P2P 그리드 컴퓨팅 환경에서 이동에이전트 기반 적응적 그룹 스케줄링 기법*

최성진[O] 최장원[†] 박찬열[†] 박학수[†] 정순영[‡] 황종선[††]
고려대학교 컴퓨터학과, 한국과학기술정보원[†], 고려대학교 컴퓨터교육과[‡]
{lotieye[O], hwang[††]}@disys.korea.ac.kr, {jwchoi, chan, hspark}@kisti.re.kr[†], jsy@comedu.korea.ac.kr[‡]

Mobile Agent based Adaptive Group Scheduling Mechanism in P2P Grid Computing Environment

SungJin Choi[O], JangWon Choi[†], ChanYeol Park[†], HarkSoo Park[†], SoonYoung Jung[‡], and ChongSun Hwang[††]
Dept. of Computer Science & Engineering, Korea University [O ††],
Korea Institute of Science and Technology Information[†],
Dept. of Computer Science Education, Korea University[‡]

요 약

P2P 그리드 컴퓨팅 환경에서 피어의 자율성으로 인한 자원 제공의 휘발성과 피어의 이질적인 특성은 스케줄링 과정에서 해결해야 할 중요한 문제이다. 이에 본 논문에서는 이동 에이전트 기반 적응적 그룹 스케줄링 기법(Mobile Agent based Adaptive Group Scheduling Mechanism: MAAGSM)을 제안한다. 제안 기법은 피어의 특성 (즉, 자원제공자 자율성 고장, 자원제공자 가용성, 자원제공 시간)에 따라 피어들을 동질적인 그룹(자원제공자 그룹)으로 구성한 후, 그룹에 적합한 다양한 스케줄링 알고리즘을 이동에이전트 기술을 이용하여 적용한다.

1. 서 론

그리드 컴퓨팅은 조직내의 다양한 컴퓨팅 자원(슈퍼컴퓨터, 데이터, 소프트웨어, 실험장치 등)을 가상 조직(virtual organization)으로 구성함으로써 고성능의 컴퓨팅 파워를 제공하는 컴퓨팅 패러다임이다 [4, 5]. 이에 반해, P2P 그리드 컴퓨팅은 인터넷에 연결된 수많은 유휴 데스크톱 컴퓨터(자원제공자라고 불림)를 사용하여 고성능 컴퓨팅 파워를 이루어 고처리의 분산 컴퓨팅을 수행하는 컴퓨팅 패러다임이다[1-12]. P2P 그리드 컴퓨팅은 하나의 수행 코드와 각기 다른 입력 값을 가진 여러 개의 작업으로 구성된 병렬 컴퓨팅 응용, 즉, 다중 매개변수 병렬 컴퓨팅 응용(Embarrassingly parallel application)에 적합하다[5-11]. P2P 그리드 컴퓨팅은 SETI@Home [1], distributed.net [2]과 같은 프로젝트의 성공과 P2P 기반 분산 컴퓨팅의 비즈니스적 관심 [3, 4], 그리고 그리드 컴퓨팅의 출현 및 성장으로 [5] 인해 더욱더 관심을 받고 있다. 최근 Bayanihan [6], XtremWeb[7], Javelin [8], BOINC [10], Entropia [11], Condor [12], Korea@Home [20]과 같은 P2P 그리드 컴퓨팅에 대한 하부 플랫폼을 제공하는 P2P 그리드 컴퓨팅 시스템에 대한 연구가 활발히 진행되고 있다.

P2P 그리드 컴퓨팅 환경은 크게 클라이언트, 자원제공자, 자원제공자 서버로 구성된다. 클라이언트는 자원제공자 서버에게 작업을 요청하는 자이다. 자원제공자는 자신의 컴퓨팅 자원을 제공해주는 자발적 자원제공자이고, 자원제공자 서버는 인터넷으로 연결된 자원제공자들과 작업을 통제하는 관리자이다. P2P 그리드 컴퓨팅 환경에서의 연산 수행 과정은 크게 다음과 같다. 클라이언트는 자원제공자 서버에게 작업을 요구하면, 자원제공자 서버는 연산에 참여한 자원제공자들에게 작업을 분배한다. 자원제공자는 할당받은 작업에 대해 연산을 수행하고 그 결과를 자원제공자 서버에게 반환한다. 마지막으로 자원제공자 서버는 자원제공자로부터 받은 결과들을 정리하여 해당 클라이언트에게 작업 결과를 반환한다.

P2P 그리드 컴퓨팅 환경에서 자원제공자는 인터넷으로 연결된 데스크톱을 기반으로 하고 있기 때문에 자원제공자의 이질적인 속성, 고장, 휘발성, 비신뢰성들로 인해 안정적인 연산 수행을 보장할 수 없다 [3-12]. 특히, 자원제공자는 어떠한 보상 없이 자원을 제공하고 연산에 참여하기 때문에 아무런 제약없이 자유롭게 연산에 가입하거나 탈퇴할 수 있다. 결과적으로 자원제공자는 다양한 자원 제공 시간을 가지게 되며, 공공 연산(public execution: 자원제공자로써의 작업에 대한 연산)은 자유로운 탈퇴로 말미암아 임의적으로 중지되는 현상이 발생한다. 더욱이, 자원제공자는 공공 연산에 전용으로 사용되는 자원이 아니기 때문에 공공 연산은 개인 연산(private execution: 개인 사용자로써의 연산)에 의해 일시적으로 중지되는 현상이 발생한다. 본 논문에서는 이런 불안정한 상황을 *자원제공자 자율성 고장*이라고 정의한다. 이런 자원제공자 자율성 고장은 연산의 봉쇄 및 지연 현상을 초래하고, 심지어 부분적 또는 전체적으로 연산을 손실하는 결과를 초래한다. 자원제공자는 연산 수행 행태에 따라 다양한 자원제공자 자율성 고장 발생 비율과 종류를 가지고 있다. 또한 P2P 그리드 컴퓨팅 환경에서는 악의의 자원제공자가 연산 결과를 손상시키고 잘못된 결과를 반환할 수 있다. 이러한 독특한 특성들로 인해 자원제공자 서버는 작업을 스케줄링하고 할당된 작업과 자원제공자들을 관리하는데 어려움을 겪는다.

따라서 P2P 그리드 컴퓨팅 환경에서 스케줄링 기법은 연산의 신뢰성과 성능을 향상시키기 위해 자원제공자의 이질적인 속성과 자원제공자의 휘발성들과 같은 특성들에 적응해야만 한다. 다시 말해, 스케줄링 기법은 자원제공자들을 동질의 특성을 가진 그룹으로 구성한 후, 각 그룹의 특성에 맞게 다양한 스케줄링, 결함포용, 복제 알고리즘을 적용하는 게 필요하다. 그러나 기존 P2P 그리드 컴퓨팅 시스템들 [6-12]은 그룹 기반 스케줄링 기법을 다루지 않고 있다. 또한 스케줄링 기법이 자원제공자 서버에 의해 중앙 집중적으로 수행된다. 이로 인해 기존 스케줄링 기법들은 자원제공자 서버와 연산 수행에 대한 높은 오버헤드를 가지며 성능 저하 문제점을 가지고 있다.

이에 본 논문에서는 이러한 동적인 P2P 그리드 컴퓨팅 환경에 적응하기 위해 이동 에이전트 기반 적응적 그룹 스케줄링 기법(Mobile Agent based Adaptive Group Scheduling Mechanism: MAAGSM)을 제안한다. MAAGSM은 *자원제공자 자율성 고장, 자원제공자 가용성, 자원 제공 시간*과 같은 특성에 적응할 수 있도록 자원제공자들을 자원제공자 그룹으로 분류하고 구성한다. 또한 MAAGSM은 자원제공자 그룹에 다양한 스케줄링과 결함 포용 알고리즘을 적응적으로 적용하기 위해 이동 에이전트 기술을 사용한다. MAAGSM에서 이동 에이전트는 각각의 자원제공자 그룹에 할당된 후 자원제공자 서버의 직접적인 통제 없이 분산된 방법으로 스케줄링, 결함포용, 복제 알고리즘을 수행한다 (본 논문에서는 결함포용과 복제 알고리즘에 대해서는 다루지 않고 스케줄링 알고리즘에 대해서만 다룬다.). 결과적으로 MAAGSM은 오버헤드를 감소시키고 성능을 향상시키며 신뢰성 있는 연산 수행을 보장한다. 성능평가 결과에서는 MAAGSM이 기존 스케줄링 기법에 비해 더 많은 작업을 완료하고 오버헤드를 감소시키는 것을 보여준다.

2. 연구 배경 및 동기

2.1 관련 연구

AgentTeamwork[13]는 이동 에이전트 기반 PC Grid 미들웨어를 제안하였다. AgentTeamwork는 정지고장 발생시 작업 이주와 자원 검색을 위해 이동 에이전트를 사용하였다. 그러나 본 제안 기법에서는 이동 에이전트를 스케줄링을 목적으로 이용한다. SETI@Home[1]과 BOINC[10]에서는 중앙 서버가 스케줄링 및 자원제공자들의 관리를 수행함으로 인해 큰 오버헤드가 발생한다. 또한 자원제공자가 정기적으로 검사점을 취하고 사용자나 고장에 의해 연산이 중단되면 마지막으로 저장된 검사점부터 다시 수행된다. Charlte[9]는 작업을 끝내고 다른 작업을 요청한 자원제공자에게 먼저 작업을 할당하는 선종결우선할당 스케줄링을 제안하였다. XtremWeb[7]은 FIFO 스케줄링 기법을 제안하고 Bayanihan[6]은 악의의 자원제공자를 포용하기 위해 선종결우선할당 스케줄링(eager scheduling)을 기반으로 과반수투표(majority voting)과 발췌검사(spot-checking) 기법을 제안하였다. Javelin[8]은 트리에 기반한 선종결우선할당 스케줄링 기법을 제안하였다. CCOF[15]는 CAN을 이용하여 동일시간대를 기반으로한 구성한 오버레이 네트워크를 스케줄링에 이용한 웨이브 스케줄러(wave scheduler)를 제안하였다. Maheswaran[16]은 계산 그리드 환경에서 다양한 스케줄링 (MCT, MET, SA, KPB, Min-min, Max-min, Sufferage) 기법을

본 연구는 한국과학기술정보연구원(KISTI) 초고속응용기술지원사업 지원에 의해서 수행되었음

제안하였다. Kondo[14]는 자원제공자를 CPU, 위치, 네트워크 상태를 기반으로 보수(conservative) 와 급진(extreme)으로 나눈 다음 스케줄링을 적용하였다. Condor[12]는 작업할당을 위한 ClassAd를 제안하였고 우선순위에 기반한 스케줄링 기법을 제안하였다.

그러나 기존 연구에서는 자원제공자의 특성에 따라 그룹을 기반한 다양한 스케줄링, 결함 포용, 복제 알고리즘을 제안하지 못하고 있다. 또한 중앙서버에 의해 스케줄링과정이 수행되고 할당된 작업과 자원제공자들이 관리되기 때문에 높은 오버헤드를 발생하고 성능저하 문제가 발생한다.

2.2 이동 에이전트의 필요성

본 논문에서는 스케줄링 기법이 동적인 P2P 그리드 컴퓨팅 환경에 적응하도록 하기 위해 이동에이전트 기술을 이용한다. P2P 그리드 컴퓨팅 환경에서 이동에이전트를 사용함으로 인해 다음과 같은 이득을 얻을 수 있다.

1) 다양한 스케줄링 기법들이 자원제공자의 특성에 따라 수행될 수 있다. 예를 들어, 다양한 스케줄링, 결함포용, 복제 알고리즘들은 이동에이전트 (즉, 스케줄링 이동에이전트)로 구현된다. 자원제공자가 특성에 따라 동질의 그룹으로 구성된 후 해당 그룹에 가장 적합한 스케줄링 이동에이전트가 할당된다.

2) 이동에이전트는 분산된 방법으로 스케줄링, 결함포용, 복제 알고리즘을 수행함으로써 자원제공자 서버의 오버헤드를 감소시킬 수 있다. 이동에이전트는 서버와의 직접적인 접촉 없이도 비동기적으로 다양한 알고리즘을 수행할 수 있다.

3) 이동에이전트는 동적인 P2P 그리드 컴퓨팅 환경에 적응할 수 있다. P2P 그리드 컴퓨팅 환경에서 자원제공자는 다양하고 이질적인 특성을 가지고 있으며 연산 수행 중 자유롭게 탈퇴를 할 수 있다. 이동에이전트는 이주와 복제 기능을 이용하여 동적인 환경 변화에 대응하면서 자율적으로 연산을 수행할 수 있다.

3. 이동에이전트 기반 적응적 그룹 스케줄링 기법

이동에이전트 기반 적응적 그룹 스케줄링 기법(MAAGSM)은 자원제공자 그룹 구성하는 방법, 스케줄링 이동에이전트를 자원제공자그룹에 할당하는 방법, 그룹 구성원에게 작업이동에이전트를 할당하는 방법으로 크게 구성된다.

3.1 자원제공자 그룹 구성 방법

3.1.1 자원제공자의 분류

자원제공자를 분류할 때 CPU, 메모리 용량은 중요한 요소이다. 그러나 더 중요한 요소는 자원제공자의 위치, 자원제공자 자율성고장, 자원제공시간, 가용성등과 같은 특성들이다. 왜냐하면 P2P 그리드 컴퓨팅 환경에서 자원제공자의 용량은 거의 유사하나 자원제공시간, 자원제공자 자율성고장, 가용성은 매우 다양하기 때문이다[17, 20]. 다시 말해, 연산 수행 시간은 후자 특성에 크게 영향을 받는다. 본 논문에서는 자원제공자를 자원제공시간, 자원제공자 가용성에 의해 분류한다.

■ 정의 1 (자원제공시간) : 자원제공시간(Y)는 자원제공자가 자신의 자원을 제공하기로 한 시간이다.

$$Y = Y_R + Y_S$$

Y_R는 자원제공자가 자신의 컴퓨팅 자원을 제공하기로 한 예약 자원제공시간이다. 대부분의 공공연산은 Y_R에 수행된다. Y_S는 예상치 못한 자원제공시간으로 대부분 개인 연산이 수행되며 때때로 공공 연산이 수행된다.

■ 정의 2 (자원제공자 가용성) : 자원제공자 가용성(α_v)는 자원제공자가 정상적으로 동작하면서 Y동안에 자원 제공 서비스를 수행할 수 있는 확률이다.

$$\alpha_v = \frac{MTTVAF}{MTTVAF + MTTR}$$

$MTTVAF$는 자원제공자 자율성고장이 발생하기전까지의 평균 시간을 나타내고 $MTTR$은 자원제공자 자율성고장이 발생하는 평균시간을 나타낸다. 결과적으로 α_v는 자원제공자 자율성고장의 정도를 반영한다. 그러나 분산시스템에서 가용성은 주로 정지고장과 관련된다.

자원제공자는 위치에 따라 지역과 홈 자원제공자로 나뉘어 진다. 홈 자원제공자는 가정에서 자원을 제공하는 컴퓨터를 말하며, 지역 자원제공자는 대학이나 기관등과 같은 조직과 관련된다. 홈 자원제공자는 인터넷으로 연결되어 있고 지역 자원제공자는 LAN으로 연결되어 있다.

자원제공자를 Y과 α_v로 분류를 하면 그림 1 (a)와 같이 4개의 부류 (A, B, C, D)로 구분된다.

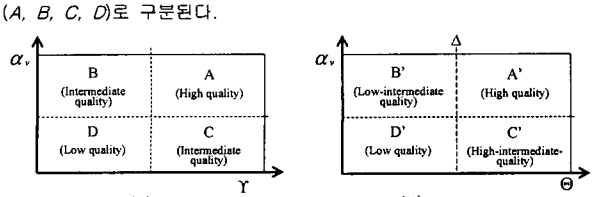

그림 1. 자원제공자와 자원제공자 그룹의 분류

3.1.2 자원제공자 그룹의 분류 및 구성

자원제공자 서버는 자원제공자 그룹 구성원을 자원제공자의 위치, 자원제공자 가용성, 자원제공 서비스시간에 따라 구성한다.

■ 정의 3 (자원제공 서비스시간) : 자원제공 서비스시간 (Θ)는 자원제공자가 자원제공시간에 공공연산에 참여하는 예상 서비스 시간이다.

$$\Theta = Y \times \alpha_v$$

Θ는 자원제공자가 자원제공자 자율성 고장이 발생하는 경우를 고려한 작업을 실제로 수행하는 시간을 나타낸다. 따라서 스케줄링 과정에서 Θ는 Y보다 더 적당하다. 자원제공자 그룹은 Θ과 α_v에 따라서 그림 1(b)와 같이 4개의 부류 (A', B', C', D')로 구성된다. 여기서 Δ는 작업에 대한 예상 수행시간을 나타낸다. 자원제공자 그룹 구성 알고리즘은 다음과 같다. 1) 등록된 자원제공자들은 자신의 위치에 따라 홈 자원 제공자나 지역 자원 제공자들로 구분된다. 2) 홈 자원제공자와 지역 자원제공자들은 α_v와 Y에 따라 각각 A, B, C, D 부류로 나뉘어 진다. 3) Θ와 Y에 따라 A', B', C', D' 자원 제공자 그룹이 형성된다.

```
//To classify the registered volunteers(V) into hoem or region volunteers
ClassifyVolunteersByLocation(V);
//To classify the home and region volunteers into A, B, C, D classes, respectively
ClassifyVolunteers(V);
//To construct volunteer groups
if (Vi.Θ ≥ Δ) then // Vi : one of the classified volunteers
    if (Vi ∈ VA || Vi ∈ VB) then // VA: A class, VB : B class
        Vi → VGA'; // → : assign, VBA' : A' volunteer group
    else // Vi ∈ VC || Vi ∈ VD, VC : C class, VD : D class
        Vi → VGC'; // VGC' : C' volunteer group
    fi;
else // Vi.Θ < Δ
    if (Vi ∈ VA || Vi ∈ VB) then // VA: A class, VB : B class
        Vi → VGB'; // → : assign, VBB' : B' volunteer group
    else
        Vi → VGD' ; // VGD' : D' volunteer group
    fi;
fi;
```

그림 2. 자원제공자 그룹 구성 알고리즘

3.2 스케줄링 이동에이전트를 자원제공자 그룹에 할당하는 방법

자원제공자 그룹을 구성한 후, 자원제공자 서버는 스케줄링 이동에이전트(S-MA)를 자원제공자그룹에 할당한다. 그러나 일부 자원제공자 그룹은 작업을 안정적으로 끝낼 수 없기 때문에 자원제공자그룹을 결합한 새로운 스케줄링 그룹 (A'D' & C'B', A'B' & C'D', A'C' & B'D')을 구성해야한다. 본 논문에서는 자원제공자 가용성과 자원제공시간을 서로 보완해주는 A'D' & C'B' 스케줄링 그룹을 사용한다.

S-MA는 작업위임자(deputy volunteer)에서 수행된다. 작업위임자는 그림 3과 같은 알고리즘에 의해 선택된다. 기본적으로 작업 위임자는 CPU, 메모리, 네트워크 대역폭, 하드디스크 용량 등의 기본적인 자원제공자 특징에 따라 선택된다. 그러나 본 논문에서는 이와 같은 기본적인 특성뿐만 아니라 자원 제공자의 동적인 특성인 자원 제공자 자율성 고장을 반영한 자원제공자 가용성과 자원제공 서비스 시간에 따라 작업 위임자를 선택한다.

```
// DVS : deputy volunteer set
// CDVS : candidate deputy volunteer set
// TDVS : temporal deputy volunteer set
// CDVS ⊆ VGA'
// TDVS = OrderedBy(CDVS.αv);
// HC : harddisk capacity, NB : network bandwidth
DVS = OrderedBY(TDVS.(Θ+HC+NB));
// Pop the best deputy volunteer from DVS
DV = PopBestDV(DVS);
```

그림 3. 작업위임자 선택 알고리즘

3.3 작업 이동에이전트를 그룹 구성원에게 할당하는 방법

S-MA를 스케줄링그룹에 할당한 후 각 S-MA는 작업 이동에이전트(T-MA)를 스케줄링그룹 구성원들에게 할당한다. 스케줄링 그룹의 종류에 따라 S-MA는 다른 스케줄링, 결함포용 알고리즘을 적용한다.

A'D' 스케줄링 그룹의 S-MA는 다음과 같은 스케줄링을 적용한다.

1) A' 자원제공자그룹을 α_v와 Θ에 따라 정렬한다.
2) T-MA를 정렬된 A'그룹 구성원에게 분배한다.
3) 만약 T-MA가 고장이 나면 해당 작업을 A' 그룹에서 선택된 다른 자원제공자에게 복사를 하거나, A'나 B' 그룹에서 선택된 다른 자원제공자에게 이주시킨다.

C'B' 스케줄링 그룹의 S-MA는 다음과 같은 스케줄링을 적용한다.

1) C'와 B' 자원제공자그룹을 α_v와 Θ에 따라 정렬한다.
2) T-MA를 정렬된 C'그룹 구성원에게 분배한다.
3) 만약 T-MA가 고장이 나면 해당 작업을 C' 그룹에서 선택된 다른 자원제공자에게 복사를 하거나, C'나 B' 그룹에서 선택된 다른 자원제공자에게 이주시킨다.

작업은 먼저 A'D' 할당된 후 C'B' 스케줄링 그룹에 할당된다. 작업은 높은 α_t와 긴 θ를 가진 자원제공자에게 먼저 할당된다. B'와 D' 그룹은 작업을 수행할 충분한 시간을 가지고 있지 않기 때문에 작업이 할당되지 않는다. 오직 구성원들의 특성을 재계산하기 위한 평가용으로 작업이 할당된다. 만약 이주가 허용된다면, B' 그룹은 A'와 C' 그룹이 고장이 난 경우 이주 창소로써 사용될 수 있다.

4. 성능평가

제안 기법인 MAAGSM을 기존의 선종결우선할당 스케줄링 기법 [6-9]과 비교 평가를 하기 위해 표 1과 같은 시뮬레이션 환경을 구성하였다. 본 시뮬레이션에서는 자원제공자 그룹이 구성된 후 스케줄링 과정이 그룹별로 적용되는지 그렇지 않은지에 따른 성능향상을 비교한다. 시뮬레이션 환경은 Korea@Home을 기반으로 구성되었다. 200개의 자원제공자가 다양한 자원제공자 자율성 고장, 자원제공자 가용성, 자원 제공 시간을 갖는다. *MTTVAF*는 1/0.2 ~ 1/0.02 분이고 *MTTR*은 3~10분으로 가정한다.

표 1. 시뮬레이션 환경

Case		A'	B'	C'	D'	Total
Case1	# of vol.	127(63%)	30(15%)	35(17%)	9(5%)	200
	α_t	0.95	0.95	0.74	0.77	0.91
	θ	43	15	31	11	35min.
Case2	# of vol.	95(47%)	26(13%)	63(32%)	16(8%)	200
	α_t	0.9	0.9	0.65	0.65	0.80
	θ	40	14	28	9	30min.
Case3	# of vol.	78(39%)	75(37%)	16(8%)	31(16%)	200
	α_t	0.95	0.95	0.70	0.61	0.88
	θ	31	11	25	8	20min.
Case4	# of vol.	52(26%)	48(24%)	23(12%)	77(38%)	200
	α_t	0.85	0.85	0.56	0.54	0.70
	θ	28	9	22	7	15min.

그림 4는 완료된 평균 작업 개수를 나타낸다. ES는 선종결우선할당 스케줄링을 나타내고 AS는 MAAGSM을 나타낸다. AS(A'D')와 AS(C'B')는 MAAGSM에서 각각 스케줄링 그룹을 나타낸다. 그림 4에 나타난 것처럼 MAAGSM은 기존 선종결우선할당 스케줄링 기법보다 더 많은 작업을 완료한다. 자원제공자그룹이 성능에 미치는 영향은 다음과 같다. 첫째, A' 자원제공자그룹은 성능향상에 많은 영향을 미친다. 예를 들어, A' 그룹의 구성원 수가 점차적으로 감소할 때 (Case 1부터 Case 4를 살펴보면), 완료된 작업 개수는 점점 감소함을 알 수 있다. 둘째, A'와 C' 자원제공자 그룹은 B'와 D' 자원제공자 그룹보다 더 중요한 역할을 수행한다. 예를 들어, Case 1과 2는 Case 3와 4보다 더 많은 작업을 완료한다. 셋째, 자원제공자 가용성은 성능향상과 밀접한 관련이 있다. 예를 들어, 높은 자원제공자 가용성을 지닌 Case1은 다른 것들에 비해 더 많은 작업을 수행한다. 마지막으로, A' 그룹 구성원의 수는 점점 감소하고 B'와 D' 그룹의 수는 점점 증가할 때 MAAGSM과 선종결우선할당기법과의 작업 개수 차이는 점점 증가한다. 선종결우선할당 기법에서는 A', B', C', D' 그룹에서 고장나거나 중단된 작업은 자신보다 질이 낮은 자원제공자에게 분배될 수 있다. 그러나 MAAGSM에서는 자원제공자 그룹을 기반으로 스케줄링과정이 일어나기 때문에 이러한 현상은 발생하지 않는다. 따라서 Case 1에서는 A' 그룹 구성원의 수는 많고 B'와 D' 그룹 구성원의 수가 작기 때문에 Case1에서의 작업 개수의 차이는 다른 것들에 비해 작음을 알 수 있다.

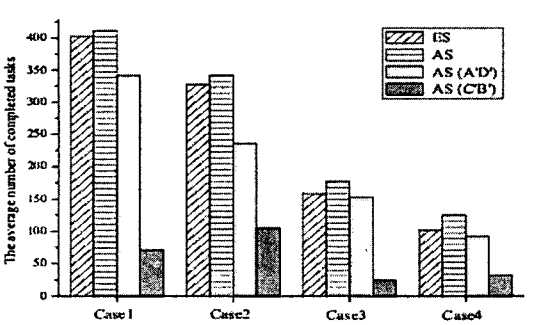

그림 4. 완료된 평균 작업 개수

5. 결론 및 향후 연구과제

본 논문에서는 동적인 P2P 그리드 컴퓨팅 환경에 적응하는 이동에이전트 기반 적응적 그룹 스케줄링기법 (MAAGSM)을 제안하였다. MAAGSM은 자원제공자의 특성, 특히, 자원제공자 자율성 고장, 자원제공자 가용성, 자원제공서비스 시간에 따라 동질의 그룹을 만들고 각 그룹별로 스케줄링 이동에이전트를 이용하여 다양한 스케줄링, 결함 포용, 복제 알고리즘을 적용한다. 결과적으로 MAAGSM은 기존의 스케줄링기법보다 더 많은 작업을 완료하고, 분산된 방법으로 이동에이전트를 사용함으로써 자원제공자 서버의 오버헤드를 줄인다.

자원제공자 그룹을 구성할 때 자원제공 서비스 시간, 자원제공자 가용성뿐만 아니라 악의의 자원제공자에 의해 계산된 손상된 결과를 검출하고 포용하는 정확성검사기법과 관련된 자원제공자 신용도(Credibility)에 따라 그룹을 구성하는 방법과 각 그룹에 적용할 스케줄링, 결함 포용, 복제 알고리즘을 연구하고 있다.

참고문헌

[1]SETI@home, "http://setiathome.ssl.berkeley.edu"
[2]Distributed.net, "http://distributed.net"
[3]D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing", HP Laboratories Palo Alto HPL-2002-57, March 2002.
[4]I. Foster and A. Iamnitchi, "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing," 2nd Int. Workshop on Peer-to-Peer Systems, LNCS 2735, pp. 118-128, Feb. 2003.
[5]F. Berman, G. C. Fox, and A. J. G. Hey, Grid Computing : Making the Global Infrastructure a Reality, Wiley, 2003
[6]L. F. G. Sarmenta and S. Hirano. "Bayanihan: Building and Studying Volunteer Computing Systems Using Java," Future Generation Computer Systems, Vol. 15, Issue 5/6, pp.675-686, Oct. 1999.
[7]O. Lodygensky, G. Fedak, F. Cappello, V. Neri, M. Livny, D. Thain, "XtremWeb & Condor : sharing resources between Internet connected Condor pool," 3rd IEEE/ACM Int. Symposium on Cluster Computing and the Grid: Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems, pp. 382-389, May 2003.
[8]M. O. Neary and P. Cappello, "Advanced eager scheduling for Java-based adaptive parallel computing," Concurrency and Computation: Practice and Experience, Vol. 17, Issue 7-8, pp.797-819, Jul. 2005.
[9]A. Baratloo, M. Karaul, Z. Kedem, and P. Wijckoff, "Charlotte: Metacomputing on the Web," Future Generation Computer Systems, Vol. 15, Issue 5-6, pp.559-570, Oct. 1999.
[10]D. P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," 5th IEEE/ACM Int. Workshop on Grid Computing, pp. 4-10, Nov. 2004.
[11]A. Chien, B. Calder, S. Elbert, and K. Bhatia, "Entropia: architecture and performance of an enterprise desktop grid system," Journal of Parallel and Distributed Computing, Vol. 63, Issue 5, pp. 597-610, 2003.
[12]D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice : The Condor Experience," Concurrency and Computation: Practice and Experience, Vol. 17, Issue 2-4, pp. 323-356, Feb. 2005.
[13]M. Fukuda, Y. Tanaka, N. Suzuki, and L. F. Bic, "A Mobile-Agent-Based PC Grid," 5th Int. Workshop on Active Middleware Services, pp. 142-150, Jun. 2003.
[14]D. Kondo, H. Casanova, E. Wing, and F. Berman, "Models and scheduling mechanisms for global computing applications," 16th Int. Parallel and Distributed Processing Symposium, pp.79-86, Apr. 2002.
[15]V. Lo, D. Zhou, D. Zappala, Y. Liu, and S. Zhao, "Cluster Computing on the Fly: P2P Scheduling of Idle Cycles in the Internet," 3rd Int. Workshop on Peer-to-Peer Systems, LNCS 3279, pp. 227-236, 2004.
[16]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," 8th Heterogeneous Computing Workshop, pp. 30-44, Apr. 1999.
[17]D. Kondo, M. Taufer, J. Karanicolas, C. L. Brooks, H. Casanova and A. Chien, "Characterizing and Evaluating Desktop Grids: An Empirical Study," 18th Int. Parallel and Distributed Processing Symposium, pp. 26-35, Apr. 2004.
[18]P. Jalote, Fault Tolerance in Distributed Systems, Prentice-Hall, 1994
[19]A. S. Tanenbaum and M. V. Steen, Distributed Systems: Principles and Paradigms, Prentice Hall, 2002.
[20]Korea@Home, http://www.koreaathome.org/eng/