# 그리드(Grid)기반 웹 서비스(Web Service)의 응답 시간 향상을 위한 스케쥴링 기법

박 미 선[0] 박 기 진
안양대학교 문리과학대학 컴퓨터학과
oooooo82@anyang.ac.kr kiejin@aycc.anyang.ac.kr

# A Scheduling Algorithm for Improving Response Time in Grid-based Web Service

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

요 약

웹 서비스의 사용이 급격히 증가함에 따라 사용자의 웹 서비스 QoS(Quality of Services)에 대한 관심도 높아지고 있다. 본 논문에서는 QoS 요소들 중 응답 시간(Response time) 향상에 대한 스케쥴링 기법을 제안하여, 서비스의 질을 향상시키고자 한다. 구체적으로는 기존의 그리드 시스템에 사용되고 있는 단일 배치 스케쥴링 기법을 기본으로 한 다중 배치 스케쥴링을 연구하였으며, 이를 통해 그리드 기반 웹 서비스의 응답 시간 성능이 개선됨을 확인하였다.

## 1. 서 론

최근 차세대 인터넷 서비스로 각광받는 그리드(Grid)는 지리적으로 분산된 컴퓨팅 자원들을 네트워크로 연동하여 공유하는 방법의 컴퓨팅 환경을 말한다. 현재 웹 서비스를 그리드 컴퓨팅과 결합시키는 방법들이 소개되면서, 그리드와 웹 서비는 별개의 분리된 문제가 아닌 서로 밀접한 연관성을 가지고 있는 분야로 인식되고 있다.

인터넷 사용이 보편화되면서 사용자들은 좀더 나은 서비스를 받기를 원하고 있으나, 그에 반해 인터넷 사용의 급증으로 인한 네트워크의 대역폭이 상대적으로 좁아지면서 많은 양의 트래픽이 발생하여 인터넷 서비스 제공 속도는 점차 느려지고 있다. 이런 상황에서 웹 서비스에서의 응답 시간은 가장 중요한 요소로 대두되고 있다. 현재 인터넷은 점점 비상업적인 컨텐츠들은 사라지고, 상업적인 컨텐츠들이 상업적인 컨텐츠들이 급격하게 증가하는 상황에서 서비스의 응답 시간은 점점 더 중요해지고 있다. 웹 서비스를 기반으로 한 전자상거래 사이트 예를 들어보면, 사이트에서 사용자는 클릭 한 페이지가 8 초 이내에 로드되지 않을 경우 사이트를 빠져 나가게 되며, 특히 여성을 상대로 하는 사이트에 있어서는 '8 초 법칙'은 무시할 수 없는 대원칙으로 작용될 만큼 응답 시간은 중요하다 [1]. 또한 '얼마나 많은 작업(Job)을 처리할 수 있는가?'에 대한 시스템 성능(단위 시간당 최대 처리 가능한 건수) 또한 중요한 문제다.

본 논문에서는 선입선출(First Come First Serve: FCFS) 기법을 사용하는 배치(Batch) 스케쥴링의 응답 시간 지연 문제를 해결하기 위해서 현재의 그리드 시스템에 사용되고 있는 단일 배치 스케쥴링을 개선 한 다중 배치 스케쥴링 기법에 관하여 논하였다. 본 논문의 2 장에서는 관련 연구를 언급하고, 3 장에서 다중 배치 스케쥴링 기법을 제시하였으며, 4 장에서는 제안한 방법의 성능 평가를 수행하고, 5 장에는 결론을 내렸다.

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

## 2. 관련 연구

그리드 시스템에서의 배치 스케쥴링은 일정 시점까지 작업들을 모아두었다가 일괄해서 처리하는 방식으로 다른 스케쥴링 방법들에 비해 높은 수요를 충족시킬 수 있어 시스템 성능이 높다는 장점이 있는데 반해, 작업대기 시간(Waiting time)이 길어진다는 단점이 있으며[2], 대부분의 그리드 관련 스케쥴링 알고리즘은 List 스케쥴링을 기본으로 하고 있다.

3 가지 대표적인 스케쥴링 기법으로는 선입선출, 랜덤 기법(Random), Backfill 기법이 있으며, 선입선출 기법은 대기 시간이 길어지기 때문에 비효율적이며, 랜덤 기법은 아직 시작되지 않은 작업들에 대해서 랜덤하게 선택하는 방법으로 공정성이 보장되는 장점이 있다. 마지막으로 Backfill 기법은 선입선출 기법에서 순서를 생각하지 않는 방법으로 큰 작업으로 인한 필요 없는 휴지 시간(Idle time)을 예방하는 장점이 있다. 이 기법들은 작업 환경에 따라 사용 용도가 결정되는데, 단일 사이트(Single-site)와 이기종(heterogeneous)환경에서는 Backfill 기법이 가장 응답 시간이 빠르며, Job-pool 환경에서는 선입선출 기법이, 계층적인 스케쥴링을 원할 때는 Random 기법이 유리하다[3]. 한편 병렬(Parallel) 시스템 상에서의 작업에 대한 스케쥴링 기법으로 그리드 스케쥴러와 작업에 대한 스케쥴링 기법을 통합한 Central 그리드 스케쥴링의 경우 사용자와 제공자 자원에 대한 Fail-safety 나 Acceptance 와 수행에 있어 결점을 가지고 있다[4].

본 연구에서는 기존 연구에서 고려하지 않은 이기종 환경인 그리드에 선입선출(FCFS) 기법을 기본으로 하고 있는 배치 스케쥴링 기법을 사용하되, 기존 배치 기법의 응답시간이 느려진다는 단점을 보완한, 그리드 기반의 웹 서비스에서의 다중 배치 스케쥴링 기법을 제안하였다.

## 3. 다중 배치 스케쥴링

본 연구에서 제안하는 다중 배치 스케쥴링은 단일 배치를 여러 개 결합하여 사용하는 아이디어이다. 웹 서비스를 요청하는 클라이언트들의 수에 따라 배치 스케쥴링의 배치 개수가 변하게 되는데, 여기서 배치 개수는 기존에는 배치 개수가 1 개였던 것에 반해, 다중 배치 스케쥴링은 배치 개수가 여러 개 존재하는 것을 의미한다. 배치의 개수를 늘림으로써 큰 작업들로 인한 작은 작업들의 지연 시간을 최소화하고자 한다.

### 3.1 다중 배치 알고리즘

단일 배치 알고리즘의 경우 요청된 작업들을 일정 시점까지 모아두었다가 처리하는 방식인데, 다중 배치 알고리즘은 그 임의의 배치 작업이 모두 끝나기 전이라도, 다른 배치를 수행할 수 있는 방식으로 아래의 순서를 따른다(그림 1 참조).

1. 각각의 클라이언트들의 요청들은 작업의 크기와 도착 시간 정보를 갖고 준비 큐에 들어오게 된다.
2. 준비 큐에 있는 작업은 가용한 첫 번째 배치에 순차적으로 할당된다.
3. 임의의 배치에 배치 사이즈(n)만큼 작업이 채워지면, 그 배치는 실행된다.
4. 배치가 실행되는 동안 도착하는 작업들은 다음 배치들에 순차적으로 작업이 할당된다.
5. 배치 사이즈만큼 채워진 배치들은 각각의 배치에서 처리해야 하는 총 작업량을 계산하여, 작업량이 작은 배치가 먼저 실행된다.

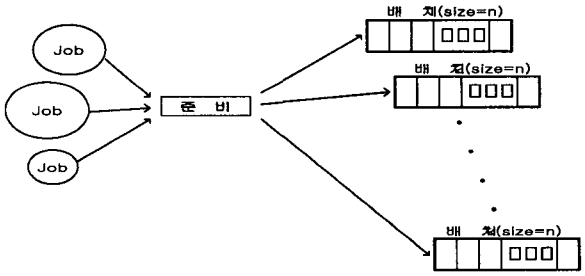


그림 1 다중 배치 스케쥴링 구조

다중 배치 스케쥴링의 배치 개수는 클라이언트들의 요청한 작업들의 수와 배치 사이즈와의 관계를 고려하여 식(1)에 의해 계산하였다[5].

Number_of_batch= total_number_of_jobs / batch_size □ (1)

다중 배치에서의 작업 할당 및 작업 실행 우선 순위 결정에 대한 알고리즘은 구현은 아래와 같다.

```
For(i=0; i<Batch_num; i++)                    //각 배치에 작업 할당
  For(j=Size_sum; j<Size_sum+Batch_Size; j++){
    If(i==Batch_Size*i)
        Wait_time[i]=0;
    Else
        Wait_time[i]=Arrival_time[(Batch_num*Batch_size)+
```

```
        Batch_size-1]-Arrival_time[i];
    Response_time[i]=Wait_time[i]+Job_size[i];
  }
for(k=1; k<Batch_size); k++)          // 우선 순위에 의해 정렬
  for(p=k+1; p<Batch_size); p++)
    if(Batch_sum[k] > Batch_sum[p])
        Batch_sum[k]=Batch_sum[p];
For(i=0; i<Batch_num; i++)            //배치 실행
  For(j=Size_sum; j<Size_sum+Batch_Size; j++)
    Sum[i]=Response_time_sum- Wait_time[i]+Job_size[i];
```

## 4. 성능 평가

그리드 요청 작업 크기와 작업 도착 시간 간격은 다양한 파라미터에 대한 알고리즘의 성능 비교를 위해 지수 분포를 사용하였으며, 그리드 작업 크기와 작업 도착 시간 간격의 중간 값들은 기존 논문에서 사용된 파라미터를 적용하였다. 평균 작업 크기는 $\left[0,\frac{T}{2}\right]$의 중간 값으로 여기서 T는 대역폭을 의미하며, 작업 도착 시간 간격은 그리드 평균 작업의 30%를 넘지 못한다[2].

표 1 시스템 파라미터 (단위:second)

| | |
|---|---|
| 그리드 작업 크기 | Exp(25) |
| 작업 도착 시간 간격 | Exp(평균 작업 크기 * 0.3) |
| 네트워크 대역폭 | 100Mbps |

그림 3 에서는 단일 배치와 다중 배치의 배치 사이즈에 따른 평균 응답 시간의 변화를 표시하고 있으며, 배치 사이즈에 무관하게 다중 배치가 우수함을 나타낸다. 이는 작은 크기의 배치 작업들에 대해 우선 순위를 부여함으로써 작은 작업의 응답 시간은 물론 전체 작업의 평균 응답 시간도 향상시킬 수 있었던 것으로 판단 된다.

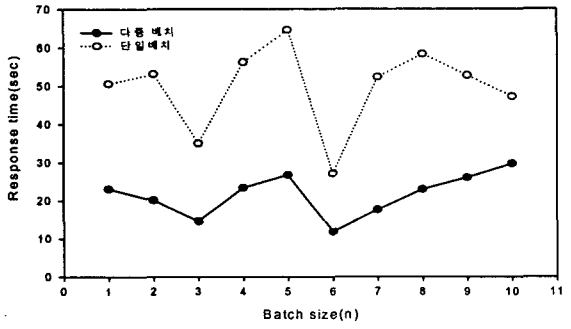


그림 3 배치 사이즈(n)에 따른 평균 응답 시간 비교

그림 4 에서는 클라이언트가 증가할수록 두 스케쥴링 기법의 응답 시간의 차이가 현저히 크게 나타나며, 다중 배치 스케쥴링의 경우 단일 배치 스케쥴링보다 더 많은 작업을 처리하는 것을 볼 수 있다. 이는 기존 단일 배치의 큰 작업으로 인한 지연 문제를 우선 순위 다중 배치를 사용하여, 작은 작업을 우선적으로 처리했기 때문이라 판단된다.

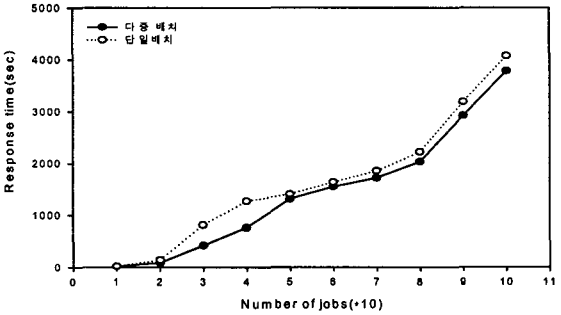


그림 4 작업 수에 따른 응답 시간 비교

## 5. 결론 및 향후 연구

본 연구에서는 기존 단일 배치 스케쥴링 기법을 기본으로 한 다중 배치 스케쥴링 기법을 이용하여, 기존 단일 배치 스케쥴링 기법의 장점은 극대화시키고, 단점은 보완하여, 그리드 기반 웹 서비스의 응답 시간과 시스템 성능을 향상시켰다. 추후에는 큰 배치에 대한 기아현상(Starvation) 문제의 연구 및 웹 서비스의 응답 시간을 향상시키기 위해서 배치 기법 이외의 새로운 스케쥴링 기법을 연구할 계획이다.

### 참고문헌

[1] http://www-903.ibm.com/kr/techinfo/websphere/download/1.html

[2] Bruno Volckaert, et. al., "Evaluation of Grid Scheduling Strategies through a Network-aware Grid Simulator," te verschijnen in Proceedings van PDPTA 2003, June 2003.

[3] Volker Hamscher, et. al., "Evaluation of Job Scheduling Strategies for Grid Computing," the 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000) at the 7th International Conference on High Performance Computing (HiPC - 2000), LNCS 1971, pp. 191-203, 2000.

[4] Carsten Ernemann, et. al., "On advantages of Grid Computing for parallel Job Scheduling," 2nd IEEE International Symposium on Cluster Computing and the Grid (CC-GRID 2002), Berlin, Germany, IEEE ComputerSociety Press, pp. 39-47, 2002.

[5] http://java.sun.com/products/jndi/tutorial/ldap/search/batch.html