그리드 컴퓨팅 환경에서 다중 큐를 이용한 작업 스케줄링 기법

강창훈* 박기진** 김성수***
극동정보대학 방송영상미디어과*
아주대학교 공과대학 산업정보시스템공학부**
아주대학교 정보통신전문대학원
chkang@kdc.ac.kr* kiejin@ajou.ac.kr** sskim@ajou.ac.kr***

A Job Scheduling Mechanism Using Multi-queue In Grid Computing Environments

Changhoon Kang* Kiejin Park** Sungsoo Kim***
Dept. of Visual Broadcasting Media, Keukdong College*
Division of Industrial & Information Systems Engineering, Ajou University**
Graduate School of Information and Communication, Ajou University***

요 약

최근 지역적으로 분산되어 있는 이질적인 고성능 컴퓨팅 자원을 하나로 묶어 거대한 시스템을 구성하는 그리드 컴퓨팅(Grid computing)에 대한 연구가 활발하게 이루어짐에 따라, 본 논문에서는 그리드 시스템에서 클라이언트 작업들을 특성에 따라 분류하여 우선순위가 높은 작업은 예약이 가능한 큐로 분배하고 우선순위가 낮은 작업은 백필이 가능한 큐로 할당하는 다중 큐 스케줄링 기법을 제안하였고 다양한 실험을 통하여 제안된 기법들의 성능을 평가하였다. 그 결과 그리드 컴퓨팅 시스템의 이용률(Utilization)이 높아지고, 작업 지연시간(Slowdown)이 줄어드는 것을 확인하였다.

1. 서 론

최근 지역적으로 분산되어 있는 이질적인 고성능 컴퓨팅 자원을 하나로 묶어 거대한 시스템을 구성하는 그리드 컴퓨팅(Grid computing)에 대한 연구가 활발하게 이루어지고 있다. 그리드 컴퓨팅은 지리적으로 분산되어 있는 고성능 컴퓨터, 대용량 저장장치, 첨단과학설비 등 컴퓨팅 자원들을 네트워크로 연동하여 병렬 분산처리 환경을 제공하는 방법으로 Globus[1]와 같은 미들웨어를 사용하여 그리드 시스템을 손쉽고 빠르게 구축할 수 있다.

그리드 컴퓨팅을 위해서는 그리드로 연결된 유휴 자원들을 찾아내는 자원 검색 서비스, 할당된 작업들의 처리 순서를 결정하여 분산시키는 스케줄링 서비스, 시스템 안정을 위한 그리드 보안 서비스, 컴퓨팅 자원들을 사용할 때 발생하는 비용 처리를 위한 사용자 계정 서비스 등이 필요하다[2]. 이 서비스들 중 중요하게 대두되는 부분인 스케줄링 기능은 자원들의 분산 처리에 관한 관리와 작업 스케줄링 기법으로 구분할 수 있다. 현재 여러 가지 병렬컴퓨팅 스케줄링 알고리즘이 연구되어 왔으나[3], 대부분 작업의 계산량을 중심으로 다루어 지역적으로 분산된 그리드 시스템에 적합하지 않은 문제점을 가지고 있다. 이기종(Heterogeneous) 병렬 컴퓨팅 환경에 적합한 스케줄링 기법 중 빠른 응답 시간을 제 공하는 백필(Backfill) 기법을 기반으로 한 다양한 스케줄링 기법이 소개되었다[4].

본 논문에서는 그리드 시스템 상의 그리드 컴퓨팅 노드로 제출된 작업을 실행시키는데 필요한 프로세서 수와 예상 작업수행 시간의 특성에 따라 그리드 요청 작업들을 구분하여, 우선순위가 높은 작업은 예약이 가능한 작업 큐(Job Queue)로 분배하고 우선순위가 낮은 작업은 백필이 가능한 백필 큐(Backfill Queue)로 할당시킴으로써 시스템의 효율을 높이는 방법을 제안하였다. 제안된 기법들을 다양한 시뮬레이션 실험을 통하여 성능을 평가하였다. 본 논문의 2장에서는 관련 연구를 언급하고, 3장에서 백필 기반의 다중 큐 스케줄링 기법을 제시하였으며, 4장에서는 제안한 방법의 성능을 평가하고, 5장에는 결론을 내렸다.

2. 관련연구

그리드 작업 스케줄링은 수행해야 할 작업을 적절한 그리드 구성 노드로 분배하는 메타 스케줄링 과정과, 그리드 구성 각 노드 내에서 적절한 스케줄링 정책에 의해 작업을 수행하는 과정으로 나눌 수 있다. 메타 스케줄링은 Centralized 스케줄링, 계층적 스케줄링, De-centralized 스케줄링으로 나눌 수 있다. 현재 대부분의 연구가 여러 노드에 분배하는 메타 스케줄링 방식이다[5].

병렬 작업을 스케줄링하는 방법으로 Global 큐, Gang 스케줄링, Partitioning 방법이 있다. 이 중 가장 많이 사용하는 가변 분할(Variable Partitioning) 방법은 작업이 들어오는 순서대로 우선순위를 갖는 방식을 따르기 때문에 수행하고자 하는 작업 중에서 현재 시스템의 유휴 자원으로 작업이 수행 가능한 경우에도 우선 순위가 낮으면 수행되지 못하는 경우가 생긴다. 이 경우 작업은 수행되지 않고 자원만 남는 단편화(Fragmentation) 문제가 발생하게 되며 이로 인해 성능이 저하될 수 있다. 이와 같은 단편화 문제를 해결하기 위해 Dynamic Partitioning[6]이나

Gang 스케줄링 방법이 제시 되었는데 각 노드의 클럭(Clock) 동기화 문제나 통신상의 오버헤드 등 구현하는데 제한 점이 많아 적용에 어려움이 있다.

단편화 문제를 해결하기 위한 또 다른 방법으로 여백의 공간에 큐에 있는 작업 중 순서를 변경하여 먼저 수행 될 수 있도록 끼워 넣는 개념인 백필 방법이 제시되었다[4]. 백필 스케줄링은 Conservative 백필과 EASY 백필 방법으로 구분할 수 있으며 EASY 백필 방식을 사용하면 Conservative 백필 방식보다 더 많은 작업을 백필하여 수행시킬 수 있어 전체 시스템의 효율을 높일 수 있지만, 사용자에게 수행 시간을 약속하기가 어려운 무한대기(Unbounded Delay)로 인해 지연시간이 길어지면 시스템의 성능이 저하되는 문제가 발생한다.

본 논문에서는 각 백필 방법의 단점인 EASY 백필의 무한대기를 방지 하고 Conservative 백필의 시스템 효율을 높이기 위하여, 예약되는 작업의 수를 EASY 백필 보다 많이 늘리고 Conservative 백필 보다는 줄여서 두 가지 방법의 단점을 보안하여 시스템 전체의 효율을 높이고자 하였다.

3. 다중 큐 스케줄링 기법

그림 1은 De-Centralized 스케줄링을 사용하는 그리드 컴퓨팅 시스템의 구조를 보여준다. 각 노드는 그리드 컴퓨팅에 참여한 컴퓨터(Worker)와 스케줄러로 구성되어 있으며, 그리드 클라이언트들이 작업을 제출하면 스케줄러는 작업 스케줄링 정책에 의해 작업 큐나 백필 큐로 작업을 분배한다. 만약 해당 노드의 과부하로 인해 해당 노드에서 작업을 실행하기 어려울 경우 다른 노드로 작업을 보낸다. 작업을 보내는 메타 스케줄링 정책은 기존에 연구된 [7]의 방법을 사용하였다.

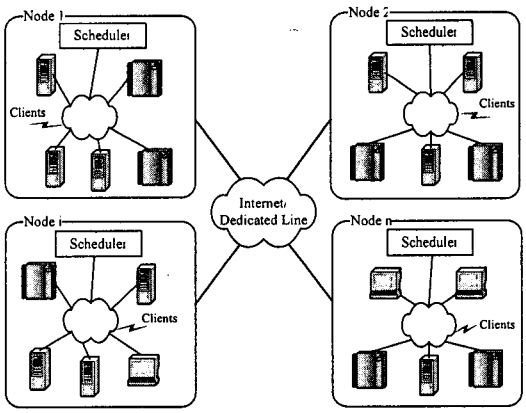


그림 1 그리드 시스템 구조도

그림 2는 본 논문에서 제시한 그리드 컴퓨팅 시스템의 한 노드에서의 다중 큐 스케줄러로 노드는 스케줄러와 그리드 컴퓨팅에 참여한 컴퓨터들로 구성된다. 스케줄러는 순서대로 실행되는 작업 큐와 백필을 하기 위해 기다리는 두 개의 백필 큐로 구성되어 있으며 클라이언트의 작업을 스케줄링 정책에 의해 작업 큐나 백필 큐에 저장하고, 다른 노드로 작업을 보낸다.
작업 큐와 백필 큐의 선택은 각 작업이 수행되기 위해서 필요한 프로세서 수와 해당 작업이 수행되는데 걸리는 시간을 기준으로 이루어진다. 작업 큐는 주로 많은 프로세서를 필요로 하는 작업들이 분배되며, 백필 큐는 주로 작은 수의 프로세서를 필요로 하는 작업들이 분배되어 백필하기에 용이한 작업들로 이루어 지게 된다.

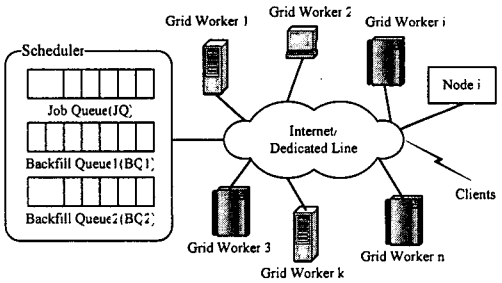


그림 2 Node i의 다중 큐 스케줄러

3.1 다중 큐 분배

작업 스케줄러는 작업번호 (i), 작업도착시간 (TA_i), 그리드 상의 노드 번호 (k) 등의 파라미터를 포함하여 식 (1)과 같이 작업 정보를 저장한다. 또한 실행에 필요한 프로세서의 수 (P_i) 와 작업을 실행시키는데 필요한 예측수행시간 (TES_i) 의 파라미터를 포함한다.

$$JOB_i(TA_i, TES_i, P_i, k) \tag{1}$$

작업을 분류하는 과정은 첫째, 작업들을 해당 노드의 전체 프로세서 수를 기준으로 세 개의 그룹으로 나누고 (식 (2)~(3)), 나누어진 각 그룹의 작업들을 예상 실행시간에 따라 두 개의 그룹으로 나눈다(식 (5)~(6)).

$$W(\text{Wide Job}) : \left\lfloor TP_i * \tfrac{2}{3} \right\rfloor \le P_i \le TP_i \tag{2}$$

$$M(\text{Medium Job}) : \left\lfloor TP_i * \tfrac{1}{3} \right\rfloor \le P_i < \left\lfloor TP_i * \tfrac{2}{3} \right\rfloor \tag{3}$$

$$N(\text{Narrow Job}) : 1 \le P_i < \left\lfloor TP_i * \tfrac{1}{3} \right\rfloor \tag{4}$$

$$L(\text{Long Job}) : TES_i \ge TMEAN_i \tag{5}$$

$$S(\text{Short Job}) : TES_i < TMEAN_i \tag{6}$$

두 번째 과정으로 평균 실행 시간을 기준으로 예상 실행 시간이 크면 긴 작업으로 그렇지 않으면 짧은 직업으로 분류한다. 평균 실행 시간은 실제 많은 작업을 실행하기 전까지는 평균 값이라 보기 어려울 수도 있지만, 많은 작업을 실행하면 실제적인 평균치에 가까워 지게 된다. 그림 3은 위와 같이 두 단계 후에 작업들을 실제 큐에 배치하기 위하여 우선순위에 맞게 배치한 형태로, 가장 왼쪽에 있는 작업들이 가장 우선 순위가 낮고 오른쪽으로 갈수록 우선순위가 높아지게 배치하였다.

NS (Narrow Short)	NL (Narrow Long)	ML (Medium Long)	MS (Medium Short)	WL (Wide Long)	WS (Wide Short)
<---------- 우선순위 ---------->					
낮음					높음

그림 3 분류된 작업의 우선순위

3.2 예약기법

무한대기를 방지하기 위하여 작업 큐에서 실행이 보장되는 작업의 수를 여러 개로 늘렸다. EASY 백필 방법은 예약되는 작업의 수가 큐의 처음에 있는 한 개의 작업만을 대상으로 하지만 본 논문에서는 예약 작업의 수를 여러 개로 늘려서 시스템의 효율성을 높이고 무한대기를 방지하였다. 현재 작업 큐에 있는 예약되는 작업의 수를 J_r, 작업의 수를 J_n, 예약 작업의 수를 조정하기 위한 임계치를 α 라 할 때 예약 작업의 수는 식 (7)에 의해 구하였다.

$$J_r = J_n * \alpha, \quad 0.1 \le \alpha \le 0.5 \tag{7}$$

4. 성능평가

본 논문에서 제안한 다중 큐 스케줄링 기법의 성능 측정을 위하여 시스템의 자원 효율과 작업의 지연 시간을 비교 분석하였다. 시스템 자원 효율과 작업의 평균 지연 시간은 식 (8), (9)와 같이 얻을 수 있다[8].

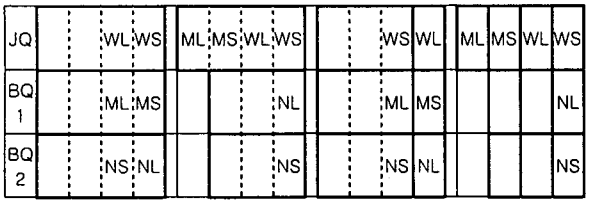
$$Utilization = \frac{Used\,\mathrm{Re}source}{Total\,\mathrm{Re}source} \times 100 \tag{8}$$

$$AverageSlowdown = \frac{\sum_{i=1}^{N} \frac{resp(i)}{width(i) \times \max(S, exe(i))}}{N} \tag{9}$$

시스템의 자원 효율은 작업을 수행하는데 사용된 자원을 전체 시스템 자원으로 나누어 식 (8)에 의해 구하였으며, 또한 작업의 평균 지연시간은 식 (9)와 같이 정의 하였다. 여기서 N은 전체 작업 수, S는 가장 수행이 짧은 작업의 길이를 나타낸다. resp(i)는 작업이 제출되어 끝날 때까지의 시간을 의미하며 exec(i)는 작업의 수행 시간을 나타낸다.

성능평가는 PC(Pentium4)에서 C언어를 사용하여 실험하였으며, EASY 백필, Conservative 백필, SJF(Shortest Job First), SJF(Shortest Job First), BJF(Big Job First) 방식들을 실험 비교하였으며 작업 부하로 Feitelson Archive[9]의 작업부하 로그부터 얻은 1) KTH의 IBM SP2 100노드의 Workload, 2) CTC 의 IBM SP2 512노드의 Workload, 3) 확률 함수를 사용하여 만들어진 RGD(Randomly Generated Data)를 입력 데이터로 사용하였다.

작업을 분배하는 방법을 그림 4와 같이 4가지 형태로 실험 하였다. 작업 큐(JQ)에 배치된 작업들은 순서대로 실행 가능한 시기에 실행되는 작업들이 배치되고, 백필 큐(BQ1, BQ2)에 있는 작업들은 프로세서 수에 의해 선택하여 실행 가능할 때에 백필되고 실행된다.



(a)방법 1 (b)방법 2 (c)방법 3 (d)방법 4

그림 4 작업의 큐 분배 방법

5. 결론

본 논문에서는 그리드 시스템에서 클라이언트 작업들을 특성에 따라 분류하여 우선순위가 높은 작업은 예약이 가능한 큐로 분배하고 우선순위가 낮은 작업은 백필이 가능한 큐로 할당하는 다중 큐 스케줄링 기법을 제안하였고 다양한 실험을 통하여 제안된 기법들의 성능을 평가하였다. 그 결과 그리드 컴퓨팅 시스템의 이용률이 높아지고, 작업 지연시간이 줄어드는 것을 확인하였다. 추후에는 효율적인 메타 스케줄링 정책의 연구와 복합적인 작업 스케줄링 정책의 연구가 필요하다.

참고문헌

[1] I. Foster, et al., " Globus: A Metacomputing Infrastructure Toolkit, " The International Journal of Supercomputer Applications and Performance Computing, Vol. 11, No. 2, pp. 115-128, Oct. 1997.

[2] K. Krauter, et al., " A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing, " Software Practice and Experience Journal, Vol. 32, No. 2, pp. 135-164, Feb. 2002.

[3] T. D. Braun, et al., " A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, " Journal of Parallel and Distributed Computing, Vol. 61, pp. 810-837, 2001.

[4] A. W. Mualem, et al., " Utilization, Predictability, Workloads and User Run time Estimates in Scheduling the IBM SP2 with Backfilling, " IEEE Trans. Parallel and Distributed System, Vol. 12, No. 6, pp. 529-543, June 2001.

[5] V. Subramani, et al., " Distributed Job Scheduling on Computational Grids Using Multiple Simultaneous Requests, " The 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11 2002), pp. 359-368, 23-26 July 2002.

[6] R. McCann, et al., " A Dynamic Processor Allocation Policy for Multiprogrammed Sharedmemory Multiprocessors, " ACM Trans. on Computer System, Vol. 11, No. 2, pp. 146-178, May 1993.

[7] Q. Wang, et al., " De-centralized Job Scheduling on Computational Grids Using Distributed Backfilling, " Grid and Cooperative Computing - GCC 2004: Third International Conference, pp. 285-292, Oct. 21-24, 2004, Proceedings. Lecture Notes in Computer Science 3251 Springer 2004.

[8] D. Zotkin, et al., " Job-Length Estimation and Performance in Backfilling Schedulers, " The 8th IEEE International Symposium on High Performance Distributed Computing (HPDC'99), 3-6 August, 1999.

[9] D. Feitelson, " Logs of Real Parallel Workloads from Production Systems, " http://www.cs.huji.ac.il/labs/parallel/workload/logs.html.