SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Završni rad br. 2580

Sustav za raspodijeljeno upravljanje korisničkim poslovima

Stjepan Antivo Ivica

Voditelj: *Prof. dr. sc.. Domagoj Jakobović*

Zagreb, lipanj, 2012

Sadržaj

1. Uvod

Tema ovog rada jest osmisliti i izgraditi programsku potporu za raspodijeljeno upravljanje korisničkim poslovima.

Postoje korisnički poslovi koje korisnik želi istovremeno pokrenuti na više računala. Kako bi korisnik pokrenuo nekakav posao na proizvoljnom broju računala, trebao bi samostalno na svakom računalu pripremiti taj posao tako da se može izvesti i potom ga pokrenuti. Zatim, postoje korisnički poslovi koji su takvi da se trebaju obavljati paralelno na više računala, a prije nego što bi mogao pokrenuti taj posao korisnik bi također morao pripremiti podatke na svakom računalu i pokrenuti ga. Naravno uza spomenute probleme postoje i poslovi koje korisnik želi pokrenuti više puta, a na raspolaganju mu je više računala.

Potrebno je načiniti sustav koji bi primao slijedne i paralelne poslove od strane više korisnika, izvršio zadane poslove te odgovarajućim korisnicima vratio rezultate posla kojeg su zadali.

Pojmom grozd računala označavamo skup računalnih resursa međusobno integriranih u cjelinu [1]. Računala unutar grozda su danas najčešće umrežena brzom lokalnom mrežom. Uz visok stupanj povezanosti i koodrinacije rada među komponentama grozda ostvaruje se snažan računalni sustav.

Grozd računala na kojemu će se primjenjivati sustav izrađen u okviru ovog završnog rada sastoji se od neodređenog broja nezavisnih računala koji su za potrebe odrade paralelnog zadatka u mogućnosti odraditi međusobnu komunikaciju i raspodijeliti posao među čvorovima tog grozda.

Standard kojim se može ostvariti povezanost među računalima u svrhu izvedbe paralelizacije jest *Message Passing Interface*. Bitno je naglasiti da je MPI sučelje , a ne konkretna implementacija. Njegova dužnost je osigurati virtualnu topologiju, sinkronizaciju i komunikaciju između niza procesa gdje postoji izravno preslikavanje između čvorova i procesa. Bitno ograničenje standarda je to što se količina procesa u jednom MPI programu ne može mijenjati za vrijeme svog izvođenja tj. nije

predviđena mogućnost stvaranja novog procesa kao ni isključivanja nekog od postojećih (ipak u implementaciji standarda MPICH2 ovo je omogućeno, no to ne koristimo).

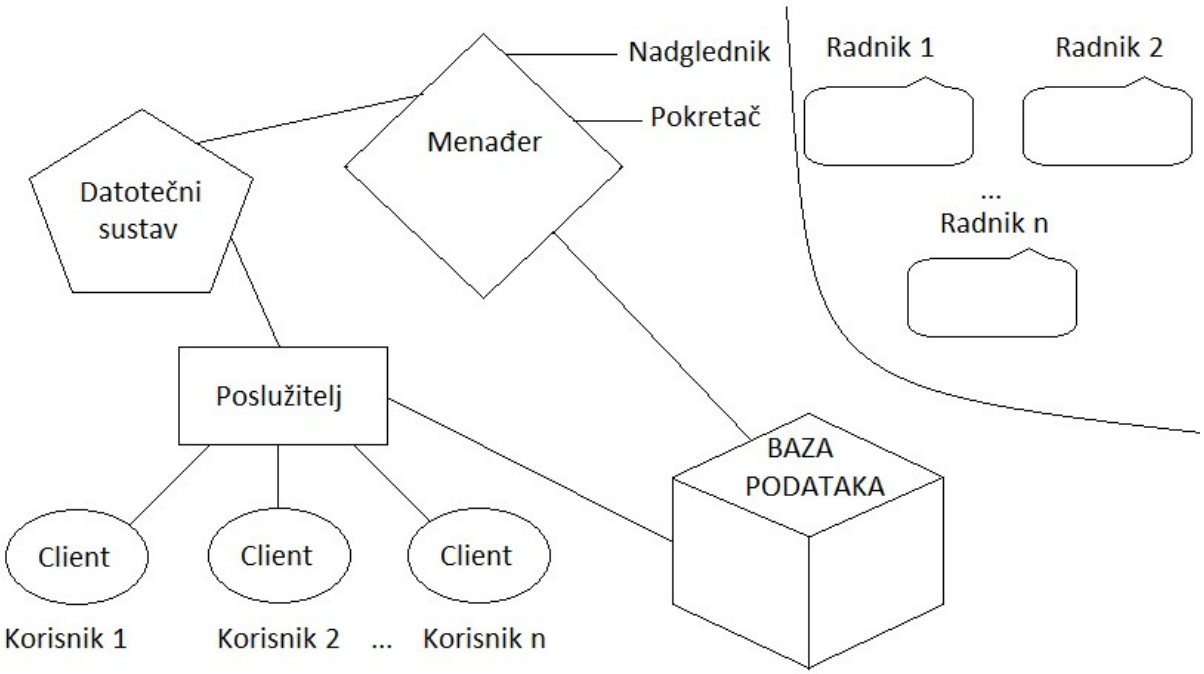
2. Osnovni pregled arhitekture sustava

Izgrađeni sustav možemo podijeliti na četiri bitna dijela. To su klijent, poslužitelj, baza podataka i upravljač.

Program upravljač čvorovima u grozdu računala. Zadaje neizvršene poslove koji su prethodno dostavljeni od strane poslužitelja, nadgleda izvršavanje poslova te sakuplja rezultate da budu dostupni poslužitelju.

Poslužiteljeva zadaća je preko mreže primati korisničke poslove koje treba izvršiti, organizirati datotečni sustav te preko mreže slati korisniku rezultate izvršenih poslova.

Klijent je program koji omogućuje interakciju korisniku sa poslužiteljem preko mreže. On šalje korisničke poslove poslužitelju i sprema rezultate na korisničkom računalu



Slika 1 Jednostavan prikaz arhitekture sustava

Na Slika 1 Jednostavan prikaz arhitekture sustava. Mogu se razaznati neke osnovne komponente sustava, njihov razmještaj i međusobni odnos.

3. Klijent

Klijent je program koji se spaja na poslužitelj i omogućuje interakciju sa poslužiteljem. Podatci između klijenta i poslužitelja prenose se TCP protokolom. Klijentu je poznata IP adresa poslužitelja i vrata na kojima se može spojiti sa poslužiteljem. Ukoliko je poslužitelj pokrenut klijent će se pri pokretanju pokušati spojiti na njega, u suprotnom klijent dojavljuje poruku o odsutnosti poslužitelja i gasi se.

Kada pristupi poslužitelju, klijent omogućuje korisniku da unese svoje korisničko ime i lozinku. Klijent te podatke šalje poslužitelju, ukoliko nakon zaprimljenih podataka poslužitelj pošalje poruku o grešci klijent se gasi. Primi li klijent poruku od poslužitelja da ima pravo pristupa tada će klijent ponuditi korisniku akcije koje može izvršiti. Akcije koje su ponuđene korisniku su: 1) Zadati novi posao., 2) Preuzeti rezultate posla koji je izvršen., 3) Upitati za informacije o stanju poslova., 4) Upit za potpunim informacijama. i 5) Odjaviti se sa sustava.

3.1. Zadavanje novog posla

Ako korisnik odabere zadati novi posao tada će ga klijent zatražiti da navede stazu (relativnu ili apsolutnu) do ugovor - datoteke. Ugovor - datoteka je tekstualna datoteka koja treba biti oblikovana prema određenim pravilima.

U prvom redu datoteke se nalazi ime posla koji će se zadati. Ime mora biti valjano ime datoteke. Ime je valjano ime datoteke ako nije duže od dvjesto pedeset i pet znakova te ne sadrži zabranjene znakove ('!', '#', '$', '%', '&', '\", '(', ')', '+', ',', '-', '.', ';', '=', '@', '[', ']', '^', '_', '`', '{', '}', '~').

U drugom retku se može nalaziti veliko ili malo slovo "s" te veliko ili malo slovo "p". Slovo s označava da je posao koji se predaje slijedan, a ukoliko se nalazi slovo p to znači da je posao paralelan.

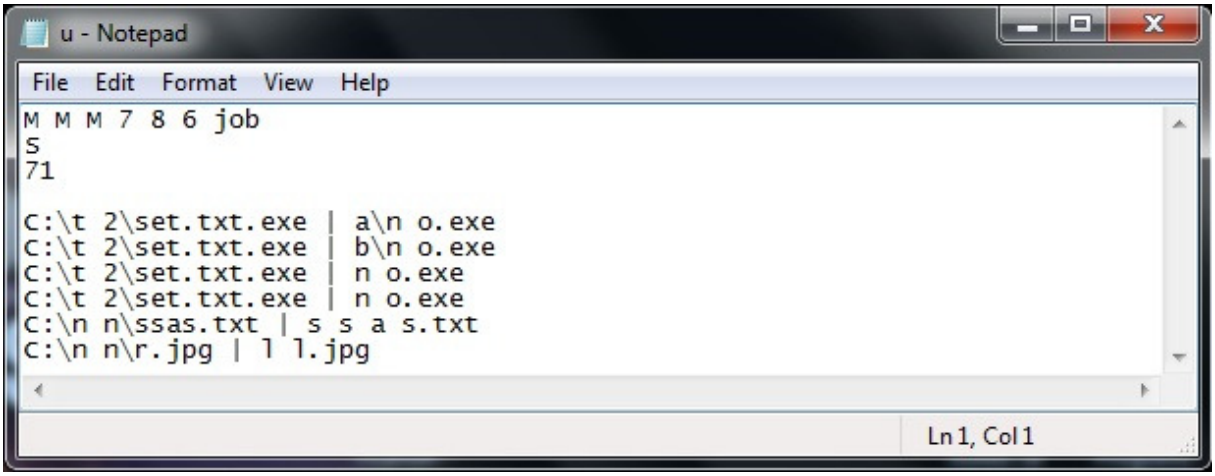
U trećem retku se treba nalaziti broj računala potrebnih za izvođenje tog posla naravno zapisan znamenkama. Ako je posao paralelan tada zatraženi broj računala ukazuje na koliko računala mora sudjelovati u paralelnoj izvedbi. Ako je zadani posao

slijedan tada zatraženi broj računala ukazuje koliko puta jedan takav posao treba biti izvršen na nekom od računala.

U četvrtom retku se mogu nalaziti parametri s koji će se proslijediti tome poslu. Ako korisnik ne želi da se prosljede ikakvi parametri tada u tom retku se treba nalaziti barem jedno prazno mjesto. Ograničenje sustava koje je nastalo usred nepažnje jest da trenutno nije moguće zadati parametar koji u sebi sadrži znak '#'. Upisane parametre treba zadati kao da se zadaju program u komandnoj liniji tj. Ukoliko želimo za parameter proslijediti niz znakova koji u sebi sadrži znak razmaka, cijeli znakovni niz valjda ograničiti dvostrukim navodnicima.

U petom retku se nalazi relativna ili apsolutna putanja izvršne datoteke na računalu gdje je pokrenut klijent te znakom „|" (okomita crta) odvojena relativna putanja gdje će se nalaziti datoteka na udaljenom računalu za koju vrijedi da se datotečna imena odvajaju obrnutom kosom crtom. Korisniku je poznato da će se njegov posao smjestiti u nekom njemu nepoznatom direktoriju, ali on može unutar tog direktorija sam rasporediti svoje datoteke na način koji želi.

U sljedećim redcima koje korisnik može navesti, ali i ne mora, nalazi se popis datoteka koje će se prenjeti na poslužitelj u istoj notaciji kao i izvšna datoteka, kao staza na lokalnom računalu te relativna staza na udaljenom računalu kao što je to prikazano na Slika 2 primjer ugovor datoteke.



Slika 2 primjer ugovor datoteke (ime datoteke u.txt)

Kada korisnik navede stazu do ugovor - datoteke prvo će provjeriti ispravnost svih podataka (postoje li datoteke na lokalnom računalu, da li je moguća relativna staza na udaljenom računalu, itd.). Ako klijent zaključi da su podatci ispravni prvo šalje podatke o imenu posla, broju potrebnih računala, tipu posla, parametrima te putanju gdje će biti spremljena izvršna datoteka. Odbije li poslužitelj podatke klijent prestaje sa radom. Ako poslužitelj prihvati podatke klijent FTP protokolom šalje izvršnu datoteku. Proces se nastavlja slanjem putanja ostalih datoteka, a zatim i samih datoteka sve dok postoje datoteke koje treba poslati ili dok poslužitelj ne odbije primiti nove pakete podataka (što je najvjerojatnije uzrokovano prestankom rada poslužitelja).

U slučaju da posao nije primljen klijent prestaje s radom i javlja poruku o grešci, inače ispisuje poruku o uspješnosti i vraća korisnika na početni izbornik.

3. 2. Preuzimanje rezultata

Odabere li korisnik u izborniku da želi preuzeti rezultate posla koji je , klijent će ga zatražiti ime tog posla. Potom će zatražiti da unese stazu do mape u kojoj želi spremiti primljene rezultate, staza može biti relativna ili apsolutna. Ukoliko staza koju je klijent zadao ne postoji tada klijent obavlja sljedeću akciju: ukoliko je staza ispravna osim posljednje mape tada stvara odgovarajuću mapu, u suptrotnome dojavljuje grešku i prestaje s radom.

Potom klijent šalje poslužitelju ime poslja kojeg je korisnik prethodno unio. Ukoliko posao nije dovršen ili ga taj korisnik nije ni zadao, klijent će dojaviti poruku o grešci i prestati s radom.

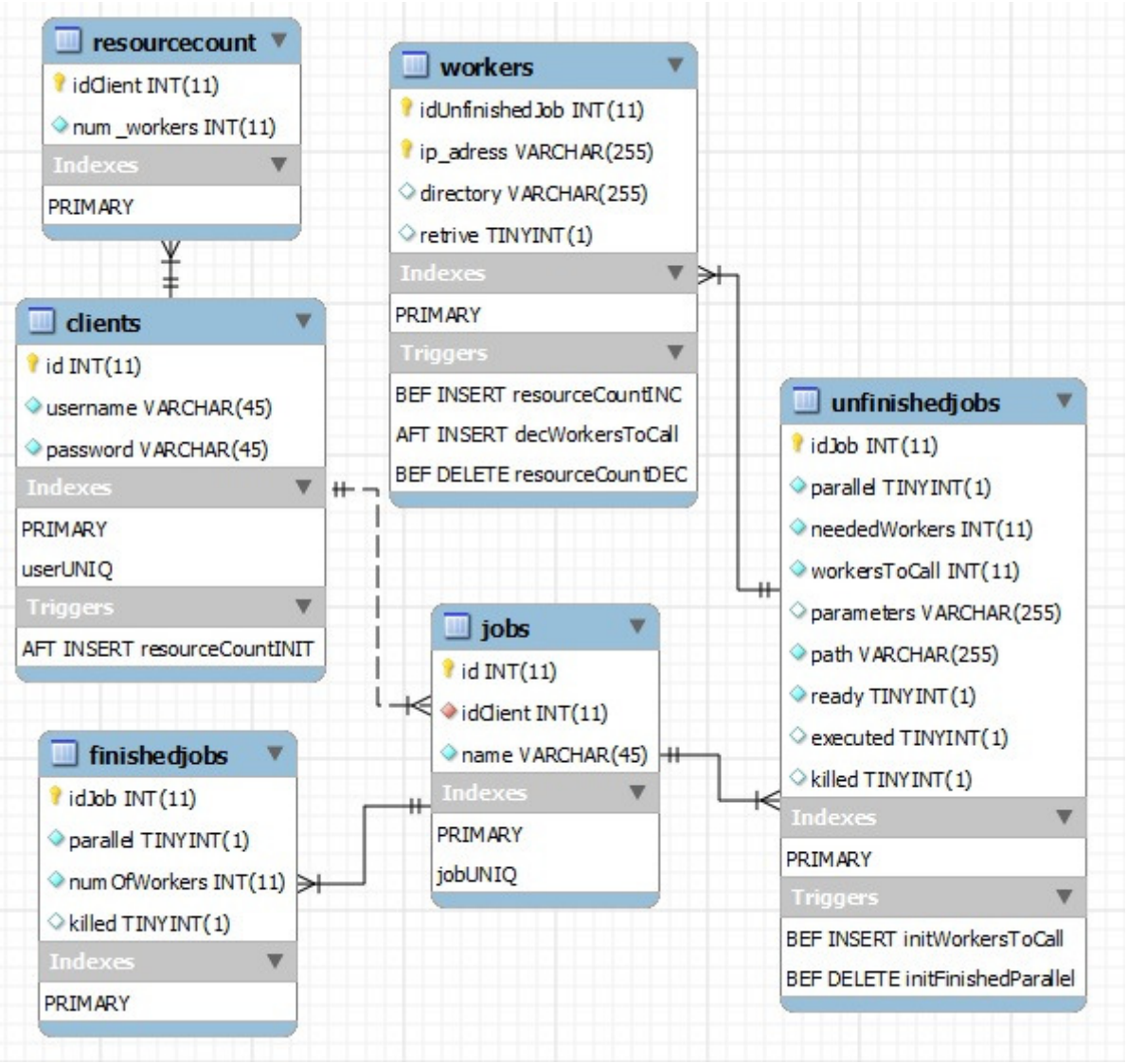
Svi podatci koje klijent prima od poslužitelja spremaju se u mapu koju je korisnik naveo. Prijenos podataka se odvija tako da klijent prima poruku koja sadrži relativnu stazu u kojoj treba smjestiti datoteku, a zatim FTP protokolom prihvaca datoteku. Ukoliko prijenos svih podataka bude neuspješan, primljene datoteke automatski se brišu i klijent se gasi.

3.1 3. 3. Preostale akcije

Treći izbor koji korisnik može odabrati će uputiti klijenta da pošalje upit poslužitelju koji će vratiti podatke o tome koliko je korisnikovih poslova dovršeno, koliko se poslova izvodi i koliko ih još čeka na red. Četvrti izbor je sličan prethodnome s razlikom da će navesti imena poslova unutar skupina. Posljedni izbor omogućuje korisnika da ugasi klijenta.

4. Baza Podataka

Baza podataka služi kao skladište podataka o postojećim korisnicima i njihovim poslovima. Njena uloga je služiti kao medij preko kojeg se kordinira rad poslužitelja i sustava raspodjeljenog upravljanja korisničkim poslovima. Sastoji se od pet tablica: *clients, jobs, unfinishedJobs, workers, finishedjobs, resourcecount* kao što je prikazano na Slika 3 E – R model baze podataka (Crow's foot notation).



Slika 3 E – R model baze podataka (Crow's foot notation)

4.1 Tablica *clients*

Posjeduje atribute koji predstavljaju idetifikacijski broj, korisničko ime te lozinku. Primarni ključ te tablice je idetifikacijski broj, a jedinstveni ključ korisničko ime, što znači da nijedan entitet u toj tablici neće imati istu vrijednost indetifikacijskog broja niti će imati isto korisničko ime. Na tablici je ugrađen okidač resourceCountINIT čija je zadaća da nakon svakog unosa odataka o novom klijentu stvori entitet u tablici resourcecount koji će pamtiti da taj stvoreni klijent trenutno koristi točno nula računala na grozdu računala.

4.2 Tablica *jobs*

Tablica *jobs* je generalizacija tablica *unfinishedJobs* i *finishedJobs*. U tablici se spremaju osnovni podatci o poslovima koji su zadani za izvršavanje ili su izvršeni. Podatci koje tablica sadrži za jedan posao su identifikacijski broj posla, identifikacijski broj klijenta kojem posao pripada te ime posla. Primarni ključ tablice je identifikacijski broj posla. Postoji ograničenje stranog ključa na atribut identifikacijski broj klijenta koji se referencira na tablicu *clients* i odgovarajući atribut. To znači da je tablicaslabi entitet tablice clients što zapravo znači da entitet u tablici jobs ne može postojati ukoliko ne postoji odgovarajući entitet u tablici *clients* s kojim se može povezati.

Valja napomenuti kako bi tablica bila i korisnija kada bi se u njoj nalazi podatak o tome da li je posao paralelan ili slijedni jer se taj podatak poznaje odmah pri unosu posla i nalazi se u obje postojeće specijalizacije tablice. To je propust koji u vrijeme pisanja ovog rada nije ispravljen.

4.3 Tablica *unfinishedJobs*

Ova tablica je specijalizacija tablice *jobs*. Strani ključ te tablice je atribut identifikacijski broj posla koji se referencira na odgovarajći atribut tablice *jobs*. Primarni ključ tablice je ujedno i strani ključ. Sarži podatke o poslovima koji nisu izvršeni. Njeni atributi su identifikacijski broj posla, tip posla (ili slijedan ili paralelan), koliko radnika je zatražio korisnik, koliko radnika nam preostaje zauzeti prije nego posao bude izvršen, parametri, relativna staza na udaljenom računalu gdje će se

nalaziti izvršna datoteke, zastavica koja pokazuje da li je posao spreman na izvršavanje, zastavica koja pokazuje u kojem je stanju izvršavanja posao, te zastavica koja ukazuje da li je proces ubijen pri izvršavanju.

Na tablici se nalaze dva ugrađena okidača. Prvi okidač initWorkersToCall će prilikom unosa entiteta u ovu tablicu postaviti atribut koji predstavlja koliko radnika nam preostaje zauzeti prije nego posao bude izvršen na isti iznos atributa koliko radnika je zatražio korisnik. Drugi okidač će prije no što se obriše entitet u ovoj tablici unjeti novi entitet u tablicu *finishedJobs* koji će sadržavati identifikacijski broj posla, tip posla, koliko radnika je izvršavalo taj posao te da li je proces bio ubiven.

4.4 Tablica *workers*

U tablici workers obavlja se evidencija o računalima koja izvode poslove. Tablica ima četiri atributa, to su indetifikacijski broj posla, IP adresu računala koji trenutno izvršava posao s tim indetifikacijskim brojem, stazu na kojoj su spremljeni datoteke potrebne za izvršavanje posla te zastavicu koja označava da s dotične staze treba dohvatiti podatke. Primarni ključ tablice jest indetifikacijski broj posla te IP adresa računala čime ne dopuštamo da se više instanci jednog posla istovremeno izvršava na jednom računalu. Tablica ima ograničenje stranog ključa, a to je indetifikacijski broj posla koji referencira tablicu *unfinishedJobs*.

Na tablici su ugrađena tri okidača. Prvi okidač resourceCountINC brine se da se u tablici *resourceCount* pri svakom unosu entiteta za odgovarajućeg radnika podatci o klijentovom zauzeću broja računala uveća za jedan. Drugi okidač decWorkersToCall u tablici *unfinishedJobs* s smanjuje broj radnika koje je preostalo pozvati prije no što se posao izvrši. Treći okidač resourceCountDec brine se da se u tablici *resourceCount* pri svakom brisanju entiteta iz tablice za odgovarajućeg radnika podatci o klijentovom zauzeću broja računala smanji za jedan.

4. 5 Tablica *finishedJobs*

Ova tablica je specijalizacija tablice *jobs.* Strani ključ te tablice je atribut identifikacijski broj posla koji se referencira na odgovarajći atribut tablice jobs. Primarni ključ tablice je ujedno i strani ključ. Sarži podatke o poslovima koji su

izvršeni. Atributi tablice su identifikacijski broj računala, tip posla, broj radnika koji je izvršavao posao te zastavica koja ukazuje da li je posao ubiven.

4.6 Tablica *resourceCount*

Tablica resourcecount pohranjuje vrijednosti koje predstavljaju koliko je računala dodijeljeno za izvršavanje poslova nekog klijenta. Atributi su indetifikacijski broj klijenta te broj računala koji je tom klijentu dodijeljen. Primarni ključ tablice je indetifikacijski broj klijenta, što je ujedno i strani ključ.

5. Poslužitelj

Poslužitelj se pri pokretanju priprema za moguće prihvaćanje klijenata na vratima koja su poznata njemu i klijentu. Klijenti se mogu spojiti s bilo koje IPv4 adrese ako pristupaju na dotična vrata. Nakon što je spreman prihvaćati klijente poslužitelj pokreće još dvije dretve. U jednoj dretvi može prihvaćati naredbe sa terminala. Trenutno podržava samo jednu naredbu, a to je exit. Po upisu naredbe exit u terminal poslužitelj gasi dretve, oslobađa memorisjki prostor, prestaje osluškivati na vratima te pokreće proces gašenja. U drugoj dretvi se u idealnim uvjetima bez naredbe za gašenjem poslužitelja vrti beskonačna petkja u kojoj čeka na klijenta koji se pokušava spojiti na poslužitelja. Kada naiđe takav klijet poslužitelj stvara spojnicu s tim klijentom. Spojnice s klijentima sprema u podatkovnu strukturu kojoj se može pristupiti iz cijelog procesa poslužitelja, stoga se pritup istoj programskoj strukturi (skup) uvjek zaštićuje uporabom semafora. Kritičnim odječkom mora zaštiti još rječnik koji sadrži vezuje podatake o spojnicama i podatke o klijentima. U rječnik se dodaju podatci odmah nakon što se spojnica uvrsti u popis trenutno aktivnih spojnica. Podatci o klijentu spremaju se u strukturu CLIENT.

```
typedef struct {
  State state;
  int sended;
  int serial;
  std::string user;
  std::string currentJob;
  std::string nextPath;
  int id;
} CLIENT;
```

Slika 4 Struktura podataka u kojoj poslužitelj sprema podatke o klijentu

Struktura CLIENT sastoji se od strukture State koja vodi evidenciju o tome u kojem se stanju trenutno nalazi klijent, dvije cjelobrojne vrijednosti koje služe kao podatci o datotekama koje poslužitelj šalje klijentu. Zatim još sadrži znakovne nizove koji u kojima je pohranjeno klijentovo korisničko ime, trenutni posao koji se šalje i prima, te trenutna staza na kojoj će se spremiti primljena datoteka ili poslati datoteka

prema klijentu. Struktura također sadrži vrijednost indetifikacijskog brioja klijenta koja je zapisana kao cjelobrojan vrijednost. Stvaran isječak programskog koda za tu strukturu može se vidjeti na Slika 4 Struktura podataka u kojoj poslužitelj sprema podatke o klijentu.

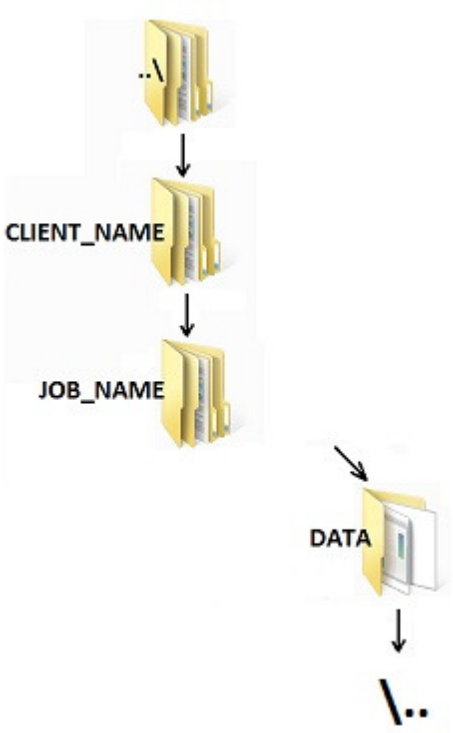
U glavnoj dretvi procesa vrti se petlja sve dok se u petlji za zadavanje naredbi nad poslužiteljem ne unese exit. U toj petlji poslužitelj pristupa skupu trenutno spremljenih poveznica i stvara kopiju te strukture. Razlog stvaranja kopije je već spomenut, a leži u tome da je ta struktura dostupna ostalim dretvama te ju treba zaštititi kritičnim odsječkom. Kako ostale dretve ne bi morale čekati dok glavna dretva ne oslobodi sredstvo, radimo njegovu kopiju. Nakon što stvori kopiju skupa poveznica, iz kopije skupa izbacuje poveznice prema onim klijentima koji nisu poslali poruku prema poslužitelju. To je skup poveznica s kojim dalje nastavlja raditi. Za svaku poveznicu unutar tog skupa prima poruku od klijenta te obavlja odgovarajuće akcije sukladno sa zapisanim stanjem tog klijenta i porukom koju je klijent poslao. Kada posluži sve klijente vraća se na početak petlje. Ukoliko se klijent odspoji, a poslužitelj to ustvrdi prilikom slanja ili primanja poruke ući će u kritični odječak i iz glavnog skupa u kojem drži popis poveznica obrisati poveznicu tog klijenta i podatke koje drži u memoriji vezane uz tog klijenta.

Valja objasniti način na koji poslužitelj poslužuje klijenta, no prije toga treba naglasiti da je dužnost poslužitelja ravnomjerno posluživati klijente. Kako bi poslužitelj ravnomjerno posluživao klijente pri svakom posluživanju jednog klijenta poslužitelju je namjera poslati maksimalno jednu poruku ili prenjeti jednu datoteku te nastaviti posluživati ostale klijente.

Poslužiteljevo posluživanje klijenata se može opisati kao deterministički konačni automat. Objasnimo redom stanja i prijelaze tog automata. Kada dretva za zaprimanje klijenata spremi poveznicu prema klijentu, u strukturi u kojoj drži podatak o stanju tog klijenta navodi da se nalazi u stanju kada je sljedeća akcija koju treba učiniti poslati korisničko ime. Sljedeći put kada klijent pošalje poruku poslužitelj će pretpostaviti da se radi o korisničkom imenu, zapamtiti taj podatak i zapamtiti da je sljedeća akcija koju klijent će učiniti poslati lozinku i zatim poslati poruku klijentu da je primio njegovo korisničko ime. Kada klijent pošalje iduću poruku, poslužitelj će

zaključiti da se radi o lozinki te će poslati upit prema bazi podataka postoji li takav klijent i ako postoji da mu se dohvati njegov indetifikacijski broj. Postoji li klijent u bazi podataka poslužitelj pamti njegov ID, zapisati da sljedeća akcija koju taj korisnik može obaviti je predati odluku o tome koja će biti akcija koja će slijediti iza ove koja prethodi, te šalje poruku klijentu o rezultatu njegove prijave na poslužitelj. Ako klijent ne postoji briše se njegova poveznica iz glavnog skupa poveznica i svi podatci na poslužitelju o tom klijentu. Kada zaprimi iduću poruku od klijenta poslužitelj će ovisno o njenom sadržaju odlučiti daljni skup akcija koje treba poduzeti. Poruka smije sadržavati jednu znamenku broja između jedan i četiri uključivo. Sve ostale poruke će uzrokovati prekid veze između poslužitelja i klijenta. Ako je poruka znamenka jedan poslužitelj će znati da klijent želi predati posao, ako se u poruci nalazi znamenka dva zabilježiti će da klijent želi preuzeti posao. U slučaju znamenke tri i četiri poslužitelj neće promijeniti stanje klijenta. Znamenka tri ukazuje da klijent želi podatke o broju poslova koji su izvršeni, koji se izvode i koje ćekaju na izvođenje, znamenka četiri simbolizira zahtjev za istim podatcima no umjeto broja tih poslova zahtjeva konkretna imena. Poslužitelj će ostavariti vezu prema bazi podataka, dohvatiti potrebne podatke i poslati ih klijentu. Za slučaj da je klijent najavio da će poslati posao, poslužitelj će ući u stanje u kojem poruka koju primi predstavlja esencijalne podatke o poslu, a to su ime posla, tip s obzirom na to da li je paralelan, potreban broj radnika, paramatri s kojima se taj program poziva te relativna putanja izvršne datoteke. Kada poslužitelj u tom stanju zaprimi poruku provjeriti će format poruke tj. može li iz te poruke izvući tih pet podataka i provjeriti da li su njihove vrijednosti unutar domene u kojoj se te vrijednosti smiju nalaziti. Poslužitelj dodatno provjerava postoji li posao pod takvim imenom zapisana u bazi podataka. Prilikom provjere predanih podataka ako greška nije nastala poslužitelj sprema primljene podatke u bazu podataka, interno bilježi da sljedeće što taj klijent smije napraviti jest poslati datoteku, pamti relativnu stazu na kojoj ta datoteka treba biti spremljena te šalje klijentu poruku o uspješnosti slanja podataka o poslu. U sljedećem stanju u kojem poslužitelj može samo primiti datoteku od klijenta, dohvaća upamćenu stazu na kojoj se treba nalaziti ta datoteka te stvara putanju do mjesta na kojem ta datoteka treba biti stvorena. Prethodno je rečeno da predana putanja je samo relativna, a klijent ne treba znati gdje će poslužitelj spremiti primljene podatke.

Datotečni sustav organiziran je unutar staze poznate polsužitelju. Unutar te staze kako bi se spremio posao stvara se mapa ako prethodno nije bila načinjena po imenu klijenta. Zatim u toj mapi stvara se mapa čije je ime jednako imenu posla koji se predaje. Unutar te mape stvara se folder "DATA" u kojem se dodatno rade poddirektoriji ako su navedeni u relativnoj stazi, inače se datoteka smješta u toj mapi prema Slika 5 Stanje datotečnog sutava pri prihvaćanju podataka od klijenta.

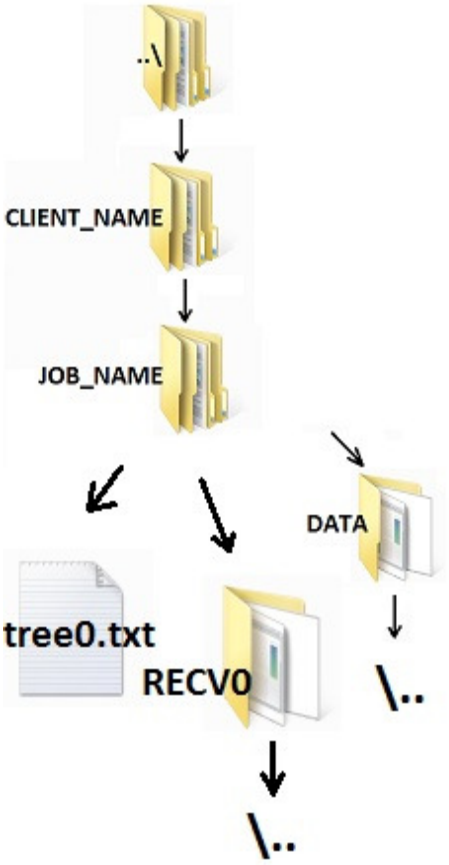


Slika 5 Stanje datotečnog sutava pri prihvaćanju podataka od klijenta

Ako uspješno primi datoteku od korisnika pčekuje dodatnu poruku od klijenta u kojoj mu klijent poručuje ima li još datoteka za predati. Ako klijent odgovori da nema poslužitelj pristupa bazi podataka i pohranjuje podatak da je posao koji je upravo primio spreman za izvođenje, šalje klijentu poruku o uspjehu i dolazi u stanje u kojem mu poruka od klijenta predstavlja izbor između akcija koje klijent može učiniti. Ako klijent odgovori da još nije prenio sve datoteke potrebne za izvršenje svoga posla, poslužitelj će ući u već posjećeno stanje kada očekuje od klijenta da mu pošalje relativnu stazu na kojoj želi smjestiti datoteku koju će poslati. Jedini način da posao bude označen kao spreman za korištenje jest da klijent preda datoteku, pošalje

poruku da je to posljednja datoteka i da pritom nisu nastale greške. U suprotnom ako nastanu greške, poslužitelj briše spojnicu između sebe i klijenta, briše primljene podatke u datotečnom sustavu te briše pohranjene podatke u bazi podataka koji su vezani uz taj posao.

Ukoliko je klijent odabrao da želi preuzeti posao koji je već izvršen na računalu. Poslužitelj bi prvu klijentovu poruku dočekao kao poruku o spremnosti klijenta da primi zadatak. Poruka prije slanja ikakve datoteke je nužna jer je poslužitelj osmišljen tako da nikada samoinicijativno ne šalje poruke klijentu ukoliko nisu odgovor na neku klijentovu primljenu poruku. Poslužitelj provjerava da li je poruka poruka greške ili uspjeha te ovisno o rezultatu nastavlja sa radom. Podaci koji su pripremljeni kako bi se poslali natrag klijentu od strane sustava za raspodijeljeno upravljanje korisničkim poslovima nalaze se u poddirektoriju ili više njih koji se nalazi u mapi posla. Imena mapa u kojima se nalaze nazivaju se u obliku RECVn, gdje je n broj od nula do broja rezultata koji se treba prenjeti isključivo (npr. RECV0). Uz svaku takvu mapu postoji i odgovarajuća tekstualna datoteka pod imenom treen.txt, gdje je n isti broj kao i u već spomenutoj mapi (npr. tree0.txt). Jednostavan primjer se jasno prikazuje na Slika 6 Stanje datotečnog sustava pri slanju podataka klijentu.



Slika 6 Stanje datotečnog sustava pri slanju podataka klijentu

U tim tekstualnim datotekama uneseni su podatci o svim datotekama unutar odgovarajuće recv – mape. Za svaku datoteku unutar spomenute mape nalazi se redak u kojem se navodi apsolutna staza do te datoteke na računalu gdje je pokrenut poslužitelj. Poslužitelj kada prenosi posao klijentu čita redak po redak iz tree – datoteke i šalje odgovarajuću datoteku. Kako svu pažnju poslužitelj ne bi dao samo jednom klijentu kojem prenosi posao, organiziran je tako da jednom klijentu odjednom šalje samo jednu datoteku, a zatim ako postoje drugi klijenti da nastavi njih posluživati. To je rješeno na sljedeći način. Upitom prema bazi doznaje koliiko postoji recv – mapa. Paralelni poslovi će imati samo jednu takvu mapu dok će slijedni poslovi imati broj takvih mapa jednak broju računala koji su bili potrebni za taj posao. Maksimalni indeks mape će spremiti u podatke koje može spremiti za svakog klijent. Osim toga za svaku tree – datoteku će spremiti koliko je redaka iz nje pročitao. Ako

nakon što pročita redak te datoteke vidi da je došao do kraja, umanjit će brojač datoteka koji su mu ostali za pročitati i postaviti će brojač pročitanih redaka na nulu. Prije samog slanja datoteke klijentu će poslati relativnu stazu te datoteke od mape koja nosi naziv posla do završnog objekta. Tako da se na klijentovom računalu unutar istih recv – mapa organiziraju poodatci. Datoteke čita i šalje podatke spremljene na zabilježenoj putanji sve dok brojač datoteka ne padne ispod nula. Kada pošalje zadnju datoteku dojavljuje klijentu da mu je poslao sve datoteke. Sve dok ne pošalje i zadnju datoteku klijentu dojavljuje da još nije gotov sa slanjem te da ima još datoteka koje treba poslati. Ukoliko se dogodi greška u slanju datoteka poslužitelj će se ukloniti spojnicu sa klijentom i odstraniti podatke o korisniku koji više nije prijavljen na poslužitelju. Pošalje li posljednju datoteku i potvrdi li klijent da je primio sve datoteke, poslužitelj će obrisati sve podatke o tom poslu koji se nalaze u datotečnom sustavu i bazu podataka.

Dođe li do nepopravljive greške u radu poslužitelja on će se postaviti zastavice u svakoj dretvi da treba prestati sa svojim izvođenjem. Glavna dretva će sačekati da se ostale dvije dretve ugase te će zatim počistiti njih prostor, očistit skup poveznica, podatke o klijentima i ostale podatke koji su nastali tjekom rada i ugasiti poslužitelj. Svi neposluženi klijenti će biti zanemareni, a akcije koje nisu dovršili odbačene.

6. Upravljač

Pod komponentom upravljač smatramo program menađer te prateće programe i funkcionalnosti koje program menađer koristi. Program menađer mora znati stazu mape na kojoj program poslužitelj sprema i dohvaća podatke prema klijentu. Treba napomenuti da ubijanje procesa nije u potpunosti podržano u okviru ovog završnog rada. Prostor za tu funcionalnost i ostali potrebni mehanizmi su načinjeni, no menađer nikada ne poziva program „ubojica“, razlog tomu je nedovoljno testiranje navedenog programa i potrebnih ispitivanja postoje li nuspojave koje se mogu dogoditi pri njegovom pozivanju. Program tijekom svog izvođenja koristi dva programa (treći bi bio program ubojica). Ta dva programa su pokretač i nadglednik. Pokretač i nadglednik se moraju nalaziti na stazama koje su dostupne radnicima i programu menađer.

6.1 Pokretač

Program pokretač je vrlo jednostavan i kratak program koji koristi standard MPI (implementacija MPICH2). Za argumente prihvaća broj radnika na kojem se program treba izvesti, ip adrese tih radnika, staze do mapa tih računala gdje su pohranjeni podatci za posao, relativnu adresu do izvršne datoteke koja je jednaka za sva računala te parametre s kojima se posao poziva ukoliko takvi postoje. Iz argumenata koje je primio slaže naredbu koja će se pokrenuti na svakom od računala, te se na svakom od računala na kojem je pokrenut program pokreće za njega određena naredba i na svakom računalu se preusmjerava ispis programa u *log* datoteku u korjenu staze gdje se nalaze datoteke potrebne za izvršavanje posla. Proces završava onog trena kada na svim računalima na kojima se pozvala naredba za pokretanje posla izvršavanje tog posla bude dovršeno. Program se pokreće pomoću izvršnog programa mpiexec.exe.

6.2 Nadglednik

Program nadglednik je također jednostavan i kratak program koji koristi standard MPI (implementacija MPICH2). Za argument prihvaća ime procesa. On provjerava da li se na svim radnicima na kojima se pokrenuo taj program još uvjek

izvršava proces pod zadanim imenom . Kako se program pokreće pomoću izvršnog programa mpiexec.exe na svakom od radnika na kojem se izvodi instanca ovog procesa će provjeriti da li se još uvjek izvodi proces pod imenom koji je zadan kao arguent programa nadglednik. Ukoliko je pozivanje programa bilo uspješno i njegov tijek izvođenja se dovrši svaka od instanci procesa će vratiti jednu od dvije unaprijed definirane vrijednosti. Manja od dvije unaprijed definirane vrijednosti simbolizira da je program izvršen, dok veća sibolizira da se program još uvjek izvršava. Program mpiexec.exe će svome pozivatelju vratiti vrijednost koja je bila najveća od svih instanci procesa koje je mpiexec.exe započeo. Dakle ukoliko se na barem jednom računalu izvodi navedeni proces program mpiexec.exe će svome pozivatelju vratit unaprijed definiranu veću vrijednost.

6.3 Menađer

Buduí da se pomo'ni programi pokreću preko programa mpiexec.exe menađer mora poznavati putanju na kojoj se nalazi taj program. Pri pokretanju menađer pristupa bazi podataka i zapisuje koji su poslovi pokrenuti, a dosad nisu dovršeni i njihovi rezultati nisu spremljeni. Zatim zahtjeva od administratora da unese stazu do konfiguracijske datoteke koja može biti relativna ili apsolutna.U konfiguracijskoj datotecu u svakom od redova se može nalaziti IPv4 adresa i staza direktorija razmaknuti sa razmakom. To je popis svih računala koji su dostupni sustavu za obaljanje poslova i dijeljenih direktorija na kojemu se mogu obavljati poslovi. Sustav prvo prolazi kroz datoteku i provjerava ispravnost podatka koji se nalaze u njoj te nakon što alocira potrebnu memoriju za spremanje tih podataka ponovo prolazi kroz datoteku i preuzima podatke koji se nalaze u njoj. Jedino ograničenje na konfiguracijsku datoteku jest da mora biti barem jedan radnik upisan u njoj. Ograničenje je nastalo propustom koji je poslije uočen, a u trenutnoj verziji sustava nije ispravljen. Nakon što učita datoteku sustav uspoređuje podatke sa popisom nedovršenih poslova. U svome popisu dostupnih računala gleda postoji li zapis s tim računalom za koje zna da je zauzeto i mijenja njegovu vrijednost na trenutno zauzeto.

Kada je inicijalizirao sve podatke menađer pokreće dodatne dvije dretve. U prvoj dretvi prima naredbe od administratora. Trenutno jedina podržana naredba je naredba exit. U drugoj dretvi nadgleda izvršavanje poslova, a u glavnom programu se brine za pokretanje zadanih poslova.

U glavnom programu zatim ulazi u beskonačnu petlju sve dok administrator ne naredi gašenje programa ili se ne dogodi neka neočekivana fatalna greška koja će pokrenuti proceduru gašenja sustava. Ponovo pristupa bazi podataka i ovisno o broju trenutno dostupnih radnika te ovisno o tome koliko svaki klijent je zauzeo radnika stvara listu poslova koje može izvršiti. Pri odabiru poslova treba napomenuti da paralelni posao može biti odabran za izvršavanje samo ako je potreban broj radnika dostupan na mreži, dok će slijedni poslovi odabrani ako postoji jedan ili više slobodnih radnika na grozdu te će se ukupno izvoditi onoliko puta koliko je to klijent tražio. Pri odabiru izvođenja slijednih poslova pokušava određuje najveći broj radnika koje može pozvati u tom trenutku. Kada je sastavio popis poslova koje će izvršiti izvršava jedan po jedan posao kako su navedeni u popisu. Ukoliko je popis prazan, suspendirati će svoje izvođenje na određeno vrijeme i kako bi sačekao dok ne bude u mogućnosti pokretati poslove na slobodnim radnicima. Kada može pokrenuti neki posao sakuplja ip adrese i putanje radnih direktorija radnika na kojima će se proces izvoditi (ako je posao slijedni tada za jednu istancu sve čini samo na jednom radniku), pamti ip adresu i putanju radnog direktorija onog radnika s kojega će preuzeti rezultate izvršenog posla, prebacuje potrebne datoteke za posao na sve radnike te pokreće posao. Posao pokreće pomoću programa mpiexec.exe iz MPICH2 implementacije standarda MPI pokrene program pokretač koji će na svim zadanim radnicima pokrenuti posao. Program mpiexec.exe u ovom slučaju pokreće tako što naređuje operacijskom sustavu da započne neovisni proces o menađeru.

U dretvi za nadgledanje radnika menađer treba odrediti da li su poslovi izvršeni i da li bi ih trebalo ubiti. Dretva ulazi u besknačnu dretvu dok korisnik ne zatraži prekidanje rada menađera ili se dogodi fatalna greška koja će uzrokovati poziv procedure za prekid rada menađera. Nakon što uđe u beskonačnu petlju stvara skup u kojem pohranjuje popis svih pokrenutih poslova koji nisu dovršeni te nakon unaprijed određenog broja iteracija ako se ti poslovi još uvjek nalaze u tom skupu

procese tih poslova treba ubiti, menađer ne ubija procese no kada bi ih ubio za svaki ubijeni paralelni posao dohvatio bi podatke sa glavnog radnika koji je izvodio posao, a za slijedne bi dohvatio samo preostale instance koje su se izvršavale. Dakle, unutar beskonačne petlje ulazi u konačnu petlju sa unaprijed zadanim brojem iteracija. Na početku te petlje kopira sakuplja popis svih poslova koji se trenutno izvode. Nakon što sakupi popis za svaki od tih poslova pokreće program mpiexec.exe (objašnjen prethodno) koji na svim radnicima na kojima se izvodi posao pokreće program nadglednik kojemu prosljeđuje argument imena procesa tog posla. Izvođenje programa se nastavlja kada se taj program izvrši i ovisno o vrijednosti koju je vratio taj program izvršavaju se odgovarajuće akcije.Ukoliko vraćena vrijednost javlja da se program još izvršava na barem jednom čvoru tada se vraćamo na početak petlje i ispitujemo za idući posao na popisu. Ukoliko je posao dovršen treba dohvatiti podatke sa radnika na kojem se nalaze rezultati. Ukoliko se dohvaćaju podatci izvršenog paralelnog posla dohvaća se samo jedan radnik te se sprema u datotečnom . Ako je posao bio slijedan dohvaćaju se podatci sa upravo tog radnika koji je i izvodio program i podatci se organiziraju unutar datotečnog sustava na način na koji ih poslužitelj očekuje. Dohvaćeni podatci od radnika spremaju se unutar datotečnog sustava redom u mape koje se nalaze unutar direktorija odgovarajućeg posla, a nazivaju se RECVn gdje n predstavlja redni broj direkorija sa podatcima. Nakon stvaranja svake mape stvara se opisna teksutalna datoteka označena brojem koji je isti kao i broj u mapi čiji opis topologije sadrži. Potom se brišu sve datoteke na radnicima, u vlastitoj strukturi za evidenciju radnika oslobađaju se radnici koji su prisustvovali u izvršavanju posla. U bazi podatka briše se posao iz popisa nedovršenih poslova čime se automatski ubacuje odgovarajući entitet u popis dovršenih poslova te naravno briše se zapis o poslu iz popisa nedovršenih poslova. I kraju petlje izvođenje se suspendira na vrlo kratko unaprijed određeno vrijeme.

7. Zaključak

Pri izradi programa klijenta i poslužitelja valja paziti na to da program poslužitelj svim ravnopravnim instancama klijenta po mogućnosti osigura jednako pravo na uslugu. Također pri izradi poslužitelja valja paziti da klijent ne bude u mogućnosti srušiti program poslužitelja svojim naglim prekidom. Sve provjere uvjek, obavezno treba napraviti na poslužitelju jer klijent koji se spojio na taj poslužitelj nije nužno pregledao podatke koje šalje i uvjerio se da su ispravni. Te naravno valja osigurati robusnost oba programa pri neočekivanim greškama i mogućim greškama u mreži.

Pri izradi baze podatka valja valja pripaziti da količina tablica u bazi podataka je primjerena veličini sustva kao i na to da ne pohranjujemo nepotrebne podatke. Također bazu podataka treba organizirati da ne postoje česta spajanja među tablicama pri dohvaćanju određenih podataka. Kako se pri izradi ovog rada baza podataka dovoljno često (pa čak i zamjetno) mijenjala u ovakvim prilikama kada je baza u središtu komunikacije među komponentama sustava valja pripaziti da se komponentama očuva pristup onim bitnim podatcima jer izmjene na bazi uzrokuju i izmjene na komponentama.

Pri izradi sustava za raspodjeljeno upravljanje korisničkim poslovima bitno je odabrati dobar skup alata kojim se ostvaruje povezanost među čvorovima grozda. MPI kao standard je od posebne zanimljivosti budući da sam standard propisuje preko više od stotinu funckija, a poznat je kao standard za kojeg vrijedi da se sa vrlo malim poznatim skupom funkcija mogu ostvariti mnoge funkcionalnosti. Osim toga treba dobro paziti za koju je platformu namjenjen. Zbog određenih poziva prema operacijskom sustavu potrebno je pripaziti na razlike u sintaksi čak i među srodnim operacijskim sustavima.

8. Literatura

[1] Prvi izvor [1] Tech Terms, Cluster

http://www.techterms.com/definition/cluster

[2] Predavanja iz kolegija Paralelno programiranje

http://www.fer.unizg.hr/_download/repository/Paralelno_programiranje_predavanja%5B8%5D.pdf

9. Sažetak

Sustav za raspodjeljeno upravljanje korisničkim poslovima

Arhitekturu sustava možemo podijeliti na četiri zasebne cjeline. Cjeline su redom klijent, poslužitelj, baza podataka te Upravitelj. Svi korisnici registrirani u bazi podataka imaju jednake ovlasti i prava pri zadavanju poslova ili primanju rezultata. Komunikacija između poslužitelja i upravitelja u svrhu usklađivanja rada obavlja se preko baze podataka. Klijent je program koji omogućuje korisniku interakciju sa poslužiteljem. Poslužitelj je program koji prihvaća klijetnove zahtjeve i služi mu kao vrata prema resursima na grozdu računala. Upravitelj se brine o izvršavanju poslova u skladu s mogućnostima grozda računala. Upravitelj pokreće i nadgleda izvršavanje poslova.

Ključne riječi: klijent, poslužitelj, baza podataka, grozd računala, *Message Passing Interface*

Distrubited job manager in a computer network

System architecture can be divided into four separate entities. Into client, server, database and manager. All users registered in the database have the same rights in applying jobs or receiving the results. Communication between the server and the manager in order to harmonize the work is done through the database. The client is a program that allows the user to interact with the server. The server is a program that accepts requests clients and serves to them as a door to resources on the computer cluster. The manager takes care of performing the work in accordance with the possibilities of computer cluster. The manager starts and monitors the execution of tasks.

Keywords: client, server, database managment system, computer cluster, Message Passing Interface