잠재 부하 정보와 HTTP 연결 시간을 이용한 웹 서버 부하 분산 정책

김 시 연, 김 성 천
서강대학교 공과대학 컴퓨터학과
{sykim11, ksc}@arqlab1.sogang.ac.kr

Load Distribution Policy of Web Server using Subsequent Load and HTTP Connection Time

Si-Yeon Kim, Sung-Chun Kim
Department of Computer Science, Sogang University

요 약

본 논문에서는 HTTP/1.1을 효율적으로 지원하는 부하 분산 정책을 제안하고자 한다. 이 정책은 사용자의 첫 번째 요청이 전달되면 그 요청의 내장 객체 정보와 현재 살아 있는 HTTP 연결의 에이징(aging)을 고려하여 서버를 선택하는 알고리즘이다. 그리고 디스패처의 잘못된 분산 결정으로 인해 시스템의 성능에 누적되는 악영향을 최소화하기 위한 기법도 제시한다.

1. 서론

최근 웹 페이지는 보통 HTML 문서와 수많은 내장 객체(Embedded Objects)들로 이루어져 있다. 일반적으로 HTML 문서는 약 20여 개 이상의 내장 이미지를 포함하고 있다. 웹 환경에서 이런 내장 이미지는 각각 독립적인 객체이며, 서버로부터 객체를 요청할 때 개별적으로 분리되어 전송된다. 그러므로 웹 클라이언트의 일반적인 동작은, 기본 HTML 문서(컨테이너 페이지)를 요청하여 서버로부터 응답을 받아온 뒤, 즉시 기본 HTML 문서에 포함된 내장 객체에 대해 서버에 요청을 전송한다.

이런 최근의 웹 환경을 고려하여 HTTP 프로토콜은 지속적인 TCP 연결 기법을 사용하여 내장 객체를 전송하기 위해서 매번 TCP 연결을 수립하던 오버헤드를 극복하였다.

그러나 실제로 클러스터 기반의 웹 서버에서는 문제점을 발생시킨다. 서버에서는 사용자의 단일 요청을 기본 단위로 하여 작업을 처리하는 반면에, 디스패처에서는 부하 분산을 처리하기 위해 HTTP 연결을 기본 단위로 처리한다는 것으로부터 문제가 발생한다.

2. 웹 서버 시스템의 구조와 기본 알고리즘

사용자에게 빠른 응답 시간과 더 나은 서비스를 제공하기 위하여 여러 개의 프로세서를 클러스터로 묶어서 웹 서버로 사용하는 방법을 고려한다. 클러스터 웹 서버를 구성하는 아키텍처 중에서 서버 효율성과 사용자 응답 시간 측면에서 가장 뛰어난 성능을 나타내는 디스패처 기반의 클러스터를 가정한다[1].

부하 분산 정책은 클러스터 기반의 웹 서버 시스템의 전체적인 성능에 매우 중요한 이슈가 된다. 부하 분산 정책은 클러스터를 구성하고 있는 서버간의 부하를 공평하게 나누고, 사용자의 요청에 대해 최소한의 응답 시간을 제공할 수 있도록 해야 한다. 지난 수년간의 연구를 통해 수많은 부하 분산 알고리즘이 제안되었다. 그 중에서도 가장 기본적으로 사용되는 몇 가지를

소개한다면 다음과 같다[2].

- 라운드 로빈 정책과 가중치 라운드 로빈 정책 (Round Robin, Weighted RR)
- 최소 연결 정책과 가중치 최소 연결 정책 (Least Connection, Weighted LC)
- 최소 부하 정책(Least Loaded)
- 내용 기반 정책(Content-Based)

HTTP/1.1의 지속적인 연결에서 부하 분산이 중요한 이유는, 한번 수립된 연결을 통해 앞으로 부담하게 될 잠재적인 부하가 존재하기 때문이다. HTTP/1.1에서는 기본적으로 분산 정책을 각 요청 단위로 적용할 수 없다. 사용자가 보내는 요청 중에 맨 처음에 들어오는 요청만을 이용하여 HTTP 연결을 할당해야 한다. 그리고 일단 수립된 HTTP 연결을 통해 사용자의 요청이 계속해서 들어오면 디스패처는 이를 별도로 구별하지 않기 때문에 이 요청들에 대해서는 부하 분산 처리를 하지 않는다. 이런 문제점을 고려해보면 결국 잠재적인 사용자의 요청을 어떤 식으로 예측하여 부하 분산에 적용하는지가 관건이 된다.

3. 효과적인 HTTP/1.1 연결 분산 정책

이 논문에서 제안하는 기법은 웹 페이지에 포함되어 있는 내장 객체의 정보를 이용하여 하나의 HTTP 연결을 통해 요청될 잠재적인 부하를 미리 부하 분산에 적용시키는 것이다. 사용자의 첫 번째 요청에 의해 알아낸 잠재 부하 정보를 디스패처의 로드 테이블(load table)에 저장한다. 그리고 로드 테이블의 정보를 바탕으로 하여 새로운 HTTP 연결을 할당할 서버를 선택해야 한다.

그러나 이렇게 초기 예측 부하량을 반영만 시켜놓고 그냥 방치하면 시간이 흐름에 따라 서버가 처리하는 작업에 대해 부하가 줄어드는 것을 적용하지 못하게 된다. 그러므로 TCP 연결이 맺어진 이후의 경과 시간을 계산하여 서버의 서비스 율에 맞춰 현재까지 처리된 부하량을 예측하고, 이것을 부하 인덱스에 적용시켜야 한다. 이런 특성에 기인하여 이 기법을 에이징(aging)이라 부르고, 이 논문에서 제안하는 분산 정책을 에이징 부하 분산 정책이라 부르기로 한다.

TCP 연결의 서비스 처리 결과를 예측하는 알고리즘을 고려해보자. 에이징 분산 정책을 적용하기 위한 알고리즘을 정리하면 다음과 같다.

```
while( There are more connections on server_i){
  if( the first of all established connections on server_i)
    age(Conn_k) = time_current - time_init(Conn_k)
    processed(Conn_k) = age(Conn_k) x u
    load(Conn_k) = load_init(Conn_k) - processed(Conn_k)
  else
    load_now(Conn_k) = load_init(Conn_k)
  load(server) = load(Conn_1) + load(Conn_2) +
            ... + load(Conn_n)
}
next server = Min{load(server_1), ... , load(server_n)}
```

웹 서버는 일반적인 방식으로 작동하다가, HTTP 연결을 종료해야 할 때(FIN 패킷을 보낼 때)에만 패킷을 디스패처를 통해 보내도록 한다. 그러면 디스패처는 FIN 패킷의 정보를 사용하여 로드 테이블을 업데이트한다. 이렇게 하면 잘못된 부하 분산 결정의 오차가 심화되는 것을 방지할 수 있으므로, 악영향이 시스템 전체 성능에 누적되는 것을 방지할 수 있다.

4. 시뮬레이션 및 결과 분석

입력 트래픽에 대해서는 1997년 발표된 SURGE Workload Generator를 활용하였다[3]. 입력 파일의 크기는 SPECWeb99의 파일 크기 분포에 기인하여 구성하였다.

서버의 처리율은 요청 파일 크기에 따른 평균 서비스 시간을 계산하였는데, P-III 450 시스템의 Apache 1.3.12를 설치하여 실험한 결과, 약 21MB/sec의 처리 속도를 찾아내었다. 그리고 디스패처에서 실제로 패킷을 백-엔드 서버에게 전달하는 오버헤드가 있지만, 이 오버헤드는 모든 분산 정책에서 동일하게 발생하므로 0으로 가정한다. 그리고 디스패처에서 일어나는 부하 분산 결정 오버헤드와 TCP 종료 오버헤드는 실험을 통하여 적정값을 가정한다.

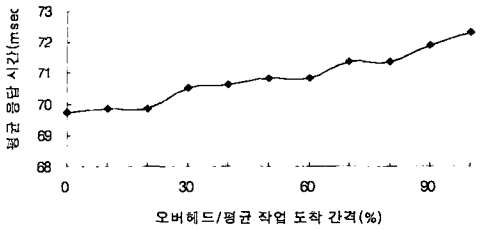

그림 4.1 부하 분산 오버헤드에 따른 성능 변화

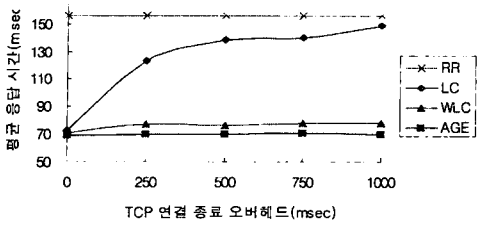

그림 4.2 TCP 연결 종료 오버헤드에 따른 성능 변화

실험에서 가정하고 있는 서버의 개수는 4대이며, 입력 트래픽의 평균 작업 크기가 약 200KB이므로 평균 작업 도착 간격이 2.3msec이면 트래픽이 양호하다고 말할 수 있다. 실험의 대상이 되는 알고리즘은 라운드 로빈 정책(RR)과 최소 연결 정책(LC), 가중치 최소 연결 정책(WLC)이다.

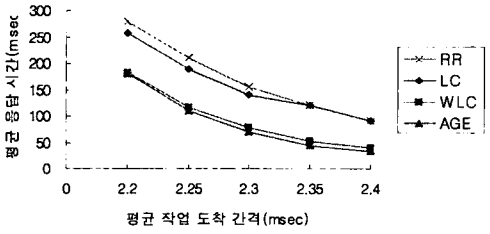

그림 4.3 기존 알고리즘과 에이징 기법의 성능 비교

그림 4.3의 결과를 분석하면 RR 정책은 전체적으로 가장 저조한 성능을 보인다. LC는 동적인 부사 분산 결정 요소인 TCP 연결의 개수를 사용하지만 각 서버의 성능 차이가 발생하거나 TCP 연결 종료 오버헤드가 길어지면 성능이 매우 민감하게 영향을 받으므로, 항상 효율적인 성능을 낸다고 보장할 수 없다. WLC는 가중치의 영향으로 LC보다 훨씬 나은 성능을 보이는데, LC에 비해서 트래픽이 양호한 경우에 평균 43.88%의 성능향상을 보인다. 그리고 WLC와 AGE를 비교한 실험 결과, 트래픽이 복잡한 경우에는 평균 1.71%, 트래픽이 양호한 경우에는 평균 10.25%, 트래픽이 한산한 경우에는 평균 16.82%의 성능 향상이 일어난다.

마지막으로 서버 활용도의 지표로서 최소 부하 서버 선택 적중률을 고려해보았다. 이 적중률은 부하 분산 결정 순간에 최소 부하 서버를 선택하는 횟수를 백분율로 나타낸 것이다.

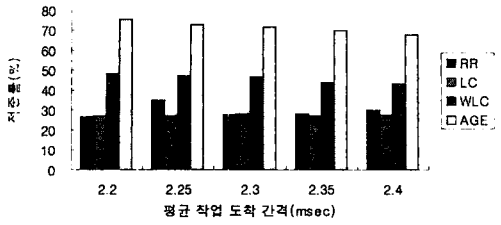

그림 4.4 최소 부하 서버 선택 적중률

5. 결론

이 논문에서 제안하고 있는 알고리즘은 HTTP 연결의 잠재적인 부하라는 동적인 요소를 가지고 간단한 에이징 기법을 사용하여 정적인 분산 결정을 할 수 있다. 게다가 매번 하나의 HTTP 연결이 끊길 때에는 서버가 FIN 패킷을 디스패처를 통해 클라이언트에게 보내도록 하므로, 디스패처의 로드 테이블 정보를 보정할 수 있다. 이런 두 가지 기법을 사용하여 지속적인 TCP 연결을 효율적으로 이용할 수 있는 부하 분산 정책을 적용할 수 있다.

6. 참고 문헌

[1] T. Schroeder, S. Goddard and B. Ramamurthy, "Scalable Web Server Clustering Technologies." IEEE Network, May/June, pp.38-45 2000.

[2] G. D. Hunt, G. S. Goldszmidt, R. P. King and R. Mukhejee, "Network Dispatcher : a connection router for scalable Internet services." Proceedings of the Seventh International World Wide Web Conference, Brisbane, Australia, April, 1998. Available at www7.scu.edu.au/programme/fullpapers/1899/ com1899.htm

[3] P. Barford and M. Crovella, How to use SURGE(Scalable URL Reference Generator). http://cs-people.bu.edu/barford/surge_1.00a.tar.gz.