♥

하이티센 3주차

웹서버란 무엇인가

was란 무엇인가

웹서버랑 was의 차이

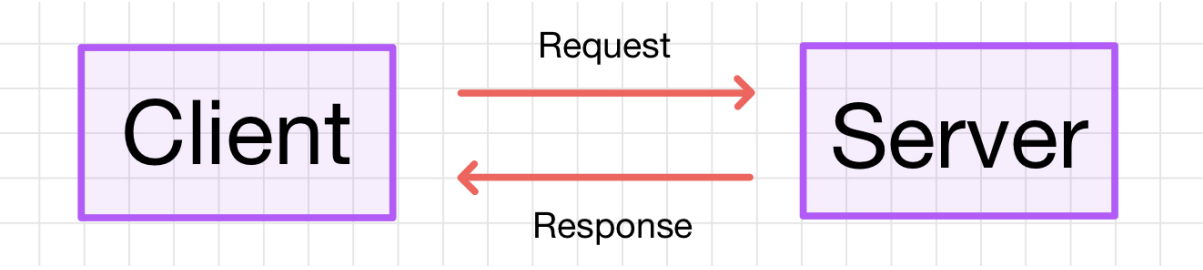
웹서버끼리의 비교 : 아파치 엔진액스
was끼리 비교 : 톰캣 jboss

+서블릿 컨테이너

웹 서버란 무엇인가?

웹의 동작원리

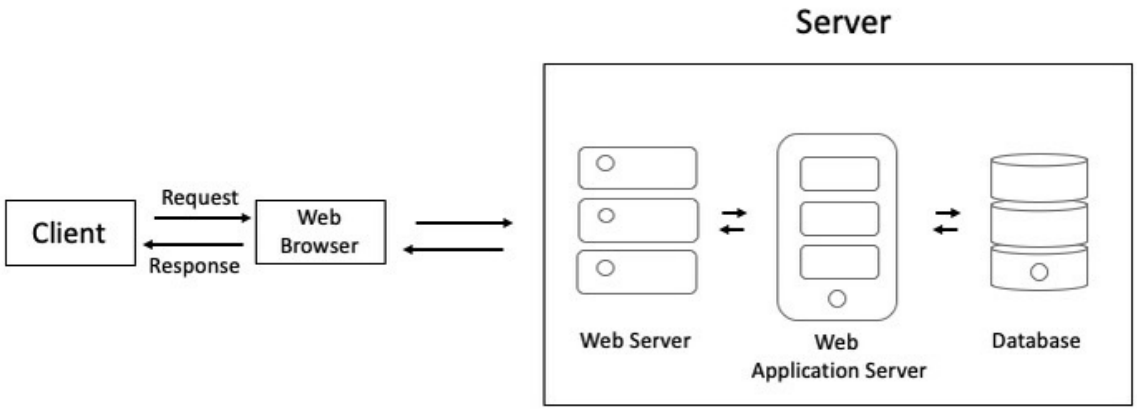
클라이언트(사용자)가 웹 브라우저(크롬, 익스플로러 등등)를 사용하여 서버에 요청을 보내고 서버가 해당 요청에 대해 응답하는 것이 Web 기본 동작이다



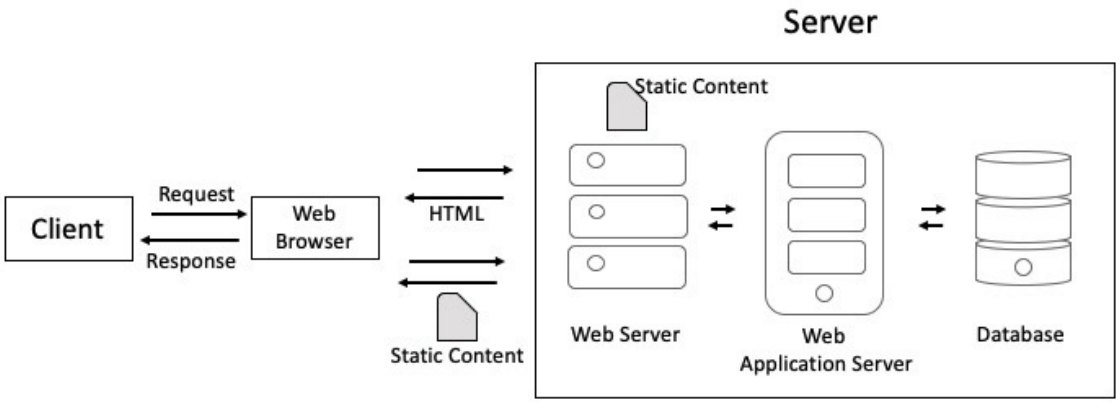
기본 동작을 그림으로 간단하게 표현하면 다음과 같다

웹브라우저에서 검색하고 주문을 하고 페이지를 새로고침한다.

이런 행위들은 사실 매번 서버에게 새로운 요청을 하고 그 응답에 대한 결과물로 웹 브라우저의 화면을 접한다



Web Server



웹 서버는 말 그대로 작성된 html 페이지 등을 네트워크망에 종속되지 않고, 웹서비스를 할 수 있도록 하는 어플리케이션이다.

- 웹서버는 소프트웨어와 하드웨어로 구분된다.

1) 소프트웨어 웹서버 : 웹 브라우저 클라이언트로부터 HTTP 요청을 받아들이고, 정적인 컨텐츠 (HTML, .jpeg, .css 문서)와 같은 웹 페이지에서 흔히 찾아 볼 수 있는 자료 콘텐츠에 따라 HTTP 에 반응하는 컴퓨터 프로그램

2) 하드웨어 웹서버 : 위에 언급한 기능을 제공하는 컴퓨터 프로그램을 실행하는 컴퓨터

- 클라이언트가 서버가 준 HTML 을 볼 때 안에 삽입된 이미지를 한 번에 가져온 것이 아니라 우선 HTML 문서를 먼저 받고 그에 맞게 필요한 이미지 파일들을 다시 서버에 요청하면 그 때 이미지 파일들을 받아온다

- Web Server 의 기능

- HTTP 프로토콜을 기반으로 하여 클라이언트(웹 브라우저 또는 웹 크롤러) 의 요청을 서비스하는 기능을 담당한다.

- 요청에 따라 다음의 기능 중 적절하게 선택하여 수행한다.

* 기능 1) 정적인 컨텐츠 제공

WAS 를 거치치 않고 바로 자원을 제공한다.

* 기능 2) 동적인 컨텐츠 제공을 위한 요청 전달

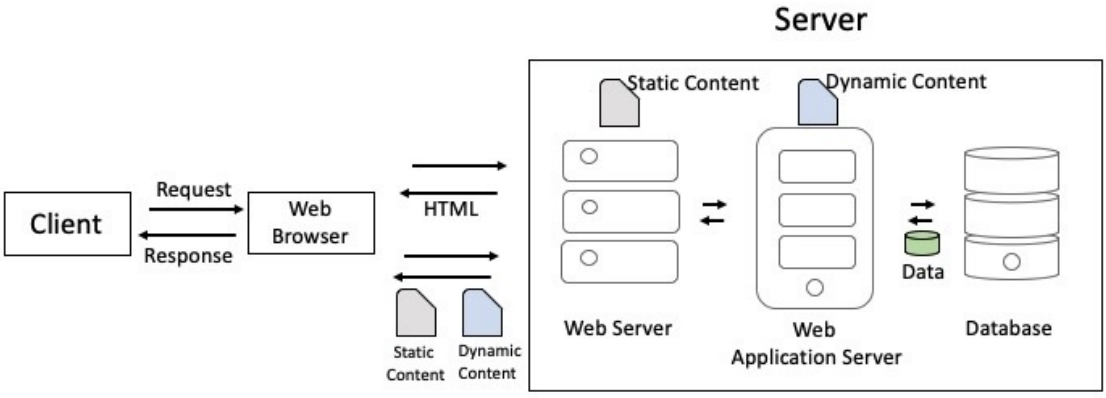
클라이언트의 요청을 WAS 에 보내고, WAS 가 처리한 결과를 클라이언트에게 전달(응답)한다.

클라이언트는 일반적으로 웹브라우저를 의미한다.

- Web Server 의 예 : Apache Server, Nginx

++ 이때 웹서버를 이용하면 Application Server 까지 가지 않고 빠르게 다시 응답할 수 있다. 웹서버에서는 정적 컨텐츠만 처리하도록 기능을 분배하여 서버의 부담을 줄이는 것이다.

WAS 란 무엇인가?



- WAS 란 DB 조회나 다양한 로직 처리를 요구하는 동적인 컨텐츠를 제공하기 위햇 만들어진 Application Server 다.
- HTTP 를 통해 컴퓨터나 장치에 애플리케이션을 수행해주는 미들웨어 (소프트웨어 엔진) 이다.
- “웹 컨테이너(Web Container)” 혹은 “서블릿 컨테이너(Servlet Container)” 라고도 불린다.

  - 컨테이너란 JSP, Servlet 을 실행시킬 수 있는 소프트웨어를 말함

  - 즉, WAS 는 JSP, Servler 구동환경을 제공한다.

- 웹 페이지에는 정적 컨텐츠(Static Content) 뿐만 아니라 동적 콘텐츠(Dynamic Content) 가 모두 존재한다. 만약 위의 웹서버만 이용할 시 미리 모든 데이터를 만들어 놓고 클라이언트를 기다려야 한다 (하지만 이는 불가능)
- 그래서 보통 WAS는 요청에 맞는 데이터를 DB에서 가져와 비즈니스 로직에 맞게 그때그때 결과를 만들어 클라이언트에게 제공한다.
- 이로써 한정된 자원을 좀 더 효율적으로 사용할 수 있다.

역할

WAS = WEB SERVER + WEB CONTAINER

- Web Server 기능들을 구조적으로 분리하여 처리하고자 하는 목적으로 제시되었다.

  - 분산 트랜잭션, 보안, 메시징, 쓰레드 처리 등의 기능을 처리하는 분산환경에서 사용된다.

  - 주로 DB서버와 같이 수행됨

[정리]

- 프로그램 실행 환경과 DB 접속기능 제공
- 여러 개의 트랜잭션(논리적인 작업 단위 관리 기능)

- 업무를 처리하는 비즈니스 로직 수행

WAS 의 예 : Jboss, (Tomcat)

[WEB과 WAS의 비교]

Web Container의 유무로 WEB 과 WAS를 나눌 수 있다.

웹서버는 HTML 문서 같은 정적 컨텐츠를 처리하는 것(HTTP 프로토콜을 통해 읽힐 수 있는 문서)

WAS 는 asp, php, jsp 등 개발 언어를 읽고 처리하여 동적 컨텐츠, 웹 응용 프로그램 서비스를 처리하는 것이다.

[차이점]

목적이 다름

웹 서버는 정적인 데이터를 처리하는 서버이다. 이미지나 단순 html 파일과 같은 리소스를 제공하는 서버를 웹서버를 통하면 WAS를 이용하는 것보다 빠르고 안정적이다.

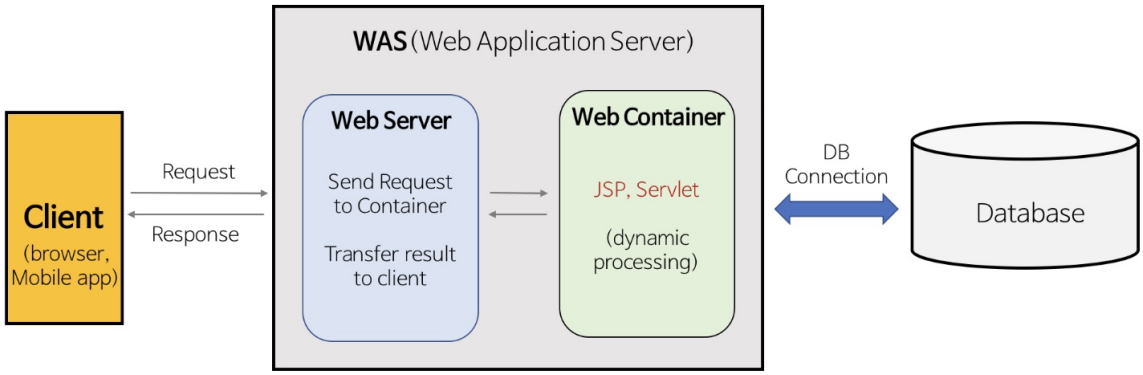
WAS 는 동적인 데이터를 처리하는 서버이다. DB 와 연결되어 데이터를 주고 받거나 프로그램으로 데이터 조작이 필요한 경우에는 WAS 를 활용해야 한다.

우리가 만드는 웹페이지는 정적 컨텐츠와 동적 컨텐츠를 함께 노출하게 한다.

만약 WAS 가 정적 데이터를 처리하게 되면, 동적 컨텐츠의 처리가 지연이 될 것이고, 이로 인한 페이지 노출 시간이 늘어나게 된다. WAS 는 동적 처리에 최적화 되어 있는 서비스 이기 때문에 처리속도를 위해, 정적 처리는 웹서버에서 처리를 하고 동적 컨텐츠는 WAS 에서 처리하게 된다.

사용자가 클라이언트(브라우저)에 요청을 하게 되면 이를 웹서버에서 반응하여 WAS 의 처리를 거쳐 웹페이지로 다시 웹서버에서 클라이언트(브라우저)에 응답 메시지를 주는 것이다.

Web Server 와 WAS 를 구분하는 이유



- Web Server 가 필요한 이유

: 클라이언트(웹 브라우저)에 이미지 파일(정적 컨텐츠)을 보내는 과정을 생각해보자

- 이미지 파일과 같은 정적인 파일들은 웹 문서(HTML)가 클라이언트로 보내질 때 함께 가는 것이 아니다.

- 클라이언트는 HTML 문서를 먼저 받고 그에 맞게 필요한 이미지 파일들을 다시 서버로 요청하면 그때서야 이미지 파일을 받아온다.

- Web Server 를 통해 정적인 피일들을 Application Server 까지 가지 않고 앞단에서 빠르게 보내줄 수 있다.

⇒ 따라서 웹 서버에서는 정적 컨텐츠만 처리하도록 기능을 분배하여 서버의 부담을 줄일 수 있다.

- WAS 가 필요한 이유

: 웹 페이지에는 정적 컨텐츠와 동적 컨텐츠가 모두 존재한다.

- 사용자의 요청에 맞게 적절한 동적 컨텐프를 만들어서 제공해야 한다.

- 이때, 웹 서버만을 이용한다면 사용자가 원하는 요청에 대한 결과값을 모두 미리 만들어 놓고 서비스를 해야한다.

- 하지만 이렇게 수행하기에는 자원이 절대적으로 부족하다.

⇒ 따라서 WAS 를 통해 요청에 맞는 데이터를 DB 에서 가져와서 비즈니스 로직에 맞게 그때 그때 결과를 만들어서 제공함으로써 자원을 효율적으로 사용할 수 있다.

- WAS 가 Web Server의 기능도 수행하면 되지 않나…?

1. 기능을 분리하여 서버 부하 방지

- WAS는 DB 조회나 다양한 로직을 처리하느라 바쁘기 때문에 단순한 정적 컨텐츠는 웹 서버 에서 빠르게 클라이언트에게 제공하는 것이 좋다.

- WAS는 기본적으로 동적컨텐츠를 제공하기위해 존재하는 서버이다.

- 만약 정적 컨텐츠 요청까지 WAS 가 처리한다면 정적 데이터 처리로 인해 부하가 커지게 되고, 동적 컨텐츠의 처리가 지연됨에 따라 수행 속도가 느려진다.

- 즉, 이로 인해 페이지 노출 시간이 늘어나게 될 것이다.

2. 물리적으로 분리하여 보안 강화

- SSL 에 대한 암복호화 처리에 웹서버를 이용한다.

: SSL(Secure Socket Layer) 프로토콜은 웹서버와 브라우저 사이의 보안을 위해 만들었다. SSL 은 Certificate Authority라 불리는 서드 파티로부터 서버와 클라이언트의 인증을 하는데 사용된다.

3. 여러 대의 WAS를 연결 가능

- Load Balancing을 위해서 웹서버를 사용

- fail over(장애 극복), fail back 처리에 유리

- 특히 대용량 웹 어플리케이션의 경우(여러 개의 서버 사용) Web Server 와 WAS 를 분리하여 무중단 운영을 위한 장애 극복에 쉽게 대응할 수 있다.

ex) 앞 단의 웹서버 에서 오류가 발생한 WAS 를 이용하지 못하도록 한 후 WAS 를 재시작함으로써 사용자는 오류를 느끼지 못하고 이용할 수 있다.

4. 여러 웹 어플리케이션 서비스 가능

ex) 하나의 서버에 PHP Application 과 Java Application 을 함께 사용하는 경우

5. 기타

- 접근 허용 IP 관리, 2대 이상의 서버에서의 세션 관리 등도 Web Server 에서 처리하면 효율적이다.

핵심은 정적 컨텐츠와 동적 컨텐츠가 모두 존재한다는 것이다.

이때 WAS 에게 정적, 동적 컨텐츠의 생성을 맡겨버리면 이미 DB 조회를 비롯한 다양한 로직을 처리하고 있는 WAS 에 부담을 줄 수 있으며 느려지게 될 수 있다.

그렇기에 Web Server 에서 처리할 수 있는 작업은 빠르게 작업하여 클라이엍으에게 응답하는 것이 좋다.

또한 Web Server 를 WAS 앞단에 배치하면서 WAS의 환경설정 파일등을 외부에 노출시키지 않아도 된다.

즉, 자원 이용의 효율성 및 장애 극복, 배포 및 유지보수의 편의성을 위해 웹서버 와 WAS 를 분리한다.

대표적인 웹서버

아파치

: Apache 는 Apache Software Foundation 에서 만든 웹 서버 프로그램

: 거의 모든 OS 에서 실행되고, 다른 유명한 소프트웨서 프로젝트와의 문서화가 잘 되어 있고 통합 지원 등의 이점이 있다.

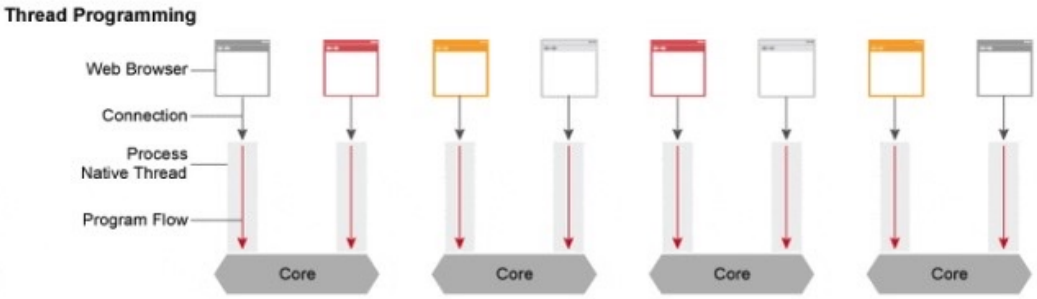
손님이 올때마다 프로세스를 새로 생성하는 방식이 있고 한 프로세스 안에서 스레드를 새로 생성하는 방식이 있다.

mpm prefork, 즉 손님마다 프로세스를 두는 방식은 손님이 새로 올 때마다 상담 테이블을 따로 가져와서 앉히는 것 그리고 이 테이블을 계속 돌아다니면서 여러 손님들을 동시에 상대

mpm worker, 한 프로세스에서 손님마다 스레드를 생성한다는 건 가로로 길다란 테이블을 하나 두고, 손님이 올 때마다 나란히 앉힌 다음 역시 좌우로 돌아다니면서 동시에 업무를 처리하는 것

- **주요 특징**

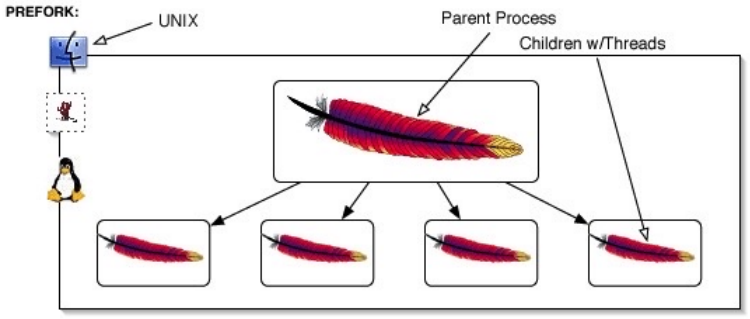
1. **스레드 / 프로세스 기반 구조**



- Apache 는 클라이언트 요청당 하나의 스레드가 처리하는 구조
- 사용자가 많으면 스레드 생성, 메모리 및 CPU 낭비가 심하다.

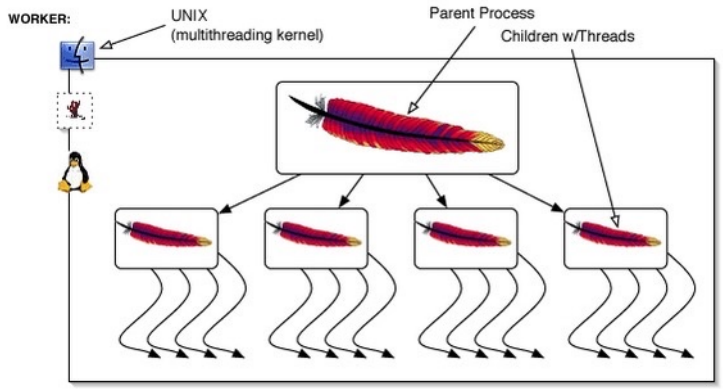
2. **MPM(Multi-Process Module)**

*** PreFork 방식 (다중 프로세스)**



- Client 요청에 대해 Apache 자식 프로세스를 생성하여 처리한다.
- 요청이 많을 경우 Process를 생성하여 처리한다. ← 이 방식은 Apache 설치 시 default 로 설정이 되어 있다.
- 하나의 자식 프로세스당 하나의 스레드를 갖는 구조로, 자식 프로세스는 최대 1024개 까지 가능하다.
- 스레드간 메모리 공유를 하지 않는다. ← 이 방식은 독립적이기에는 안정적이지만 메모리 소모가 크다는 단점이 있다.
- 실행 중인 프로세스를 복제하여 실행한다. (메모리 영역까지 복제)
- 응답 프로세스를 미리 띄워놓고 클라이언트 요청시 자식 프로세스가 반응하는 방식
- 디버깅이 빈약한 플랫폼에서 쉬운 디버깅이 가능
- 일반적으로 Single CPU 또는 Dual CPU 에서 성능이 좋다.

*** Worker 방식 (멀티 프로세스 - 스레드)**



- Prefork 보다 메모리 사용량이 적고, 동시 접속자가 많은 사이트에 적합 ← 각 프로세스의 스레드를 생성해 처리하는 구조
- 스레드 간의 메모리 공유가 가능
- 프로세스당 최대 64개의 스레드 처리가 가능하며, 각 스레드는 하나의 연결만을 부여받음
- 일반적으로 Multi CPU 시스템에서 성능이 좋다.

3. **동적 컨텐츠 처리**

4. **다양한 모듈**

호환성/ 확정성/ 안정성

* 단점

- 클라이언트 접속 시 마다 프로세스 또는 스레드를 생성하는 구조이기 때문에 대량의 클라이언트(1만 이상)가 동시 접속한다면 CPU/메모리 사용이 증가하고 프로세스/스레드 생성 비용이 드는 등 요청에 한계가 있다.
- Apache 서버의 프로세스가 blocking 되면 요청을 처리하지 못하고 처리가 완료될 때 까지 계속 대기한다.

Keep Alive(접속 대기)를 이용해 해결이 가능하지만, Keep Alive 때문에 대량 접속시 효율이 떨어짐

NginX

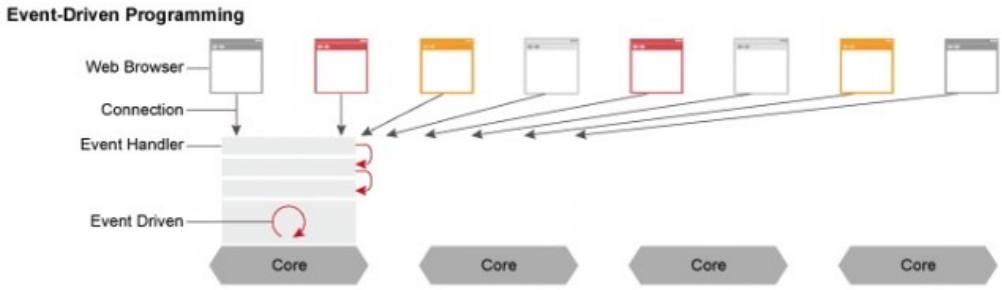
: Apache 의 C10K 문제점 해결을 위해 만들어진 Event-Driven 구조의 웹서버 소프트웨어

: 즉, 프로그램의 흐름이 이벤트에 의해 결정되는 방식

+) C10K : 1만개의 클라이언트 문제 : 한 시스템에 동시 접속자 수가 1만명이 넘어갈 때 효율적 방안

- 주요 특징

1. Event-Driven 처리 기반 구조



- 한 개 또는 고정된 프로세스만 생성하고, 여러 개의 Connection 을 모두 Event-Handler 를 통해 비동기 방식으로 처리한다.

- 적은 양의 스레드만 사용되기 때문에 Context Switching 비용이 적고, CPU 소모가 적다.

- Apache 와 달리 동시 접속자 수가 많아져도 추가적인 생성 비용이 들지 않는다.

- CPU 와 관계없이 모든 I/O들을 전부 Event Listener 로 미루기 때문에 흐름이 끊기지 않고 응답이 빠르게 진행되어 1개의 프로세스로 더 빠른 작업이 가능하다.

이 덕분에 메모리를 적게 사용한다.

+) Context Switching

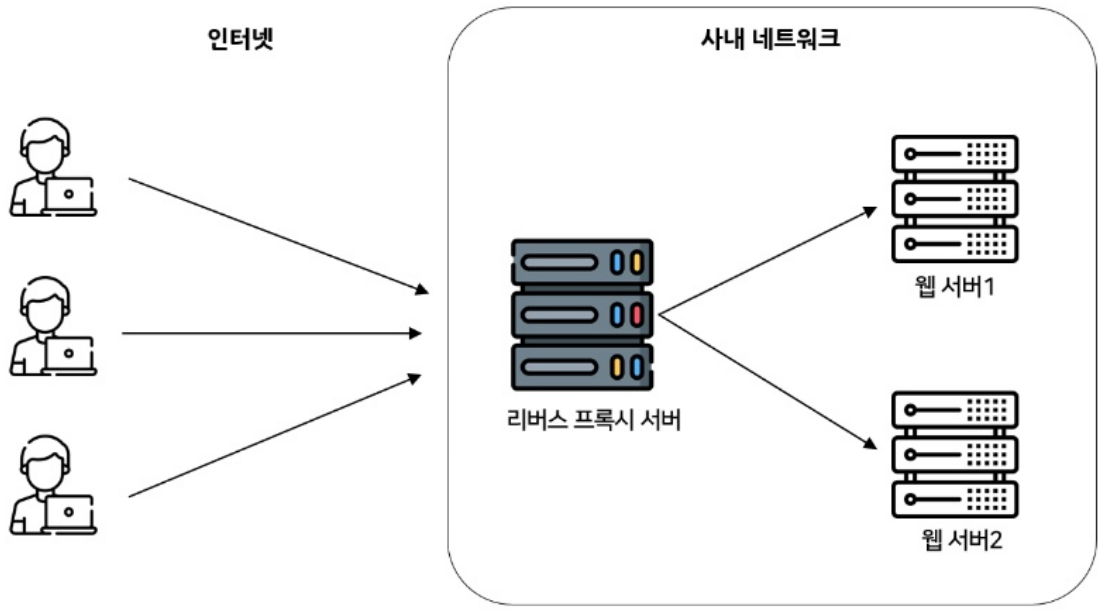
- Context : 스레드가 작업을 진행하는 동안 작업 정보(레지스터, 커널스택, 사용자스택등)를 보관

- OS 가 A 작업을 진행할 때 A 스레드의 Context 를 읽어오며, B 스레드로 전환할 때 A 스레드의 Context 를 저장하고 B 스레드의 Context 를 읽어오는 일련의 반복을 수행

- 스레드의 갯수가 많아질 수록 context switching 작업은 더 빈번하게 일어나고, 이 때문에 성능이 저하될 수 있음

2. reverse proxy 로 배치 가능

- NginX 의 빠른 처리 속도를 활용해 클라이언트의 모든 요청을 처리한다.



우선 리버스 프록시 서버는 로드 밸런싱(load balancing)에 사용됩니다. 유명한 웹 사이트는 하루에도 수백만명이 방문합니다. 그리고 그러한 대량의 트래픽을 하나의 싱글 서버로 감당해 내기란 어렵습니다. 하지만 리버스 프록시 서버를 여러개의 서버 앞에 두면 특정 서버가 과부화 되지 않게 로드밸런싱이 가능합니다.

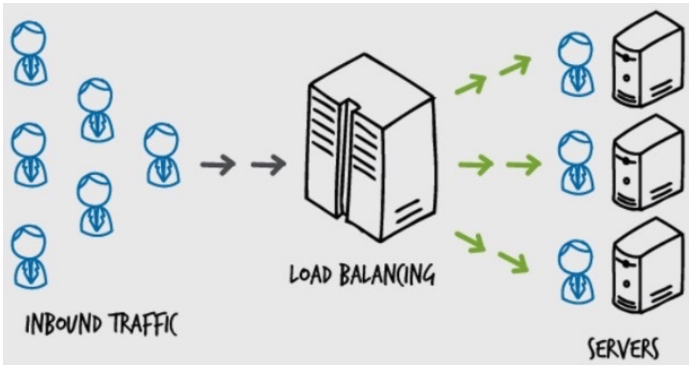
또한 리버스 프록시를 사용하면 보안에 좋습니다. 리버스 프록시를 사용하면 본래 서버의 IP 주소를 노출시킬 필요가 없습니다. 따라서 해커들의 DDoS 공격과 같은 공격을 막는데 유용합니다. 대신 CDN과 같은 리버스 프록시 서버가 공격의 타겟이 될수는 있습니다.

그리고 리버스 프록시 서버에는 성능 향상을 위해 캐시 데이터를 저장할 수 있습니다. 만약 어떤 한국에 있는 유저가 미국에 웹서버를 두고 있는 사이트에 접속할때, 리버스 프록시 서버가 한국에 있다고 해봅시다. 그러면 한국에 있는 유저는 한국에 있는 리버스 프록시 서버와 통신합니다. 따라서 리버스 프록시 서버에 캐싱되어 있는 데이터를 사용할 경우에는 더 빠른 성능을 보여줄수 있는 것입니다.

마지막으로 SSL 암호화에 좋습니다. 본래 서버가 클라이언트들과 통신을 할때 SSL(or TSL)로 암호화, 복호화를 할 경우 비용이 많이 듭니다. 그러나 리버스 프록시를 사용하면 들어오는 요청을 모두 복호화하고 나가는 응답을 암호화해주므로 클라이언트와 안전한 통신을 할수 있으며 본래 서버의 부담을 줄여줍니다.

웹서버 보완 - 특히 C10K 동시 커넥션 문제

+) 로드 밸런싱



로드 밸런싱이란 말 그대로 서버가 처리해야 할 업무 혹은 요청(Load)을 여러 대의 서버로 나누어 (Balancing) 처리하는 것을 의미

한 대의 서버로 부하가 집중되지 않도록 트래픽을 관리해 각각의 서버가 최적의 퍼포먼스를 보일 수 있도록 하는 것이 목적

단점

- 동적 컨텐츠를 기본적으로 처리할 수 없습니다.외부 프로세서로 전달하고 렌더링 된 컨텐츠를 다시 전송할 때 까지 기다려야 합니다. (프로세스 속도 저하)
- Apache에 비해 다양한 모듈이 없습니다.

대표적인 WAS

서블릿 컨테이너

출처 : https://maenco.tistory.com/entry/Web-Server-와-WAS

https://helloworld-88.tistory.com/71

https://velog.io/@han_been/서블릿-컨테이너Servlet-Container-란

https://gmlwjd9405.github.io/2018/10/27/webserver-vs-was.html

https://velog.io/@deannn/Apache와-NginX-비교-차이점

https://techblog.cjenm.com/apache-nginx-8ccad4259622

https://losskatsu.github.io/it-infra/reverse-proxy/#3-리버스-프록시reverse-proxy-서버란

https://tecoble.techcourse.co.kr/post/2021-11-07-load-balancing/

==========

서버라는 말이 다른 의미로도 사용되는데 어떤 컴퓨터로 하여금 서버 역할을 하도록 해주는 소프트웨어를 무슨무슨 서버라고 부르기도 한다. 그래서 백엔드 개발자가 서버를 개발한다고 한다.

이 웹서버도 소프트웨어의 개념이다 - 아파치, NginX 가 대표적인 제품

웹 사이트가 서비스 될 때 필요한 것들이 무엇이 있을까

: 웹사이트는 브라우저에서 돌아가는 것 크롬이나 사파리나 파이어폭스, 엣지 같은 것

그 브라우저가 읽을 수 있는 파일들 HTML, CSS , 자바스크립트 파일들이랑 각종 이미지, 기타 여러 데이터들을 갖다가 서버에서 사용자의 컴퓨터로다가 보내줄 수 있어야 한다.

이 파일들은 원래 서버 컴퓨터에 저장돼 있다. 이 서버의 특정 폴더, 디렉토리에 이것들을 넣어두면 이 폴더를 외부에서 접근 가능하도록 개방해서 서버에 지정된 웹사이트 주소로 접속하면 이것들을 받아갈 수 있도록 하는 것이 웹서버의 기본적인 역할 중 하나이다.

이 아파치, 엔진엑스 이 프로그램들로 서버 컴퓨터에 있는 어떤 폴더를 개방을 해가지고다가 거기 들어있는 HTML 등의 파일들로 웹사이트를 제공할 수 있는 것

서버에 정해진 사이트 주소로 접속을 하면 그 파일들을 꺼내가서 웹사이트를 띄우게끔 - 이것은 정적 웹이다.

동적웹을 제공하는 것도 웹서버의 고유 역할로 정의해야할 지는 애매하지만 아파치나 엔진엑스의 모듈로 할 수 있다.

오랫동안 사용되어온 방법으로 아파치랑 PHP, MySQL 을 연동시켜서 동적인 PHP 웹사이트를 제공하는 방식이 있다. - 아파치에 PHP 를 해석할 수 있는 모듈을 세팅해놓으면 웹접근이 있을 때마다 PHP 코드에 적힌 레시피대로 MySQL 에 있는 정보들을 가져와서 아파치가 만들어줌 엔진엑스도 가능

요즘은 스프링부트에 톰캣이 내장되기 때문에 직접은 많이 안접하지만 자바랑 JSP 로 만든 웹 또는 API 어플리케이션을 실행할 때 이 톰캣이 사용된다.

WAS - 동적사이트를 전문적으로 처리해주는 것

reverse proxy

아파치는 다중 프로세스, 엔진 엑스는 이벤트로 일을 처리함

아파치 - MPM, 멀티 프로세스 모듈 방식으로 일함

손님이 올때마다 프로세스를 새로 생성하는 방식이 있고 한 프로세스 안에서 스레드를 새로 생성하는 방식이 있다.

mpm prefork, 즉 손님마다 프로세스를 두는 방식은 손님이 새로 올 때마다 상담 테이블을 따로 가져와서 앉히는 것 그리고 이 테이블을 계속 돌아다니면서 여러 손님들을 동시에 상대

mpm worker, 한 프로세스에서 손님마다 스레드를 생성한다는 건 가로로 길다란 테이블을 하나 두고, 손님이 올 때마다 나란히 앉힌 다음 역시 좌우로 돌아다니면서 동시에 업무를 처리하는 것

반면 NginX 의 event driven 방식을 비유하자면 작은 데스크 하나만 두고 손님들을 한 줄 로 쭉 세우는 것

그리고 다음 손님이 오는 대로 업무별로 집중해서 일을 처리하는 것

https://www.youtube.com/watch?v=Zimhvf2B7Es