GRID 환경에서의 스케쥴링 알고리즘 성능분석

조정우[U] 김진석
서울시립대학교 컴퓨터과학부
{jwjo95, jskim}@venus.uos.ac.kr

Performance Evaluation for Scheduling Algorithm on GRID Environment

Jeongwoo Jo[U] Jin Suk Kim
School of Computer Science, University of Seoul

요 약

최근 들어 이질적인 컴퓨팅 자원들을 이용하는 GRID같은 연구가 진행 중에 있다. 이는 여러 지역에 분산되어있는 고성능의 시스템들을 네트워크로 연결하여 작업을 좀더 빠르게 수행시키는데 목적을 두고 있다. 이러한 시스템에서 작업을 수행하면 수행시간을 단축시킬 수 있다는 장점을 가지고 있으나 컴퓨팅 자원들이 여러 지역에 분산되어 있고 각 자원들의 성능이 모두 다르다는 단점 또한 가지고 있다. 따라서 이러한 시스템에서 스케쥴링 정책은 자원의 특성을 고려해야 한다는 문제점을 갖는다. 본 논문에서는 GRID 환경에서 기존의 스케쥴링 알고리즘을 적용가능한지, 그리고 기존의 성능과 유사한 결과를 보이는지를 시뮬레이션을 통해 살펴보았다.

1. 서 론

최근 들어 이질적인 컴퓨팅 자원들로 하나의 거대한 시스템을 구성하여 작업을 처리하고자하는 연구가 많이 진행 중에 있다. 그 중 하나의 예로 GRID를 들 수 있다. GRID는 분산되어 있는 여러 고성능 슈퍼컴퓨터나 클러스터 시스템을 네트워크로 연결하여 작업을 수행하고자하는 프로젝트이다. <그림 1>은 GRID의 예를 보여주고 있다. 그림과 같이 GRID는 클러스터들로 구성되어있으며 각 클러스터들간에는 네트워크로 연결되어 있다. 또한 각 클러스터들은 서로 다른 프로세서들로 구성되어 있다.

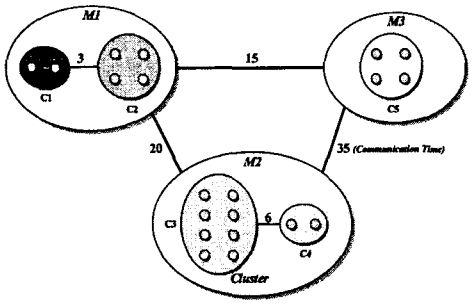

그림 1. GRID 시스템

이러한 시스템에서 작업을 수행하면 수행시간을 단축시킬 수 있다는 장점이 있으나 시스템들의 특성이 서로 다르기 때문에 스케쥴링 정책에서 자원의 특성을 고려해야 한다는 문제점이 발생한다. 이러한 스케쥴링 문제는 NP-Complete 이며 현재까지 다양한 스케쥴링 알고리즘이 제안되었다.

GRID 시스템에서의 스케쥴링 알고리즘에서의 고려사항은 각 클러스터들마다 노드의 수가 다르다는 것과 각 클러스터들의 성능에 차이가 있을 수 있다는 것이다. 본 논문에서는 단일 시스템에서의 스케쥴링을 수행하는 기존의 알고리즘을 GRID에 적용시켜서 성능을 측정하였다. 그리고 기존의 알고리즘이 GRID에서도 같은 성능을 보이는지 시뮬레이션을 통해 살펴보았다. 또한 GRID환경에서 스케쥴링 알고리즘을 실험하기 위해 시뮬레이터를 제작하였다.

2. 관련용어 및 기존 스케쥴링 알고리즘

2.1 관련용어

클러스터링 시스템은 하나의 시스템에 여러개의 노드들이 있으며 각 노드는 빠른 네트워크로 연결되어는 시스템을 말한다. 또한 이질적인 시스템(Heterogeneous System)은 자원의 특성이 서로 다르고 성능 또한 다른 자원들로 구성된 시스템을 말한다.

본 논문에서는 알고리즘의 성능을 측정하는 위해 여러 종류의 시간을 검사한다. 실행시간 (execution time)은 작업을 특정 자원에서 실행시켰을 때 수행되는 시간을 말하며, 준비시간

(ready time)은 특정자원에서 자원에 주어진 모든 작업의 수행을 마친 후 이용가능한 시간을 말한다. 또한 완료시간(completion time)은 준비시간과 실행시간의 합을 이용하여 계산한다. 마지막으로 본 논문에서 알고리즘의 성능을 평가하기 위하여 makespan을 사용하였다. maskspan은 시스템에 주어진 작업이 모두 완료되었을 때의 시간을 나타낸다.

2.2 기존 스케줄링 알고리즘

이질적인 자원으로 구성된 컴퓨팅 환경에서 작업을 스케줄링하는 알고리즘은 정적 스케줄링 방식과 동적 스케줄링 방식으로 나뉜다. 정적 스케줄링 방식은 시스템에 주어지는 작업의 개수와 선후관계를 알 때 사용되는 방식이며, 동적 스케줄링 방식은 작업들간의 선후관계가 불분명하고 자원이 유동적일 때 사용되는 방식이다. 알고리즘으로는 MET(Minimum Execution Time), MCT(Minimum Completion Time), MECT(Minimum Execution and Completion Time) 등이 있다 [3, 8]. MET는 작업을 실행할 때 실행시간이 가장 짧은 자원에 작업을 할당하는 방식이고, MCT는 완료시간이 가장 짧은 자원에 작업을 할당하는 알고리즘이다. MECT는 MET와 MCT방식을 혼합하여 스케줄링하는 알고리즘이다. MET가 가장 단순한 알고리즘인 반면 성능은 가장 낮다. 또한 MECT가 가장 복잡하지만 MCT보다 우수한 성능을 보인다.

하지만 기존의 알고리즘은 이질적인 자원이 단일노드를 갖는다는 가정에서의 스케줄링 알고리즘이기 때문에 하나의 자원이 다중노드를 갖는 GRID 시스템에 바로 적용하기는 힘들다. 따라서 본 논문에서는 MET, MCT, MECT 알고리즘을 GRID 시스템에 적용할 수 있도록 수정하였다.

3. 스케줄링 알고리즘 수정

기존의 스케줄링 알고리즘을 GRID에 적용시키기 위해서는 우선 몇 가지의 추가적인 가정이 필요하다. 첫번째로 모든 작업은 실행에 필요한 노드의 수를 명시해야 한다. 즉 몇 개의 노드에서 해당작업을 수행할 것인지를 명시하여야 한다. 또한 하나의 작업은 하나의 클러스터에서만 수행된다고 가정하였다. 즉 작업이 두 개 이상의 클러스터에서 실행되지 않는다고 가정하였다. 따라서 작업이 필요로 하는 노드의 수보다 적은 노드를 갖는 클러스터에서는 작업이 수행되지 않는다. 본 가정을 제외하고는 기존의 단일노드에서 수행되는 알고리즘과 동일하다.

4. 실험 및 결과

본 논문에서 시뮬레이션을 하기 위해 필요한 기대시간은 작업·자원행렬을 사용하여 미리 계산되어진다. 각 행은 작업을 나타내고 열은 자원을 나타낸다. 또한 각 원소는 작업의 실행시간을 나타낸다. 시뮬레이션을 위해 작업의 이질성, 자원의 이질성과 작업의 개수, 자원의 개수를 입력하고 Matrix의 특성을 나타내는 Model을 선택한다.

작업의 이질성은 작업들간의 실행시간의 차이를 두기 위해 0부터 입력한 값 사이의 랜덤값으로 생성되고, 자원의 이질성은 클러스터들간의 성능차이를 두기 위해 마찬가지로 랜덤으로 생성된다. 각 원소는 각 작업의 실행시간을 나타내는데 계산식은 다음과 같다.

$$\textit{실행시간} = \textit{작업의 이질성(Random)} * \textit{자원의 이질성(Random)}$$

본 Matrix를 생성하는 프로그램은 <그림 2>와 같다.

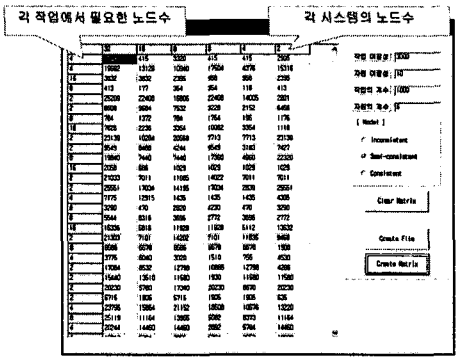

그림 2. Job Matrix 생성 프로그램

위에서 생성된 작업 Matrix를 GRID에 적용 가능하도록 수정한 MET, MCT, MECT 스케줄링 알고리즘에서 수행하여 결과를 비교하였다. 시뮬레이터는 다음 <그림 3>과 같이 우선 작업을 선택하고 각 알고리즘별로 Makespan을 계산한다.

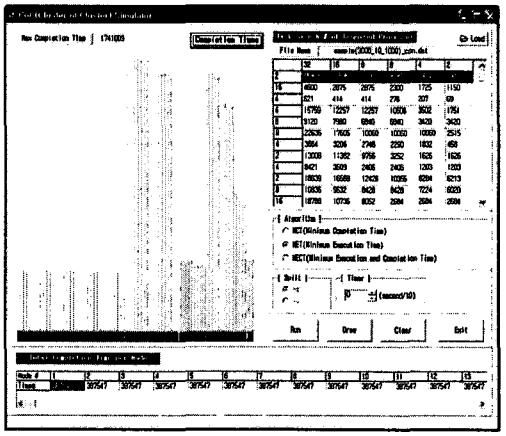

그림 3. 스케줄링 알고리즘 시뮬레이터

실험은 작업의 이질성과 자원의 이질성에 따라 각 알고리즘의 성능이 어떻게 차이가 나는지 살펴보았다.

4.1 자원의 이질성에 따른 성능 평가

본 논문에서는 자원의 이질성에 따라 각 스케줄링 알고리즘이 GRID 시스템에서 어떤 성능을 보이는지 살펴보았다. 자원의 이질성은 10, 120으로 설정하였으며 스케줄링 알고리즘의 성능은 모든 작업의 수행이 완료되었을 때의 Makespan을 비교하였다. 성능은 <그림 4, 5, 6>에 나타나 있듯이 자원의 이질성이 커짐에 따라 가장 단순한 알고리즘인 MET가 성능이 낮음을 알 수 있다. 하지만 기존의 단일노드에서의 결과와는 달

리 MCT와 MECT의 성능차이는 거의 없음을 알 수 있다.

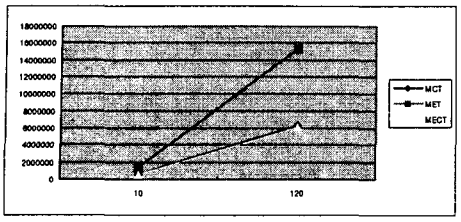

그림 4. Consistent Model

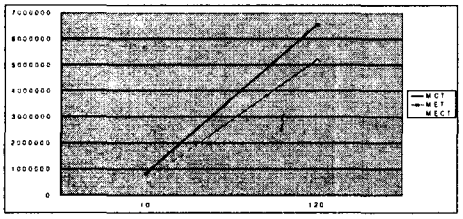

그림 5. Inconsistent Model

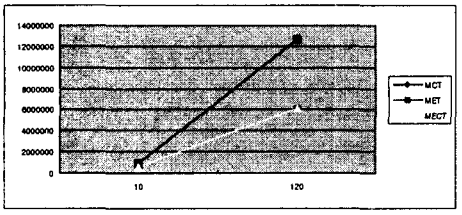

그림 6. Semi-consistent Model

4.2 작업의 이질성에 따른 성능 평가

작업의 이질성에 따라 각 알고리즘이 어떤 성능 보이는지 살펴 보기 위해 작업의 이질성을 1000, 3000으로 놓고 실험을 하였다. 작업의 이질성의 차이도 자원의 이질성의 차이와 마찬가지로 MET가 가장 나쁜 성능을 보이고 MCT와 MECT는 유사한 결과를 보인다.

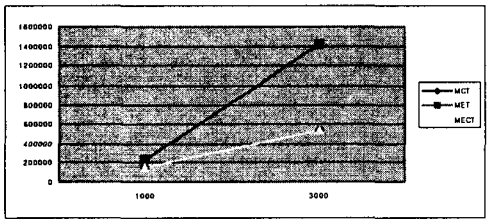

그림 7. Consistent Model

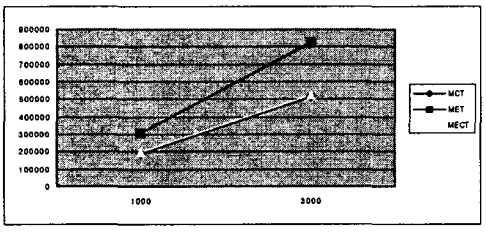

그림 8. Inconsistent Model

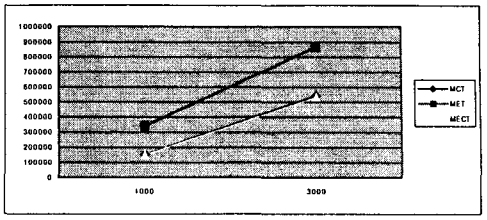

그림 9. Semi-consistent Model

5. 결론

본 논문에서는 기존의 단일노드에서의 스케쥴링 알고리즘 중 MET, MCT, MECT를 GRID환경에서 실행하여 성능을 확인하였다. 본 실험결과는 단일노드에서의 스케쥴링 알고리즘과 거의 유사한 성능을 보였다. 하지만 기존에는 MECT가 MCT보다 성능이 앞섰으나 GRID환경에서는 좋은 성능을 보이지 않는다는 것을 확인할 수 있었다.

참고문헌

[1] Shreenivasa Venkataramaiah and Jaspal Subhlok, "Performance Estimation for Scheduling on Shared Networks," *9th Workshop on Job Scheduling Strategies for Parallel Processing*, 2003.

[2] Gerald Sabin, Rajkumar Kettimuthu, Arun Rajan and P. Sadayappan, "Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment," *9th Workshop on Job Scheduling Strategies for Parallel Processing*, 2003.

[3] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic Matching and Scheduling of a Class of Indenpendent Tasks onto Heterogeneous Computing Systems," Proc. of the 8th Heterogeneous Computing Workshop, pp. 30-44, April, 1999.

[4] O. H. Ibarra and C. E. Kim, "Heuristic Algorithm for Scheduling Independent Tasks on Nonidentical Processors," Journal of the ACM, vol. 24, no. 2, pp. 280-289, April, 1977.

[5] H. Barada, S. M. Sait, and N. Baig, "Task Matching and Scheduling in Heterogeneous Systems using Simulated Evolution," Proc. of the 15th Parallel and Distributed Processing Symposium, pp. 875-882, 2001.

[6] B. Hamidzadeh, Lau Ying Kit, and D.J. Lilja, "Dynamic Task Scheduling using Online Optimization," Journal of Parallel and Distributed Systems, vol. 11, pp. 1151-1163, 2000.

[7] H. Topcuoglu, S. Hariri, and Min-You Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 3, March 2002.

[8] 김학두, "이질적인 분산 컴퓨터 시스템을 위한 동적 스케쥴링 알고리즘," 석사학위논문, 서울시립대학교 컴퓨터·통계학과, 2003.