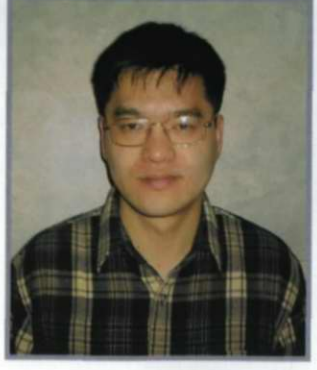
캐시 · 웹 · 미디어 서버를 이용한 콘텐츠의 동기화

지난 호에 글로벌 로드밸런싱에 대해 알아보았다. 글로벌 로드밸런싱의 기본 전제는 여러 네트워크에 분포돼 있는 서버의 콘텐츠가 모두 동일하게 동기화 돼 있다는 것이다. 이번 호에서는 서버의 콘텐츠 동기화에 대해서 알아보겠다. 네트워크에 분포돼 있는 서버의 종류로는 서비스 내용에 따라 캐시서버, 웹서버, 미디어서버 등이 있는데 여기서는 이 3가지 서버의 콘텐츠 동기화 방법에 대해 기술했다.

연 재 순 서

1. 동적 콘텐츠 캐싱 기술
2. 글로벌 로드밸런싱 기술
3. 콘텐츠 동기화 기술 (이번호)

이윤근
씨디네트웍스 기술연구소 개발팀장



캐시서버의 경우 기능상 사용자의 요청을 받아 자기 자신의 내부 디스크에 캐시된 정보가 있으면 그것을 사용자에게 응답해 준다. 하지만 자기 자신의 내부 디스크에 정보가 없을 경우 오리진 웹서버에 데이터를 요청해 그 데이터를 자기 자신의 내부 디스크에 저장한 후 서비스를 요청한 사용자에게 응답을 해주게 된다. 캐시 서버의 경우 일반적으로는 캐시 서버의 자체 알고리즘상 항상 최신의 데이터를 유지하고, 없으면 오리진 웹서버에 요청을 해 가져다 주기 때문에 데이터 동기화 부분이 크게 문제될 것은 없다.

캐시서버 동기화

하지만 오리진 웹서버에서 특정 데이터를 삭제하거나, 같은 이름으로(더 나아가 같은 사이즈의 같은 이름으로) 변경된 경우 캐시서버가 이를 인지하지 못하는 경우가 있다. 물론 콘텐츠가 계속 새롭게 생성만 되고, 삭제되거나 변경되지 않는다면 문제될 것은 없다.

오리진 웹서버에서 특정 콘텐츠가 변경이 되거나 삭제됐는데도 캐시서버가 이를 인지하지 못해 이전의 콘텐츠를 계속 보유하고, 이를 서비스한다면 운영자의 의도와는 다르게 이전의 콘텐츠를 보여주거나 오리진 웹서버에서는 삭제된 콘텐츠를 보여주게 된다. 이런 오류를 피하기 위해 오리진 서버의 콘텐츠 변경에 따른 즉각적인 캐시 서버의 반영을 필요로 하게 된다.

오리진 서버에서 콘텐츠가 삭제되거나 변경됐을 때, 이를 캐시서버에 어떻게 즉각적으로 동기화를 시킬 것인가?

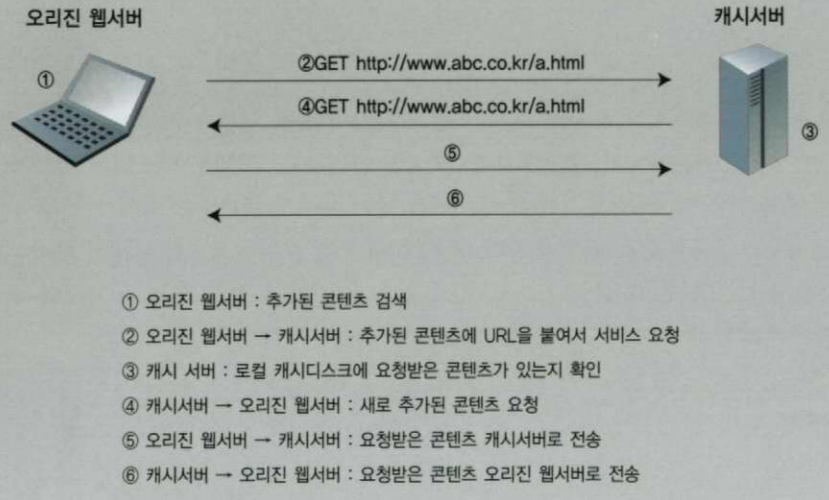
HTTP 스펙 내에는 Cache Consistency를 유지하기 위한 표준이 마련돼 있는데, IMS(If-Modified-Since)와 Expires 방식이 그것이다. 캐시서버가 사용자로부터 웹 콘텐츠에 대한 GET 요청을 받고 캐시에서 복사본을 찾을 때, 그 복사본을 사용자에게 보낼지 아니면 웹 서버로부터 다시 원본을 불러와서 캐시된 복사본을 업데이트 할지를 결정하기 위해서 캐시된 복사본의 나이(age)와 만료시간(expiration time)을 비교한다. 나이란 웹 콘텐츠가 캐시된 이래 경과한 시간을 의미하는데, 캐시된 복사본의 나이가 만료시간보다 적으면 캐시된 복사본을 보내고 그렇지 않으면 웹 서버로부터 원본을 불러와서 캐시 데이터를 업데이트한다.

이런 과정 중에 IMS와 Expires 방식이 사용된다. 즉 오리진 웹서버로부터 다시 원본을 불러

오기 위해 캐쉬서버가 HTTP Request 내에서 사용하는 것이 IMS 방식이고, 웹 콘텐츠의 만료시간(expiration time)을 명시하기 위해 웹서버가 HTTP Response 내에서 사용하는 것이 Expires 방식이다. 대부분의 웹 콘텐츠들은 만료시간이 명시되지 않기 때문에 보통 캐시 서버가 만료시간을 예측해 지정하게 된다.

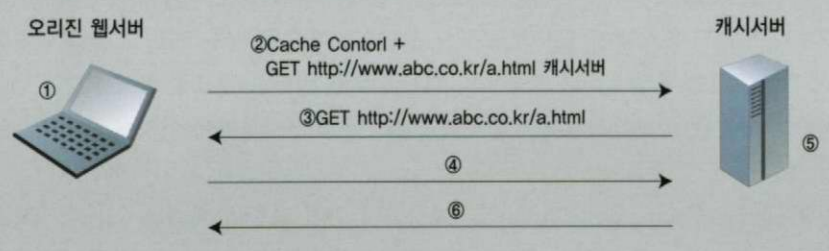
〈그림1〉은 오리진 웹서버에 콘텐츠가 새로이 추가된 경우 캐시서버에 동기화 시키는 방법이다. 이는 오리진 웹서버에서 콘텐츠의 추가된 내용을 감지해 서비스에 사용되는 URL을 이용해 마치 사용자처럼 캐시서버에 서비스를 요구하는 방법이다.

〈그림1〉 웹서버에 콘텐츠가 추가된 경우 캐시서버 동기화



① 오리진 웹서버 : 추가된 콘텐츠 검색
② 오리진 웹서버 → 캐시서버 : 추가된 콘텐츠에 URL을 붙여서 서비스 요청
③ 캐시 서버 : 로컬 캐시디스크에 요청받은 콘텐츠가 있는지 확인
④ 캐시서버 → 오리진 웹서버 : 새로 추가된 콘텐츠 요청
⑤ 오리진 웹서버 → 캐시서버 : 요청받은 콘텐츠 캐시서버로 전송
⑥ 캐시서버 → 오리진 웹서버 : 요청받은 콘텐츠 오리진 웹서버로 전송

〈그림2〉 웹서버에 콘텐츠가 변경된 경우 캐시서버 동기화



① 오리진 웹서버 : 변경된 콘텐츠 검색
② 오리진 웹서버 → 캐시서버 : 변경된 콘텐츠에 URL을 붙여서 서비스 요청
(오리진 웹서버로 확인을 강제하는 캐시 컨트롤 추가)
③ 캐시서버 → 오리진 웹서버 : 요청받은 콘텐츠 요청
④ 오리진 웹서버 → 캐시서버 : 요청받은 콘텐츠 캐시서버로 전송
⑤ 캐시서버 : 로컬 캐시디스크에 요청받은 콘텐츠 갱신
⑥ 캐시서버 → 오리진 웹서버 : 요청받은 콘텐츠 오리진 웹서버로 전송

캐시서버는 이런 요청을 사용자에게 서비스 요청을 받았을 경우와 동일하게 캐시된 콘텐츠 목록을 조사할 것이고, 당연히 캐시돼 있지 않으므로 오리진 웹서버에 접속해 해당 콘텐츠를 가져다가 캐싱을 할 것이다.

이때 정책적으로 결정할 사항이 있다. 캐시서버의 특성상 히트율이 높은 콘텐츠만을 캐시서버에 보관하게 되는데, 신규로 생성되는 콘텐츠를 무조건적으로 캐시서버에 로딩을 해 놓는다면 캐시서버에 저장돼 있는 상위권에 속하는 콘텐츠는 삭제될 가능성이 있게 되는 것이다. 신규로 생성되는 콘텐츠라 하더라도 히트율이 낮거나 사용자에게 요청될 가능성이 없는 콘텐츠를 캐시서버에 미리 로딩을 해놓는 것은 낭비가 될 소지가 있다. 예를 들어 오늘 새로운 이미지 콘텐츠를 하나 만들어서 오리진 웹서버에 올려 놓았는데, 동기화 프로그램이 이를 감지해 캐시서버에 미리 로딩을 해놓을 경우 이 이미지 콘텐츠가 며칠 뒤, 혹은 몇 주 뒤에 실제 웹페이지에 링크가 걸린다면 며칠동안 캐시서버에는 이용도 되지 않는 콘텐츠가 자리만 차지하고 있게 되는 것이다. 또 심하게는 며칠 뒤 히트율이 낮다는 이유로 자동으로 삭제가 될 수도 있게 되는 것이다. 이로 인해 전체적으로 캐시서버의 히트율이 낮아질 수도 있으므로 새로 생성되는 콘텐츠의 성격과 활용시기 등을 감안해서 캐시서버로 동기화를 진행해야 할 것이다.

〈그림2〉는 오리진 웹서버에 기존에 존재하던 콘텐츠가 새로운 내용으로 갱신되는 경우 캐시서버의 복사본을 갱신하는 방법이다. 같은 이름의 콘텐츠를 갱신하는 방법은 새로운 콘텐츠를 사전에 로딩하는 방법과 유사한데, 추가적으로 캐시 콘트롤을 포함해 요청하는 것이다. 캐시 콘트롤의 역할은 캐시서버가 특정 콘텐츠에 대해 관리하는 만료시간 등의 속성값을 무시하고 무조건 오리진 웹서버에 접속해 콘텐츠의 만료여부를 확인하도

록 하는 기능이다.

실제로 캐시서버가 오리진 웹서버에 접속해 요청받은 콘텐츠의 만료여부를 다시 확인할 때 콘텐츠의 사이즈나 생성시간, 만료시간 등이 다르기 때문에 현재 캐시서버 내부에 보관하고 있는 복사본이 변경됐다고 인식하고, 새롭게 복사본을 갱신하고 사용자의 요청에 응답을 하게 되는 것이다.

〈그림3〉은 오리진 웹서버에 있는 콘텐츠가 삭제될 경우 캐시서버에서도 삭제하는 방법을 나타낸 것이다. HTTP 프로토콜상에 Delete라는 Method가 존재해 오리진 웹서버에 해당 콘텐츠를 삭제할 수 있는 방법이 있는 것이다. 하지만 이런 것은 어디까지나 표준에 들어가 있는 방법이고, 실제 웹서버에는 이러한 Method를 지원하고 있지 않다. 삭제기능을 지원하면 웹서버의 콘텐츠가 언제 삭제될지 모르는 위험을 감수해야 하는데 누가 이런 기능을 원하겠는가? 보안상의 이유로 웹서버에서 지원을 하지 않기도 하고, 실제로 웹서버에 등록하는 콘텐츠의 소유자와 웹서버의 실행권한이 서로 맞지 않는 경우에도 삭제되지 않는다.

그렇다면 오리진 웹서버에서 삭제된 콘텐츠를 어떻게 캐시서버에 동기화 할 수 있을까? 이는 위에서 설명한 방법으로 응용이 가능하다. 〈그림2〉에서 설명한대로 캐시서버가 오리진 웹서버로 콘텐츠에 대한 정보를 확인하게 하면 어떻게 될까? 콘텐츠가 삭제된 경우도 콘텐츠가 변경된 경우와 같이 오리진 웹서버에 콘텐츠에 대한 강제적 확인을 하도록 요청을 보내면 캐시서버는 오리진 웹서버로 콘텐츠의 사이즈나 날짜, 만료시간에 대한 정보를 요청할 것이고, 오리진 웹서버에서는 해당 콘텐츠가 삭제됐으므로 해당정보를 삭제되고 없다고 보내줄 것이다. 그러면 캐시서버도 역시 요청받은 콘텐츠가 삭제됐다고 인식하고 로컬에 저장하고 있는 복사본을 삭제한 후 요청자에게 콘텐츠를 찾을 수 없다는 에러 메시지를 보낼 것이다.

일반적으로 캐시서버를 설명할 때 "오리진 웹서버 - 캐시서버 - 사용자" 3가지 기능으로 구분해 설명을 하는데 캐시서버의 동기화에 대한 설명을 하면서 오리진 웹서버와 사용자를 동일한 서버로 간주하고 설명을 해

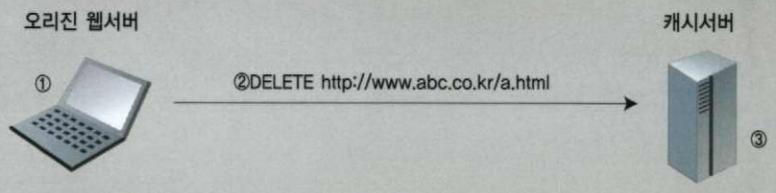
약간은 이해하는데 어려움이 있을 것이다. 하지만 시간을 가지고 천천히 살펴보면 그리 어렵지 않게 이해할 수 있으리라 생각된다.

캐시 동기화와 관련된 명령어는 HTTP 표준을 이용한 것이고, 보다 자세한 정보는 표준 문서인 RFC 문서를 참조하기 바란다(www.ietf.org/rfc.html).

웹서버 데이터 동기화

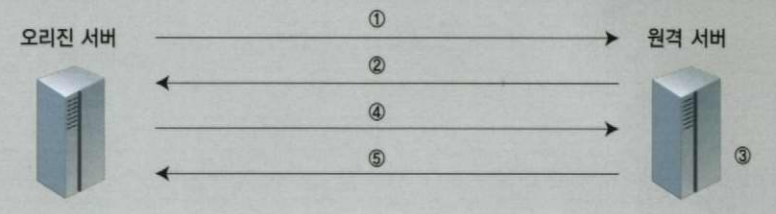
동일한 콘텐츠를 로컬 디스크에 보유하고 동일한 서비스를 하는 경우 여러 네트워크에 분포돼 있는 서버들은 서버의 안정성만큼이나 콘텐츠의 동기화도 중요할 것이다. 어느 한 서버에 콘텐츠가 동기화 되지 못해 어떤 콘텐츠가 존재하지 않는다면,

〈그림3〉 웹서버에 콘텐츠가 삭제된 경우 캐시서버 동기화



① 오리진 웹서버 : 삭제된 콘텐츠 검색
② 오리진 웹서버 → 캐시서버 : 삭제된 콘텐츠에 URL을 붙여서 서비스 요청
③ 캐시서버 : 캐시 로컬디스크에서 요청받은 콘텐츠 삭제

〈그림4〉 미러링 프로그램을 이용한 동기화



① 원격서버 → 오리진 서버 : 콘텐츠 리스트 요청
② 오리진 서버 → 원격서버 : 콘텐츠 리스트 제공
③ 원격서버 : 로컬 콘텐츠와 오리진 서버의 콘텐츠 리스트 비교
④ 원격서버 → 오리진 서버 : 새로 추가된 콘텐츠 요청
⑤ 오리진 서버 → 원격서버 : 요청받은 콘텐츠 전송

그 서버에 접속하는 사용자는 에러를 경험하게 될 것이다.

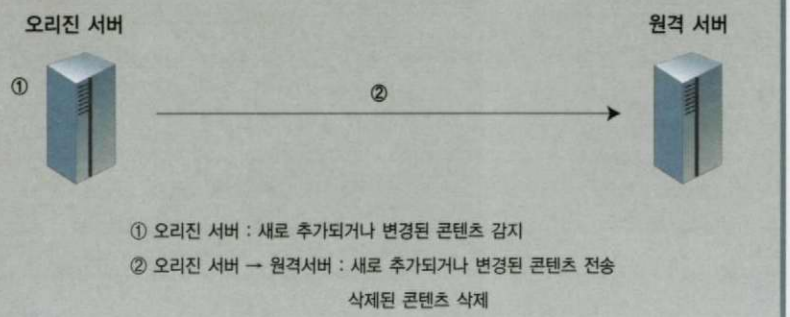
일반적으로 콘텐츠를 동기화하는 가장 쉬운 방법은 여러 미러링 사이트에서 이용하는 방법을 쓰는 것으로 미러링 관련 프로그램이나 유닉스계열 rsync를 이용하는 방법이다.

미러링 사이트는 한 서버의 부하가 심하거나 지리적인 이유로 인해 접속이 느린 경우 별도의 서버를 마련한 사이트이다. 이런 경우는 별도의 URL을 이용하기 때문에 지금 논하는 글로벌 로드밸런싱과는 조금 차이가 있지만, 이런 서버들을 운영하기 위해서는 필수적으로 데이터의 동기화가 필요하다. 미러링 사이트에는 CPAN, Tucows, Linux, FreeBSD, Apache, Mysql 등이 전 세계적으로 운영되고 있다. 미러링 프로그램을 이용한 경우는 〈그림4〉와 같다. 이 방법은 수동적인 방법이다. 콘텐츠의 추가, 변경여부 확인과 데이터의 이동에 관한 책임이 원격서버에 있는 것이다.

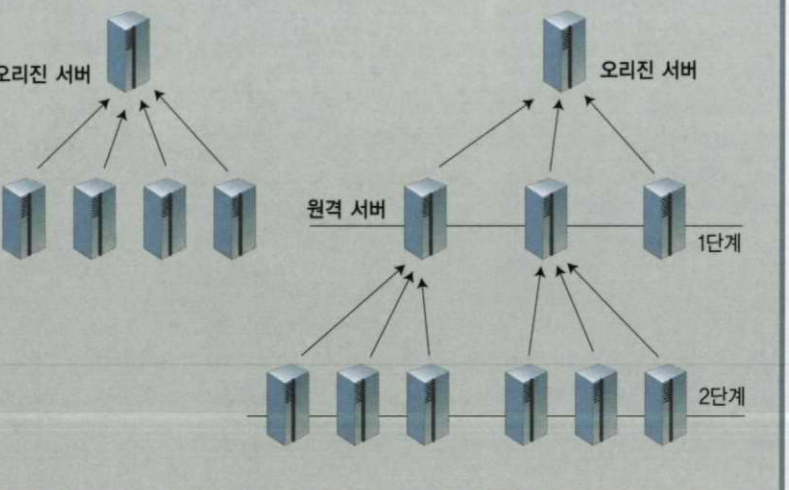
〈그림5〉는 rsync와 같은 동기화 프로그램을 이용하는 방법으로, 콘텐츠의 추가 변경여부 확인과 데이터의 이동에 관한 책임이 오리진 서버에 있다. 동기화를 위해서는 먼저 원본 서버에 있는 데이터의 변경 여부를 감지해야 한다. 이렇게 변경된 콘텐츠만을 골라서 원격 서버에 전송을 하는 것이다. 원격서버로의 콘텐츠 동기화 방법으로 유닉스 계열 서버에서 제공하는 rdist나 rsync, rcp 혹은 보안을 고려한 scp 등을 이용하거나 별도의 프로그램을 개발해 이용할 수도 있을 것이다.

프로그램에 따라서 새로이 추가되거나 변경된 콘텐츠를 오리진 서버에서 원격서버로 전송할 수도 있고, 때에 따라서 변경되거나 삭제된 리스트를 원격서버에 제공해 원격서버에서 필요한 조치를 할 수도 있다.

〈그림5〉 rsync를 이용한 동기화



〈그림6〉 동기화 계층 구조



이제까지 동기화의 주체에 따른 분류 방법을 살펴보았다. 원격서버의 수가 많다면 동기화 되는 속도와 오리진 서버의 부하를 고려해야만 할 것이다. 동기화를 위해서 오리진 서버를 별도로 둔다면 별문제겠지만, 효율적인 서버의 운영을 위해서 오리진 서버 역시 서비스에 투입을 해야 하는 경우라면 부하를 고려한 계층적인 동기화를 고려해 보아야 할 것이다.

〈그림6〉은 동기화의 계층 구조적인 측면을 살펴본 것이다. 단일 계층에서는 계층이 1단계여서 속도는 빠르게 동기화가 진행되겠지만, 원격서버들이 오리진 서버에 동시에 접속을 하게 된다면 오리진 서버의 부하를 고려해 보아야 한다. 또한 계층을 여러 단계로 둔다면 오리진 서버의 부하는 경감이 되겠지만, 1단계에 동기화가 이루어지고 이후에 2단계에 동기화가 진행되므로 동기화가 완료되는 시간은 2배 이상으로 길어질 것이다. 이러한 구조를 설계할 때 동기화에 따른 속도, 부하 문제를 고려해야 한다.

계층적인 동기화 방법 가운데 속도를 개선하기 위한 방법으로 1단계에 있는 원격서버들이 동기화를 진행하면서 로컬 디스크에 콘텐츠를 저장한 후 2단계 서버로 콘텐츠를 동기화 시켜주는 방법이 있다. 하지만 이 보다 1단계에 있는 서버가 콘텐츠를 받는 즉시 메

모리 상에서 2단계 서버로 콘텐츠를 넘겨주면서 로컬 디스크에 저장을 한다면 전체 동기화 되는 시간이 단축될 것이다.

또 다른 방법으로는 P2P를 이용한 방법이 있다. 요사이 인기를 끌고 있는 P2P 프로그램들을 보면 원하는 하나의 콘텐츠를 여러 Peer에서 조금씩 부분적으로 가져와서 이것들을 로컬에서 하나로 합쳐 완전한 콘텐츠를 구성하고 저장하고 있다.

이런 방법을 콘텐츠 동기화에 응용을 한다면 콘텐츠의 동기화가 계층구조가 아니라 그물구조를 이루게 돼 속도가 많이 개선될 것이다. 또한 다른 Peer(다른 원격서버)에서 완전한 콘텐츠를 가지고 있을 때만 가져오는 것이 아니라 일부분만을 가지고 있더라도 그 일부분의 콘텐츠를 제공받을 수 있다면 이론적으로는 오리진 서버의 부하없이 단일 계층구조의 동기화 속도와 거의 가깝게 동기화를 할 수 있다.

동기화시 고려해야 할 사항은 콘텐츠의 용도이다. 실제로는 드문 케이스이지만, 게임 패치 파일의 경우 서버에 기록된 콘텐츠의 작성시간을 기준으로 패치를 진행하기도 한다. 이런 경우 콘텐츠의 동기화시 콘텐츠의 부가정보인 작성시간까지 동기화 해야 하는 것이다. 이렇듯 콘텐츠의 용도에 따라 단순히 복사를 한다기보다는 콘텐츠의 부가적인 정보들까지도 동기화를 진행해야 하는 경우도 있다.

스트리밍 서버 데이터 동기화

일반적인 스트리밍 서버를 운영한다면 스트리밍을 위한 콘텐츠를 동기화 하는 방법은 위의 웹서버 콘텐츠 동기화 방법을 그대로 이용할 수 있다. 하지만 최근 스트리밍 데이터가 대용량화되고, 많은 양의 콘텐츠를 보유하고 있는 사이트에서 동일한 데이터를 여러 서버에서 가지고 서비스를 한다는 것은 스토리지 비용이 만만치 않게 들 것이다. 실제로 인터넷 교육업체와 인터넷 방송국, 인터넷 영화 등의 사이트들은 몇 테라 바이트의 데이터를 가지고 있으며, 콘텐츠를 SAN을 이용해 운영을 하고 있지만 비용 때문에 여러 네트워크에 분산 운영할 엄두를 못내고 있는 것이 현실이다.

이런 스토리지 비용에 대한 고민을 해결해 주는 것이 캐싱 서버이다. 아직까지 스트리밍 캐시서버가 보편화 돼 있지는 않지만, 일부 업체에서 만든 캐싱 서버들은 일반 HTTP 캐시서버와 비슷하게 동작하고 있다. HTTP 콘텐츠의 경우 사이즈가 비교적 작은데 비해 스트리밍 데이터의 경우 몇십 ~ 몇백 메가의 데이터이기 때문에 스트리밍 캐시서버라 하더라도 고용량의 스토리지를 필요로 하며, 실제 콘텐츠의 저장도 하나의 파일을 전체적으로 캐시하고 있다.

이러한 운용상의 문제로 인해 요즈음 프리픽스 캐싱이라는 개념이 나오고 있다. 프리픽스 캐싱이란 일부 히트율이 높은 콘텐츠의 경우는 콘텐츠 파일 전체에 대해 캐싱을 하고 있지만, 그렇지 않은 대부분의 콘텐츠에 대해서는 전체가 아니라 앞부분 일부에 대해서 캐싱을 하고 있다가 요청이 들어오면 처음에는 캐싱돼 있는 내용으로 서비스를 해주는 것이다.

이와 동시에 아직 캐싱돼 있지 않은 해당 콘텐츠의 뒷부분은 오리진 서버에서 가져다가 서비스를 해준다. 실제 많은 동영상 스트리밍 서비스의 이용 행태를 볼 때 끝까지 다 보는 것이 아니라 첫 부분 일부만 보고 끝내는 경우가 거의 대부분인 상황을 고려해 볼 때 스트리밍 프리픽스 캐싱은 디스크 절감에 큰 효과가 있을 것이다. 또한 80:20 법칙이 적용되면 전체 스토리지 용량의 30%만 캐시서버의 디스크로 운영하면 거의 대부분의 스트리밍 서비스를 처리할 수가 있다.

하지만 80:20 법칙에서 많이 벗어나 있거나, 음악과 같은 짧은 분량의 스트리밍 콘텐츠를 운영하는 경우에는 오리진 서버에서 스트리밍 캐시서버로 흘러가는 데이터 트래픽도 고려해야 한다. 여러 네트워크에 서버를 운영한다면 서버간의 동기화를 위한 트래픽도 비용이 들어가게 된다. 일반 파일처럼 한번에 동기화 되는 것이 아니라 캐시의 개념이 들어가 있기 때문에 같은 콘텐츠를 여러 번 전송해야 하는 경우가 발생할 경우는 비용도 고려해야 할 것이다. 스트리밍 프리픽스 캐시의 이용은 콘텐츠의 종류, 사용자들의 이용 패턴, 스토리지 비용, 네트워크 비용 등을 고려해서 결정해야 할 것이다.

최근 각종 콘텐츠의 작성, 수집, 관리, 배급하는 콘텐츠의 생산에서부터 활용, 폐기에 이르기까지 전 과정을 시스템화한 콘텐츠관리시스템(CMS, Contents Management System)의 논의가 활발하게 이루어지고 있으며, 특히 웹 콘텐츠 관련해 웹콘텐츠관리(WCM, Web Contents Management) 제품이 많이 나오고 있다. 여기에 글로벌 로드밸런싱과 관련해 여러 네트워크에 분산 배치돼 있는 원격서버로의 콘텐츠 배포기술이 포함된다면 좋지 않을까 생각된다.

참고 사이트

- **캐시 서버 관련 RFC 문서**
  HTTP 1.0 : RFC 1945(http://www.ietf.org/rfc/rfc1945.txt number=1945)
  HTTP 1.1 : RFC 2616(http://www.ietf.org/rfc/rfc2616.txt number=2616)
- **파일 동기화 관련 솔루션**
  브레인즈 스퀘어 – FastStream(http://www.brainz.co.kr/)
  피어링 포탈 – Parallel Harvest(http://www.peeringportal.com/)
- **프리픽스 스트리밍 캐시 솔루션**
  Vividon(http://www.vividon.com/)