이질적 다중서버 시스템에서의 공정큐잉 기반 스케줄링

박경호° 김지혜 이주현 박창윤* 민상렬
서울대학교 컴퓨터공학부 중앙대학교 컴퓨터공학부*
{kalynda°, kimjh, jhlee, symin}@archi.snu.ac.kr, cypark@cau.ac.kr*

Scheduling based on Fair Queueing in Heterogeneous Multiple Server Systems

Kyeongho Park° Ji Hye Kim Ju Hyun Lee Chang Yun Park* Sang Lyul Min
School of Computer Science and Engineering, Seoul National University
School of Computer Science and Engineering, Chung-Ang University*

요 약

하나의 컴퓨터 시스템은 여러 종류의 이질적인 서버들로 구성된 것으로 간주할 수 있다. 본 논문에서는 이질적 다중서버 시스템에서의 프로세스 스케줄링 기법에 대해 연구한다. 스케줄링에서는 서비스의 공정성과 시스템 성능의 두 가지 측면이 고려되어야 한다. 여기에서는 공정성을 위하여 기존의 단일 서버 시스템에서 사용되던 공정 큐잉에 기반한 스케줄링 기법을 제안한다. 또한 각 서버간의 이용률 차이를 감안한 시스템 전체 성능 향상을 위한 방법에 대해서도 고찰한다.

1. 서 론

일반적으로 컴퓨터 시스템은 이질적 서버들로 구성된 복합 시스템으로 간주할 수 있다. 예를 들어 모든 시스템은 CPU 작업을 담당하는 CPU 서버와 IO 작업을 담당하는 IO 서버를 최소 하나씩 가지고 있다고 볼 수 있다. 이 때 각 프로세스들은 CPU 요구와 IO 요구가 혼합되어 연속으로 들어오는 일종의 스트림으로 표현된다.

프로세스의 스케줄링시에는 공정성(fairness)을 제공해 주어야 한다. 이는 각 프로세스에게 보장된 서비스를 해 주어야 함을 의미하며, 보장된 서비스를 위해서는 각 프로세스에게 가중치(weight)로 정의되는 최소한의 서비스율을 제공해야 한다. 그러나 이질적 서버들로 구성된 시스템의 경우, 기존의 단일서버 시스템에서의 공정 큐잉(fair queueing) 알고리즘을 그대로 사용할 수 없다는 한계가 있다.

또한 스케줄링은 시스템 전체의 성능(performance)의 향상을 도모해야 한다. 성능 향상의 척도로는 시스템의 자원 이용률을 들 수 있다. 이질적 다중 서버로 구성된 시스템에서 작업들이 서버간을 이동하며 서비스를 받는 경우 한 서버의 스케줄링은 다른 서버들에 영향을 줄 수 있으므로 스케줄링의 순서를 적절히 변경하여 시스템의 전체 성능을 높일 수도 있다. 예를 들어 CPU 위주의 작업이 많은 경우 IO 위주의 작업을 선호하여 스케줄링하면 결과적으로 IO 서버의 이용률이 높아지고 전체적 시스템 성능이 향상될 수 있다.

본 논문에서는 이질적 다중 서버로 구성된 시스템을 위한 공정 서비스 기법을 제안하고, 성능 향상을 위한 추가적인 방법을 고찰해 본다.

2장에서 스케줄링과 관련된 기존의 연구 동향을 살펴보고, 3장에서는 본 논문에서 제안하는 공정 큐잉 기반의 다중 서버 스케줄링 방법에 대해 설명한다. 4장에서는 이 모델을 기반으로 한 시뮬레이션 결과를 보이며, 5장에서는 요약 및 추후 연구방향에 대해 토론한다.

2. 관련 연구

기존의 단일 서버에서의 스케줄링 기법에 대해서는 네트워킹과 프로세스 스케줄링 분야에서 많은 연구가 있어왔다. GPS(Generalized Processor Sharing)[1] 모델은 공정한 서비스를 제공하기 위한 이상적 모델로서, 각 프로세스들은 보장된 서비스를 받을 수 있으며 악의적인 서비스들과 무관하게 타 서비스들에 독립적인 서비스를 받는다. 그러나 GPS 방법의 경우 구현상의 어려움이 있기 때문에 이를 실제 시스템에서 사용할 수 있도록 근사한 PGPS[1], WFQ[2], SCFQ[3], WF^2Q[4], SFQ[5] 등의 연구가 있었다. 이들은 작업보전(work-conserving)의 특성을 지니며, 모두 단일 서버 시스템의 동질적 요구에 대해 보장된 서비스와 공정성을 제공한다.

동질적 다중서버 시스템에서의 스케줄링에 관한 연구도 찾아볼 수 있는데, 대표적인 것이 SFS[6]이다. 이 방법은 다중프로세서 시스템에서 쓰레드를 스케줄링할 때 실제 요구된 가중치들을 가능한(feasible) 가중치로 재조정하고 각 프로세서를 독립적인 GPS 서버로 모델링하여 스케줄링한다. 그러나 본 논문에서 대상으로 하는 이질적 다중서버 모델의 스케줄링에 대해서는 연구가 그리 많지 않았다.

3. 다중서버 시스템에서의 스케줄링

공정 큐잉 기반의 스케줄링 알고리즘에서 프로세스의 진행은 가상 시간(virtual time)으로 결정이 된다. 각 프로세스마다 서비스를 받은 뒤의 가상 시간은 l^j가 받은 서비스 양, r이 프로세스의 예약 대역폭일 때, 아래의 수식과 같이 정의된다.

$$virtual\_finish\_time^j = virtual\_start\_time^j + l^j / r$$

단, 예약 대역폭 r은 서버의 용량을 1로 가정했을 때의 가중치이다.

제안하는 스케줄링 알고리즘에서는 동질적 다중 서버 시스템에서와 마찬가지로 이질적 다중 서버 시스템을 위한 유체 흐름(fluid flow) 모델을 제시하고 이에 기반한 현실적 모델을 제안한다.

이질적 다중 서버 시스템에서의 유체 흐름 모델은 그림 1과 같이 단일 서버 시스템에서 제시된 유체 흐름 모델을 각 서버에 적용하여, 공정성 보장의 기반을 마련한다. 또한, 한 프로세스가 각 서버에서 받은 서비스의 양을 하나의 가상 시간으로 표현하여, 전체 시스템 차원에서 스케줄링을 위한 정보로 사용한다. 가상 시간은 상속되며, 프로세스가 서버를 바꿀 때마다 이전 서버에서의 가상 종료 시간이 다음 서버에서의 가상 시작 시간이 된다.

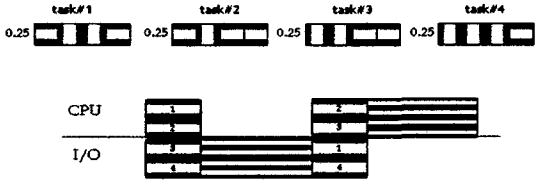

그림 1 다중 서버 시스템의 유체 흐름 모델 예

3.1 기초모델

위의 방법은 이상적 모델이므로 현실에 적용하기에는 문제가 있다. 여기에서는 단일 서버에서 사용되던 SFQ 방법을 다중 서버의 각 서버에 개별적으로 적용하였다. 알고리즘은 다음과 같다.

- Process virtual time
  - When a process is created:
    - $virtual\_start\_time_1 = server\_virtual\_time$
  - After receiving a jth unit of service, with size l
    - $virtual\_finish\_time_j = virtual\_start\_time_j + l/r * virtual\_capacity$
    - $virtual\_start\_time_{j+1} = virtual\_finish\_time_j$, where $virtual\_capacity = the\_sum\_of\_busy\_process\_rat * server\_utilization$
  - Then enter run-queue again.
- Server virtual time
  - 0 when a server is initialized and when all server is idle.
  - When a server is busy:
    - $server\_virtual\_time = virtual\_start\_time\_of\_a\_served\_process$
  - When a new process enters an idle server:
    - $server\_virtual\_time = virtual\_start\_time\_of\_the\_new\_process$
- Scheduling
  - Each server schedules a process that has the smallest virtual start time value from its run-queue.

그림 2는 기초모델 스케줄의 예를 보여준다.

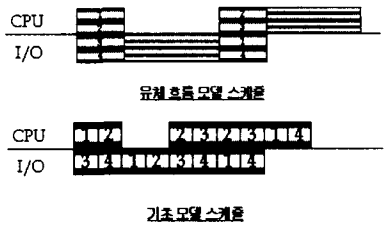

그림 2 기초 모델 스케줄 예

3.2 용량 정규화 모델

기초모델 스케줄은 각 서버가 균등하게 이용될 경우 유체흐름 모델에 근사하게 나오지만, 불균등하게 이용될 경우 부하가 적은 서버에서 온 프로세스를 기아상태에 두는 문제가 있다. 이는 서버마다 가상 시간의 증가율이 다르기 때문이다. 이 문제를 해결하기 위하여 각 서버의 용량이 동적인 작업량의 양에 비례하여 증가하도록 하는 모델이 용량 정규화 모델이며, 다음과 같은 식을 적용하여 각 서버간의 가상시간 증가를 동기화한다.

$virtual\_finish\_time^j = virtual\_start\_time^j + l^j / r * \Sigma_{i \in Bs} r_i$

단 r_i는 프로세스 i의 예약 대역폭, B_s는 서버 S의 프로세스 집합이다. 그림 3에 이 모델을 적용한 경우의 스케줄링 예가 나와 있다.

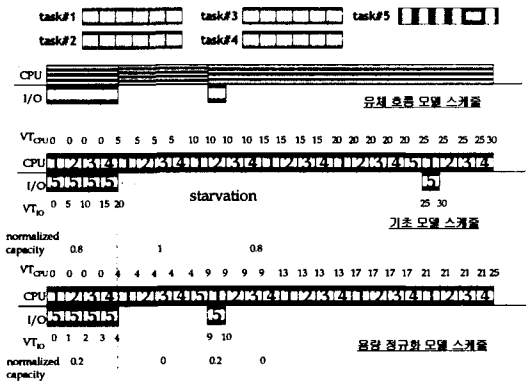

그림 3 용량 정규화 모델 예

3.3 이용률 감지 용량 정규화 모델

시스템 내의 각 서버들의 이용률에 차이가 많을 경우, 이용률이 낮은 서버를 많이 이용하는 프로세스에 우선순위를 줌으로써 전체 시스템의 성능을 향상시킬 수 있다. 이용률 감지 용량 정규화 모델은 서버의 이용률을 가중치로 반영하는 모델이다. 이용률이 낮은 서버에서 받은 프로세스의 가상 시간의 증가 속도를 늦추어 이용률이 높은 서버로 갔을 때 그 프로세스를 선호하게 한다. 이 경우 다음과 같은 식으로 가상 시간을 계산하게 된다.

$virtual\_finish\_time^j = virtual\_start\_time^j + l^j/r * U_s * \Sigma_{i \in Bs} r_i$

단 r_i는 프로세스 i의 예약 대역폭, B_s는 서버 S의 프로세스 집합, U_s는 서버 S의 이용률이다. 그림 4에 이 모델을 적용한 경우의 스케줄링 예가 나와 있다. 이 경우 서비스율이 다소 불공정하게 나타나는 대신에 시스템 전체의 성능은 향상된다.

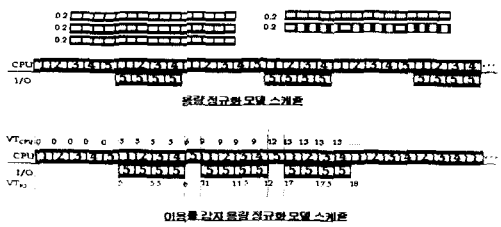

그림 4 이용률 감지 용량 정규화 모델 스케줄 예

4. 시뮬레이션 결과

본 논문에서 제안한 방법들의 성능을 평가하기 위하여 시뮬레이션을 통해 각 프로세스의 서비스율과 서버 이용률을 측정하였다. 서비스율은 프로세스 입장에서 보아 예약 대역폭 값에 비례하여 나타나는 것이 좋으며, 서버 이용률은 높을수록 좋다. 여기에서는 유체 흐름 모델, 기초 모델, 용량 정규화 모델, 이용률 감지 용량 정규화 모델과 사용량 부식(Decay-Usage) 스케줄러[7]에 대하여 실험하였다. 사용량 부식 스케줄러는 IO 위주의 작업에 높은 CPU 우선순위를 주는 방법이다. 이용률 감지 정규화 모델의 경우 exponential averaging을 통해 과거의 정보를 0.9, 현재의 정보를 0.1 정도로 반영하였다.

그림 5는 한 예로 두 개의 CPU 위주의 프로세스와 한 개의 IO 위주의 프로세스를 스케줄링했을 때 각 서버별 서비스율을 보인 것이다. 각 프로세스의 CPU:IO의 비율은 10:0, 9:1, 3:7로 가정하였다. 또한 표 1은 이 때의 서버 이용률이다.

표 1 서버 이용률

	GPS	기초	용량정규	이용률감지	Decay
CPU	1.0000	1.0000	1.0000	1.0000	1.0000
I/O	0.4693	0.3636	0.4823	0.6025	0.7254

용량정규화 모델의 경우 GPS 모델과 유사한 수치를 보여줌을 알 수 있다. 그리고 이용률 감지 용량 정규화 모델의 경우 공정성을 어느 정도 희생하여 전체 시스템 성능을 높일 수 있다. 사용량 부식 방법의 경우 이 실험에서는 가장 좋은 성능을 보이지만 프로세스들의 특성을 다르게 하여 실험한 경우 이용률의 변화가 심하였다.

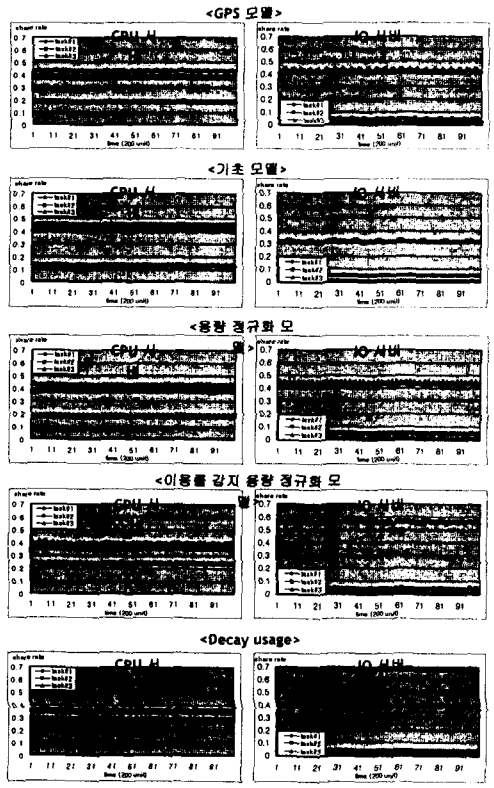

그림 5 시뮬레이션 결과

5. 결 론

본 논문에서는 이질적 서버들로 구성된 다중 서버 시스템에서 프로세스 스케줄링을 하기 위한 기법에 대하여 고찰하였다. 공정성을 부여하기 위하여 단일 서버에서 사용되는 유체 흐름 모델을 다중 서버 시스템으로 확장하여 적용하였다. 한편, 이용률이 낮은 서버를 많이 사용하는 프로세스에 우선순위를 부여함으로써 시스템의 전체적 성능향상이 있을 수 있음을 보였다.

추후 위에서 제시한 기법들의 정확한 분석 및 좀더 일반화된 모델의 제시를 위한 연구가 진행될 것이다.

[참고문헌]

[1] A.K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," IEEE/ACM Trans. on Networking, vol. 1, no. 3, pp. 344-357, Jun 1993.

[2] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," SIGCOMM'89, pp. 1-12, 1989.

[3] S.J. Golestani, "A Self-Clocked Fair Queueing Scheme for High Speed Applications," INFOCOM'94, pp. 636-646, 1994.

[4] J. C. R. Bennett and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms," SIGCOMM '96, pp. 143-156, 1996.

[5] P. Goyal, H.M. Vin, and H. Cheng, "Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," IEEE/ACM Trans. on Networking, vol. 5, no. 5, pp. 690-704, Oct 1997.

[6] A. Chandra, M. Adler, P. Goyal, and P. Shenoy, "Surplus Fair Queueing: A Proportional-Share CPU Scheduling Algorithm for Symmetric Multiprocessors," Symposium on Operating Systems Design and Implementation(OSDI) 2000.

[7] J. L. Hellerstein, "Achieving Service Rate Objectives with Decay Usage Scheduling," IEEE Trans. on Software Engineering, vol. 19, no. 8, pp. 813-825, Aug 1993.