콘텐츠 복제 웹 서버에서 캐쉬 복제를 통한 성능 향상

김호중° 맹승렬
한국과학기술원 전자전산학과 전산학전공
hjkim@camars.kaist.ac.kr° maeng@cs.kaist.ac.kr

Performance Improvement in the Content-Replicated Web Servers Using Cache Replication

Hojoong Kim° Seungryoul Maeng
Division. of Computer Science, Korea Advanced Institute of Science and Technology

요 약

최근 웹 서비스에서 동적 콘텐츠의 비율이 증가함에 따라 캐쉬 적중률을 높임으로써 서버 CPU의 부하를 줄이는 일의 중요성이 커지고 있다. 서버의 성능을 증가시키기 위한 방법으로 콘텐츠 복제 서버 클러스터가 널리 활용되고 있다. 그러나 콘텐츠 복제 서버의 각 노드는 유사한 작업을 수행함에도 불구하고 서로 독립적으로 동작하므로 캐쉬 적중률이 감소한다. 본 논문에서는 한 서버 노드에서 캐싱하는 웹 콘텐츠를 다른 서버 노드의 캐쉬에 복제함으로써 서버 팜의 각 노드의 캐쉬 내용을 유사하게 관리하여 캐쉬 적중률을 높인다.

1. 서론

인터넷의 사용량이 폭발적으로 증가함에 따라 대량의 웹 콘텐츠를 다수의 사용자에게 효율적으로 제공하기 위한 연구들이 활발하게 진행되고 있다. 이러한 연구는 크게 프록시 캐쉬(proxy cache)에서의 성능 향상과 서버에서의 성능 향상으로 나눌 수 있다.

프록시 캐쉬에서의 성능 향상은 콘텐츠를 인터넷 상에 효율적으로 복제하여 사용자와 웹 서버 사이의 전송 지연 시간(delay)을 줄이고 네트워크와 서버의 부하를 줄임으로써 얻을 수 있다[1]. 그러나 프록시 캐쉬는 다수의 사용자와 웹 서버들을 중개하므로 비교적 캐쉬 적중률(hit rate)이 떨어지며 콘텐츠 요청의 상당 부분은 여전히 서버에서 처리하게 된다. 또한 점차 비중이 증가하고 있는 동적 콘텐츠(dynamic contents)의 경우 이미지 파일과 같은 정적 콘텐츠(static contents)에 비해 서버에 많은 부하를 줄 뿐 아니라 일반적으로 프록시 캐쉬에서 캐싱하기 어려우므로[2] 대부분의 요청을 서버에서 처리하게 된다. 따라서 웹 서버의 성능을 향상시키는 일이 중요하다.

웹 서버의 성능을 향상시키는 일반적인 방법은 여러 대의 서버 노드를 클러스터로 묶어 사용자의 요청을 분산시키는 것이다. 이는 크게 두 가지 방법으로 나눌 수 있다. 첫째, 웹 콘텐츠를 여러 노드에 복제하여 각각의 노드에서 동일한 서비스를 제공함으로써 작업을 분산하는 방법이다[1]. 둘째, 각각의 서버 노드에 서로 다른 작업을 할당하고 사용자의 요청에 따라 적합한 서버를 선택하는 방법이다. 콘텐츠 복제 방법은 서버 구성이 간단하고 확장성이 높은 반면 복제된 콘텐츠의 일관성을 유지하는 오버헤드(overhead)가 생기며 서비스 분산 방법은 콘텐츠를 관리하기 용이하지만 부하 분산기(load balancer)에서 사용자의 모든 요청을 일일이 확인하고 적합한 서버에 전달해야 하므로 확장성이 떨어진다.

본 논문에서는 콘텐츠 복제 웹 서버에서의 성능 향상 방법에 대하여 연구한다. 콘텐츠 복제 방법에서는 여러 서버 노드가 동일한 서비스를 수행한다. 그러나 각각의 노드는 독립적으로 동작하므로 콘텐츠를 중복하여 캐싱하거나 동일한 콘텐츠에 대하여 여러 노드에서 디스크 접근이 발생하는 등 시스템 자원을 효율적으로 활용하지 못한다. 따라서 본 논문에서는 독립적으로 동작하는 여러 노드들의 캐쉬 내용을 동일하게 유지함으로써 디스크 접근을 줄이고 유사한 서비스를 효율적으로 제공할 수 있도록 한다.

2. 관련연구

클러스터 시스템 상에서 여러 노드의 캐쉬를 공유함으로써 전체 시스템의 캐쉬 성능을 높이고자 하는 협력 캐쉬(cooperative cache) 연구가 많이 진행되고 있다[3]. 주로 파일 시스템 분야에서 수행된 이러한 연구들은 전체 시스템의 효율적인 메모리 자원 이용에 중점을 두고 있으며 웹 서비스의 특성에는 적합하지 않다. 따라서 웹 콘텐츠를 위한 클러스터 파일 시스템[4]이 개발되었으나 이 또한 파일 시스템의 성능 향상에 중점을 둔 연구이다. 웹 서버에서 협력 캐쉬를 구현한 연구로는 Holmedahl[5] 등의 연구를 들 수 있다. 이러한 협력 캐쉬 연구들은 일반적으로 캐쉬 미스 발생시에 다른 노드로부터 캐쉬된 콘텐츠를 찾으므로 전송 지연 시간이 증가하는 단점이 있으며 다른 노드의 캐쉬 정보를 유지하는 오버헤드가 생긴다.

서버 캐쉬의 성능을 높이는 다른 방법으로 리버스 프록

시(reverse proxy)를 사용하는 방법이 있다. 리버스 프록시는 서버 전단(front end)에 설치되어 서버의 콘텐츠를 캐싱함으로써 서버의 부하를 줄인다. 리버스 프록시는 캐쉬 적중률이 높아 대규모 웹 서버에서 널리 채택되고 있다. 그러나 프록시 캐쉬 상에 콘텐츠가 존재하는지 검색하기 위해 모든 사용자의 HTTP 요청을 확인해야 하므로 동시 사용자 수가 많아지거나 캐쉬 적중률이 낮은 경우 프록시 캐쉬가 병목현상을 일으킬 수 있다.

3. 캐쉬 복제 시스템의 설계

본 논문에서는 캐쉬 복제 시스템을 제안한다. 캐쉬 복제 시스템이란 한 노드에서 캐싱하는 콘텐츠를 다른 노드의 캐쉬에 능동적으로 적용하는 것이다. 기존의 협력 캐쉬 방법에서는 자신의 노드에서 캐쉬 미스가 발생하는 경우에 한하여 미리 갖고 있는 다른 노드의 캐쉬 정보를 살펴보고 콘텐츠를 요청하는 반면 캐쉬 복제 시스템에서는 자신이 캐싱하는 콘텐츠를 다른 노드의 캐쉬에 미리 가져다 두므로 그 노드에서 콘텐츠 요청이 발생하였을 때 캐쉬로부터 바로 읽어올 수 있다.

캐쉬 관리자는 두 개의 콘텐츠 리스트를 관리한다. 하나는 자기 노드의 캐쉬 관리자가 읽어들인 콘텐츠를 관리하는 지역 캐쉬(local cache) 리스트이고 다른 하나는 다른 노드의 캐쉬 관리자로부터 복제된 원격 캐쉬(remote cache) 리스트이다. 그림 1에서와 같이 캐쉬 미스가 발생하는 경우 캐쉬 관리자는 우선 디스크로부터 콘텐츠를 읽어들여 자신의 지역 캐쉬에 저장한 다음 다른 노드들의 원격 캐쉬에 콘텐츠를 전달한다. 해당 콘텐츠가 이미 캐싱되어 있는 경우 리스트의 처음으로 연결하며, 캐싱되어 있지 않은 경우 리스트로부터 퇴거(evict)될 콘텐츠를 선정하여 캐쉬 교체(replacement)를 수행한다. 원격 캐쉬 리스트에서 캐쉬 적중이 발생하는 경우 해당 콘텐츠를 지역 캐쉬 리스트로 옮긴다.

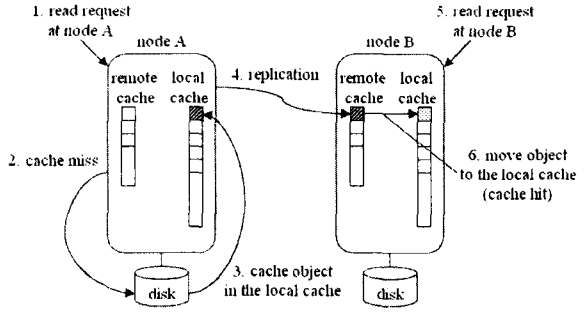

그림 1. 캐쉬 관리자의 구조

캐싱된 콘텐츠를 적극적으로 복제하는 경우 다음과 같은 문제점이 생길 수 있다. 첫째, 서버간의 트래픽이 증가하고 불필요한 데이터 전송이 발생한다. 상대방 노드의 캐쉬 내용을 고려하지 않고 복제한 콘텐츠를 전송하는 경우 콘텐츠가 중복될 수 있다. 둘째, 복제된 콘텐츠로 인하여 사용 가능한 캐쉬 메모리 사용량이 줄어들고 캐쉬 적중률이 낮아진다. 반면 소극적으로 복제하는 경우 캐쉬 복제를 사용하지 않는 기존 시스템과 유사하게 동작한다. 따라서 적절한 캐쉬 복제 정책을 선택하는 것이 중요하다. 본 연구에서는 적극적 복제 방식을 채택하였으며

4. 성능 평가

본 논문에서는 1998년 월드컵 로그[6, 7]를 사용하여 제안한 캐쉬 복제 시스템의 성능을 평가한다. 전체 로그 중 서버에서 가장 많은 웹 트래픽을 처리한 하루를 선택하여 시뮬레이션하였다. 월드컵 로그의 서버 구성은 파리를 포함하여 크게 네 군데의 지역 서버로 분산되어 있으며 각각의 지역 서버는 다시 여러 대의 서버 노드로 구성된다. 본 논문에서는 파리 지역 서버에서 수행된 로그만을 선택하였다. 파리 지역 서버는 영어와 프랑스어 콘텐츠를 모두 지원하므로 다른 지역 서버에 비해 캐쉬 적중률이 낮다.

본 연구에서 가정하는 서버 환경은 round robin으로 동작하는 부하 분산기에 4개의 서버 노드가 연결되어 있다. 구체적인 서버 구성은 표 1과 같이 Hyun[8]의 연구에서 가정한 환경을 따른다. 서버 노드는 Pentium III 700MHz CPU를 사용하고 256MB SDRAM, 5400RPM 디스크를 장착하였다.

표 1. 시뮬레이션에 사용한 측정값[8]

connection establishment or teardown	132μ s
Process fork	1400μ s
Context switch	35μ s
basic processing steps	213μ s
IMS response time	30μ s
Transmission of 512bytes to network	37μ s
disk seek and rotational latency	15600μ s
disk transfer time for 4Kbytes block	300μ s

한편 본 논문에서는 각각의 서버 노드가 독립적으로 디스크를 가지며 모든 콘텐츠가 각 서버 노드의 디스크에 복제되어 있는 것으로 가정한다. 캐쉬 대체 알고리즘은 LRU를 사용한다.

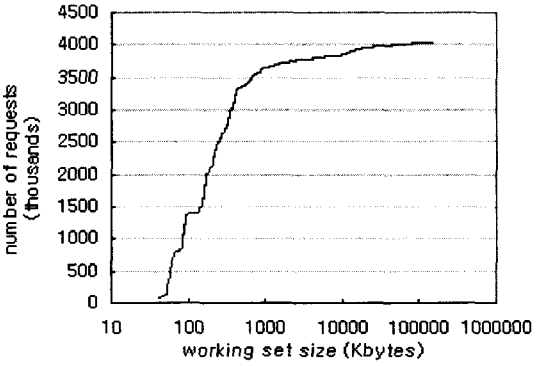

그림 2. 월드컵 로그의 콘텐츠 요청 분포

실험에 사용한 로그는 그림 2에서 보는 바와 같이 전체 약 404만개의 콘텐츠 요청 중 90% 이상이 1Mbytes 이하의 콘텐츠에 집중되므로 캐쉬 적중률이 매우 높다. 그림 3에서와 같이 노드 당 4Mbytes의 캐쉬를 사용하는 경우 캐쉬 적중률이 96%를 넘으며 byte당 캐쉬 적중률도 90%를 넘는다. 캐쉬 적중률에 비하여 byte당 캐쉬 적중률이 낮은 이유는 커다란 파일을 요청하는 빈도가 낮아 이들 파일의 요청시 캐쉬 미스가 발생하기 때문이다.

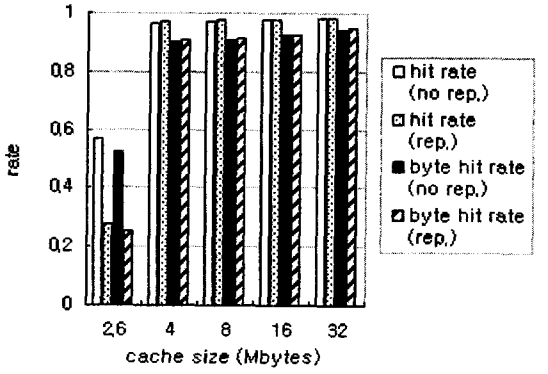

그림 3. 캐쉬 적중률과 byte당 캐쉬 적중률

그림 3에서 캐쉬 메모리가 어느 정도 이상으로 커지는 경우 제안한 캐쉬 복제 방법이 약 1% 정도의 캐쉬 적중률 향상을 보인다. 이는 캐쉬 미스가 일어날 콘텐츠를 미리 가져다놓기 때문이기도 하지만, 자신의 노드에서 캐쉬 중인 콘텐츠가 캐쉬에서 대체되기 전에 다른 노드에서 캐쉬 적중이 일어나 다시 LRU 리스트의 처음으로 등록되기 때문이다. 전자에 비해 후자에 인한 캐쉬 미스 감소가 약 12~36배에 달한다.

즉, 본 논문에서 제안하는 캐쉬 복제 방법은 캐쉬 적중률이 높은 콘텐츠의 캐쉬 대체 확률을 크게 감소시킨다.

이처럼 캐쉬 적중률이 증가함에 따라 노드 당 평균 CPU 수행 시간도 기존 콘텐츠 복제 방법과 비교하여 그림 4와 같이 줄어든다.

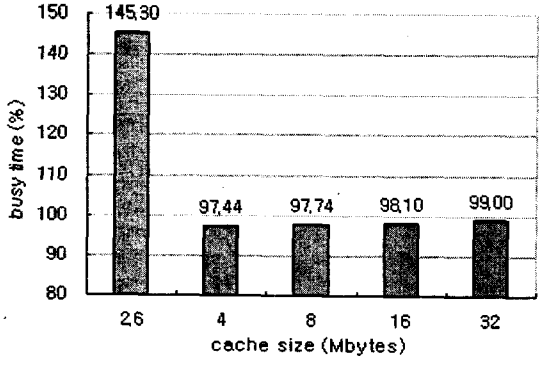

그림 4. 캐쉬 복제 방법에 의한 성능 향상

한편 캐쉬 메모리 사이즈가 작아서 빈번하게 캐쉬 미스가 발생하는 경우 캐쉬 복제 방법은 오히려 성능을 크게 낮추는 요인이 된다. 이를 해결하는 방법은 실제로 콘텐츠를 복제하는 것이 아니라 각 노드에 캐쉬 사실만을 통보하는 것이다. 앞서 살펴본 것처럼 대부분의 성능 향상은 캐쉬 적중률이 높은 콘텐츠를 캐쉬 내에 유지함으로써 얻어지므로 이와 같은 방법으로 콘텐츠 복제로 인한 캐쉬 메모리의 낭비 없이 높은 캐쉬 적중률을 유지할 수 있다. 또한 여러 개의 통보 메시지를 모아서 한번에 전송함으로써 네트워크의 부하를 줄이고 메시지 중복을 줄일 수 있다.

5. 결론

본 논문에서 제안하는 캐쉬 복제 방법은 한 노드에서 캐쉬 적중이 일어난 콘텐츠를 다른 노드의 캐쉬에 복제함으로써 캐쉬 적중률을 높이는 방법이다.

본 연구에서는 정적 콘텐츠만을 대상으로 하였으나, 메모리 자원에 비해 CPU 자원을 많이 필요로 하는 동적 콘텐츠 서비스에 본 기법을 적용하면 캐쉬 적중률 향상으로 인하여 서버 CPU의 부하를 줄이고 더욱 높은 성능을 얻을 수 있을 것이다.

6. 참고문헌

[1] R. Burns, R. Rees, and D.Long, " Efficient Data Distribution in a Web Server Farm" , IEEE Internet Computing, July/August 2001.

[2] J.Yin, L.Alvisi, M.Dahlin, and A.Iyengar, " Engineering Server-Driven Consistency for Large Scale Dynamic Web Services" , ACM International World Wide Web Conference, May 2001.

[3] P.Sarkar and J.Hartman, USENIX Symposium on Operating Systems Design and Implementation, October 1996.

[4] W.Liu, M.Wu, X.Ou, W.Zheng, and M.Shen, " Design of an I/O Balancing File System on Web Server Cluster" , International Conference on Parallel Processing, August 2000.

[5] V.Holmedahl, B.Smith, and T.Yang, " Cooperative Caching of Dynamic Content on a Distributed Web Server" , IEEE International Symposium on High Performance Distributed Computing, July 1998.

[6] M.Arlitt and T.Jin, " A Workload Characterization Study of the 1998 World Cup Web Site" , IEEE Network, Vol.14, No.3, May/June 2000.

[7] M.Arlitt and T.Jin, " Workload Characterization of the 1998 World Cup Web Site" , Technical Report, HPL-1999-35, September 1999.

[8] J.Hyun, I.Jung, J.Lee and S.Maeng, " Content Sniffer based Load Distribution in a Web Server Cluster" , IEICE Transactions on Information and Systems, Vol.E85-D, No.1, January 2003.