이질적 다중서버 시스템에서 공정한 잔여용량 공유기법

이주현° 박경호 김강희 민상렬
서울대학교 컴퓨터공학부
{jhlee°, kalynda, khkim}@archi.snu.ac.kr symin@dandelion.snu.ac.kr

A Fair Extra Capacity Sharing Scheme for Heterogeneous Multi-Server Systems

Ju Hyun Lee° Kyeongho Park Kanhee Kim and Sang Lyul Min
School of Computer Science and Engineering
Seoul National University

요 약

단일서버 시스템에서 응용간 자원의 공정한 분배는 GPS(Generalized Processor Sharing)을 통해 실현될 수 있다. 그러나 다중서버시스템에서 GPS를 적용한다면, 각 서버 내에서의 공정성은 보장해 주지만, 시스템 관점에서의 공정성은 더 이상 보장되지 않는다. 본 논문에서는 멀티서버 시스템에서 시스템 관점에서의 공정성 보장을 위한 기법을 제시한다. 이 기법은 시스템에서 발생하는 각 서버의 잔여용량을 응용간 공정하게 분배하는 모델과 모델을 참조하여 실행하는 실제 스케줄링 알고리즘으로 구성된다. 모델에서는 긴 시간동안 각 서버에서 발생하는 잔여용량의 사용을 관측하여, 향후 각 응용의 요청들에게 제공할 잔여용량 분배를 결정한다. 스케줄링 알고리즘은 모델에 의해 결정된 잔여용량 분배를 실제 각 응용들이 제대로 가져갈 수 있도록 제어한다. 실제 모델을 참조하여 공정 잔여용량 분배를 수행하는 스케줄링 알고리즘의 동작은 시뮬레이션을 통해 검증하였다.

1. 서 론

시스템에서 공정한 자원분배는 각 응용에게 예측 가능한 성능을 제공한다. 단일 서버시스템에서 공정한 자원분배는 Generalized Processor Sharing(GPS)을 기반으로 한다. 실제 GPS를 구현하는 알고리즘으로 Packetized generalized processor sharing(PGPS)[1]와 weighted fair queueing(WFQ)[2]이 제안되었다. 위 알고리즘의 복잡도와 성능을 개선시키기 위해 start-time *fair queueing*(SFQ)[3]과 *worst-case fair weighted fair* queueing(WF^2Q)[4] 역시 제안되었다. 한편 GPS를 기반으로 멀티서버 시스템을 위한 몇몇 공정서비스 알고리즘도 제안 되었다. Surplus fair scheduling(SFS)[5]과 multi-server fair queueing(MSFQ)[6]은 각각 다중처리기 시스템과 다중링크 환경에서 공정서비스 알고리즘으로 제안되었다. 그러나, SFS와 MSFQ는 동질적 다중서버 시스템만을 고려하고 있다. 예외적으로, Move-To-Rear List Scheduling(MTR-LS)[7]는 이질적 다중서버 시스템을 다루고 있지만, 이 알고리즘에서는 각 응용이 각 응용이 가지는 예약 용량의 보장의 관점만 고려했을 뿐, 실제 서버 잔여용량을 어떻게 응용 간 공정하게 서비스 할지는 고려하지 않고 있다.

위에 언급한 알고리즘들은 GPS를 기반으로 공정서비스를 제공한다. 그러나 이러한 알고리즘을 멀티서버 시스템으로 확장시, 불공정 서비스를 받을 수 있는 경우가 발생한다. GPS를 기반으로 하는 알고리즘에서는 서버에서 발생하는 서버 잔여용량을 현재 그 서버에서 서비스 중인 응용에게만 분배한다. 이런 방법을 통해 잔여자원을 분배한다면, 어떤 응용이 항상 많은 응용들이 서비스되는 서버에만 도착한다면, 적은 응용들이 서비스되는 서버에 도착하는 응용에 비해 항상 적은 잔여 용량을 서비스받게 되어 불공정 서비스가 된다.

본 논문에서는 이러한 문제점을 개선하고자, 서버 잔여용량을 지능적으로 분배하는 새로운 잔여용량 분배 기법을 제시한다. 이 기법에서는 각각의 서버가 긴 기간 동안 자신의 서버에서 얼마나 많은 잔여서비스용량이 발생하는 지를 관측하고, 이를 통하여 현재 서버에 존재하는 응용들이 어느 정도의 서버 잔여용량을 할당받아야 되는지를 결정한다. 이렇게 잔여 용량을 각 응용이 가지는 가중치에 따라 골고루 분배함으로써 공정서비스를 실현한다.

본 논문의 구조는 다음과 같다. 2장에서는 공정한 잔여용량를 제공하기 위한 모델을 제시한다. 3장에서는 이 모델을 실현하기 위한 구체적인 스케줄링 알고리즘을 설명한다. 4장에서는 제안된 알고리즘을 시뮬레이션을 통해 검증하고 5장에서 결론을 맺는다.

2. 잔여용량 분배모델

본 논문에서는 다음과 같은 기호를 정의한다. 시스템은 다수의 이질적 서버 S_j로 구성되어 있다. 각 서버는 용량 C_j를 가지며, 작업이 존재 시 항상 동작하는 작업보존(work conserving) 속성을 갖는다. 본 논문에서는 각 서

버의 용량을 1로 가정한다. 서버 j에서 각 응용 Ai는 요청 $R_i^j(k)$, k=1,2,...,을 생성하며, 가중치 w_i^j를 갖는다. 각 응용은 서버 j로부터 다음과 같은 예약된 용량 $r_i^j = C_j W_i^j$를 받을 자격을 가지며, 서버 용량 중 현재 서버에 대기 혹은 서비스 중인 응용들의 예약된 용량들의 합을 제외한 용량이 잔여서비스 용량이 된다.

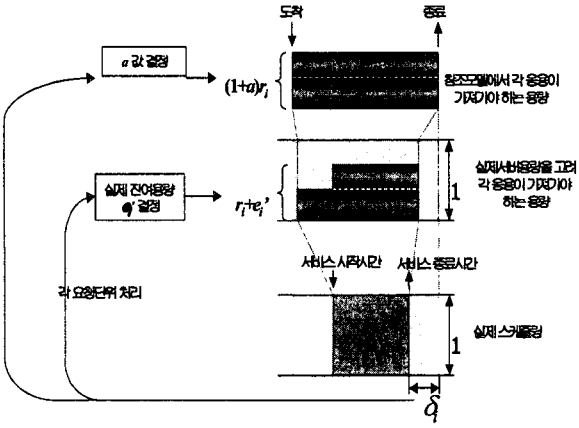

그림 1. 잔여용량 분배 모델

그림 1은 제안된 잔여용량 분배모델을 3개의 파이프 형태로 표현하고 있다. 첫 번째 파이프는 각 응용이 서버에서 가져가야 하는 예약용량 r_i^j과 잔여용량 $a_j \cdot r_i^j$을 나타낸다. 이 용량들을 참조하여 응용의 각 요청이 이상적으로 끝나야 하는 시간을 결정할 수 있다. 여기에서 예약용량은 각 응용이 시스템에 처음 도착 시 요청하는 용량이며, 잔여용량은 그 서버에서 발생하는 잔여용량을 긴 시간 동안 관측하여 얻은 값이 된다. 그러나 서버의 실제 용량은 1이기 때문에 이를 고려해 각 응용이 가져가야 하는 용량을 두 번째 파이프가 나타내고 있다. 각 응용은 현재까지 첫 번째 파이프가 제시하는 이상적인 서비스 용량에 비해 얼마나 많이 혹은 적게 서비스를 받았는지를 나타내는 매개변수 Δ를 유지하는 데, 이 값을 기준으로 실제 서버에서 받을 잔여용량을 결정한다. 실제 서버에서의 잔여용량 e'i이 결정되면, 이 잔여용량과 그 응용의 예약용량을 기반으로 스케줄러는 그 요청의 종료시간을 계산하게 되고, 요청 중 종료시간이 가장 적은 것을 선택하여 서비스하게 된다. 한 응용 요청의 서비스가 종료되었을 때, 그 요청의 이상적인 종료시간과 실제 종료시간과의 차이 δi가 구해지며, 이 차이는 Δ를 갱신하는 데 사용된다. 즉 스케줄러는 첫 번째 파이프를 통해 이 서버에서 어느 정도의 잔여용량이 발생하는 지를 파악 후, 이 잔여 용량을 응용들에게 응용의 예약용량에 비례하게 분배한다.

3. 알고리즘

이 장에서는 a_i의 유도방법, 잔여용량의 구체적인 분배, 그리고 실제 스케줄링 순서 결정방법에 대해 설명한다.

3.1 a_i 값 결정

본 논문에서는 서버 j에서 응용 i가 참조모델을 참고하여 이상적으로 받아야 되는 서비스 양과 실제 받은 서비스 양의 차이를 Δ_i^j 로 표시하였다. 실제 스케줄러는 참조모델을 그대로 따라가야 하므로 결국에는 $\Delta_i^j \approx 0$ 이 되도록 a_i 값이 정해져야 한다.

정리 1. 값 a_i는 서버 j에서 잔여용량으로 서비스된 작업량과 예약용량으로 서비스된 작업용량의 비이다.

자세한 증명은 [8]을 참조하기 바란다.

3.2 실제 잔여용량 분배

서버 j에서 응용 i가 이상적으로 받아야 하는 서비스 양과 실제 받은 서비스 양의 차이인 Δ_i^j를 줄이는 방향으로 서버의 잔여용량 분배가 이루어져야 한다. 본 논문에서는 최소의 Δ를 가지는 응용에게 모든 잔여용량을 제공하는 알고리즘을 사용한다. 최소의 Δ를 가진다는 의미는 그 응용이 현재까지 다른 응용에 비해 가장 적은 서비스를 받았다는 것을 나타낸다.

3.3 스케줄링 순서결정

제안된 알고리즘에서 각 응용은 요청을 서비스하기 위해 각 요청에 대해 3개의 파라메타를 유지한다. 즉 요청의 도착시간, $S(R_i^j(k))$, 요청의 참조모델에서의 종료시간, $F_{ref}(R_i^j(k))$, 실제잔여 용량을 할당 시 종료시간, $F_{eff}(R_i^j(k))$ 이다.

이 밖에도 응용은 자신이 시스템으로부터 이상적으로 받아야 되는 서비스 양과 실제 받은 서비스양의 차이를 나타내는 파라메타인 Δ를 유지한다. 이 값으로부터 응용이 현 요청이 가져야 하는 실제 잔여용량 $\overline{e_i^j}$을 결정한다. 이 잔여 용량으로부터 이 요청의 실제 잔여용량 할당 시 종료시간을 다음과 같이 유도할 수 있다.

$$F_{eff}(R_i^j(k)) = S(R_i^j(k)) + \frac{R_i^j(k)}{r_i^j(1+\overline{e_i^j})}$$

서버는 모든 요청에 대해 이 종료시간을 결정한 후, 이 시간의 오름차순으로 서비스를 한다.

4. 실험 결과

본 논문에서는 몇 가지 실험을 통해서 제안한 스케줄링 알고리즘을 검증하였다. 실험은 두 가지로 이루어졌으며, 첫 번째 실험에서는 각 응용의 요청크기와 서버전이 확

률이 균일분포를 따른다고 가정했고, 두 번째 실험에서는 세 개의 응용들이 고정된 요청길이를 가지며, 주기적인 서버방문패턴에 따라 서버를 방문한다고 가정하였다.

그림 2는 각각의 예약용량이 0.1인 5개의 응용들이 (A1~A5) 받은 서비스를 보여주고 있다. 여기에서 최대 요청크기는 10으로 가정하였다. 그림 2에서 보듯이, 각 응용들이 받는 서비스는 시간이 변하여도 거의 같은데, 이는 제안된 기법이 장기간 공정서비스를 제공하고 있다는 것을 보여준다.

그림 3은 각 응용의 예약 용량과 서버로의 작업량을 달리하여 실행한 두 번째 실험의 결과를 보여주고 있다. 응용 A1, A2, 그리고 A3의 예약용량은 각각 0.1, 0.2, 그리고 0.3이며, S1과 S2로의 작업량의 비는 각각 1:2, 3:5, 그리고 5:8이다. 그림에서 보여주듯이 세 개의 응용이 장시간동안 각 서버에서 받은 서비스는 위의 작업량의 비에 매우 근접함을 알 수 있다.

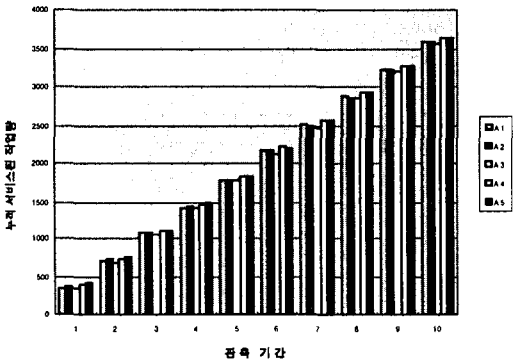

그림 2. 같은 예약 용량을 가지고, 임의의 요청패턴을 생성하는 응용들의 누적된 서비스 량

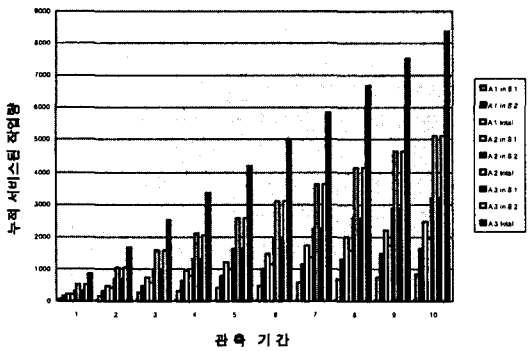

그림 3. 다른 예약 용량을 가지고, 주기적인 요청패턴을 생성하는 응용들의 누적된 서비스 량

5. 결론

본 논문에서는 잔여서버 용량을 공정 분배하는 새로운 기법을 제안하였다. 이 기법은 서버에서 장기간 동안 사용된 잔여서버용량을 관찰하여 각 서버에서 어느 정도의 잔여 서버용량이 발생하였는지를 알아내고, 이를 기반으로 현재 서비스중인 각 응용에게 응용의 가중치에 비례하여 잔여서버용량을 할당한다. 추후, 이 알고리즘을 확장하여 공정서비스를 보장하면서, 성능을 강화하는 개선책을 고려중이며, 실제 운영체제에 구현하여 실효성을 검증하려고 한다.

참고문헌

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

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

[3] 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, 1997

[4] J. C. R. Bennett and H. Zhang,"WF2Q: Worst-case Fair Weighted Fair Queueing," in Proc. of IEEE INFOCOM, pp. 120-128, 1996

[5] A. Chandra, M. Adler, P. Goyal, and P. Shenoy, "Surplus Fair Scheduing: A Proportional Share CPU Scheduing Algorithm for Symmetric Multiprocessors," in Proc. of Operating System Design and Implementation(OSDI), 2000.

[6] J. Blanquer and B. Özden, "Fair Queueing for Aggregated Multiple Links," in Proc. of ACM SIGCOMM, pp. 189-197, 2001.

[7] J. Bruno, E. Gabber, B. Özden and A. Silberschatz, "Move-to-Rear List Scheduling: A new Scheduling Algorithm for Providing QoS *Guarantee*, " *in Proc. of ACM Multimedia, pp.* 63-73, 1997.

[8] J. H. Lee, K. Park, K. Kim, and S. L. Min, "A Fair Queueing Scheme for Heterogeneous Multi-Server Systems, " submitted for publication in Proc. of ISOCC 2004.