# Replicator

U projektu je implementirano rešenje za mogućnost pokretanja procesa koji mogu da generišu podatke I pomoću asinhrone komunikacije sa replikatorom skladište ih na udaljenom serveru. Procesi takođe imaju mogućnost da se ugase I da nakon ponovnog uključenja u sinhronoj komunikaciji sa replikatorom dobave svoje podatke koje su prethodno poslali na replikaciju.

## Dizajn

Diagram

Description automatically generated

Slika

Sistem se sastoji iz 4 komponente. *Komponenta Process* je klijentska aplikacija koja se pri ukljucivanju javlja prvom replikatoru I time se registruje. Nakon registracije ima mogućnost slanja podataka na replikaciju ili slanje zahteva za dobavljanje podataka nakon čega pasivno čeka odgovor od replikatora.

Komponenta sa kojom Process neposredno komunicira je *komponenta Replicator1* . Replicator1 je zadužen za primanje zahteva od samih procesa. Radi u konkurentnom režimu I za svaki registrovani proces ima po jednu nit koja mu je na usluzi. Zahtevi koje ima sposobnost da obradi kada je u pitanju direktna komunikacija sa procesom su: registracija procesa, prijem podataka od procesa I zahtev za dobavljanje podataka sa udaljenog servera. Replicator1 takodje neposredno komunicira sa Replicatorom2. Za tu komunikaciju on koristi tri niti koje preuzimaju zahteve iz kružnog bafera I prosleđuju ih Replicatoru2. I još tri niti koje slušaju poruke sa dobavljenim podacima od Replicatora2.

Komponenta Replicator2 pored neposredne komunikacije sa Replicatorom1 ima I direktnu vezu sa komponentama tipa Replica. Replicator2 ima odgovornost da na zahtev Replicatora1 podigne novu instancu aplikacije Replica, a za svaku pokrenutu repliku imace po jednu nit koja služi isključivo za njihovu međusobnu komunikaciju. Slično kao kod prethodne komponente I Replicator2 ima tri niti, za prijem poruka od Replicatora1, koje primljene poruke smeštaju u kružni bafer. I još tri niti koje šalju odgovore Replicatoru1 iz kružnog bafera u kom se nalaze dobavljeni podaci.

Krajnja komponenta Sistema je Replica. Replica se pokreće od strane Replicatora2, nakon pokretanja javlja se Replicatoru2 koji kreira nit za datu repliku. Replika ceka poruke od Replicatora2 I u zavisnosti od toga koja je poruka, skladišti podatak u strukturu tipa liste ili dobavlja sve podatke iz liste I šalje ih u odgovoru Replicatoru2.

## Strukture podataka sistema

Text

Description automatically generated

Slika – Struct message

Strukturu *message* koriste procesi za slanje poruka replikatoru. Podaci koji se razmenjuju su tipa string maksimalne dužine 50. Polje *processId* postoji kako bi replikator mogao automatski da pošalje objekat strukture drugom replikatoru i kako bi on znao kojoj replici da prosleđuje podatak.

Text

Description automatically generated

Slika

Struktura *retrievedData* koristi se za vraćanje podataka od replike prema procesu. U polju tipa niz karaktera nalazi se string sa svim porukama iz liste povučene sa replike razdvojeni ’\n’ karakterom. Polje *processId* je potrebno da bi replikator znao kom procesu da vrati podatke. Polje *dataCount* je potrebno kako bi prijemna strana, odnosno proces znao koliko ima podataka u polju *data* i kako bi mogao da ih razdvoji i procita.

Text

Description automatically generated

Slika

*RingBuffer* struktura se koristi za bafer koji se koristi za skladištenje podataka na replikatorima primljenih od procesa. Polje *tail* pokazuje na kraj bafera, *head* pokazuje na početak bafera, polje *count* čuva broj elemenata u redu i polje *data* služi za samo čuvanje podataka tipa gore pomenute strukture message.

Text

Description automatically generated

Slika

*RingBufferRetrieved* struktura se koristi za bafer koji se koristi za skladištenje podataka na replikatorima primljenih od replika za slanje procesima. Polje *tail* pokazuje na kraj bafera, *head* pokazuje na početak bafera, polje *count* čuva broj elemenata u redu i polje *data* služi za samo čuvanje podataka tipa gore pomenute strukture retrievedData.

Text

Description automatically generated

Slika 6

*ThreadArgs* struktura služi za prosleđivanje potrebnih parametara funkcijama koje izvršavaju niti. Funkcije se koriste za *TCP* komunikaciju, stoga su im potrebni soket i adresa čvora sa kojim treba da uspostave komunikaciju. Funkcije takođe treba da čitaju i smeštaju podatke u bafere za slanje i prijem. Kako funkcije bafera koriste kritične sekcije, niti moraju da poseduju kriticnu sekciju koju će slati funkcijama bafera u trenutku poziva. Niz replika replikatoru2 služi da ne podiže novu repliku nakon ponovnog povezivanja starog procesa. I polje end je indikator da se program završava i da niti treba da napuste svoje funkcije.

Text

Description automatically generated

Slika

Struktura *listItem* se koristi za čuvanje podataka na replikama kao i kod komponente *Process*. Struktura liste je odabrana iz razloga što se lako dodaje na kraj i tokom rada sistema nije potrebno pristupati pojedinačno elementima liste već samo postoji preuzimanje celokupne liste.

Komponente *Replicator1* I *Replicator2* koriste strukturu podataka tipa kruzni bafer za komunikaciju izmedju niti. Kako bi jedna nit stavljala podatke u bafer koje druga nit treba da preuzima I obradi. Odabran je kružni bafer zbog uštede memorijskog prostora. Komponente *Process* I *Replica* koriste strukturu liste za jednostavno skladištenje podataka.

## Rezultati testiranja

A picture containing text

Description automatically generated

Slika 8

Prva dva *snapshot-a* u toku prvog testiranja zabeležena su na početku programa I nakon kreiranja lokalnih promenljivih. Treći *snapshot* napravljen je nakon kreiranja tri niti koja slušaju poruke od komponente Replicator2 I tri niti koje čitaju poruke za vraćanje iz bafera I šalju ih replikatoru2. Četvrti je snimljen nakon pokretanja sedam procesa koji se konektuju na Replicator1 nakon čega on kreira po jednu nit za svaki od procesa. U petom *snapshot*-u se vidi oslobađanje *heap*-a što se desilo kakon završetka svih pokrenutih niti I dodatno oslobađanje se dešava kada se pozove *CloseHandle*() funkcija nad svim prethodno kreiranim *heandle*-ovima.

A picture containing calendar

Description automatically generated

Slika 9

Prvi *snapshot* drugog testa zabeležen je na početku programa. Drugi je zabeležen nakon kreiranja 3 niti za prijem poruka i 3 niti za slanje poruka replikatoru2. Treći *snapshot* je snimljen nakon pokretanja dva procesa za koja su se kreirale dve niti. Zatim još jedan *snapshot* nakon pokretanja još dva procesa odnosno niti. Procesi su u ovom testu na svake dve sekunde slale poruku replikatorima. Posle završetka svih niti zabeležen je peti *snapshot* nakon završetka izvršenja niti. I poslednji nakon poziva *CloseHandle*() metode za sve *handle*-ove prethodno kreirane. Na kraju izvršenja programa na *heap*-u ostane zauzeto 17.88 KB.

Chart

Description automatically generated

Slika 10

Potencijalno unapređenje u sistemu moglo bi biti uvođenje vremenskog ograničenja čuvanja podataka na replikama kako ne bi došlo do prevelikog opterećenja nakon nekog vremena. Ukoliko se podaci ne dobave ni jednom u nekom predefinisanom vremenskom intervalu replika bi se samouništila a pre toga obavestila replikatora2 o svom gašenju.

Teodora Ruvčeski

Nebojša Gordić

14.01.2022.