Distribuirane kraljice

Projekat

1 Pregled zadatka

Realizovati funkcionalan distribuirani sistem koji će da rešava problem n kraljica. Sistem korisniku omogućava sledeće:

- Pokretanje izračunavanja za problem proizvoljne veličine.
- Distribuirano izračunavanje za zadati problem.
- Prikaz rešenja na svim čvorovima.
  - Treba pronaći sva rešenja za zadatu dimenziju table.
- Pauziranje izračunavanja.
- Startovanje više izračunavanja paralelno.
  - U jednom trenutku je aktivno samo jedno izračunavanje, ostala su pauzirana.
- Prikaz napretka pokrenutih izračunavanja.
- Proizvoljno uključivanje i isključivanje čvorova, bez otpornosti na otkaze.
- Svi čvorovi su konstantno opterećeni pomoću work stealing pristupa.

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

Posao koji čvorovi obavljaju je ispitivanje da li se N kraljica može rasporediti na šahovskoj tabli dimenzija N x N tako da se sve međusobno ne napadaju. Neophodno je pronaći sve ovakve rasporede za zadato N.

Ovaj problem treba rešavati brute-force pristupom, tj. ispitivanjem svih mogućih opcija. Dozvoljeno je koristiti bilo koji algoritam za pretragu - rekurzivni ili iterativni.

2.1 Konfiguracija čvora

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

- Port na kojem će čvor da sluša.
- IP adresa i port bootstrap čvora (odeljak 3.1).
- Limit - procenat posla ispod kojeg krađa posla nije dozvoljena (odeljak 3.3).

2.2 Komande i izveštaji

Korisnik može da zada sledeće komande sistemu:

- **`status`** - Prikazuje stanje svih započetih izračunavanja, i naznačava koje je trenutno aktivno, ako ga ima. Moguća stanja su: **`active, paused, done, fuzzy`**. Samo jedno izračunavanje može da bude **`active`** u jednom trenutku, dok sva ostala moraju da budu **`paused`** ili **`done`**. Za izračunavanja koja nisu završena, ispisuje koliko mogućih pozicija je ispitano do sada, izraženo u procentima. Ova komanda treba da pita sve čvorove za stanje poslova. Ako za neki posao postoji nesuglasica kod makar dva čvora, onda ga treba obeležiti kao **`fuzzy`**.
- **`start X`** - Započinje izračunavanje za zadati parametar X (X predstavlja dimenziju table, kao i broj kraljica koje treba rasporediti). Ova komanda je dozvoljena samo ako nije već započeto izračunavanje za parametar X. Ako je izračunavanje za parametar X prethodno startovano, treba prijaviti grešku. Ako je trenutno aktivno izračunavanje za neki parametar Y, koji je različit od X, treba ga pauzirati i započeti izračunavanje za X. Pri startovanju posla, problem se ravnomerno raspodeljuje svim čvorovima u sistemu.
- **`pause`** - Pauzira trenutno aktivno izračunavanje. Ako nema aktivnih izračunavanja, prijaviti grešku. Nakon izvršavanja ove komande, ni jedno izračunavanje ne treba da bude aktivno.
- **`result X`** - Prikazuje rezultate za završeno izračunavanje za parametar X. Ako izračunavanje za X nije započeto, ili nije završeno, prijaviti grešku. Rezultat prikazati u obliku matrice gde '_' naznačava poziciju na kojoj nema kraljice, a 'Q' naznačava poziciju na kojoj ima kraljice. Pri izvršavanju ove komande se ne šalju nikakve poruke.
- **`stop`** - Zaustavlja rad čvora. Neophodno je da se pretraga koju je ovaj čvor obavio ne ponovi. Takođe je neophodno da neki drugi čvor nastavi tamo gde je ovaj čvor stao. Neophodno je da postoji jedan čvor kojem se dodeljuje naš preostali posao, a ne da se pomoću broadcast-a posao dodeli na više čvorova. Ako je broadcast neophodan za restruktuiranje sistema, onda je dozvoljen, samo u tu svrhu. Gašenje čvora pomoću ove komande, kao i paljenje novog čvora može da se desi bilo kada tokom rada 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 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.

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, tokom kojeg izračunavanje može da bude privremeno prekinuto.
- Kompletan graf (50% poena) - svaki čvor je povezan sa svakim drugim čvorom. Komunikacija je uvek direktna.

3.2 Međusobno isključivanje

Za potrebe izvršavanja komandi: **`start`** i **`pause`**, koristiti međusobno isključivanje. Dakle, u jednom trenutku će samo jedna instanca ove komande biti aktivna na nivou čitavog sistema. Ako korisnik pokuša da izvrši dve ili više ovih komandi konkurentno, njihovo izvršavanje treba da bude serijalizovano pomoću algoritma za međusobno isključivanje. Moguće je i da će korisnik da zada više komandi na jednom čvoru veoma brzo jednu za drugom, kao i da ih zada istovremeno na različitim čvorovima. Međusobno isključivanje treba da bude implementirano pomoću tokena, i to po Suzuki-Kasami Broadcast algoritmu (poglavlje 14.11 u udžbeniku). Dozvoljeno je uzeti kritičnu sekciju i pri startovanju novog čvora.

Kada korisnik zada jednu od ovih komandi, čvor treba prvo da uzme pravo na kritičnu sekciju, potom da obavesti sve ostale čvorove u sistemu o izvršavanju ove komande, i kada dobije potvrdu od svih čvorova da su primili tu informaciju, i ažurirali svoje stanje, da oslobodi kritičnu sekciju. Konzola na čvoru koji ima kritičnu sekciju ne sme da bude blokirana za vreme kritične sekcije. Komande **`status`** i **`result`** treba da funkcionišu normalno na svim čvorovima za vreme izvršavanja neke od komandi **`start`** ili **`pause`**.

3.3 Krađa poslova

Krađa se dešava kada se čvor startuje dok je neki posao aktivan, kao i kada čvor završi svoj deo pretrage pre ostalih. Moguće je da je broj čvorova u sistemu značajno veći od parametra X zadatog pri startovanju problema. Čvorovi treba da dele posao pri startovanju posla, kao i pri krađi, što je više moguće ravnomerno.

Kada čvor završi sa svojim delom posla, treba da nastavi sa radom tako što nekom drugom čvoru uzme približno polovinu njegovog preostalog opsega za pretragu. Izbor čvora od kojeg ćemo da krademo treba da bude takav da se različite krađe ravnomerno rasporede po sistemu, tj. da imamo veliku verovatnoću da neće da se desi da često pokušavamo da krademo posao iz jednog ograničenog dela sistema. Ako čvor kojeg pitamo za posao ima da pretražuje još manje od **`limit`** % do kraja posla, onda treba pitati neki drugi čvor. Krađa poslova ne sme da se realizuje kao broadcast.

3.4 Prikupljanje rezultata

Kada čvor nema više posla, i ne može ni od koga da preuzme još posla jer su ispod limita, taj čvor je završio sa radom i broadcastuje svoj rezultat svim ostalim čvorovima. Kada neki čvor primi rezultat rada od svih ostalih čvorova u sistemu, on to prijavljuje korisniku (bez ispisa rezultata).

3.5 Opšti nefunkcionalni zahtevi

Čvorovi neće spontano otkazivati. Ako bilo koji čvor spontano otkaže ili izgubi konekciju, sistem može da se ponaša na nepredvidiv način.

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 - Distribuirani haos (projekat iz 2016/2017)
  - Dokumentacija
  - Tekst zadatka
- Primer 2 - Distribuirani video stream (projekat iz 2017/2018)
  - Dokumentacija
  - Tekst zadatka

3.6 Greške

Probleme koji nisu navedeni u tabeli nema potrebe rešavati.

R. Br.	Problem	Rešenje
1.	Korisnik startuje posao za parametar X koji je već startovan.	Prijaviti grešku.
2.	Korisnik pokušava da pauzira izračunavanje, a nema aktivnog izračunavanja.	Prijaviti grešku.
3.	Korisnik pokušava da prikaže rezultat za izračunavanje koje nije započeto ili nije završeno.	Prijaviti grešku.

Svi navedeni problemi treba da se reše graciozno, tj. da sistem nastavi normalno da funkcioniše.

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_projekat_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

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

Npr. “[KiDS] PROJEKAT student_studentic_rn0101”

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

Npr. “kids_projekat_student_studentic_rn0101.zip”

Rok za predaju je:

- Petak, 07. jun 23:59:59 za sve studente.

Od ukupnog broja poena dobijenih na zadatku će biti oduzeto 5 poena ako se desi bilo koje od:

- Sadržaj mail-a nije po navedenom obliku.
- Subject mail-a nije po navedenom obliku.
- Naziv projekta nije po navedenom obliku.
- Naziv arhive nije po navedenom obliku.
- Predaja se desi nakon navedenog roka.

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 različite implementacione platforme (programski jezici). 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 odrađen bez međusobnog isključivnaja (3.2).
- Zadatak odrađen bez krađe poslova (3.3).

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 u subotu, 08. juna. 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 / 3.4) - 10 poena
- Međusobno isključivanje (3.2) - 15 poena
- Krađa poslova (3.3) - 15 poena
- Grupni rad (4.2) - 15 poena
- Nema dokumentacije (3.5) - -10 poena
- Greške nisu razrešene (3.6) - -5 poena

- Razmatra se samo ako su stavke koje mogu da proizvedu grešku implementirane.

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.