서 평판도 기반 연산 결과 검사 기법

구수진[0§], 박학수[#], 최장원[#], 변은경[§], 길준민[+], 황종선[§]
고려대학교 컴퓨터학과[§], 한국과학기술정보연구원[#], 대구 가톨릭대학교 컴퓨터교육과[+]
{softsj, vision, hwang}@disys.korea.ac.kr[§], {hspark, jwchoi}@kisti.re.kr[#], jmgil@cu.ac.kr[+]

Reputation-based Result-Certification Mechanism in Decentralized P2P Grid Computing Environment

SooJin Koo[§], HarkSoo Park[#], JangWon Choi[#], EunJoung Byun[§], JoonMin Gil[+], ChongSun Hwang[§]
Dept. of Computer Science & Engineering, Korea University[§], Korea Institute of Science and Technology Information[#],
Dept. of Computer Science Education, Catholic University of Daegu[+]

요 약

P2P 그리드 컴퓨팅 환경에서 자원제공자가 수행한 연산 결과의 정확성을 보장하는 것은 중요한 고려사항이다. 기존의 연산 결과 검사에 대한 연구들은 중앙 서버가 연산 결과를 관리하는 중앙집중형 환경에서 이루어졌다. 그러나 중앙집중형 환경에서 중앙 서버의 부하로 인한 확장성 문제로 최근에는 분산형 환경에서 많은 연구가 진행 중이나 연산 결과 검사에 대한 연구는 없는 실정이다. 게다가 분산형 환경은 연산 결과를 관리하는 중앙 서버의 부재로 연산 결과를 신뢰하기 어렵고, 또한 검사하기도 어렵다. 이에 본 논문에서는 분산형 P2P 그리드 컴퓨팅 환경에서 자원제공자의 평판도 기반 연산 결과 검사 기법을 제안한다. 제안 기법은 자원제공자의 가용성과 평판도를 기반으로 작업 트리를 구성한 후, 트리의 단계별 특징에 따라 차별화된 연산 결과 검사와 스케줄링을 수행하여, 기존 연산 결과 검사 기법을 분산형 환경에 그대로 적용할 때보다 연산 결과의 정확성은 보장하면서 연산 결과 검사로 인한 부하는 감소시킨다.

1. 서론

P2P 그리드 컴퓨팅 환경은 그리드 컴퓨팅 환경에 P2P 컴퓨팅 기술을 접목시켜 인터넷에 연결되어 있는 수많은 데스크탑 컴퓨터(자원제공자)의 유휴한 자원을 모아 대규모 연산을 병렬 처리하는 컴퓨팅 패러다임이다. 현재 SETI@HOME[1], Distributed.net[2]과 같은 프로젝트의 성공으로 많은 연구가 진행 중이다.

그러나 P2P 그리드 컴퓨팅 환경은 자원제공자의 연산에 대한 참여와 탈퇴가 자유롭기 때문에 신뢰성에 많은 문제점을 가진다. 특히 부분 작업을 위탁 받은 악의적인 자원제공자가 잘못된 연산 결과를 전달하여 전체 연산 결과를 신뢰하지 못하게 되는 문제점[3]은 연산 결과의 정확성이 중요한 응용에서는 반드시 해결되어야 한다. 본 논문에서는 이러한 문제점을 속임수 문제(Cheating Problem)라 정의한다.

속임수 문제를 해결하기 위해 기존에 제안된 기법은 복제정책(Replication)을 이용한 과반수 투표법(Majority Voting)[4][5]과 이미 알고 있는 결과를 이용하는 발췌 검사법(Spot-Checking)[5][6]으로 분류된다. 제안된 기법 모두 중앙 서버가 자원제공자의 연산 결과를 관리하는 중앙집중형 환경에서 제안되었다. 그러나 중앙집중형 환경은 서버의 부하로 인해 확장성에 문제점을 가지기 때문에 최근에는 분산형 환경에 대한 많은 연구가 진행 중이다[8-10]. 그러나 분산형 환경에서는 속임수 문제를 해결하기 위해 제안된 기법이 없다. 또한 중앙 서버가 아닌 각 자원제공자들이 연산 결과를 검사하기 때문에 전체 연산 결과를 신뢰하기 어렵다.

분산형 환경에서 속임수 문제를 해결하기 위해서는 다음과 같은 사항을 고려해야 한다. (1) 분산형 환경에서는 전역적으로 연산 결과를 검사하는 중앙 서버가 없다. (2) 분산형 환경에서 제안된 기존 연산 모델들[8-10]은 자원제공자의 휘발성과 연산 결과의 정확성을 고려하지 않는다. 따라서 기존 연산 모델에서 사용되는 트리나 그래프 구조에서 가용성이 낮은 자원제공자가 연산 완료 전에 이탈한다면 연산 수행이 단절될 수 있다. 또한 작업의뢰자 가까이에 위치한 자원제공자가 악의적이라면 그 자원제공자에게 작업을 받아 수행한 다른 자원제공자들의 연산 결과도 신뢰하기 어렵다. (3) 중앙집중형 환경에서 속임수 문제를 위해 제안된 기법을 분산형 환경에 그대로 적용하기 어렵다. 예를 들어, 기존에 제안된 분산형 환경은 주로 트리나 그래프 구조에 작업 묶음을 할당하는 작업 가로채기(Work Stealing) 기법을 사용한다. 이 환경의 상위 단계에서 복제정책을 적용 시, 하위 단계에서는 과잉 복제 현상이 발생한다.

따라서 본 논문에서는 분산형 P2P 그리드 컴퓨팅 환경에서 속임수 문제를 해결하기 위해, 위에서 언급한 문제점을 고려한 자원제공자의 평판도 기반 연산 결과 검사 기법과 이를 위한 연산 수행 모델을 제안한다. 제안 기법은 자원제공자의 가용성(Availability)과 평판도(Reputation)를 기반으로 작업 트리를 구성한 후, 구성된 작업 트리의 단계별 특징에 따라 차별화된 연산 결과 검사와 스케줄링을 수행하여, 기존에 제안된 연산 결과 검사 기법을 분산형 환경에 그대로 적용할 때보다 연산 결과의 정확성은 보장하면서 전체 연산 결과 검사로 인한 부하는 감소시킨다.

본 연구는 한국과학기술정보연구원(KISTI) 초고속응용기술지원사업 지원에 의해서 수행되었음

2. 관련 연구

2.1 연산 결과 검사 기법

연산 결과 검사 기법은 크게 과반수 투표법과 발췌 검사법으로 분류된다. 과반수 투표법을 사용한 BOINC[4]는 동시에 여러 노드에게 같은 작업을 할당하여 과반수 이상의 같은 결과가 반환 시에 그 결과를 신뢰하는 기법을 사용한다. 또한 Bayanihan[5]은 선행처리자 우선할당 기법(Eager Scheduling)을 사용하여 일정 개수의 동일한 결과가 올 때까지 중복해서 노드에게 작업을 할당하는 기법을 제안했다. 반면 발췌 검사법을 사용한 Sabotage[5][6]는 발췌 검사를 통과한 개수로 신뢰도를 판단하여 복제의 개수를 달리하는 기법을 제안하였다. XtremWeb[7]도 발췌 검사로 작업자의 신뢰도를 판단하여 검사 부하를 달리하는 기법을 제안했다. Quiz[8]는 분산형 환경에서의 변형된 발췌 검사 기법이나 연산 결과의 정확성 보다는 담합(Collusion) 여부를 판단하는데 초점을 맞췄다. Quiz[8]를 제외한 연산 결과 검사 기법은 모두 중앙집중형 환경에서 제안된 기법이고, 현재 분산형 환경에 적용되는 연산 결과의 정확성을 보장하는 연산 결과 검사 기법은 없다.

2.2 분산형 연산 수행 모델

Organic Grid[9]는 연산이 반복될수록 연산 속도가 빠른 노드가 상위로 올라가는 트리 구조를 사용한다. Messor[10]는 에이전트를 이용하여 랜덤하게 작업을 할당하는 랜덤그래프 구조를 사용한다. Paradropper[11]는 작은 세상 네트워크(Small World Network)를 사용하여 가장 부하가 적은 노드에게 작업을 할당하는 방식을 사용한다. 위의 모델 모두 자원제공자의 휘발성과 연산 결과의 정확성을 고려하지 않았기 때문에 안정적인 연산 수행을 보장하지 못한다.

3. 평판도 및 가용성 기반 작업 트리 구성

분산형 환경에서 중앙 서버의 부재로 자원제공자의 역할은 중앙집중형 환경보다 더 중요하다. 즉, 분산형 환경의 자원제공자는 안정적인 연산 수행을 책임져야 한다. 그러나 기존에 제안된 분산형 환경 모델인 [8-10]은 연산 결과의 정확성과 자원제공자의 휘발성을 고려하지 않았기 때문에 지속적이고 신뢰적인 연산 수행을 보장하지 못한다. 본 장에서는 분산형 P2P 그리드 컴퓨팅 환경에서 연산 결과의 정확성과 자원제공자의 휘발성을 고려한 자원제공자의 평판도와 가용성을 기반으로 작업 트리를 구성하는 방법을 제안한다.

3.1 평판도와 가용성

평판도는 과거 경험을 바탕으로 특정 자원 제공자의 미래 행동의 신뢰도를 예측하는 척도이다. 자원제공자의 평판도를 판단할 때는 수식 (1)과 같이 직접 경험으로 얻는 방법과 수식 (2)와 같이 간접 경험(다른 작업제공자의 추천)으로 얻는 방법이 있다. 수식에서 P_i와 P_j, P_j와 P_k는 친구리스트에 포함된 친구 관계이고, P_i와 P_k는 친구리스트에 포함되지 않은 이웃 관계이다. 평판도는 수행한 연산의 개수와 연산 결과의 정확성으로 계산한다. 그러나 처음 참여하여 평판도 정보가 없는 새로운 자원제공자는 연산을 수행 할 수 있는 최소의 평판도를 제공하여 일정 수의 연산을 수행하도록 보장한다.

$$Rep(P_i \rightarrow P_j) = \left(\frac{O_{Succ}(P_i \rightarrow P_j)}{O_{Total}(P_i \rightarrow P_j)} + \frac{O_{Succ}(P_i \rightarrow P_j)}{O_{Total}(P_i)}\right) \times \frac{1}{2} \quad (1)$$

$$Rep(P_i \rightarrow P_k) = Rep(P_i \rightarrow P_j) \times Rep(P_j \rightarrow P_k) \quad (2)$$

가용성은 자원제공자가 자원을 제공할 준비가 되어있고, 실제로 자원을 공급할 수 있는 확률이다. 자원제공자의 가용성은 수식 (3)과 같이 계산한다. 평판도와 가용성 수식에 따른 기호는 [표 1]을 참고한다.

$$Avail(P_i) = \frac{MTTF(P_i)}{MTTF(P_i) + MTTR(P_i)} \quad (3)$$

[표 1] 평판도와 가용성 계산에 필요한 기호

기호	설명
$Rep(P_i \rightarrow P_j)$	P_j에 대한 P_i의 평판도
$O_{Total}(Pi)$	P_i가 친구 리스트의 친구들과 연산한 전체 개수
$O_{Total}(P_i \rightarrow P_j)$	P_i가 P_j와 연산한 전체 개수
$O_{Succ}(P_i \rightarrow P_j)$	P_i가 P_j와의 연산에서 정확한 결과를 반환한 개수
$Avail(P_i)$	P_i의 가용성
$MTTF(P_i)$	P_i가 자원을 공급한 시간 *(Mean Time To Failure)*
$MTTR(P_i)$	P_i가 자원을 공급하지 못한 시간 *(Mean Time To Repair)*

3.2 평판도 및 가용성 기반 작업 트리 구성 기법

본 논문이 제안하는 작업 트리 구성 기법에서는 자원제공자의 친구 리스트와 이웃 노드들의 평판도 및 가용성에 따라 작업 트리의 각 단계별 임계치를 기준으로 자원제공자를 각 단계에 배치함으로써 작업 트리를 구성한다. 특히, 지속적이고 신뢰적인 연산 수행을 보장하기 위해 평판도와 가용성이 높은 자원제공자를 상위 단계에 위치하도록 작업 트리를 구성한다.

작업 트리는 크게 네 단계로 분류된다. [그림 1]과 같이 해당 단계에서의 자원제공자는 각각 작업책임자, 작업관리자, 작업수행자, 복제자의 역할을 수행한다. 단계 0의 작업책임자는 작업의뢰자에게 전체 작업을 위탁 받아 작업관리자들에게 작업 묶음을 할당하고, 작업관리자들에 대한 정확성 검사를 수행한다. 단계 1의 작업관리자는 하위 단계(단계 2, 3) 자원제공자들에게 단위 작업을 할당하고, 하위 단계 자원제공자들에 대한 정확성 검사를 수행한다. 단계 2의 작업수행자는 할당 받은 단위 작업을 수행한 후, 결과를 상위 단계에 반환한다. 단계 3의 복제자는 상위 단계인 작업수행자의 휘발성으로 인한 연산 수행 중단 및 실패에 대처하기 위해 단위 작업에 대한 복제 연산을 수행한다. 단계 0과 1의 자원제공자들은 작업 할당 및 관리를 수행하므로 단계 2와 3의 자원제공자들에 비해 높은 평판도와 가용성을 가진다.

작업 트리 구성 알고리즘은 다음과 같다.

(1) [그림 2]의 ①과 같이 작업책임자(P_0)는 친구리스트의 평판도와 가용성을 기반으로 트리에 위치할 수 있는 단계를 계산하고 작업 트리를 구성한다.

(2) [그림 2]의 ②와 같이 구성된 작업 트리의 자원제공자(P_1~P_4)는 각각의 친구리스트를 통해서 ③, ④의 작업 트리를 구성한다.

(3) (2)의 과정을 일정 TTL(Time To Live) 동안 반복한다.

이 때, 자원제공자의 정보는 직접 경험뿐 아니라 간접 경험으로 얻을 수 있다. 또한, 부모 노드가 가질 수 있는 자식 노드의 수는 자원제공자의 가용성, 평판도, CPU 처리 속도에 의해 제한된다.

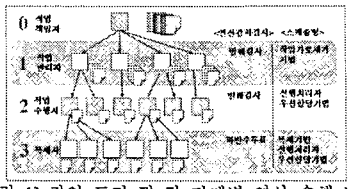
위의 과정으로 작업 트리 구성이 완료되면, 각 자원제공자는 하위 단계의 자원제공자에게 작업을 분배한다. 작업이 완료되면 각 자원제공자는 수행된 작업 결과에 따라 평판도와 가용성 정보를 갱신하고, 이를 바탕으로 작업 트리를 재구성한다.

4. 평판도 기반 연산 결과 검사 기법

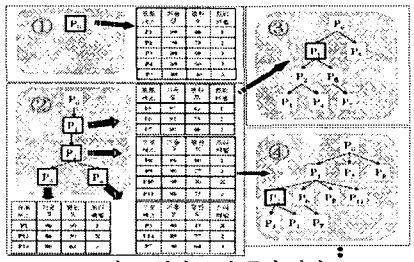
본 장에서는 자원제공자의 평판도와 가용성을 기반으로 구성된 작업 트리에 부합하는 연산 결과 검사 기법과 이를 위한 스케줄링 기법을 제안한다. 본 논문의 제안 기법에서는 작업 트리의 각 단계별로 발췌 검사법과 과반수 투표법을 다르게 사용하고, 또한 작업 가로채기 기법과 선행처리자 우선할당 기법을 각 단계의 특징에 맞게 사용함으로써, 연산 결과의 정확성을 보장하면서 연산 결과 검사로 인한 부하는 감소시킨다.

4.1 연산 결과 검사 기법

제안하는 연산 결과 검사 기법에서는 작업 트리의 상위 단계에서는 발췌 검사법을 적용하고 하위 단계에는 과반수 투표법을 적용한다. ([그림 1] 참고). 과반수 투표법은 중복된 연산 수행으로 연산 결과 검사 부하가 클 뿐 아니라 작업 트리의 상위 단계에서 복제 정책을 적용하면 하위 단계에서 과잉 복제 현상이 발생하기 때문에 하위 단계에 적용시킨다. 또한 발췌 검사법은 과반수 투표법과 달리 휘발성에



[그림 1] 작업 트리 및 각 단계별 연산 수행 방식



[그림 2] 작업 트리 구성 과정

대처하지 못하여 연산 수행 중단 및 실패 현상이 발생하므로 휘발성이 낮은 상위 단계에 적용시킨다.

따라서 단계 0의 작업책임자는 작업관리자의 정확성 검사를 위해 발췌 검사법을 사용한다. 단계 1의 작업관리자는 작업수행자의 정확성 검사를 위해 발췌 검사법을 사용하고, 복제자의 정확성 검사를 위해 과반수 투표법을 사용한다. 또한 단계 1과 단계 2의 자원제공자들의 평판도는 발췌 검사법을 통과하면 증가한다. 단계 3에서는 과반수 투표 과정을 통과한 자원제공자들의 평판도는 증가하고, 그렇지 않으면 감소한다. 연산 결과 검사 기법 알고리즘은 [그림 3]에 기술한다.

발췌 검사 개수와 과반수 투표를 위한 복제 개수는 평판도에 따라 다르게 계산한다. 수식 (4)는 복제 개수를 계산하는 수식이고 수식 (5)는 발췌 검사 개수를 계산하는 수식이다. 두 수식 모두 자원제공자의 평판도가 낮아질수록 작업당 검사 개수가 증가한다. 수식에 따른 기호는 [표 2]를 참고한다.

$$R_{Cnt}(P_i \rightarrow P_j) = 2\left\lceil \frac{m}{Rep(P_i \rightarrow P_j)} \right\rceil - 1 \quad (4)$$

$$S_{Cnt}(P_i \rightarrow P_j) = \left\lfloor \frac{1}{Rep(P_i \rightarrow P_j)} \right\rfloor \quad (5)$$

[표 2] 검사 빈도 계산에 필요한 기호

기호	설명
$R_{Cnt}(P_i \rightarrow P_j)$	P_j에 대한 P_i의 복제 개수
$S_{Cnt}(P_i \rightarrow P_j)$	P_j에 대한 P_i의 단위 작업 당 발췌 검사 개수
$Rep(P_i \rightarrow P_j)$	P_j에 대한 P_i의 평판도
m	과반수 투표를 위한 과반수 (m≥2)

```
//Node receives the result from childNode
switch (Node.level)        //case 0: Node Level 0, case 1: Node Level 1
case 0: if (job = spotter)
          if (spotter.result=false) then
            excludeChildNode(childNode), replaceChildNode()
          else updateReputation(childNode)
case 1: if (childNode.level=2 && job=spotter)
          if (spotter.result=false) then
            excludeChildNode(childNode), replaceChildNode()
          else updateReputation(childNode)
        if (childNode.level=3) then
          majorityVoting(childNodes), updateReputation(childNodes)
```

[그림 3] 연산결과검사 알고리즘

4.2 연산 결과 검사를 위한 스케줄링 기법

제안하는 스케줄링 기법에서는 작업 트리 각 단계의 특징에 맞게, 그리고 각 단계의 연산 결과 검사 기법에 부합되도록 작업 가로채기 기법과 선행처리자 우선할당 기법을 적용시킨다. 단계 0의 작업책임자는 작업관리자의 평판도와 가용성에 따라 작업 묶음을 작업관리자들에게 할당하게 된다. 단계 1의 작업관리자는 작업수행자에게 선행처리자 우선할당 기법을 사용하여 단위 작업을 할당한다. 또한 작업수행자가 휘발성으로 작업 수행이 중단(고장) 될 경우를 대처하기 위해 작업관리자는 선행처리자 우선할당 기법을 사용하여 복제 정책에 따

라 단위 작업에 대한 복제 수만큼 복제자에게 할당한다.

각 단계별로 스케줄링을 수행한 후, 단계 2와 단계 3의 자원제공자들의 연산 수행 결과에 따라 단계 1의 작업관리자들간에서는 작업 부하가 불균형한 상태가 발생한다. 따라서 부하 균형(Load Balancing)을 이루기 위해 작업관리자들간에 작업 가로채기 기법을 사용한다. 이에 따른 스케줄링 알고리즘은 [그림 4]에 기술한다.

```
//Node receives the job request from requestNode (childNode or sibling)
switch (Node.level)          //case 0: Node Level 1, case 1: Node Level 2
case 0: numOfTask = cal(Rep(node→childNode), Avail(childNode))
        if (Node.hasMoreTasks(numOfTask)) then
          giveTasksToChildNode(childNode, numOfTask)
case 1: if (requestNode = childNode) then
          if (Node.hasMoreTasks()=false) then
            if (Parent.hasMoreTasks()) then getTasksFromParent()
            else getTasksFromSibling(numOfTask)
          giveTaskToChildNode(childNode, workUnit)
          giveTaskToGrandChildNodes(numOfReplica, workUnit)
        else if (requestNode = sibling) then
          if (Node.hasMoreTasks()) then giveTasksToSibling()
```

[그림 4] 스케줄링 알고리즘

5. 성능 평가

제안 기법의 성능을 비교, 평가하기 위해 분산형 환경에서 기존의 연산 결과 검사 기법을 적용했을 때와 제안 기법을 적용했을 때의 연산 결과의 정확성과 연산 결과 검사 부하를 비교한다. 연산 결과의 정확성은 자원제공자가 수행한 연산 결과가 신뢰적일 확률, 연산 결과 검사 부하는 실제 연산을 수행한 시간과 연산 결과 검사를 위해 부가적으로 수행한 시간의 비율로 계산한다. [표 3]과 [표 4]는 비교 기법과 제안 기법의 연산 결과의 정확성과 연산 결과 검사 부하를 나타낸 수식이다. 위 수식에 따른 기호는 [표 5]를 참고한다.

[표 3] 비교 기법과 제안 기법의 연산 결과의 정확성 비교

분산형 환경	연산 결과 정확성
검사법 미적용	$\prod_{i=1}^{n}(1-f_{Pi})(1-f_{Par(Pi)})\times\frac{1}{n}$
발췌 검사법	$(\prod_{i=1}^{n}(1-\frac{f_{Pi}}{1-f_{Pi}}\times\frac{1}{se})(1-\frac{f_{Par(pi)}}{1-f_{Par(pi)}}\times\frac{1}{se}))\times\frac{1}{n}$
과반수 투표법	$(\prod_{i=1}^{n}(1-\frac{(4f_i(1-f_i))^m}{2(1-2f_i)\sqrt{\Pi(m-1)}})(1-\frac{(4f_{Par(pi)}(1-f_{Par(pi)}))^m}{2(1-2f_{Par(pi)})\sqrt{\Pi(m-1)}}))\times\frac{1}{n}$
제안 기법	$(\prod_{i=1}^{n}(1-\frac{f_i}{1-S_{Cnt(Pi)}}\times\frac{1}{S_{Cnt(Pi)}e})(1-\frac{f_{Par(Pi)}}{1-S_{Cnt(Par(Pi))}}\times\frac{1}{S_{Cnt(Par(Pi))}e}))\times\frac{1}{n}$ $(\alpha < Rep(P_i)\le 1)$ $(\prod_{i=1}^{n}(1-\frac{(4f_i(1-f_i))^m}{2(1-2f_i)\sqrt{\Pi(m-1)}})(1-\frac{f_{Par(Pi)}}{1-S_{Cnt(Par(Pi))}}\times\frac{1}{S_{Cnt(Par(Pi))}e}))\times\frac{1}{n}$ $(0 < Rep(P_i)\le\alpha)$

[표 4] 비교 기법과 제안 기법의 연산 결과 검사 부하 비교

분산형 환경	연산 결과 검사 부하
검사법 미적용	$\frac{0}{T}\ (T=\sum_{j=1}^{t}T_{RealJob(Wt)})$
발췌 검사법	$(\sum_{j=1}^{t}T_{Spotter(Wt)}\times s)\times\frac{1}{T}$
과반수 투표법	$(\sum_{j=1}^{t}T_{Replica(Wj)}\times(r-1)^{3-level})\times\frac{1}{T}$
제안 기법	$(\sum_{j=1}^{m}T_{Spotter(Wj)}\times S_{Cnt(Pi)}))\times\frac{1}{T}$ $(\alpha < Rep(P_i)\le 1)$ $(\sum_{j=1}^{m}T_{Replica(Wj)}\times(R_{Cnt(Pi)}-1))\times\frac{1}{T}$ $(0 < Rep(P_i)\le\alpha)$

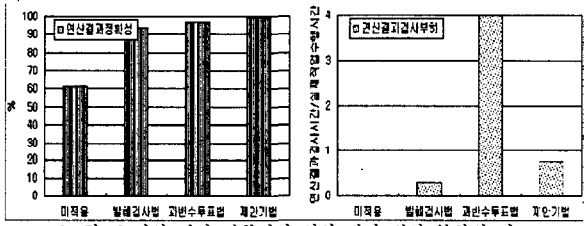
[표 5] 평가를 위해 사용된 기호

기호	설명	기호	설명
n	전체 노드 개수	f	작업제공자의 속임수 확률
s	발췌검사 개수	$T_{RealJob}$	실제작업 수행 시간
r	복제 개수	$T_{Spotter}$	발췌작업 수행 시간
t	전체 작업 개수	$T_{Replica}$	복제작업 수행 시간
W	단위 작업	S_{Cnt}	평판도에 따른 발췌검사개수
$level$	작업 트리의 단계	R_{Cnt}	평판도에 따른 복제개수
α	평판도 임계치	$Par(P_i)$	P_i의 연산결과검사 노드
m	과반수	e	자연로그

[표 6] 평가를 위해 사용된 파라미터

기호(수치)
$n(126)$, $s(5)$, $r(5)$,$t(269)$, $m(3)$, $f(0\sim50\%)$, $\alpha(70\%)$, $Rep(50\sim100\%)$ $T_{RealJob}$, $T_{Replica}(100)$, $T_{Spotter}(10)$

[그림 5]는 비교 기법과 제안 기법의 연산 결과의 정확성과 연산 결과 검사 부하를 나타낸 그래프이다. 평가를 위해 사용된 파라미터는 [표 6]과 같다. 제안 기법은 자원제공자의 평판도 임계치(α)에 따라 작업 트리의 각 단계별로 발췌 검사법과 과반수 투표법을 다르게 적용하고, 평판도에 따라 검사 부하를 다르게 하여, 그래프와 같이 연산 결과의 정확성은 보장하면서 연산 결과 검사로 인한 부하는 감소시킨다.



[그림 5] 연산 결과 정확성과 연산 결과 검사 부하의 비교

6. 결론 및 향후 연구

본 논문에서는 분산형 P2P 그리드 컴퓨팅 환경에서 속임수 문제를 해결하기 위한 평판도 기반 연산 결과 검사 기법과 이를 위한 연산 수행 모델을 제안하였다. 제안된 기법은 자원제공자의 가용성과 평판도를 기반으로 작업 트리를 구성한 후, 구성된 트리의 단계별 특징에 따라 발췌 검사와 과반수 투표 검사를 다르게 적용하고, 이를 위한 스케줄링 기법으로 작업 가로채기 기법과 선행처리자 우선할당 기법을 함께 사용하였다. 평가를 통해 제안 기법은 분산형 환경에서도 연산 결과의 정확성을 보장하면서 연산 결과 검사 부하는 감소 시킴을 확인할 수 있었다. 향후에는 본 제안 기법을 직접 구현하여 성능을 평가할 예정이다.

참고문헌

[1] SETI@HOME, "http://setiathome.ssl.berkeley.edu"

[2] Distributed.net, "http://www.distributed.net"

[3] David Molnar, "The SETI@Home Problem", E-Commerce, 2000

[4] David P. Anderson, "BOINC: A System for Public-Resource Computing and Storage", Grid 2004, pp. 4-10, Nov. 2004

[5] Luis F. G. Sarmenta, "Bayanihan: Building and Studying Web-Based Volunteer Computing Systems Using Java", Future Generation Computer Systems, Vol. 15, No. 5/6, pp. 675-686, Oct. 1999

[6] Luis F. G. Sarmenta, "Sabotage-Tolerance Mechanism for Volunteer Computing Systems", CCGrid 2001, pp. 337-346, May. 2001

[7] C. Germain-Renaud, N. Playez. "Result-Checking in Global Computing Systems", ICS 2003, June 2003

[8] Shanyu Zhao, Virginia Lo and Chris GauthierDickey, "Result Verification and Trust-based Scheduling in Peer-to-Peer Grids", P2P 2005, Sep. 2005

[9] Arjav J. Charkravarti, Gerald Baumgatner, Mario Lauria, "The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network", Systems, Man, and Cybernetics, Vol. 35, No. 3, pp. 373-384, May 2005

[10] A!berto Montresor, Heing Meling and Ozalp Babaoglu, "Messor: Load-Balancing through a swarm of Autonomous Agent", AP2PC 2002, pp. 125-137, July 2002

[11] Wen Dou, Yan Jia, Huai Ming Wang, Wen Qiang Song, Peng Zou, "A P2P Approach for Global Computing", IPDPS 2003, pp.248b, April 2003