Fakultet tehničkih nauka

Univerzitet u Novom Sadu

***Load-Balancer***

Industrijski komunikacioni protokoli

Autori:

PR 34/2021 Stefan Cvjetković

PR 25/2021 Saša Perišić

Novi Sad 2025

**Sadržaj**

1.Uvod...............................................................................3

2.Dizajn.............................................................................4

3.Strukture podataka........................................................5

3.1.Razlozi za izbor struktura podataka………………………………….……….5

3.2.Opis semantike podataka koje sadrže strukture………………………..6

4.Testiranje.......................................................................7

5.Zaključak........................................................................8

**1. Uvod**

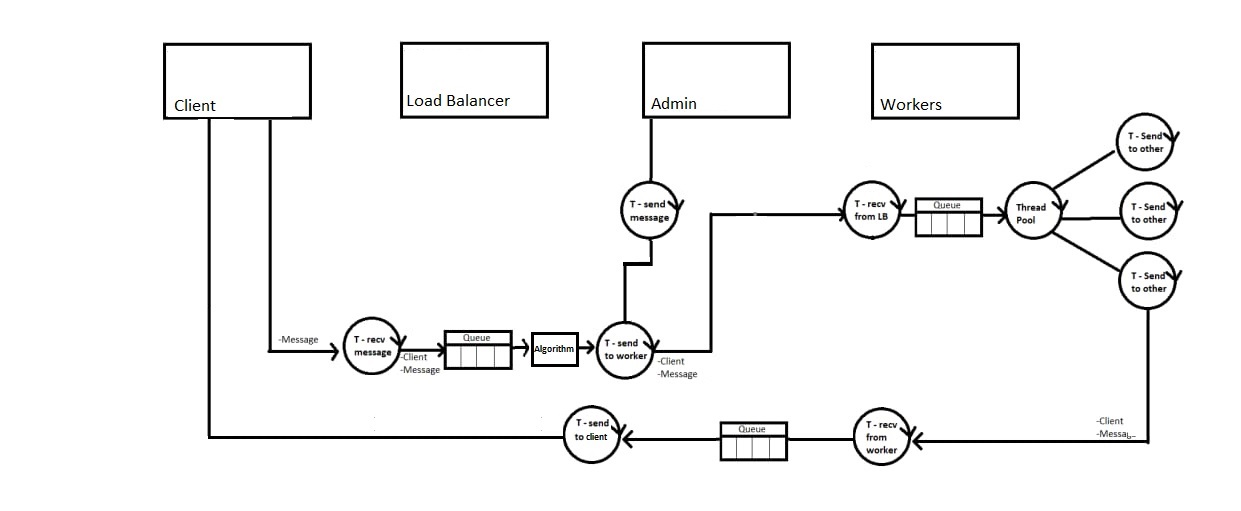
Cilj ovog projekta je implementacija sistema za balansiranje opterećenja (Load Balancer) koji osigurava efikasnu obradu zahtjeva korisnika.

Sistem se sastoji od tri osnovne komponente: client, load balancer i worker, uz dodatnu ulogu admina za upravljanje parametrima sistema. Klijent generiše zahtjeve koji se proslijeđuju kroz sistem i na kraju vraćaju klijentu u obradenom obliku.

Zadatak je implementiran prema klijent-server arhitekturi gde klijent šalje koliko želi brojeva da bude u zahtjevu i opseg tih brojeva . Load Balancer prima te brojeve generiše ih, smješta u strukturu i strukturu smješta u red. Zahtjeve proslijeđuje dostupnim workerima na obradu. Admin ima mogućnost da mijenja dužinu reda kod workera, dok se dinamično skaliranje postiže dodavanjem novih niti koje predstavljaju workere.

**2.Dizajn**

***Opis dizajna implementiranog rješenja***



Klijent šalje broj elemenata koliko će ih biti u zahtjevu i šalje opseg tih elemenata.

Load Balancer prima te brojeve i generiše elemente zahtjeva. Smješta te elemente u strukturu i strukturu u red. Elemente proslijeđuje slobodnom workeru.

Worker obrađuje zahtjeve koji su pristigli sa load balancer – a, tj. svaki worker ima svoj red u koji se smještaju zahtjevi. Worker kada obradi zahtjev smješta rezultat u red odakle se šalju klijentu.

Admin ima funkciju da promjeni veličinu reda kod workera.

**3.Strukture podataka**

**3.1.Razlozi za izbor struktura podataka**

U implementaciji ovog zadatka koristili smo strukturu reda (queue). Korišćenje strukture reda za ovaj tip problema dinamičkog balansiranja opterećenja ima nekoliko prednosti:

* FIFO pristup (First In First Out)
* Struktura reda koristi ovaj pristup što znači da zahtjevi koji prvi stignu u red čekanja će biti prvi obrađeni.
* FIFO pristup je koristan u situacijama gdje je redoslijed obrade bitan, što često i jeste slučaj kod zahtjeva koji pristižu u sistem.
* Jednostavno upravljanje redom čekanja
* Queue struktura omogućava jednostavno dodavanje novih zahteva na kraj reda i uklanjanje zahtjeva sa početka reda, što olakšava upravljanje zahtjevima sistema.
* Poboljšana kontrola opterećenja
* Red čekanja omogućava da se popunjenost (broj zahteva) jednostavno prati. Time je lako kontrolisati trenutno opterećenje sistema.
* Lakše praćenje sistema
* Kroz red čekanja je lakše pratiti koliko zahtjeva trenutno čeka na obradu. Na taj način se olakšava odluka o dodavanju ili uklanjanju WR instanci.

**3.2.Opis semantike podataka koje sadrže strukture**

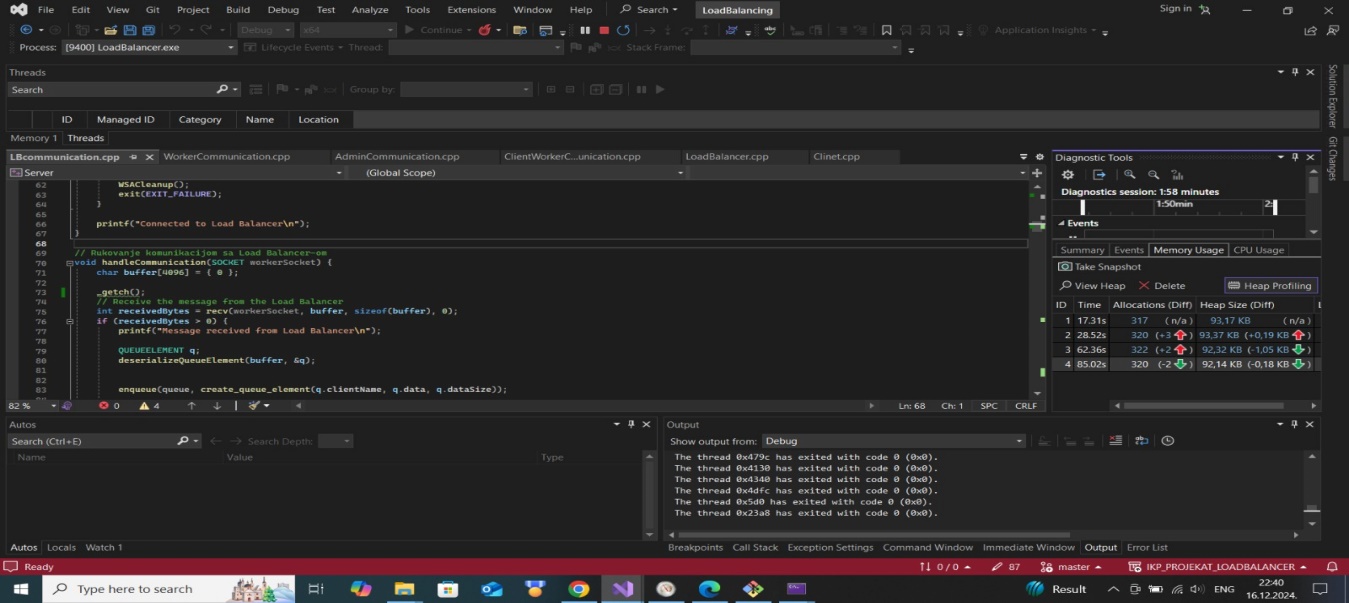
Podaci u strukturi reda imaju jasno definisano značenje koje omogućava efikasno upravljanje zahtevima i funkcionalnošću sistema. U nastavku su opisani ključni aspekti i objašnjenja semantike ovih podataka:

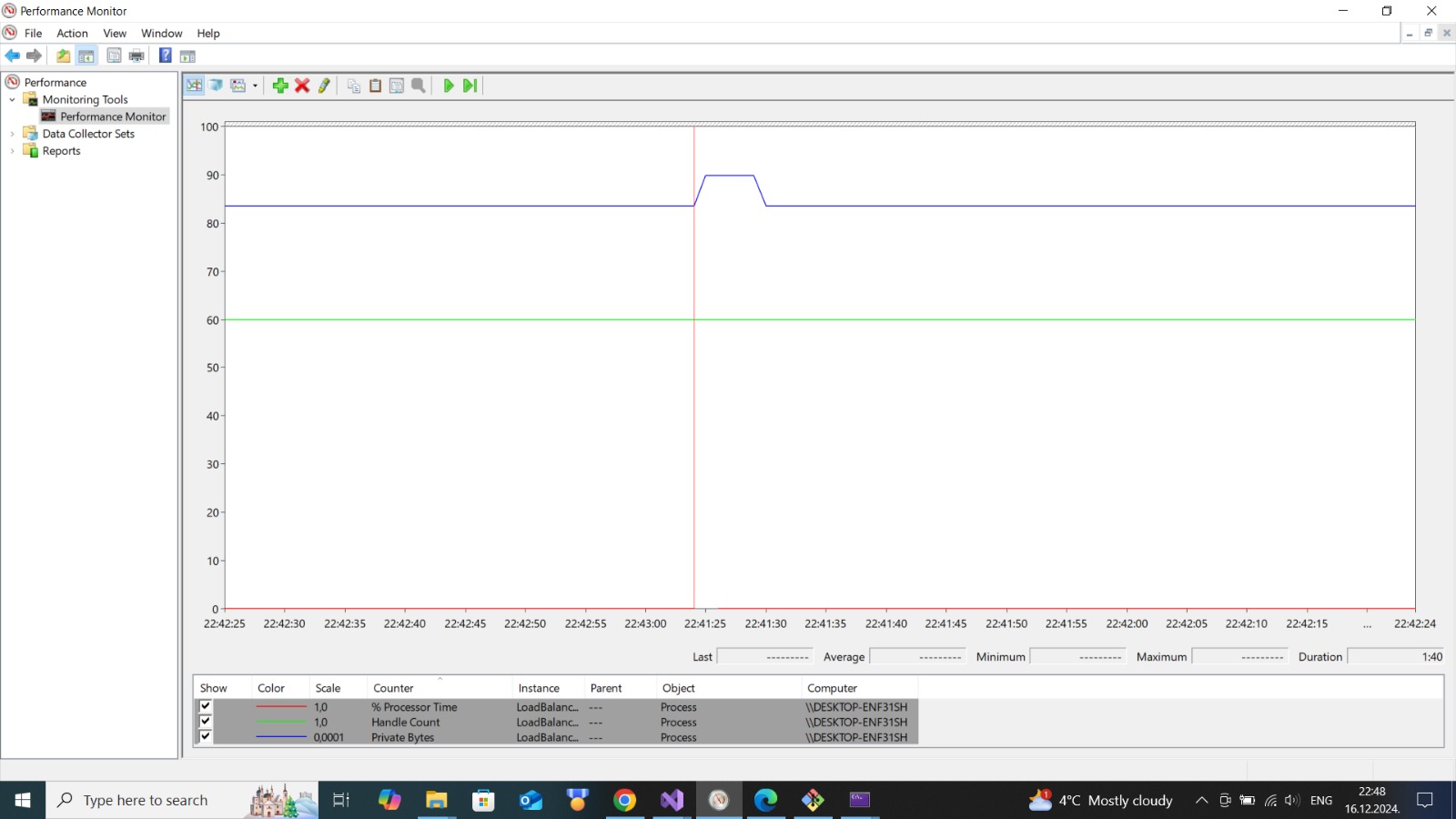
* Zahtjevi za obradu
* Podaci o zahtjevima: Svaki zahtjev koji se dodaje u red čekanja sadrži informacije koje su potrebne za obradu (ime klijenta,bafer brojeva)
* Struktura podataka za zahtjev: Moguće je da svaki zahtjev bude predstavljen strukturom podataka koja sadrži informacije o zadatku koji treba obaviti.
* Red čekanja
* Semantika reda: FIFO semantika što znači da se zahtjevi obrađuju prema redoslijedu dodavanja u red.
* Kontrola popunjenosti: Popunjenost reda se prati kako bi se procijenilo opterećenje sistema. Ako je red kod workera popunjen krerira se novi worker.
* Dinamičko dodavanje i uklanjanje zahteva:
* Dodavanje novih zahtjeva: Novi zahtjevi se dodaju na kraj reda.
* Uklanjanje zahtjeva :Kada se zahtev obradi odnosno kada WR instanca završi obradu, zahtev se uklanja sa početka reda.

.

**4.Testiranje**

Projektni zadatak je testiren putem stress testa:

• Test predstavlja "laganiji" test, u kom nije cilj bio preopterecenje procesora nego pracenje opterecenja prilokom rada aplikacije u normalnim uslovima. Podaci s klijenta su se slali na svakih 1.5 sekundi dok su worker role vrsile obradu na svakih 7 sekundi. ****



**5.Zaključak**

Na osnovu rezultata testiranja, možemo izvući nekoliko zaključaka:

* Ekasnost sistema: Preliminarna testiranja pokazuju zadovoljavajuću ekasnost sistema u upravljanju zahtjevima i dinamičkom prilagođavanju broja Worker instanci u skladu sa opterećenjem. Sistem uspešno održava stabilnost u broju radnika kada je popunjenost reda workera maksimalna kreira se novi worker i ako postoji mjesta na već kreiranim workerima popunjavaju se prvo oni.
* Performanse servera: Performanse servera tokom testiranja izgledaju zadovoljavajuće, s obzirom na to da je moguće obraditi zahteve kako ih redoslijedom primaju, čak i kada je opterećenje sistema povećano.
* Skalabilnost sistema: Testiranje sa više zahtjeva ukazuje na neophodnost prilagođavanja broja Worker-a u skladu sa povećanjem opterećenja, što sugeriše na ekasno rukovođenje resursima u cilju održavanja performansi sistema.