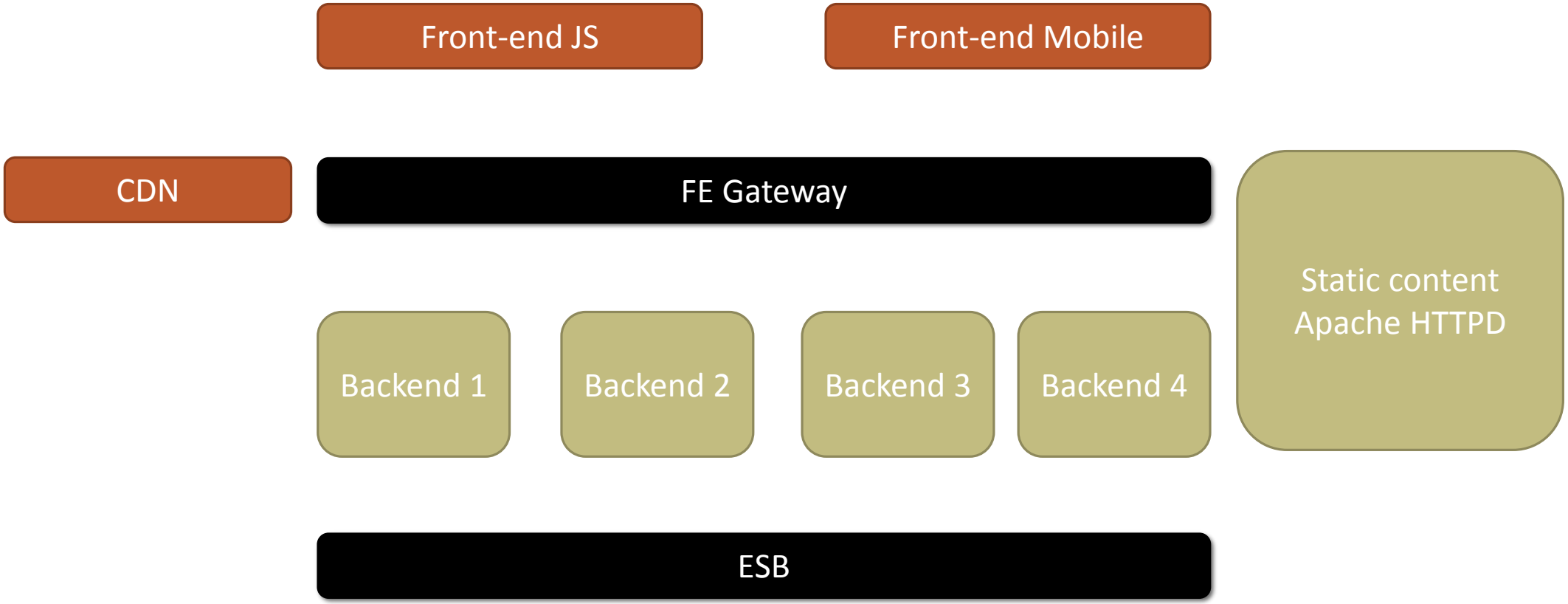
NSS - Cache

MARTIN TOMASEK

Standardní „moderní“ aplikace



Front-end – využíván klientem

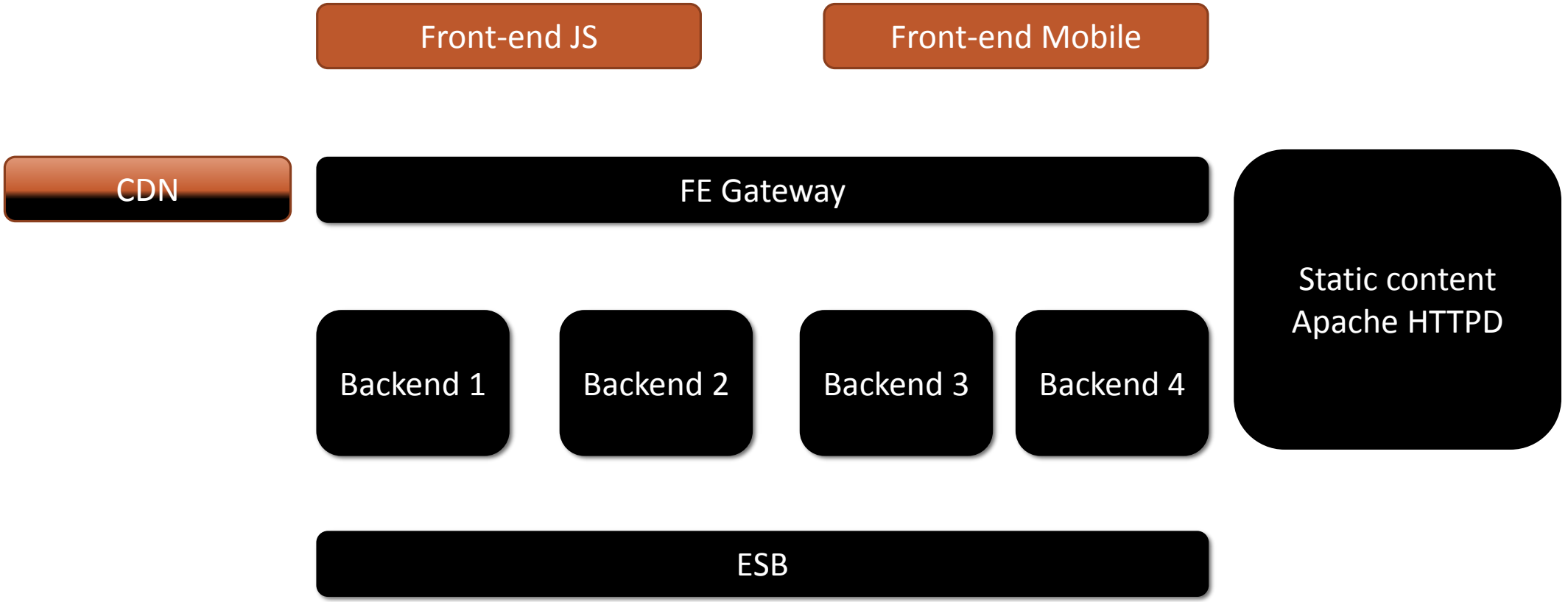
Gateways, ESB – integrace mezi systémy

BE – Data, Obrázky, Logika

Cache mechanismus

1. Lze využít k:
    1. Optimalizaci výkonu systému
    2. Snížení náročností jednotlivých operací
    3. Snížení náročností na jednotlivé vrstvy
    4. Mitigaci rizika krátkodobé nedostupnosti systému
2. Typy cache
    1. Nodové
    2. Clusterové
        1. Distribuované
        2. Replikované
3. Příklady:
    1. Hazelcast
    2. Infinispan
    3. JBoss Tree Cache

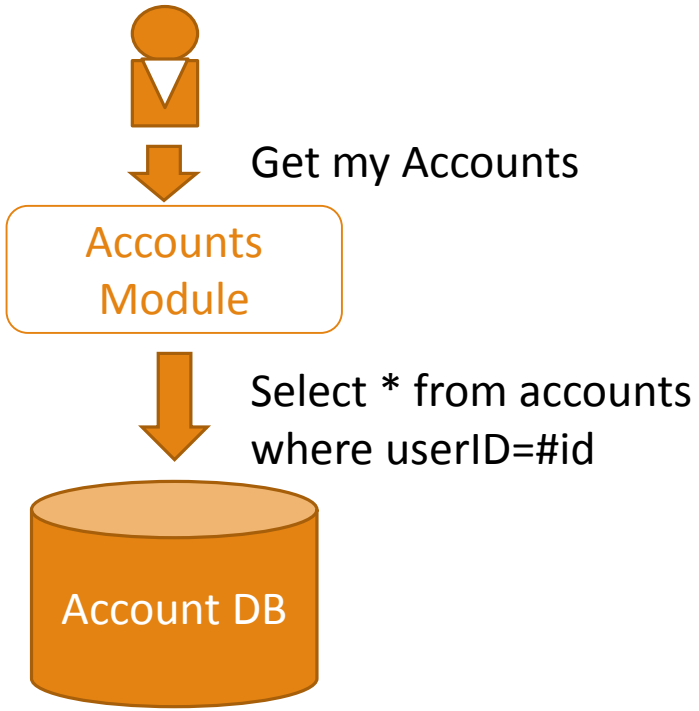
Standardní „moderní“ aplikace



Ostatní komponenty

Systémy, kde lze aplikovat Cache

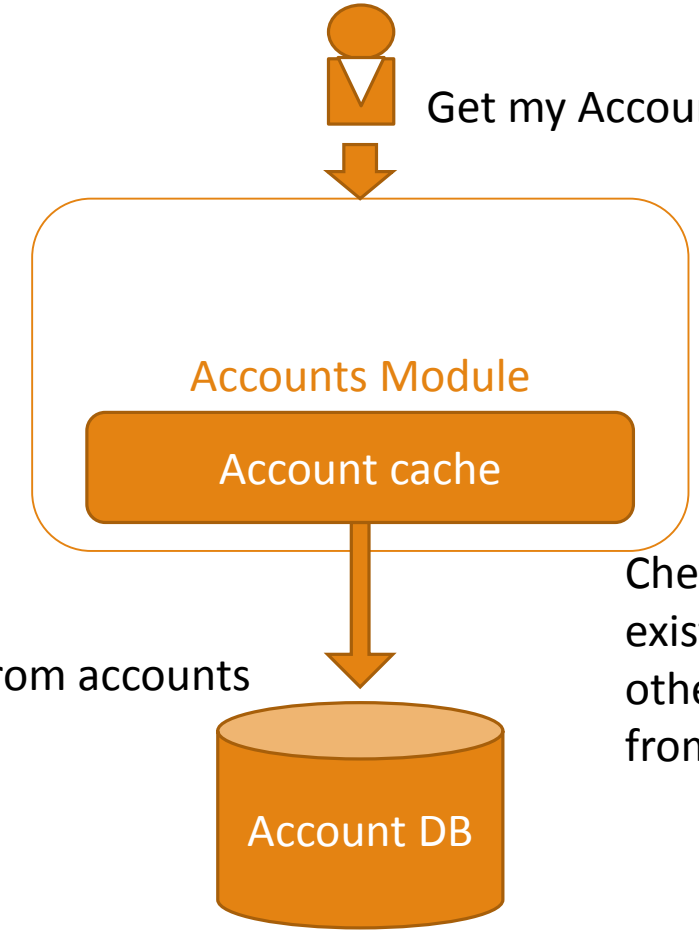
Ukázka využití cache



Počet uživatelů 1 000 000
Response time modulu (bez DB): 100 ms
Response time DB: 20ms

Počet volání Modulu	Počet volání DB	Celkový počet volání	Celkový čas (hodiny)
1 000 000	1 000 000	2 000 000	33

Ukázka využití cache



Check data in cache, if exists return it, otherwise také all data from DB

Počet uživatelů 1 000 000
Response time modulu (bez DB): 100 ms
Response time DB: 20ms
Response time cache: 3ms

Počet volání Modulu	Počet volání DB	Počet volání cache	Celkový počet volání	Celkový čas (hodiny)
1 000 000	1	1 000 000	2 000 001	28,6

Cache a co do nich

Co můžeme mít v cache?:

- Jakékoliv objekty
- Výsledky SQL dotazů (MyBatis Cache per select)

Co raději ne?:

- Velké obrázky
- Videa

Obsahově:

- Statická data
- Data, která se minimálně mění
- Data, která je výpočetně těžké získat
- Data, ke kterým přistupujeme opakovaně

Typy úložišť pro cache

In-memory

- Nejčastější použití data se načtou do paměti

FileSystem

- Zejména pro velké soubory, statické obrázky a pod
- Data jsou na File Systému mimo aplikační server.
- Zátěž se přesouvá na úložiště.
- Ideálně v kombinaci s CDN

Databáze

- Data mohou být uložená v DB
- V případě, kdy potřebujeme replikovat data z jiného systému
- Popřípadě jsou zde již transformovaná, přepočítaná, agregovaná data

Základní operace cache

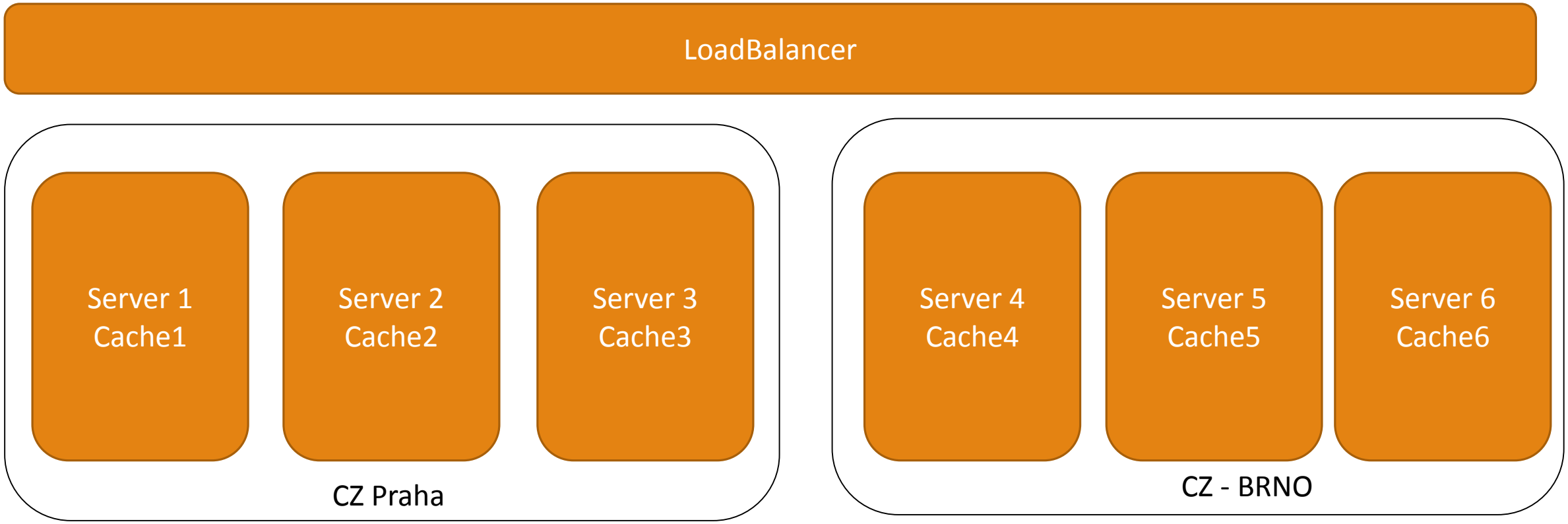
Načtení dat do cache (Write) –
- Lze vložit všechna data
- Lze vkládat data postupně

Přístup k datům v cache (Read)
- Na základě identifikátoru, hashe objektu, ….

Obnova dat v cache (Aktualizace dat)
- Obvykle aplikační záležitost
- Z principu použití cache je těžké říci kdy aktualizovat a jak

Revokace objektů (Čištění cache)
- Automatická – po vypršení TTL, po určité době kdy nebyl objekt přečten, po přeplnění vyhrazené kapacity
- Časová – automatizovaný JOB vyčistí celou cache

Case study Node cache



Nodová cache (cache per member)

Enterprise aplikace mají obvykle několik serverů (jednotky až stovky).

Výhody:

- Nodová cache má na každém z těchto serverů vlastní životní cyklus
- Není závislá na datech na ostatních serverech
- Lze využít vlastní jednoduchou implementaci

Standardní problémy:

- Data na jednotlivých node jsou různá
- Jiné cykly čištění cache
- Při revoke objektu se musí revoke provést na všech nodech

Implementace

- Vlastní implementace
- Jboss TreeCache http://docs.jboss.org/jbosscache/1.4.0/TreeCache/en/html_single/index.html
- Ehcache http://www.ehcache.org/

Cluster cache

Enterprise aplikace mají obvykle několik serverů (jednotky až stovky).

Způsoby propagace dat:

- Replikované
- Distribuované

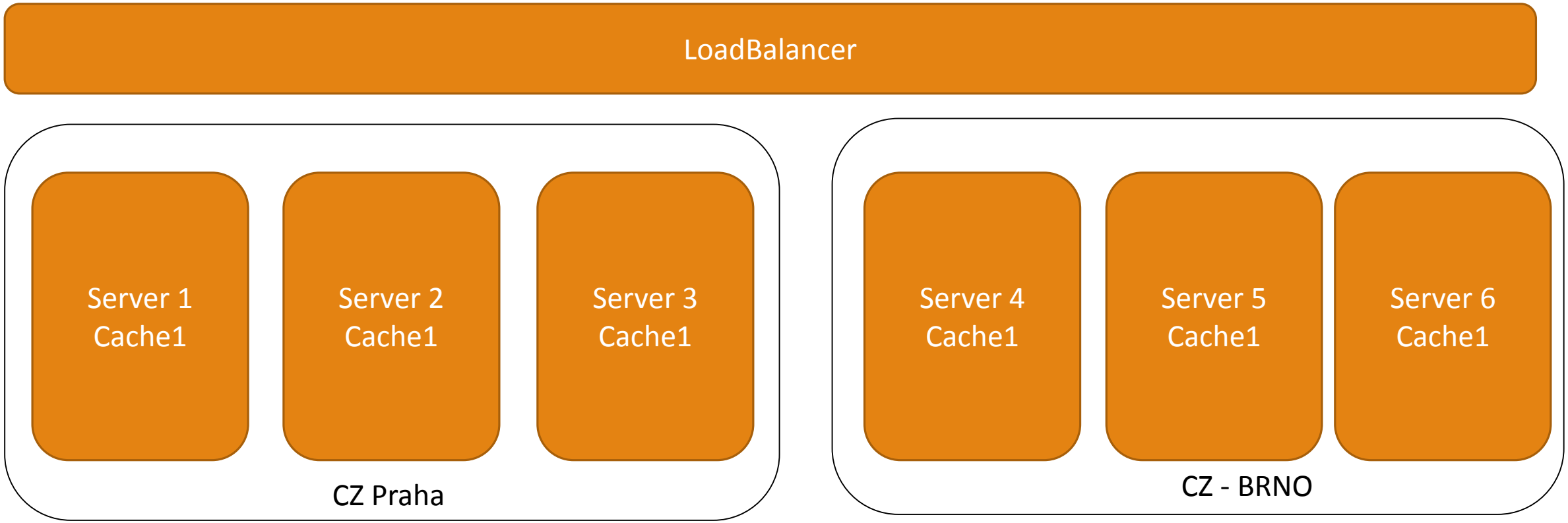
Typy synchronizací:

- TCP/IP
- UDB

Implementace

- Inifinispan - http://infinispan.org/
- Hazelcast - https://hazelcast.org/
- Jboss TreeCache http://docs.jboss.org/jbosscache/1.4.0/TreeCache/en/html_single/index.html
- Redis - https://redis.io/

Case study cluster cache



Výhody a nevýhody cluster cache

Data se získají pouze jednou a následně jsou dostupná na všech node.

Zneplatnění či reset cache má vliv na všechny nody v clusteru.

Snižuje zátěž prvku (Single Point of Failure – například DB).

V případě replikovaných cache, odpojení jednoho node z clusteru nevynutí přenačtení cache na všech serverech.

Nutná synchronizace dat.

Zápis zpravidla znamená zneplatnění cache.

Výkonový problém jednoho node v clusteru způsobí výkonový problém celé cache.

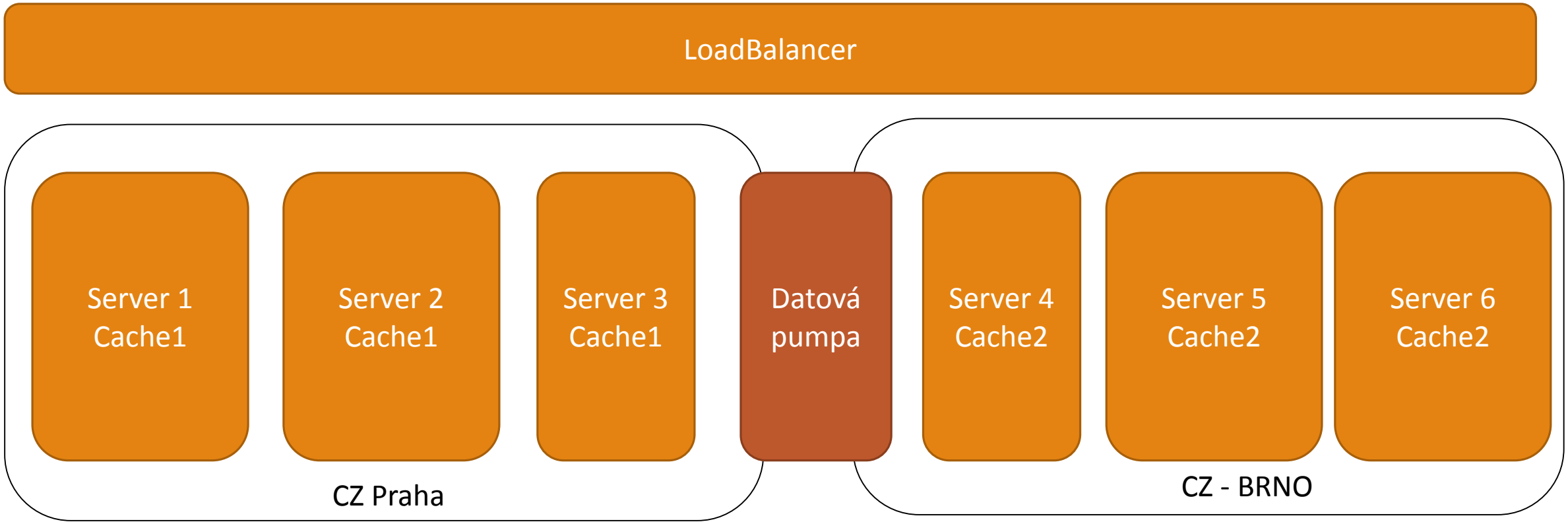
Problém přenosu velkého množství dat mezi lokalitami.

Problém při alokování výlučného zámku na cache.

Kvalitní řešení jsou často velmi drahá. Free verze neumožňují správnou diagnostiku a mnohdy cache musí nastavovat vývojář bez expertní znalosti dané implementace.

Restart serveru nevynutí reset cache.

Case study – cluster cache



Obrázky, videa?

Problém:

Aplikace pracuje s obrázky (dynamické). Jakým způsobem tyto obrázky získat a distribuovat na klienta.

Odpověď:

Statický obsah. Resp statické úložiště (Apache, Nginx).

Data jsou okamžitě dostupná, jejich získání nezatěžuje aplikační server.

Není potřeba tolik výkonu – HTTPD Apache pouze umožní získání dat.

Nutná synchronizace dat.

Data jsou obvykle dostupná všem. Nesoulad mezi aplikací a technickým řešením. Viz FB.

Obrázky – Privátní VS Public

Odkaz na fotku

Video Stream vs Download

Umístění videa v přehrávači neznamená, že video streamujete.

Stream – získávání videa po částech

- Nižší nároky na výkon systému
- Zvládne více videií na ráz
- Systém pracuje jen s aktuálním streamem
- Těžší na implementaci

Download

- Vyšší nároky na systém – video je nutné načíst do paměti
- Vyšší nároky na cache – pokud se použijí – video v nich existuje
- Vyšší nároky na datové přenosy
- Zabiják aplikačních serverů – obvykle vyčerpá HEAP
- Nižší nároky na výkon.

FE Aplikace a cachování

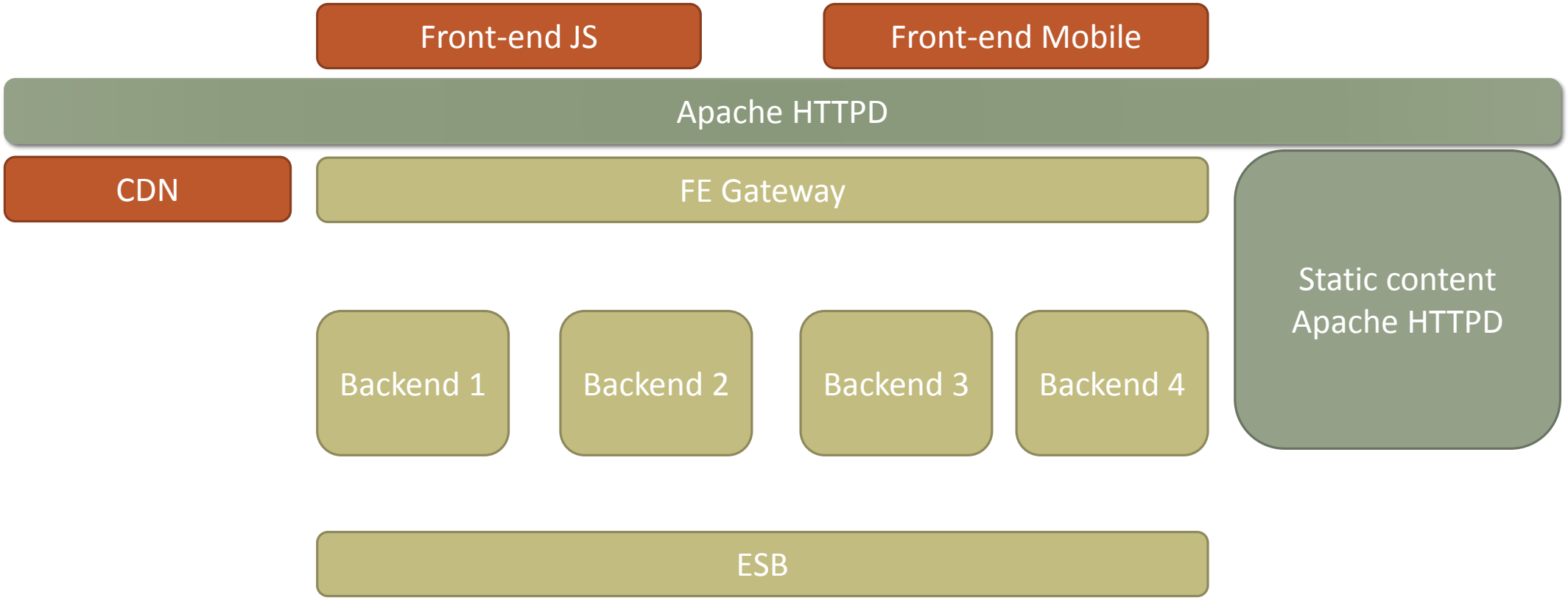
Hlavní cache = Browser

- Umožňuje načíst obrázky a soubory z paměti
- Sníží zátěž serveru
- Zrychlí načítání webu (eliminuje počet requestů na server - chrome umožňuje 6 requestů v jeden čas na jeden DNS záznam)
- Řídí se hlavičkami ze serveru
- Problém s přenačítáním nových verzí FE (řešeno hashe + timestamp či verzemi na jednotlivých souborech, které FE obsahuje)
- Problém s přenačítáním index.html a pod (řeší Apache)

Index.html - example

```
<link rel="stylesheet" href="css/bootstrap.min.css">
<link rel="stylesheet" href="css/bootstrap-flex.min.css">
<link rel="stylesheet" href="css/font-awesome.min.css">
<link rel="stylesheet" href="css/ekko-lightbox.min.css">
<link rel="stylesheet" href="css/main.css?31052017">

<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
<script src="js/jquery.min.js"></script>
<script src="js/app.js?28032018"></script>
<script src="js/controllers/mainCtrl.js?28032018"></script>
<!--Include all compiled plugins (below), or include individual files as needed -->
<script src="js/ekko-lightbox.min.js"></script>
<script src="js/bootstrap.min.js"></script>
```

Upravená architektura FE



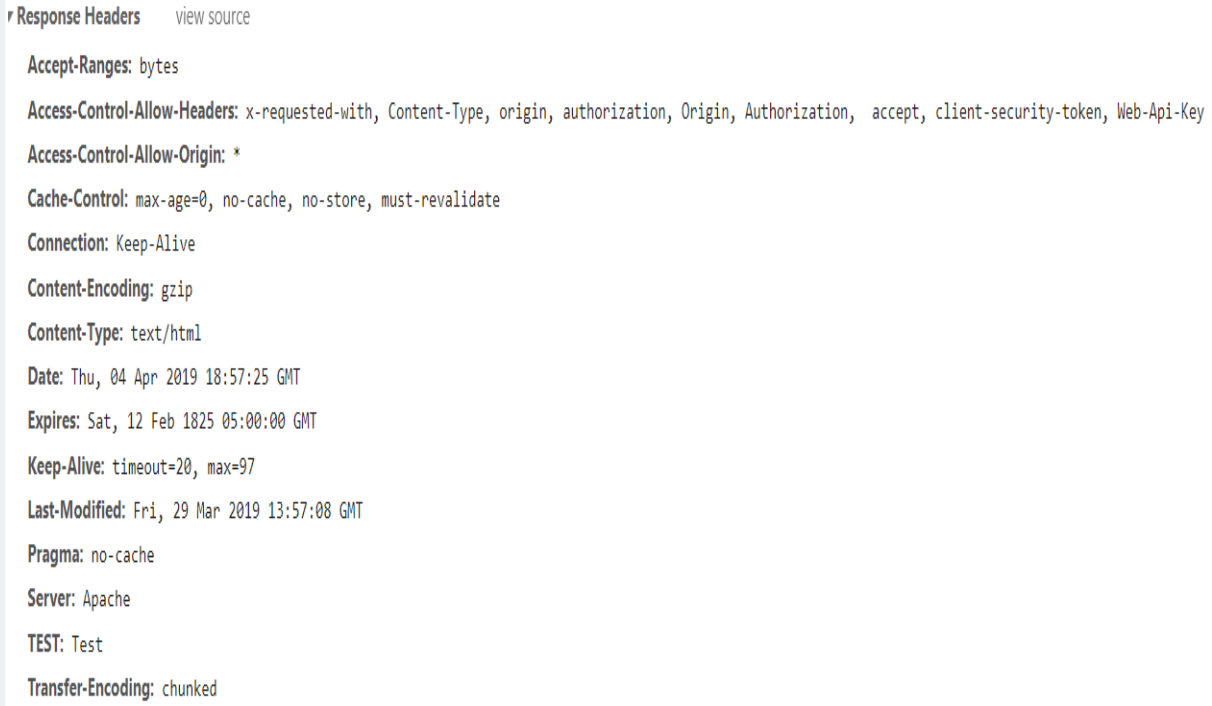
Front-end – využíván klientem

Apache pro rounting

Ostatní komponenty

Index.html Apache cache example

```
#Initialize mod_rewrite
RewriteEngine On
<FilesMatch "\.(html|htm|js|css)$">
  FileETag None
  <IfModule mod_headers.c>
    Header unset ETag
    Header set Cache-Control "max-age=0, no-cache, no-store, must-revalidate"
    Header set Pragma "no-cache"
    Header set Expires "Wed, 12 Jan 1980 05:00:00 GMT"
  </IfModule>
</FilesMatch>
```



https://stackoverflow.com/questions/22891965/how-to-disable-cache-of-apache

Apache HTTPD

Server pro HTTP požadavky

Zajišťuje - RW, HTTP->HTTPS redirect, White list, Black list, Modifikaci hlaviček, mime-type a pod

SSL offloading, Two-way-SSL

Slouží jak Proxy

Umožňuje přístup na FileSystem (FS)

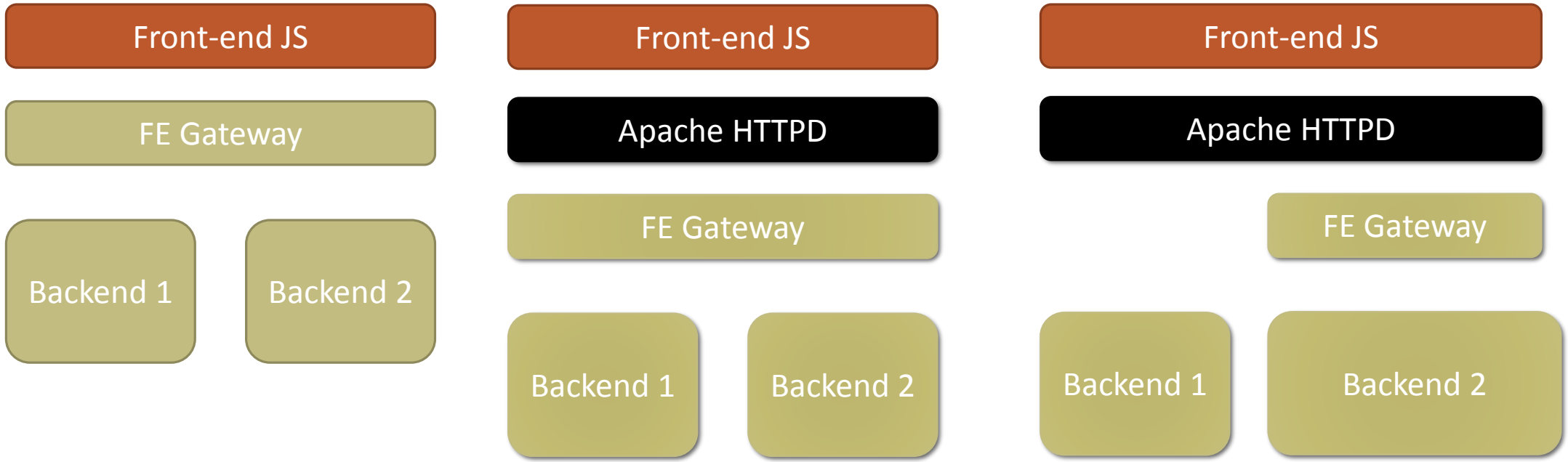
Umožňuje maskovat volání BE (cool URL)

Umožňuje routovat na základě contextů

Kompatibilní s konektory pro Aplikační servery (Weblogic, AJP apod)

Umožňuje vlastní load balancing na BE pod ním

Apache HTTP Example - integrace

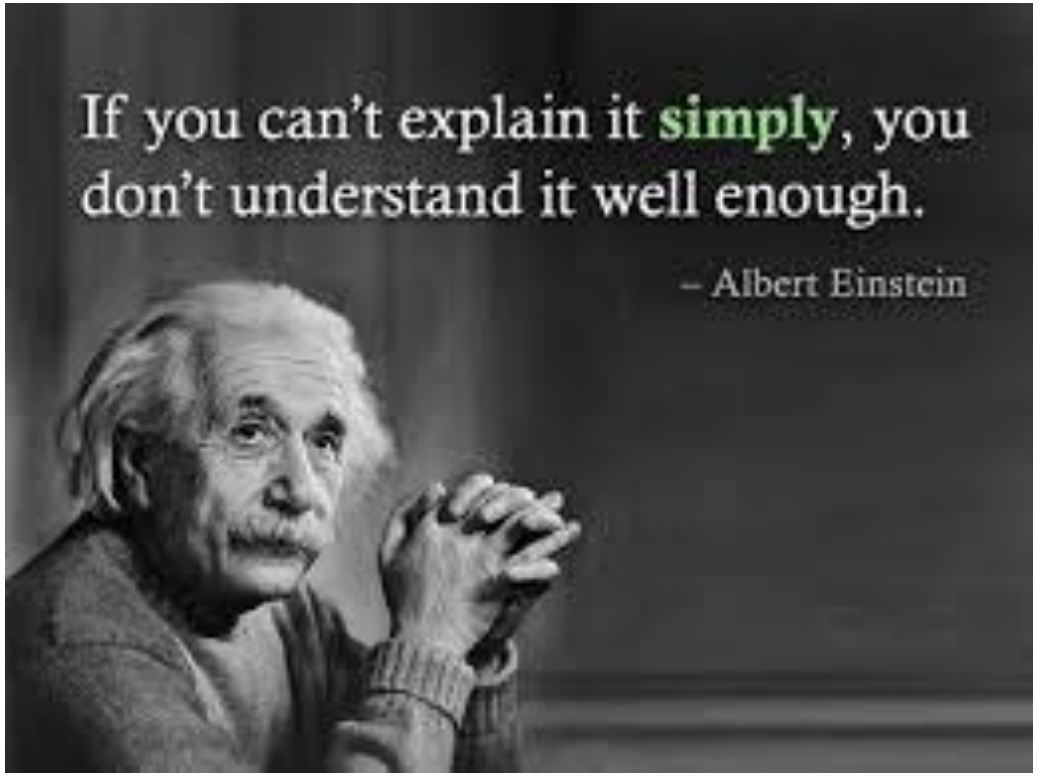


Apache HTTP example konfigurace

```
<VirtualHost *:80>
   ServerName www.abf-nadace.cz
   ServerAlias abf-nadace.cz
   DocumentRoot /var/www/abf/web
   ServerAdmin martin.tomasek@1pf.cz
   <Directory /var/www/abf/web>
        Options FollowSymlinks
        AllowOverride All
        Order allow,deny
        Allow from all
  </Directory>

</VirtualHost>


<VirtualHost *:80>

   ServerName www.siacr.cz
   ServerAlias siacr.cz
   DocumentRoot /var/www/sia/wordpress
   ServerAdmin martin.tomasek@1pf.cz
   <Directory /var/www/sia/wordpress>
        Options FollowSymLinks
        AllowOverride All
```

```
<VirtualHost *:80>


       ServerName www.static.stavebniakademie.cz
       ServerAlias static.stavebniakademie.cz
       DocumentRoot /var/www/staticContent/static
       ServerAdmin martin.tomasek@1pf.cz
       <Directory /var/www/staticContent/static>
              Options +Indexes +FollowSymlinks
              AllowOverride All
              Order allow,deny
              Allow from all
       </Directory>
       <IfModule mod_headers.c>
              Header set Access-Control-Allow-Origin: "*"
              Header set Access-Control-Allow-Headers: "X-Requested-With, Content-Type,
              Origin, Authorization, Accept, Client-Security-Token, Accept-Encoding, g-recaptcha-response"
              Header set Access-Control-Allow-Methods: "POST, GET, OPTIONS"
       </IfModule>
</VirtualHost>
```

Otázky?


If you can't explain it simply, you don't understand it well enough.
– Albert Einstein

Odhady – nezbytný background

Odhad pomáhá určit náročnost projektu

Odhadovat se dá cokoliv / jakákoliv fáze projektu

Kritický v rámci FTFP (Fix Time - Fix Price)

Existuje ohromné množství metodologií a kritérii na základě, kterých odhad stanovit.

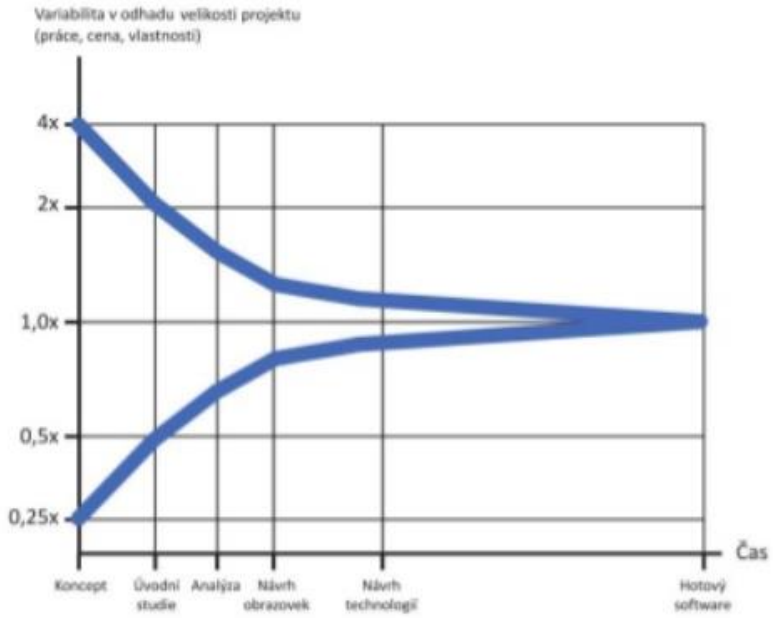
- Počet Use-case
- Použitá technologie
- Počet tabulek v DB
- Počet integrací na další systémy
- Výkon vývojářů
- Odhady z minulosti
- Expertní odhad – poslední možnost

Odhadování software – literatura / příklad

Steve McConnell - Odhadování softwarových projektů

Proveďte odhad kolik obyvatel má ČR. Stanovte spodní a horní meze.

Expertní odhad a kužel nejistoty



Kužel nejistoty reprezentuje statistickou chybu v odhadech, kterou udělali zkušení odhadovatelé na základě vstupních informací v jednotlivých fázích projektu.

Převzato z: https://image.slidesharecdn.com/pevntermnpevncenav2-151019131227-lva1-app6891/95/pevn-termn-pevn-cena-9-638.jpg?cb=1445260549