2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

작업의 잔여량을 이용한 웹서버 로드 밸런싱

염미령 홍일구
홍익대학교 전자계산학과, 컴퓨터공학과
{miryeom, ighong }@cs.hongik.ac.kr

Web Server Load Balancing via Remaining Workload

Mi-ryeong Yeom Il Gu Hong
Dept. of Computer Science, Dept. of Computer Engineering, Hong-Ik University

요약

최근 들어 매우 인기 있는 웹 사이트는 우수한 성능의 단일 서버 또는 미러(mirror) 사이트에만 의존 할 수 없을 정도로 인터넷 사용이 급증하였다. 늘어나는 트래픽을 감당해내며 사용자를 만족시키기 위해서는 클라이언트의 요청들을 여러 대의 서버에 분산 처리 할 수 있는 고성능 웹서버 클러스터링 시스템에서는 제공되어야 한다. 본 논문에서는 부하 분배기를 이용한 로드 밸런싱을 소개하고 지역성과 로드밸런싱의 균형을 이루는 분배 알고리즘을 제시한다. 본 논문에서 제안된 알고리즘은 정적 웹 환경에서의 각 요청들은 작업 시간이 고정되어 있지 않다는 점에 착안하여 유효 요청의 수와 잔여량을 고려한 부하 분배 알고리즘을 개발하였다.

1. 서 론

1990년대부터 인터넷의 발전에 가장 큰 영향을 주었던 것은 바로 www이다. 웹은 인터넷상에 광범위하게 분산된 다양한 형식의 데이터를 브라우저를 통해 손쉽게 검색할 수 있을 뿐만 아니라 모든 응용의 사용자 환경을 손쉽게 포함함으로써 새로운 데이터의 통합을 위한 가장 성공적인 기술로 평가받고 있다. 최근 이러한 웹의 고도화되고 다양한 기능으로 인해 그 사용자 수는 폭발적인 증가 추세로 2002년까지 3억 2천 여 명에 달할 것으로 예상한다[1]. 그러나, 현실적으로 웹 사용자의 효율적인 활용을 뒷받침할 수 있는 서버의 용량에도 한계를 가져오게 되었다. 서버는 사용자의 요구에 대한 수요를 감당하지 못해 서버의 응답시간이 길어지고 사용자의 인내심은 짧아졌다[2]. 그러므로 인터넷 기업의 사용자들은 서버의 성능이 나쁘다라고 믿게 되면 보안(security) 부분도 의심하게 되므로 결국엔 고객 이탈로 이어진다[3]. 대규모의 가입자를 보유하고 있는 다수의 인터넷 기업들은 사용자의 증가에 따라 서버의 용량 증가에 많은 비용을 투자하고 있는 실정이며 사용자의 요구를 만족시키기 위해 끊임없이 용량을 늘여야만 하는 상황에 이르렀다.

서버의 성능은 사용자 입장과 서버의 입장으로 양분해 볼 수 있다. 우선 서버의 입장에서는 사용자의 요청을 처리하는 처리율로 표현 될 수 있다. 초당 처리되는 요청의 수 또는 초당 전송되는 바이트 수로 측정한다. 두 번째로 사용자의 입장에서는 사용자 평균 응답 시간으로 표현될 수 있다. 응답 시간이란 사용자가 요청을 보낸 후 응답을 받기까지의 시간을 의미한다. 서버는 일시적으로 사용자의 수가 급증하여 서버와 네트워크를 연결하는 링크 용량이 포화 상태일 때와 CPU의 대역폭이 포화일 때 과적이 된다.

웹 서비스의 보편화와 접속 횟수의 급속한 증가는

확장성과 신뢰성이 우수한 시스템을 요구한다. 클러스터링은 비교적 가격이 싼 컴퓨터를 이용해 확장성과 신뢰성이 우수한 시스템을 구성할 수 있는 방법을 제공해준다. 그림 1은 일반적인 웹 클러스터링의 구조이다. 전방에 부하 분산 서버가 놓이고 후방은 후위 서버가 놓이게 된다. 부하 분산 서버는 사용자의 요청을 후위서버에게 전달하는 역할을 담당한다. 후위 서버는 실제 클라이언트의 요청을 처리하게된다.

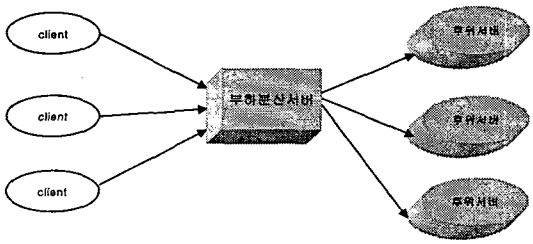

그림 1 일반적인 클러스터링 구성도

부하 분산 서버는 요청을 처리할 후위 서버를 선택한 후 클라이언트와 설정된 컨넥션(connection)을 후위 서버에게 전달하기 위한 메커니즘이 필요하다.

프락시 서버의 경우 HTTP redirection 메시지를 통해 부하 분산을 처리 할 수 있지만 불필요한 네트웍의 증가로 인해 사용자 응답시간의 증가와 클러스터링 시스템에서의 보안상에 문제가 발생 할 수 있다. 그러므로 웹 클러스터링 시스템에서는 커널의 네트웍 레벨을 수정하여 컨넥션을 전달한다.

부하 분산 서버에서 클라이언트와 설정된 컨넥션을 전달하기 위해 사용되는 방법으로 TCP - SPLICING과 TCP - HANDOFF방식이 있다. SPLICING 방식은 부하 분산 서버에서 후위서버를 결정하게되면 해당 후위 서버와 tcp 컨넥션을 설정하게되고 주소 변환을 통해 후위 서버에게 전달한다. 후위 서버로부터 전달되는 응답에 대해서도 유사한 주소 변환작업을 수행하여 분산 서버를 거쳐 클라이언트에게 전달된다. 그림2는 일반적인 SPILCING방식을 보여준다. HANDOFF 방식은 클라이언트와 부하 분산 서버 사이에 형성된 컨넥션을 후위 서버에게 넘겨주고, 후위서버로부터 의 응답은 직접 클라이언트에게 전달된다. 그림 3은 HANDOFF 방식이다.

HANDOFF 방식의 처리율은 후위 서버의 수에 비례하지만 SPLICING방식은 후위 서버의 수에 관계없이 일정한 값을 유지하므로, HANDOFF 방식이 좀 더 우수한 방식이라 할 수 있다.

웹 서버 클러스터링의 성능 향상의 관건은 로드 밸런싱을 통한 클러스터링 웹서버의 처리율을 향상시키고 사용자의 응답 시간을 줄이는 것이다. 부하 분산 서버는 후위 서버에게 공평하게 작업을 분배하기 위해 각 후위 서버의 부하 상태를 유지하여, 새로운 요청이 도착하면 가장 적은 부하가 걸린 후위 서버를 선택한다. 또, 부하 분산 서버는 서비스의 내용이나 종류를 고려하여 후위 서버를 할당하기도 한다.

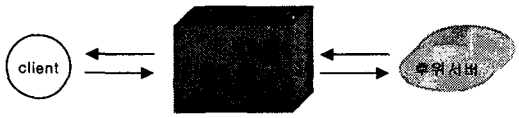

그림 2 SPLICING 방식

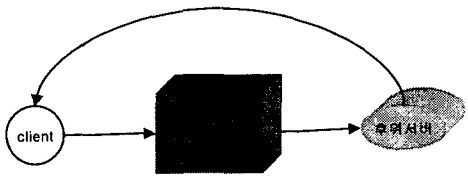

그림 3 HANDOFF 방식

예를 들어 audio/video, html/text등 컨텐츠의 종류나 특정 서비스에 맞게 서버를 특화시킬 수도 있다[10]. 이 방식은 후위 서버들이 데이터 베이스를 분산 저장하게 되므로 확장성을 높이고자 하였지만 특화된 서버는 시스템 자원을 편중되게 사용 할 수 있다. 즉, 정적 컨텐츠를 서비스하는 서버는 대부분의 작업이 디스크 접근이므로 입출력 병목이 발생할 수 있으며 CGI와 같은 동적 데이터를 요구하는 작업들만 서비스하는 서버는 CPU가 과적 될 수 있다[11].

부하 분산 서버는 후위 서버의 메모리 캐쉬 상태 정보나 디스크 큐의 상태 정보를 이용하여 클라이언트의 요청을 처리할 후위 서버를 결정 할 수도 있다. 부하 분배기가 후위 서버를 선택하는 기준으로 지역성만을 고려할 경우 특정 후위 서버에 과부하가 발생할 것이고, 공평하게 분배하는 것만 고려한다면 각 후위 서버는 빈번한 디스크 액색스를 하게 되므로 사용자 응답 시간이 지연될 것이다. 그러므로, 후위 서버를 선택함에 있어 지역성과 부하 분배의 균형을 이루는 부하 분배 알고리즘이 요구된다.

본 논문의 구성은 제 2절에서는 관련 연구를, 제3절에서는 우리가 제시하는 부하 분배 알고리즘에 대해 설명하고, 마지막으로 제 4절에서는 실험 데이터에 대한 소개를 하며 마지막으로 제5절에서는 결론 및 향후과제로 논문을 맺는다.

2. 관련연구

2.1 정적 로드 밸런싱

① 라운드로빈(Round-Robin)

서버의 상태나 네트워크의 상태들을 고려하지 않고 단순하게 요청을 전달하므로 서버의 사양이 같을 때 효율적일 수 있는 알고리즘이다.

② 가중치된 라운드로빈 (Weighted Round-Robin)

서버 중 사양이 우수하던지 또는 환경이 우수하다면 가중치를 주어 더 많은 요청을 처리 할 수 있다. 그러나, 실제 과부하가 일어날 때 서버들 간에 동적인 부하 불균형 상태가 발생할 수 있다.

2.2 동적 로드밸런싱

①최소 컨넥션 스케줄링 (Least Connection Scheduling)

최소 컨넥션 스케줄링은 가장 접속량이 작은 서버로 연결함을 의미한다. 각 서버는 동적으로 실제 접속된 연결들의 숫자를 카운트한다. 요청의 크기가 클 경우 비슷한 성능의 서버로 구성된 시스템에서는 요청들이 한 서버로만 집중되지 않는다. 그러므로 접속 부하가 클 경우에도 매우 효과적으로 작업을 분산한다. 그러나 실제로는 TCP의 TIME_WAIT 상태 때문에 좋은 성능을 낼 수는 없다. TCP_WAIT 상태는 보통 2분이다. 이 시간은 서버가 수 천개의 요청을 처리할 수 있는 아주 긴 시간이라 할 수 있다. 만약, 서버가 TCP_WAIT 상태인 컨넥션의 수를 매우 많이 갖고 있다고 하자. 이 서버는 처리 능력이 충분하더라도 총 연결 컨넥션의 수가 매우 크므로 새로운 요청을 받아 들일 수 없게 된다. 그러므로, 최소 컨넥션 스케줄링은 다양한 처리 용량을 지닌 서버로 구성되었을 경우 부하 분산이 효율적이지 못하다.

②LARD(Locality Aware Request Distribution)

LARD[4]에서는 후위 서버의 지역성을 높일 뿐만 아니라 부하를 분배하기 위해 각 후위 서버의 유효 컨넥션 수를 부하 분배에 이용한다. 부하 분배기에 새로운 요청이 도착하면 동일한 요청을 수행했던 후위 서버를 찾아내고 이것의 유효 컨넥션의 수를 측정한다. 유효 컨넥션의 수가 임계상수 이하이면 요청을 수락하지만 임계상수 이상의 부하가 걸려있으면 가장 부하가 적은 후위 서버를 다시 선택하여 요청을 할당한다.

웹 문서의 참조 패턴은 Zipf분포[8]를 따르므로 특정 문서들은 다른 문서에 비해 매우 빈번하게 참조된다. LARD[1]에서는 같은 문서를 참조하는 요청들을 같은 후위 서버에 할당하므로, 요청 빈도수가 높은 문서를 캐쉬하고 있는 후위 서버는 항상 과적 상태가 될 수 있다. 이러한 문제는 요청을 수행시킬 자격이 있는 여러 개의 후위 서버들을 집합으로 관리하여 해결한다[5]. 같은 집합 내에 들어 있는 서버들은 같은 문서를 캐쉬하고 있으므로, 문서의 요청이 들어오면 집합 내의 후위 서버들중 가장 부하가 적게 걸린 서버가 서비스해 줄 수 있다.

3. 부하 분배기의 부하분배

WWW 개체는 모든 데이터를 개체로 간주하며 세션당 전송 개체의 크기가 고정되어 있지 않다. 후위 서버의 유효 컨넥션들이 멀티미디어 데이터의 서비스와 같이 큰 작업일 경우에는 그 수가 많지 않더라도 새로이 할당되는 작업은 지연될 가능성이 아주 크다. 역으로, 후위 서버의 유효 컨넥션의 수가 많지만 대부분이 작은 작업을 처리하는 컨넥션들이라면, 비록 과적 상태일지라도 요청을 이 후위 서버에서 수행하는 것이 사용자 응답 시간을 줄일 수 있을 것이다.

우리는 사용자 응답 시간을 줄이기 위해 지역성과, 작업량을 고려한 부하 분배 알고리즘을 제시한다. 아래의 알고리즘은 정적 웹 환경을 전제로 한다. 정적 웹 환경은 대부분의 작업이 디스크 읽기 작업이며 작업의 수행 시간은 작업량에 비례한다.

3.1 분배 비용

각 후위 서버의 부하 상태는 잔여 작업량을 고려한 유효 컨넥션의 수로 Rh와 Rl로 정의한다. Rh는 후위 서버가 과부하가 되는 시점을 의미하며 Rl 는 후위서버가 아이들(idle)해지는 시점을 말한다. Bcost는 후위 서버의 평균 잔여 작업량이다. Bcost는 부하 분배기가 후위 서버를 선택하는 기준이 된다. 가장 작은 Bcost를 갖는 후위서버가 선택된다.

$$Bcost(Sj) = \sum_{i=0}^{n-1} RWi$$

$$BS=\min(Bcost(S0),Bcost(S1)..Bcost(Sj))$$

j 번째 후위서버: Sj
후위 서버 Sj에서 유효컨넥션의 수: n
i 번째 요청의 잔여작업량: RWi
선택된 후위 서버: BS

3.2 알고리즘

```
while(true)
  fetch next request r;
  if serverSet[r.target] = ∅ then
     n, serverSet[r.target] <- { least loaded
node);
  else
     n<-{least loaded node in
```

serverSet[r.target]};
m<-{most loadednode in serverSet[r.target]};
if(n.node>R_{high} && ∃node with load < R_{low})
p<-{ least loaded node};
add p to serverSet[r.target];
n<-p;
if serverSet[r.target] > 1 && time() - serverSet[r.target].lastMod > K then
remove m from serverSet[r.target];
send r to n
if serverSet[r.target] changed in this iteraction then serverSet[r.target].lastMod <- time();

4. 실험 데이터

웹 문서는 시간적 지역성을 나타내며 요구된 문서의 빈도수는 zipf[8] 분포를 따른다. 서버에 존재하는 파일의 분포는 heavy-tailed[8]분포이다. 본 논문에서는 이러한 특징을 갖는 요청을 자동으로 생성해주는 생성해주는 SURGE[9]를 구동시켜 실제의 웹 환경과 유사한 모델로 실험한다. 클라이언트 머신은 250개~400개의 스래드(UE: User Equivalent)가 HTTP 요청들을 연속적으로 생성해 내어 서버가 고부하 상태가 되도록 한다.

5. 결론 및 향후과제

부하 분배는 분산된 웹서버 클러스터링 환경의 성능을 높이기 위한 매우 중요한 요소이다,. 우리는 부하 분배기를 이용해서 후위 서버의 지역성을 높이고, 부하를 공평하게 분배하는 알고리즘을 제시하였으며 Pentium III 800, RAM 256M의 linux ver 2.4 기반의 부하 분산 서버와 Pentium III 750, RAM 128M 2대 그리고 Pentium III 550, RAM 128M으로 구성된 후위 서버에서 구현 중에 있으며 기존의 알고리즘들과 정량적인 비교 실험을 할 예정이다. 일반적인 정적 문서는 디스크 액색스가 오버해드이므로 본 논문에서 제시한 지역성을 고려한 부하 분배가 주효하지만, 동적 문서의 수행에 있어서는 CPU 분배를 고려해야 한다. 그러므로, 동적 문서와 정적 문서가 섞인 작업들이 후위 서버에 가해지는 부하를 신중히 고려해야 할 것이다.

참고문헌

[1] "The Internet, Technology 1999, Analysis and Forecast", IEEE Spectrum (January 1999).
[2] Nina Bhatti and Rich Friedrich, "Web server support for tiered services", In *IEEE Network*, Hui Zhang and Edward W. Knightly, Eds. IEEE Communications Society, September 1999.
[3] N. Bhatti, A. Bouch, and A. Kuchinsky, "Integrating User-Perceived Quality into Web Server Design", In 9th International World Wide Web Conference (WWW9), Amsterdam, May 2000.
[4] Vivek S. Pai, Mohit Aron, Gaurav Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel and Erich Nahum, "Locality-Aware Request Distribution in Cluster-based Network Servers", In Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII) , San Jose, CA, October 1998.
[5] Mohit Aron, Peter Druschel and Willy Zwaenepoel, "Efficient Support for P-HTTP in Cluster-Based Web Servers", In Proceedings of the USENIX 1999 Annual Technical Conference, Monterey, CA, June 1999.
[6] Mohit Aron, Darren Sanders, Peter Druschel and Willy Zwaenepoel. "Scalable Content-aware Request Distribution in Cluster-based Network Servers", In Proceedings of the USENIX 2000 Annual Technical Conference , San Diego, CA, June 2000.
[7] C. S. Yang, M. Y .Luo, "Efficient Support for Content-based Routing in Web Server Cluster", In Proceedings of the 2nd USENIX Symposium on Internet Technologies & Systems. 1999
[8] Mark E. Crovella, Murad S. Taqqu, and Azer Bestravros, "Heavy-Tailed Probability Distributions in the World Wide Web. In A Practical Guide To Heavy Tails", chapter 1, pages 3-26. Chapman & Hall, New York, 1998.
[9] Paul Barford and Mark Crovell, "Generating Representative Web Workloads for Network and Server Performance Evaluation", In Proceedings of SIGMETRICS '98, pages 151-160, July 1998.
[10] A. Cohen, S. Rangarajan and H.Style, " On the Performance of TCP Splicing for URL-Aware Redirection", In Proceedings of the 2nd USENIX Symposium on Internet Technologies & Systems. 1999
[11] Emiliano Casalicchio, Michele Colajanni, "A Client-Aware Dispatching Algorithm for Web Clusters Providing Multiple Services", In Proceedings of the Tenth International World Wide Web Conference, May 2001