

마이크로 서비스에 적합한 오픈소스 WAS는 무엇?

- Tomcat vs. Jetty vs. Undertow 비교
- 차세대 웹 시스템을 위한 JBoss EAP7(Undertow) 소개

웹시스템의 변화

- Complexity
 - JavaScript execution
 - Content stored on CDNs
 - Calls to external APIs

WEB PAGES ARE BIGGER AND MORE COMPLEX THAN EVER.



1995

2010

2012

평균 페이지 크기

14.1k

498k

684k

페이지 오브젝트 수

2.3

75

83

Source : strangeloop

- <http://tumblr.thefjp.org/post/7121875059/pages-are-getting-heavy>

Application Performance Management

현재 웹시스템

HTTP/1.1

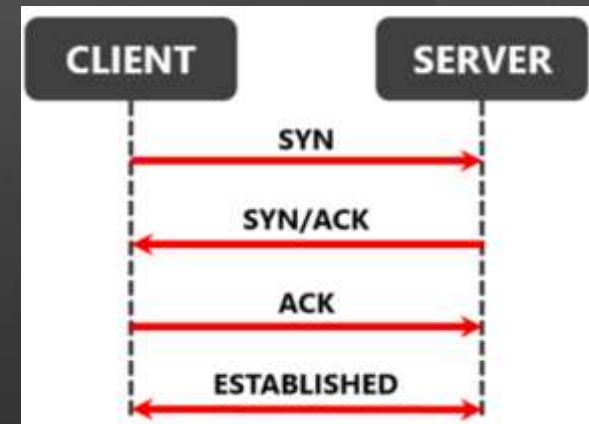
- 단점
 - Connection 당 하나의 요청만 처리
 - 3-way handshake Overhead
 - Header (Cookie) Overhead
 - HOL (Head Of Line) Blocking



▼ Request Headers [view source](#)

```

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cache-Control: no-cache
Connection: keep-alive
Cookie: WMONID=VNF0vU_XjUn; PCID=15144277212811124686447; _g
_IP_LOC%7CDOM; XSITE=1000966141; PARTNER_CD=1561; RAKE_SID_
r=0.2.11; plab.uid=59c57575-0308-4e1a-825e-a3656da5b9ce; pl
N%3A-1%3A0%3AN%3A-1; TP=scrnChk%7CY%23TB_DATA_CHK%7CN%3AY;
SID=15184482710816861860435; JSESSIONID=R5eKkedqW9dDI6JgAlM
17720; dc gtm UA-68494772-1=1
  
```



TCP connection establishment (3-way handshake)

TCP는 장치들 사이에 논리적인 접속을 성립(establish)하기 위하여 3-way handshake를 사용

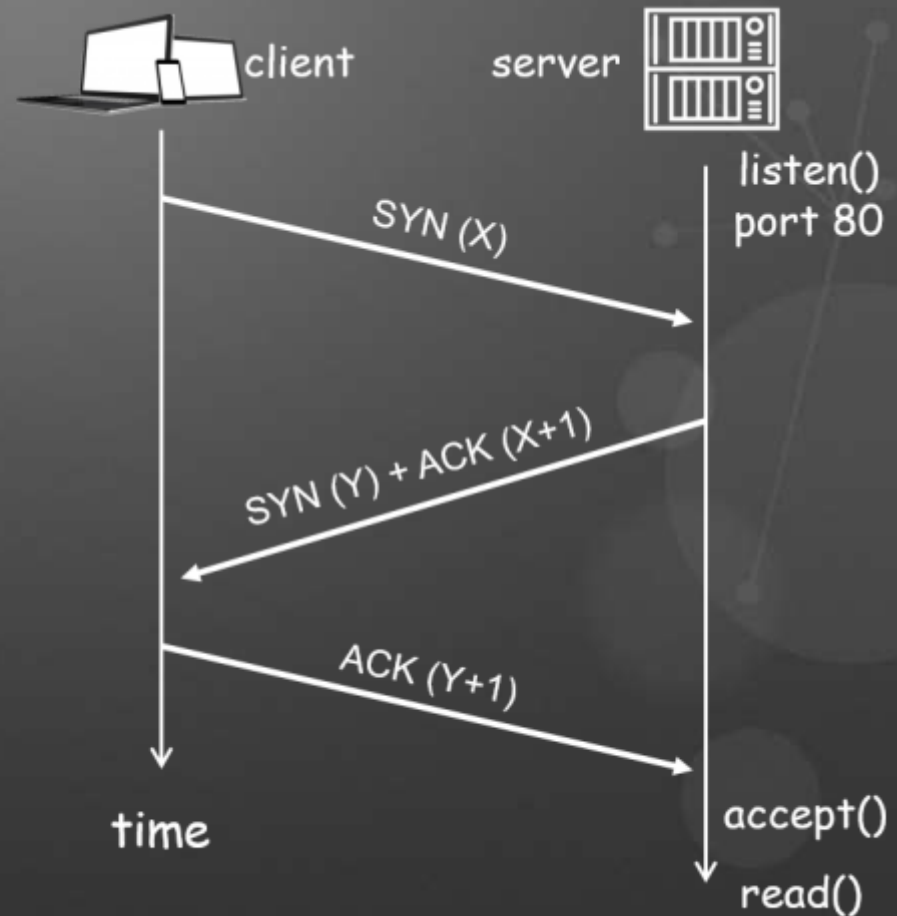
- 양쪽 모두 데이터를 전송할 준비가 되었다는 것을 보장하고, 실제로 데이터 전달이 시작하기전에 한쪽이 다른 쪽이 준비되었다는 것을 알 수 있음

매번 TCP Connection 확립

- 매번 3-way Handshake

Slow Start

- 매번 연결 시 정해진 'initial congestion window size(CWND)' 부터 어느 정도 선까지 지속적으로 증가



Performance



<https://www.percona.com/solutions/optimize>

HTTP/1.1

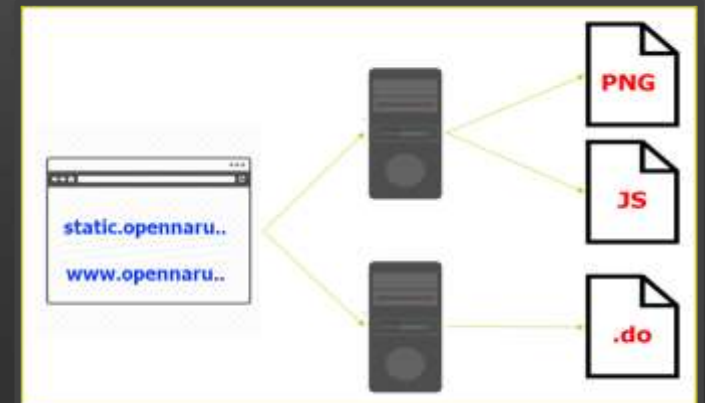
- 성능을 위한 노력들
 - Image Spriting
 - Minify css/js
 - Data URI Scheme
 - Domain Sharding
 - Load Faster (css top, js bottom)
 - CDN Cache
- 기타
 - Multiple Connection (Browser)
 - Keep Alive

```

```



```
/*! KHAN [arm] - v5.1.0 (build#1.0) */$(document).rea  
function b(a){var b=void 0;return void 0!=="angular.el  
"PageViewController")) .scope().localize(a)),void 0==  
,"success":function(a,b,c){a.status>200&&a.status<30  
"KhanJS error #" +a.code+": "+a.msg;if(b)throw c;loge(  
function c(){c.prototype=a.prototype,b.prototype=new  
&&(a={});for(d in b)b.hasOwnProperty(d)&&(c[b]=b[c]&&  
nodeType?a[d]=a[d]||{}),c[a[d]=b[d]];return a};for(  
{return"Object Array"}===Object.prototype.toString.  
{return V.timezoneOffset}function l(a){var b=new Re  
{var b=String(a);return l==b.length&&(b="0"+b),b)va  
abs(b))+":00",d.getUTCFullYear()+"-"+c(d.getUTCMonth(  
Number(d.getUTCMilliseconds())/1e3).toFixed(3)).slice  
void 0==b&&(b=V.timezoneOffset),d.setHours(d.getHour
```



Application Performance Management

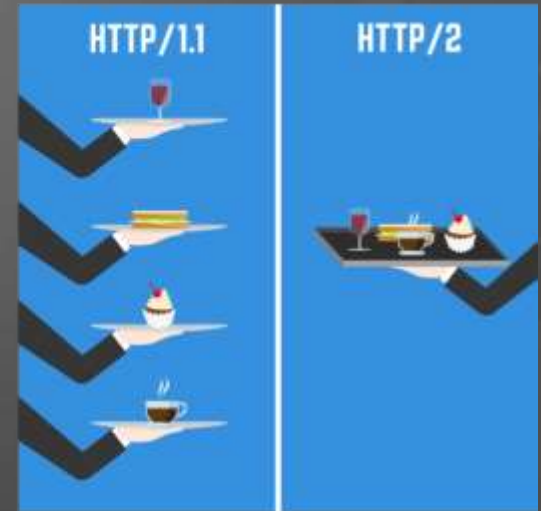
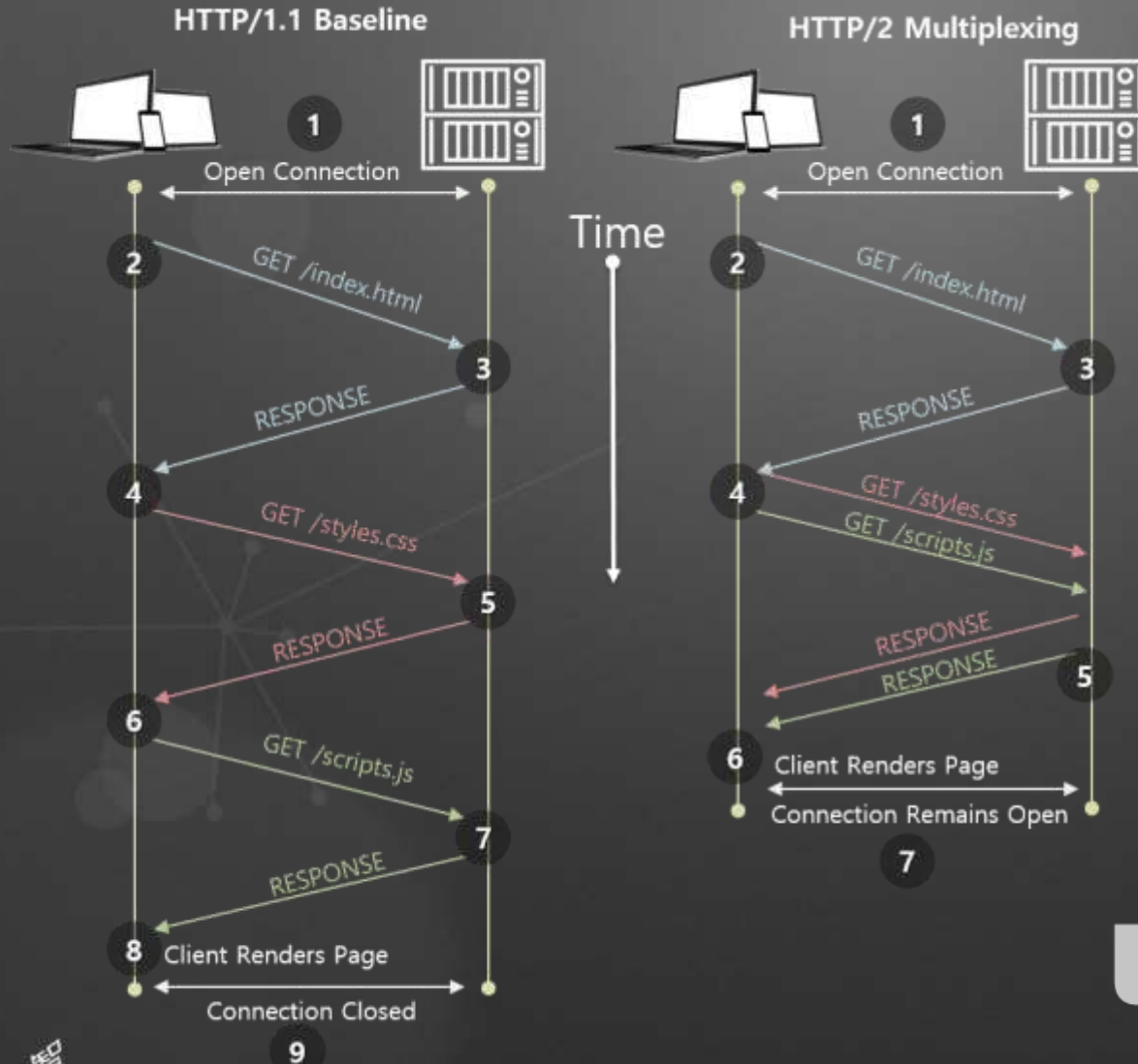
HTTP/2

HTTP/2

- 장점
 - Multiplexed Streams
 - Stream Priorization
 - Server Push
본문 전송 시 서버가 알아서 함께 전송 (js, css 등)
 - Header Compression (HPACK)



HTTP/1 vs. HTTP/2 비교



<http://ruzhekov.com/switching-to-http2-considerations-optimization-future/>



HTTP/1.1 vs HTTP/2

HTTP

1.83s



1153288396.rsc.cdn77.org의 응답을 기다리는 중...

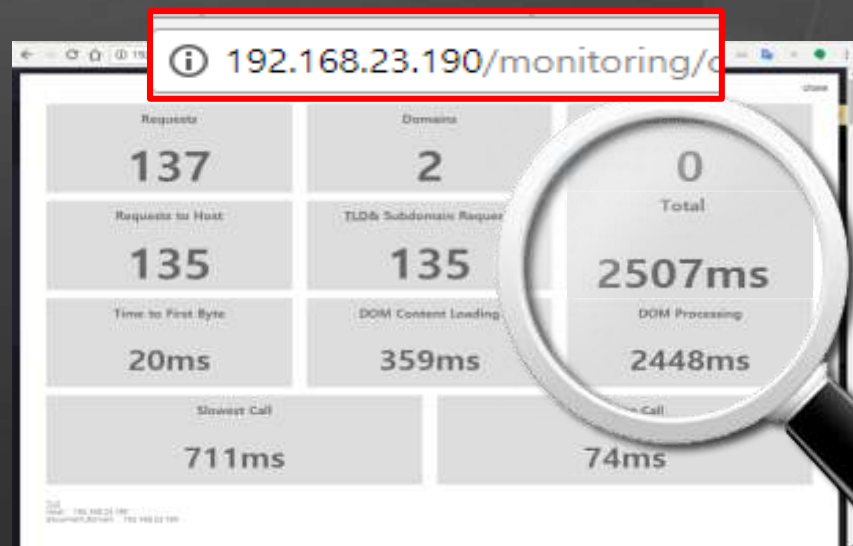
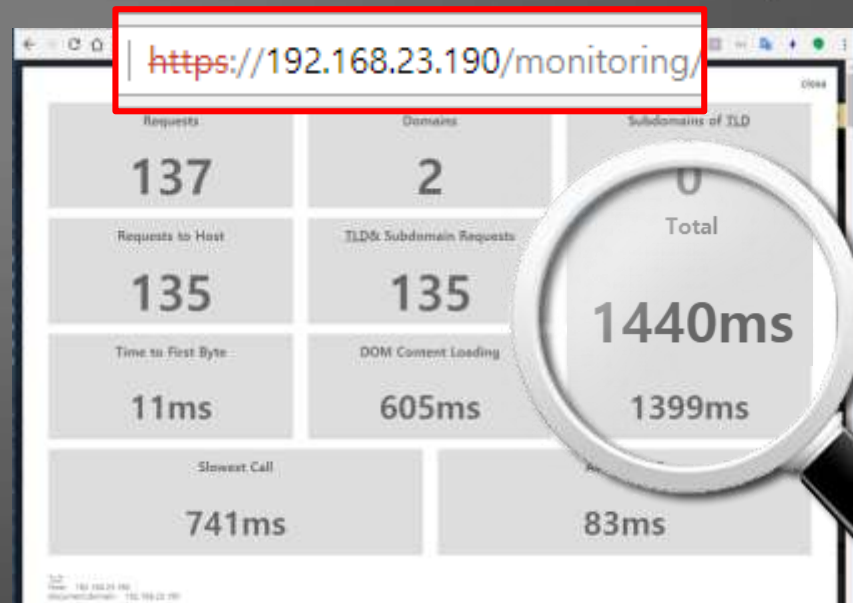
HTTPS (SSL)

0.91s



1153288396.rsc.cdn77.org/http2/http1.html

HTTP/1.1 vs HTTP/2

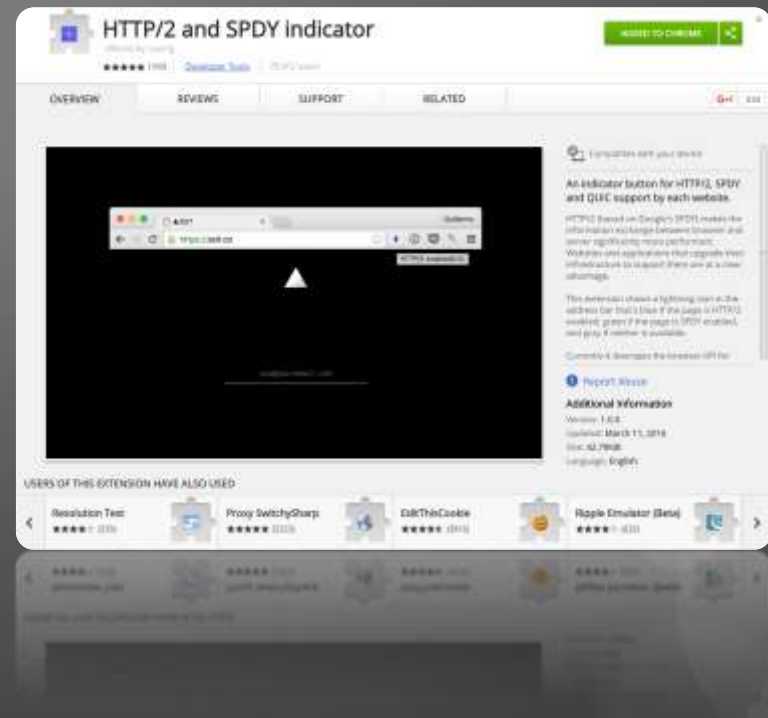


HTTP/2 제약 사항

- ALPN (Application Layer Protocol Negotiation)
TLS (Transport Layer Security) 기반의 확장
 - JAVA EE 7 (JDK 8)에서는 ALPN을 사용 불가
별도의 ALPN 구현체 설치/설정
 - JDK 9에서 포함
- OpenSSL 업그레이드
- Undertow (JBoss EAP 7.1)
 - 별도의 설치/설정 과정 없이 JDK 8에서 가능

어떻게 HTTP/2를 사용하는 사이트인지 알 수 있을까?

- Chrome Browser plugins
 - HTTP/2 and SPDY indicator
- HTTP Request Headers



▼ Request Headers view source


Accept: */*

Accept-Encoding: gzip, deflate

Accept-Language: ko-KR,ko;q=0.9,en-US

Cache-Control: no-cache

Connection: keep-alive



▼ Request Headers

:authority: khanapm.com

:method: GET

:path: /

:scheme: https

Source: <https://chrome.google.com/webstore/detail/http2-and-spydy-indicator/mpbpobfflnpcgagijhmgncggcjblin>

Application Performance Management



WebSocket

- 웹시스템의 발전에 따른 요구사항
 - 다양한 SNS 서비스의 등장으로 실시간 메세징이나 푸시 기술 요구
 - 여러 사용자가 함께 문서를 편집하고 참조할 수 있는 글로벌 협업 도구에 대한 요구
 - 웹이나 모바일 환경에서 옥션 서비스나 스포츠 이벤트, 금융정보에 대한 실시간 제공 등 실시간 정보 제공에 대한 요구 증가
- 실시간 서버 푸시나 커뮤니케이션을 위한 기술 요구
 - 기존 HTTP를 이용한 클라이언트와 서버 간의 양방향 통신에 대한 요구 증가
 - 기존 HTTP를 사용하여 클라이언트가 일정 간격으로 서버에 요청하는 폴링 방식으로는 새로운 요구를 수용하기 어려움
 - Comet 이나 Server Sent Event 방식이 있었으나 기술적인 한계와 개방성에 대한 이슈 때문에 많이 확산되지 않음

기존 HTTP의 문제점

- 기존 방식이고, HTTP 통신에 의지 할 수 밖에 없다
 - 동시 연결 수가 많으면 메모리 등의 자원을 많이 소모한다.
(1 요청 / 응답 헤더 자체는 수 KB)
- HTTP 는 비상태 프로토콜
 - 별도로 세션 관리를 해야 함
- 매번 요청되는 Header 크기

GET /resources/public-data/ HTTP/1.1

Host: bar.other

User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US; rv:1.9.1b3pre) Gecko/20081130 Minefield/3.1b3pre

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Connection: keep-alive

Referer: http://foo.example/examples/access-control/simpleXSIInvocation.html

Origin: http://foo.example

400 Bytes over!

왜 고속 통신이 가능한가

송신 데이터가 "**Hello, world**"의 경우

- **HTTP**

- 12 bytes + 400 bytes

→ **412 Bytes**

- **97.1 %** 이 Header

- **WebSocket**

- 12 bytes + 6 bytes

→ **18 Bytes**

- **33.3 %** 이 Header

같은 문자열을 보낼 때 **약 23 배의 감소**

WebSocket Header

Request

```
GET / demo HTTP / 1.1 Host
: example.com Connection :
Upgrade
Sec-WebSocket-Key2 : 12998 5 Y3 1 .P00
Sec-WebSocket-Protocol : sample Upgrade :
WebSocket
Sec-WebSocket-Key1 : 4 @ 1 46546xw % 0l 1 5 Origin :
http://example.com ^ n : ds [4U
```

Response

```
HTTP / 1.1 101 WebSocket Protocol Handshake
Upgrade :
WebSocket Connection : Upgrade
Sec-WebSocket-Origin : http://example.
com
Sec-WebSocket-Location : ws : //example.com/
demo
Sec-WebSocket-Protocol : sample 8jKS'y : G *
Co, Wxa-
```

Websocket

- 브라우저 지원 문제
- 웹서버의 신뢰 문제
- 만능의 대체 프레임워크 등장
 - Node.js & Socket.io
 - Vertx & SockJS



WebSocket

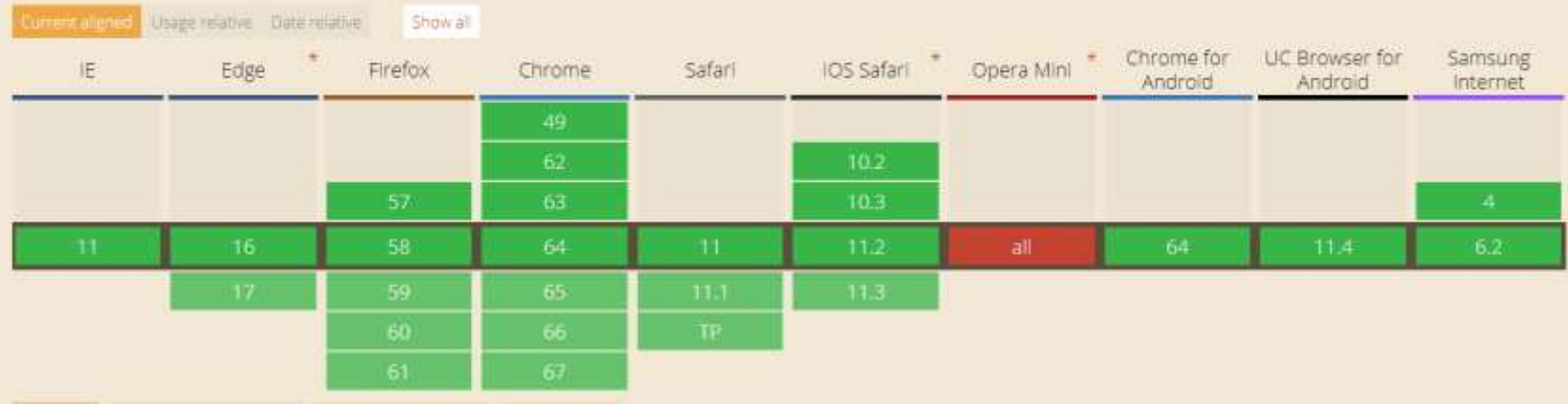
Web Sockets - LS

Bidirectional communication technology for web apps

Usage: % of all users

Global 94.17% + 0.3% = 94.48%

unprefixed: 94.17% + 0.26% = 94.44%



Web Sockets - LS

Bidirectional communication technology for web apps

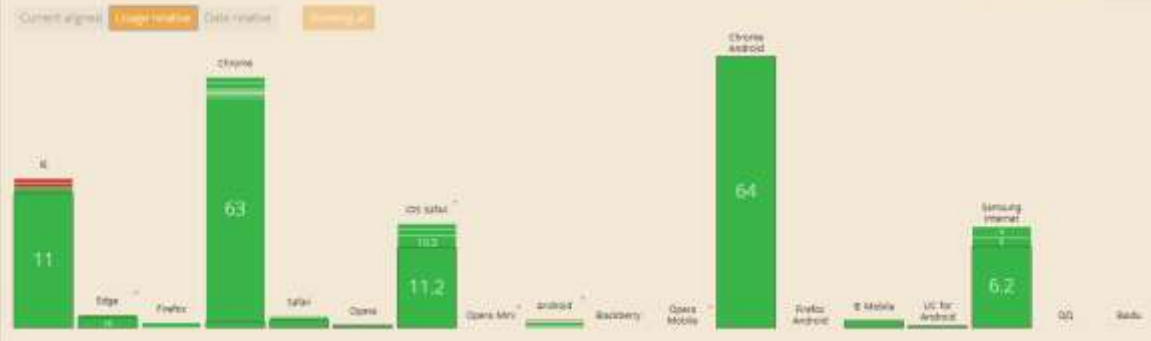
Usage: % of all users

Korea, Republic of 98% + 0.1% = 98.09%

unprefixed: 98% + 0.1% = 98.09%

Global 94.17% + 0.3% = 94.48%

unprefixed: 94.17% + 0.26% = 94.44%



WebSocket Use Cases

Collaboration



- Collaborative Apps
- Multiplayer Games
- Multimedia Chat
- Social Feeds

Real-Time



- Financial Tickets
- Clickstream Data
- Sports Updates

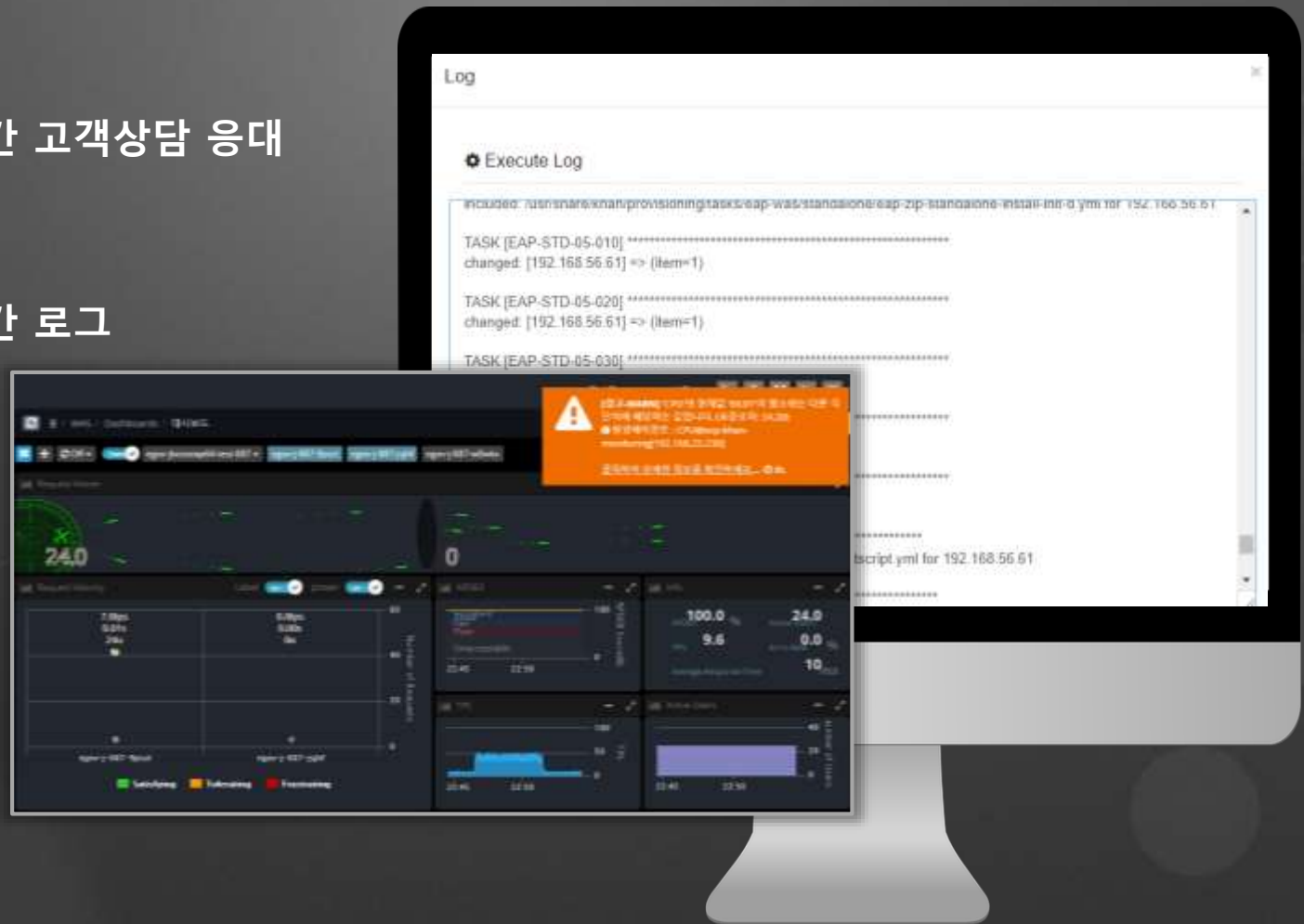
Business Event



- Online Education
- Location-based Apps
- Online Auction

WebSocket

- 활용
 - 실시간 고객상담 응대
 - 알람
 - 실시간 로그





Undertow

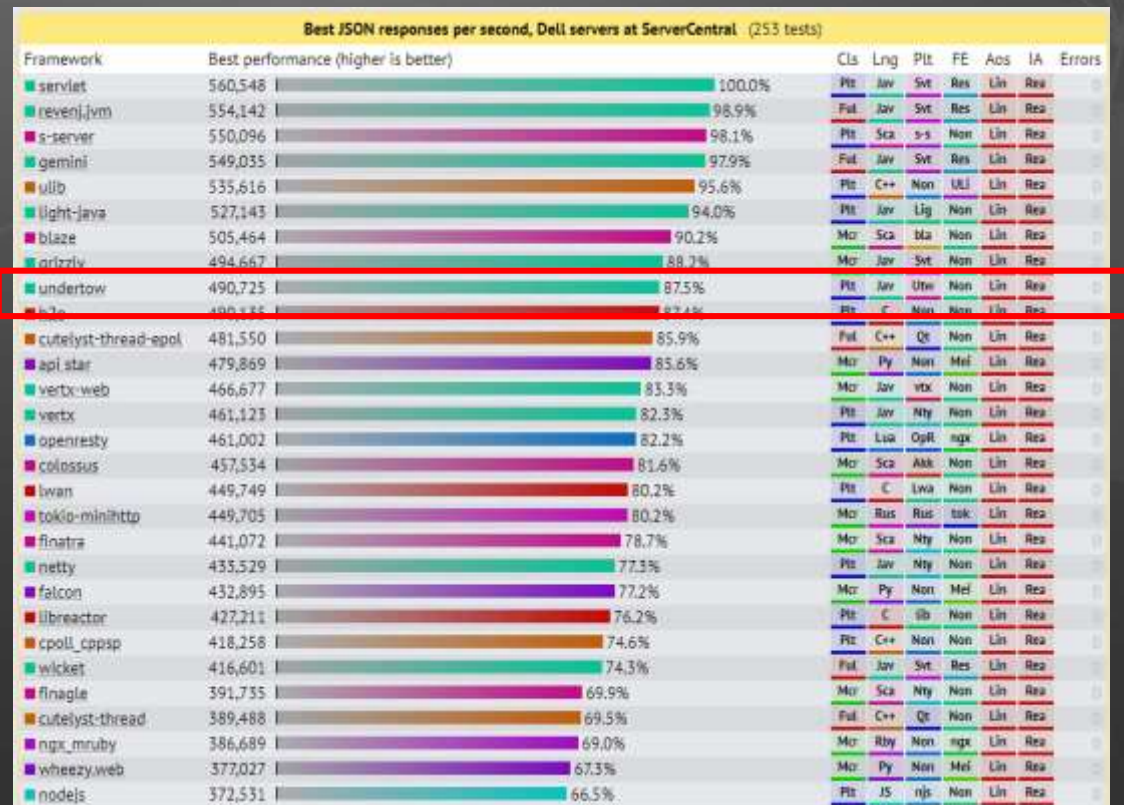
Undertow

- **Lightweight**
1Mb 이하, 4Mb 미만의 Heap
- **Embeddable**
- **HTTP Upgrade**
- **WebSocket**
- **Servlet 3.1**
- **Flexible
Handlers**



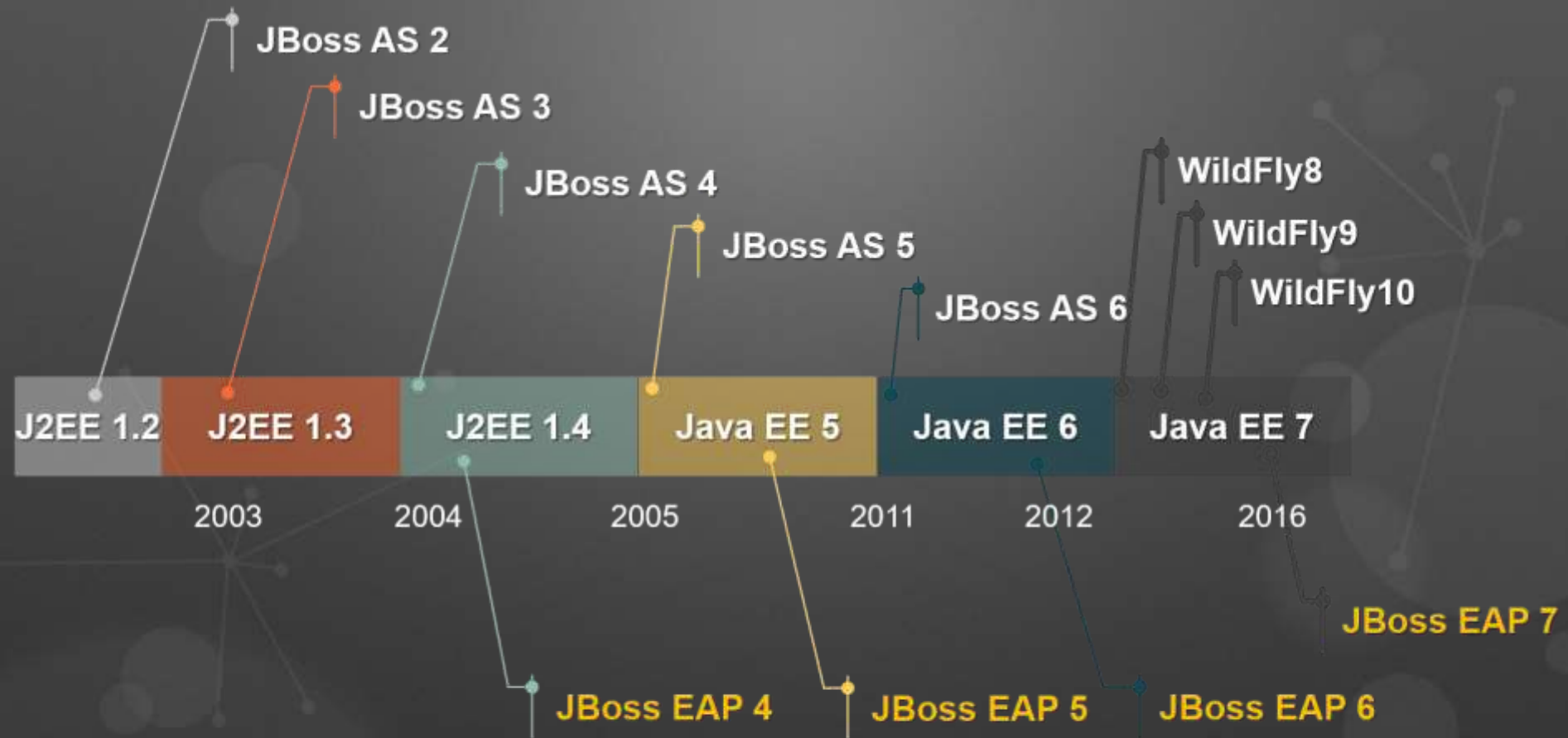
Undertow

- JAVA 언어로 작성된 웹서버
 - JBoss 가 후원하고, JBoss EAP 7, Wildfly 가 사용하는 기본 웹서버



<https://www.techempower.com/benchmarks/#section=data-r14&hw=ph&test=json>

Community & JBoss EAP with Java EE Spec









WildFly 8/9/10



RED HAT® JBOSS®
ENTERPRISE
APPLICATION
PLATFORM **7**

JBoss 웹컨테이너 진화

웹 컨테이너	Red Hat Commercial	Red Hat Community
		
		

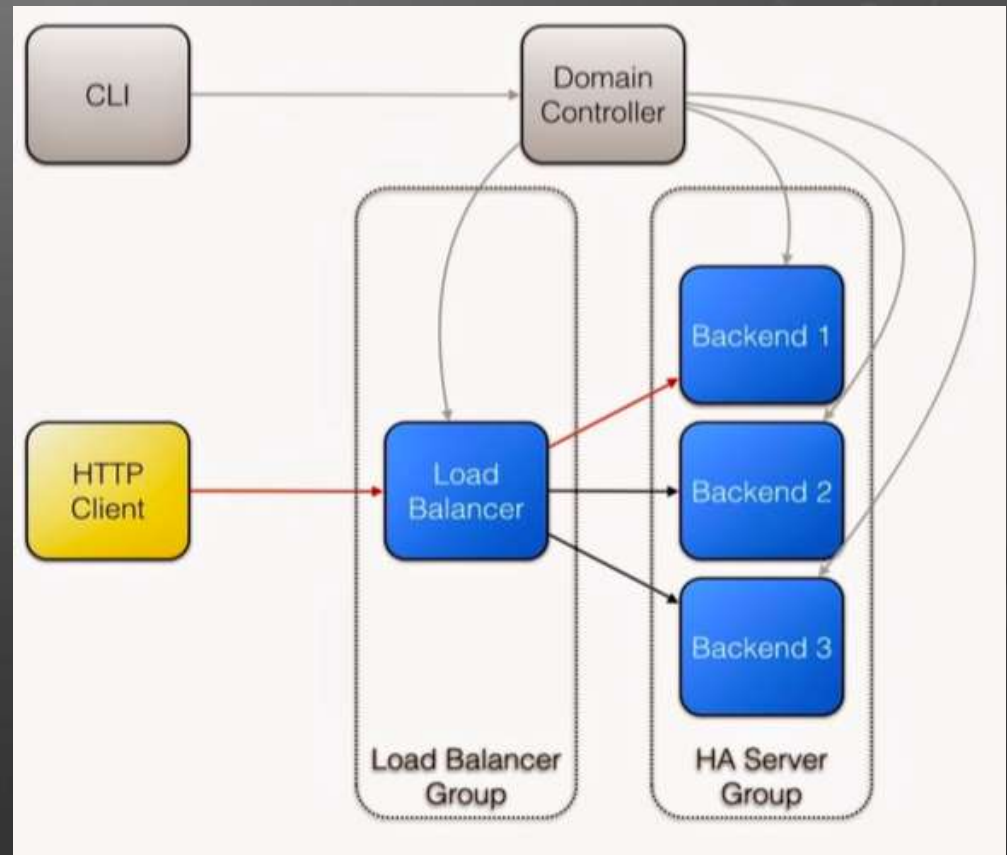
- JBoss Web (AS7 이전)
 - <http://jbossweb.jboss.org>
 - Tomcat 기반
 - 성능과 고가용성 확보
 - 관리도구 확장



- Undertow (WildFly에서)
 - <http://undertow.io>

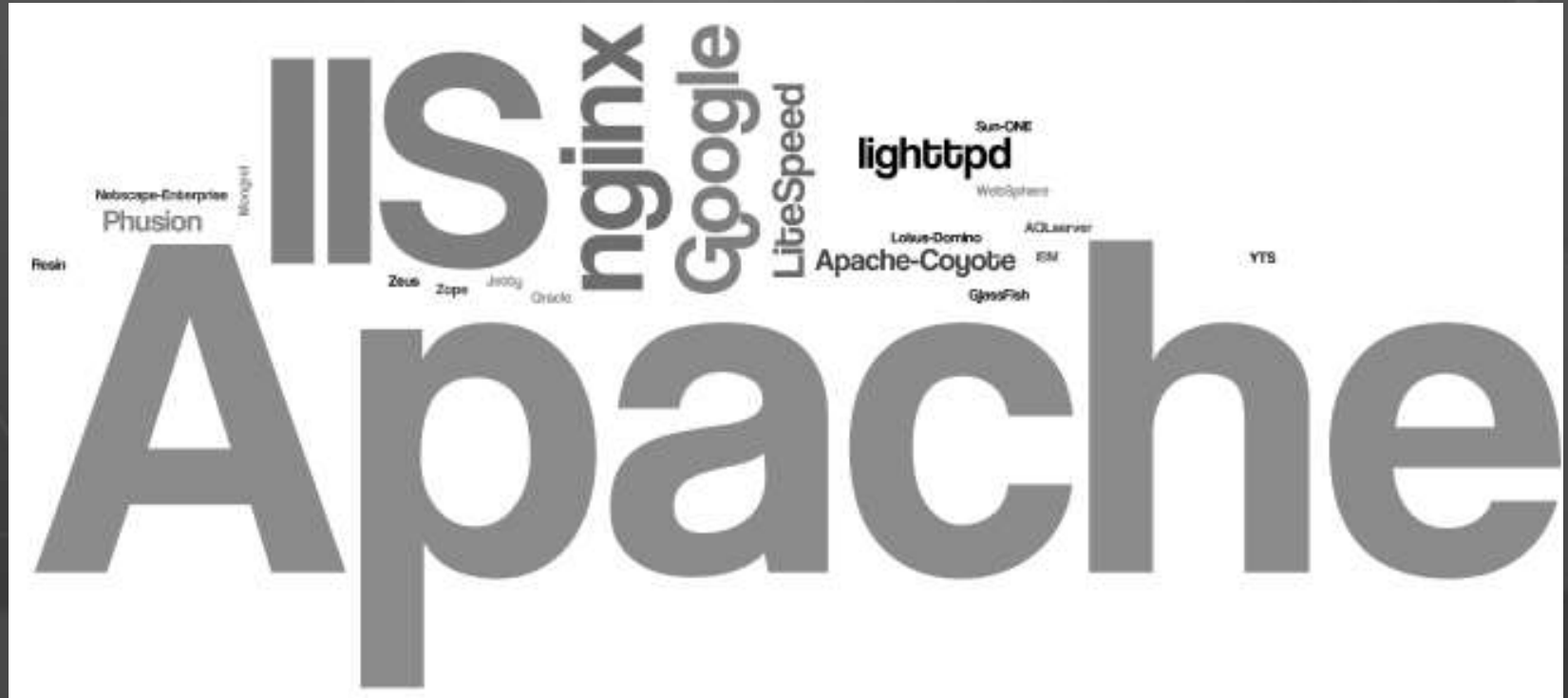
Load Balancer

- 기존의 Apache 와 같은 네이티브 웹서버를 사용하지 않고,
로드 밸런스 설정 가능
 - Java 로 구현된 로드 밸런스
 - mod_cluster 구현
 - HTTP, AJP, HTTP/2
 - Websocket



Load Balancer

- 대체 가능 한가



Load Balancer

- IP 주소 한개에 여러 도메인으로 웹사이트 운영하기
 - Virtual Host

```
NameVirtualHostHandler vhostHandler =  
Handlers.virtualHost();  
ProxyHandler wikiProxyHandler =  
ProxyHandler.builder().setProxyClient(  
    new LoadBalancingProxyClient().addHost(  
        new URI("http://localhost:8280"))) .build();  
vhostHandler.addHost("wiki.opennaru.com",  
wikiProxyHandler);
```

```
<host name="other-host" alias="www.opennaru.com">
```

Load Balancer

- 원격 서버에서 문서를 가져와 클라이언트에 전달하는 Proxy
 - Reverse Proxy

```
LoadBalancingProxyClient  
loadBalancer = new  
LoadBalancingProxyClient()  
                .addHost(new  
URI("http://localhost:8180  
"));
```

```
<location name="/"  
handler="myproxy"/>  
<reverse-proxy  
name="myproxy">  
    <host  
name="http://localhost:8180"  
path="/" />  
</reverse-proxy>
```

Load Balancer

- URL 을 조작 할 수 있는 강력 한 모듈
정규식, 표현식을 통해 강력하게 이용
 - Rewrite
 - Redirect

```
PredicatesHandler predicatesHandler = Handlers.predicates(  
    PredicatedHandlersParser.parse("" +  
    "path-prefix('/ddakker') -> {rewrite ['/opennaru'];} \n" +  
    "regex('/test(.*)') -> {rewrite ['/abc/test/${1}'];} \n" +  
    "regex('/redirect1$') -> redirect ['/redirect1/ed'] \n" +
```

```
<filter-ref name="rewrite-test" predicate="regex('^/test(.*)$')"/>  
<rewrite name="rewrite-test" target="/abc/test${1}.jsp"  
redirect="false"/>
```

Load Balancer

- 클라이언트의 IP 주소에 따라 접근 제어
 - IP Address Access Controller
- WEB-INF/undertow-handlers.conf

```
IPAddressAccessControlHandler handler = new  
IPAddressAccessControlHandler()  
    .setDefaultAllow(false)  
    .addAllow("localhost") .addAllow("127.0.0.1"));
```

```
<expression-filter name="ipAccess" expression="path-prefix[/] ->  
ip-access-control[default-allow=false, acl={' localhost allow',  
'127.0.0.1 allow'}]"/>
```


Load Balancer

- WAS Clustering 을 위한 cluster
 - mod_cluster
 - 동적 구성
 - 서버 측 부하 계산
 - 세밀한 생명주기 제어
 - http, https, ajp 지원

```
final ModCluster modCluster  
=  
ModCluster.builder(worker).b  
uild();  
modCluster.start();  
server.start();
```

```
<filter-ref name="b_eap71" /  
  <mod-cluster name="b_eap71" advertise-socket-  
binding="modcluster" management-socket-binding="http"  
enable-http2="true" security-key="app" />
```

JBoss EAP (Wildfly)

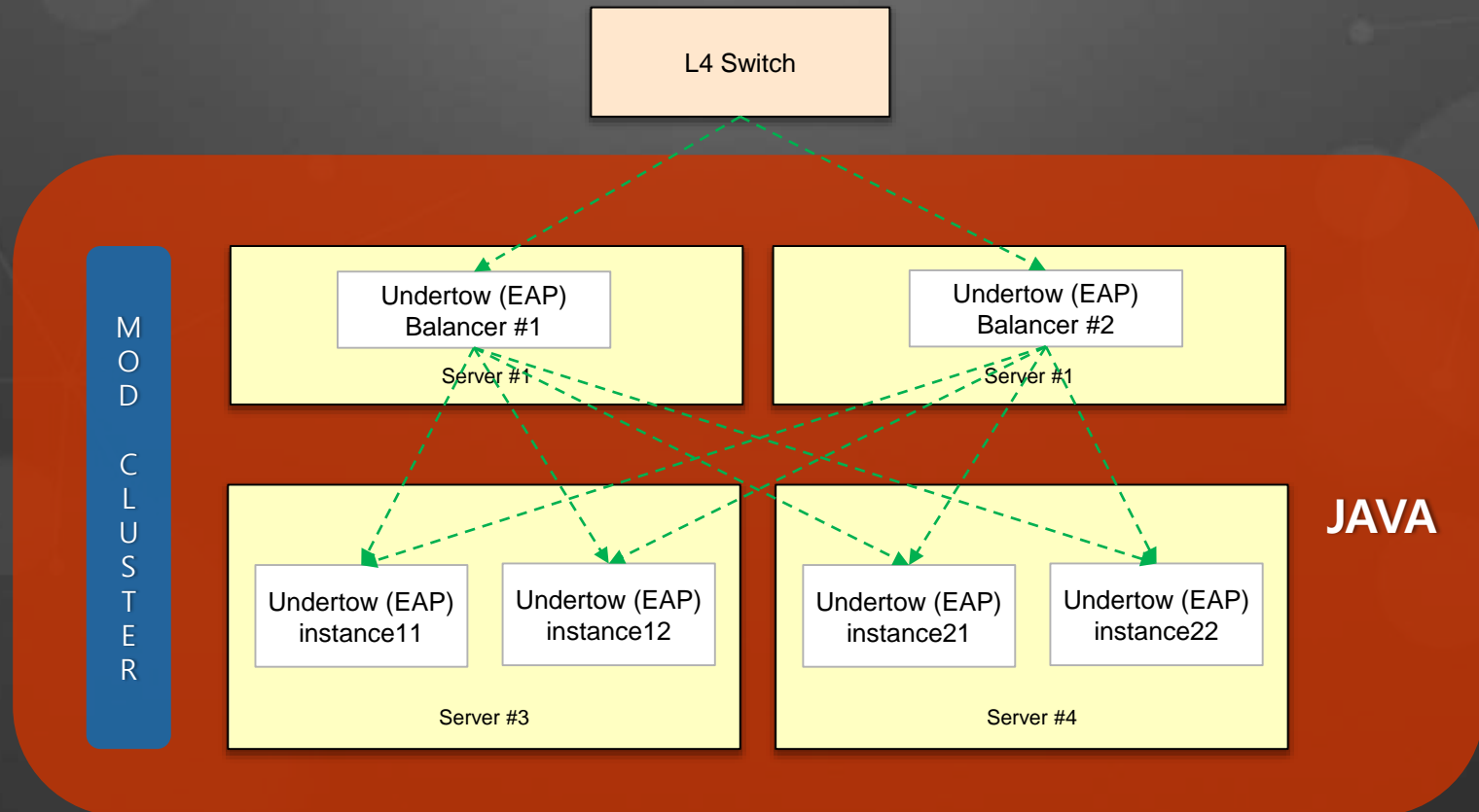
- Undertow Filter

- custom-filter
- error-page
- expression-filter
- gzip
- mod-cluster
- request-limit
- response-header
- rewrite

```
<host name="default-host" alias="localhost">  
  <filter-ref name="rewrite-test" predicate="regex('/test1') "/>  
  ..  
</filters>  
  <rewrite name="rewrite-test" target="/abc/test1"  
  redirect="false"/>
```

WAS 역할의 변화

- Only with JAVA
- Static Content
- Native Module 필요 없음
- Load Balancer
- 세밀한 제어
(Suspend, Resume, Gracefull or mod_cluster mgr)



“살아 남는 종(種)은 강한 종이 아니고,
또 우수한 종도 아니다.
변화에 적응하는 종이다.”

- Charles Darwin, 1809

Application Performance Management

감사합니다.





제품이나 서비스에 관한 문의

콜 센터 : 02-469-5426 (휴대폰 : 010-2243-3394)

전자메일 : sales@openmaru.com

2019년도 신입 · 경력 채용 공고

당신과 **함께** 오픈소스 정상에서
같이 하고 싶습니다.

- 홈페이지 www.opennaru.com/recruit
- 이력서 접수 apply@opennaru.com
- 채용문의 02-469-5426

