혼합 다중 큐를 사용한 그리드 컴퓨팅의 작업 스케줄링 기법

강창훈*, 최창열**, 박기진***,김성수**
* 극동정보대학 방송영상미디어과
** 아주대학교 정보통신전문대학원
*** 아주대학교 공과대학 산업정보시스템공학부
e-mail : chkang@kdc.ac.kr

A Job Scheduling Scheme of Grid Computing using Hybrid Multi-queue

Chang-Hoon Kang *, Chang-Yeol Choi**, Kie-Jin Park ***, Sung-Soo Kim **
* Dept. of Visual Broadcasting Media, Keukdong College
** Graduate School of Information and Communication, Ajou University
*** Division of Industrial & Information Systems Engineering, Ajou University

요 약

고성능 컴퓨팅 자원(Computing Resources)을 네트워크를 통하여 하나로 연동하여 단일 컴퓨팅 작업을 수행하는 그리드 컴퓨팅(Grid Computing)에 대한 연구가 최근 활성화 되고 있다. 이에 본 논문에서는 그리드를 구성하는 노드에 작업을 분배하는 메타 스케줄링(Metascheduling)과 노드내의 작업을 프로세서에 적절히 할당하는 작업 스케줄링 방법을, 작업 큐와 두 개의 백필 큐를 사용하는 혼합 기법을 사용하여, 작업 지연율(Slowdown Ratio)을 줄이고 그리드 컴퓨팅의 이용률(Utilization)을 높이는 방법을 제안하였으며 다양한 실험을 통하여 성능을 평가하였다.

1. 서론

그리드 컴퓨팅은 지리적으로 분산되어 있는 고성능 컴퓨터, 대용량 저장장치, 첨단과학설비 등 컴퓨팅 자원들을 네트워크로 연동하여 병렬 분산처리 환경을 제공하는 방법이다[1]. 현재 그리드 컴퓨팅 기술은 병렬 컴퓨팅 기술의 새로운 패러다임으로 평가를 받고 있어, 병렬 컴퓨팅 환경에서의 기존 병렬 작업 스케줄링 기법을 그리드 컴퓨팅 환경에 적용시키는 방법이 시도되고 있다[2]. 이기종(Heterogeneous) 병렬 컴퓨팅 환경에 적합한 스케줄링 기법 중 빠른 응답 시간을 제공하는 백필(Backfill) 기법을 기반으로 한 다양한 스케줄링 기법이 연구 되었으며[3], 백필을 사용한 스케줄링에서는 작업의 특성과 그 우선순위에 따라서, 시스템의 자원 효율을 높이면 작업의 지연시간이 늘어나고, 작업의 지연 시간을 줄이면 시스템의 자원 효율이 낮아지는 Trade-off 가 발생한다. 이에 따라 작업의 특성을 고려하여 우선순위를 조정함으로써, 전체 시스템의 자원 효율을 증가시키거나 작업의 지연시간을 단축시켜야 한다.

그리드 컴퓨팅 서비스들 중 중요하게 대두되는 부분인 스케줄링 기능은 자원들의 분산 처리에 관한 관리와 작업 스케줄링 기법으로 구분할 수 있다[4]. 현재 여러 가지 병렬컴퓨팅 스케줄링 알고리즘이 연구되어 왔으나, 대부분 작업의 계산량을 중심으로 다루었기 때문에, 지역적으로 분산된 그리드 시스템에 적합하지 않은 문제점을 가지고 있다. 또한 그리드를 구성하는 각 노드에 작업들을 분배하는 메타 스케줄링 방법들이 많이 연구되고 있으나 각 작업들을 각 노드에 분배 한 후에 각 노드 내에서 작업 스케줄링을 병행하여 처리하는 방법은 거의 연구되지 않고 있는 실

본 연구는 2005 년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행되었음." (KRF-2005-041-D00630)

정이다.

이에 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 제안하였으며, 다양한 시뮬레이션 실험을 통하여 성능을 평가하였다. 본 논문의 2 장에서는 관련 연구를 언급하고, 3 장에서 백필 기반의 혼합 다중 큐 스케줄링 기법을 제시하였으며, 4 장에서는 제안한 방법의 성능 평가를 수행하고, 5 장에는 결론을 내렸다.

2. 관련연구

그리드 작업 스케줄링은 수행되는 작업을 처리하는 관점에 따라 크게 두 가지로 볼 수 있다. 첫째는, 수행해야 할 작업을 적절한 그리드 구성 노드로 분배하는 메타 스케줄링 과정이고, 둘째는 그리드 구성 각 노드 내에서 적절한 스케줄링 정책에 의해 작업을 수행하는 과정으로 나눌 수 있다. 현재까지 대부분의 연구는 이 두 과정을 서로 다른 관점에서 다루었으나, [2]에서는 다중 큐를 이용하여 로컬(Local) 작업과 원거리(Remote) 작업을 서로 다른 큐에 배치/예약하여 그리드 컴퓨팅 환경에 적합한 방법을 연구하였다.

그리드 상의 각 노드에서 수행되는 작업은 일괄처리 작업(Batch Job), 상호대화형 작업(Interactive Job), 병렬 작업(Parallel Job)으로 구분될 수 있다[5]. 일괄처리 작업이나 상호대화형 작업은 한 노드에서 수행되기 때문에 현재 부하가 가장 적은 노드에서 수행시키거나 부하가 한 노드에 집중되는 경우 수행중인 작업을 다른 노드로 옮겨주는 방식으로 스케줄링하여 수행시킨다. 병렬 작업은 동시에 여러 프로세서를 필요로 하며, 각 노드에서 수행중인 프로세스들이 동시에 수행되지 않으면 프로세스들간의 통신에 지연이 생겨 전체 작업 수행시간이 길어질 수 있다[6].

병렬 작업을 스케줄링하는 방식은 공간을 공유하는 공간공유(Space-sharing) 방식과 시간을 공유하는 시간공유(Time-sharing) 방식으로 나눌 수 있다. 공간공유 방식은 작업이 시작하여 수행이 끝날 때까지 프로세서가 할당되는 방식이고, 시간공유 방식은 작업을 수행할 때 프로세서를 시간을 기준으로 여러 부분으로 나누고, 각 부분을 작업에 할당하는 방식이다. 시간공유 방식의 대표적인 방법으로 Global 큐와 Gang 스케줄링이 있고, 공간공유 방식의 대표적인 방법으로 Partitioning 이 있다.

병렬 작업을 스케줄링 하는 방법 중 가장 많이 사용하는 가변 분할(Variable Partitioning) 방법은 수행하고자 하는 작업의 수행시간(Time)을 x 축으로 하고 필요로 하는 자원 공간(Space)을 y 축으로 하여 작업을 직사각형 모양으로 표현하여 작업을 할당해주는 방식이다. 이 과정에서 사각형을 제외한 여백 부분이 많아지면 시스템의 자원 효율이 떨어지게 된다. 이 방식은 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 이나 Gang 스케줄링 방법이 제시 되었는데 각 노드의 클럭(Clock) 동기화 문제나 통신상의 오버헤드 등 구현하는데 제한 점이 많아 거의 사용되지 않는다.

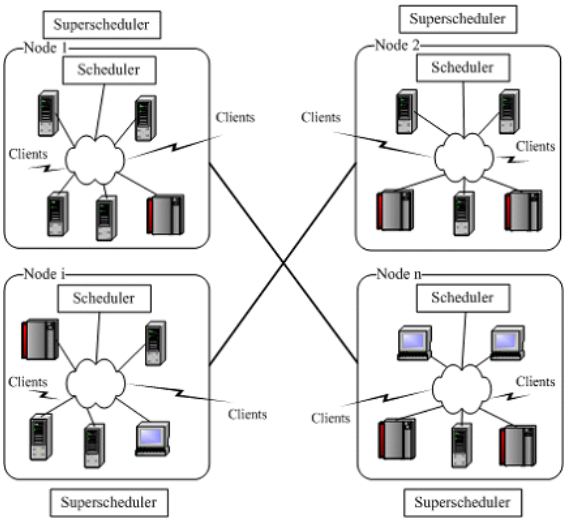
단편화 문제를 해결하기 위한 또 다른 방법으로 여백의 공간에 큐에 있는 작업 중 순서를 변경하여 먼저 수행 될 수 있도록 끼워 넣는 개념인 백필 방법이 제시되었으며, 본 기법은 현재 유휴 자원으로 수행이 가능하지만 우선 순위가 낮아 수행되지 못하는 작업을 자신보다 우선순위가 높은 작업을 지연시키지 않는 경우 먼저 수행시키는 스케줄링 방법이다[3]. 백필 스케줄링을 기반으로 하여 시스템의 성능을 높이고자 여러 방법이 연구되었는데, 수행 시간이 짧은 작업의 우선순위를 높게 하여 작업의 지연 시간을 줄이는 방법[7], 작업의 우선 순위를 작업이 제출(Submit)되어 기다린 시간으로 두어 작업의 지연 시간을 줄이는 방법[8], 우선순위가 낮은 작업을 백필시켜 수행하고자 할 때 지연시간이 짧은 경우에는 지연을 허용함으로써 시스템 자원 효율을 높인 방법 등이 있다. 그러나 기존의 많은 연구들은 시스템의 자원 효율과 작업의 지연 시간 사이의 Trade-off 를 고려하지 않고 한쪽에만 초점을 맞추었기 때문에 시스템의 자원 효율을 높이면 작업의 지연 시간이 늘어나고 작업의 지연 시간을 줄이면 시스템의 자원 효율이 낮아진다는 단점을 가지고 있다.

3. 혼합 다중 큐 스케줄링 구조

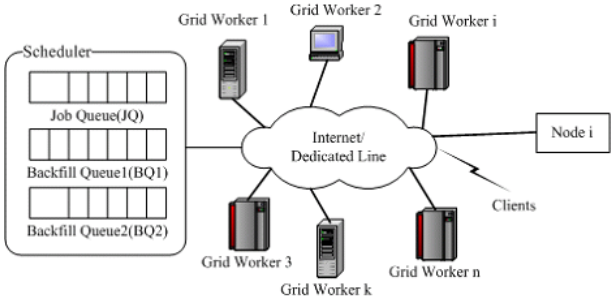
그림 1 은 혼합 스케줄링을 사용하는 그리드 컴퓨팅 시스템의 구조를 보여준다. 각 노드는 그리드 컴퓨팅에 참여한 컴퓨터(Worker)와 지역 스케줄러 및 전역 스케줄러(Superscheduler)로 구성되어 있다. 따라서 Hybrid 스케줄링 구조란 노드 내의 스케줄링 부하를 최소화하기 위한 계층적 스케줄링 전략과 노드간 협업을 위한 전역 스케줄러간 상호작용을 지원하기 위한 De-Centralized 스케줄링 전략을 동시에 고려하기 위한 그리드 시스템 구조이다. 따라서, 이는 전통적인 계층적 스케줄링 전략의 지역/전역 작업 스케줄링이 가능하다는 장점과 De-Centrailzed 스케줄링 전략의 통신 병목현상을 제거하고 SPOF(single point of Failure)가 없으며, 확장성이 좋다는 장점을 모두 활용하기 위해 채택한 혼합 구조이다. 그러므로 혼합 구조를 적용한 시스템은 공동으로 컴퓨팅에 참여하는 컴퓨터들의 논리적인 하나의 그룹으로 인터넷을 통하여 연결되며 마치 하나의 컴퓨터 시스템처럼 동작한다. 다시 말해 혼합 스케줄링 구조는 메타 스케줄링과 지역 작업 스케줄링을 병행하여 전체 시스템의 효율성 및 가용성을 증가시킬 수 있는 구조이다.

각 노드는 스케줄러와 그리드 컴퓨팅에 참여한 컴

퓨터들로 구성된다. 스케줄러는 순서에 의해 차례대로 실행되는 작업 큐와 백필을 하기 위해 기다리는 두 개의 백필 큐로 구성된다. 스케줄러는 노드 내의 모든 컴퓨터들의 상태와 작업들의 상태를 파악하고 관리하며, 인터넷을 통하여 제출되는 클라이언트의 작업을 스케줄링 정책에 의해 작업 큐나 백필 큐에 저장하고, 다른 노드로 작업을 보내거나 다른 노드로부터 발송된 원거리 작업을 원거리 작업 스케줄링 정책에 의해 두 개의 백필 큐에 분배한다.



(그림 1) 혼합 그리드 시스템 구조도



(그림 2) 노드 i 의 하이브리드 큐 스케줄러

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.

$$JOB_i(TA_i, TES_i, P_i, k) \quad (1)$$

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

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

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

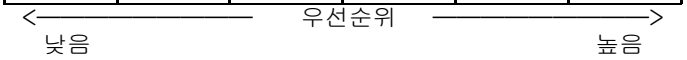
$$N(Narrow\ Job): 1 \le P_i < \left\lfloor TP_i * \tfrac{1}{3} \right\rfloor \quad (4)$$

$$L(Long\ Job): TES_i \ge TMEAN_i \quad (5)$$

$$S(Short\ Job): TES_i < TMEAN_i \quad (6)$$

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 3 은 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)



(그림 3) 작업 분류 및 우선순위

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.

$$J_r = J_n * \alpha, \quad 0.1 \le \alpha \le 0.5 \quad (7)$$

백필 큐의 예약 방법으로 백필 큐에 있는 작업들은 실행되기에 필요한 프로세서수가 확보되지 않으면 오랫동안 실행되지 못하고 무한대기에 빠질 수 있다. 백필 큐에서 백필 기회를 기록하여 백필되지 못한 수치가 일정 값보다 커질 경우 이 작업을 작업 큐로 보내어 일정 시간 후에 실행을 보장 받도록 하여 무한 대기를 방지하였다. 백필 큐에서 백필 가능 여부를 검사할 때 해당 작업이 백필의 기회가 부여되었으나 백필되지 못한 횟수를 J_m 라 하고, 백필 횟수의 최대 임계치를 β 라 할 때, 식 8 에 의해 검사되어 작업 큐로 옮겨지게 되어 예약 된다.

$$J_m \ge \beta \quad (8)$$

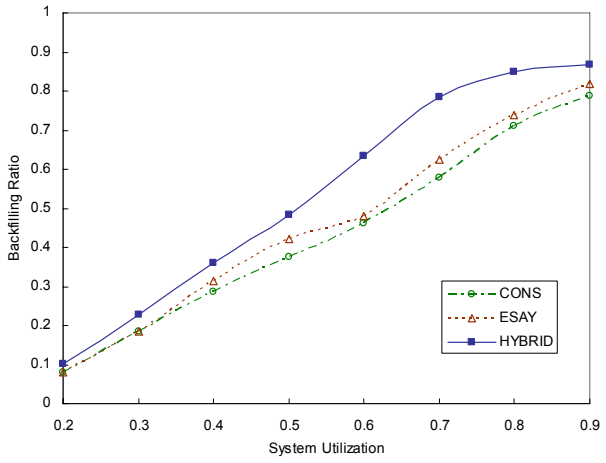
4. 성능평가

본 논문에서 제안한 혼합 다중 큐를 사용한 스케줄링 기법의 성능 측정을 위하여 백필 비율(Backfilling Ratio) 조정을 통해 시스템의 자원 효율성을 분석하고 작업의 평균 지연율(Slowdown Ratio)를 통해 다중 큐와 원격 작업 처리로 인해 얻을 수 있는 성능 개선 효과에 대해 분석하였다. 또한 스케줄링 전략에 따라 발생할 수 있는 작업 지연 효과를 평균 응답 시간(Average Response Time)으로 비교·분석하였다. 작업의 평균 응답 시간은 식 9 와 같이 얻을 수 있다[7]. 여기서 N 은 전체 작업 수, S 는 가장 수행이 짧은 작업의 길이를 나타내며, *resp*(*i*)는 작업이 제출되어 끝날 때까지의 시간을 의미하고, *exec*(*i*)는 작업의 수행 시간을 나타낸다. 또한 *width*(*i*)는 작업이 실행되기 위해 필요한 프로세서 수를 의미한다.

성능 평가를 위한 실험 데이터 생성을 위해 본 논문에서는 평균 작업 도착 시간(Mean Arrival Time), 평균 작업 요청 시간(Mean Estimated Execution Time), 평균 필요 프로세서 수(Mean Width, Mean Number of Processor) 등을 실험을 위해 필요한 파라미터 값을 Feitelson Archive[9]의 작업부하 로그(Workload logs)의 집합으로부터 추출하였고, 해당 파라미터를 기본값으로 하는 확률 함수를 근간으로 만들어진 RGD(Randomly Generated Data)를 입력 데이터로 사용하였다.

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

$$SlowdownRatio = \frac{AverageSlowdown_1 - AverageSlowdown_m}{\min(AverageSlowdown_1 - AverageSlowdown_m)} \quad (10)$$



(그림 4) 시스템 부하량에 따른 백필 비율

그림 4 는 그리드 전체 시스템의 작업 부하량(시스템 이용률)에 따라 백필되는 작업 비율을 Conservative 백필 방식과 Easy 백필 방식과 비교한 결과이다. 시스템 이용률($\rho_{mean}=\lambda_{mean}/\mu_{mean}$)이 증가함은 단위 시간당 새로 생성된 작업 요청이 빈번하게 발생함을 뜻하며, 노드별 작업 도착 시간과 처리 시간을 평균을 통해 전체 시스템 이용률을 계산한 것이다. 전체적으로 Conservative 백필 방식보다Easy 백필 방식이 백필되는 횟수가 많으며, Easy 백필 방식보다 혼합 다중 큐 스케줄링 방식이 백필 횟수가 많다. 이는 혼합 다중 큐 스케줄링 방식이 전체 그리드 시스템의 프로세서 사용률을 향상시켜 보다 자원 관리가 효율적으로 이뤄짐을 보여준다. 또한 시스템 부하량이 증가할수록 백필 가능성이 높은 작업 생성 가능성이 높아지므로 3 방식 모두 백필 비율이 증가함을 볼 수 있다.

5. 결 론

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 정책 연구를 수행할 예정이다

참고문헌

[1] I. Foster, et al., "Grid Services for Distributed System Integration," Computer, Vol. 35, No. 6, pp. 37-46, 2002.
[2] B. Lawson and E. Smirni, "Multiple-queue Backfilling Scheduling with Priorities and Reservations for Parallel Systems," The 8th International Workshop, JSSPP 2002 Edinburgh, Scotland, UK, pp. 72-87, July 24, 2002.
[3] A. Mualem and D. Feitelson, "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.
[4] H. Shan, et al., "Job Superscheduler Architecture and Performance in Computational Grid Environments," In SC2003 Conference, 2003.
[5] B. Bode, et al., "The Portable Batch Scheduler and the MauiScheduler on Linux Clusters," in Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, Georgia, Oct. 2000.
[6] D. G. Feitelson, et al., "Theory and Practice in Parallel Job Scheduling," Job Scheduling Strategies for Parallel Processing, IPPS'97 Workshop, Geneva, Switzerland, pp. 1-34, Apr. 5, 1997, Proceedings. Lecture Notes in Computer Science 1291 Springer 1997.
[7] 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.
[8] S. Srinivasan, et al., "Characterization of Backfilling Strategies for Parallel Jobs Scheduling," 31st International Conference on Parallel Processing Workshops (ICPP 2002 Workshops), pp. 514-522, 20-23 Aug. 2002.
[9] D. Feitelson, "Logs of Real Parallel Workloads from Production Systems," http://www.cs.huji.ac.il/labs/parallel/workload/logs.html.