그리드 컴퓨팅의 최악 대기시간 개선을 위한 이중 큐 작업선택 기법

남궁미정[0] 박기진
안양대학교 문리과학대학 컴퓨터학과
mjmj201@anyang.ac.kr[0] kiejin@aycc.anyang.ac.kr

A Double-queue Job Selection Mechanism for Improving Worst-case Waiting Time in Grid Computing

Mijung Namgung[0] Kiejin Park
Department of Computer Engineering, Anyang University

요 약

본 논문에서는 그리드 컴퓨팅 시스템 성능 향상을 위해 기존의 스케쥴링의 단점인 최악대기시간 문제를 보완한, 작업(Job) 크기에 따른 이중 큐 작업선택 기법을 제안한다. 제안된 기법은 작업의 크기에 따라 이들을 서로 다른 큐에 넣은 후, 라운드 로빈(Round-Robin) 방식으로 작업을 처리함으로써, 그리드 컴퓨팅 시스템 최악 대기시간 성능을 개선시켰다.

1. 서론

그리드 컴퓨팅은 지리학적으로 분산되어 있는 고성능 컴퓨팅 자원을 네트워크로 상호 연동하여 조직과 지역에 관계없이 사용할 수 있는 환경을 말하는 새로운 컴퓨팅 모델로 차세대 인터넷 서비스를 말한다. 네트워크의 고속화로 인하여 분산되어 있는 고성능 컴퓨팅 자원을 네트워크로 상호 연동하여 막대한 양의 컴퓨팅 파워를 얻기위해 많은 단체들에서 활발한 연구를 하고 있다. 최근에 정부, 학계 및 산업계에서도 이를 기반으로 슈퍼컴퓨터나 클러스터 컴퓨터들을 연결하여 막대한 양의 컴퓨팅 파워를 얻는데 초점을 두고 있다.

그리드 컴퓨팅은 사용되지 않는 PC 를 지속적으로 네트워크에 연결하여, 다른 사람들이 필요로 하는 PC 작업에 활용할 수 있으며, 다수의 서버를 하나의 시스템으로 묶어, 복잡한 연산 작업을 여러 서버에서 분산 처리하는 방식으로 되어 있다[1]. 또한, 그리드 컴퓨팅은 컴퓨팅 자원들에서 수행되는 수많은 작업을 관리를 해야하고 네트워크로 상호 연동되는 대화식 시스템이기 때문에, CPU 이용률과 처리율의 극대화와 반환, 대기, 응답 시간의 극소화를 위해 효율적인 스케쥴링이 필수적이며, 스케쥴링 기법에 따라 시스템 성능이 많은 영향을 받게된다.

마스터(Master)와 워커(Worker)의 시스템 구조 모델하에 스케쥴링이 집중형(Centralized)과 비집중형(Decentralized)으로 나눌 수 있으며, 집중형 스케쥴링은 중앙에 있는 마스터가 워커의 가용 시스템의 상태 정보를 관리하므로써 작업 스케쥴링 하는 형태로, 개념적으로는 매우 효율적인 스케쥴링이나 마스터가 워커들의 상태 정보를 수집할시에 병목현상이 발생한다는 단점이 있다. 그에 반해, 비집중형 스케쥴링은 마스터가 없이 워커들끼리 서로 작업과 시스템 정보를 교환하면서 이루어지는 스케쥴링 형태로, 병목현상이 없으나 모든 작업과 시스템 정보를 실시간으로 요청을 받아야 한다는 단점이 있다. 집중형 스케쥴링으로는 Backfill 기법이, 비집중형 스케쥴링으로는 선입선처리(First Come First Serve: FCFS)가 시스템 성능향상에 적합하다[3].

선입선처리 기법은 짧은 프로세스나 I/O 이동 중심 프로세스들에게 불리하고 평균 대기시간이 늦어지는 단점이 있으며, Backfill 기법은 대규모 작업이 지연되지 않는다는 전제하에서 작은 크기의 작업들을 보장해주기 때문에 선입선처리 기법보다 효율적이다. 그러나 " 대규모 작업이 지연되지 않는다" 는 가정이 현실적이지 못하므로, 본 연구에서는 기존 단일 큐에서 발생하는 최악 대기시간

본 연구는 한국과학재단 목적기초연구(R05-2003-000-10345-0)지원으로 수행되었음.

문제점을 개선하기 위해서 이중 큐 작업선택을 기법을 채택하였다.

본 논문의 2 장에서는 관련 연구에 대해 기술하였고, 3 장에서는 이중 큐의 라운드 로빈 스케쥴링 기법을 제시한다. 4 장에서는 제안한 방법을 시뮬레이션을 통해서 성능 평가를 수행하고, 5 장에서는 제안한 기법에 대한 결론 및 차후 연구에 대해 다룬다.

2. 관련 연구

작업을 배치(batch) 처리 할 경우, 비배치 처리에 비해 보다 많은 수요를 충족시키고, 응답시간이 줄어 시스템 성능이 향상되지만[2], 배치 스케쥴링 기법은 선입선처리 기법을 기반으로 사용된 스케쥴러로, 최악의 대기시간이 늘어난다는 단점을 가지고 있다. Backfill 기법을 적용하면 선입선처리 기법에 비해 평균 응답시간이 보다 빨라지고, 이용률도 향상되나, 대규모 작업이 지연되지 않는다는 가정이 필수적이다[3]. 작업 크기와 다양한 작업부하로 인하여, 병렬 작업 스케쥴링 시스템(Parallel job scheduling system)을 이용할 수 있으나, 단일 큐를 사용함으로, 크기가 큰 작업과 작은 작업의 대기시간을 공평하게 고려하지 않기 때문에, 시분할에서는 미약하다[4].

본 논문에서는 작업의 크기와 이중 큐(Queue), 라운드-로빈(Round-Robin)을 이용해 선입선처리 및 Backfill 기법의 단점을 보안하여 최악 대기시간을 향상 시키고자 하며, 선입선처리, Backfill 기법 및 본 논문에서 제안한 이중 큐 작업선택 기법 간의 대기시간 성능을 시뮬레이션을 통해 비교하였다.

3. 이중 큐 작업선택 기법(Double-Queue Job Selection Mechanism)

선입선처리 기법은 작업 요청 순서대로 처리되기 때문에 대규모 작업들이 먼저 요청되어 처리될 경우, 나중에 들어온 크기가 작은 작업들은 대기시간이 커지게 되는 상황이 발생한다(그림 1 참조). 반면에 Backfill기법은 작은 작업부터 우선적으로 수행함으로써 작은 작업들의 최악 대기시간 문제는 해결되나, 대규모 작업들은 계속 기다리게 되는 문제점(Starvation)이 있다. 이를 해결하기 위해 작업 큐를 두 개 사용함으로써, 하나의 큐에는 크기가 큰 작업들을 배치하고, 나머지 큐에는 크기가 작은 작업들을 할당하였다.

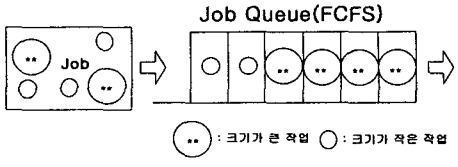

그림 1. 단일 큐에서 크기가 작은 작업의 대기시간이 커지는 예

큰 작업이 할당되는 큐를 LQ라고 하고, 작은 작업이 할당되는 큐를 SQ라고 하며, LQ와 SQ는 선입선처리 방식으로 작업을 수행한다(그림 2 참조).

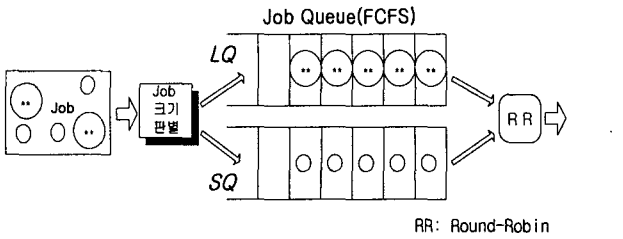

그림 2. 이중 큐와 라운드 로빈 기법을 이용한 작업선택 기법

작업들은 작업 크기와 도착시간 파라미터에 의해 구분되며, 도착한 작업들을 이중 큐에 분배시키기 위해, 자신의 작업 크기와 평균 작업 크기를 비교하여, 자신의 작업 크기가 평균 작업 크기보다 클 때는 LQ로, 작을 때는 SQ로 할당한다. 한편, 크기가 큰 작업과 작은 작업을 공평하게 실행시키기 위해서, 이중 큐에 할당된 작업들은 라운드 로빈 기법을 이용하여 처리되며, 시간 할당량만큼 작업들이 수행된다. LQ의 큰 작업을 실행할 경우, 라운드 로빈 시간 할당량에 맞게 작업을 분할 시키나, SQ에서는 작업의 크기가 크지 않기 때문에 작업이 분리되지 않는다. 라운드 로빈으로 인한 작업 분할을 최소화시키기 위해서, 라운드 로빈의 할당량(time quantum)을 작업 크기 판별 기준과 동일하게 하였다(그림 3 참조).

```
For(i=0; i<NumofJob;i++)                          //작업 크기 판별
  If(JobNum[i]<=MeanJobSize) SQ[n]=JobNum[i];
  Else SQ[n]=JobNum[i];

For(i=0;i<NumofJob: i++)
  If(Turn==Small){                          //작업 턴이 SQ일 때
     For(j=0; j<MeanJobSize; ){
        If(FinishTime<=SQ[n].ArrivalTime) WaitingTime=0;
     //작업완료시간이나 후에 작업이 요청될 경우: 작업대기시간=0
       Else WaitingTime=FinishTime-ArrivalTime;
                    //작업대기시간=작업완료시간-작업도착시간
     j=j+SQ[n].JobSize;
     }
   Turn=Large;                              //턴을 LQ로 넘겨 줌
  }
  Else{
   If(LQ[m].JobSize> MeanJobSize) LQ[m].JobSize= LQ[m].JobSize-
      Quantum;              //큰 작업을 퀀텀에 맞게 작업 분할
   Else WaitingTime=FinishTime-(LQp[m].ArrivalTime+Quantum);
              //작업대기시간=작업완료시간-(작업도착시간+퀀텀)
   Turn=Small;                              //턴을 SQ로 넘겨 줌
  }
```

그림 3. 이중 큐 작업선택 기법의 구현 예

4. 성능 평가

단일 큐와 이중 큐의 성능 평가를 위해, 선입선처리, Backfill 및 이중 큐 작업선택 기법을 작업 수에 따라 평균 대기시간과 최악 대기시간을 비교하였으며, 작업 크기의 비율에 따라 최악 대기시간을 측정하였다. 4700 초를 본 실험의 평균 작업 크기(작업 판별 기준, 라운드 로빈 시간 할당량)으로 사용하였고, 작업 간의 평균 도착 시간 간격은 950 초로 설정하였다[5]. 그리드 시스템에서 작업 요청 시간 간격은 서로 독립적이기 때문에 지수 분포를 사용하여, 시뮬레이션을 수행하였다.

그림 4 에서 작업 수에 따른 평균 대기시간을 보면, 이중 큐 작업선택 기법이 선입선처리의 평균 대기시간보다 단축되나, Backfill 기법의 평균 대기시간보다는 지연된다는 것을 볼 수 있다. 이를 통해 크기가 작은 작업을 우선적으로 처리하는 전략이 평균 대기시간 측면에서는 바람직하다는 것을 파악했다.

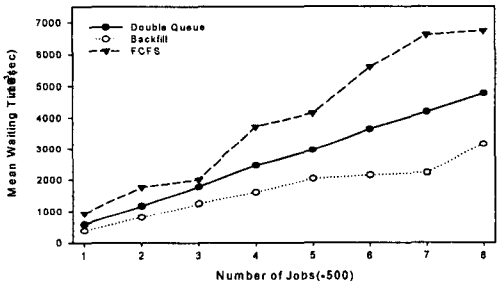

그림 4. 작업 수에 따른 각 기법의 평균 대기시간

그림 5 를 보면 이중 큐 작업선택 기법이 다른 기법보다 최악의 대기시간에서는 우수함을 보였다. 이중 큐 작업선택 기법은 선입선처리에 비해 9.47(*10^5s)만큼 줄어들었고, Backfill 보다는 30.07(*10^5s)만큼 최악 대기시간이 단축된 것을 볼 수 있다. 이는 이중 큐를 사용함으로써, 크기가 큰 작업과 작은 작업을 동등하게 수행시켰기 때문이라 판단된다.

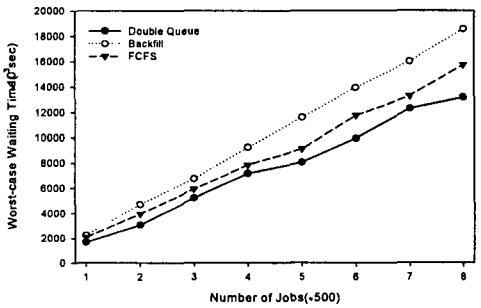

그림 5. 작업 수에 따른 각 기법의 최악 대기시간

또한, 그림 6 에서 작업 수가 2000 일 때, 큰 크기의 작업과 작은 크기 작업의 비율이 3:7 이하에서는 선입선처리 기법의 최악 대기시간이 짧으나, 그 이후부터는 이중 큐 작업선택 기법의 최악 대기시간이 단축됨을 볼 수 있다. 이는, 큰 작업의 수가 많을수록 이중 큐 작업선택 기법이 유리하다는 것을 의미한다.

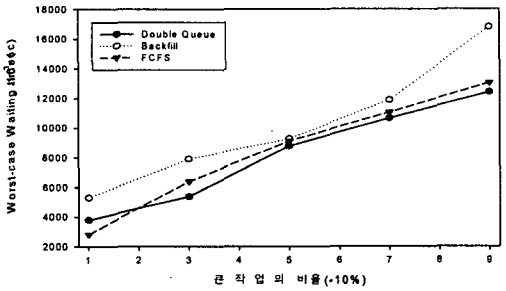

그림 6. 작업 크기의 비율에 따른 최악 대기시간

5. 결론

선입선처리와 Backfill 에서 사용하던, 단일 큐로 작업을 처리하는 경우보다, 본 논문에서 제안한 이중 큐 작업선택 기법을 이용하여 처리할 때, 최악 대기시간이 단축되며, 크기가 큰 작업이 증가 할수록 성능이 향상되는 것을 알 수 있다. 따라서, 최악의 대기시간에 있어, 집중형이나 비집중형 시스템 구조에 관계없이 모든 시스템 구조에 이중 큐 작업선택 기법을 이용한 스케쥴링이 우수함을 파악했다. 차후에는 본 논문에서 제안한 이중 큐 작업선택 기법을 시뮬레이션과 수식 모델을 결합하여 연구할 계획이다.

참고문헌

[1] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid : Enabling Scalable Virtual Organization," International J. Supercomputer Applications, Vol. 1, No. 3, 2001.

[2] Bruno Volckaert, et. al., " Evaluation of Grid Scheduling Strategies through a Networkaware Grid Simulator," teverschijnen in Proceedings van PDPTA 2003, June 2003.

[3] Volker Hamscher, et. al., "Evaluation of Job-Scheduling Strategies for Grid Computing," GRID 2000, PP. 191-203, 2000.

[4] H. Franke et.al., " An Evaluation of Parallel Job Scheduling for ASCI Blue-Pacific," In Proceedings of SC99, Portland, OR, November 1999. IBM Research Report RC21559.

[5] A.Streit. " On Job Scheduling for HPC-Clusters and the dynP Scheduler," In Proceedings of the 8th International Conference on High Performance Computing(HiPC 2001), Vol. 2228 of LNCS, PP 58-67. Springer, Dec. 2001.