Industrijski komunikacioni protokoli u elektroenergetskim sistemima

SERVIS ZA DISTRIBUIRANO SKLADIŠTENJE PODATAKA

Projektna dokumentacija

Autori:

- Nikola Bajagić PR-43/2019
- Nemanja Petrović PR-44/2019

Mentor:

- Marina Stanojević

Sadržaj

1. Uvod

Zadatak:

Razviti servis za distribuirano skladištenje podataka u cilju postizanja visoke pouzdanosti. Distribuirana baza se sastoji od proizvoljnog broja čvorova koji međusobno komuniciraju, a takođe svaki čvor može biti i pristupni servis. Upis podataka u bazu treba da bude distribuirana transakcija koja se izvršava kroz *2PC* protokol. Takođe, prilikom pokretanja novog čvora, čvor trebe da se poveže sa svim ostalim čvorovima u mreži i da uradi *integrity update*.

Klijent treba da ima mogućnost pristupa bilo kom čvoru mreže i slanje podataka o studentu. Nakon slanja zahteva, klijent ostaje aktivan i nudi mogućnost ponovnog slanja sve dok se ne unese komanda za kraj.

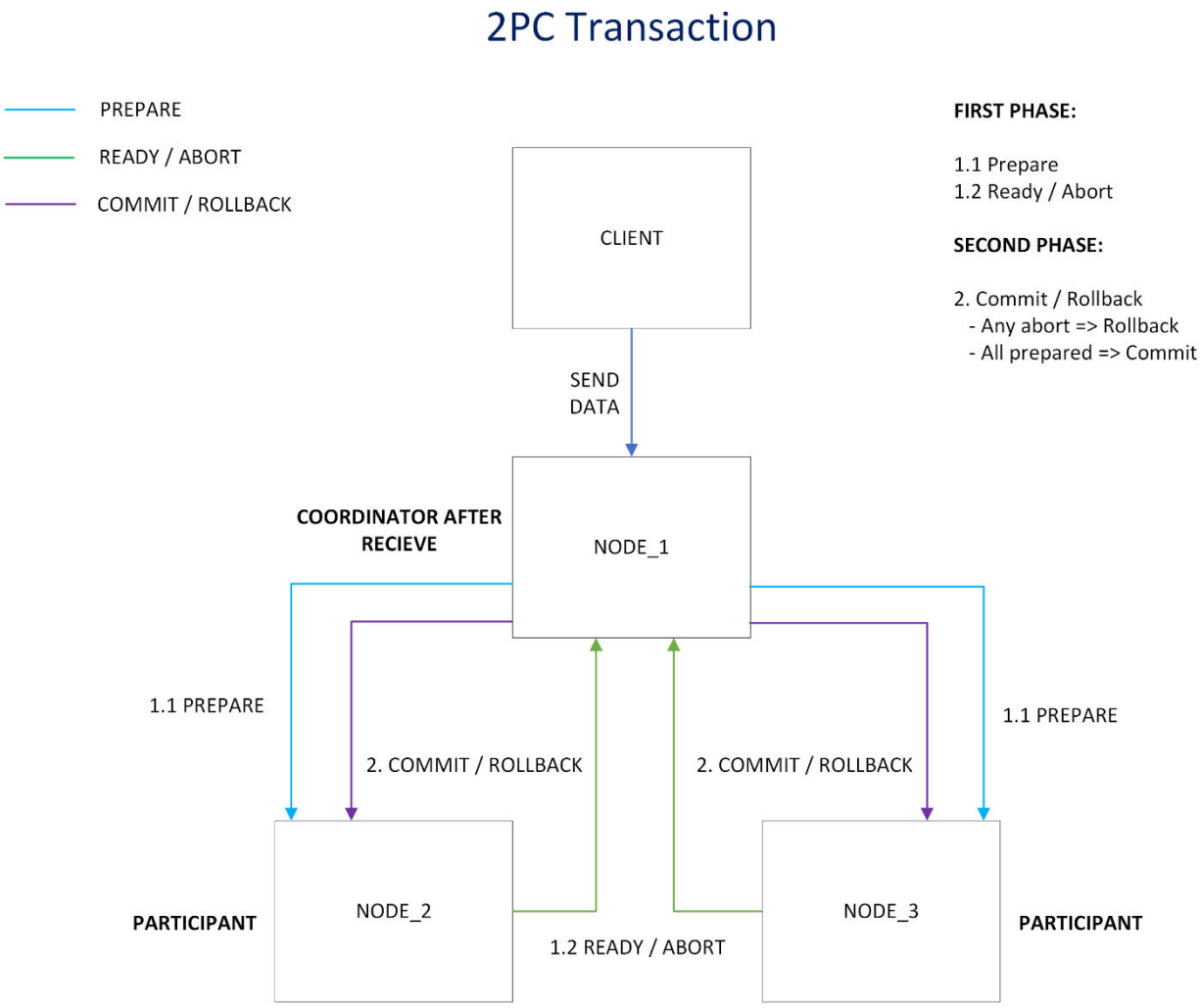
Ciljevi:

- Razviti odgovarajući protokol za komunikaciju između komponenti sistema
- Opravdana upotreba niti
- Opravdana upotreba struktura podataka za odgovarajući problem
- Implementacija *Proizvođač-Potrošač* šablona

Testiranje:

- Izmeriti propusnost sistema sa 1, 3 i 5 čvorova
- Provera curenja memorije
- Adekvatno rukovanje sa završetkom instance programa
- Zatvaranje konekcija

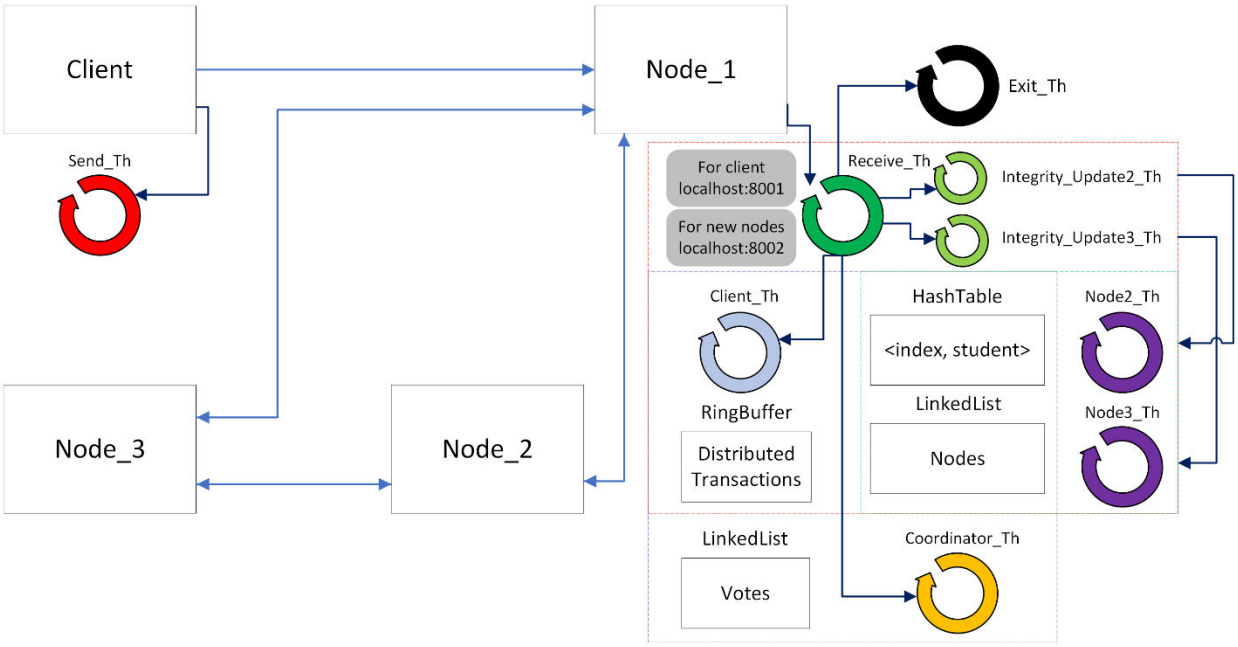
2. 2PC protokol



3. Dizajn

U sistemu su prepoznate sledeće komponente:

1. Klijent (*Client*)
2. Čvor/ovi (*Node*)



3.1 Upis studenta u distribuiranu bazu – 1 čvor

Posmatra se slučaj upotrebe kada imamo samo jedan čvor na mreži i klijenta koji šalje zahtev za upis studenta u distribuiranu bazu.

Nakon što klijent uspostavi konekciju sa željenim čvorom, za dalji tok komunikacije *Receive_Th* kreira posebnu nit - *Client_Th*. Klijent sada šalje studenta kojeg želi da upiše u distribuiranu bazu, a *Client_Th* prvo proverava da li student već postoji u strukturi za skladištenje studenata i razlikujemo dva slučaja:

1. Ukoliko se utvrdi da student sa tim indeksom **već postoji**, zahtev za unos studenta se **odbacuje**.

2. Ukoliko student **ne postoji**, pristupa se proveri strukture za skladištenje informacija o ostalim čvorovima na mreži. U našem slučaju gde imamo samo jedan čvor, ova struktura će biti prazna, što znači da nema potrebe za pokretanjem 2PC protokola. Umesto toga, student se upisuje u strukturu za skladištenje studenata.

3.2 Upis studenta u distribuiranu bazu – N čvorova

Posmatra se slučaj upotrebe kada imamo više čvorova na mreži i klijenta koji šalje zahtev za upis studenta u distribuiranu bazu.

Nakon što klijent uspostavi konekciju sa željenim čvorom, za dalji tok komunikacije *Receive_Th* kreira posebnu nit - *Client_Th*. Klijent sada šalje studenta kojeg želi da upiše u distribuiranu bazu, a *Client_Th* prvo proverava da li student već postoji u strukturi za skladištenje studenata i razlikujemo dva slučaja:

1. Ukoliko se utvrdi da student sa tim indeksom **već postoji**, zahtev za unos studenta se **odbacuje**.

2. Ukoliko student **ne postoji**, pristupa se proveri strukture za skladištenje informacija o ostalim čvorovima na mreži. Kako postoje drugi čvorovi na mreži, kreira se distribuirana transakcija koja se stavlja u kružni bafer. Distribuiranu transakciju iz kružnog bafera preuzima *Coordinator_Th* i šalje svim čvorovima na mreži. Sve dok ne dobije odgovor od svih čvorova, *Coordinator_Th* ne može preuzeti novu transakciju za obradu.

   NAPOMENA: Moguće je izvršavanje samo jedne transakcije istovremeno

Na drugoj strani, sa obzirom da svi čvorovi imaju posebnu nit u kojoj čekaju poruku od drugog čvora, nakon prijema poruke od koordinatora u n-toj niti provera se da li taj čvor može ili ne može da izvrši upis studenta. Ukoliko kod njega već postoji student sa tim indeksom ili iz nekoga razloga ne može da izvrši upis, odgovoriće koordinatoru sa *No* (0). U suprotnom odgovor će biti *Yes* (1). Nakon slanja odgovora koordinatoru, čeka se na odluku koju donosi koordinator (*Commit or Rollback*).

2.1. Svi odgovori koji su pristigli su **potvrdni** (1), koordinator zaključuje da treba da potvrdi transakciju i šalje *Commit* (1) svim čvorovima. Upisuje studenta u svoje lokalno skladište. Takođe, čvorovi koji čekaju na odgovor od koordinatora za potvrdu transakcije nakon pristiglog *Commit*-a upisuju studenta u svoje skladište.

2.2. Među odgovorima koji su pristigli, postoji barem jedan koji **nije potvrdan** (0), koordinator zaključuje da treba da poništi transakciju i šalje *Rollback* (0) svim čvorovima. Ne upisuje studenta u svoje lokalno skladište i čvorovi koji čekaju na potvrdu transakcije takođe ne upisuju studenta u svoja skladišta.

3.3 Novi čvor na mreži – Integrity Update

Prilikom pokretanja novog čvora unosi se port na kom čvor osluškuje zahteve klijenta i port na kom osluškuje zahteve novih čvorova koji su se pojavili na mreži. Takođe, neophodno je izabrati jednu od dve ponuđene opcije:

1. Prvi čvor na mreži – ako se izabere ova opcija, kako ne postoje drugi čvorovi, izvršiće se samo nophodne inicijalizacije struktura

2. Nije prvi čvor na mreži – neophodno je uneti portove svih čvorova na mreži. Na osnovu unetih portova, uspostaviće se konekcije i popuniti lista čvorova. Sa druge strane i čvorovi koji osluškuju zahteve novih čvorova, dodaće u svoje liste novi čvor i kreirati novu nit koja će biti zadužena za izvršavanje *Integrity Update*-a. *Integrity_Update_Th* čeka od novog čvora na mreži poruku koja će mu reći da li može da **oslobodi resurse i da se terminira** ili **treba da mu pošalje podatke o studentima**. Novi čvor uzima podatke o prvom čvoru koji je upisan u listi čvorova i šalje mu zahtev za podacima o studentima. Svim ostalim čvorovima šalje poruku da mogu da se terminiraju. Nakon prihvatanja podataka o studentima, kreira svoj lokalni *HashTable* koji popunjava sa dobijenim podacima i pokreće po jednu nit za svaki čvor na mreži.

Posle izvršenih svih inicijalizacija i *Integrity Update*-a, novi čvor na mreži se ponaša kao i svaki drugi regularan čvor.

3.4 Terminiranje čvora

Ukoliko je na nekom od čvorova pritisnuto **ESC**, *Exit_Th* će to registrovati kao komandu za terminiranje čvora. Prvo *Exit_Th* treba da obavesti *Receive_Th* o terminiranju, nakon toga *Receive_Th* obaveštava sve čvorove na mreži da više neće biti dostupan na mreži i da ga uklone iz svoje liste čvorova. Poslednji korak je oslobađanje svih zauzetih resursa i zatvaranje svih konekcija.

4. Strukture podataka

Kako bi rešenje bilo što optimalnije, korišćene su strukture podataka kojima postižemo brži pristup elementima, dodavanje proizvoljnog broja elemenata i implementaciju proizvođač-potrošač šablona.

4.1 Hash table

Za skladištenje studenata, najbolje rešenje je *Hash Table* struktura podataka. *Hash Table* nam omogućava da iskoristimo dobre osobine niza i jednostruko spregnute liste. U najboljem slučaju vremenska kompleksnost prilikom pristupa elementu je O(1), dok u najgorem je ona O(n). Pored dobrih performansi prilikom pristupa, možemo teoretski da dodamo neograničen broj studenata uvezivanjem u jednostruko spregnutu listu.

4.2 Singly linked list

Čvorovi na mreži i pristigli glasovi od čvorova čuvaju se u jednostruko spregnutoj listi. Jednostruko spregnuta lista nam omogućava da dodamo proizvoljan broj elemenata za vreme izvršavanja i da ih lako uvežemo. U ovom slučaju nije potreban brz direktan pristup pa je upotreba jednostruko spregnute liste opravdana.

4.3 Ring buffer

Za potrebe implementacije proizvođač-potrošač šablona neophodan nam je kružni bafer. Komunikacija između *Client_Th* i *Coordinator_Th* odvija se putem kružnog bafera. Nakon prihvata zahteva od klijenta i kreiranja transakcije, transakcija se dodaje u kružni bafer odakle je preuzima *Coordinator_Th* i počinje njena obrada.

Sa kružnim baferom izbegli smo potencijalne probleme sa curenjem memorije, zauzeli smo fiksni prostor i sve operacije se izvode u O(1).