Servisno orijentisane arhitekture

Predavanje 6:Mikroservisi i paterni, Obrasci za pronalaženje servisa

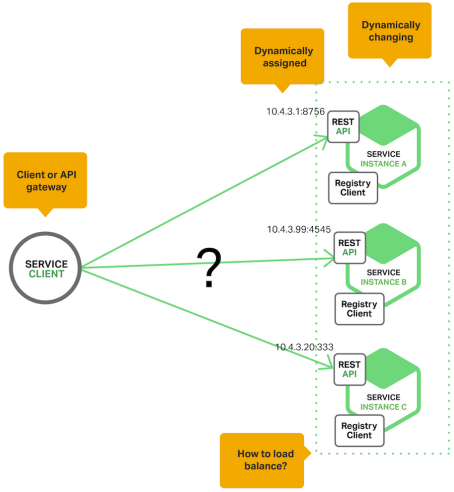
Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

- Svaki servis koji želi da komunicira sa nekim drugim servisom, treba da zna gde se on nalazi
- Ovo može biti izazovno, pogotovo u dinamičkim okruženjima
- Za rešenje ovog problema, postoje razne tehnike:
  - Pronalaženje od strane klijenta (client-side discovery)
  - Pronalaženje od strane servera (server-side discovery)
  - Registar servisa
  - Samoregistracija servisa
  - Registracija posredstvom treće strane

- Servisi po pravilu imaju potrebu da medjusobno komuniciraju – da pozivaju jedni druge
- U monolitnoj aplikaciji servisi mogu da pozovu usluge drugog servisa koristeći direktne pozive metoda na nivou programskog jezika.
- U tradicionalnom distribuiranom sistemu, servisi rade (uglavnom) na fiksnim dobro poznatim lokacijama (host/ port) i mogu (uglavnom) lako pozvati jedan drugog preko RPC ili HTTP/REST poziva.
- U modernoj mikroservisnoj arhitekturi broj instanci pojedinog servisa koji su na raspolaganju i njihova lokacija su promenljivi.
- Mora postojati mehanizam koji omogućava klijentima da upute zahtev dinamički promenljivom skupu servisnih instanci.

Problem

- Kako klijent nekog servisa (API gateway ili neka druga servisna komponenta koja nastupa u ulozi klijenta) može otkriti lokaciju servisne instance servisa koji joj je potreban?



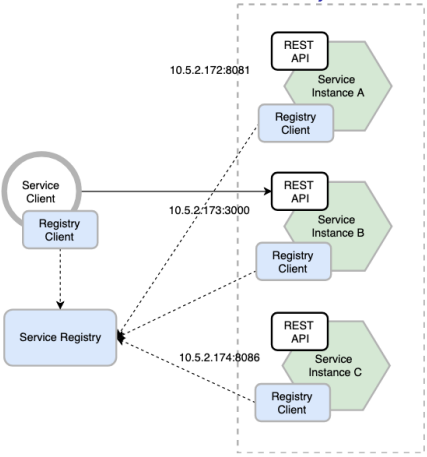
(https://walkingtree.tech/service-discovery-microservices/)

Faktori koji utiču na izbor

- Svaka instanca servisa ima javno dostupana API preko neke pristupne tačke (HTTP/REST ili neki drugi protkol) na odredjenoj lokaciji (host i port)
- Broj instanci odredjenog servisa i njihova lokacija se menjaju tokom vremena
- VM i kontejneri po pravilu dobijaju dinamički dodaljene adrese.

Pronalaženje od strane klijenta (client-side discovery)

- Kada se poziva servis, klijent lokaciju servisa dobija tako što kontaktira servisni registar, koji ima ažuran spisak lokacija na kojima je moguće pristupiti instancama servisa.
- Obično ovu funkcionalnost obezbedjuje razvojni okvir za razvoj mikroservisnih aplikacija.



(https://www.codeprimers.com/service-discovery-in-microservice-architecture/)

Dobre strane

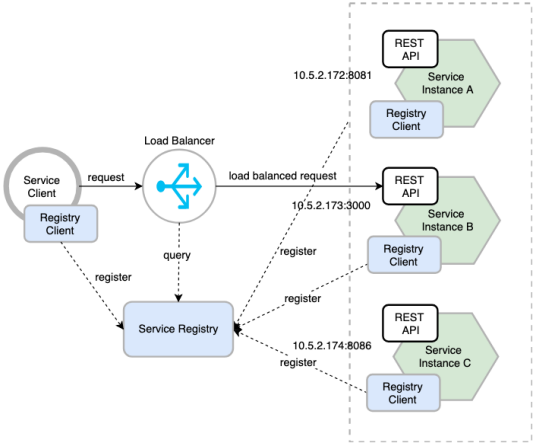
- Manji broj komponenti čije se lokacije menjaju
- Moguća dodatna optimizacija klijenta

Mane

- Stvara se jaka zavisnost klijenta od servisnog registra.
- Neophodno je implementirati logiku pronalaženja servisa u sam klijent, i to za svaki programski jezik koji je korišćen za razvoj klijentskih aplikacija.

Pronalaženje od strane servera (server-side discovery)

- Kada se poziva servis, klijent zahtev šalje posredstvom rutera (ima i ulogu balansiranja opterećenja) kojem se pristupa preko dobro poznate adrese
- Ruter proziva servisni registar kako bi saznao stvarne adrese dostupnih instanci servisa



(https://www.codeprimers.com/service-discovery-in-microservice-architecture/)

Prednosti

- Pojednostavljen klijentski kod
- Ponudjači cloud usluga često nude ovu funkcionalnost

Mane

- Osim ako se koristi cloud okruženje, ruter je nova komponenta koja se mora dodati u sistem i konfigurisati, a mora biti i repliciran za obezbedjenje visoke dostupnosti i pouzdanosti.
- Ruter mora podržavati sve potrebne protokole.
- Vlše mrežnih koraka izmedju klijenta i servera.

Uvod

- Klijent koristi mehanizam otkrivanja servisa (klijentski ili serverski) kako bi pronašao lokaciju servisne instance servisa koji mu treba.
- Problem - Kako klijent(u slučaju pronalaženja od strane klijenta) i/ili ruteri (u slučaju pronačlaženja od strane servera) može da utvrdi tačnu lokaciju raspoloživog servisa?

Faktori koji utiču na izbor

- Svaka instanca servisa svoj pristupni API (npr. HTTP/REST) mora učiniti dostupnim na nekoj mrežnoj lokaciji.
- Broj servisnih instanci i njihova mrežna lokacija mogu da se menjaju.
- VM i kontejneri dobijaju dinamičke IP adrese.
- Ne možemo se osloniti na to da će VM ili kontejner uvek imati istu adresu

Moguće rešenje

- Implementirati registar servisa, u vidu baze podataka servisa, njihovih instanci i njihvoih lokacija
- Sam registar može pozivati poseban API na servis instancama kako bi se uverio da su "žive i zdrave" – ping
- Moguće je i da servisi sami javljau svoje stane – healtcheck
- Klijenti servisa vrše upit nad ovim registrom kako bi saznali tačne lokacije instanci servisa koji su im potrebni
- Upit se vrši ili preko jedinstvenog identifikatora, ili u kombinaicji sa još nekim obeležijima
- Primer ovakvog registra je recimo Netflix Eureka.

Osobine

- Prednosti:
  - Klijent je u mogućnosti da lako pronadje adresu dinamički alociranih servisnih instanci
- Mane:
  - Osim ako je registar deo servisne infrastrukture, ova komponenta se mora dodati u sistemsko rešenje, podesiti i održavati.
  - Servisni registar (ako se ništa ne preduzme) je kritična tačka sistema.
  - Klijenti ne znaju unapred adrese pojedinačnih servisa, ali moraju znati adresu na kojoj se nalazi registar.
- **Neophodno je dodatno razmotriti mehanizam registracije servisa u registar**

Uvod

- Implementiran je neki od mehanizama za pronalaženje servisa i postoji registar servisa
- Instance servisa moraju se registrovati kod servisnog registra po završetku svog pokretanja i odraditi deregistraciju prilikom zaustavljanja.
- Moramo voditi računa da se jedan isti servis može pokrenuti nekoliko puta (kopija)

Problem

- Kako registrovati instance servisa na registru dostupnih servisnih instanci?
- I kako ih odjaviti iz registra.?

Faktori koji utiču na izbor:

- Instance servisa moraju se registrovati prilikom pokretanja i odjaviti prilikom zaustavljanja.
- Instanca servisa koja se tokom rada srušila mora se odjaviti iz registra.
- Servisne instance koje još uvek rade ali nisu u stanju da obradjuju zahteve, takodje treba da se odjave iz registra.

Samoregistracija servisa

- Servisna instanca je sama odgovorna za prijavljivanje (registrovanje) kod registra servisa
- Prilikom pokretanja, u momentu kada je spremna da prihvati zahteve, servisna instanca se prijavljuje registru (saopštavajući svoju adresu i port) što je čini vidljivom za proces pronalaženja
- Klijent mora povremeno da obnovi registraciju čime se potvrdjuje da je još uvek aktivan (živ)
- Prilikom zaustavljanja servisna instanca se sama odjavljuje iz registra
- NPR: Netflix Eureka koristi ovaj princip.

Osobine

- Prednosti:
  - Instanca servisa zna svoje sopstveno stanje tako da jemoguće implementirati model stanja koji je detaljniji od prostog UP/DOWN (STARTING, AVAILABLE, ...)
- Mane:
  - Sam servis i servisni registar su jako spregnuti.
  - Neophodno je implementirati kompletnu logiku za registraciju i odjavljivanje u svakom servisu (a to znači i u svakom jeziku koji je korišćen za pisanje servisne komponente)
  - Instanca servisa koja je pokrenuta, ali nije dostupna i sposobna za procesiranje zahteva vrlo često toga nije sama svesna pa neće ni odraditi odjavu sa registra.

Registracija posredstvom treće strane

- Pomoćna (3rd party) komponenta koja služi kao registrator preuzima odgovornost za pravovremenu registrtor prijavi tu instancu u registar, kada se instanca zaustavi registrator ukloni podatke o toj instanci iz registra
- Netflix registraciju instance servisa na registar i odjavljivanje instanci kada postanu nedostupne
- Kada se instanca servisa pokrene, Prana i AWS Autoscaling Group, kao i različiti alati za upravljanje klasterima ontejnera koriste ovaj princip.

Osobine

- Prednosti:
  - Kod samog servisa se pojednostavljuje u odnosu na situaciju kada se radi sasmoregistracija.
  - Regitrator može povremeno da proveri “zdravlje” servisa i da ga odjavi kada on prestane da se odaziva.
- Mane:
  - Registrator može da ima samo površno saznanje o internom stanju servisa, pa može biti uskraćen za informaciju da li servis koji je RUNNING stvarno i sposoban da obradjuje zahteve koji mu se upute
  - Ipak ovo može da se prevazidje redovnom proverom “zdravlja” servisa.
  - Osim u slučaju da je sam registrator deo infrastrukture - to je još jedna komponenta koju je potrebno dodati u sistem, održavati i konfigurisati
  - A mora biti i visoko dostupan, jer njegov otkaz ceo sistem može učiniti nefunkcionalnim.

Dodatni materijali

- Building Microservices, Sam Newman
- Microservices • Martin Fowler • GOTO 2014
- What are microservices?
- Microservices patterns

Kraj predavanja

Pitanja? :)