RafBook

Petar Nesic

Arhitektura i koncept

Arhitektura bazirana na CHORD-u, gde su cvorovi podeseni u vidu prstena i svaki cvor ima znanje o svom prethodniku i sledbeniku. Svakom cvoru se dodeljuje jedinstveni identifikator koji se dobija tako sto se hesira njegova ip adresa i port. Pri ubacivanju i brisanju novog cvora azuriraju se vrednosti u CHORD strukturi samim tim prethodnici i sledbenici.

Podaci koji su nuzni da bi se napravio novi cvor se nalaze u konfiguracionoj klasi koja je jedinstvena za celu aplikaciju u njoj se sve osnovne informacije da bi se napravio jedan cvor, kao sto su port adresa, broj cvorova i korenski direktorijum odakle ce se citati fajlovi pri ubacivanju u chord.

Cvorovi su zaduzeni da cuvaju fajlove koji se ubacuju u prsten i dobijaju svog vlasnika po chord mehanizmu. Fajlovi se nalaze u direktorijumu u koji je definisan u konfiguracionoj datoteci. Svaki cvor moze da ima svoje prijatelje sto dozvoljava odredjene privilegije u sistemu.

Zbog konkurentnosti samog sistema iskoriscen je Suzuki-Kasami distribuirani mutex koji omogucava rad sa zajednickim resursima i stanjima.

Bootstrap komponenta koja je zaduzena da prihvati prvi cvor i kasnije dodeli nove cvorove u sistem. Bootstrap server se pokrece na sopstvenom portu i komunikacija je minimalna sa ostatkom programa.

Za prihvatanje komandi se koristi komponenta CLIParser koja je zaduzena da cita sa konzole, dok poruke prima SimpleServentListener gde su oba pokrenuta u zasebnim threadovima. Komunikacija medju cvorovima se vrsi preko poruka i njihovih hendlera.

Dodavanje prvog i novog cvora u sistem

Pokretanje BootstrapServera je nuzno da bi se ostali nodovi ukljucili u sistem.
Iz konfiguracione klase se citaju podaci i pokrece se inicijalizator koji salje poruku BootstrapServeru HAIL da bi signalizirao da zeli da udje sistem.
Ukoliko je prvi servent u sistemu bootstrap vraca -1, u suprotnom salje se NewNodeMessage, da bi se cvor prikljucio.
Ako nije nastala kolizija cvor se dodaje na odgovarajuce mesto i salje se WelcomeMessage za uspesno prikljucivanje.

Komande

- AddCommand add_file fileName.txt:public

-Dodavanje nove datoteke u sistem
-Poziva se chordState.putValue()
-putValue() proverava da li treba na tom cvoru da se smesti datoteka u suprotnom prosledjuje poruku AddMessage drugom cvoru.
-ukoliko je odgovarajuci cvor poziva se FileUtil gde se nalazi mutex koji zakljucava kriticnu sekciju i cita datoteku iz korenskog direktorijuma i smesta ga u Chord prsten.

- AddFriendCommand add_friend address:port

-Dodavanje prijatelja cvoru.
-Pokretanjem ove komande dodajemo sebe kao prijatelja cvoru sa navedenom adresom i portom.
-Svaki cvor ima listu prijatelju, zakljucavamo mutexom listu i dodajemo novog prijatelja.

- GetCommand get_file fileName.txt

-Dohvatanje odredjene datoteka sa chord prstena
-Poziva se chordState.getValue() gde se prosledjuje hesirani fajl
-Ukoliko trenutni cvor poseduje fajl vraca se u suprotnom prosledjuje se poruka GetMessage dalje
uz zakljucavanje mutexom

- RemoveCommand remove_file fileName.txt

-Brise datoteku iz sistema
-Poziva se chordState.getValue() gde se prosledjuje hesirani fajl
-Ukoliko postoji fajl lokalno brise direktno sa cvora u suprotnom prosledjuje RemoveMessage sledecem cvoru.
-Pri gasenju fajlovi se premestaju.

- RemoveServentCommand remove_servent

-Izbacivanje iz sistema trenutnog cvora.
-Cvor u kome je zadata komanda se brise iz sistema.
-Poziva se metoda chordState.removeNode()
-removeNode() salje poruke da se azuriraju sledbenik i prethodnik cvora koji treba da bude obrisan tako da pokazuju jedan na drugog, UpdatePredecessorMessage i UpdateSuccessorMessage.
-nakon toga se brise cvor iz sistema i azurira se tablica sledbenika.

- ViewFilesCommand view_files address:port

-Prikaz javnih i privatnih datoteka u zavisnosti od prijateljskih privilegija

-Iscitavaju se datoteke za odredjeni cvor tako sto se salje poruka ViewFileRequestMessage odgovor na tu poruku prikazuje listu naziva fajlova tako sto posalje ViewFileResponseMessage.

- ViewFriendsCommand view_friends

-Prikaz prijatelja cvora
-Prolazi kroz listu prijatelja cvora i stampa njihbe identifikatore

StopCommand stop
-Prekida rad cvora i izbacuje ga iz sistema
-Poziva gore pomenutu removeNode() metodu
-Prekida rad CLIParsera i SimpleServentListener-a.

*Postoji jos par pomocnih komandi oko pregleda lokalnih fajlova i replika koje se salju pri dodavanju novog fajla.

Suzuki Kasami Distribuirani Mutex
Svaki cvor u sistemu cuva podatak da li je vlasnik tokena.
Token se cuva u kao TokenMessage u svakom cvoru.
Koristi se rn mapa gde se nalazi broj ulaza u kriticnu sekciju za svaki cvor.
Inicijalno vlasnik tokena je prvi node koji udje u sistem.

SuzukiMutex:

.lock()
-Broadkastujemo poruku da nam treba kljuc
-Ceka se sve dok ne postane vlasnik kljuca tj. dok ne stigne na red.
-Ulazi se u kriticnu sekciju

.unlock()
-Izlazi se iz kriticne sekcije
-Pogledamo red da li neko ceka token.
-Broadkastujemo token poruku.
-Ako je neko trazio token dok smo bili u kriticnoj sekciji stavljamo njega na listu cekanja
-Salje se token dalje.

Prenos tokena po cvorovima se vrsi preko RequestTokenMessage, TokenMessage i njihovim handlerima.

Heartbeat sistem
Nije u potpunosti implementiran zbog roka, ali je blizu neke funkcionalnosti.
Funkcionise tako sto se u chordState-u salje poruka Heartbeat poruka u

odredjenim vremenskim intervalima koji su podesivi u kodu, kada se primi HeartbeatMessage cvor koji je primio treba da vrati AckMessage poruku ukoliko je ne vrati posle odredjenog intervala oznacava se kao da je mrtav.