웹 호스팅

웹 호스팅 = 콘텐츠 리소스를 저장, 중개, 관리하는 일

하드웨어(장비)나 소프트웨어를 직접 관리하기 어렵다면 -> 호스팅 업체에서 제공해주는 웹 호스팅 서비스를 사용하게 된다.

- 다수의 웹 사이트를 같은 서버에 **가상 호스팅**하는 방법
- 트래픽이 많아도 **안정적인 사이트**를 구축하는 법
- 웹 사이트 로딩 빠르게 하는 법

호스팅 서비스

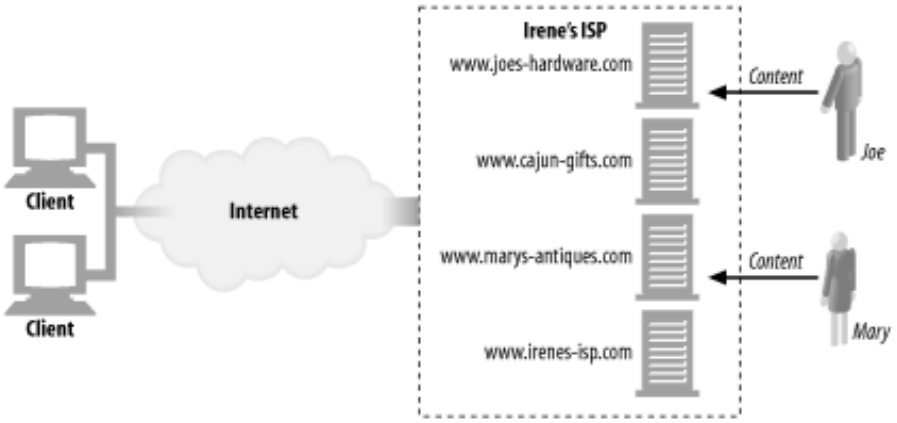
웹 서비스가 대세가 되면서 호스팅하려는 수요 up.

호스팅하기 위해 서버 및 냉난방 장치 구매, 도메인 이름 등록, 네트워크 대역폭을 구매할 기술과 시간이 부족해짐.

전문적으로 웹 호스팅 서비스를 제공하는 사업 등장.

간단한 전용 호스팅 방식

Figure 18-1. Outsourced dedicated hosting



아이린은 고성능 웹 서버들을 제공하는 호스팅 isp를 가지고 있음.

죠와 메리가 아이린의 isp로부터 서버를 별개로 구매하여 **전용 웹서버**를 사용.

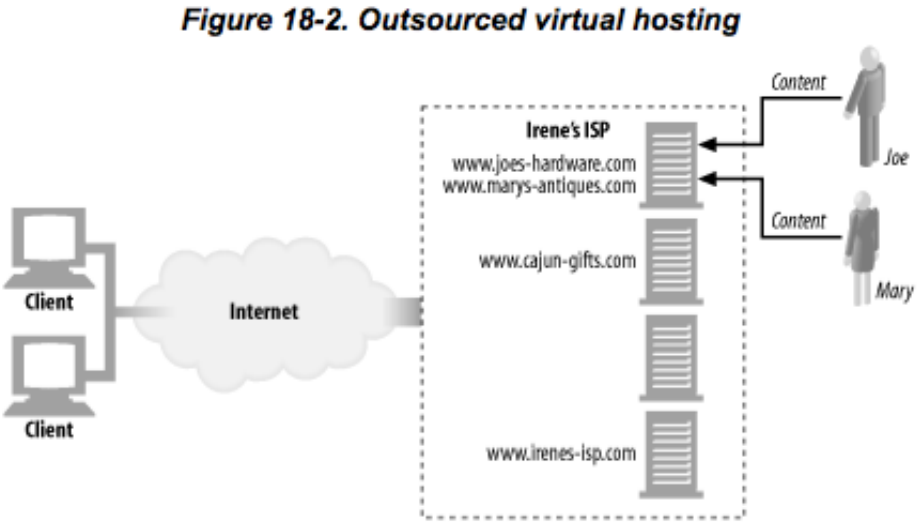
추후에 죠나 메리의 사이트가 인기가 많아지면 추가적인 서버를 신속하게 제공받을 수 있음.

그러나, 호스팅되는 모든 사이트가 인기있는 것은 아니기 때문에 **전용 웹서버를 독점하고 있는 것은 낭비**임.

그래서 웹 호스팅 사업자는 **장비 한대를 다수의 고객이 공유**하게 해서 더 저렴한 호스팅 서비스를 제공하게 됨 -> **가상 호스팅(공유 호스팅)**

가상 호스팅

사용자 관점에서는 각 웹 사이트가 다른 서버에서 호스팅 되는 것처럼 보이지만, 사실은 물리적으로 같은 서버에서 호스팅되게 하는 방식을 말함. -> 비용, 공간, 관리에 이점

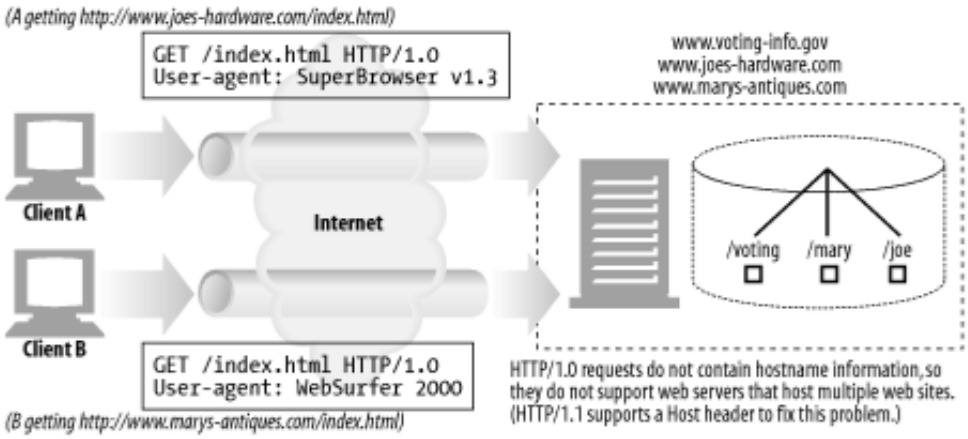


위와 같이 하나의 서버를 죠, 메리 둘다 사용하면 트래픽이 올라가기 전까지는 비용을 절약할 수 있음.

그러나 HTTP/1.0에서는 공용 웹 서버가 여러 개의 사이트를 가상 호스팅하고 있으면, **사용자가 어떤 사이트로 접근하려고 하는 것인지** 아는데 필요한 정보가 충분치 않음. -> 사이트의 **호스트 정보가 요청에서 제거 되기 때문에** 아래와 같이 **동일한 요청으로 간주**됨

(HTTP/1.1에서는 호스트 헤더를 제공해 문제없음)

1. 클라이언트 A가 joes/index.html에 접속 -> GET/index.html 요청이 웹 서버에 전송
2. 클라이언트 B가 marrys/index.html에 접속 -> GET/index.html 요청이 웹 서버에 전송



이는 초기 HTTP 설계 시 각 웹 서버가 한 웹 사이트만 호스팅할 것이라고 생각한 실수임.

-> url에 있는 호스트 명 정보는 필요없는 것으로 여겨 명세에서 단순히 경로 컴포넌트만 전송하도록 함.

-> 완전한 http 요청 메세지에 완전한 url을 포함하도록 변경함

-> 그러나 기존에 있던 앱들이 이 명세에 맞추어 업그레이드하는데 시간이 걸림

-> 그 와중에 다음 4가지를 사용한 가상 호스팅 기술이 나옴.

1. url 경로
2. 포트번호
3. ip 주소
4. host헤더

url 경로 통한 방법

공용 서버에 있는 각 사이트에 **서로 다른 url 경로**를 할당해서 구분

- 죠의 사이트 : http://www.example.com/joe/index.html -> GET/joe/index.html
- 메리의 사이트 : http://www.new-example/mary/index.html -> GET/mary/index.html

서버에 요청이 도착하면 **경로의 정보를 통해** 죠와 메리를 구분 가능

그러나, 불필요하고 혼잡스러운 접두어가 붙음 + http://www.example.com/ 와 같이 홈페이지로 갈 때 사용하는 url이 동작 안됨.

-> 좋지 않은 방법

포트번호를 통한 방법

경로 명을 변경하는 대신 죠와 메리 사이트에 **서로 다른 포트번호를 할당**

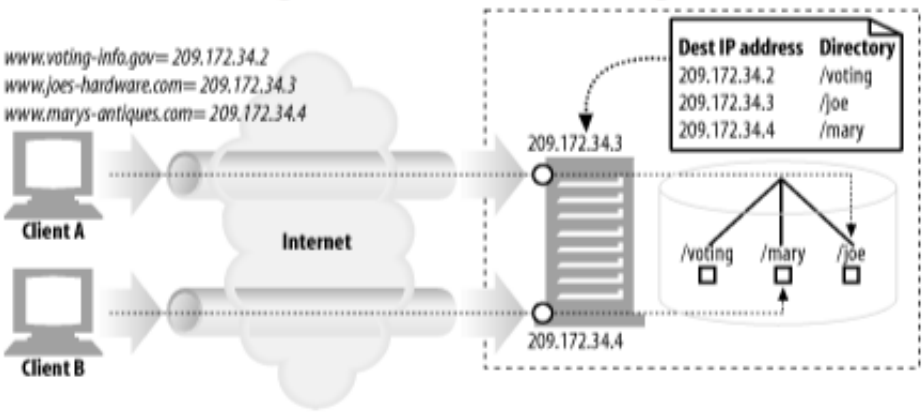
하지만 이 역시 같은 문제 -> 사용자는 url에 **비표준 포트**를 쓰지 않고서도 리소스를 찾기 원하기 때문

가상 ip 할당하는 방법

각 **웹 사이트에 유일한 ip주소를 한 개 이상 부여**하고 **모든 주소는 같은 공용 서버에 연결**되어 있음

서버는 **목적지 ip주소를 보고** 클라이언트가 어떤 사이트에 연결하려고 하는지 알 수 있음

Figure 18-4. Virtual IP hosting



- 209.172.34.3을 www.joes-hardwar.com
- 209.172.34.4를 www.marys-antiques.com

클라이언트는 죠의 사이트 url 경로로 요청하면, 209.172.34.3 주소에 있는 공용 웹 서버에 요청을 보내고 웹 서버는 해당 주소에 맞게 처리를 하고 응답을 내려줌.

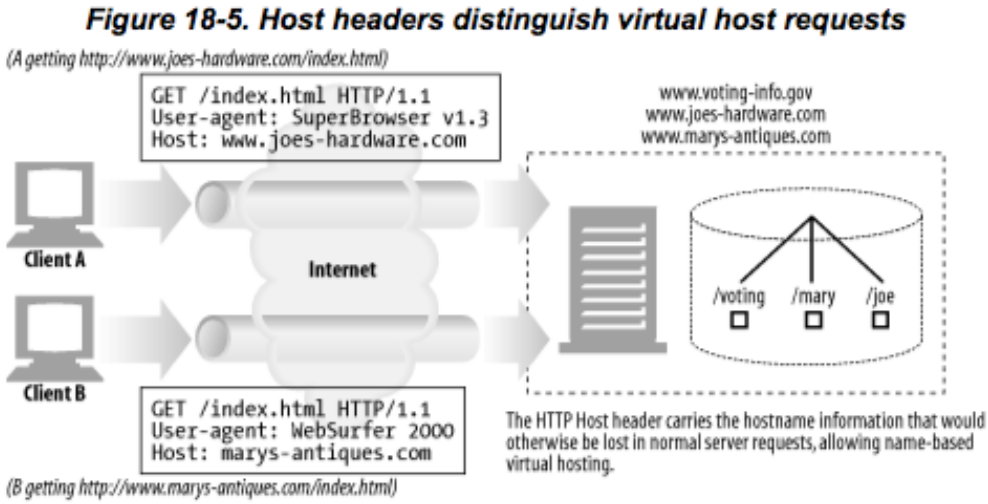
이는 잘 동작하지만, 다음과 같은 문제가 있음

- 장비의 ip주소 **개수에는 제한**있음
- ip주소는 **한정적**임. 사업자는 호스팅하고자 하는 모든 웹 사이트에 할당할 충분한 가상 IP 주소를 얻지 못할 수 있음.
- 부족한 IP 주소 문제는 서버를 복제하게 되면, 복제된 서버에 IP 주소를 부여해야 하므로 **IP 주소는 복제 서버 개수 만큼 더 필요**해짐.

host 헤더 통한 방법

위 ip 주소 낭비에 관한 문제를 피하기 위해 **http를 확장함**

모든 요청에 호스트명(+포트)를 host 확장 헤더에 기술해서 전달함.



HTTP/1.1 명세에는 Host헤더를 반드시 기술해야함. -> 대부분의 브라우저와 서버가 지원함.

```
Host = "Host" ":" 호스트[ ":" 포트]
```

다음과 같은 규칙들이 있음

- Host 헤더에 포트가 기술되어 있지 않으면, 해당 스킴의 기본 포트를 사용.
- URL에 IP 주소가 있으면, Host 헤더는 같은 주소를 포함.
- URL에 호스트 명이 기술되어 있으면,
  - Host 헤더는 같은 호스트 명을 포함야함.
  - Host 헤더는 호스트 명이 가리키는 IP 주소를 포함해서는 안 됨.

    > 여러 개의 사이트를 한 개의 IP 주소에 연결한 가상 호스트 서버에서 문제가 될 수 있기 때문.

- 클라이언트가 특정 프록시 서버를 사용시 -> Host 헤더에 프록시 서버가 아닌 원 서버의 호스트 명과 포트를 기술해야함.
- 웹 클라이언트는 모든 요청 메시지에 Host 헤더를 기술해야함.
- 웹 프록시는 요청을 전달하기 전에 요청 메시지에 Host 헤더를 추가해야함.
- HTTP/1.1 웹 서버는 Host 헤더 필드 없이 HTTP/1.1 요청 메시지를 받으면 `400` 상태 코드로 응답
- 가상 호스팅 지원하지 않는 원 서버는 요청 받는 호스트에 따라 리소스가 달라지지 않기 때문에 Host헤더값 무시함.

웹 서버의 Host 헤더 해석 방식

1. HTTP 요청 메세지에 전체 URL이 기술
    - 되어 있으면 : Host헤더에 있는 값은 무시하고 URL 사용
    - 안되어 있으면 : 요청의 Host헤더에서 호스트명 + 포트 를 가져옴
2. 호스트 결정 못하면 400(bad request)에러 응답 반환

안정적인 웹 사이트 만들기

일반적으로 호스팅하면 생기는 장애 - 서버 다운, 트래픽 폭증, 네트워크 장애

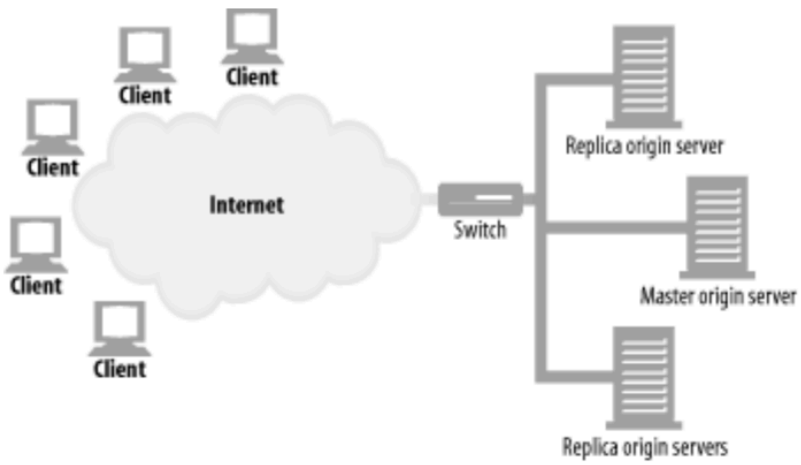
예측하고 대응하는 몇가지 방법이 있음

미러링된 서버 팜

서버 팜 : 서로 대신할 수 있고 식별할 수 있는 웹서버들 모음.

미러링을 통해 데이터를 하나 이상의 장치에 중복 저장함.

-> 서버에 있는 콘텐츠들은 한 곳에 문제가 생기면 다른 곳에서 전달할 수 있음

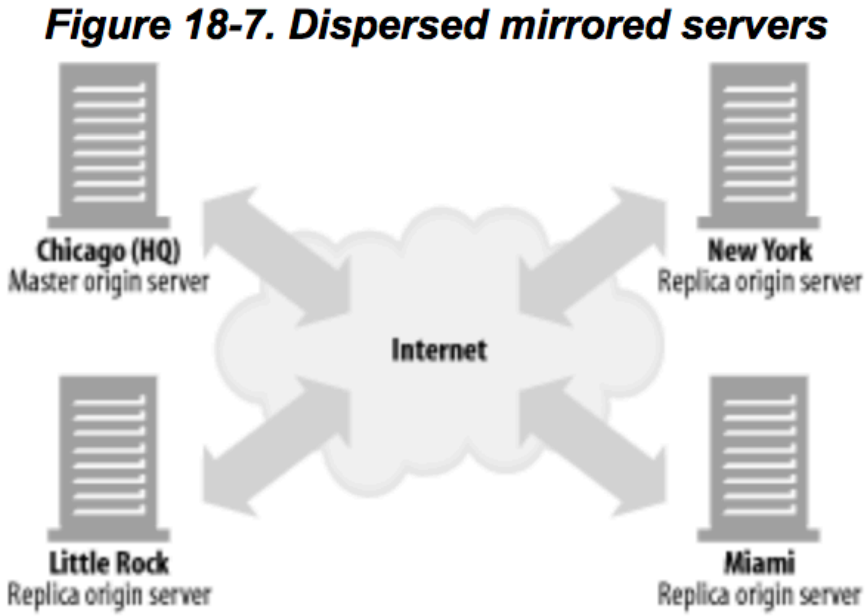


- master origin server : 원본 콘텐츠를 가지고 있는 것처럼 행동하는 마스터 서버

마스터 서버가 복제 서버에 콘텐츠를 퍼트림

- replica server : 미러링된 서버들

네트워크 스위치로 서버에 분산 요청해서 서버 팜에 간단하게 배포할 수 있음.



위 그림을 보면 클라이언트 요청이 특정 서버로 가는 두가지 방법이 있음

- HTTP 리다이렉션 : 콘텐츠에 대한 url이 마스터 서버(시카고)의 ip를 가리키고, 마스터는 **요청을 받는 즉시 복제 서버로 리다이렉트**
- DNS 리다이렉션 : 콘텐츠 url은 **네 개의 ip**를 가리킬 수 있고, **dns 서버가 클라이언트에게 전송할 ip주소를 선택할 수 있음**.

콘텐츠 분산 네트워크(CDN)

특정 콘텐츠의 분산을 목적으로 하는 단순한 네트워크

네트워크의 노드는 서버 / 대리 서버(리버스 프록시) / 프록시 서버 가 될 수 있음.

- CDN 대리 캐시

  복제 서버를 대신해 사용될 수 있음.

  다른점은 대리 서버는 원서버의 전체 콘텐츠를 복사 x -> **클라이언트가 요청하는 콘텐츠만 저장**함

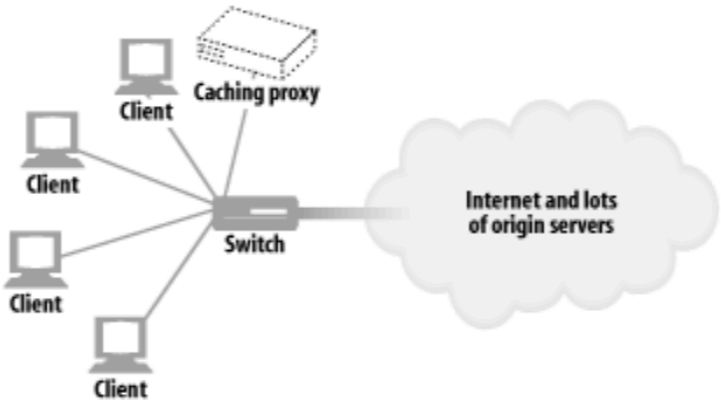
  따라서 대리 서버의 캐시에 콘텐츠가 분산되는 방식은 **해당 서버가 받는 요청에 따라 달라지게 됨**.

  -> 많은 요청이 있는 콘텐츠를 빠르게 제공

- CDN 프록시 캐시

  어떤 웹 서버의 요청도 다 받을 수 있음. -> 원본 서버 콘텐츠를 정확히 복제함.

Figure 18-8. Client requests intercepted by a switch and sent to a proxy



위 그림처럼 네트워크 장비(스위치 혹은 라우터)가 중간에서 트래픽을 가로채 프록시로 보냄.

서버 팜, CDN을 활용해 네트워크 트래픽과 콘텐츠를 분산시켜서 클라이언트와 서버 사이의 전송 시간을 단축시킬 수 있음.

그리고 속도를 높이는 또 다른 방법은 클라이언트가 받은 압축을 해제할 수 있다는 가정하에 콘텐츠를 인코딩하는 것. - 15장에서 설명했음