Napredni algoritmi i strukture podataka

Ograničenje stope pristupa (Rate Limiting), Token Bucket, TTL

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Ograničenje stope pristupa — uvod

- U računarskim mrežama, ograničenje brzine/stope se koristi za kontrolu brzine/stope zahteva poslatih ili primljenih od strane kontrolera mrežnog interfejsa
- Ograničavanje stope/brzine pristupa (Rate Limiting) je procedura koja nam omogućava kontrolu brzine kojom korisnici mogu da šalju zahteve sistemu
- **Rate Limiting** se uglavnom koristi za zaštitu servera od neželjenih rafala, zlonamernih napada
- Zaštita sistema od prekomerne upotrebe ograničavanjem koliko često korisnici mogu da im pristupe, ima nekoliko prednosti

- Pomaže protiv napada *denial-of-service*, pokušaja prijave *brute-force* i drugih vrsta nasilnog ponašanja korisnika
- Može i da se koristi kod različitih servisa da se vidi da li imamo dovoljno finansija da pristupimo nekakvom resursu
- Web servisi mogu da ga korite kada pružaju usluge korisnicima da bi odbili zahteve, ako su prekoračili limit

- Postoji razni tipovi Rate Limiting-a npr:
  - **Rate limiter korisnika** omogucćava nekim grupama korisnika ograničen pristup sistemu — broj/trajanje zahteva korisnika obično je vezan za njihove ključeve ili IP adrese
  - **Rate limiter istovremene/serverske brzine** prati koliko je paralelnih sesija ili veza dozvoljeno za nekim grupama korisnika — ublažava DDoS napade
  - **Rate limiter lokacije** ograničava brzine/stope pristupa za neke regione, kao i za definisani vremenski period — moguće je definisati različite stepene pristupa za razne lokacije

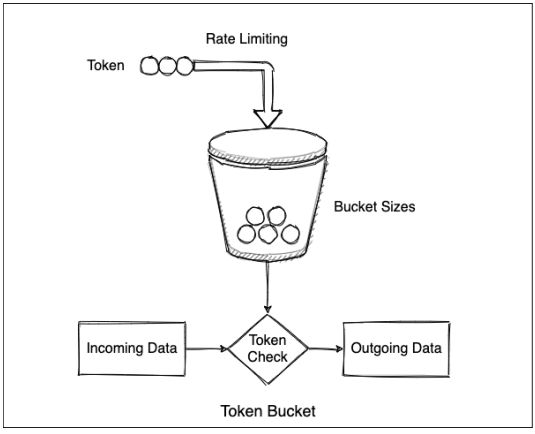
Rate limiter kod sistema za skladištenje podataka

- Neki sistemi za skladištenje velikie količine podatak su implementirali ovaj mehanizam
- To ih nekada izdvaja od drugih sličnih sistema i zato su čest izbor korisnika
- **RocksDB**, na primer, direktno podržava ovaj mehanizam, i zato je ponekad češći izbor od recimo **LevelDB**
- Ali pored problema koje ova grupa algoritama rešava, za ovaj tip sistema vezuje se još zanimljivih upotreba

- Kod upotrebe ovih sistema, korisnici možda žele da priguše maksimalnu brzinu pisanja u okviru nekog ograničenja iz mnogo razloga
- Na primer, brzi zapisi izazivaju strašne skokove u kašnjenju čitanja ako prekorače definisani prag
- RocksDB ima mogućnoti da korisnici mogu da podese Rate limiter kako njima odgovara
- Pruža cak i mogućnost dinamičkog ograničenja — **Auto-tuned Rate Limiter**
  (RocksDB Docs, http://rocksdb.org/blog/2017/12/18/17-auto-tuned-rate-limiter.html)

Tocken Bucket — uvod

- Ovo je najjednostavniji algoritam za ograničavanje brzine pristupa
- Jednostavno pratimo broj zahteva napravljenih u zadatom vremenskom intervalu
- Zbog svoje jednostavnosti, dosta se koristi
- Google cloud koristi ovaj algoritam (ili je koristio), za Task Queue opciju koja se nudi koirsnicima kao usluga
- Jednostavno se implementira, i lako može da se poveže sa velikim brojem različitih slučajeva korišćenja



(What is Token Bucket and Leaky Bucket algorithms)

Tocken Bucket — algoritam

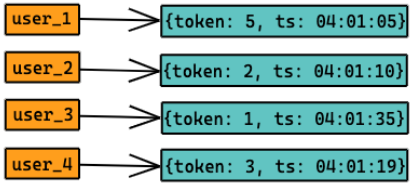
- Za svaki zahtev korisnika treba:
  - Proveriti da li je vreme proteklo od poslednjeg resetovanja brojača vremena
  - Ako vreme nije isteklo, treba proveriti da li korisnik ima dovoljno preostalih zahteva da obradi dolazni zahtev
  - Ako korisniku nije preostalo slobodnih zahteva, trenutni zahtev se odbacuje uz nekakvu poruku
  - U suprotnom, smanjujemo brojač za **1**, i vršimo obradu dolaznog zahteva
  - Ako je vreme proteklo, tj. razlika resetovanog vremena i trenutnog vremena je veća od definisanog intervala, resetujemo broj dozvoljenih zahteva na unapred definisano ograničenje, i definišemo novo vreme resetovanja

Tocken Bucket — primer

Pimer: 3/min:

- REQ **11:01:20** – > BUCKET [11:01:05, 3] => OK
- REQ **11:01:25** – > BUCKET [11:01:05, 2] => OK
- REQ **11:01:30** – > BUCKET [11:01:05, 1] => OK
- REQ **11:01:35** – > BUCKET [11:01:05, 0] => FAIL
- REQ **11:03:00** – > BUCKET [11:03:00, 2] => OK // uradimo update vremena, broja tokena, pustimo zahtev i smanjimo broj tokena za **1**
- ...

- Za aplikativne stvari, obično u memoriji ili nekoj sistemu koji čuva podatke u memoriji zbog brzine
- Pošto mi pravimo sistemsku stvar, i pravimo storage engine — pa močemo čuvati u našem sistmu :)
- Svaki korisnik može da bude **ključ**, dok **vrednost** može da sadrži vremensku odrednicu i broj tokena



Zadaci

- Implementirati Token Bucket algoritam, kao pomoc koristiti fajl *helper.go* — konfiguracione elemente kao i same podatke čuvati u memoriji
- Proširiti implementaciju sa prethofnih vežbi tako da se Token Bucket algoritam integriše u vaš sistem
- Sva podešenja vezano za algoritam treba da budu specificirana kroz konfiguracioni fajl
- Podatke vezano za Token Bucket čuvati u vašem sistemu
- **Dodatni zadatak:** Integrisati Token Bucket sa vaši sistemom, li podatke cuvati striktno u memoriji ili nezavisnom fajlu; uporediti performanse i jednostavnost implementacije