UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Računarstvo u oblaku

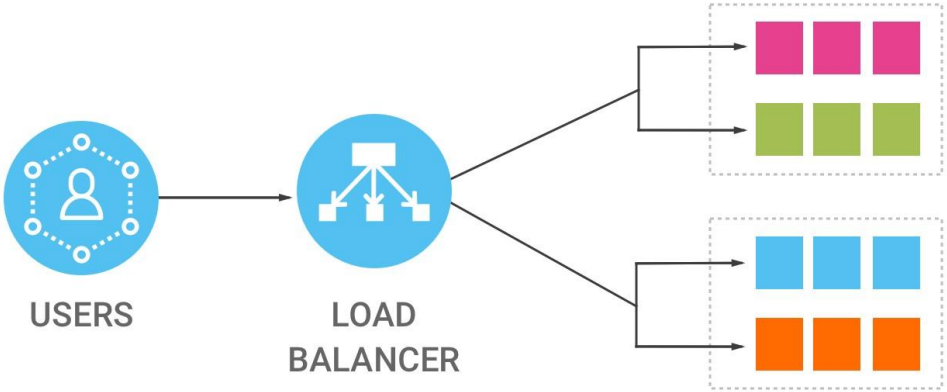
ms Helena Anišić

Zimski semester 2022/2023.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi

LOAD BALANCING



Šta je load balancing?

- Savremeni veb-sajtovi sa velikim prometom moraju da opslužuju stotine hiljada, ako ne i milione, istovremenih zahteva korisnika ili klijenata i da vraćaju ispravan tekst, slike, video ili podatke aplikacija, sve na brz i pouzdan način.
- Da bi se isplatilo skaliranje i da bi se zadovoljile ove velike količine, najbolja praksa modernog računarstva generalno zahteva dodavanje više servera.
- Balansiranje opterećenja se odnosi na efikasnu distribuciju dolaznog mrežnog saobraćaja preko grupe pozadinskih servera, takođe poznatih kao farma servera ili skup servera.

Šta je load balancing?

- **Load balancer** se ponaša kao „saobraćajni policajac“ koji sedi ispred vaših servera i usmerava zahteve klijenata na sve servere koji mogu da ispune te zahteve na način koji maksimizira brzinu i iskorišćenost kapaciteta i obezbeđuje da nijedan server nije preopterećen, što bi moglo da ugrozi performanse.
- Ako se jedan server pokvari, load balance-er preusmerava saobraćaj na preostale servere na mreži. Kada se novi server doda u grupu servera, load balance-er automatski počinje da mu šalje zahtev.

Šta je load balancing?

- Na ovaj način, **load balancer** obavlja sledeće funkcije:
  - Efikasno distribuira zahteve klijenata ili mrežno opterećenje na više servera
  - Obezbeđuje visoku dostupnost i pouzdanost slanjem zahteva samo na servere koji su onlajn
  - Pruža fleksibilnost za dodavanje ili oduzimanje servera prema zahtevima

Algoritmi za load balancing

- Različiti algoritmi za balansiranje opterećenja pružaju različite prednosti; izbor metode balansiranja opterećenja zavisi od vaših potreba:
  - **Round Robin** – Zahtevi se distribuiraju po grupi servera uzastopno.
  - **Least connections** – Novi zahtev se šalje serveru sa najmanje trenutnih veza sa klijentima. Relativni računarski kapacitet svakog servera se uzima u obzir da bi se odredilo koji od njih ima najmanje veza.
  - **Least time** – Šalje zahteve serveru izabranom formulom koja kombinuje najbrže vreme odziva i najmanje aktivnih veza. Ekskluzivno za NGINX Plus.

Algoritmi za load balancing

- **Hash** – distribuira zahteve na osnovu ključa koji definišete, kao što je IP adresa klijenta iliURL zahteva. NGINX Plus može opciono da primeni konzistentan heš da bi minimizirao redistribucijuopterećenja ako se promeni skup upstream servera.
- **IP hash** – IP adresa klijenta se koristi za određivanje koji server prima zahtev.
- **Random With Two Choices** – Nasumično bira dva servera i šalje zahtev na onaj koji je izabran primenom algoritma Najmanje veze (ili za NGINX Plus algoritam najmanjeg vremena, ako je tako konfigurisan).

Prednosti load balacing-a

- Smanjeno vreme zastoja
- Skalabilnost
- Suvišnost
- Fleksibilnost
- Efikasnost

NGINX

- Nginx je jedan od najpopoularnijih veb servera koji može da se koristi i kao reverse proxy, load balance, mail proxy, HTTP cache.
- Nginx je besplatan i open-source softver.
- Nginx Plus je omogućen uz pretplatu i nudi određeni broj dodatnih funkcionalnosti.

Primer 1

- Nginx konfiguracioni fajl

conf.d > my_conf.conf

```
upstream api {
    server api1:3000 weight=10;
    server api2:3000 weight=90;
}
server {
    listen 80;
    location / {
        proxy_pass http://api;
    }
}
```

compose.yaml

```
services:
  api1:
    #container_name: nodejs_api
    build: .
    ports:
      - "3001:3000"

  api2:
    #container_name: nodejs_api
    build: .
    ports:
      - "3002:3000"

  nginx:
    image: nginx:latest
    volumes:
      - ./conf.d:/etc/nginx/conf.d
    depends_on:
      - api1
      - api2
    ports:
      - 8081:80
```

Primer 2

- Nginx konfiguracioni fajl sa skaliranjem kontejnera
  - docker-compose up --scale api=2

conf.d > my_conf.conf

```
server {
    listen 3000;
    location / {
        proxy_pass http://api:3000;
    }
}
```

compose.yaml

```
services:
  api:
    #container_name: nodejs_api
    build: .

  nginx:
    image: nginx:latest
    volumes:
      - ./conf.d:/etc/nginx/conf.d
    depends_on:
      - api
    ports:
      - "3000:3000"
```

Unutar Visual Studio Code-a

-

Dobre prakse upotrebe Dockera

1. Upotrebiti oficijalnu docker sliku kao base sliku
    a. Primer node slika
2. Specificirati tačnu verziju docker slike
    a. Ne koristiti latest tag
3. Koristiti što je tanje moguće verziju OS-a
    a. Alpine linux
4. Keširanje slojeva
    a. Redosled navođenja slojeva je važan za optimizaciju
5. Koristiti .dockerignore za sve što ne treba u kontejneru
6. Koristiti multi-stage build
7. Koristiti najmanje privilegovanog korisnika
    a. Kreirati posebnog korisnika i grupu
8. Skenirati sliku kontejnera kako bi se prikazale ranjivosti
    a. dockerhub

Multi-stage build

- Jedan od načina optimizacije Dockerfile-ova pri čemu su oni i dalje čitljivi i održivi
- Jedan od najvećih problema prilikom buildovanja slika je veličina date slike.
    - Svakom RUN, COPY i ADD instrukcijom Dockerfile dodaje još jedan sloj na sliku
- Nekada je bilo često da se u toku razvoja koriste određene slike kontejnera, dok se za produkciju koriste mnogo lakše verzije.
    - Ovo je poznato kao builder pattern i nije idealno rešenje

Multi-stage build

- Određeni fajlovi nam trebaju tokom buildovanja slike, ali nam ne trebaju za samo pokretanje aplikacije
  - Neki artefakti koji nastaju tokom buildovanja slike
    - Alati za kompajliranje i buildovanje
    - Dependecies za unit testove
    - Privremeni fajlovi
- Primer: pom.xml, package.json nam trebaju za instaliranje biblioteka i dodataka, ali nam ta dva fajla ne trebaju za rad aplikacije
- Primer: JDK nam treba za kompajliranje Java koda, ali nam ne treba za izvršavanje Java aplikacije

Multi-stage build PRIMER

```
# Build stage
FROM maven AS build
WORKDIR /app
COPY myapp  /app
RUN mvn package
#Run stage
FROM tomcat
COPY --from=build /app/target/file.war /usr/local/tomcat/..
...
```

Materijali

- Dobre prakse: https://www.youtube.com/watch?v=8vXoMqWgbQQ