韓國 컴퓨터情報學會 論文誌
第13卷 第4號, 2008. 7.

2008-13-4-1-4

다중 그리드 사이트에서 어플리케이션 특성을 고려한 동적 작업 재배치 정책

류 경 후*, 이 원 주**, 전 창 호***

A Dynamic Job Relocation Strategy Considering Application's Characteristics in Multiple Grid Sites

Kyung Hoo Ryu*, Won Joo Lee**, Chang Ho Jeon***

요 약

본 논문은 다중 그리드 사이트에서 어플리케이션의 특성을 고려한 동적 작업 재배치 정책을 제안한다. 이 정책은 다중 그리드 사이트에서 수행할 어플리케이션의 특성에 따라 계산 집약 어플리케이션 또는 네트워크 집약 어플리케이션으로 분류한다. 또한, 네트워크 집약 어플리케이션을 단일 사이트에 할당하여 사이트 간의 통신을 제거함으로써 전체 작업실행시간을 단축한다. 하지만 단일 사이트에 네트워크 집약 어플리케이션을 수행할 유휴 노드 수가 존재하지 않으면 네트워크 지연을 최소화하도록 다중 사이트에 먼저 할당한다. 그리고 단일 사이트에 네트워크 집약 어플리케이션을 수행할 수 있는 유휴 노드의 수가 생성되면 다중 사이트에서 수행중인 어플리케이션을 단일 사이트로 재배치하여 전체 작업실행시간을 단축한다. 본 논문에서는 시뮬레이션을 통하여 제안한 동적 적업 재배치 정책이 기존 스케줄링 정책에 비해 데이터 그리드의 성능 향상 면에서 우수함을 보인다.

Abstract

In this paper, we propose a dynamic job relocation strategy that considering application's characteristics in multiple grid sites. This scheme classifies application to execute in multiple grid sites by their characteristics: computing intensive application, network intensive application. Also, it eliminates the communication between sites by allocating the network intensive application in single site, thus reducing the total job execution time. But if a number of free nodes to execute the network intensive application aren't found in single site, the proposed scheme the first allocates the network intensive application in multiple sites to minimize network latency. Then if the network intensive application being executed in multiple sites suitable free nodes are found in single site, the proposed scheme relocates the application being executed in multiple sites to another single site. This results in reducing the total job execution time. Through simulation, we show that the proposed dynamic job reallocation strategy improves the performance of Data Grid environment compared with previous strategies.

▸ Keyword : 다중 그리드(Multiple Grid), 재배치(Reallocation), 네트워크 지연(Network Latency)

• 제1저자 : 류경후 교신저자 : 이원주
• 접수일 : 2008. 5. 14, 심사일 : 2008. 5. 27, 심사완료일 : 2008. 7. 25.
* LG전자 MC R&D 센터 연구원 ** 인하공업전문대학 컴퓨터정보과 부교수 *** 한양대 전자컴퓨터공학부 교수

Ⅰ. 서 론

병렬처리 라이브러리의 표준 규약인 MPI (message passing interface)는 병렬처리 과정에서 발생하는 정보 교환 기능을 제공한다[1]. MPI 어플리케이션은 대부분 네트워크 지연이 적은 단일 클러스터 환경에서 사용되었다. 그러나 인터넷 환경에서는 지리적으로 분산된 단일 클러스터들을 WAN으로 연결하여 확장 가상 클러스터 환경을 구성한다[2]. 이러한 확장 가상 클러스터환경에서 기존의 스케줄링 정책을 사용하여 MPI 어플리케이션을 실행하면 시스템의 성능 저하가 발생한다.

기존의 스케줄링 정책은 자원의 이용 상태를 고려하여 자원의 이용률을 높일 수 있었다. 그러나 확장 가상 클러스터 환경을 구성하는 네트워크가 LAN에서 WAN으로 변경되면서 네트워크 지연과 대역폭에 큰 차이가 발생하는 것을 고려하지 못하였다[3]. 또한 어플리케이션의 특성을 고려하여 작업을 스케줄링함으로써 얻을 수 있는 전체 작업실행시간 단축 효과도 얻지 못하였다. 따라서 그리드 환경의 성능 향상을 위해서는 WAN 환경에 적용할 수 있는 새로운 스케줄링 정책이 필요하다.

본 논문에서는 자원의 동적 할당이 가능한 다중 그리드 사이트 환경에서 어플리케이션 특성을 고려한 동적 작업 재배치 정책을 제안한다. 이 정책은 다중 그리드 사이트에서 수행할 어플리케이션의 특성에 따라 계산 집약 어플리케이션 또는 네트워크 집약 어플리케이션으로 분류한다. 네트워크 집약 어플리케이션은 단일 사이트에 할당하여 사이트 간의 통신을 제거함으로써 전체 작업실행시간을 단축한다. 하지만 단일 사이트에 네트워크 집약 어플리케이션을 수행할 유휴 노드 수가 존재하지 않으면 네트워크 지연을 최소화 할 수 있는 다중 사이트에 먼저 할당한다. 그리고 단일 사이트에 네트워크 집약 어플리케이션을 수행할 수 있는 유휴 노드의 수가 생성되면 다중 사이트에서 수행중인 어플리케이션을 단일 사이트로 재배치하여 전체 작업실행시간을 단축한다.

본 논문의 구성은 다음과 같다. 2장에서는 그리드 환경에서 사용하는 기존의 그리드 스케줄링 정책과 MPI 어플리케이션 특성에 대하여 설명한다. 3장에서는 제안하는 동적 작업 재배치 정책에 대하여 설명한다. 그리고 4장에서는 시뮬레이션을 통하여 제안하는 동적 작업 재배치 정책의 성능을 평가하고, 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

2.1 기존 스케줄링 정책

최소실행시간 예측에 의존하는 기존의 스케줄링 정책은 그리드 환경에 적합하지 않다. 지리적으로 분산된 컴퓨팅 환경과 이기종 컴퓨팅 자원의 유동성과 활용을 고려하지 않기 때문이다. 기존의 스케줄링 정책에서는 컴퓨팅 자원에 할당된 작업이 종료되거나 취소될 때까지 해당 자원에서만 처리한다. 그리고 스케줄러는 스케줄링 시점까지 요청된 작업에 대하여 최적의 자원을 선택하여 할당한다[4][5][6]. 그러나 그리드 환경에서는 많은 사용자들에 의해 예측 불가능한 시간에 다양한 유형의 작업 처리 요청이 발생한다. 이때 새로 운 작업에 가장 적합한 자원이 다른 작업을 수행하고 있다면 할당할 수 없기 때문에 시스템의 성능을 저하시킨다. 시스템의 성능 저하를 줄이기 위해서는 입력 작업에 대한 최적의 자원 배치가 동적으로 이루어져야 한다. 또한 최적의 자원이 아닌 다른 자원에 할당된다면 작업실행시간이 증가한다. 따라서 최적의 자원에서 수행중인 작업을 다른 자원으로 이주시키고 할당할 수 있는 정책이 필요하다.

그리드 환경에서 자원 할당을 위한 스케줄링 시스템의 구성은 그림 1과 같다.

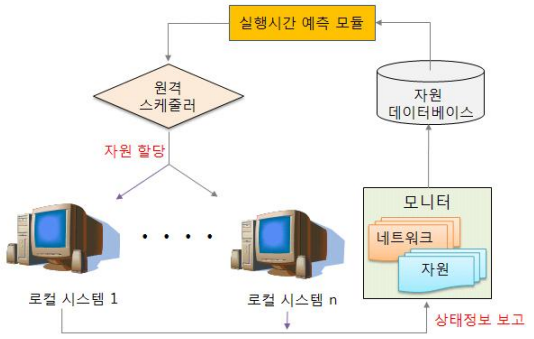


그림 1. 그리드 스케줄링 시스템의 구성
Fig. 1. Component of Grid scheduling system.

그림 1에서 모니터는 네트워크 모니터와 자원 모니터로 구성된다. 네트워크 모니터는 네트워크의 전송속도 및 사용 상태를 모니터링하고, 자원 모니터는 로컬 시스템의 사용 상태를 모니터링 한다[7].

2.2 MPI 어플리케이션의 특성 분석

병렬컴퓨터의 성능 평가를 위하여 널리 사용하는 MPI 어플리케이션은 NPB이다. NPB는 계산 유체 역학(Computational Fluid Dynamic)에서 널리 사용되는 어플리케이션으로 8개의 대표적인 서브프로그램으로 구성되어 있다[9][10]. 이러한 서브프로그램을 대상으로 노드 수에 따른 성능 변화와 사이트 간의 네트워크 지연에 따른 성능 변화를 살펴봄으로써 어플리케이션의 특성을 알 수 있다.

먼저 노드 수에 따른 성능 변화를 살펴본다. CG는 노드 수 40개까지는 시스템의 성능이 향상되지만 노드 수 140개 이상에서는 노드간의 통신 빈도 증가에 따라 시스템의 성능이 기준이하로 떨어진다. 따라서 CG는 노드간의 통신이 많음을 알 수 있다. EP는 노드 수의 증가에 따라 시스템의 성능이 선형적으로 증가한다. 따라서 EP는 노드간의 통신량의 증가에 따른 영향을 거의 받지 않음을 알 수 있다. LU는 노드 수의 증가에 따라 성능 변화는 CG의 결과와 유사하지만 시스템의 성능이 최고점에 도달 할 때의 노드 수는 70개로 CG보다 크다. 따라서 LU는 노드간의 통신이 CG에 비해 적음을 알 수 있다.

이러한 결과를 분석해 보면 노드간의 통신에 따라 시스템의 성능이 민감한 어플리케이션은 CG 〉 LU 〉 EP 순 임을 알 수 있다. 또한 노드 수에 따라 시스템의 성능이 가장 민감한 어플리케이션은 EP 〉 LU 〉 CG 순 임을 알 수 있다.

이러한 어플리케이션 특성을 고려하여 작업을 스케줄링 한다면 시스템의 성능을 향상시킬 수 있다.

III. 동적 작업 재배치 정책

본 논문에서는 어플리케이션 특성을 고려하여 작업을 재배치함으로써 작업처리시간을 최소화하는 스케줄링 정책을 제안한다. 이 정책은 노드간 통신에 민감한 어플리케이션을 단일 사이트에 할당하여 사이트 간의 통신을 제거한다. 하지만 어플리케이션이 필요로 하는 노드 수가 단일 사이트의 노드 수 보다 클 경우에는 다수의 사이트에 할당한다. 이때 네트워크 지연이 최소가 되도록 할당하여 시스템의 성능 저하를 줄인다.

3.1 용어

먼저 본 논문에서 사용하는 용어에 대하여 설명한다.

- 작업(Job) : 작업은 사용자에 의해 투입되는 어플리케이션의 집합이다.
- 재배치 (Reallocation) : VMM의 Guest OS Migration을 이용하여 작업을 재배치한다.
- 작업처리시간: 작업에 포함된 모든 어플리케이션의 수행을 완료하는데 소요되는 시간이다.
- 네트워크 집약 어플리케이션(Network Intensive Application) : 네트워크 환경에 민감하게 영향을 받는 어플리케이션이다.

어플리케이션은 독립적으로 실행 가능한 각각의 병렬프로그램이고, CG와 LU 등이 네트워크 집약 어플리케이션에 해당한다.

3.2 동적 작업 스케줄링 알고리즘

제안하는 동적 작업 스케줄링 알고리즘은 FIFO(First In First Out)에 따라 작업을 처리한다. 동적 작업 스케줄링 알고리즘은 그림 2와 같다.

```
Queue Job_queue; //작업 요청 큐
//전체시스템의 유휴 노드 수
System.nodes.available.count;
task.nodes_require; //어플리케이션 요구 노드 수
while(task == Job_queue.pop())
{
  if(System.nodes.available_count >
    task.nodes_require)
  {
    if(nodes == Find_SingleSite(task)) //단일사이트
       AllocateTo(task, nodes);
    else{ // 단일 사이트에서 실행할 수 없을 경우
       //네트워크 지연이 최소인 다중사이트 탐색
       nodes = Find_LowestSites(task);
       AllocateTo(task, nodes);
       //이주 가능 노드 탐색
       if(is_migrationable(Ta, Tb))
              Do_Migration();
    }
  } else {  // 작업을 처리할 노드가 없음
  //작업이 종료될 때까지 대기
  Wait_for_task_end();
  //이주 가능한 노드 탐색
  if(is_migrationable(Find_High_Comm_Task(
     System.nodes.available.Count),null))
  {
     Do_migration(Find_High_Comm_Task(
       System.nodes.available.Count),
       System.nodes.available);
  }
  continue;
}
```

그림 2. 작업 재배치 스케줄링 알고리즘
Fig. 2. Job reallocation scheduling algorithm

그림 2에서 처리할 작업들은 Job_queue에 대기한다. 그리고 전체 시스템의 유휴 노드 수(System.nodes.available.count)와 어플리케이션 요구 노드 수(task.nodes_require)를 구한다. 이때 전체 시스템의 유휴 노드 수가 어플리케이션 요구 노드 수보다 크면 Job_queue에서 패치 된 작업을 할당한다. 하지만 반대의 경우에는 할당 가능한 유휴 노드가 생성될 때까지 대기한다. 작업 할당이 가능한 경우에는 작업의 모든 어플리케이션에 대하여 할당 가능한 단일 사이트가 존재하면 해당 단일 사이트에 할당한다. 하지만 단일 사이트가 존재하지 않으면 네트워크 지연을 최소화 할 수 있는 다중 사이트에 할당한다. 다중 사이트에 할당된 어플리케이션은 네트워크 지연을 최소화하기 위해 단일 사이트내의 노드로 이주시킨다.

그림 2의 작업 재배치 스케줄링 알고리즘에서 사용하는 관련 함수는 그림 3과 같다.

```
Bool is_migrationable(Task Task_A, Task Task_B)
{
  Task_A.count;  //이주 대상 작업A의 노드 수
  // MPI_Send()로 보낸 초당 통신량
  Task_A.comm_size;
  Task_B.count;     // 이주 대상 B의 노드 수
  //이용 가능한 노드 수
  System.nodes.available.count;
  if(Task_A.count + System.nodes.available.count
     >= Task_B.count)
  if(Task_A.comm_size > Task_B.comm_size)
      return   true;
}

Task Find_High_Comm_Task(int
            available_nodes_count)
{
  while(System.tasks.highcomm[i].nodes_require >
        available_nodes_count)
  {
      i++;
      return   System.tasks.highcomm;
  }
}
```

그림 3. 작업 재배치 스케줄링 알고리즘의 관련 함수
Fig. 3. Functions for job reallocation scheduling algorithm

그림 3에서 is_migrationable() 함수는 이주 가능한 최적의 노드를 찾는 기능을 제공한다. Find_High_Comm_Task() 함수는 네트워크 집약 어플리케이션을 찾는 기능을 한다.

IV. 성능 평가

그리드 환경에서는 자원의 이용 상태가 실시간으로 변하고, 각 사이트의 부하와 네트워크 환경이 불규칙하다. 또한 스케줄링 알고리즘 적용을 위한 고정적인 자원 확보가 현실적으로 어렵다. 따라서 본 논문에서는 시뮬레이션을 통하여 제안한 작업 재배치 스케줄링 정책이 기존의 스케줄링 정책에 비해 시스템의 성능 향상 면에서 우수함을 검증한다.

4.1 시뮬레이션 환경

그리드 환경을 모델링하고 시뮬레이션 하기 위해 SIMGRID[11]를 사용한다. 작업의 어플리케이션으로 NPB의 서브프로그램인 CG, LU, EP 를 사용한다. 시뮬레이션을 위한 그리드 환경은 그림 4와 같다.

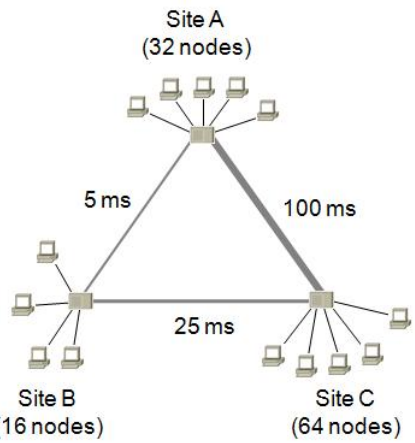
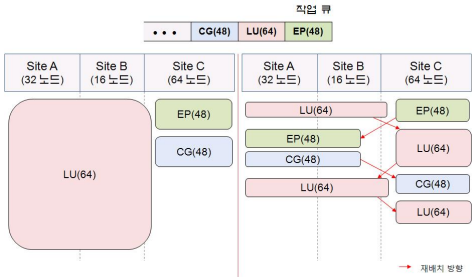


그림 4. 그리드 사이트 모델
Fig. 4. Grid Site Model

그림 4에서 그리드 사이트 A, B, C는 각각 32, 16, 64개의 노드를 가지고 있으며, 사이트 A-B, B-C, A-C 간의 네트워크 지연은 각각 5, 25, 100 ms 이다. 이러한 그리드 사이트 모델링 파일은 xml 로 작성하고, 각 노드의 CPU 속도, 네트워크 대역폭 및 지연, 노드 간의 라우팅 경로 등의 정보를 정의한다.

4.2 시뮬레이션 결과 분석

첫 번째 시뮬레이션은 어플리케이션의 특성에 따른 작업처리시간을 알아보기 위해 그림 5와 같이 작업을 처리한다. 작업 큐에 대기 중인 작업은 EP(48), LU(64), CG(48) 순서로 처리한다. 작업 EP(48)과 CG(48)을 처리하기 위해서는 각각 48개의 노드가 필요하고, LU(64)는 64개의 노드가 필요하다.



(a) 기존 스케줄링 정책 (b) 제안한 재배치 스케줄링 정책

그림 5. 작업 스케줄링 과정 예
Fig. 5. Example of job scheduling process.

그림 5의 (a) 기존 스케줄링 정책을 적용한 결과를 살펴본다. 48개의 노드를 필요로 하는 작업 EP(48)는 노드간 통신을 줄이기 위해서 Site C에 먼저 할당한다. 이때 Site C는 64개 노드 중에 48개 노드에 작업 EP(48)를 할당받아 수행하기 때문에 나머지 16개의 유휴 노드가 발생한다. 그리고 작업 LU(64)는 64개의 노드를 필요로 하기 때문에 Site A, B, C에 할당하여 수행한다. 작업 EP(48)의 수행이 종료된 후에 작업 CG(48)를 Site C에 할당한다. 이때 작업 LU(64)는 Site A, B, C에 걸쳐 실행되기 때문에 네트워크 지연에 따른 성능 저하로 전체 작업실행시간이 길어진다. 이것은 기존 스케줄링 정책이 어플리케이션의 특성을 고려하지 않기 때문이다.

(b)제안한 재배치 스케줄링 정책의 경우, 48개의 노드를 필요로 하는 작업 EP(48)를 Site C에 먼저 할당한다. 작업 LU(64)는 64개의 노드를 필요로 하기 때문에 Site A, B, C에 할당하여 수행한다. 하지만 작업 LU(64)는 작업 EP(48)에 비해 노드간의 통신에 민감한 어플리케이션이기 때문에 재배치 스케줄링 정책은 작업 EP(48)를 Site A, B로 재배치하고, 작업 LU(64)를 단일 사이트인 Site C에 재배치하여 노드간의 통신 빈도를 줄임으로써 시스템의 성능을 향상시킨다. 또한 작업 CG(48)는 작업 EP(48) 종료 후에 Site A, B에 할당한다. 하지만 작업 CG(48)는 작업 EP(48)에 비해 노드간의 통신에 민감한 어플리케이션이기 때문에 재배치 스케줄링 정책은 작업 CG(48)를 단일 사이트인 Site C에 재배치하고, 작업 EP(48)를 Site A, B로 재배치하여 노드간의 통신 빈도를 줄인다.

이러한 시뮬레이션에 대한 결과는 그림 6과 같다.

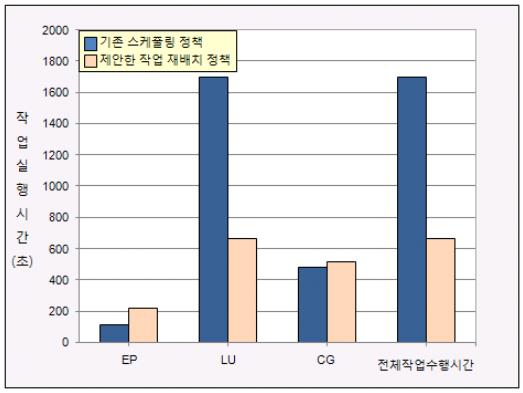
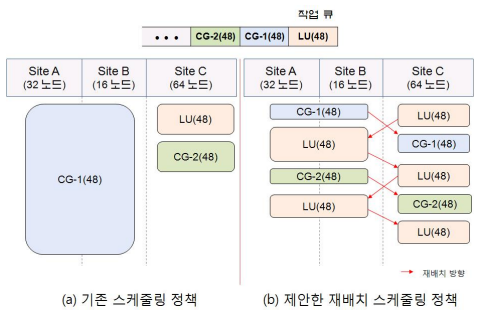


그림 6. 시뮬레이션 결과
Fig. 6. Result of simulation.

그림 6을 살펴보면 작업 EP, CG에서는 제안한 작업 재배치 스케줄링 정책이 기존의 스케줄링 정책보다 약간 불리하지만 작업 LU에서는 성능 향상이 크다는 것을 알 수 있다. 따라서 전체 작업수행시간 면에서도 제안한 작업 재배치 스케줄링 정책이 기존의 스케줄링 정책보다 성능 향상이 큼을 볼 수 있다.

두 번째 시뮬레이션은 그림 7과 같이 48개의 노드를 요구하는 LU와 2 개의 CG를 처리한다.



(a) 기존 스케줄링 정책 (b) 제안한 재배치 스케줄링 정책

그림 7. 작업 스케줄링 과정 예
Fig. 7. Example of job scheduling process.

그림 7의 (a) 기존 스케줄링 정책을 적용한 결과를 살펴본다. 48개의 노드를 필요로 하는 작업 LU(48)는 노드간 통신을 줄이기 위해서 단일 사이트인 Site C에 먼저 할당한다. 그리고 작업 CG-1(48)은 48개의 노드를 필요로 하기 때문에 Site A, B에 할당하여 수행한다. 작업 CG-2(48)는 작업 LU(48)의 수행이 종료되어 48개의 유휴 노드가 생성되는 시점에 Site C에 할당된다. 이때 CG-1(48)은 네트워크 지연 5 ms인 Site A, B에 할당되어 수행되기 때문에 네트워크 지

연에 따른 성능 저하로 인해 전체 작업실행시간이 증가한다.

(b)제안한 재배치 스케줄링 정책의 경우, 48개의 노드를 필요로 하는 작업 LU(48)를 단일 사이트인 Site C에 먼저 할당한다. 그리고 작업 CG-1(48)은 48개의 유휴 노드를 필요로 하기 때문에 Site A, B에 할당하여 수행한다. 이때 작업 CG-1(48)은 작업 LU(48)에 비해 네트워크 지연에 민감한 어플리케이션이다. 따라서 재배치 스케줄링 정책에 따라 작업 LU(48)를 Site A, B로 재배치하고, 작업 CG-1(48)을 단일 사이트인 Site C에 재배치함으로써 시스템의 성능을 향상시킨다. 또한 작업 CG-1(48)이 종료한 후에 Site A, B에 할당되어 수행중인 작업 LU(48)를 단일 사이트인 Site C에 재배치한다. 그리고 작업 CG-2(48)를 Site A, B에 할당한다. 이때 작업 CG-2(48)는 작업 LU(48)에 비해 네트워크 지연에 민감한 어플리케이션이다. 따라서 재배치 스케줄링 정책에 따라 작업 LU(48)를 Site A, B로 재배치하고, 작업 CG-2(48)을 단일 사이트인 Site C에 재배치함으로써 시스템의 성능을 향상시킨다.

이러한 시뮬레이션에 대한 결과는 그림 8과 같다.

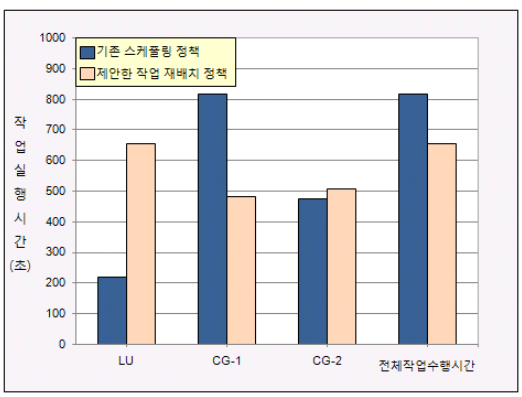


그림 8. 시뮬레이션 결과
Fig. 8. Result of simulation.

그림 8을 살펴보면 제안한 재배치 스케줄링 정책이 기존의 스케줄링 정책에 비해 시스템의 성능 향상도가 크지 않다. 이것은 어플리케이션 CG, LU는 모두 네트워크 집약 어플리케이션이기 때문에 다중 사이트에 걸쳐 작업을 실행하면 시스템의 성능 저하가 증가하기 때문이다. 즉 네트워크 집약 어플리케이션은 작업 재배치 과정에 동반하는 오버헤드 보다 작업 재배치를 통하여 단축한 작업실행시간이 크지만 시스템의 성능 향상 효과는 크지 않은 것이다. 그림 8에서 작업 LU(48)은 다중 사이트인 Site A, B와 단일 사이트인 Site C로 재배치를 반복하였기 때문에 작업실행시간이 증가하였다. 하지만 작업 CG-1(48)는 재배치 스케줄링에 따라 단일 사이트인 Site C에 재배치됨으로써 작업실행시간이 크게 감소하였다. 따라서 시스템의 전체수행시간에서 제안한 작업 재배치 스케줄링이 기존의 스케줄링 정책에 비해 18% 정도의 성능 향상을 보이고 있다.

V. 결론

그리드 컴퓨팅 환경에서 스케줄링 정책은 정적 스케줄링과 동적 스케줄링으로 분류할 수 있다. 정적 스케줄링 정책은 작업의 실행시간 예측이 정확하지 않고, 실시간으로 변하는 작업처리 요청에 유연하게 대응하지 못하기 때문에 처리 효율이 감소한다. 하지만 동적 스케줄링 정책은 그리드 자원의 이용상태와 네트워크 지연 등과 같은 정보를 실시간으로 반영하여 작업을 스케줄링함으로써 시스템의 성능을 향상할 수 있다는 장점이 있다.

따라서 본 논문에서는 MPI 어플리케이션의 특성을 고려하여 작업을 재배치함으로써 전체 작업실행시간을 단축할 수 있는 작업 재배치 스케줄링 정책을 제안하였다. 이 정책에서는 네트워크 지연에 민감한 어플리케이션을 단일 사이트에 우선 배치한다. 또한 통신 빈도가 상대적으로 높은 어플리케이션을 단일 사이트에 우선 재배치함으로써 전체 작업실행시간을 단축하였다. 시뮬레이션을 통한 성능평가에서는 기존의 스케줄링 정책에 비해 제안한 작업 재배치 스케줄링 정책이 18%~60% 정도 시스템의 성능을 향상시킬 수 있음을 보였다.

향후 연구에서는 네트워크 지연 또는 네트워크 경로 손실로 인해 그리드 환경에 참여할 수 없는 노드가 발생하는 경우, 대체 노드를 찾아 작업을 재할당함으로써 시스템의 성능 저하를 방지하는 정책에 대한 연구를 할 수 있다.

참고문헌

[1] Message Passing Interface Forum, "MPI: a Message Interface Standard," 1995.
[2] Ian Foster, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann," 2003.
[3] L. Zhang, "Scheduling Algorithm for Real-Time Application in Grid Environment Systems," 2002 IEEE International Conference on Vol. 5, pp. 6-9 Oct. 2002.

〔4〕 P. Ruth, J. Rhee, D. Xu, R. Kennell, and S. Goasguen, "Autonomic Live Adaptation of Virtual Computational Environments in A Multi-Domain Infrastructure," In the 3rd IEEE International Conference on Autonomic Computing (ICAC'06), June, 2006.
〔5〕 황선태, 김영학, "계산 그리드에서 워크플로우 기반의 사용자 환경 설계 및 구현," 한국컴퓨터정보학회 논문지, 제 10권, 제 4호, pp. 165-171, Sept. 2005.
〔6〕 조수현, 김영학, "계산 그리드 상에서 프로그램의 특성을 반영한 작업 프로세스 수의 결정에 관한 연구," 한국컴퓨터정보학회 논문지, 제 11권, 제 1호, pp. 71-85, Mar. 2006.
〔7〕 H. H. Mohamed and D. H. J. Epema, "Experiences with the Koala co-Allocating Scheduler in Multiclusters," In the 5th IEEE/ACM Int'l Symp. on Cluster Computing and the GRID (CCGrid2005), May, 2005.
〔8〕 R. Butler and E. Lusk, "Monitors, Messages, and Clusters: the P4 Parallel Programming System," Parallel Computing, 1994.
〔9〕 W. Gropp, E. Lusk, and A. Skjellum, "A High-Implementation of the MPI Message Passing Interface Standard," http://wwwunix. mcs.anl.gov/mpi/mpich1/papersmpicharticle/ paper.html
〔10〕 Web-site:nas parallel benchmarks. http://www.nas.gov/Software/NPB
〔11〕 SIMGRID Project, http://simgrid.gforge.inria.fr

저 자 소 개

류 경 후

2005: 한양대학교 전자컴퓨터공학부 학사.
2007: 한양대학교 컴퓨터공학과 석사.
현 재:LG전자 MC연구소 연구원.
관심분야 : 병렬처리시스템, 모바일 컴퓨팅, Grid 컴퓨팅

이 원 주

1989: 한양대학교 전자계산학과 공학사.
1991: 한양대학교 컴퓨터공학과 공학석사.
2004: 한양대학교 컴퓨터공학과 공학박사
현 재: 인하공업전문대학 컴퓨터정보과 부교수
관심분야: 병렬처리시스템, 성능분석, Grid 컴퓨팅, 센서네트워크

전 창 호

1977: 한양대학교 전자공학과 학사.
1982: Cornell University 컴퓨터공학과 석사.
1986: Cornell University 컴퓨터공학과 박사.
1977-1979: 전자통신연구소 연구원.
현 재 : 한양대학교 전자컴퓨터공학부 교수.
관심분야: 병렬처리시스템, 성능분석, Grid 컴퓨팅, 센서네트워크