사용자 패턴을 감안한 리눅스 레이어-7 웹 클러스터 시스템의 구현†

홍일구 조재욱° 도인환 노삼혁
홍익대학교 컴퓨터 공학과, 정보컴퓨터공학부
{ighong, mrtajo°, ihdoh}@cs.hongik.ac.kr, samhnoh@hongik.ac.kr

An Implementation Study of a Linux Layer-7 Web Clustering System that Incorporates Client Request Patterns

Ilgu Hong Jaewook Jo° Inhwan Doh Sam H. Noh
Dept. of Computer Engineering, Information and Computer Engineering, Hongik Univ.

요 약

폭발적인 웹 사용자의 증가를 수용하기 위해서 가격 대 성능비가 우수한 웹 클러스터링 시스템이 선호되고 있다. 많은 연구 결과들은 웹 요청의 패턴이 소수의 사용자 요청에서 매우 높은 확률과 빈도로 참조됨을 보여주고 있다. 이러한 참조 패턴을 반영하기 위해서 서버 시스템은 각각의 사용자 요청의 특성에 맞는 스케줄링 방법을 제공해야 한다. 본 논문에서는 TCP-handoff protocol을 이용한 Layer-7 기반의 클러스터링 시스템을 Linux에 구현하였고, 웹 사용자 요청 패턴에 기반을 둔 DS(Dual Scheduling) 부하 분산 알고리즘을 적용하여 기존의 부하 분산 알고리즘과 비교하였다. 실험적으로 DS 알고리즘은 기존의 알고리즘에 비해 35% 이상의 성능향상을 보여준다.

1. 서 론

웹이 보편화되면서 사용자 수는 매년 폭발적인 증가 추세를 보이고 있다. 이러한 사용자의 증가를 수용하기 위한 시스템으로 웹 클러스터링 기술이 선호되고 있다. 스위치 기술을 바탕으로 저가의 서버 시스템들로 구성된 클러스터링 시스템은 사용자의 요청을 클러스터에 포함된 각 노드에 분산시켜 처리한다. 결국 단일 시스템에 가해지던 부하는 클러스터링 시스템을 구성하는 각 서버 노드로 분산된다. 본 연구에서는 Aron[1]이 FreeBSD에 구현하였던 Layer-7 기반의 클러스터링 시스템을 Linux에 구현하였다. 이 시스템은 전위 서버가 병목지점이 되는 현상을 제거하기 위해 전위 서버의 패킷 전송 기능을 각 후위 서버가 담당하는 방법을 사용한다.

웹 사용자 요청의 패턴은 일반적으로 Zipf 분포를 따른다. 즉, 소수의 요청이 전체 부하의 대부분을 차지하고 있다. 이러한 사용자의 패턴을 반영하기 위해서 웹 서버의 경우 캐싱 기술을 이용하여 자주 참조되는 페이지를 메모리에 상주시켜 지역성을 활용할 수가 있다. Layer-7 기반의 웹 클러스터링 시스템의 경우 본질적으로 지역성을 효과적으로 지원한다. 하지만 사용자 패턴을 효과적으로 반영하기 위해서는 지역성과 더불어 시스템 자원을 효율적으로 할당할 수 있어야 한다. 본 연구에서는 클러스터 시스템의 자원을 사용자 요청의 패턴에 따라 할당할 수 있는 DS(Dual Scheculing) 알고리즘을 제안하였다. 실험을 통해 기존 알고리즘에 비해 약 35% 이상의 성능향상을 가져옴을 보인다.

본 논문의 구성은 다음과 같다. 2장에 지금까지 제안되었던 클러스터링 시스템에 관하여 간단하게 다룬다. 3장과 4장에 걸쳐 Linux에 TCP-handoff 기반의 클러스터링 시스템의 구현과 DS 부하 분산 알고리즘에 대해 알아본다. 5장에서는 실험 및 결과를 살펴보고, 마지막으로 6장에서 결론을 내린다.

†이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구 되었음 (KRF2002-041-D00426).

2. 관련연구

웹 클러스터링 시스템은 전위서버와 후위서버로 구성된다. 전위서버는 사용자의 요청을 분산시키는 역할을 담당하고 후위서버는 사용자에게 실제 서비스를 제공하게 된다. 전위서버에서 후위서버를 결정하는 방법에 따라 크게 Layer-4 방식과 Layer-7 방식으로 나눌 수 있다.

Layer-4 방식의 부하 분산 과정은 사용자의 요청과 무관하며 후위 시스템의 부하에 따라 커넥션을 할당하게 된다. 이에 비해 Layer-7 방식은 사용자의 요청에 따라 부하를 분산한다.

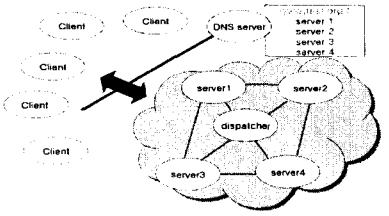
2.1 기존의 Layer-7 클러스터링 시스템

기존의 Layer-7 기반의 클러스터 시스템에서 가장 문제점으로 지목된 부분은 바로 전위 서버의 병목 현상일 것이다. 이를 해결하기 위해서 후위 서버의 응답을 곧바로 클라이언트로 보내도록 새로운 방식의 클러스터 시스템이 제안되기도 했지만, 요청의 내용으로 분산을 조절하는 부하 분산 알고리즘의 복잡한 연산에 의해서 전위 서버의 부하가 완전히 해결될 수는 없었다.

2.2 확장성이 향상된 Layer-7 클러스터링 시스템 구성

Aron[1]등은 전위 서버의 부하로 확장성이 떨어지는 문제를 해결하기 위하여 확장성 있는 시스템을 위한 새로운 클러스터링 구성 기법을 제시하였다. 이 방식에서는 기존에 전위 서버가 가지고 있던 부하를 크게 두 부분으로 나눈다. 첫째는 후위 서버를

결정하는 부하 분산 부분이고 둘째는 전위 서버와 후위 서버의 커넥션을 관리하는 부분이다. 이 두 부분에서 상대적으로 부하가 큰 후자를 후위 서버에서 처리하도록 하여 시스템의 확장성을 보장하였다.



<그림 1. 확장성이 향상된 Layer-7 클러스터링 시스템>

클러스터링 시스템 전위에 Round-Robin DNS 서버 혹은 Layer-4 스위치를 두고 사용자 요청을 서버들 사이에 분산 시킨다. 사용자 요청은 각각의 서버로 전달되고 각 서버는 사용자의 요청을 패킷에서 읽어 서버 ID와 함께 부하 분산을 담당하는 서버에 전달한다. 부하 분산 서버는 각 서버의 상태와 지역성을 고려하여 이를 처리할 서버를 결정하고 재 전송한다. 최초 사용자의 요청을 받은 서버는 핸드오프 프로토콜을 이용하여 해당 서버로 커넥션을 전달하게 된다. 그림 1에서는 이러한 확장성이 향상된 웹 클러스터링 시스템을 보여준다.

3. Layer-7 기반 Linux 웹 클러스터링 구현

본 논문에서는 Aron 외[1]의 연구에서 FreeBSD 시스템에서 구현하였던 클러스터링 시스템을 Linux kernel 2.4.16 버전에 구현하였다. Linux 시스템은 FreeBSD와는 네트워크 코드가 상이하여 새로운 설계와 구현을 하게 되었는데, 이 과정에서 새로운 부하 분산 알고리즘을 도입하여 성능 개선 및 기존의 연구와 비교할 만한 실험 결과를 도출해 내게 된 의미 있는 실험이었다. 다음은 실제로 구현한 클러스터링 시스템에서 주요 부분을 정리한 것이다.

3.1 새로운 sock 핸들러 설정 및 sock 구조 수정

TCP/IP 네트워크 스택에서 BSD 소켓 인터페이스로 전달되는 메시지를 가로채서 이를 처리할 새로운 sock 핸들러를 정의하였다. 이 핸들러를 통하여 핸드오프와 부하 분산 과정이 처리된다. 각 sock 구조는 커넥션 종류를 구별하기 위한 상태 필드와 핸드오프와 관계된 주소 정보를 저장하기 위한 필드를 추가하였다.

3.2 Handoff 프로토콜 설정

TCP-handoff 기법은 사용자와 최초 사용자의 요청을 받아들인 서버 사이에 생성된 커넥션을 부하 분산 서버에 의해 결정된 서버로 전달하는 메카니즘이다. 이를 가능하게 하기 위하여, handoff 프로토콜을 TCP 계층 위에 구현하였다.

4. 부하 분산 알고리즘

웹 사용자 요청 패턴에 대한 연구들은 웹 데이터가 가지는 성격들을 다음과 같이 정리하였다. 첫째, 웹 데이터의 참조량은 Zipf 분포와 유사한 성격을 갖게 된다[2]. 즉, 자주 참조되는 적은 수의 요청이 웹 요청 *workload*의 대부분을 차지한다는 것이다. 둘째, 웹 데이터의 크기 분포는 heavy-tailed한 성격을 갖는다[3]. 이러한 웹 데이터의 성격을 바탕으로 Breslau 외[2]는 참조량의 Zipf-like 성격을 이용하여 특정시간에 특정 사용자 요청 횟수는 참조량과 밀접한 관계가 있다는 것을 보여주었으며 이를 통해 웹 데이터가 지역성을 가지고 있다는 것을 보여주고 있다.

웹 데이터의 지역성을 시스템에 효율적으로 반영하기 위해서는 각 요청의 패턴에 따른 스케쥴링 알고리즘이 필요하다. Layer-7 기반의 웹 클러스터링 시스템에 적용된 대표적인 부하 분산 알고리즘인 LARD는 지역성에 바탕을 두고 부하 분산을 한다[1, 4]. 하지만 지역성만으로는 웹 요청의 패턴을 정확히 반영하지 못한다. 본 논문에서는 이러한 문제를 해결하기 위하여 DS(Dual Scheduling) 알고리즘을 제안하였고 기존의 LARD 알고리즘과 비교하였다.

4.1 LARD 알고리즘

LARD (Location Aware Request Distribution) 알고리즘은 지역성에 바탕을 둔 부하 분산 알고리즘이다. 이 알고리즘은 가장 최근에 특정 요청을 받은 서버에 동일 요청을 할당하여 불필요한 디스크 I/O를 줄임으로써 전체 시스템의 성능을 향상시키는 것을 목적으로 한다. 이때 서버의 부하가 집중되어 병목현상이 발생하는 것을 막기 위하여 서버의 부하가 특정 값 이상으로 증가하지 못하도록 관리한다. 이 때 활용되는 것이 부하 분산기에서 관리하는 각 서버의 부하 정보와 사용자 요청-실행 서버 매핑 테이블이다.

4.2 사용자 요청 빈도를 감안한 DS (Dual Scheduling) 알고리즘

DS 스케줄링 기법은 단위 시간의 특정 웹 요청의 빈도에 따라 시스템의 자원을 적극적으로 활용할 수 있는 기법을 제공하면서 더불어 지역성을 활용한 부하 분산 기법을 제공한다. 예를 들면 웹 요청은 부하가 크지 않은 경우 DS 스케줄링 기법은 부하 분산기를 통해 지역성을 고려한 스케줄링을 한다. 만약 특정 웹 요청의 빈도가 증가하게 되면 부하 분산기는 이를 처리할 서버들을 증가시킨다.

이 때 선택된 각각의 서버들은 특정 웹 요청의 빈도가 임계 값 이상일 경우 부하 분산기를 거치지 않고 서버에서 직접 처리하며 웹 요청의 빈도가 임계 값 이하로 줄어들 경우 다시 부하 분산기에 웹 요청의 스케줄링을 요청한다. 즉, 부하 분산기의 입장에서 특정 웹 요청의 빈도가 증가하게 되면 이를 처리할 서버의 수를 증가시켜 자원을 할당하며 요청의 빈도가 줄어들면 자동으로 자원을 회수하게 된다.

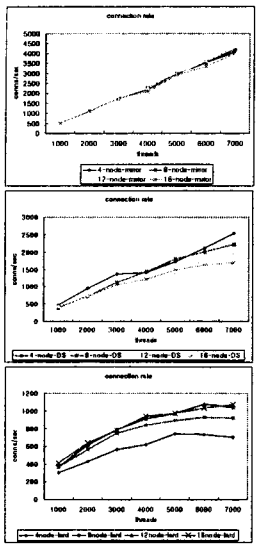
5. 실험 및 결과

실험 환경을 구성하기 위하여 클러스터 서버 노드로 Pentium III 800Mhz, 128MB RAM의 시스템 16대를 사용하였고, 사용자 요청을 생성하기 위하여 같은 사양의 시스템 4대를 사용하였다. 각각의 시스템은 100Mbps 이더넷 카드를 사용하고 있으며 100Mbps 스위치 허브에 연결되어 있다. 클러스터링 시스템은 Linux 커널 2.4.16을 사용하고 있다. 사용자 요청을 생성하기 위해서 Surge[5] 프로그램을 사용하였다.

실험은 클러스터 노드를 4의 배수로 증가시키고 각각의 노드 수에 대해 스레드의 수를 1000에서 7000까지 증가시키면서 수행하였다. Surge의 스레드는 사용자의 접근 패턴에 따라 서버의 요청을 생성하여 전달하게 된다. Content의 수는 2000개로 한정 지었으며 약 50MB의 부하로 구성되었다. 각각의 실험에서 프로세스의 수는 80개로 고정시키고 스레드의 수를 변화시켰다. 시스템의 성능을 비교하기 위해 클러스터의 노드 수와 동일한 노드로 구성된 미러링(mirroring) 시스템을 구성하였다.

미러링 시스템은 50MB의 부하를 고려해 볼 때 이상적인 시스템으로 설정하였다. 실험에서 시스템은 모든 content를 메모리에 수용할 수 있으므로 디스크 I/O에 대한 부하는 거의 없게 된다.

5.1 처리율

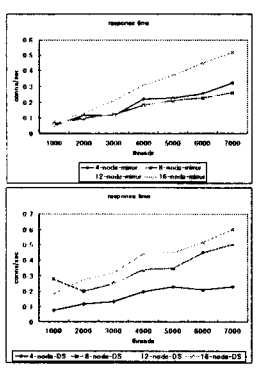


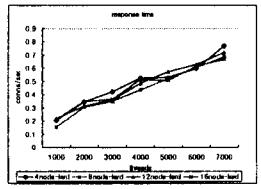
<그림 2. Mirror, DS, LARD 시스템 응답시간>

첫 번째 실험은 각 알고리즘을 이용한 시스템의 처리율을 보여주고 있다. 그림 2에서 알 수 있듯이 이상적인 미러링 시스템의 성능이 가장 좋은 것으로 나타나고 있다. 7000 스레드에서 미러링 시스템은 초당 약 4300개의 사용자 요청을 처리하며, 가장 성능이 나쁜 것은 LARD 시스템으로 초당 약 1000개의 사용자 요청을 처리한다.

DS 알고리즘을 적용한 시스템은 LARD 시스템과 미러링 시스템의 중간 정도의 특성을 보여주고 있다. 실험에서 노드 당 부하수에 따라 DS 시스템은 LARD와 미러링 시스템 사이에 놓이게 된다. 노드 당 부하가 적을 경우 LARD 시스템에 가까운 성격을 보여주며, 노드 당 부하가 증가함에 따라 미러링 시스템의 성격을 따르게 된다. 그림 2에서 관찰할 수 있는 또 한가지 특이 사항은 DS 시스템에서 노드 수가 증가할수록 성능이 떨어진다는 것이다. 이것은 단위 노드 당 사용자 요청 수가 클러스터의 노드 수가 적을수록 높기 때문에 나타나는 현상으로 적은 노드의 클러스터에서 사용자 요청이 증가함에 따라 Hot 한 요청이 나타나게 되고 DS 스캐줄링이 적용되면서 부하 분산기를 거치지 않고 로컬에서 처리되기 때문이다. 또한 이 시점에서 요청을 처리하는 노드의 수가 부하 분산기에 의해 증가하게 된다.

5.2 응답 시간





<그림 3. Mirror, DS, LARD 시스템 응답시간>

그림 3은 응답시간 결과를 보여주고 있다. LARD 시스템의 경우 응답시간이 거의 균등하게 나타나는 반면에 DS 알고리즘을 적용한 시스템은 노드 수에 따라 매우 큰 차이를 보여주고 있다. 4.2절에서 살펴 보았듯이 DS 알고리즘이 적용되어 부하 분산기를 거치지 않고 처리된 요청은 불필요한 패킷 교환과 핸드오프를 처리하는데 사용되는 시간을 줄이게 된다. 결국 DS 알고리즘은 각 요청의 빈도에 따라 시스템 자원의 할당을 조절하게 되고 이를 통해 부가적으로 패킷의 처리 시간을 줄이는 효과를 얻게 된다.

6. 결론

요청의 접근 빈도는 각 요청의 특성에 따라 달라지게 된다. 예를 들어 특정 홈 페이지의 첫 번째 파일-index.html-은 다른 페이지들에 비해 요청 빈도가 높을 것이다. 또 다른 상황을 살펴보면 9.11 사태와 같은 예기치 못한 상황에서 특정 홈페이지로의 요청의 빈도는 급증하게 될 것이다. 이러한 상황에 대비하기 위해 일반적으로 웹 서버는 캐싱 메커니즘을 사용하기도 한다.

본 논문에서는 Aron[4]이 제안한 웹 클러스터링 시스템 환경에 사용자 빈도에 따른 부하 분산 알고리즘을 적용하였다. 실험을 통해서 DS 알고리즘은 사용자의 요청 빈도에 따라 적응성을 갖는 시스템이라는 것을 알 수 있었다. 단순한 LARD 시스템에 비교해 16 노드 시스템의 경우 약 50%의 성능 향상을 보이고 있으며, 전체적으로는 약 35%의 성능 향상을 보여주고 있다. DS 시스템은 빈도가 낮을 경우 LARD 시스템의 성격을 갖게 되며 빈도가 증가함에 따라 미러링 시스템의 성격을 보여주고 있다. 즉, 사용자의 요청 빈도를 잘 수용하고 있다.

본 연구를 통해 TCP-handoff 기반 Linux 클러스터링 시스템을 구현하였고, 사용자 요청에 빈도를 반영하기 위한 DS 부하 분산 알고리즘을 제안하였다.

참고 문헌

[1] 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, 2000.

[2] L. Breslau, P. Cao, Li Fan, G. Phillips and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications", In Proceedings of Infocom'99, April 1999.

[3] M. Arlitt and C. Williamson, "Web server workload characteristics: The search for invariants", In Proceedings of ACM SIGMETRICS'96, May 1996.

[4] Vivek S. Pai, Mohit Aron, Gauray Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel and Erich Nahum, "Locality-aware Request Distribution in Cluster-based Network Servers", In Proceedings of ASPLOS-VIII, 1998.

[5] Paul Barford and Mark Crovella. "Generating Representative Web Workloads for Network and Server Performance Evaluation", In Proceedings of the ACM SIGMETRICS '98, pages 151-160, Madison, WI, June 1998.