5장 - HTTP와 연계하는 웹 서버

1대로 멀티 도메인을 가능하게 하는 가상 호스트

- `HTTP/1.1`에서는 `하나의 HTTP 서버에 여러 개의 웹 사이트를 실행`할 수 있다.
- 이를 위해 `가상 호스트(Virtual Host)라는 기능`을 사용하고 있다.
  - 가상 호스트 기능을 사용하면 물리적으로는 서버가 1대지만 가상으로 여러 대가 있는 것처럼 설정하는 것이 가능하다.
  - `자체 운영 체제(OS)`와 `응용 프로그램`을 실행할 수 있는 환경을 제공
  - 가상 호스트는 소프트웨어적인 측면에서 물리적인 서버를 가상화하여 여러 개의 독립적인 가상 환경을 운영하는 소프트웨어
    - 반면에 가상 머신은 하드웨어적인 측면에서 가상화된 컴퓨터 시스템을 의미
- 인터넷에서 도메인명은 DNS에 의해서 IP 주소로 변환되고 나서 액세스하게 된다.
- 결국 리퀘스트가 서버에 도착한 시점에는 IP 주소를 기준으로 액세스 한다.
- 호스트명과 도메인 명을 완전하게 포함한 URI를 지정하거나, Host 헤더 필드에서 지정해야 한다.

통신을 중계하는 프로그램

HTTP는 클라이언트와 서버 이외에 `프록시, 게이트웨이, 터널과 같은 통신을 중계하는 프로그램과 서버를 연계하는 것도 가능`하다.

프록시(Porxy)

서버와 클라이언트의 양쪽 역할을 하는 `중계` 프로그램

- 리퀘스트를 서버에 전송하고 리스폰스를 클라이언트에게 전송한다.
- 기본적인 동작은 클라이언트로부터 받은 리퀘스트를 다른 서버에 전송하는 것이다.
- 오리진 서버(Origin Server : 리소스 본체를 가진 서버)로부터 되돌아온 리스폰스는 프록시 서버를 경유해서 클라이언트에 돌아온다.
- 중계할 때에는 Via 헤더 필드에 경유한 호스트 정보를 추가해야 한다.

- 프록시 서버를 사용하는 이유
  - `캐시`를 사용해서 `네트워크 대역` 등을 효율적으로 사용하는 것
    - 네트워크 대역 : 네트워크에서 데이터를 전송하는 데 사용할 수 있는 최대 데이터 전송 속도
  - 조직 내에 특정 웹 사이트에 대한 `액세스 제한`

    → 지리적 제한 우회 : 일부 서비스는 특정 지역에만 제한되어 있을 수 있습니다. 프록시를 사용하면 해당 지역으로 보이는 IP 주소를 사용하여 지리적 제한을 우회할 수 있습니다.
  - `액세스 로그를 획득하는 정책`을 철저하게 지키려는 목적

    → 익명성 보호 : 프록시를 통해 인터넷에 접속할 때 클라이언트의 실제 IP 주소를 숨길 수 있습니다.
- 프록시의 사용 방법 (2개의 기준으로 분류)
  1. 캐시하는지 안하는지 여부로 구분
  2. 메시지를 변경하는지 안하는지로 구분

캐싱 프록시 (Cashing Proxy)

- 프록시 서버 상에 리소스 `캐시`를 보존해 두는 타입의 프록시
- 다시 같은 리소스에 리퀘스트가 온 경우, 오리진 서버로부터 리소스를 획득하는 것이 아닌 캐시를 리스폰스로서 되돌려 주는 것

  → 일반적으로 캐싱 프록시에서는 메모리에 데이터를 저장하는 `인메모리 데이터베이스`가 많이 사용됩니다.

---

1. 웹 프록시 서버:
   웹 프록시 서버는 가장 일반적인 캐싱 프록시의 예시입니다. 대표적으로 Varnish, Squid, `Nginx` 등이 있습니다. 이러한 웹 프록시 서버는 웹 서버와 클라이언트 사이에서 동작하며, `이전에 요청된 웹 페이지나 리소스를 캐시에 저장하여 동일한 요청이 있을 때 웹 서버로부터 데이터를 다시 가져오지 않고 캐시된 데이터를 제공`합니다. 이를 통해 웹 페이지의 로딩 속도를 향상시키고 서버의 부하를 줄일 수 있습니다.
2. CDNs (Content Delivery Networks):
   콘텐츠 전송 네트워크(CDN)는 전 세계에 분산된 서버 네트워크를 통해 콘텐츠를 배포하는 시스템입니다. CDNs는 캐싱 프록시의 개념을 활용하여 웹 콘텐츠를 캐시 서버에 저장하고, 사용자가 가장 가까운 캐시 서버에서 콘텐츠를 제공하여 로딩 속도를 향상시킵니다. 대표적인 CDN 서비스로는 Cloudflare, Akamai, Fastly 등이 있습니다.

3. 데이터베이스 캐싱:
   데이터베이스 캐싱은 데이터베이스 쿼리의 응답 결과를 캐시에 저장하여 동일한 쿼리에 대한 처리 속도를 향상시키는 방법입니다. 캐시에 저장된 데이터는 원래 데이터베이스에 접근하지 않고 바로 제공됩니다. Redis는 데이터베이스 캐싱에 자주 사용되는 인메모리 데이터베이스입니다. 다른 데이터베이스 시스템인 Memcached도 널리 사용됩니다.
4. API 캐싱:
   API 호출을 캐시하여 API 서버의 부하를 줄이고 응답 시간을 단축시키는 방법도 있습니다. API 응답은 캐시에 저장되고, 동일한 요청이 있을 때 캐시된 데이터를 반환합니다. 이를 통해 API 호출 횟수를 줄이고 성능을 향상시킬 수 있습니다. 캐싱 프록시로는 Redis나 Memcached와 같은 인메모리 데이터베이스를 사용할 수 있습니다.

투명 프록시 (Transparent Proxy)

- `메시지 변경`을 하지 않는 타입의 프록시
- 메시지에 변경을 가하는 타입의 프록시 → 비투과 프록시

게이트웨이(Gateway)

- 다른 서버를 중계하는 서버
- 리퀘스트를 `리소스를 보유한 서버인 것처럼` 수신
- 동작은 프록시와 매우 유사
- 다른점 : 게이트웨이의 경우 그 다음에 있는 서버가 HTTP 서버 이외의 서비스를 제공하는 서버
- 클라이언트와 게이트웨이 사이를 `암호화 등으로 안전(Secure)하게 접속함으로써 통신의 안전성을 높이는 역할`

---

1. 프로토콜 변환: 게이트웨이는 사용자가 웹 브라우저를 통해 HTTP 프로토콜로 데이터를 요청할 때, 해당 요청을 `내부 서버에서 사용하는 다른 프로토콜로 변환`합니다. 예를 들어, 게이트웨이는 요청을 HTTP에서 웹 서비스가 사용하는 특정 프로토콜로 변환하여 내부 서버에 전달합니다.
2. 로드 밸런싱: 웹 서비스에 많은 사용자가 접속하면 `트래픽이 증가`하고 `부하가 분산`되어야 합니다. 게이트웨이는 로드 밸런싱을 통해 여러 대의 내부 서버에 트래픽을 분산

시킵니다. 이는 사용자 요청을 처리하는 서버의 부하를 줄이고 응답 시간을 최적화하는 데 도움이 됩니다.

3. 보안 기능: 게이트웨이는 웹 서비스에 접근하는 사용자들과 내부 서버 사이의 보안을 강화합니다. 게이트웨이는 방화벽 기능을 제공하여 악의적인 트래픽이나 해킹 시도를 차단합니다. 또한, 암호화 프로토콜을 사용하여 데이터의 안전한 전송을 보장합니다.
4. 캐싱 기능: 게이트웨이는 일부 동적 컨텐츠를 캐싱하여 사용자 요청에 더 빠르게 응답할 수 있습니다. 예를 들어, 반복적으로 요청되는 이미지나 페이지를 캐싱하여 내부 서버에 다시 요청하지 않고 바로 사용자에게 제공할 수 있습니다.

터널(Tunnel)

- 서로 떨어진 두 대의 클라이언트와 서버 사이를 중계
- 접속을 주선하는 중계 프로그램
- 요구에 따라서 다른 서버와의 통신 경로를 확립
- 클라이언트는 SSL 같은 암호화 통신을 통해 서버와 안전하게 통신을 하기 위해 사용
- 터널 자체는 HTTP 리퀘스트를 해석하려고 하지 않는다.

리소스를 보관하는 캐시

- 캐시(Chase)란 ?
  - `프록시 서버`와 `클라이언트의 로컬 디스크`에 보관된 `리소스의 사본`
  - 리소스를 가진 서버의 액세스를 줄이기 때문에 `통신량과 통신 시간을 절약`
- 캐시 서버란 ?
  - 프록시 서버의 하나로 캐싱 프록시로 분류
  - 장점
    - 캐시를 이용함으로써 같은 데이터를 몇 번이고 오리진 서버에 전송할 필요가 없다.
    - 클라이언트는 네트워크에서 가까운 서버로부터 리소스를 얻을 수 있게 되어 서버는 같은 리퀘스트를 매번 처리하지 않아도 된다.

- 캐시의 유효기간
  - 캐시되어 있는 `리소스의 유효성`과 관계
  - 클라이언트의 요구나 캐시의 유효 기간 등에 의해서 오리진 서버에 리소스의 유효성을 확인하거나 새로운 리소스를 다시 획득하러 가게 되는 경우가 있다.
- 클라이언트 측의 캐시
  - 클라이언트가 사용하고 있는 브라우저에도 캐시를 가질 수 있다.
  - 인터넷 임시 파일 : 클라이언트가 보존하는 캐시
  - 브라우저가 유효한 캐시를 가지고 있는 경우, 같은 리소스의 액세스는 서버에 액세스하지 않고 로컬 디스크로부터 불러온다.