SillyGit

Projekat

1 Pregled zadatka

Realizovati funkcionalan distribuirani sistem koji će da obezbedi verzioniranje za ASCII kodirane tekstualne datoteke. Sistem korisniku omogućava sledeće:

- Dodavanje nove datoteke sa jedinstvenim nazivom i putanjom u sistem.
- Dohvatanje proizvoljne datoteke iz distribuiranog sistema.
- Održavanje verzija za datoteke i razrešavanje konflikta.
- Topološka organizacija sistema omogućava brže dohvatanje bitnih 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

SillyGit sistem služi da obezbedi verzioniranje za ASCII kodirane tekstualne datoteke. 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:

- Add - Dodavanje nove datoteke u sistem.
- Pull - Dohvatanje datoteke koja trenutno nije prisutna na čvoru.
- Commit - Upisivanje izmenjene datoteke u sistem.
- Remove - Uklanjanje datoteke iz sistema.

Na lokalnom sistemu treba da postoje dva direktorijuma koji su na početku prazni, i koji se koriste pri radu sa sistemom:

- Radni koren - gde će se nalaziti datoteke sa kojima korisnik želi aktivno da radi.
- Skladište - gde će se nalaziti datoteke koje su skladištene na ovom čvoru, na osnovu topologije sistema. Korisnik ne sme da radi direktno sa ovim datotekama.

2.2 Konfiguracija čvora

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

- Radni koren - putanja na lokalnom sistemu gde se skladište datoteke za rad. (string)
- Skladište - putanja na lokalnom sistemu gde su smeštene datoteke za koje je ovaj čvor topološki zadužen. (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)
- Tim kojem pripada ovaj čvor - odeljak 3.3. ([a-z] string)
- Limit - granica za aktiviranje timske optimizacije - odeljak 3.3. (short)

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 name`** - Dodaje datoteku `name` u sistem. Kao `name` može da se navede i direktorijum, u kom slučaju se cela njegova sadržina rekurzivno dodaje u sistem. Ako virtuelni sistem već sadrži datoteku sa ovim nazivom na ovoj putanji, prijaviti grešku i ne dodati datoteku u sistem. Datoteka dobija verziju 0 pri dodavanju u sistem.
- **`pull name [version]`** - Dohvata datoteku `name` iz sistema, u verziji `version`. Verzija može da se izostavi, i onda se dohvata aktuelna verzija datoteke. Naziv datoteke se navodi kao puna putanja do datoteke i njeno ime. Kao `name` može da se navede i direktorijum, u kojem slučaju se vrši commit operacija rekurzivno za čitav direktorijum.
- **`commit name`** - Osvežava datoteku `name` u sistemu za verzioniranje i povećava verziju. Sve datoteke kreću sa verzijom 0, i pri commit operaciji se verzija datoteke inkrementira. Kao `name` može da se navede i direktorijum, u kojem slučaju se vrši commit operacija rekurzivno za čitav direktorijum. Ako datoteka nije menjana od prethodne verzije, ne treba joj menjati verziju pri commit-ovanju. Ako je datoteka već ažurirana od strane nekog drugog čvora, tj. došlo je do konflikta, pitati korisnika kako želi da razreši konflikt (odeljak 2.4).
- **`remove name`** - Uklanja datoteku `name` iz sistema. Kao `name` može da se navede i direktorijum, u kojem slučaju se vrši remove operacija rekurzivno za čitav direktorijum.
- **`quit`** - 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.

2.4 Razrešavanje konflikta

Ako se pri commit operaciji desi da je datoteka već ažurirana, a da trenutni čvor toga nije bio svestan, neophodno je da se razreši taj konflikt. Korisnik treba da može da bira između sledećih opcija:

- **`view`** - Dohvatanje datoteke iz sistema pod privremenim nazivom, kako bi se ispitao njen sadržaj. Nakon ove operacije se korisnik ponovo pita za opciju razrešavanja konflikta.
- **`push`** - Prepisivanje datoteke u sistemu sa lokalnom datotekom.

- **`pull`** - Prepisivanje lokalne datoteke sa datotekom iz sistema.

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.
- 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 1000ms) 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, on treba da preuzme podatke za koje je bio zaduzen cvor koji je otkazao. Ako ne postoji slobodan čvor, neophodno je da se sistem restruktuira tako da svaki deo fajl strukture bude prisutan na nekom cvoru.

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

Neophodno je da kompletan virtuelni sistem datoteka ne bude centralizovan, već da se datoteke prirodno distribuiraju po čitavom sistemu. Ova distribucija treba da teži da bude uniformna po čitavom sistemu, uključujući redundantne podatke koji treba da obezbede podnošenje otkaza. Skladištenje datoteka na ovaj način ne treba da bude vidljivo korisniku, tj. ove datoteke se neće nalaziti unutar lokalnog radnog korena, već unutar lokalnog skladišta, sa kojim korisnik ne treba direktno da interaguje.

Pored toga, treba da postoji optimizacija za dohvatanje datoteka koje su relevantne za tim:

- Sistem treba da prati koliko članova svakog tima trenutno radi sa nekom datotekom (tj. koliko njih je uradilo `pull` za datoteku koju posmatramo).
- Ako broj članova nekog tima koji rade sa datotekom pređe granicu zadatu u konfiguracionoj datoteci (Limit), neophodno je da se napravi dodatna kopija ove datoteke na nekom čvoru unutar tima, ako to već nije slučaj. Svi čvorovi unutar tima ubuduće treba da primarno rade sa ovom kopijom, a samo čvor na kojem se nalazi ta kopija treba da osvežava globalnu kopiju datoteke, gde god se ona nalazila.
- Čvorovi koji pripadaju istom timu treba da budu topološki blizu jedan drugom, kako bi se obezbedila efikasna komunikacija unutar tima.

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. Čak i ako se testiranje vrši na jednoj mašini, nije dozvoljeno fiksirati IP adresu odredišta pri slanju poruke kao “localhost”.

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 1 - Distribuirane kraljice (projekat iz 2018/2019)
    - Dokumentacija
    - Tekst zadatka
- Primer 2 - Distribuirani haos (projekat iz 2019/2020)
    - Dokumentacija
    - Tekst zadatka

4 Predaja zadatka

4.1 Način predaje zadatka

Zadatak se predaje putem mail-a na bmilojkovic@raf.rs. Java projekat imenovati na sledeći način: “kids_projekat_ime_prezime_ind”. Npr. “kids_proj_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
- Grupa, po zvaničnom spisku
- Programski jezik i razvojno okruženje

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

Npr. “[KiDS] PROJ student_studentic_rn0101”

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

Npr. “kids_proj_student_studentic_rn0101.zip”

Rok za predaju je:

- Subota, 05. jun 23:59:59 za sve studente.

4.2 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 15 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. Grupni zadatak se prihvata uz:

- Kompletno odrađen zadatak.
- Zadatak bez otpornosti na otkaze.
- Zadatak bez timske optimizacije.

4.3 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. Zadatak se boduje na sledeći način:

- Osnovna funkcionalnost (2, 3.1) - **15 poena**
    - Osnovni rad sa datotekama - *5 poena*
    - Verzioniranje - *5 poena*
    - Razrešavanje konflikta - *5 poena*
- Otpornost na otkaze - **15 poena**
    - Parcijalni poeni za detektovanje otkaza koje generalno radi, ali u posebnim slučajevima ne funkcioniše i slično.
- Timska optimizacija - **10 poena**
    - Parcijalni poeni za korišćenje globalne kopije umesto timske i slično.
- Grupni rad (4.2) - 15 poena
- Nema dokumentacije (3.3) - -10 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.