제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

재전송 타임아웃 간격의 범위 조절에 의한 Web 서버의 성능향상

김진희 권경희
단국대학교 전자계산학과
whitej28@hotmail.com, khkwon@dankook.ac.kr

Improving a Web Server Performance By Modifying Interval of Retransmission Timeout

Jin Hee Kim , Kyung Hee Kwon
Computer Science, Dankook University

요 약

본 연구는 접속 요구가 많은 웹 서버의 부하를 두개의 웹 서버로 분산시켜 보다 효율적인 웹 서비스를 제공함과 동시에, 두 웹 서버의 용도에 맞게 재전송 타임아웃(RTO:Retransmission Timeout)을 독립적으로 설정하여 네트워크의 효율성을 제고하기 위해 시도되었다. 이를 위해 Linux 기반의 PC에 라우터를 구축하고, 접속의 요구에 따라 두개 중 하나의 Web 서버에 접속되게끔 라우터를 설정하였다. 웹 서버는 내부 접속용과 외부 접속용으로 구분하였으며, 각각의 서버마다 재전송 타임아웃(RTO:Retransmission Timeout)값을 다르게 설정하여 네트워크에서의 지연(delay)을 최적화시켜 클라이언트의 요청에 대한 응답시간을 최소화시켰다.

전송거리가 짧은 내부 접속용 웹 서버에는 패킷 손실이 거의 없으므로 작은 RTO 값을 설정하고, 외부 접속용 웹 서버에는 큰 RTO 값을 설정하였는데 RTT(Round Trip Time:응답시간)와 Tput(Throughput:처리율)의 출력 결과를 통해 Web 서버의 성능 향상을 확인 할 수 있다.

1 서론

1.1 연구목적

인터넷 사용을 하면서 경험할 수 있는 것 중의 하나가 사용자의 증가로 인한 많은 접속 요구에 따라 Web 서비스가 그에 부응하지 못한다는 것이다. 이러한 예로서 느린 접속 속도 및 시간초과로 인한 재접속 등을 들 수 있으며 이는 서버의 과부하를 유발시킨다. 이러한 서버의 과부하는 패킷 손실 및 서비스 지연, 서버 다운 등의 문제로 이어지면서 서버 분산의 필요성을 인식시켰다. 본 논문에서는 학교 내부에서의 접속시 빠른 응답속도에 비해 학교 외부에서의 접속시 느린 응답속도에 대한 문제점의 개선을 위해, Web 서버의 부하 부산방법을 이용하였다. 서버 부하 분산방법으로 리눅스용 라우터를 구축하였으며 학교를 기준으로 내부와 외부 IP를 구분하여 서버에 할당하는 방법으로 트래픽을 분산 처리하게끔 하였다.

TCP는 UDP와는 달리 신뢰성을 보장하는 전송 방식으로서 연결된 두 종단 사이에서 데이터 패킷의 송·수신시에 데이터의 손실 혹은 과부하로 인한 전송지연으로 인해 특정 시간내에 ACK(Acknowledge)를 받지 못하면 타임아웃을 적용, ACK가 없는 패킷에 대하여 다시 한번 전송이 이루어진다. 이때의 타임아웃의 주기 및 재전송 발생 간격 등은 네트워크상에서 지연에 영향을 미칠 수 있다. 데이터 전송시 전체적인 지연은 전송거리에 비례한다고 할 수 있다. 거리가 길면 당연히 전송시간도 길게 마련이며 거리가 짧은 내부용 서버보다 재전송 간격의 최대치를 길게 두어야 함을 의미한다. 이러한 지연은 RTO의 간격이 좁아서 지나치게 많은 재전송이 이루어지면 네크워크의 과부하를 더욱 조장할 뿐만 아니라 너무 긴 RTO 간격은 더욱 긴 지연을 가져올 뿐이다.

RTO 간격을 TCP parameter의 변경을 통해서 조절해 봄으로써 최대 전송 지연을 최적화 시키고 네트워크의 효율성 향상을 목적으로 하였다.

1.2 연구내용

본 논문에서는 특정 서버의 접속이 내부에서 이루어지는 접속인지 혹은 외부에서 이루어지는 접속인지를 구분하여 서버를 내부 접속용 서버와 외부 접속용 서버로 나누었다. 이는 일반적인 Web 서버의 접속이 대부분이 외부 접속인데 반해 내부 접속과 외부 접속의 빈도가 비슷한 환경 하에 있는 서버를 대상으로 하였으며 본 논문에서는 단국대학교 전자계산학과내의 서버를 연구대상으로 하였다.

기존의 Solaris 서버에 Linux 서버를 하나 더 추가 시켜 트래픽이 분산되게끔 하였으며, 이러한 트래픽 분산을 위하여 Linux가 탑재된 PC를 사용하여 라우터를 구축하였다.

라우터를 경유하여 들어오는 IP를 외부 접속시에는 Solaris에 내부 접속시에는 추가된 Linux 서버에 접속되게 하고 서버 분산 전·후의 트래픽량과 cpu 이용률을 계산해 본다.

또한 네트워크 성능을 향상시키기 위해서 서버 분산 후의 Solaris 서버와 Linux 서버 각각의 RTO 최소, 최대치

값을 조절하여 역시 조절 전·후의 응답시간인 RTT(Round Trip Time: 연결된 두 종단 사이에서 데이터 패킷의 송신에 대한 수신 응답 시간)와 처리율 Tput(throughput:주어진 시간동안 처리하는 비트 수)의 변화를 알아본다. 이는 교내에서의 접속은 전송지연에 있어서 전송거리가 짧으므로 서버의 과부하나 패킷 손실 시 RTO의 대기 시간이 외부 접속에 비해 짧아야 한다는데 기인한 것으로 외부 접속용으로 이용되는 Solaris는 RTO값을 내부 접속용 Linux보다 크게 줌으로써 전송 지연을 최소화 해 볼 수 있다.

2. 기술 및 시장동향

서버 분산의 필요성은 이미 서론에서 언급되었듯이 인터넷의 수요급증에 따른 Web 서버의 부하가 급증하면서 대두된 사항으로 네트워크 설계시 필수 요소로 자리잡은 지 오래이다. 그러나 80년대 이후 분산처리 방식이 급류를 타면서 지나치게 많은 분산이 이루어졌으며 이는 관리의 어려움과 비용 증가등의 문제점을 야기시켰다. 이에 최근에는 분산시켰던 다량의 서버를 가능한 적은 수의 서버로 다시 모으는 시스템 재통합이 붐을 타고 있기도 하다. 그러나 여전히 서버 분산의 필요성은 중요하게 인식되고 있으며 서버분산의 방법으로 많이 이용되는 것 중의 하나로 로드밸런싱을 들 수 있다. 로드밸런싱이란 Web 서버의 성능이 사용자의 요구를 따르지 못하는 것에 대하여 서버를 upgrade 한다거나 하는 복잡하고 비용에 있어서의 부담을 줄 일 수 있는 방법으로 n개의 서버에서 같은 Web 서비스를 운영하면서 부하가 증가하게 되면 서버만 늘려서 부하를 분산시켜 성능을 향상시키는 방법이다. 현재까지 출시된 제품으로는 하드웨어 기반, 소프트웨어 기반, 로드밸런싱 스위치로 나눌 수 있으며 Web 서버를 분산시키는 방법으로는 동일한 Web 서비스를 여러 개의 서버로 나누어서 서비스 하는 방식이 있는가 하면, port별로 나누어서 서비스하는 방식도 있다. 예를 들어 mail, ftp, telnet등으로 나누어진 방식이 그것이다.

본 논문에서는 가상 서버를 이용한 소프트웨어 기반의 로드밸런싱에 관하여 간단히 설명해 본다.

로드밸런스를 이용한 방법은 클라이언트가 서버에 접근하려 할 때 로드밸런서에서 서버 클러스터가 한 개의 IP를 가진 하나의 가상서버로 보이게 하는 것이다. 물론 실제로는 서버 밑에 여러 개의 서버가 위치하고 있다.

가상서버의 세가지 방식에는 NAT 기반의 가상서버, IP 터널링 이용한 가상 서버, 다이렉트 라우팅을 이용한 가상서버가 있으며 이러한 가상서버에서 서버들을 선택하는 4가지 스케줄링 방법이 있다.

- 라운드 로빈 스케줄링

모두 동등하게 연결을 돌아가면서 할당하는 방식으로 기존의 RR-DNS와 비슷하지만 하나의 도메인 이름을 여러 개의 IP주소로 바꾼다는 점과 호스트 기반의 스케줄링이라는 점 등 캐싱 기능에 영향을 미친다는 점 때문에 실제 서버들간의 로드 불균형을 초래하게 되는 단점이 있다.

- 가중 라운드 로빈 스케줄링

클러스터에 속한 서버들의 처리 용량, 능력에 따라 가중치를 두어서 할당하는 방법으로 서버별 연결되는 횟수가 계산되지 않으므로 불균형이 초래할 수 있다.

- 최소 커넥션 스케줄링

글자 그대로 서버에 연결된 수가 가장 적은 것을 우선으로 스케줄링한다. 서버별 성능이 비슷한 클러스터에 적합하다.

- 가중 최소 커넥션 스케줄링

서버별 가중치를 두어서 큰 가중 값을 가진 서버가 실제 커넥션의 많은 부분을 차지하게 되는 것으로 서버별 성능이 다른 클러스터에 적합하다.

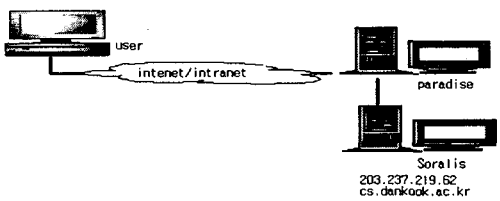
위의 알고리즘을 이용하여 서버의 과부하를 체크 하여 밸런스를 유지시켜 서버에 할당하는 방식을 로드밸런싱이라하며 본 연구 논문은 위의 NAT와 비슷한 방식으로 운영체제의 종류에 관계없이 실제 서버로 사용할 수 있다. IP의 사용시 부하분산용 서버만 공인 IP를 사용하며 나머지 Linux 라우터 밑에 보이지 않는 실제 서버가 2대 놓이며 이 서버에는 가상 IP라하여 내부 IP를 사용한다.

내부 IP는 인터넷 주소 할당기구(The Internet Assigned Number Authority:IANA)에서 IP 주소중 3가지 영역을 두어 개인용 네트워크를 위하여 예약된 주소이다. 본 논문에서는 192.168.10.0 네트워크와 255.255.255.0의 class-C 서브넷 마스크를 사용하였다.

3. 네트워크 구성

기존의 네트워크 구성이 그림 [1]과 같다. 전자계산학과의 Web 서버인 Solaris에서 http, smtp, ftp, telnet 모두를 서비스 해주고 있다. 이에 서버 분산을 통해 파라다이스 서버에서 게시판 및 DB 부분을 별도로 운영할 것이며 static한 Web 서비스는 Linux와 Solaris 두 개의 서버에서 분산처리 할 것이다. 실시간 처리를 요하는 게시판은 따로 분리하여 파라다이스로 이동시켰다. 이를 위한 네트워크 구성이 그림 [2]와 같다.

리눅스 라우터를 만들기 위해서 기존의 리눅스에 Lan 카드 두장을 설치하였다. 하나는 공인 IP(203.237.219.62)를 잡아줄 것이며 또 다른 하나는 내부 IP(192.168.10.10)를 잡아주는 것으로서 라우터 밑의 두개의 서버를 연결시켜주는 내부 인터페이스의 역할을 할 것이다. 여기서 중요시 할 것은 두 개의 실제 서버에서의 게이트웨이로 반드시 Linux 라우터가 되어야 하며 리눅스 라우터의 경우는 기존의 게이트웨이가 그대로 이용되어야 한다는 것이다. 이렇게 준비된 Linux 배포 본에 마스크레이드에 관계된 모듈들이 컴파일 되어져 있다면 커널 컴파일을 할 필요가 없으나 그렇지 않다면 다운로드 후 설치, 커널 컴파일을 통해서 라우터의 역할을 할 수 있는 리눅스를 구축해야 한다..



그림[1] 서버 분산전의 네트워크 구성

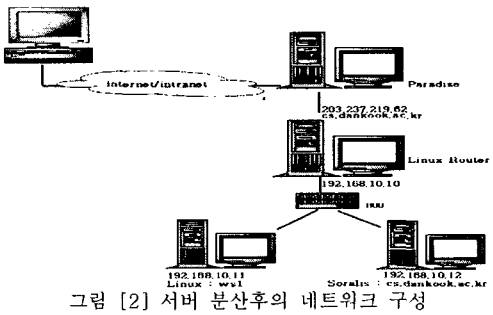


그림 [2] 서버 분산후의 네트워크 구성

4. 네트워크 성능 분석

네트워크의 성능을 분석한다는 것은 크게 물리적인 장치의 장애, 성능 등을 비교 분석하는 것과 사용자의 관점에서 서비스 등의 특성을 통한 분석 등을 생각 할 수 있다. 이러한 예로서 장치의 처리율(Tput :Throughput)과, 응답시간(RTT:Round Trip Time)등을 들수 있으며 본 논문에서는 분석 방법으로 네트워크 트래픽, Tput,CPU이용률과 RTT등을 살펴 보았다.

우선 packet을 검사하는 툴의 일종으로 Solaris에서는 snoop를 Linux에서는 tcpdump를 사용하였다. 이렇게 수집된 자료는 tcptrace라는 통계 분석 툴을 통하여 tcp connection에 대한 정보를 얻을 수 있다. Log file 분석 툴로는 Web Trends log Analayzing를 이용하였다.

4.1 RTO의 최소 최대값의 범위 조절 방법

본 논문에서 이용된 두개의 서버는 각기 서로 다른 운영체제를 사용하고 있다. 따라서 TCP parameter의 수정을 통한 RTO의 제한값을 설정하는 방법도 서로 다르고 여기서는 각 서버별, Linux와 Solaris에서의 parameter의 설정 방법을 설명하고 변경 전·후의 RTT를 비교해 본다.

```
cs# ndd  -set  /dev/tcp tcp_rexmit_interval_min 200
cs# ndd  -set  /dev/tcp tcp_rexmit_interval_max 120000
```

그림 [3] Solaris에서의 RTO 설정 방법

일반적으로 RTO의 최대값으로 적당한 수치를 64000ms로 정의하고 있다.7) 이에 Default로 잡혀있는 Solaris의 RTO 200ms를 외부 접속용임을 고려하여 120000ms과 240000ms으로 수정하였다

```
Static __inline__void tcp_bound_Rto(struct tcp_opt
*tp)
  {
    if (tp->Rto > 120*HZ)
          tp->Rto=120*HZ;
      if (tp->Rto < HZ/5)
  tp-> Rto = HZ/5;
  }
```

그림 [4] Linux에서의 RTO 설정 방법

Linux는 소스화일에서 parameter 값을 변경 후 커널 컴파일을 통해서 적용한다.

앞서 재전송의 발생 간격 등은 네트워크 상에서 지연에 영향을 미칠 수 있다고 밝힌 바 있다. 타임아웃의 간격이 좁으면 잦은 재전송으로 인한 네트워크의 과부하를 더욱 조장할 뿐만 아니라, 너무 길면 더욱 긴 지연만 가져올 뿐이다. 따라서 적절한 수치는 전체적인 지연에 영향을 미치게 될 것이다.

4.2 네트워크 성능 분석

아래의 그림은 분산 이후 각각의 두 서버(Soralis와 Linux)에 RTO값을 다르게 적용시켰을 경우 RTT와 Tput의 변화를 그래프로 그려본 것이다.

외부 접속용으로 이용된 Soralis 서버의 그래프를 통해 웹서버의 성능향상을 확인해 볼 수 있다.

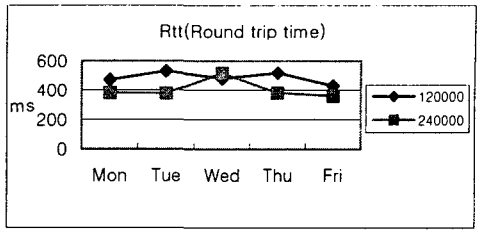


그림 [5] Solaris에서 RTO변화에 따른 RTT

RTO의 디폴트 값으로 240000ms일때와 120000ms으로 바꾼 후의 그래프의 변화를 비교해 볼 수 있다. 오히려 240000ms로 주어졌을때가 120000ms로 주어졌을 때보다 RTT값은 더 낮고 Tput은 더 높은 수치를 나타남을 알 수 있다. 이는 Solaris 서버가 외부 접속용 서버로 이용된 만큼 120000ms이라는 수치는 전송 거리 등을 고려했을 때 작게 설정 된 것으로 짐작 할 수 있으며, 이는 오히려 전체적인 응답속도를 저하시킴을 확인 할 수 있다.

1 주일을 기준으로 전체적인 트래픽량이 수요일에 가장 많고 금요일이 가장 적게 나타남을 확인 할 수 있었으며 이러한 트래픽량은 그림 [5]의 RTT에서도 확인 할 수 있다. 다른 요일에 비해 트래픽이 많은 수요일은 120000ms, 240000ms 라는 값이 큰 의미가 없다는 것을 확인 할 수 있다.

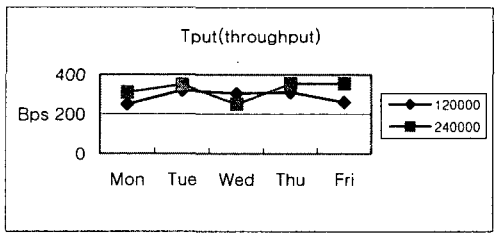


그림 [6] Solaris에서 RTO변화에 따른 Tput

그림 [6]은 그림 [5]를 통해서 확인하였듯이 평균 RTT가 짧아지면서 주어진 시간동안 처리하는 비트수(처리율)가 증가하였음을 나타낸다.

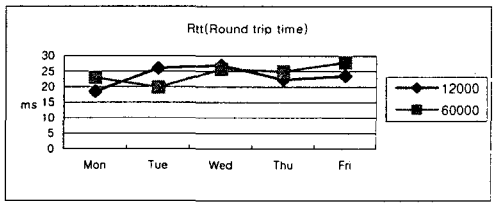


그림 [7] Linux에서의 RTO변화에 따른 RTT

Solaris 서버에서와는 다르게 크게 RTT의 변화를 볼 수 없다. 이는 송·수신 거리가 짧은 내부 접속만 이루어지므로 전송지연이라든가 패킷손실이 없는 상태에서는 재전송도 이루어지지 않으므로 재전송 간격이 전체적인 RTT에 미치는 영향이 없음을 의미한다. 단지 이동되는 패킷양에 의해서 RTT의 평균값이 달라질 뿐이다.

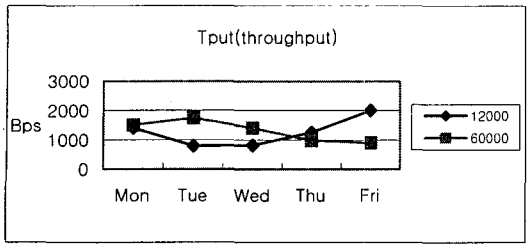


그림 [8] Linux에서의 RTO변화에 따른 Tput

6. 결론

네트워크의 성능을 분석하는데 있어서 가시적으로 확인할 수 있는 것이 RTT이다. 이는 사용자의 입장에서 느껴지는 것으로서 서비스 처리시간을 뜻하며, 이외에도 많은 것들, 처리율, 장애율, 인터페이스 이용률 등이 있으나 이는 눈에 보이는 것이 아니기 때문에 일반적으로 RTT에 대한 분석을 네트워크 성능 분석에 있어서의 첫 번째로 생각하는 것이다.

본 논문에서는 Web 서버의 분산을 통해 RTT의 성능 향상을 시도해 보았다. 기존의 서버 분산 방법은 일반적으로 부하량을 체크하여 부하가 많은 곳의 서버를 피해서 다른 서버에 할당시키는 로드밸런싱 방식이지만 본 논문에서는 부하량 체크를 통한 서버별 분배 방식이 아니라 학교라는 상황을 고려하여 Web 서버를 학교 내부용과 외부용으로 구분하였다. 이는 일반적인 Web 서버 같은 경우 외부 접속이 다수인 반면 학교 같은 경우에는 서버 이용률이 내부와 외부로 나누어지고 특별하게 어느 한쪽으로 기울지 않는다는 상황을 고려한 것이다.

내부 접속용 서버에서는 너무 크지 않은 RTO값을, 외부 접속용 서버에서는 너무 작지 않은 RTO 값을 정한다는 것은 잦은 재전송으로 인한 패킷의 과부하를 피하기 위함이며, 역시 최대 값이 너무 작게 설정되면 느린 인터넷 환경하에서는 최대 대기 시간의 부족으로 전송이 제대로 이루어 지지 않을 수 있다. 연구 실험 결과 실제로 외부접속용 서버로 이용된 Solaris서버에서는 RTO의 최대치 값이 가져다 주는 의미가 적지 않다는 것을 확인할 수 있었다. 실제로 외부 접속용 서버에서는 120000ms로 설정 됐을 때보다 240000ms로 설정돼 있는 것이 더 효율적임을 평균 RTT와 Tput을 통해서 확인 할 수 있었으며 120000ms로 설정돼 있을 때의 RTT가 더 높게 나타난 것은 전송거리에 비례하여 전체적인 지연이나 패킷손실시 이루어지는 RTO 간격의 최대치 값이 너무 작게 설정되어 있으므로, 잦은 재전송을 유발하여 오히려 좋지 않은 결과를 가져다줌을 확인할 수 있었다. 반면에 내부용 서버로 이용된 Linux용 서버 같은 경우는 RTO가 주는 의미가 크지 않다는 것을 확인 할 수 있었다. 더욱이 전자계산학과 서버를 사용하는 경우는 같은 건물 내에서의 접속이 많은 경우로서 네트워크상에서의 물리적인 문제점으로 인한 전송지연 이라든가 패킷손실이 거의 없으며 따라서 재전송의 발생 빈도가 0%에 가깝기 때문이라고 판단 할 수 있었다.

이로써 서버의 과부하시 패킷손실이나 전송지연이 발생하는 경우, 학교와 같은 서버 접속 그룹이 내부와 외부로 그리고 그 접속 비율이 비슷하게 나누어지는 경우에는 본 연구의 내용이 어느 정도의 효율성을 가져다 줄 수 있다는 결론을 얻을 수 있었다.

참고 문헌

[1] Larry L.Peterson, Bruce S. Davie, "Computer Netw-orks", pp. 345-349
[2] Douglas E. Comer, "Computer Networks and Internets"
[3] Stallings, "Data and Computer Communications"
[4] 강맹규, "네트워크와 알고리즘"
[5] 권경희, "웹 엔지니어링"
[6] "TCP/IP 동작원리와 트러블 대책", 동서 네트워크 연구원
[7] Craig Hunt, "TCP/IP administration", Addison Wesley Publishing Company
[8] W. Reichard Stevens, "TCP/IP Illustrated Volume1", Addison Wesley, pp. 223-356, 491-498, 525-538
[9] Gary R. Wright W. Reichard Stevens, "TCP/IP Illustrated Volume2(1)", Addison Wesley
[10] W.Reichard Stevens, "Tcp/IP Illustrated Volume2(2)", Addison Wesley, pp817-848
[11] W.Reichard stevens, "TCP/IP Illustrated Volume3", Addison Wesley, pp. 91-158
[12] R MAGNUS, U KUITX, M DZIADZKA D VERWORNER, MBECK, H NOHME, "LINUX KERNEL INTERNALS", Addison Wesley
[13] "네트워크 분석 도구의 종류와 기능" <http://www.infoage.co.kr/lantimes/9804/lecture.html>
[14] "네크워크 분석을 위한 사례 연구 <http://www.infoage.co.kr/lantimes/9805/lecture1.html>
[15] "AIX Newsletter 제 120호" <http://www.kr.ibm.com./products/rs6000/techinfo/aixn112.htm>
[16] "TCP/IP and Routing" <http://poem.cheuj.ac.kr/document/hckim/semina/sa/sa/node4.html>
[17] "Linux Virtual Server Project", "Linux IPCHAINS-HOWTO", "IP Masquerading" <http://www.kldp.org/KoreanDoc/html//Virtual_Server...>
[18] "Tcptrace" <http://jarok.ca.ohiou.edu/software/tcptrace/tsg.html>