자원제공자 컴퓨팅 환경에서 공헌도 기반 차별화 스케줄링 기법*

변은정°, 최성진†, 윤준원†, 유헌창‡, 최장원□□, 황종선†
고려대학교 컴퓨터학과 분산시스템 연구실†, 고려대학교 컴퓨터교육과‡, 한국과학기술정보연구원□□
{vision°, lotieye†, zebra†, hwang†}@disys.korea.ac.kr, yuhc@comedu.korea.ac.kr‡, jwchoi@kisti.re.kr□□

Differential Scheduling Mechanism based on Contribution Rate in Volunteer Computing Environment

Eun-Joung Byun°, Sung-Jin Choi†, Joon-Won Yoon†, Chong-Sun Hwang†
Dept. of Computer Science & Engineering, Korea University
Hyun-Chang Yoo‡
Dept. of Computer Science & Education, Korea University
Jang-Won Choi□□
Supercomputing center, KISTI

요 약

자원제공자 컴퓨팅(Volunteer Computing) 환경에서 자원제공자의 연산에 대한 참여(join) 및 이탈(leave), 결함 등의 동적인 연산 수행 특성을 고려한 스케줄링 기법은 안정적인 연산 수행을 지원하는 시스템의 설계에 있어서 중요한 고려사항이다. 기존의 자원제공자 컴퓨팅 연구에서는 자원제공자의 자유로운 연산 참여와 이탈 및 연산 수행 중 일시적인 연산 중단 등의 휘발성(volatility)을 고려한 결함 포용적 스케줄링 기법이 없어 연산 수행 중에 빈번한 중단이 발생하고, 연산 수행 시간이 지연된다. 본 논문에서는 이러한 자원제공자의 휘발성으로 인해 발생되는 문제를 해결하기 위해 자원제공자 컴퓨팅 환경에 부합하는 다양한 가용성(availability)을 정의, 분류하고, 공헌도(Contribution Rate) 기반 차별화 스케줄링 기법을 제안한다. 공헌도 기반 차별화 스케줄링 기법은 헌신도(Dedication Rate)와 능력도(Faculty Rate)를 이용하여 주어진 작업량에 적합한 연산 수행 능력을 가지는 자원제공자를 선출하여 작업을 할당하고, 결함이 발생할 경우에는 결함의 종류에 따라 유연하게 대처하여 불안정한 자원 제공과 연산 중단 현상을 완화시킨다. 이로써, 연산 수행에 대한 완료성과 신뢰성을 향상시켜 연산 지연 및 전체 연산 수행 시간을 단축시킨다.

1. 서 론

자원제공자 컴퓨팅[1]은 인터넷에 연결된 수많은 유휴 컴퓨팅 자원들의 자발적 참여를 통해 대량의 연산을 병렬 처리하는 방식이다[2]. 자원제공자 컴퓨팅은 PC의 계산 능력의 향상과 인터넷의 발전을 바탕으로[3], 분산된 대규모 유휴 컴퓨팅 자원을 활용하여 저비용으로 고성능 대용량 계산 능력을 창출한다[4].

자원제공자 컴퓨팅에서 인터넷으로 연결된 자원제공자는 연산 수행 중 자원제공자의 자율적인 연산 참여와 이탈 및 사용자의 간섭에 의해 일시적, 장기적으로 중단되는[1] 등의 휘발성을 가진다[2]. 이로 인해 지속적이고, 안정적인 연산 수행을 보장하지 못하여 연산의 신뢰성이 저하되고, 연산 시간의 지연과 전체 연산 수행 시간의 증가로 시스템의 성능이 저하된다.

기존 관련 연구들에는 휘발성과 같은 자원제공자 컴퓨팅의 다양한 특성을 반영하고, 자원제공자의 동적 환경 변화에 대처하는 스케줄링 기법이 없으므로[2], 안정적인 연산 수행을 위해 보다 적응적[3], 결함 포용적 스케줄링 기법이 요구된다[4].

본 논문에서는 자원의 휘발성으로 인해 발생되는 문제를 해결하기 위해 자원제공자 컴퓨팅에서의 가용성을 새롭게 정의, 분류하고, 자원제공자의 헌신도와 능력도를 반영한 공헌도 기반 차별화 스케줄링 기법을 제안한다. 헌신도는 자원제공자의 연산 참여 및 수행의 규칙성과 중단 발생 빈도 등을 반영해 연산 수행 패턴을 예측하고, 능력도는 연산 수행 지속력을 나타낸다. 자원제공자의 공헌도를 고려한 스케줄링은 연산 지연시간을 감소시켜 연산 수행시간을 단축시키고, 결함의 종류와 지연 시간에 따른 유연한 결함 포용으로 연산 신뢰성을 향상시킨다.

* 본 연구는 한국과학기술정보연구원(KISTI) 초고속응용기술지원사업 지원(G-04-GS-I01)에 의해서 수행되었음

2. 관련 연구

기존의 자원제공자 컴퓨팅 연구들의 스케줄링 기법을 살펴보면 다음과 같다. 첫 번째, 스케줄링 기법을 사용하지 않는 경우[4], 두 번째, 단순한 선행처리자 우선할당(eager scheduling) 기법을 사용하는 경우[1], 세 번째, 진보된 선행처리자 우선할당 기법을 사용하는 경우[3]이다.

첫 번째, 별도의 스케줄링 기법이 없는 Korea@Home[4]은 연산 수행 모델이 비효율적이며, 결함 포용을 위한 방법도 없다.

두 번째, 우선적으로 작업을 끝낸 우위의 자원제공자에게 느린 자원제공자의 작업을 재할당하는 선행처리자 우선할당 기법[1]을 사용하는 Bayanihan[1], Charlotte[2]는 자원제공자의 이탈이나 결함 발생이 발생할 경우, 다른 자원제공자에게 작업을 단순히 재할당하는 방식을 사용하여 안정적인 연산 수행을 보장하지 못하며 전체 연산 수행 시간 지연 등을 야기한다.

세 번째, Javelin++[3]은 작업의 부하균형과 동적 작업분배를 위해 진보된 형태의 선행처리자 우선할당 기법을 사용하였으나, 자원제공자의 휘발성을 고려한 연산 수행 모델이 없다.

이처럼 대부분의 기존 연구들이 가지는 문제점은 불안정한 자원 제공 및 일시적인 연산 간섭이나 네트워크 단절 등과 같은 자원제공자 컴퓨팅의 동적인 특성을 체계적으로 고려한 스케줄링 및 결함 포용 기법이 부족하다는 것이다. 이로 인해 연산의 지속성과 완료성을 보장하지 못하고, 시스템 성능이 감소된다.

최근 자원제공자 컴퓨팅과 유사하게 대부분의 자원제공자가 PC인 데스크탑 GRID 분야[5]나 대규모 P2P 시스템[6]에서는 각 자원의 빈번한 참여와 이탈로 인한 자원의 휘발성을 해결하기 위해 가용성을 중요한 이슈로 다루고 있는 추세이나, 아직까지 체계적인 가용성의 정의와 분류 및 가용성을 이용한 스케

줄링 기법 등이 부족한 실정이다.

3. 자원제공자 컴퓨팅 환경

본 논문에서 가정하는 자원제공자 컴퓨팅 환경은 자원제공자 VN(Volunteer Node), 중앙관리서버 CS(Central Server), 작업위탁자 WC(Work Commissioner)등의 기본적인 요소[1]와, 중앙서버의 부하 경감을 위한 그룹관리자 GM(Group Manager)[7]과 자원제공자의 정보를 유지하는 DB로 이루어진다[4]. 작업위탁자가 연산 수행을 위해 작업을 중앙관리서버에게 위탁하면, 중앙관리서버는 작업을 작업 간에 의존성이 없는 작업단위(WorkUnit)로 분할하고[1], 스케줄링 기법을 이용하여 작업을 수행할 자원제공자를 선정한 뒤, 작업을 할당한다. 자원제공자는 할당받은 연산을 수행하고 연산의 결과를 중앙서버에 반환한다. **그림 1**은 본 논문이 가정하는 환경을 나타낸다.

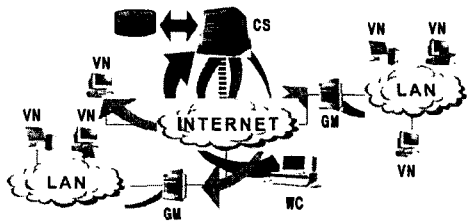

그림 1 . 자원제공자 컴퓨팅 환경

4. 공헌도

자원제공자의 연산 중단 및 이탈 등의 휘발성으로 인한 연산 시간의 지연과 연산의 미완료를 완화시키기 위해 자원제공자의 연산 참여의 질을 반영한 연산수행 지속력이 고려되어야 한다.

4.1 가용성(Availability)

자원제공자 컴퓨팅에서의 가용성은 다음과 같이 정의된다.

정의1. ***가용성****은 자원제공자의 자원 사용에 대한 요청을 수락하고 자원을 사용할 수 있는 상태의 양이다.* □

자원제공자 컴퓨팅에서 가용성은 시간적 가용성(Temporal Availability)과 공간적 가용성(Spatial Availability)으로 구분된다. 시간적 가용성은 호스트 가용성(Host Availability)과 연산 가용성(Execution Availability)으로 구성되고, 공간적 가용성은 처리 가용성(Throughput Availability)으로 구성된다.

4.1.1 시간적 가용성(Temporal Availability)

정의2. ***시간적 가용성****은 자원 사용의 시간양으로 다양한 연속적 시간의 함수로 표현되는 가용성이다.* □

시간적 가용성은 **그림 2**와 같이 시간적 관점에서 볼 때, 자원제공자의 연산 참여 시간양인 호스트 가용성[8]과 실제 연산 수행 시간양인 연산 가용성으로 구분한다.

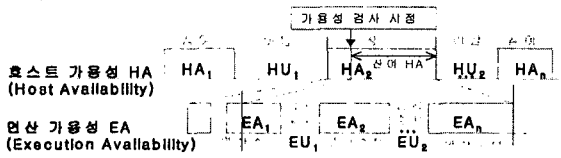

그림 2 . 시간적 가용성의 호스트 가용성과 연산 가용성

그림 2에서 HA_i는 i번째 호스트 가용성이고, EA_i는 i번째 연산 가용성이고, $HA=\sum EA+\sum EU$ 를 만족한다.

4.1.2 공간적 가용성(Spatial Availability)

정의3. ***공간적 가용성****은 자원의 물리적 용량(physical capacity) 이나 공간적 능력 자체 또는, 물리적 용량과 공간적 능력을 측정하는 단위로 표현되는 가용성이다.* □

기존 연구[8]에서 고려된 CPU 가용성과 메모리양, 대역폭 등이 본 논문의 공간적 가용성이다. 본 논문은 계산 집약적 연산을 수행하여, 높은 처리율을 요구하는 자원제공자 컴퓨팅의 목적을 고려하여 공간적 가용성으로 CPU 가용성 등을 직접적으로 사용하지 않고, 자원제공자가 단위시간당 처리할 수 있는 작업량인 처리가용성TA(Throughput Availability)을 이용한다.

4.2 헌신도(Dedication Rate)

정의4. ***헌신도DR****은 자원제공자 컴퓨팅 시스템에 대한 자원제공자의 시간적 가용성인 호스트 가용성의 비율과 연산 가용성의 비율의 규칙성을 반영한 연산 수행 집중률로 아래의 식으로 구할 수 있는 수치이다.*

$$DR=\frac{(HAR\cdot\alpha+EAR\cdot\beta)}{TSNR}$$ □

HAR(Host Availability Rate)은 시스템에 대한 자원제공자의 호스트 가용성의 비율이며, 자원제공자의 연산 참여 시간의 비율의 엔트로피로 구한다.

$$HAR=\log\left(\frac{HA_i}{HA_{VNmax}}\right)\cdot(E_{SYSmax}-E_{VN})$$

EAR(Execution Availability Rate)은 시스템에 대한 자원제공자의 연산 가용성의 비율이며, 자원제공자의 연산 수행 시간의 비율의 엔트로피로 구한다.

$$EAR=\log\left(\frac{EA_i}{EA_{VNmax}}\right)\cdot(E_{SYSmax}-E_{VN})$$

TSNR(Total Stop Number Rate)은 시스템에 대한 자원제공자의 연산수행 중 연산 중단 비율로 총 중단횟수 TSN에 대한 엔트로피로 구한다. $TSN_i=EU_i$이다.

$$TSN=\log\left(\frac{TSN_i}{TSN_{VNmax}}\right)\cdot(E_{SYSmax}-E_{VN})$$

E_{SYSmax}은 자원제공자 컴퓨팅 시스템의 최대 엔트로피로서 $E_{SYSmax}=[-\sum p_i\log p_i]_{max}$ 이고, E_{VN}은 자원제공자의 엔트로피로서 $E_{VN}=-\sum_{i=1}^{n} p_i\log p_i$ 이다. α, β는 HAR, EAR에 대한 가중치 값으로 $4\alpha=\beta$, $\alpha+\beta=1$를 만족한다.

시간적 가용성에 강하게 결합되어 있는 헌신도는 자원제공자의 연산참여 시간양의 비율과 실제 연산수행 시간양의 비율이 얼마만큼 규칙적이고, 예측이 가능한지를 나타내는 엔트로피를 이용하여 연산 수행의 질을 측정한다. 헌신도를 통해 연산 수행의 규칙성과 연산 수행시간의 예측성, 편차 등을 알 수 있다.

4.3 능력도(Faculty Rate)

정의5. ***능력도FR****은 공간적 가용성의 처리 가용성에 잔여 호스트 가용성을 반영한 자원제공자의 잔여 작업 처리력으로 아래의 식으로 구할 수 있는 수치이다.*

$$FR \overset{def}{=} TA \cdot RHA$$ □

능력도는 공간적 가용성에 시간적 가용성을 강하게 결합시킨 가용성으로 공간적 가용성의 처리 가용성 TA가 얼마만큼의 잔여 호스트 가용성 RHA(Residual Host Availability)의 시간만큼 연산을 지속할 수 있는지를 나타낸다.

4.4 공헌도(Contribution Rate)

정의6. ***공헌도CR****은 헌신도와 능력도를 반영한 자원제공자의 질적 연산 수행 지속력을 나타내는 값으로 아래의 식으로 구할 수 있는 수치이다.*

$$CR \overset{def}{=} DR \cdot FR$$ □

공헌도는 자원제공자의 시간적 가용성인 헌신도와 공간적 가용성인 능력도를 고려하여 자원제공자가 얼마만큼의 처리력을 가지고, 안정적인 연산 수행을 지속할 수 있는지를 나타낸다.

공헌도를 고려하여 스케줄링 함으로서, 처리할 작업에 알맞은 작업지속력을 갖는 자원제공자에게 작업을 할당하여 자원제공

자의 효용성을 높이고, 작업의 완료성 및 성능을 향상시킨다.

5. 공헌도 기반 차별화 스케줄링 기법

공헌도 기반 차별화 스케줄링 기법은 자원제공자의 공헌도에 따라 단위작업량을 차별화하여 할당하는 방식이다. 작업이 위탁되면 작업을 작업 간에 의존성이 없는 작업단위(WorkUnit)인 단위작업량(WorkLoad)으로 분할하고, 자원제공자의 프로파일을 이용해 공헌도를 구하여 공헌도를 기반으로 단위작업량에 알맞은 자원제공자를 선정하여 작업을 할당한다. 작업의 완료성 향상을 위해 작업 수행 중 결함이 발생하면, 결함의 종류와 지연시간의 추이에 따라 유연하게 대처하여 결함을 포용한다.

공헌도 기반 차별화 스케줄링 기법은 위탁받은 작업을 작업단위로 분할하고, 호스트 가용성이 유효한 자원제공자들을 헌신도가 높은 순으로 정렬한다. 헌신도를 이용해 낮은 연산 중지율과 높은 연산 가용성을 가지는 자원제공자에게 우선적으로 작업을 할당하므로 연산 수행 패턴 예측의 정확도를 높일 수 있다. 능력도를 이용한 자원제공자의 연산 지속 수행력의 예측을 통해 단위작업량에 알맞은 연산 지속력을 가지는 자원제공자를 선정하여 작업을 할당하므로 자원 사용의 효율성 역시 향상시킨다.

공헌도 기반 차별화 스케줄링 기법은 안정적 연산 수행을 향상시키고, 결함을 방지, 포용하기 위해 동일한 작업을 여러 자원제공자에게 복제하여 할당하고, 결과값들은 정확성 검사에 이용한다. 연산 중단이 발생하면, 연산 수행 도중 일어날 수 있는 다양한 중단의 원인과 중단 시간의 추이를 고려하여 유연하게 대처한다. 단위작업량의 크기가 큰 경우에는 체크포인팅 기법을 이용하여 결함이 발생 시에 다른 자원제공자로 이주하여 실행할 수 있도록 한다. 자원제공자가 고장 정지나 이탈 결함이 발생할 경우에는 공헌도를 이용하여 알맞은 자원제공자를 재선정하여 작업을 재할당하고, 연산 수행 중단으로 인해 지연시간이 길어질 경우에는 현재 자원제공자의 지연시간과 시스템의 지연시간을 비교하여 작업의 지속 여부와 재할당 여부를 결정한다. **그림 3**은 공헌도기반 차별화 스케줄링 알고리즘이다.

```
CalculateCR(CalculateDR(VN), CalculateFR(VN));
ListOfVN = OrderVN(DR);
if (SameOrder) then ReOrderVN(ListOfVN, FR); fi;
if(Work) then
{   while(Work == WorkLoad)
        SplitIntoWorkUnit(Work);
}elseif (WorkLoad) then CRbasedSchedule(WorkLoad); fi;
CRbasedSchedule(WorkLoad)
{   CR_target= getCR(FindFittestVN(ListOfVN, WorkLoad));
    if (CR_target ≒ WorkLoad) then
    SelectedVN = ChooseVN(ListOfVN, CR_target);
    WorkAllocation(SelectedVN, WorkLoad);
    elseif (CR_target ≫ WorkLoad) then
      WorkLoad WorkLoadList[];
      SelectedVN=ChooseVN(ListOfVN, (DR*WorkLoad));
      NumberOfWorkLoad = CR_target/WorkLoad;
          for(j=0 ; j=NumberOfWorkLoad-1 ; j++)
          {   WorkLoadList[j] = WorkLoad; }
      WorkAllocation(SelectedVN, WorkLoadList);
    fi; }
if (Check(State == CRASH)) then
if (Check(State == LEAVE)) then
CRbasedSchedule(WorkLoad);
elseif (Check(State == STOP)) then
        while (DelayTime_VN < DelayTime_SYSavg)
            Wait();
CRbasedSchedule(WorkLoad);
fi;
```

그림 3. 공헌도 기반 차별화 스케줄링 알고리즘

6. 모의 실험 및 평가

공헌도 기반 차별화 스케줄링 기법은 연산 지속 시간을 향상시켜 작업의 완료성을 향상시키며, 지연 시간을 감소시켜 전체 수행 시간을 단축시킨다. 두 스케줄링 기법의 작업 완료성을 평가하기 위해, 연산 수행 성공확률을 비교하면, 공헌도 기반 차별화 스케줄링 기법이 보다 높은 연산수행 성공 확률을 가진다. 호스트 가용성의 분포가 초지수(hyperexponential) 분포를 따른다고 가정할 때[5], 두 스케줄링 기법의 연산 수행 성공 확률이다.

- 공헌도 기반 차별화 스케줄링 기법의 연산 수행 성공 확률

$$P(S_{contribution}) = \sum_{i=1}^{n} P_i \lambda_i e^{-\lambda \Delta}$$

- 선행처리자 우선할당 기법의 연산 수행 성공 확률

$$P(S_{eager}) = \sum_{i=1}^{n} P_i \lambda_i e^{-\lambda(\Delta + (1-DR)\Delta)}$$

λ는 실패율로 연산 중단 발생률이고, Δ는 실제 연산을 수행하는데 필요한 시간으로 자원제공자 컴퓨팅 시스템에서 Δ는 헌신도가 반영된 시간이다. **그림 4**는 연산 중단 발생률이 증가함에 따른 두 기법의 연산수행 성공확률의 변화를 비교한 그래프이다.

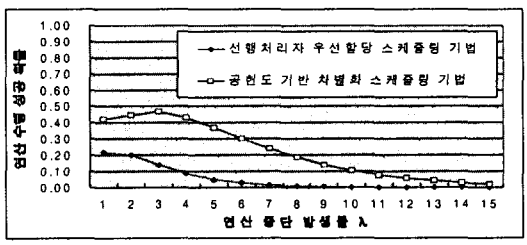

그림 4. 연산중단 발생율 증가에 따른 연산수행 성공확률

그림 4와 같이 연산 중단 발생에 대해 고려하지 않은 선행처리자 우선할당 기법은 연산 중단 발생률 λ가 커질수록 연산수행 성공 확률이 공헌도 기반 차별화 스케줄링 기법의 연산수행 성공 확률에 비해 현저히 감소되고 있다. 연산수행 성공 확률이 낮아지므로 연산의 완료성이 감소되고, 연산의 재할당으로 인한 전체 수행 시간의 증가와 함께, 시스템 성능이 저하된다.

7. 결론 및 향후 연구 과제

본 논문은 자원제공자 컴퓨팅 환경에서 자원제공자의 연산 수행 특성을 고려한 헌신도와 잔여 연산 수행 능력을 고려한 능력도를 이용한 차별화 스케줄링 기법, 연산 중단의 종류와 연산 지연 시간의 추이에 따라 유연하게 대처하는 결함 포용적 연산 결정 기법을 제안하였다. 불안정한 자원제공과 연산 중단 현상을 완화하여, 연산 수행에 대한 완료성과 신뢰성을 향상시키고, 결과적으로 전체 연산 수행 시간을 단축시킨다.

향후, 공헌도 기반 스케줄링 기법을 Korea@Home 프로젝트에 실제로 적용하여 구현하고, 자원제공자 컴퓨팅의 다양한 특성을 복합적으로 고려한 스케줄링 기법을 연구하고자 한다.

참고 문헌

[1] Luis F. G. Sarmenta, Volunteer Computing, Ph.D. thesis. Dept. of Electrical Engineering and Compute. Science, MIT, 2001.
[2] A.Baratloo, M.Karaul, Z.Kedem, and P.Wyckoff. "Charlotte : Metacomputing on the web", PDCS, 1996.
[3] Michael O. Neary, Sean P. Brydon, Paul Kmiec, Sami Rollins, Peter Cappello, "Javelin++:scalability issues in global computing", Concurrency : Practice and Experience Volume 12, Issue 8, pp. 727-753, 2000.
[4]"Korea@Home", http://www.KOREAatHOME.org
[5] John Brevik, Daniel Nurmi, Rich Wolski, "Modeling machine availability in enterprise and wide-area distributed computing environment", TR, CS2003-28, 2003.
[6]Bhagwan, Savage, and Voelker, "Understanding availability", IPTPS 2003.
[7] 김홍수, 강인성, 최성진, 황일선, 황종선, 유헌창 "인터넷 기반 병렬 컴퓨팅에서 중간 관리자 구성과 결함포용 기법", 한국정보과학회 가을 학술발표논문집(3), 제 30권 2호, pp. 643-645, 2003.
[8] Derrick Kondo, Michela Taufer, John Karanicolas, Charles L. Brooks, Henri Casanova and Andrew Chien, "Characterizing and Evaluating desktop Grids - An Empirical Study", IPDPS 2004.