2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

클러스터링 웹서버 시스템에서의 QoS 보장 스케줄링 기술 및 관리 기법

이도영*, 이상문*, 박종규*, 김학배*
*연세대학교 전기전자공학과
e-mail:toldy@yonsei.ac.kr

Scheduling Technique and Management Method Guaranteeing QoS in a Clustering Web Server System

Do-Young Lee*, Sang-Moon Lee*, Jong-Gyu Park*, Hagbae Kim*
*Dept of Electrical and Electronic Engineering, Yonsei University

요약

클러스터링 웹서버에서 트래픽 문제에 대한 장기적인 해결책으로, 서로 다른 종류의 트래픽들에 우선순위를 부여하고, 이에 따라 네트웍에서 트래픽을 지능적으로 조절하기 위하여 QoS 보장을 위한 목표를 설정한다. 구체적으로 CBQ 구현에 관련된 실시간 스케줄링 기법을 이용하고, 패킷크기의 최적화와 접속별 대역폭 공정배분을 통한 트래픽의 효율적인 관리를 할 수 있음을 보인다.

1. 서론

인터넷 사용인구의 폭발적인 증가에 비례하여 인터넷 사용 증가로 인해 예상되는 트래픽(traffic)의 발생은 웹 서버단의 큰 문제를 발생시킬 것이다. 특정 사이트에 사용자가 폭주하여 서비스의 병목현상이 발생하고 웹서버의 처리 불가로 사이트가 다운되는 경우를 자주 접할 수 있다. 일반적으로 기존의 단일 웹서버에서는 고가용성 및 고성능의 시스템을 구축하여 웹서버에서 발생될 수 있는 이러한 문제들을 해결하고 있으나, 급격한 접속수에 따른 서버의 확장성 및 고가용성 확보가 어렵다. 고가용성의 기본적인 목표는 적당한 가격에 고장포용시스템을 제공하는 것이다.

이에 대한 해결책으로 컴퓨터를 이용한 클러스터링 기법이 이용되고 있는데, 크게 분산 클러스터링과 HA(High Availability) 클러스터링으로 나눌 수 있다. 이 가운데 분산 클러스터링은 말 그대로 하나의 작업을 여러 대의 컴퓨터에서 분산 처리해 전체적인 처리율을 높이기 위한 것이고, HA 클러스터링은 클러스터내의 컴퓨터 가운데 하나가 다운되더라도 작업이 중단되지 않고 지속적으로 수행될 수 있도록 하기 위한 방법이다.

이중 HA 클러스터링의 구조에서 고가용성을 구현하기 위해서는 네트워크 서비스 이용, 리얼서버의 상태, 그 밖의 환경적 문제 등 각각의 서버에서 제공하는 서비스의 가용성을 모니터링하는 시스템이 필요한데, 로드밸런서는 MON을 이용하여 클러스터를 구성하는 모든 리얼서버들의 부하상태를 파악하고 데몬의 동작여부를 파악하여 고장이 발생한 서버는 서비스 요청을 할당하지 않고 그 역할을 Backup Pool에서 갖어온 다른 서버가 대신하게 함으로서 가상서버의 전반적인 가용성을 높인다.

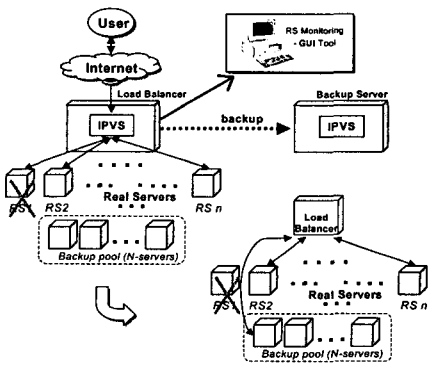

그림 1 고가용성 보장을 위한 클러스터링 서버

서버들의 웹서버 데몬의 상태를 모니터링하는 프

로그램은 임의의 리얼서버가 동작하지 않는다는 신호를 받으면 스크립트를 실행시켜 해당 서버를 부하분산 라우팅 테이블에서 삭제하여 고장난 리얼서버로는 서비스 요청을 분배하지 않도록 한다. 반대로 고장났던 리얼서버가 복구되거나 새로운 서버가 추가되었을 경우에는 복구된 리얼서버가 서비스 요청을 처리할 수 있도록 하여 언제나 최적의 상태로 가상서버가 동작하게 되어 고가용성을 보장하게 된다.

2. QoS 보장 스케줄링

트래픽 문제에 대한 장기적인 해결책은 QoS (Quality of Service)에서 찾아야 한다. QoS는 서로 다른 종류의 트래픽에 우선순위를 부여하고, 이에 따라 네트웍에서 트래픽의 중단, 속도 저하 또는 상승이 지능적으로 가능하게 된다는 것을 강조하고 있으나, 이는 완전한 해결책이 아니다. 이를 위해 실시간 스케줄링 기법이 필요하지만, 로드밸런서에 적용되는 기존의 스케줄링 알고리즘과 같은 직관적인 알고리즘은 구현하기는 쉽지만, 오버헤드가 커진다는 단점이 있다. 그러나, 웹에서의 모든 처리모델이 database의 transaction모델과 비슷하므로, 이에 실시간 스케줄링 기법을 이용하여 직관적인 기법보다 높은 성능을 낼 수 있도록 한다. 즉, 웹 서버가 처리하여야 할 작업이 일반적인 데이터를 요구하는 것이 아니라, 시간지연이 발생했을 때 큰 손실을 가져다 주는 실시간 작업이나 제어시스템에 사용되는 입출력 데이터일 경우 그 시간 제약성을 보장해줄 수 있는 스케줄링이 필요하며, 이를 위해서 스케줄링시 고려가 되어야할 여러 요소인 실시간 및 비실시간 작업의 분류, 서버의 상태 또는 클러스터 내의 네트웍 부하량 등을 선정하고, 이를 바탕으로 한 적절한 스케줄링 알고리즘이 요구된다.

이때 각각의 서비스는 데이터를 TCP단으로 넘김으로서 데이터는 각 단계를 거쳐 인터넷으로 전송되게 되는데, 이를 통하여 우리가 구현 하고자 하는 QoS는 이 패킷의 특성에 맞게 데이터의 전송을 제어하는 스케줄링의 문제로 바뀌게 된다.

인터넷이나 다른 네트웍 상에서, QoS는 전송율, 에러율, 그리고 측정과 개선이 가능하며, 어느 정도는 미리 보장할 수 있는 특성을 갖고 있다. 그러나 현재 인터넷 상에서는 이러한 QoS보장을 위한 제어 알고리즘이 구현되어 있지 않고, 최선형(best-effort) 형태로 서비스를 하고 있다. 따라서, 인터넷 트래픽이 과중하지 않을 때는 어느 정도 연속적인 성능이 유지되지만 과중해지면 전혀 보장해 줄 수 없다.

3. 트래픽 관리

로드밸런서에서 우선순위를 배정하여 리얼서버로의 부하를 분산시키는 것은 실제로 많은 문제를 안고 있다. 무엇보다 QoS 개념을 로드밸런서에 추가한다는 것은 트래픽을 분류한다는 것을 의미하는데, 이렇게 새로운 기능을 추가하게 되면, 그 처리속도가 떨어지게 되고, 이로써 신호의 지연이 발생하고 관리기능에 부담이 가게 된다.

데이터 트래픽을 관리하기 위해서는 효과적인 기술이 필요하다. 트래픽 관리 기술은 독립적으로 사용될 수도 있지만 대표적인 QoS 관리 모델로 알려진 인터넷의 통합서비스(Integrated Service)모델과 차등서비스(Differentiated Service)모델을 같이 복합적으로 사용되면서 효과적으로 통합이 된다.

3.1. CBQ / 큐 관리(Queue management)기법 연구

CBQ(Class-Based Queuing)는 priority queuing의 변형으로서 서비스 클래스에 따라 서비스 큐를 정의하고 있으며, 특정 서비스 클래스가 낮은 우선순위로 인해 자원을 할당받지 못하고 높은 우선순위의 다른 서비스 클래스의 트래픽이 시스템 자원과 대역폭을 독점하는 것을 막음으로써 공평성을 제공하여, 굶주림(starvation)을 막기 위하여 고안되었다. 그러나 CBQ는 일정 수효의 사용자가 대역폭을 차지하면, 나머지 사용자들은 모두 대기정렬에서 탈락하기 때문에 충분한 것은 아니다.

3.2. CBQ 구현에 관련된 스케줄링

CBQ에서 사용되는 스케줄링 알고리즘으로는 링크-공유 스케줄러(link-sharing scheduler)와 GPS (Generalized Packet Scheduler)이다. 링크-공유 스케줄러는 탑-레벨 링크-공유 스케줄러로 망이 과부하 되었을 때 동작하여 우선 순위가 낮은 클래스의 링크 사용량을 한정된 값으로 제한하고 우선 순위가 높은 클래스에 할당된 대역폭보다 더 많이 사용할 수 있도록 대역폭의 사용량을 조절하는 기능을 수행한다. 대역폭의 사용량을 조절하는 방법은 오버플로우 된 큐의 데이터를 버리거나 데이터의 전송을 지연시킨다. GPS는 PRR(Packet-by-packet Round Robin) 패킷 스케줄러와 WRR(Weighted Round Robin)패킷 스케줄러로 구현되어 우선 순위에 기초하여 스케줄링을 수행한다.

탑-레벨 링크-공유 스케줄러는 링크 공유 구조에서 링크 공유의 조건을 유지하기 위하여 사용되어지고, PRR 패킷 스케줄러와 WRR 패킷 스케줄러는 일반적인 패킷 스케줄링 방법으로 지연과 지터, 데이터 손실 등과 같은 응용들의 요구 사항에 기초하여 패킷들 사이에서 서비스를 차별화하는데 사용되어진다.

탑-레벨 링크-공유 스케줄러는 서비스 클래스가 자신의 사용 한계값에 도달되지 않았거나, 클래스 계층 구조에서 자신의 상위 레벨의 사용 한계값에 여유가 있으며 상위 레벨이 거의 탑-레벨이라고 볼 수 있을 때에만 규제되지 않은 트래픽을 발생시킨다. 이 조건에서 하나만 위배되어도 서비스 클래스는 overlimit으로 간주되어 탑-레벨 링크-공유 스케줄러에 의해 규제되어진 트래픽이 발생된다. 탑-레벨 링크-공유 스케줄러는 서비스 클래스의 사용 한계값(rate)을 유지하기 위하여 overlimit된 서비스의 트래픽들을 버리거나 다른 서비스 클래스로 재정의한다.

PRR 패킷 스케줄러와 WRR 패킷 스케줄러는 우선 순위에 기반하는 스케줄링 기법으로 같은 우선 순위에서 패킷을 스케줄링하는 방법에 따라 구분되어진다. PRR 패킷 스케줄러는 단순한 라운드로빈 스케줄링 방법으로 같은 우선 순위를 갖는 서비스 클래스 내에서 자신의 스케줄링 라운드동안 패킷 단위로 서비스된다. 반면 WRR 패킷 스케줄러는 서비스 클래스에 할당된 대역폭의 비에 따라 각 서비스 큐에 가중치(weight)를 주어 서비스한다.

구현된 CBQ는 낮은 우선 순위의 굶주림 문제를 해결하고 높은 우선 순위를 갖는 서비스 클래스의 QoS 요구 조건도 만족시켜 줄 수 있다. 또한 각 서비스 클래스들이 사용하는 링크의 사용 한계값을 정의하고 제한함으로써 링크의 사용량을 일정한 값으로 유지시키게 되어 새롭게 생성되는 응용들이 발생시키는 실시간 트래픽들을 망에서 수용할 수 있는 융통성을 제공해 준다.

3.3. 트래픽 쉐이핑

트래픽 쉐이핑은 원래 네트워크 내부로 유입되고 유출되는 트래픽의 양과 유출되는 트래픽의 속도를 조절하는 메커니즘이다. 이러한 기능은 가상서버 내부 네트웍의 구조에서도 적용될 수 있다. 로드밸런서에서 유입되는 트래픽의 분량과 속도를 제어할 때에, 우선 사용할 수 있는 TCP 트래픽 조율 기능은 TCP 윈도우 크기 조율(Window Sizing)이다. 그런데, 이 기능은 작은 크기의 데이터 패킷 전송 시에 문제가 있다. 트래픽 조율과 대기정렬 기능은 서로 같이 협력하도록 사용하는 것이 이상적이다. 이때 트래픽 조율의 다른 제약은 전체 윈도우 크기만 조율할 수 있고, 개별 패킷의 크기는 조율할 수 없다는 점이다.

3.3.1. 패킷 크기 최적화

패킷 크기 최적화(Packet-size Optimization)라는 기능은 전송 트래픽의 패킷 크기를 작게 하여 입력 트래픽의 지연을 최소화하는 것이다. 예를 들어, 64Kbps 회선에서 1500-byte 패킷을 전송하는 데는 187ms가 소요된다. 이것은 입력 데이터와 같은 높은 순위의 패킷은 전송되기 전에 최소 187ms 동안 기다려야 함을 의미한다. 패킷 크기 최적화 기능은 큰 패킷을 사용하는 TCP 접속을 약 512bytes 이하의 최소 크기로 작게 분할하여 이 문제를 해결한다. 결과적으로 패킷은 전송되기 전에, 허용 가능한 64ms, 또는 왕복 시간을 고려하더라도 최장 128ms만 기다리면 전송 될 수 있다.

3.3.2. 접속별 대역폭의 공정 배분

접속별 대역폭의 공정 배분(Fair allocation of bandwidth by connection)은 접속별 대역폭의 공정 배분 기능을 사용하여, 망 관리자는 같은 등급안에서 각 접속별로 대역폭을 공정하게 배분하여 대역폭의 할당을 좀 더 세밀하게 제어할 수 있다.

TCP 전송율 조정(TCP rate shaping)시 전송율 조정은 SMTP, FTP 및 웹 서버에 의하여 네트웍에 송신되는 트래픽의 양과 전송율을 제어하는 기능을 제공한다. SMTP, FTP 및 웹 접속 윈도우 크기를 조정하여, 이러한 트래픽이 폭발적으로 증가하는 것을 제어하여 데이터의 전송에 지장을 주지 않도록 한다.

그러나, TCP 전송율 조정 기능은 단지 TCP 기반의 트래픽만 제어하므로, 분명히, 대기정렬과 같은 다른 QoS 기능과 함께 사용되어야 한다. 또한, 서버에서 송신하는 데이터의 양은 적고 전송율이 낮은 경우에는 프로그램에 일정한 전송율을 계속 제공하지 못한다. 이런 경우에 등급별 대기정렬 및 패킷 크기 최적화 기능이 유효하게 사용될 수 있다.

특히 저속의 회선에서, 관리자들은 모든 사용 가능한 대역폭을 최대로 사용할 수 있어야 한다. 이는

같은 웹 자료에 대한 반복되는 요청을 방지함을 의미한다. 이때 캐싱 기능은 자주 접속하는 웹 페이지 데이터를 캐시에 저장하여, 광역통신망의 트래픽 부하를 줄이고 보다 많은 대역폭이 배정되도록 한다. 만약 웹 기반의 사용자 인터페이스를 지원하는 경우에 사용자 스크린의 정적 부분(예, 로고, 이미지, 텍스트 등)을 캐시에 저장할 수 있다. 대체로, 웹 자료의 30-40%를 망 종단의 캐시에 저장할 수 있고 따라서, 대역폭을 크게 절약할 수 있다.

3.4. 수락제어

어떤 종류의 트래픽을 리얼서버로 전송하고 받아들일 것인가를 결정하는 정책이다. 이러한 정책없이 운영할 경우 가상서버 내부에서 발생할 수 있는 다양한 문제점들에 대해 해결책을 제시할 수 없게 되어 QoS보장이 불가능하게 된다.

3.5. 폴리싱

로드밸런서에서 각각의 트래픽을 분류하는 것은 중요하다. 또한, 각각의 데이터의 종류에 따라서 적정 크기의 대역폭이 할당되어야 한다. 이를 통하여 사용자들이 자원의 할당을 보다 세밀하게 제어할 수 있도록 해준다.

효과적인 QoS 과정은 하나의 싸이클을 구성한다. 즉, QoS는 트래픽을 지속적으로 감시하여, 모든 종류의 트래픽이 사용하고 있는 대역폭을 보고해 주는 것이다. 이 정보를 이용하여, 로드밸런서에서는 관리 기준을 더욱 정밀하게 조정하고, 다시 새로 설정된 기준의 시행효과를 관측하는 것이다.

그림 2 QoS과정

감시는 GUI에 의한 보고 기능을 통하여 로드밸런서가 얼마나 효율적으로 부하를 분배하는지를 파악할 수 있게 되고, 이를 통하여 새로운 정책을 수립할 수 있다.

3.6 혼잡관리

엄격한 수락제어와 큐관리에도 불구하고 다양한 트래픽을 수용하다 보면 혼잡상황을 완전히 피할 수는 없게 된다. 혼잡은 로드밸런서의 작동을 예측하기 힘들게 만들고, 시스템 버퍼들을 채워서 데이터 손실률을 높이며, 다시 재전송률을 증가시켜 악순환을 반복하게 하는 원인이 된다.

4. 모니터링 기술

QoS보장이 된 후 사용자들이 가장 궁금해하는 것은 제공중인 QoS가 요구사항에 따라 제대로 지켜지고 있느냐 하는 것이다. 이를 위해 네트워크 장비 및 네트워크 상태 정보를 규칙적으로 측정하여 분석하고 정책 관리 기능에 피드백하여 QoS관리에 활용하며, 또한 이를 사용자가 이해할 수 있는 형태의 정보로 변경하여 분배함으로써 요구한 조건이 잘 지켜지고 있는지를 확인시킬 필요가 있다.

5. 결론

네트워크의 어느 부분에서 트래픽을 관리할 것인지, 그리고 효율적으로 평가하기 위해서는 트래픽 관리 기능을 정의하는 것이 중요하다. 지금까지는 로드밸런서에서 트래픽을 관리함에 있어, 직관적인 알고리즘을 주로 이용하며, 큐잉(대기정렬)의 기능에만 치중하였다. 그러나 QoS보장 실시간 스케줄링 기법을 이용하여, 전체적으로는 실시간 트래픽들을 가상서버 구조내에서 융통성을 제공해주며, 여기에 패킷크기의 최적화를 통하여 입력 트래픽의 지연을 최소화하여 실제 정보를 공급받는 사용자의 느낌에 의해서 평가되는 QoS의 향상을 얻을 수 있다.

참고문헌

[1] William Stallings, "TCP/IP and ATM Design Principles", Prentice-Hall International, Inc.

[2] V.Cardellinu, M.Colajanni & P.Yu, "Redirection Algorithms for Load Sharing in Distributed Web-Server Systems", Distributed Computing Systems, 1999. Proceedings. 19th IEEE International Conference on, pp.528-535, 1999.

[3] A. Mourad and H. Liu, "Scalable Web Server Architectures", Computers and Communications, 1997. Proceedings., Second IEEE Symposium on, pp.12-16, 1997.

[4] B. Narendran, S. Rangarajan and S. Yajnik, "Data Distribution Algorithms for Load Balanced Fault-Tolerant Web Access", Reliable Distributed Systems, 1997. pp.97-106, 1997.

[5] K. Shin and X. Cui, "Effects of computing time delay on real-time control systems," Proc. of 1988 ACC, pp.1071-1076, 1988.