RafBook

Projekat

1 Pregled zadatka

Realizovati funkcionalan distribuirani sistem koji će da obezbedi rad sa ASCII kodiranim tekstualnim datotekama i slikama. Sistem korisniku omogućava sledeće:

- Dodavanje nove datoteke sa jedinstvenim nazivom i putanjom u sistem.
- Dohvatanje proizvoljne datoteke iz distribuiranog sistema.
- Dodavanje prijateljskih čvorova.
- Brisanje datoteke, na lokalnom serventu.
- Topološka organizacija sistema omogućava brže dohvatanje datoteka.
- Otpornost na otkaze.

Projekat može da se implementira u proizvoljnom programskom jeziku, dok god zadovoljava sve funkcionalne i nefunkcionalne zahteve. Dozvoljeno je koristiti distribuirani sistem sa vežbi kao polaznu tačku. Da bi bili dodeljeni bilo kakvi poeni, neophodno je da implementirane funkcionalnosti rade na distribuiranom sistemu.

Funkcionalni zahtevi za sistem su opisani u odeljku 2.

Nefunkcionalni zahtevi za sistem su opisani u odeljku 3.

Bodovanje zadatka, kao i instrukcije za predaju zadatka su dati u odeljku 4.

2 Funkcionalni zahtevi

2.1 Osnovne funkcionalnosti

RafBook sistem služi da obezbedi rad sa ASCII kodiranim tektualnim datotekama i slikama. Datoteke treba da budu organizovane u virtuelnoj strukturi datoteka, gde na svakom pojedinačnom čvoru može da bude prisutan proizvoljan podskup ove čitave strukture.
Interakcija sa sistemom se obavlja preko komandne linije (CLI). Virtuelni sistem datoteka treba da se čuva dok god postoji barem jedan aktivan čvor. Kada se poslednji čvor ugasi, time prestaje da postoji i virtuelni sistem datoteka, i ne treba da bude moguće rekonstruisati ga naknadno.

Osnovne funkcionalnosti za sistem uključuju:

- Dodavanje čvora u listu prijatelja
- Dodavanje nove datoteke u mrežu sa opcijom privatnog ili javnog deljenja.
- Pregled javnih i privatnih datoteka od prijatelja ili nekog drugog čvora.
- Uklanjanje datoteke sa mreže.

Na lokalnom sistemu treba da postoji direktorijum koji je na početku prazan, i koji se koristi pri radu sa sistemom:

- Radni koren - gde će se nalaziti datoteke sa kojima korisnik želi aktivno da radi.

2.2 Konfiguracija čvora

Pri pokretanju čvora, automatski se iščitava konfiguraciona datoteka u kojoj se navode sledeći atributi:

- Radni koren - putanja na lokalnom sistemu gde se skladište datoteke za rad. (string)
- Port na kojem će čvor da sluša. (short)
- IP adresa i port bootstrap čvora - odeljak 3.1. (string i short)
- Slaba granica otkaza - odeljak 3.2. (int)
- Jaka granica otkaza - odeljak 3.2. (int)

Pretpostavlja se da će svi čvorovi imati usklađene konfiguracione datoteke, i nema potrebe dodatno proveravati da li je to slučaj. Dozvoljeno je da sistem ne funkcioniše usled nepravilno podešene konfiguracione datoteke.

2.3 Komande

Korisnik može da zada sledeće komande sistemu:

- add_friend [adresa:port] - Dodavanje čvora u listu prijatelja
- add_file [path] [private/public] - Dodavanje nove datoteke u mrežu sa opcijom privatnog ili javnog deljenja.
- view_files [adresa:port] - Pregled javnih i privatnih datoteka od prijatelja ili nekog drugog čvora.
- remove_file [filename] - Uklanjanje datoteke sa mreže.
- stop - Uredno gašenje čvora.

Kod svih komandi se pri navođenju naziva datoteke očekuje putanja relativna u odnosu na radni koren, koji je naveden u konfiguracionoj datoteci. Nazivi datoteka nikada neće imati razmake, i nema potrebe podržavati ih.

3 Nefunkcionalni zahtevi

3.1 Arhitektura sistema

Dozvoljeno je da postoji bootstrap server, koji nije čvor u mreži (tj. sve napomene u ovom dokumentu koje se odnose na čvorove se ne odnose na bootstrap server). Za bootstrap važe sledeće pretpostavke:

- Koristi se isključivo za prvo uključivanje čvora u mrežu. Čim bootstrap prosledi novom čvoru adresu i port nekog čvora iz sistema, komunikacija sa bootstrap-om se prekida.
- Bootstrap server ima veoma ograničen protok. Komunikacija sa bootstrap serverom mora biti svedena na minimum.
- Nije dozvoljeno da bootstrap server bude svestan arhitekture sistema, te da on bude taj koji će je organizovati. Sistem mora da bude samoorganizujući.

Treba da bude moguće uključivati i isključivati čvorove u bilo kom trenutku rada sistema, uključujući dok se drugi čvorovi uključuju ili isključuju.

Postoje dve varijante za arhitekturu sistema koje se različito boduju:

- **Prost graf (100% poena)** - pošto graf nije kompletan, da bi čvor A prosledio poruku čvoru B, on mora da pronađe (ne nužno kompletnu) putanju kroz sistem do čvora B. Ovde je neophodno da broj skokova između A i B teži logaritamskoj zavisnosti od ukupnog broja čvorova. Ako broj skokova između proizvoljnih A i B teži linearnoj zavisnosti, implementacija se boduje kao da je rađen kompletan graf (50% poena). Da bi se graf računao kao prost, broj suseda za sve čvorove mora da ima logaritamsku zavisnost od ukupnog broja čvorova. Ne sme da postoji centralna tačka otkaza (čvor nakon čijeg stopiranja sistem prestaje da radi). Ne sme da postoji bottleneck - bottleneck za potrebe ovog projekta definišemo kao čvor (ili više čvorova kojima je fiksan broj) kroz koji komunikacija često teče. Ako postoje čvorovi kroz koje komunikacija često teče, ali njihov broj zavisi od broja čvorova u sistemu, tako da se komunikacija prirodno raspodeljuje među njima, onda oni nisu bottleneck. Startovanje novih čvorova, kao i stopiranje aktivnih čvorova može i treba da prouzrokuje restruktuiranje sistema.
  Primeri: Chord, Kademlia, Pastry, sopstvena struktura sistema
- **Kompletan graf (50% poena)** - svaki čvor je povezan sa svakim drugim čvorom. Komunikacija je uvek direktna.

3.2 Detektovanje otkaza

Otkazivanje čvora se detektuje u dve faze. Kao konstantan parametar sistema treba da postoje slaba granica otkaza (npr 4000) i jaka granica otkaza (npr 10000ms), obe date kao vreme koje se čvor ne javlja na ping.

Ako čvor prekorači slabu granicu otkaza, markiramo ga kao sumnjivog, ali ne započinjemo uklanjanje čvora i restrkutiranje sistema, već tražimo nekom drugom stabilnom čvoru da nam on potvrdi da je sumnjivi čvor zaista problematičan. Ako dobijemo potvrdu da je čvor sumnjiv, i nakon toga istekne jaka granica otkaza, čvor se eliminiše iz sistema i započinje se restruktuiranje.

Kada čvor otkaže, njegov dotadašnji posao ne sme da se izgubi. Ako postoji slobodan čvor (jedan on njegovih buddy-a), on treba da preuzme podatke za koje je bio zaduzen cvor koji je otkazao.

Sistem treba da bude sposoban da se izbori sa “worst case” situacijom u kojoj inteligentni maliciozni napadač može da probere i simultano obori bilo koja dva čvora na svakih pet minuta.

3.3 Raspored podataka u sistemu

Datoteke se nalaze na onom čvoru na kom su napravljene. I mora da postoji negde kopija te datoteke. U suštini, neki back up. Kod implementacije Chord-a mogu da ostanu gde bi bilo po Chord-u.

3.4 Opšti nefunkcionalni zahtevi

Sistem mora da funkcioniše na pravoj mreži, gde svaka poruka ima proizvoljno kašnjenje i kanali nisu FIFO. Ako se sistem testira na jednoj mašini, neophodno je uvesti veštačka kašnjenja pri slanju poruka, radi realističnog testiranja.

Sva komunikacija mora da bude eksplicitno definisana i dokumentovana. Ako čvoru stigne poruka koja nije po protokolu, treba je odbaciti i ignorisati. Dokumentacija protokola minimalno treba da sadrži sve što je neophodno da se napiše novi servent koji će da učestvuje u radu sistema. Tipično, to je spisak svih poruka koje postoje u sistemu i njihov format – redosled vrednosti koje se prosleđuju, njihov tip i njihovo značenje. Ako postoji neki specifičan redosled slanja poruka, onda navesti i to. Primer dokumentacije.

4 Predaja zadatka

4.1 Način predaje zadatka za studente koji prvi put slušaju predmet

Zadatak se predaje slanjem putem Git Classroom-a.

Rok za predaju svih grupa je 7.6.2024. i Github Classroom link je:
https://classroom.github.com/a/_usaz5p7

Ako nemate nalog na GitHub-u, morate ga napraviti. Ako već imate nalog, prijavite se na GitHub sa Vašim postojećim nalogom.

Nakon što prihvatite zadatak, GitHub će automatski kreirati repozitorijum za Vas sa odgovarajućim imenom. Za početnu tačku možete uzeti kod sa vežbi koje vam najviše odgovaraju.

Da biste radili na zadatku, morate klonirati repozitorijum na svoj lokalni računar. To možete uraditi koristeći komandu git clone u terminalu sa linkom do Vašeg repozitorijuma.

Nakon što klonirate repozitorijum, možete početi da radite na zadatku lokalno na svom računaru.

4.2 Način predaje zadatka za ponovce

Zadatak se predaje putem mail-a na ml_jovanovic@raf.rs. Java projekat imenovati na sledeći način: “kids_pr_ime_prezime_ind”. Npr. “kids_pr_student_studentic_rn0101”.

Arhivirati ovaj direktorijum (.zip), okačiti na svoj drive i u mail-u poslati link ka arhivi, pošto će Google mail verovatno blokirati direktan attachment.

U tekstu mail-a obavezno navesti:

- Ime i prezime
- Broj indeksa
- Programski jezik (java / go) i razvojno okruženje (Eclipse / Intellij)

Subject mail-a mora da bude u obliku: “[KiDS 2024] PR ime_prezime_ind”.

Npr. “[KiDS 2024] PR student_studentic_rn0101”

Naziv arhive mora da bude u obliku: “kids_pr_ime_prezime_ind.zip”

Npr. “kids_pr_student_studentic_rn0101.zip”

4.3 Grupni rad

Rad u grupi nije obavezan. Grupa se sastoji od tačno tri studenta. Grupni zadatak se sastoji od toga da, pored normalnog rada, sve tri implementacije mogu zajedno da čine sistem koji radi. To jest, neophodno je da svi čvorovi koriste identičan protokol. Pored toga, zahtev je da se koriste različiti programski jezici. Ako niste sasvim sigurni da li se vaši jezici računaju kao različiti, obavezno se konsultujte sa asistentom unapred. Na primer, C i C++ nisu različiti jezici, ali C i C# jesu. Spring nije jezik, tako da ne može da se koristi pored čiste Java implementacije, itd. Dozvoljeno je da postoji samo jedna implementacija bootstrap servera, kao i jedan dokument koji opisuje protokol za čitavu grupu.

Za uspešno završen grupni zadatak svi studenti u grupi dobijaju po dodatnih 10 poena. Svi članovi grupe moraju da imaju kompletiran grupni zadatak da bi bilo ko dobio dodatne poene. Dodatni poeni se dodeljuju do maksimuma od 70 na predispitnim obavezama.

4.4 Odbrana i bodovanje

Odbrana projekta je obavezna. Termin za odbranu projekta će biti u toku druge kolokvijumske nedelje. Tačan termin odbrane za svakog studenta će biti objavljen ubrzo nakon isteka rokova za predaju projekta. Ako ste iz bilo kog razloga sprečeni da prisustvujete odbrani, obavezno to najavite što pre, kako bismo mogli da zakažemo vanredni termin za odbranu.

Svrha odbrane je da se pokaže autentičnost zadatka. Ovo podrazumeva odgovaranje na pitanja u vezi načina izrade zadatka, ili izvršavanje neke izmene nad zadatkom na licu mesta. U slučaju da odbrana nije uspešna, dodeljuje se -40 poena na projektnom zadatku, odnosno nećete imati mogućnost da izadjete na ispit.

VAŽNO: Studiranje i istraživanje algoritama kroz korišćenje koda pronađenog na internetu je dozvoljeno i može vam pomoći u boljem razumevanju materije. Međutim, ukoliko više studenata koristi identičan izvorni kod pronađen na internetu, to će biti smatrano kao korišćenje istog koda i svim uključenim studentima će biti dodeljeni negativni poeni. Ista pravila važe i za korišćenje algoritama generisanih veštačkom inteligencijom.
Cilj domaćih zadataka na akademskim studijama je samostalno istraživanje i sticanje znanja bez oslanjanja na rešenja sa interneta ili rešenja generisanih od strane veštačke inteligencije.

Zadatak se boduje na sledeći način:

- Osnovna funkcionalnost (2, 3.1): 15 poena
    - Dodavanje nove datotetke: 5 poena
    - Dohvatanje neke datoteke: 5 poena
    - Brisanje neke datoteke: 5 poena
- Otpornost na otkaze (primeri: ping-pong, heartbeat): 15 poena
    - Parcijalni poeni za detektovanje otkaza koje generalno radi, ali u posebnim slučajevima ne funkcioniše i slično.
    - Detektovanje otkaza: 10 poena
    - Održavanje back up sistema: 5 poena
- Distribuirani fer* mutex (fer mutex primeri: Suzuki-Kasami): 10 poena
    - Ako mutex nije fer: 5 poena
- Grupni rad (4.2): 10 poena
- Nema dokumentacije (3.3): -5 poena
    - Parcijalni negativni poeni ako dokumentacija nije dovoljno detaljna.

Zadatak je moguće raditi parcijalno, ali obavezno je da se kompajluje i da može da se pokrene kao distribuiran sistem, kao i da je moguće pokazati da implementirana stavka ispunjava funkcionalne i nefunkcionalne zahteve. Bez jasnog pokazatelja (pokretanja, ispisa, testa, i sl.) da je stavka implementirana kao što je traženo, neće se bodovati.

* Fer mutex obezbedjuje da čvorovi dobiju pristup mutex u poretku u kom su ga zatražili bez nasumičnosti.