웹서버에서의 웹가속기 활용 기술

송 준 화*, 박 지 훈**
(*KAIST 전자전산학과, **연세대 전기전자공학과)

1. 서 론

지난 몇 년 동안 World Wide Web은 사용자, 사이트 그리고 데이터의 수 등 모든 측면에서 폭발적으로 성장하였다. 이러한 상황은 과도한 접속량을 발생시키고, 또한 과도하게 편중된(skewed) 엑세스를 일으킨다.

이러한 문제에 대응하기 위해 고성능 웹서버를 개발하고 그것을 관리하는 기술 또한 발전을 거듭하고 있다. 하지만, 다음과 같은 몇 가지 요소들에 의해 웹서버의 전체적인 성능이 저해된다. 우선 웹서버에 내장되어 있는 운영체제가 웹서버의 처리량을 감소시킬 수도 있다. 또한 요청(request)에 응답하기 위해 데이터는 파일 시스템과 응용프로그램 사이의 소프트웨어 계층을 넘나들면서 복사되는데 이런 현상은 운영체제 커널을 통과하면서, 또는 디바이스 드라이버(device driver) 레벨에서도 발생하여 시스템의 부하로 작용한다. 또한 운영체제의 스케쥴러나 인터럽트 프로세싱 등도 성능 저하의 요인이 된다.

높은 요구율을 감당하기 위해서 하드웨어를 추가하는 것이 가장 간단한 방법이지만 총 비용을 줄이기 위해 단일 또는 복수의 고성능 캐시인 웹 가속기를 웹서버 앞(front-end)에 장착할 수 있다.

가속기의 월등한 성능은 대부분 내장형 운영체제와 최적화된 캐시에 의해 이루어 졌다. 버퍼 복사(buffer copying)는 최소한으로 제한되었으며 운영체제가 멀티쓰레딩을 지원하지는 않도록 기능이 제한되는 등 일반적인 용도의 소프트웨어를 구현할 수는 없다. 하지만, 통신을 지원하기 위해 최적화 되었으므로 웹서버 가속 등의 특화된 네트워크 어플리케이션에는 더 잘 적용될 수 있다. 또한 웹가속기의 적중률(hit rate)을 최대화하고 갱신된 캐시를 유지 관리하기 위해서 응용 프로그램이 캐시에 저장된 데이터를 추가(add),삭제(delete), 갱신(update) 등의 명시적인 방법으로 해석하는 API를 제공한다. 따라서 필요 없어진 페이지는 어플리케이션이 언제든지 무효화하기 때문에 동적인 웹 페이지들을 정적인 웹 페이지처럼 캐시하는 것이 가능해진다.

이러한 웹가속기의 성능을 더욱 향상시키기 위해 다중처리장치(multiprocessor) 가속기를 사용할 수 있다. 다중처리장치 시스템 구조는 웹 가속기 노드의 클러스터와 전단(front-end) 부하분산기로 이루어져 있다. 각각의 노드는 캐시 멤버(cache member)로, 그것들의 집합은 캐시 배열(cache array)로 불린다. 확장성의 측면에서 배열은 단순히 멤버의 처리량을 합할 뿐만 아니라, 배열의 멤버들에 할당된 캐시 공간을 결합한다. 가속기는 높은 부하를 견뎌야하므로 모든 객체들은 속도 향상을 위해 메모리에 캐시된다. 따라서 다중 캐시 멤버의 사용은 전체적인 캐시메모리(메인 메모리)를 늘려서 시스템의 처리량을 증가시키는 효과를 나타낸다.

2. 웹서버 가속기의 설계와 특성

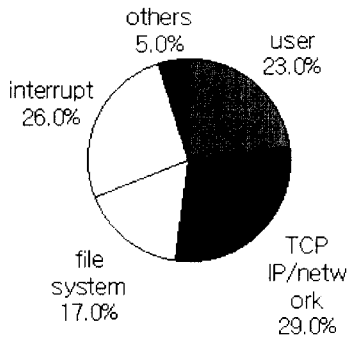

그림 1. CPU time break

그림 1에서는 일반적인 웹 접근 요청(access request)의 결과를 보이고 있다. 여기에는 TCP/IP와 네트워크, 인터럽트 핸들링, 사용자 공간 그리고 파일 시스템의 4가지 사항에 대해 CPU 사용시간이 어떻게 배분되는 지 나타난다.

가속기는 다음과 같은 원리로 시스템의 부하를 줄인다. 첫째로 가속기가 제공하는 캐시는 주 메모리(main memory) 캐시이므로 파일시스템에 연관된 부하가 없다. 두 번째로

가속기는 상호소통이 가능한 네트워크 머신에 기반하고 있으므로 데이터 전송에 최적화 된 소프트웨어 구조를 활용해 스케쥴링을 피하고 인터럽트가 최소한으로 유지될 수 있으므로 프로세스를 최적화 시킨다. 또한, 시스템은 내장형 운영체제 상에서 구현되었으므로 범용 운영체제 상에서 작동할 때 발생하는 불필요한 요소들이 제거된다. 마지막으로 시스템이 제공하는 API들은 요구 빈도가 높은 페이지를 미리 캐시함으로써 기능을 더욱 더 최적화 시킬 수 있다.

매우 큰 용량의 데이터가 요구되어 다중 웹서버가 사용되는 사이트에서 가속기가 웹서버 집합의 전면에 위치한다. 하나의 가속기는 부하 분산기와 캐시로 구성된다. 부하 분산기는 그 배후에 있는 많은 수의 서버와 캐시를 대표하여 단 하나의 IP로 외부에 나타낸다. 부하 분산기는 콘텐츠 라우터와 TCP 라우터의 두 가지 형태를 갖는다. URL 요청에 따라 라우팅하는 콘텐츠 라우터는 만약 요청한 페이지가 캐시에 있으면 그 페이지를 사용자에게 전달하고, 그렇지 않으면 라운드 로빈 혹은 서버의 상태를 고려하는 다른 방법을 사용해서 실제 서버로의 라우팅을 수행한다.

TCP 라우팅은 DNS(Domain Name Server)를 사용하는 라운드 로빈 방법에 비해 좋은 부하분산 효과를 나타낸다. TCP 라우팅의 상세한 과정은 다음과 같다. 데이터 요청이 있으면 가속기는 각각의 캐시에 요구한 데이터가 존재하는지 확인한다. 그러나 그 시간 동안 서버와 사용자간의 접속이 순간적으로 끊어진 상태가 된다. 따라서 캐시 미스(miss)가 발생한 후에는 사용자가 서버에 요청을 직접 전달할 수 없으므로 가속기가 서버에게 정보를 요구하고 그것을 사용자에게 전달한다. 캐시 미스가 발생할 경우 부하 분산기는 클라이언트에게 proxy 서버로서 작동해야 하므로 캐싱은 어느 정도의 부담을 지닌다. 반대로, 캐시기능이 중지되었을 때 부하 분산기는 TCP 라우터로서 작용한다. TCP 라우터는 요청에 해당하는 TCP 연결을 완성시키지는 않고 그 요청에 대응할 수 있는 노드를 선택하여 요청한다. 선택된 노드는 연결을 완성하여 요청이 들어올 때와는 달리 가속기를 통하지 않은 채 요청한 웹 페이지를 보내준다.

콘텐츠 기반 라우팅을 수행할 때의 장점은 가속기가 URL에 기초한 요청을 어디로 라우팅할 지에 대해 지능적으로 결정할 수 있다는 것이다. 예를 들면, 가속기는 한 그룹의 서버들에 정적인 데이터에 대한 요청을 할당하고 다른 한 그룹에 동적 데이터를 할당할 수 있다. 또 다른 예로 서버들의 콘텐츠가 동일하지 않은 경우에도 가속기는 URL에 기반한 보다 복잡한 알고리즘을 사용하여 각각의 서버에 요청들을 부과할 수 있다.

캐시는 자동모드(automatic mode)나 동적모드(dynamic mode) 또는 둘이 결합된 상태에서 동작한다. 자동 모드에서는 캐시 미스가 있는 경우 자동으로 데이터가 캐시되는데 이를 위해 웹 관리자는 자동으로 캐시되는 URL에 대한 파라메터를 작성한다. 예를 들면 정적 이미지 파일, 정적 비 이미지(nonimage) 파일, 동적 페이지 등에 대해 각기 다른 캐시 파라메터를 갖고 어느 정도 시간동안 캐시에 존재할 것인지를 결정한다. 하지만 서버의 응답에 포함된 HTTP 헤더들은 기본적인 캐시 파라메터 보다 우선권을 가지므로 특정한 URL의 콘텐츠가 캐시되어야 하고 그것이 얼마나 지속되어야 하는지를 지정하는 데 사용될 수 있다.

동적모드에서, 캐시 콘텐츠는 각각의 노드와 가속기 모두에서 동작할 수 있는 어플리케이션에 의해 제어된다. API는 응용프로그램이 URL의 콘텐츠에 대해 캐시, 무효화(invalidate), 조회(query)하고 지속시간을 지정할 수 있도록 한다. 하지만 동적 모드를 사용하는 응용 프로그램은 최적화된 동작을 요구하는 경우가 많으므로 프로그래머의 부담이 늘어난다. 따라서 동적 모드는 자주 사용되는 항목을 그것이 요청되기 이전에 캐시(prefetching)하거나 정해진 지속시간이 얼마인지 알 수 없는 데이터를 캐시에서 지우는 경우 등에 사용된다.

웹서버가 동적 웹 페이지를 생성하기 위해 같은 크기의 정적 페이지가 사용하는 것보다 수십 혹은 수백 배의 시간 동안 CPU를 점유하므로 많은 양의 동적 콘텐츠를 가진 웹 사이트는 성능 향상을 위해 동적 페이지를 캐시하는 것이 필수적이다. 만약 가속기가 동적 페이지를 캐시하도록 허가되지 않았다면 그 가속기의 존재를 인식하지 못할 정도로 성능이 감소한다. 캐시된 데이터를 디스크에 저장하면 가속기의 동작 속도가 극도로 저해되므로 모두 메모리에 저장해야한다. 따라서 캐시 데이터의 크기는 메모리의 크기를 초과할 수 없고 제한된 조건에서 최적의 캐시 갱신을 위해 최소 최근 사용(the Least Recently Used ,LRU) 알고리즘을 사용한다.

3. 확장형 웹 서버 가속기의 설계

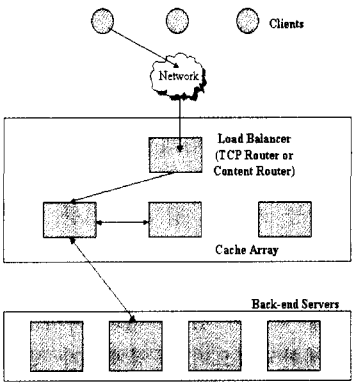

그림 2. 확장성 웹서버 가속기가 적용된 시스템 구조

확장형 가속기에서 부하 분산기는 웹 페이지 요구를 가속기 노드들 중 하나에 연결한다. 웹 URL 공간은 캐시 멤버의 숫자만큼으로 나누어지고(partition) 각각의 캐시멤버에 할당된다.

TCP 라우터는 콘텐츠 자체를 검사하지 않은 채 노드들의 부하만을 고려하여 라우팅 하므로, 이런 접근 방법 하에

서는 객체를 캐시하지 않은 첫 번째 캐시 노드에 일단 라우팅 될 가능성이 매우 커진다. 이런 상황이 발생하면, 첫 번째 노드는 그 요청을 다음의 둘 중 하나의 방법을 사용하여 객체의 소유자인 두 번째 캐시 노드에 전달한다. 만약 요구된 객체의 크기가 작다면 첫 번째 노드는 UDP 인터페이스를 사용하여 두 번째 캐시로부터 객체를 넘겨받고 그것을 사용자에게 보낸다. 만약 객체가 크면 첫 번째 캐시가 TCP 연결 자체를 두 번째 캐시에게 넘겨주는 것이 더 좋은 성능을 나타내는데, 이것을 TCP handoff 라고 한다. 그러면 두 번 째 캐시는 첫 번째 캐시를 거치지 않고 사용자에게 직접 데이터를 보낸다.

TCP 라우터가 캐시된 객체를 잘못된 노드에 라우팅할 확률을 줄이기 위해, 요청 빈도가 높은 객체(hot object)를 여러 개의 캐시 노드에 복사(replicated)한다. 이것은 캐시 교체(replacement)와 라우팅 전략(routing strategy)에 따라 수행된다. 캐시 복사와 캐시 교체 알고리즘은 다른 시간에 다른 지점에서 병목 현상이 발생할 수 있는 다양한 경우를 고려한다. 예를 들면 TCP 라우터, 캐시 노드, 웹서버 노드 그리고 네트워크 등이 병목이 될 수 있다. 따라서 어떤 종류의 자원(resource)이 병목 현상을 일으켰는지에 따라 다른 캐시 교체와 라우팅 전략의 결합이 적용된다.

콘텐츠 라우터는 요청을 다른 캐시 노드에 할당하기 위하여 그것의 내용을 검사하고, 부하 분산기는 URL에 따라 라우팅하여 TCP 연결을 수행하므로 TCP 라우터와 비교하여 노드의 CPU 사용 시간을 줄인다. 그러나 이것은 TCP 연결을 수행할 때 부하 분산기에 심각한 과부하를 발생시키고 병목현상을 일으킬 수 있다. 콘텐츠 기반 라우팅이 항상 좋지만은 않은 다른 상황도 있다. 몇몇 구조(architecture)에서는 객체가 라우터도 인식치 못한 사이에 캐시 사이를 이동(migration)하는 경우가 있다. 그러면 콘텐츠 기반 라우터는 다른 캐시 노드에게 객체를 요청하는 불필요한 동작을 한다. 따라서 라우터가 콘텐츠 기반의 요청에 대한 대응 여부에 관계없이 라우터와 캐시노드가 상호작용(interoperation)할 수 있도록 하여 캐시 노드 집합이 콘텐츠 기반 라우팅을 수행할 수 없는 경우에도 좋은 성능을 유지하도록 해야 한다.

부하 분산기는 배후에 있는 캐시 멤버와 서버의 수에 관계없이 단 하나의 IP주소만을 사용자에게 나타낸다. 그러므로 사용자가 알지 못하는 사이에 캐시 멤버와 서버를 추가하거나 제거하는 것이 가능하다. 그림2 에서처럼 부하 분산기가 분리된 노드에서 동작하게 하면 부하 분산기가 더 많은 요청을 관리할 수 있으므로 출력량을 최대로 할 수 있다. 캐시 배열이 작은 수의 노드로 구성되어서 부하 분산기가 병목현상을 일으키지 않을 때에는 부하 분산기가 캐시 멤버 노드에서 작동하도록 설정될 수도 있다.

4. 적용사례

웹 서버 가속기는 US 오픈 테니스 경기, 마스터스 골프 대회, 윔블던 오픈 테니스 경기 등의 다양한 스포츠 경기의

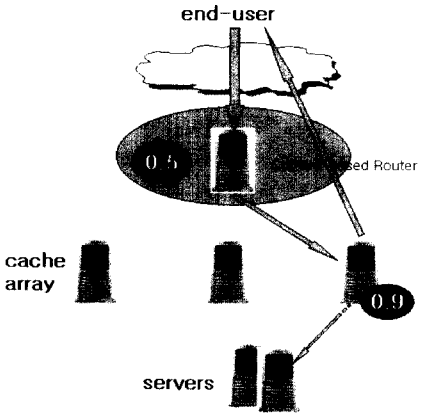

그림 3. 부하 분산기로 사용된 콘텐츠 기반 라우터
(숫자는 CPU 사용률)

사이트에서 사용되었다. 하나의 예로 1999년 6월 23일에서 7월 2일 사이에 개최되었던 윔블던 테니스 대회 사이트의 서버는 세 군데에 분산되어 설치되었는데(Schaumburg, Illinois; Columbus, Ohio; and Bethesda, Maryland;) 총 50개의 웹 서버와 4개의 웹서버 가속기가 사용되었다. (처음에는 두 개의 가속기가 사용되었고 나중에 두 개가 추가되었다.) 가속기의 소통량(traffic)은 부하 분산기의 역할로 인해 일반적인 웹서버보다 4배가 많았다. 약 2 주간의 대회 기간 동안 총 10억 회 정도의 데이터 요청이 있었는데, 7월 2일의 10 시간 동안에만 3천 8백만 번의 요청이 있었고, 오전 10:30 경에는 Bethesda 사이트에서 하나의 가속기에 의해 평균 매분 66,095번의 요청이 처리되었다.

네 개의 가속기에 의해 관리된 총 요청량은 8,932,303이다. 그 중에 8,190,429 번의 캐시가 적중(hit)하였고 나머지가 미스(miss)되어서 91.7%의 적중률을 보였다.

웹서버 가속기는 아마도 현재로서는 가장 높은 접속량을 기록할 것으로 예상되는 2000년 시드니 올림픽 웹사이트에도 적용되어 성능을 향상시키게 될 것이다.

5. 결 론

인터넷 환경의 확대에 따른 막대한 데이터 요구 증가를 해결하기 위해 하드웨어의 추가 즉, 서버의 수를 확대하는 것만으로는 데이터의 폭발적인 증가에 대응하기 어렵다. 이를 해결하기 위해 네트워크 머신에 기반 한 웹 가속기를 사용하면 추가 하드웨어 소요량을 최소화할 수 있고 향상된 데이터 서비스가 가능하다.

웹가속기는 효율적인 데이터 캐싱과 시스템의 역할에 특화된 내장형 운영체제를 통해 성능을 향상시킨다. 만약 데이터 요구량이 매우 클 경우 다중처리장치 가속기를 사용하면 전체적인 캐시 메모리를 확장하는 역할을 하여 시스템의 처리량과 속도를 대폭적으로 증가시키고 가용성을 향상시킨다.

참고문헌

[1] E. Levy, A. Iyengar, J. Song, and D. Dias, ''Design and Performance of a Web Server Accelerator'', Proc.of IEEE INFOCOM, pp. 135-143, Jan. 1999

[2] J. Song, E. Levy, A. Iyengar, and D. Dias, ''A Scalable and Highly Available Web Server Accelerator'', World Wide Web Conference, April, 1999.

[3] J. Song, E. Levy, A. Iyengar, and D. Dias, ''Design Alternatives for a Scalable Web Server Accelerator'', Proc. of IEEE International Symposium on Performance Analysis of Systems and Software, April 24-25, 2000, Austin, TX.

저 자 소 개

송준화(宋俊和)

1997년 미국 매릴랜드대 전산학 박사. 1997년 9월~2000년 8월 Research Staff Member at IBM T. J. Watson Research Center. 2000년 8월~현재 KAIST 전자전산학과 조교수. 연구분야: Internet Technologies, High Performance Web Serving, Internet Intermediaries, E-Commerce, Distributed Multimedia Systems

박지훈(朴智勳)

2000년도 연세대학교 졸업. 현재 연세대학교 대학원 석사과정. 연구분야: 실시간 처리 및 고장포용 기술