RAFBook

Konkurentni i distribuirani sistemi

Arhitektura sistema

Arhitektura sistema temelji se na primeru sa vežbi, implementirajući Chord kao distribuirani sistem. Sve izvršene izmene su integrisane u osnovni Chord-ov sistem, što omogućava da sistem nastavi da radi po uobičajenim principima, ali sada sa dodatnim funkcionalnostima koje poboljšavaju njegovu fleksibilnost.

Distribuirani fer mutex

Mutex u ovom sistemu je implementiran po Suzuki-Kasami algoritmu sa određenim prilagođavanjima kako bi bio kompatibilan sa Chord distribuiranim sistemom. Sistem koristi jedan token, koji omogućava izvršavanje kritične sekcije onome ko ga poseduje. Ako čvor nema token, a želi da izvrši kritičnu sekciju, šalje REQUEST poruku svim susedima. Svaki čvor ima listu sekvencijalnih brojeva za svaki drugi čvor, što predstavlja redni broj poslednjeg zahteva koji je stigao od tog čvora. Kada čvor pošalje REQUEST poruku, povećava svoj lični brojač. Prijemom REQUEST poruke, čvor povećava brojač u svojoj listi za taj čvor.

Token sadrži red i listu koja označava ko je sledeći za izvršavanje kritične sekcije. Ova lista predstavlja poslednji završen zahtev za svaki čvor. Ako čvor koji poseduje token primi REQUEST poruku, povećava brojač u svojoj listi za taj čvor. Kada završi sa kritičnom sekcijom, proverava da li postoji čvor čiji je sekvencijalni broj za jedan veći od onog koji se prenosio u tokenu. Takvi čvorovi se dodaju u red čekanja.

Čvor koji je prvi u redu uzima se iz vrha, i njemu se šalje token sa redom i listom. Taj čvor zatim izvršava kritičnu sekciju, proverava i nastavlja proces dok se red ne isprazni. Ako čvor završi kritičnu sekciju i red je prazan, čeka na prijem REQUEST poruke kako bi poslao token tom čvoru i ažurirao sve potrebne brojače.

Pošto više niti unutar jednog čvora može tražiti pristup kritičnoj sekciji, dozvoljeno je samo jednoj u jednom trenutku da pristupi. Izuzetak od ovog pravila jeste kada se detektuje otkaz, potreban nam je token i ta nit ima prednost, odnosno ona ga odmah dobija, bez obzira na to da li ga neka druga nit trenutno koristi.

Dodavanje nove datoteke sa jedinstvenim nazivom i putanjom u sistem

Svaki korisnik(čvor) u svom ChordState-u sadrzi mapu Map<Integer, Map<String, MetaFile>> valueMap. Ova mapa kao ključ ima broj koji se dobija kada se Path od fajla hešira. Zbog kolizija koje se mogu dešavati u sistemu, vrednost koja se čuva u mapi jeste unutrašnja mapa Map<String, MetaFile> gde je ključ putanja do fajla, a MetaFile sadrži informacije da li je fajl privatan/javan, putanja i ko je njegov vlasnik. Prilikom dodavanja, traži se čvor (korisnik) koji je zadužen za čuvanje ključa tog podatka.

Ukoliko se doda novi čvor (korisnik) u sistem, on preuzima deo podataka za koje je prethodni čvor ranije bio zadužen, tako što u UPDATE poruci prosleđujemo sve nove fajlove, pa novi čvor može da vidi za koje je on zadužen.

U sistemu se prvo traži token, nakon čega se šalje PUT poruka, koja se prosledjuje narednom čvoru ukoliko ključ za koji treba da se doda fajl nije u vlasništvu tog čvora. Ukoliko jeste, onda se dodaje za taj ključ, vrednost koja se nalazi u poruci u valueMap tog čvora, i takođe se šalje svojim komšijama radi backup-a. I potom se vraća PUT_UNLOCK poruka čvoru koji je inicirao dodavanje.

Dodavanje čvora u listu prijatelja

U sistemu čvorovi mogu da se “pretplate” na neki drugi čvor u sistemu. Na taj način kada oni traže pristup nekom fajlu koji je privatan, oni će moći da mu pristupe. Ako je čvor A u listi prijatelja čvoru B, to ne znači da je i B u listi prijatelja čvoru A.

Dohvatanje proizvoljne datoteke iz distribuiranog sistema

Dohvatanje proizvoljne datoteke se obavlja tako što se u sistemu traži čvor (korisnik) koji je odgovoran za ključ podatka koji se traži. Priliko traženja nekog podatka, njegov path se hešira i dobija se ključ na kojem bi trebalo da se čuva taj podataka. Nakon pronalaženja ko je odgovoran za taj ključ, iz njegove valueMap-e se uzima vrednost i putem poruke se vraćaju vrednosti onom čvoru koji je inicirao dohvatanje. Prilikom dohvatanja vodi se računa o tome da li je podatak private ili public. Ukoliko je private onda se proverava da li je vlasnik tog fajla u listi ljudi na koje sam se pretplatio. Ukoliko jeste, može da se pročita fajl.

U sistemu se prvo traži token, nakon čega se šalje ASK_GET poruka čvoru. Ukoliko taj čvor nije zaduzen za ključ koji se traži onda se ova poruka prosledjuje sledecem čvoru. U trenutku kada se naidje na odgovarajuc čvor, vraca se TELL_GET poruka čvoru koji je inicirao dohvatanje. Nako prijema TELL_GET poruke, proverava se da li nam je vlasnik tog podatka prijatelj ili nije i da li je podatak private ili public.

Uklanjanje datoteke sa mreže

Brisanje datoteke se vrši tako što se pronadje čvor koji je odgovoran za ključ koji želimo da uklonimo. Kada se pronađe takav čvor, prvo proverimo da li je onaj ko je uputio zahtev za brisanje zapravo vlasnik fajla i ukoliko jeste mi je brišemo iz valueMap-e, ali takođe to kažemo i našim susedimo koji čuvaju backup.

U sistemu, inicijator šalje DELETE poruku čvoru pored sebe. Ako taj čvor nije odgovoran za taj ključ, poruka se prosleđuje sledećem čvoru koristeći eksponencijalni skok. Kada poruka stigne do čvora koji je odgovoran za taj ključ, on briše vrednost iz mape ukoliko je vlasnik ključa uputio zahtev. Zatim, taj čvor šalje DELETE_BACKUP poruku svojim susedima i vraća DELETE_UNLOCK poruku inicijatoru sa ishodom operacije.

Uredno gašenje čvora

Uredno gašenje čvora se vrši tako što čvor prvo zahteva od svih svojih suseda da dobije token. Nakon što dobije token, čvor šalje SHUTDOWN poruku svom nasledniku. Ova poruka se propagira kroz sistem kako bi svi čvorovi uklonili traženi čvor iz svoje tabele. Kada poruka stigne do prethodnika čvora koji se isključuje, to znači da je krug završen i prethodnik dobija token. Prethodnik tada šalje GOODBYE poruku čvoru koji želi da se isključi, nakon čega se taj čvor isključuje iz sistema.

Otpornost na otkaze

Svaki čvor ima u pozadini nit koja na svaku fiksnu, malu količinu vremena šalje PING poruku svom prethodniku, na koju on odgovara PONG porukom. Ukoliko prethodnik ne odgovori na tu poruku duže od vremena definisanog kao WEAK_LIMIT, šalje se broadcast poruku čvorovima da kontaktiraju taj čvor. Ukoliko neko uspe da stupi u kontakt s njim, obavestiće onoga ko je inicirao broadcast poruku, nakon čega se može resetovati vreme poslednjeg javljanja. Ako prođe duže od vremena definisanog kao STRONG_LIMIT, taj čvor se smatra mrtvim i šalje se broadcast poruka svim čvorovima u sistemu da ga izbace.