분산형 P2P 그리드 시스템에서 자가 조직적 계산 오버레이 네트워크 기반 결함 포용적 스케줄링 기법*

김석인[O†], 박찬열[‡], 최장원[‡], 김홍수[†], 길준민[¶], 황종선[†]
고려대학교 컴퓨터학과[†], 한국과학기술정보연구원[‡], 대구 가톨릭대학교 컴퓨터교육과[¶]
{sikim[O], hera, hwang}@disys.korea.ac.kr[†], {chan, jwchoi}@kisti.re.kr[‡], jmgil@cu.ac.kr[¶]

Fault-Tolerant Scheduling Mechanism based on Self-organizing Computation Overlay Network in Decentralized P2P Grid System

SeokIn Kim[O†], ChanYeol Park[‡], JangWon Choi[‡], HongSoo Kim[†], JoonMin Gil[¶] and 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

분산형 P2P 그리드 시스템을 구축하는데 있어 연산 수행을 위한 노드 구성 기법과 구성된 토폴로지에 적합한 연산 수행 모델 및 스케줄링 기법은 필수 요소이다. 하지만 기존 연구에서는 자원 제공자의 휘발성을 고려하지 않은 연산 수행 모델을 사용하였기 때문에 연산의 안정적인 수행이 보장되지 못하고, 시스템의 성능이 떨어지는 문제점이 발생한다. 이에 본 논문에서는 가용성 기반의 자가 조직적 계산 오버레이 네트워크(SelfCON:Self-organizing Computation Overlay Network) 구성 기법과 구성된 토폴로지에 적합한 연산 수행 모델 및 스케줄링 기법을 제안한다. 제안 기법은 자원 제공자 노드의 휘발성을 고려하여 안정성을 높임으로써 전체 연산 성능을 향상시킨다.

1. 서론

P2P 그리드 컴퓨팅은 데스크톱 컴퓨터의 유휴 계산 자원을 이용하여 고성능의 연산을 수행하는 것을 목적으로 한다[1,2,4,5,6,7,8,9]. 그리고 작업 분배 모델에 따라 크게 중앙집중형 방식과 분산형 방식으로 나뉜다. 첫째, 중앙집중형 P2P 그리드 시스템은 중앙 서버가 클라이언트에게 작업을 받아 작업자에게 분배해주는 방식을 사용한다. 이 방식에서는 작업 분배, 작업 수집, 스케줄링이 모두 중앙 서버에서 이루어진다[7,8,9]. 따라서 확장성 저하 문제와 병목 현상 문제가 발생한다. 둘째, 분산형 P2P 그리드 시스템에서는 작업 분배 및 수집을 수행하는 어떠한 중앙 노드도 존재하지 않는다. 분산형 방식에서 작업 분배 및 수집은 각각의 구성원 노드들이 가지고 있는 부분적 정보를 통해 이루어진다[1,2,4,5]. 분산형 환경은 기존의 중앙집중형 환경의 확장성 저하 문제와 병목 현상 문제를 해결할 수 있는 하나의 대안으로 연구되고 있다.

분산형 P2P 그리드 시스템을 구축하는데 있어 연산 수행을 위한 노드 구성 기법과 구성된 토폴로지에 적합한 연산 수행 모델 및 스케줄링 기법은 안정적 연산 수행을 위해 꼭 필요한 요소이다. 그러나 기존 연산 수행 모델에서는 자원 제공자가 연산 중간에 탈퇴하는 성질인 휘발성을 고려하지 않았기 때문에 연산 수행 중 노드의 휘발성으로 인해 연산 수행이 중단되고 실패하는 문제점이 발생한다. 이로 인해 연산 수행시간이 증가되어 시스템 성능이 떨어지는 문제점이 발생한다.

이러한 문제점을 해결하기 위해 본 논문에서는 가용성에 기반한 자가 조직적 계산 오버레이 네트워크 구성 기법과 결함 포용적 스케줄링 기법을 제안한다. 제안 기법에서는 자원 제공자들을 가용성에 따라 하나의 조정자 그룹과 다수의 자원 제공자 그룹으로 구성된 연산 수행 토폴로지를 조직화하고 구성된 토폴로지의 특징에 맞는 새로운 연산 수행 모델과 자원 제공자의 휘발성을 고려한 결함 포용적 스케줄링 기법을 제안한다. 제안 기법은 안정적인 연산 수행을 보장하고 자원 제공자들간의 작업에 대한 부하 균형을 제공함으로써 전체 연산 성능을 향상시킨다.

2. 관련 연구

Organic Grid[1]는 노드의 성능을 고려한 오버레이 네트워크 구성방식과 그 위에서 동작하는 스케줄링 알고리즘을 제안한다. *Messor*[5]는 분산형의 비구조적(unstructured)인 무작위 네트워크(random network)에서 이동 에이전트를 이용하여 전체 노드의 부하를 일정하게 유지하는 부하 균형(load balancing) 기법을 제안한다. *Paradropper* [2]는 작은 세계 네트워크(small world network)를 구성하여 새로운 작업 위탁이 들어오면 가장 부하가 작은 노드에 할당한다. *CCoF*[4]는 작업 자들을 가용성이 높은 시간대별로 구분하여 해당 시간에 가용성이 높은 노드가 작업을 수행한다. 기존 시스템에서는 노드의 휘발성을 고려하지 않고 연산 수행 네트워크를 구성하여 스케줄링을 수행함으로써 작업 수행중 자원 제공자의 휘발성으로 연산이 중단되고 실패하는 현상이 발생한다.

한편, 그리드 환경에서 분산형 구조에 대한 대표적인 연구로는 *Condor의 동적인 Flocking 기법*[3]이 있다. [3]에서는 두 개 이상의 클러스터가 존재할 때, 각 클러스터의 중앙 관리자(Central Manager)가 링구조를 가지는 오버레이 네트워크로 구성한다. 그러나 오버레이 네트워크는 동적이고 자가조직적으로 구성되지 않을 뿐만 아니라 휘발성도 고려하고 있지 않다.

3. 자가 조직적 계산 오버레이 네트워크 구성 기법

3.1. SelfCON 구조

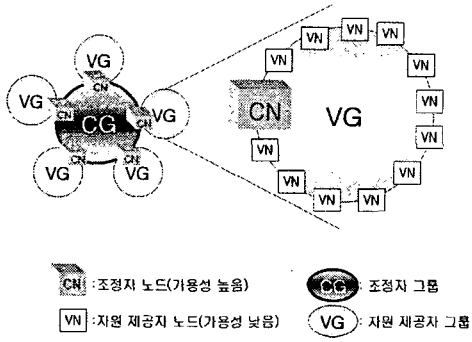
*자가 조직적 계산 오버레이 네트워크*는 자원 제공자가 유지하고 있는 이웃 노드들의 정보를 이용하여 자가 조직적으로 구성된 작업 수행을 위한 네트워크이다. SelfCON은 크게 [그림 1]과 같이 하나의 조정자 그룹과 다수의 자원 제공자 그룹으로 구성된다.

*조정자 그룹(CG:Coordinator Group)*은 가용성이 높은 자원 제공자(조정자 노드)들로 구성된다. 조정자 그룹 내의 조정자 노드는 클라이언트로부터 작업 묶음을 가져와 자원 제공자 그룹내의 자원 제공자들에게 단위 작업을 할당하고 관리하는 역할을 수행한다. 또한 부하 균형을 위해 다른 조정자 노드와 협력한다.

*자원 제공자 그룹(VG:Volunteer Group)*은 조정자 노드를 제외한 노드들로 구성된다. 자원 제공자 그룹 내의 노드들은 자신이 속한 조정자로부터 단위 작업을 할당받아 수행한 후 결과를 반환한다. 조정자 그룹과 자원 제공자 그룹은 각각 Chord[11] 형태의 분산 해쉬 테이블(DHT:Distributed Hash Table)을 사용하여 구성된다.

이렇게 SelfCON에서는 가용성이 높은 자원 제공자를 조정자 그룹으로 구성하고 조정자 노드가 작업 분배, 작업 결과 수집 등의 노드 관리 기능을 수행함으로써 안정적인 연산 수행을 보

* 본 연구는 한국과학기술정보연구원(KISTI) 초고속응용기술지원사업 지원에 의해서 수행되었음



[그림 2] SelfCON 구조

장한다.

3.2. SelfCON 구성 알고리즘

SelfCON은 자원 제공자의 가용성에 따라 다음과 같은 알고리즘에 의해 구성된다.

1) 클라이언트(작업을 소유한 노드)는 자신이 유지하고 있는 이웃 노드들의 정보(가용성, 위치, 성능)를 이용하여 조정자 그룹과 자원 제공자 그룹을 구성한다.
2) 클라이언트는 이웃 노드들에게 "SelfCON 구성 메시지"를 전달한다.
3) SelfCON 구성 메시지를 받은 노드는 이웃 노드들의 정보를 이용하여 조정자 그룹과 자원 제공자 그룹을 구성한 후 다시 자신의 이웃 노드들에게 "SelfCON 구성 메시지"를 보낸다. 이 과정은 TTL(Time To Live) 만큼 반복 수행된다.

조정자 노드는 가용성이 경계값 β보다 높고 성능이 높은 노드가 우선적으로 선출된다. 그리고 자원제공자 그룹은 위치가 같은 노드들로 구성된다.

위 알고리즘을 통해 SelfCON이 구성된 후에, 임의의 노드가 SelfCON에 참여하고자 한다면 해당 노드는 참여 메시지를 자신이 알고 있는 이웃 노드들에게 보내게 된다. 참여 메시지를 받은 노드가 조정자로써 이미 참여하고 있다면, 자신의 자원 제공자 그룹으로 받아들인다. 만약 조정자 노드가 아니면 자신의 조정자 노드를 참여를 원하는 노드에게 알려주게 된다.

4. 결함포용적 스케줄링 기법

본 장에서는 구성된 SelfCON의 특징에 맞는 스케줄링 기법과 결함 포용 기법을 제안한다. 스케줄링은 크게 조정자 그룹 내의 스케줄링과 자원 제공자 그룹 내의 스케줄링으로 구성된다.

4.1. 조정자 그룹 내의 스케줄링

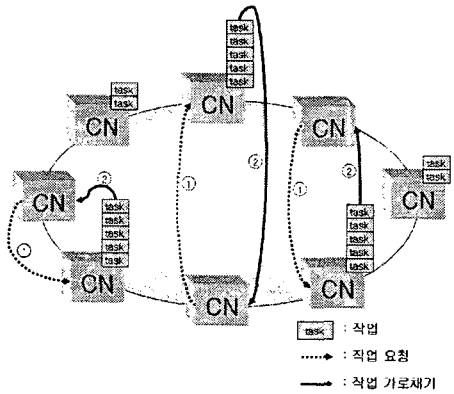
조정자 그룹 내의 스케줄링은 클라이언트로부터 조정자 노드에게 작업 묶음을 할당하는 과정과 조정자 노드들간 작업 가로채기 과정으로 이루어진다. 먼저, 클라이언트가 위탁해야 할 작업이 생기면, 클라이언트는 SelfCON을 구성한 다음 조정자 그룹에, 즉, 가용성이 높고 성능이 높은 조정자 노드부터 작업 묶음을 할당한다.

조정자 노드가 자신의 자원 제공자 그룹에 작업을 할당한 후 조정자 노드들간의 부하 불균형이 발생하면 부하를 일정한 범위로 유지시키는 *느슨한 부하 균형(LLB:Loosely Load Balancing)* 과정을 수행한다.

4.1.1 느슨한 부하 균형(LLB:Loosely Load Balancing)

느슨한 부하 균형(LLB:Loosely Load Balancing)은 다음과 같이 정의된다. 지금까지 수행한 단위 작업의 수를 CT_v, 단위 작업 k를 완료하는 데 걸린 수행 시간을 FT_v라고 하면, 노드 v의 성능 $Perf(v)$는 다음과 같이 나타낼 수 있다.

$$Perf(v)=\frac{CT_v}{\sum_{k=1}^{\sigma} FT_k} \quad (1)$$



[그림 3] 작업 가로채기를 이용한 스케줄링

어떤 자원 제공자 v의 작업큐에 있는 작업의 개수를 QT_v이라고 하면 노드 v의 부하 $Load(v)$는 다음과 같이 나타낼 수 있다.

$$Load(v)=\frac{QT_v}{Perf(v)} \quad (2)$$

LLB는 각 노드의 부하를 일정한 범위로 유지시키는 방법을 의미한다. 즉, 모든 자원 제공자 v는 상수 σ_{min}, σ_{max}에 대하여 다음을 만족한다.

$$\forall v, \sigma_{min} \le Load(v) \le \sigma_{max} \quad (3)$$

LLB를 유지하는 것은 가급적이면 많은 노드가 연산을 수행하는 것을 목적으로 한다.

4.1.2 LLB를 위한 작업 가로채기 알고리즘

어떤 노드 v에 대하여 $Load(v)<\sigma_{min}$이면, 자신의 그룹에 속한 다른 노드의 작업 중 일부를 가져온다. 이를 *작업 가로채기(work stealing)*라고 한다. 이 과정에서 작업을 전달할 노드를 효율적으로 찾아내기 위해서 이동 에이전트를 사용한다. [그림 2]는 이동 에이전트의 작업 가로채기를 이용한 스케줄링을 나타낸다.

어떤 노드 t가 작업 가로채기를 해야 하는 상황이 되면, 노드 t는 자신의 그룹에 속한 인접 노드에 이동 에이전트를 보내어 작업을 가져온다. 에이전트는 노드들 사이를 돌아다니면서 작업큐의 크기가 가장 큰 노드를 기억한다. 이 노드를 이동 에이전트 a에 대해서 a_{max}라고 부르기로 한다. TTL 값이 0이 되면 에이전트는 노드 t로 되돌아온다. 이 때, 노드 a_{max}의 부하가 t보다 크다면 t는 a_{max}로부터 작업을 가져와서 두 노드의 부하를 같게 한다. 노드 t에 새롭게 추가되어야 하는 작업의 개수를 NT_t라고 하면, $\frac{QT_t+NT_t}{Perf(t)}=\frac{Load(t)+Load(a_{max})}{2}$이 되어야 하므로 t에 $NT_t=\lfloor\frac{Load(a_{max})Perf(t)-QT_t}{2}\rfloor$개의 작업을 위탁한다. [그림 3]은 이동 에이전트의 작업 가로채기 알고리즘을 나타낸다.

4.2. 자원 제공자 그룹 내의 스케줄링

자원 제공자 그룹 내의 스케줄링은 조정자가 자원 제공자들에게 단위 작업을 할당하는 과정이다. 자원 제공자 그룹 내의 스케줄링에서는 그룹 내의 자원 제공자가 수행할 작업이 없는 경우 조정자에게 작업을 요청하면 조정자 노드는 단위 작업을 할당하는 선종결 우선 할당 스케줄링(Eager Scheduling) 기법[7]을 사용한다.

4.3. 결함 포용 알고리즘

조정자 그룹 내에서 조정자 노드의 고장에 대처하기 위해 다른 조정자 노드들에 자신의 스케줄링 정보와 작업 결과를 저장한다. 만약 한 조정자 노드가 다른 조정자 노드의 고장을 탐지

```
var
   a : mobile agent; // 이동 에이전트
   n : node;         // 노드가 현재 위치한 노드
   t : node;         // 작업 가로채기를 원하는 노드
   TTL : Time-To-Live // Time-To-Live

if( Load(t) < σ_min ) {
   while(TTL > 0) {
      n = randomNeighborNode();
      a.moveTo( n ); // 임의의 이웃 노드로 이동
      if( Load(n) > Load(a_max))
         a_max = n; // 부하가 가장 큰 노드를 기억함
      TTL = TTL - 1;
   }

   a.moveTo( t ); // 노드 t로 복귀

   /* 부하 균형 과정 */
   if( Load(a_max) > Load(t) {
         NT_t = ⌊ (Load(a_max)Perf(t) - QT_t) / 2 ⌋
         t.workStealing( a_max, NT_t)
           // 노드 t가 a_max에서 NT_t개의 작업을 가져옴
   }
}
```

[그림 4] 이동 에이전트 기반 작업 가로채기 알고리즘

한 경우 조정자 그룹에 저장되어 있는 스케줄링 정보를 이용하여 고장난 조정자 그룹내의 노드들에게 "ChangeCoordinator 메시지"를 보내 조정자 노드를 자신으로 바꾸게 한다.

만약 고장난 조정자 그룹 내의 노드들 중에서 가용성이 임계값 β보다 높은 후보 노드가 있다면 후보 노드를 조정자 노드로 대체시키고 스케줄링 정보와 연산 결과를 보내게 된다.

자원 제공자 그룹 내에서 임의의 노드가 고장이 난 경우 해당 노드는 작업 결과를 조정자 노드에게 반환하지 못하는 결과가 발생한다. 다시 말해, 조정자 노드는 임의의 노드에서 작업 결과를 일정 시간(Timeout)까지 받지 못하면 해당 노드가 고장난 것으로 판단하고 작업을 다른 노드에 다시 할당한다.

5. 토론 및 분석

본 장에서는 기존 기법과 제안 기법에 대하여 토폴로지 구성 방법, 결함 포용, 안정성, 스케줄링 등을 비교한다. 비교 대상이 되는 기존 기법은 크게 트리 구조[1]와 그래프 구조[2,5]로 나뉜다. [표 1]에서와 같이 제안 기법은 기존 기법과 달리 가용성을 고려하여 계산 오버레이 네트워크를 구성하기 때문에 노드의 휘발성이 발생할지라도 안정적 연산 수행을 보장한다.

다음은 노드 결함시 트리 구조와 SelfCON 구조에서 발생하는 노드 재구성 비용 NRR을 나타낸다. 단, 이 때 연산에 참여하는 k개의 노드들을 가용성에 따라 오름차순으로 배열했을 때, i번째 노드의 가용성을 a_i, 이웃 노드의 수를 g_i, 전체 노드수에 대한 조정자 노드의 비율을 δ라고 한다.

$$NRR_{TREE} = \sum_{i=1}^{k} a_i g_i \quad (4)$$

$$NRR_{SELFCON} = \sum_{i=1}^{(1-\delta)k} a_i + \sum_{i=(1-\delta)k+1}^{k} a_i g_i \quad (5)$$

트리 구조는 노드가 실패 했을 경우, 인접 노드에 모두 노드 재구성 메시지를 보내야 하므로, 트리 구조의 노드 재구성 비용은 식 (4)와 같다. SelfCON 구조는 가용성이 높은 노드로 이루어진 조정자 노드만 인접 노드에 재구성 노드를 보내면 되므로, SelfCON 구조의 노드 재구성 비용은 식 (5)와 같다. 식 (4)가 식 (5)에 비해 더 많은 오버헤드를 가지게 됨을 알 수 있다.

결론적으로, 제안 기법은 가용성이 높은 조정자 노드가 작업 관리, 작업 분배 등의 중요한 역할을 담당하여 연산의 안정성을 높인다.

6. 결론 및 향후 연구

본 논문에서는 분산형 P2P 그리드 시스템에서 안정성을 높이기 위해 가용성을 고려한 SelfCON 노드 구성 기법과 그 위에서 동작하는 연산 수행 모델 및 결함 포용적 스케줄링 기법을 제안하였다. 제안 기법은 자원 제공자의 참여와 탈퇴가 자유로운 환경에서 안정적인 연산 수행을 보장하고 시스템 전체의 연산 성능을 향상시킨다.

향후에는 SelfCON 노드 구성 위에서 동작하는 복제 기법 및 정확성 검사 기법을 연구하고자 한다.

항목	트리 구조[1]	그래프 구조[2,5]	SelfCON 구조
노드 구성 및 작업 분배시 고려사항	성능	부하	가용성, 성능, 부하, 지역
토폴로지 구성 형태	성능을 기반으로 한 트리 구성. 성능이 높은 노드일수록 레벨이 낮음.	무작위 네트워크[2], 작은 세계 네트워크[5]로 구성.	가용성, 성능, 지역을 고려한 중첩형 링구조.
휘발성	고려하지 않음	고려하지 않음	가용성을 고려
결함 포용	각각의 노드는 조상 노드를 모두 기억하고, 부모 노드 결함시 조상 노드에게 알림.	고려하지 않음	조정자 노드의 결함 발생시 후보 노드가 그 역할을 대체. 자원 제공자의 결함은 조정자 노드가 처리.
결함 발생시 노드 재구성 비용	$\sum_{i=1}^{k} a_i g_i$	고려하지 않음	$\sum_{i=1}^{(1-\delta)k} a_i g_i + \sum_{i=(1-\delta)k+1}^{k} a_i$

[표 2] 기존 기법과 제안 기법의 비교

참고 문헌

[1] A. J. Chakravarti, G. Baumgartner and M. Lauria, "Self-Organizing Scheduling on the Organic Grid", International J. of High Performance Computing Applications, 2006

[2] D. Wen, J. Yan, L. Zhong and Z. Peng, "A Simple Constructing Approach to Build P2P Global Computing Overlay Network", ICANN, LNCS 2714, 2003

[3] A. R. Butt, R. Zhang and Y. C. Hu, "A Self-Organizing Flock of Condors", Conference on Supercomputing, 2003

[4] V. Lo, D. Zappala, D. Zhou, Y. Liu and S. Zhao, "Cluster Computing on the Fly: P2P Scheduling of Idle Cycles in the Internet", IPTPS, LNCS 3279, pp.227-236, 2004

[5] A. Montresor, H. Meling and ö. Babaoğlu, "Messor: Load-Balancing through a Swarm of Autonomous Agents", AP2PC, LNAI 2530, 2003

[6] SETI@home, "http://setiathome.ssl.berkeley.edu"

[7] M. O. Neary and P. Cappello, "Advanced eager scheduling for Java-based adaptive parallel computing", Concurrency - Practice and Experience, 2005

[8] L. F. G. Sarmenta and S. Hirano, "Bayanihan: Building and Studying Web-Based Volunteer Computing Systems Using Java", FGCS, 1999

[9] D. P. Anderson, "BOINC: A System for Public-Resource Computing and Storage", GRID, 2004

[10] H. Meer and C. Koppen, "Self-Organization in Peer-to-Peer Systems", Peer-to-Peer Systems and Applications, LNCS 3485, pp.247-266, 2005

[11] I. Stoica, R. Morris, D. L-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications". IEEE/ACM Trans. Netw., 2003

[12] D. Nurmi, J. Brevik, R. Wolski, "Modeling Machine Availability in Enterprise and Wide-Area Distributed Computing Environments", Euro-Par, 2005