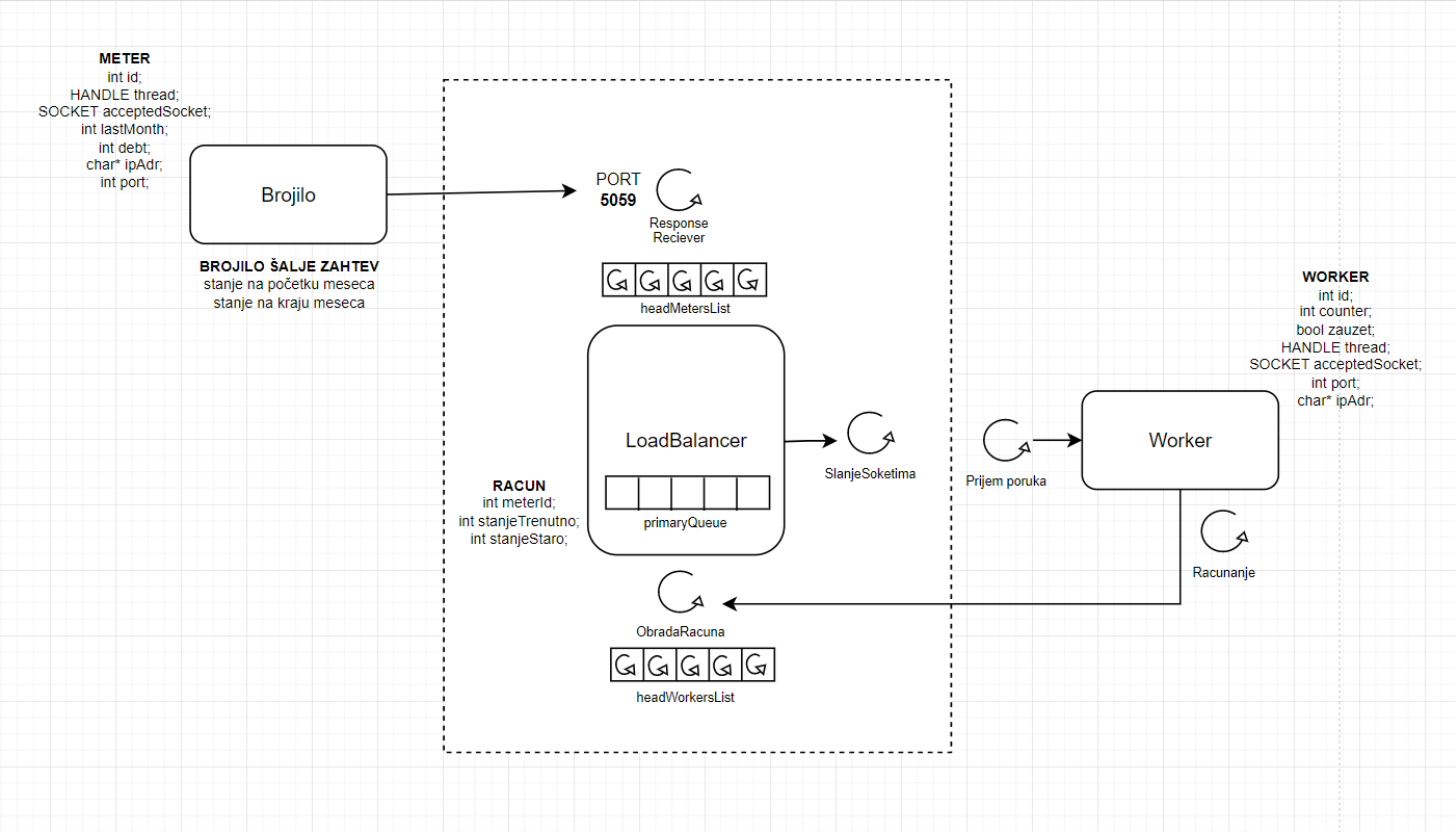
Load Balancer (LB-G1)

UVOD

Uvođenjem pametnih brojila nastaje problem dugog vreme proračuna potrošnje računa. Cilj ovog projekta je simulacija rada Load Balancer sistema za obračun mesečne potrošnje struje pametnih brojila (meter), kako bi raspodelili opterećenje proračuna i ubrzali proces za veće količine klijenata. Ovaj sistem ima primetne benefite kod većeg broja brojila, dok je za manje brojila to manje primetno.

DIZAJN

Aplikacija se sastoji od 3 tipa komponenti: *LoadBalancer, Worker, Meter*.
Prilikom prijave na sistem, meter se javlja balanceru na portu 5059 sa ciljem da se registruje, gde ga on prihvata i čuva u headMeterList. Ostvaruje se konekcija. Svakom meteru dodeljuje novi thread koji osluškuje na soketu račune koje će registrovani meteri slati i zatim ih čuva u primaryQueue tipa RingBuffer-a.
Sa druge strane jedna nit očekuje poruke i zahteve za registraciju novih workera koji će biti na raspolaganju Load Balanceru u headWorkersList redu.
Ukoliko se u primaryQueue pojave poruke i imamo workera na raspolaganju, naša nit (funkcija Slanje soketima) uzima prvog slobodanog workera i sledeći račun koji je na redu. Računa ga i čeka odgovor u niti ObradaRacuna workera koja radi samo dok ne dobije odgovor. Promeni zaduženje u strukturi brojila i nit se zatvara.
Worker samo čeka zahteve i vrši proračune. Dug u toku meseca koji je dobio računanjem šalje nazad LoadBalancer-u kako bi ga on sačuvao. (Zaduženje se računa tako što od trenutnoStanje oduzmemo stanjeProslogMeseca i pomnožimo sa cenom 1KWh, npr. u našem slučaju 117 dinara).



STRUKTURE PODATAKA

U projektu su korišćene strukture podataka *RingBuffer, List/Queue.*
RingBuffer (Circular Buffer) je FIFO struktura koja skladišti pristigle zahteve za računanje tj. račune sa strane brojila, čuva ih i kada se stvori prilika, šalje ih ka slobodnom workeru na obradu. Koristimo ga jer želimo da šaljemo zahteve redom, a ukoliko dođe do zakrčenja i punjenja buffera možemo se vratiti na početak i opet obračunati dug. Neće doći do gubitka računa jer račun čuva trenutnu vrednost i vrednost na kraju prošlog obračunskog perioda iz metera.
Lista je takođe FIFO struktura u koju skladištimo Metere i Workere i olakšava nam pretragu i prebrojavanje istih.

OPISIVANJE NAJBITNIJIH FUNKCIJA

Funkcije na workeru:

`void SetNonblocking(SOCKET* socket)` – funkcija koja se obavlja na postojećim soketima i ona soket konfiguriše u neblokirajući režim. Takođe postoji funkcija koja soket prebacuje u blokirajući režim.

`SOCKET SetConnectedSocket(u_short port)` – funkija koja inicijalizuje soket na zadatom portu i vraća ga.

Funkcije na meteru:

`SOCKET SetConnectedSocket(u_short port)` – funkcija koja inicijalizuje soket na zadatom portu i vraća vrednost soketa.

`int GenerateRandomNumber(int minNumber)` – funkcija koja random generiše int vrednost, koja je veća od zadate vrednosti.

Funkcije na LB:

`DWORD WINAPI WorkWithSockets(void* vargp)` – funkcija koja prati dešavanja na prosleđenom soketu, selektujemo ga i prati odgovore. Ukoliko pristigne poruka zahteva za konekciju, povratna vrednost selekcije bude > 0 i znamo da je pristigao neki zahtev. Pravimo novi meter i čuvamo ga na kraju liste.

`DWORD WINAPI PrijemDaljihPoruka(void* vargp)` – funkcija koju pokreće WorkWithSockets u zasebnim tredovima koje čuvamo u meteru. Svrha funkcije je da očekuje poruke od registrovanih metera i da pravi račune koje ce čuvati u RingBuffer.

`DWORD WINAPI WorkWithSocketsWorker(void* vargp)` – funkcija koja osluškuje na drugom soketu i čeka zahteve za registraciju novih workera, koje kreira i čuva u novom redu.

`void SlanjeSoketima()` – funkcija koju pozivamo u main niti. Njena uloga je da proverava broj workera i broj računa koji čekaju na računanje. Ukoliko ih imamo, vršimo preuzimanje računa iz buffera i šaljemo prvom slobodnom workeru. Worker u svojoj strukturi dobija thread koji poziva funkciju ObradaRačuna(worker).

`DWORD WINAPI ObradaRacuna(void* vargp)` - pozivamo je za slobodnog workera koga zauzme tako što property zauzet postavi na true i pošalje mu zadatak u formatu METER ID/STANJE STARO/STANJE NOVO. Zatim primi odgovor od workera i pozove funkciju UvecajDug koja izmeni meter i oslobodi workera. Tu se ta nit završava do sledeće upotrebe tog workera.

ZAKLJUČAK

Rešenje ovog zadatka je rešeno jednostavnom komunikacijom pomoću TCP transportnog protokola. Prilagođeno je istovremenom radu sa više workera i metera zahvaljujući upotrebi threadova. LoadBalancer pamti listu svih workera i metera kako bi vodio evidenckiju o njima i znao kome da šalje zaduženja. Ovo je ostvareno uz pomoć strukturiranja unutar lista i kružnog buffera.

POTENCIJALNA UNAPREĐENJA

Potencijalno rad sa redom u kojem čuvamo metere mogli bi zameniti sa čuvanjem u neki fajl ili bazu podataka. Takođe u bi se funkcije metera mogle unaprediti sa plaćanjem i umanjenjem zaduženja kod Load Balancera za taj meter.

Stefan Popović PR75/2019
Rastislav Zaskalicki PR77/2019