클러스터 웹 서버에서 컨텐츠인식 부하 분산 모델

최면욱° 현종웅 정인범
강원대학교 컴퓨터정보통신공학과
{muchoi°, jbjung}@snslab.kangwon.ac.kr, jwhyun@calab.kaist.ac.kr

Content Aware Based Load Distribution in a Cluster Web Server

Myunuk Choi° Jongwoong Hyun Inbum Jung
Dept. Computer Information & Telecommunication Engineering

요 약

최근 클러스터 웹 서버의 로드 벨런싱으로서 L4 스위치가 널리 사용되고 있다. 전형적인 L4 스위치는 연결 요구 발생시 back-end 노드들의 부하를 평가하여 적은 부하를 갖고 있는 노드로 요청을 보내는 기능을 수행한다. 그러나 L4스위치의 부하 분산은 사용자 연결 횟수를 이용해 분산함으로서 사용자의 다양한 컨텐츠 요청에 적절한 부하분산 스케줄링을 적용하고 있지 않다. 본 논문에서는 서버에 전송되는 컨텐츠 요구를 L4스위치 계층에서 인식하여 부하배분에 사용하는 부하분산 모델을 제시한다.

1. 서 론

인터넷 사용의 급속한 성장은 인터넷 서버 시스템에 커다란 변화를 가져왔다. 대용량의 처리 규모를 가진 웹 서버라 할지라도 연속적이고 불균형적인 사용자 요구 증가로 인해 서버의 처리량을 넘어서고 있다. 이로 인해 확장성, 고가용성, 경제성을 지닌 클러스터 웹 서버 형태로 시스템이 전환되게 되었다[1,2,3]. 초창기의 웹 서버의 대부분의 트래픽은 비교적 작은 정적 파일들이 차지했다. 하지만 오늘날 대부분의 트래픽은 그래픽 데이터와 다양한 크기를 갖는 멀티미디어 데이터가 차지하고 있다[4]. 특히 최근에는 전자 상거래 같은 새로운 응용프로그램에 대한 사용자 인터페이스로 웹 사용이 증가함에 따라 다양한 컨텐츠에 대한 사용자 요구 비율이 점차 증가하고 있다[5,6]. 이러한 인터넷 환경 변화들은 클러스터 웹 서버 사이의 부하 분산정책에 대한 새로운 연구를 필요로 하고 있다.

클러스터 웹 서버에서 구조적 투명성과 중앙 집중제어의 용이성 때문에 최근에 L4(Layer 4) 스위치들이 부하 분산을 위한 라우터로 널리 이용되고 있다[7,8,9]. 그러나 L4 스위치에서는 부하 분산의 방법으로 사용자 연결 횟수만을 사용하고 있으며, 사용자 요청 데이터의 종류 및 각 노드들의 캐쉬에 있는 내용은 스케줄링에 반영되고 있지 않다. 본 논문에서는 L4 스위치 수준에서 사용자 요구 내용을 기반으로 back-end 노드들의 CPU 및 디스크 부하를 평가하여 부하분산 문제를 효율적으로 처리하는 방법을 제안하고자 한다.

2. 관련 연구

L4 스위치는 하드웨어 또는 소프트웨어적으로 구성될 수 있다. 이런 L4 스위치가 위치한 노드를 front-end 서버라고 하며 외부적으로 VIP(Virtual IP) 주소를 가지고 있고 클러스터를 구성하는 back-end 노드의 나머지 노드들을 대표하게 된다. 하나의 실제 서버주소를 대체하는 VIP 주소는 DNS(Domain Name Server) 안에 Application Server 주소로 등록되어지고 모든 패킷은 VIP를 통하여 서비스를 받게 된다[10].

L4 스위치 수준에서 IP 변경하는 방식으로는 TCP 라우팅 메커니즘에서 NAT(Network Address Translation)방식 또는 패킷 포워딩의 두 가지 방법 중 하나로 구성된다. NAT 방식은 실제 서버로 오가는 모든 패킷의 주소를 재작성해야 하므로 병목현상이 발생할 수 있기 때문에 클러스터를 구성할 수 있는 서버의 개수에 제한을 받는다. 일반적으로 선택된 서버로 직접 라우팅하는 패킷 포워딩 방식이 확장성과 라우팅 처리율이 우수한 것으로 알려져 있다[10].

L4 스위치에서 부하 분산 정책으로는 패킷이 도착하는 순서대로 back-end 노드들에게 차례로 분배해주는 라운드 로빈(Round-Robin) 방식, 서버마다 서로 다른 가중치를 두어 분산하는 방식인 가중치 라운드 로빈(Weighted Round-Robin) 방식, 서버에 연결된 서비스 요청 개수에 따라 분산하는 방식인 최소 연결(Least-Connection) 방식 등이 있으며 이들 중 WRR 방식이 구현의 어려운 단점이 있지만 다른 정책보다 전반적 성능이 우수함을 보이고 있다[11].

L4 스위치에서는 back-end 노드의 부하 측정을 기반으로 부하 분산을 하기 때문에 그로 인한 과부하나 측정된 정보들의 우선순위를 결정하는 것이 쉽지가 않다. 이러한 L4 스위치의 제기된 문제에 대한 대안으로 최근에 L7 스위치의 컨텐츠 인식 라우터에 대한 연구들이 제시되고 있다. L7 스위치 방식은 사용자 연결을 우선 설정 후 새로운 연결마다 사용자 요구 내용을 파악하여 적합한 서비스를 제공한다. 하지만 L7 스위치는 클라이언트 연결 설정을 유지하기 위한 과부하, L7 프로토콜 처리시간 그리고 최종적으로 정해진 노드로 연결하는데 따른 과부하로 인해 일반적인 부하수준에서도 L7스위치 자체가 병목현상을 발생하는 것으로 알려져 있다[12].

이러한 현재 웹 서버의 부하 분산을 위해 사용되는 L4 및 L7 스위치 방식은 각각 장단점을 가지므로 이들의 장점을 살릴 수 있는 통합적 형태의 부하 분산 방법의 연구가 필요하다. 본 연구에서는 L4 스위치처럼 back-end 노드의 부하 측정을 이용하지 않고 L4 기반에서 L7 방식의 컨텐츠 인식을 하여 공정한 부하 분산을 하고자 한다.

3. 컨텐츠 인식 기반 부하 분산

3.1 주요 개념

이 장에서는 L4 기반에서 컨텐츠 인식 기반의 부하 분산 방법을 제안한다. N개의 back-end 클러스터 서버와 커널수준에

* 본 연구는 한국과학재단 목적기초연구(R05-2003-000-12146-0) 지원으로 수행되었습니다.

서의 패킷 포워딩 방식을 사용하는 L4 스위치에서 동작하는 것을 기준으로 한다. 모든 back-end 노드들은 임의의 컨텐츠 요구에 동등하게 응답할 수 있으며 프로세스 구동 모델을 따르는 Apache 웹서버 프로그램이 구동되는 것을 가정한다[4].

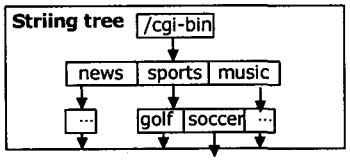
일반적으로 클라이언트 요청은 서버에서 순차적 단계의 처리 과정을 거치게 된다. 웹 서버에서 사용자의 요구가 서버에 도착하게 되면, 요구를 처리하는 과정에서 소모되는 시간은 고정된 부분과 비고정 부분으로 나누어지게 된다. 고정 부분은 운영체제 커널에서 소모되는 CPU 시간과 패킷 연결설정, HTTP 헤더분석, 요청 로깅(loggin)과 같은 부분이고 비고정 처리시간은 요구된 컨텐츠의 종류에 따라서 디스크 및 메모리로부터 파일의 인출, 동적 내용물들의 생성, 응답 결과물의 전송등에 따라 달라진다.

본 논문에서 제안하는 CALD(Content Aware Load Distribute)는 사용자 요청 패킷 정보와 미리 측정된 과부하 데이터를 비교하는 기능을 사용하여 클러스터 웹 서버의 부하 배분을 한다. 컨텐츠는 정적 파일들은 크기에 따라 분류하고 동적 파일들은 URI 패턴 및 기능 수행을 위한 응용프로그램의 특징에 따라 분류한다. 모든 컨텐츠들은 웹 서버 구축시 컨텐츠 내용을 알 수가 있기에 컨텐츠에 따른 동적, 정적 컨텐츠들의 부하량을 미리 계산할 수가 있다. 이런 분류된 종류들에 대하여 각각에 대한 평균 처리시간을 측정하고 L4스위치 내부에 테이블 자료구조 형태로 저장한다. 따라서 L4 스위치 서버로 들어오는 사용자 요청 패킷을 분석하고 각 컨텐츠들의 소요 시간을 미리 측정한 부하측정 테이블과 비교하여 최적의 back-end 노드로 라우팅하는 것이다.

3.2 컨텐츠 인식

사용자 요청에 의한 모든 패킷은 TCP 라우터에 의해 라우팅된다. HTTP 요청이 들어오면 소프트웨어 L4스위치의 TCP라우터에서 패킷의 기본적인 정보를 가지고 있기 때문에 이러한 요청을 커널 차원에서 분석하여 미리 측정된 부하 테이블과 비교하여 적절한 back-end 노드에 보내게 된다. 부하측정 테이블은 CPU소모 시간인 ccost, 디스크에서 불러들이는 시간인 dcost, 파일 크기를 나타내는 size로 구분을 하고 각 URI의 소모 시간을 미리 계산을 하여 부하측정 테이블을 만든다. 이 테이블은 URI에 의해 전역 해쉬 테이블로 인덱스화 되어 있고 사이트의 모든 정적 파일들의 엔트리를 포함한다.

Input URIs:
uri1 /cgi-bin/sports/soccer?event=WorldCup2002
uri2 /cgi-bin/sports/soccer?country=Italy&league=Serie-A
uri3 /cgi-bin/music/chart?genre=Pop&year=2002&week=17
uri4 /cgi-bin/music/album?genre=Pop&artist=Beatles



0	ccost=120 dcost=0 size=8K
1	ccost=150 dcost=50 size=8K
2	ccost=210 dcost=50 size=8K
2	ccost=210 dcost=50 size=8K

그림 1. 부하측정 테이블에서의 클래스 탐색 예

웹 서버에서 부하발생은 컨텐츠의 경로에 의해서가 아닌 어플리케이션에서 발생한다. 그림1을 보면 uri1과 uri2는 경로는 같지만 어플리케이션이 event, content로 다르고 uri3과 uri4는 경로는 다르지만 genre로 어플리케이션이 같다. 따라서 uri1과 uri2는 다른 클래스에 속하고, uri3과 uri4는 같은 클래스에 속하게 된다. 동적 URI가 인식되었을 때, 스트링 트리에서 경로 이름과 어플리케이션 이름 그리고 파라메터 식별자를 클래스 테이블에서 찾아보고 그 클래스의 부하 데이터를 가지고 부하량을 예측하게 된다. 따라서 L4스위치 기반에서 사용자 요청 패킷의 컨텐츠를 인식 하고 부하측정 테이블과 비교하여 적절한 서버에 분배를 할 수 있게 된다.

4. 실험 및 분석

4.1 실험 환경

WRR 방식이 다른 정책보다 전반적인 성능이 우수함을 보이므로 CALD 방식과 비교하였다. 실험에서 사용된 WRR 방식은 연결된 수, 서버응답시간, 디스크로부터 데이터를 읽고 처리하는 시간의 세 가지 변수를 가지고 측정하였다[9]. '98 동계 올림픽 사이트의 공식 로그를 얻어 내용을 분석하였고 부하 분산 성능 측정에 사용될 URI 패턴들의 부하 분석을 한 결과 HTTP/1.1 프로토콜에 의한 요구가 20% 정적인 내용물 요구가 70% 정도를 차기하고 있음을 알 수가 있었다. 로그 분석을 통해 얻은 패턴들을 바탕으로 부하측정 테이블을 만들었다.

시뮬레이션 환경으로는 back-end 노드로 구성된 Apache 1.3.16 웹 서버를 사용하였고, P3 700MHz, 256M 메모리, 5,400RPM 하드디스크로 구성된 서버로 측정 하였다. 그림2를 보면 시뮬레이터의 부하 측정에서 서버 가중치 W_K는 각 서버당 ccost와 dcost를 합하여 부하를 누적한 L_K 값을 가진다. 그리고 각 서버의 가중치 W_K는 계속 누적되는 L_K 값을 줄이도록 하여 부하측정 누적 값이 threshold를 넘지 않도록 했다. threshold 값도 일정 기간동안 요청되는 수에 따라서 동적으로 조절할 수 있게 하여 부하 특성에 따라 동적으로 최적화 되도록 하였다.

```
/* 부하 측정의 통계 */
N_r - 일정기간 요청 개수      L_K - 부하 누적량
W_K - 서버 가중치             N - 노드 수

/* 서버가중치 측정 */
for (min=max=k=0; k<N; k++)
  if (L_K > max)        max = L_K;
  else if (L_K < min)   min = L_K;
t_diff = N_r / N;
Dec =(max-min<= t_diff) ? 1.0 : t_diff / (max_min);
For(k=0; k<N; k++)
W_k=(max.allowable weight) - (L_k-min)*dec
```

그림 2. 부하측정 서버 가중치 예제

4.2 실험 결과 및 분석

그림3은 정적 파일들의 요청 수를 증가시켜 서버응답 시간을 측정한 것이다. CALD 값이 WRR 과 비교해보면 그리 커다란 성능 변화는 없는데 정적 파일들로 이루어 진 경우 부하 분배 알고리즘에 따른 큰 차이점이 없음을 알 수가 있다. 이는 정적 화일의 경우 동적 파일 요청에 비해 CPU처리가 적고 파일 크기에 따라서 소요시간이 비례하기 때문이다.

두 그래프 모두 클라이언트 요청이 증가함에 따라 정적 파일에 대한 캐쉬 적중률 향상으로 CPU와 메모리 오버헤드가 줄어들어 성능향상을 이루다가 일정 시점에서 서버 응답시간이 크게 늘어나는 것을 볼 수가 있다. 이는 서버의 부하 분배량의 threshold값을 넘어 서서 발생하는 문제인데 갑자기 서버응답 시간이 크게 증가하는 이유는 클라이언트에서 보낸 요청이 응답을 못 받았을 때에 다시 재 전송하는 특성으로 인한 것으로 해석된다.

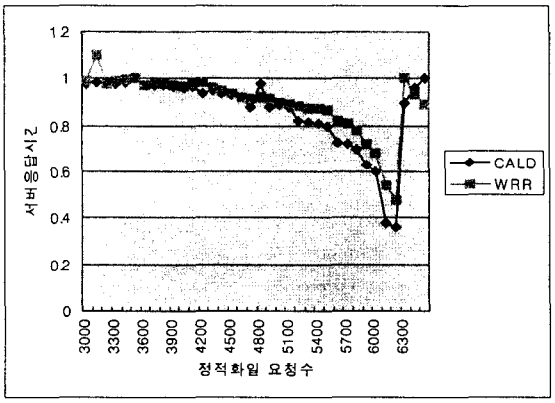


그림3. 정적파일 평균처리율 측정결과

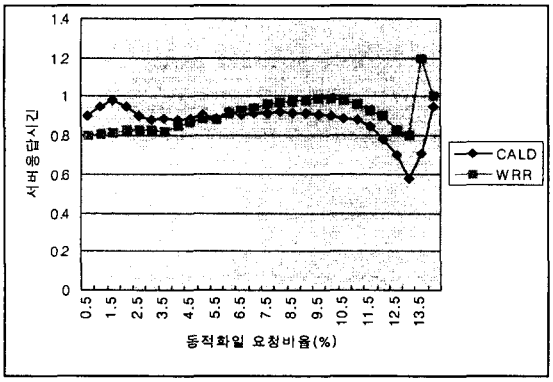


그림4. 동적 요청파일 평균처리율 측정결과

그림 4에서는 동적파일 요청 비율을 증가시키면서 측정한 결과이다. 동적파일 요청 비율이 증가함에 따라 CALD 방식이 WRR 방식보다 성능이 좋아졌음을 알 수가 있다. 두 그래프 사이에 교차점이 발생하는 이유는 동적화일 요구량이 증가함으로서 CPU처리 역할이 중요하게 되고 지속적으로 부하를 측정하여 CPU에 부담을 주는 WRR 방식과는 달리 미리 측정된 부하 데이터를 가지고 부하를 분배하는 것이 더 효율적이라는 것을 보여주고 있다.

5. 결론 및 향후 연구

본 논문은 클러스터 웹 서버 성능 향상을 위해 다양한 컨텐츠로 인한 클러스터 서버 내부의 부하를 L4스위치 계층에서 적용하는 방법을 제시하였다. 제안된 방식은 back-end 노드들의 지속적인 부하 측정에 대한 과부하를 줄이기 때문에 기존의 WRR 방식보다 좋은 성능을 가진다.

앞으로의 연구 방향은 제안된 부하배분 정책을 클러스터 웹 서버의 가상 서버(Linux Virtual Server)의 모듈에 통합 구현하고자한다. 또한 대규모 부하가 발생되는 경우 CALD 방식의 성능이 다른 방식들과 비교하여 효율적인지 시험할 예정이다.

6. 참 고 문 헌

[1] D.Andersen, T. Yang, and O.Ibarra, Toward a scalable distributed WWW server on workstation clusters, Journal of Parallel and Distributed Computing 42(1) pp.91-100, 1997.

[2] A.Fox, S. Gribble, Y. Chawathe, E. Brewer, and P. Gauthier, Cluster-based scalable network services, Proceedings thier, Cluster-based scalable network services, Proceedings of the 16th ACM Symposium on Operationg System Principles(SOSP), pp.78-91, 1997.

[3] E.D. Katz, M.Butler, and R.McGrath, A scalable HTTP server: The NCSA prototype, Computer Networks and ISDN Systems, 27(2), pp.155-164, 1994.
pp.1117-1126, 2000.

[4] Balachander Krishnamurthy and Jennifer Rexford, Web Protocols and Practice : HTTP/1.1, networking protocols, caching, and traffic measurement, Addision Wesley, 1st edition, 2001.

[5] A. Iyengar, M. Squillante, and L. Zhang, Analysis and characterization of largescale web server access patterns and performance, World Wide Web Journal, 2(1) pp.85-100, 1999.

[6] H.Zhu and T.Yang, Cachuma: class-based cache management for dynamic web content, Proceedings of the IEEE INFOCOM 2001, pp.1215-1224, 2001.

[7] Cisco Systems Inc., http://www.cisco.com.

[8] Foundry Networks Inc., http://www.foundrynet.com

[9] G.Hunt, G. Goldszmidt, R. King, and R. Mukherjee, Network Dispatcher: A connection router for scalable internet services, Computer Networks and ISDN Systems, 30(1-7) pp.347-367, 1998

[10] V. Cardelini, M Colajanni, and P. Yu, Dynamic load balancing on web-server systems, IEEE Internet Computing, ancing on web-server systems, IEEE Internet Computing, 3(3) pp.28-39, 1999.

[11] J. Song, E.Levy, A.Iyengar, and D. Dias, Design alternatives for scalable web server accelerators, Proceedings of the 2000 IEEE International Symposium on Performance Analysis of Systems and Software(ISPASS),2000.

[12] M. Aron et al., scalable content-aware request distribution in cluster-based network servers, Proceedings of the USENIX Annual Technical Conference, June 2000.