Dušan Panić

DFTPD – Distribuirani FTP Daemon

DFTPD je distribuirani FTP server sa velikom skalabilnošću u pogledu organizacije njegove infrastrukture. To znači da sistem može biti centralizovan ili decentralizovan. Sinhronizacija podataka se vrši uz pomoć dvosmernog sinhronizacionog mehanizma nazvanog ListFetcher koji dodatno povećava ažurnost sistema. Serveri međusobno sinhronizuju liste u kojima se nalazi detaljan popis njihovih fajlova/direktorijuma. Ovaj sistem omogućava krajnjem korisniku bržu distribuciju fajlova u odnosu na klasičan FTP server. Upotreba distribuiranog FTP servera eliminiše potrebu za upotrebom i traženjem mirrora. Load balancer implementiran u sistem daje mu sposobnost traženja najbližeg, najisplativijeg puta do postojećeg podatka.

Uvod

Distributed computing jeste proces izvršavanja jednog kompjuterskog zadatka na najmanje dva računara, koji su fizički odvojeni, npr. povezani u globalnu mrežu, Internet. Grid computing tehnologija nudi model rešavanja obimnih računarskih problema uz pomoć velikog broja računara organizovanih u sisteme nalik klasterima u distribuiranoj telekomunikacionoj infrastrukturi. Prema IBM-ovim tvrdnjama mainframe računari (odnosno veliki, snažni i skupi računari korišćeni uglavnom od velikih kompanija za obradu ogromne količine podataka, kao što su bankovne transakcije) ne rade ništa 40% svog vremena, UNIX serveri ne rade ništa 90% svog radnog vremena, a najneiskorišćeniji su desktop računari sa velikim procentom idle timea čak do 95%. Upravo zbog ove neiskorišćenosti distributed computing teh-nologiji se pridaje veliki značaj. Kako ExtremeTech (www.extremetech.com) zaključuje, u odnosu na konkurentne tehnologije, kao što su super računari i klasteri, distribuirani sistemi su višestruko jeftiniji, racionalniji, fleksibilniji; upravo je zbog ovoga distributed computing trenutno jedan od najlogičnijih načina deljenja računarskih resursa. Osnovi ove tehnologije su zanovani na organizaciji više sistema u jednu virtuelnu jedinicu. Jedan od poznatijih grid sistema je Sunov N1 Grid. U većini slučajeva arhitektura distribuiranog sistema se sastoji iz klijentskog softvera (softver koji se izvršava na računaru koji je deo distribuirane mreže), koji preciznije definišemo kao agente koji su instalirani na klijentskim sistemima, i jednog ili više mnogo složenijih servera koji se nalaze na glavnim računarima koji kontrolišu ceo distribuirani sistem. Ostali distribuirani projekti vredni pomena su: distributed.net (koji se bavi razbijanjem crypt algoritama), Folding@Home (bave se medicinskim proučavanjima), Genome@Home (prvenstveno se bave proučavanjem RNA), Grid.org (velika distribuirana mreža pod pokroviteljstvom Sjedinjenih Američkih Država) i na samom kraju čuveni Seti@Home koji se bavi pretragom izvanzemaljskog života, ali je njegova osnovna namena masivno paralelna obrada i esitmacija signala.

Naziv DFTPD je proistekao spajanjem pojmova Distributed, FTP i Daemon kao **D**istributed **FTP** **D**aemon.

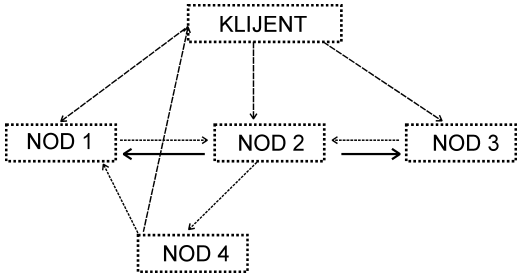
Opis sistema

DFTPD je distribuirani FTP server. Klasičan FTP server se sastoji iz Server PI-a, Server DTP-a i veze sa lokalnim fajl sistemom, što sve zajedno čini FTP daemon ili FTP Server. Ovaj distribuirani sistem poseduje dvosmernu sinhronizaciju podataka, cache proxy, load balancer, kao i FTP virtuelni fajl sistem. Dvosmerna sinhronizacija podataka ima ulogu da poveća ažurnost sistema. Bitna razlika DFTPD-a

Dušan Panić (1987), Loznica, Partizanska 23, učenik 2. razreda Gimnazije "Vuk Karadžić" u Loznici

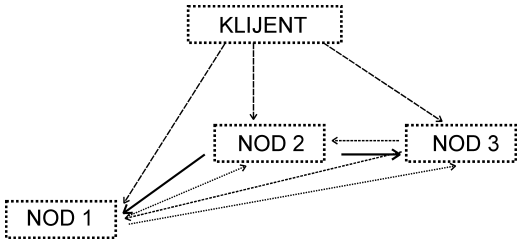
i bilo kog drugog FTP Daemona je ta što ovaj sistem nema direktnu vezu server–fs, već je ta veza obavljena indirektno preko FTP VFS-a. Server je pasivnog tipa, što znači da se veza sa ostalim nodovima aktivira po potrebi.

Cache proxy umnogome smanjuje "opterećenje" mreže, tj. pravljenje nepotrebnog protoka. Upotreba load balancera u ovom sistemu je veoma bitna, jer on ravnomerno raspoređuje opterećenje između procesa, računara itd. DFTPD je napisan u Perl programskom jeziku, što mu omogućuje široku prenosivost.



Slika 1. Primer centralizovne infrastrukture, gde je centralni nod 2

Figure 1. Example of centralized structure, where nod 2 is central



Slika 2. Primer decentralizovne infrastrukture

Figure 2. Example of decentralized structure

Dizajn

DFTP Daemon je za krajnjeg korisnika potpuno transparentni distribuirani FTP Server. Organizacija ovog distribuiranog sistema se razlikuje od organizacije drugih sistema, kao npr. DrFTPd ili Entropia. Sistem je skalabilan u pogledu organizacije njegove infrastrukture, što znači da on može biti centralizovan, odnosno decentralizovan.

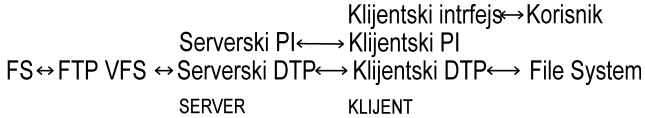
Decentralizovanost se dobija podešavanjem svakog noda da prati svaki drugi nod. Ovakva infrastruktura je veoma dobra za mreže malog obima, jer se povećanjem broja praćenih nodova smanjuje ažurnost sistema, odnosno gubi se na vremenu sinhronizacije, što znači da "ispadanjem" bilo kog servera iz mreže sistem nastavlja da radi potpuno nesmetano.

Centralizovanost se dobija drugačijim podešavanjem nodova, recimo nod 2 prati nodove 1, 3 i 4 dok nod 1 prati nod 2, i sl. (slika 1). Ispadanjem noda 2 iz mreže ovaj deo sistema će prestati sa radom. Postoje distribuirani sistemi čiji dizajn podseća na dizajn DFTPD-a, međutim niko do sad nije napravio ništa slično. Većina se svodi na centralizovani dizajn distribuiranih sistema.

Mana ovakve organizacije je otežana konfiguracija sistema, jer se svaki nod (uređaj povezan na mrežu, npr. računar ili ruter) mora konfigurisati pojedinačno. Međutim, vrlo lako se može napraviti alat uz pomoć kojeg se sa jednog računara mogu jednostavno iskonfigurisati svi ostali. Potrebno je dodati filter DFTPD-u koji bi određene fajlove koji se uploaduju sa određenih lokacija smeštao u konfiguracioni direktorijum. Konfiguracioni fajlovi bi trebali biti u .xml formatu. Komunikacija između nodova je pasivna, što znači da se konekcija uspostavlja po potrebi.

Implementacija

Sistem se sastoji iz više zasebnih celina, a to su: FTPD, liste (ListFetcher, ListManager), FTP Virtual File System, Load balancer i Cache Proxy. Za implementaciju FTP protokola (rfc412, rfc463, rfc640, **rfc959**, rfc238, rfc448, rfc506, rfc775) korišćen je POE::Component::Server::FTP. POE sistem je veoma dobar, jer poseduje lightweight niti koje se bitno razlikuju od Perlovih, prvenstveno po tome što



Dizajn DFTPD aplikacije

DFTPD application design

su "lakše" pa zauzimaju dosta manje sistemskih resursa. Sistem je potpuno skalabilan i omogućava veliku fleksibilnost pri dizajniranju topologije distribuirane mreže.

Kako sve to izgleda kada se pokrene DFTPD:

CleanListDir – očisti direktorijum sa listama
GenList – generiši lokalnu listu
GetAllLists – dovuci liste ostalih nodova
RunFTPD – pokreni ftpd na ip adresi 0.0.0.0, na portu 21

Pri pokretanju DFTPD-a pravi se nova lokalna lista, a brišu se sve prethodne postojeće liste. Kada je lista napravljena kreće se sa dobavljanjem lista nodova koji su u listi praćenja. Prilikom dobavljanja lista sa udaljenih nodova vrši se dvosmerna sinhronizacija lista uz pomoć ListFetcher algoritma. Na osnovu timestampa, drugačije nazvanog list id, koji je sastavni deo imena fajla koji sadrži listu, proverava se da li je lokalna lista servera dobavljača na udaljenom serveru sveža. Ako nije, izvršiće se dvosmerna sinhronizacija podataka između servera dobavljača i udaljenog servera. Kada su sve liste dobavljene može se krenuti sa generisanjem finalne liste. Posle izgenerisane finalne liste server je spreman za rad. Finalna lista se generiše na fiksan vremenski period koji određuje sam administrator sistema ili pri svakoj promeni na FTP VFS-u. FTP VFS je virtuelni fajl sistem koji se snabdeva podacima iz finalne liste. Svi podaci se posmatraju virtuelno, pa i lokalni.

Liste i pravljenje lista

FTP virtuelni fajl sistem predstavlja spregu između servera i sistemskog FS-a. Za rad FTP VFS-a su liste koje sadrže popis svih fajlova, linkova i direktorijuma. Server pri svakoj promeni na virtual fajl sistemu generiše "drvo" sačinjeno od fajlova i direktorijuma koji se nalaze u njegovom root direktorijumu. ListMaker prolazi kroz sve direktorijume, popisuje ih, popisuje fajlove koje oni poseduju, beleži veličine i md5 checksume fajlova. U slučaju da nema fajla ili direktorijuma, odnosno da je određeni dir prazan, on zapisuje nulu. Primer jedne liste, list-rac10.psc.ac.yu-20040706162938:

- list – oznaka liste (prefiks je korišćen zbog mogućnosti implementacije metode dobavljanja svih listi odjednom)
- rac10.psc.ac.yu – oznaka noda
- 20040706162938 – id liste, koji se pravi na osnovu time stamp-a 20040706162938 = 2004-07-06 16:29:38

```
[ /home/dftpd index,size,md5sum;.
        bash_profile,size,md5sum; project;lists ]
[ /home/dftpd/project ListFetcher.pl.bck,
                        size,md5sum; 0; ]
[ /home/dftpd/lists list-rac09.psc.ac.yu-
                20040706155839,
                        size,md5sum; 0; ]
```

Lista se napravi pri pokretanju DFTPD-a a praviće se i pri svakoj promeni na FTP VFS-u. Ovime još više dobijamo na ažurnosti sistema. Svaki server (nod) pored svoje lokalne liste poseduje i liste drugih nodova koje prati.

Testiranje brzine algoritma ListMakera			
ukupno fajlova i direktorijuma	1234	3917	285176
vreme kreiranja liste	4.1 s	15 s	1101 s

ListFetcher (dobavljač listi)

Uloga ListFetchera je sinhronizacija listi između servera. ListFetcher "dovlači" liste sa nodova koji se prate, odnosno sa kojima je lokalni nod povezan. ListFetcher je zapravo jedan nit koji na svakih n sekundi izvrši jednu iteraciju gde pozove funkciju za dobavljanje listi. Upotrebom ovog algoritma sasvim je nepotrebno da "povezanost" servera bude kružna, jer će podaci sa svih servera postati vidljivi na svim nodovima veoma brzo.

Pseudokod ListFetcher algoritma:

```
konektujemo se na udaljeni host
ako smo konektovani prelazimo u direktorijum /lists
u listu beležimo "liste" koje poseduje udaljeni host
a u drugu listu liste koji mi imamo u lokalu

    prolazimo kroz listu udaljenih za sve elemente
{
     dodeljujemo state = 0
      prolazimo kroz listu lokalnih za sve
                                        elemente {
          ako je (lokalni element = udaljenom eleme-
                ntu i ako (udaljeni
                        element nije naš nod) ) {
```

```
        ako je (lokalni timestamp manji od
                        udaljenog timestampa) {
          state = 1
          dovuci "udaljenu listu"
          obriši lokalno "staru listu"
        }
      } sledeći slučaj (udaljena lista jednaka
        nasem nodu i (lokalna lista
                        jednaka našem nodu) {
        pošalji "našu listu" udaljenom serveru
      }
    }
    ako je (state = 0 i ako (remote lista nije
      jednaka '' i nije jednaka '.' i ako
                        nije jednaka '..') {
      ako je (remote lista nije lista našeg noda) {
        dovuci udaljenu listu
      }
    }
  }
```

Dvosmerna sinhronizacija predstavlja mogućnost klijenta koji nabavlja liste od servera da uploaduje serveru neke svoje liste u slučaju da su novije ili da ih server uopšte nema.

Finalne liste

Kad su sve liste sakupljene kreće se sa generisanjem finalne liste. Ova lista se pravi periodično, a ne po potrebi ili promeni. Uzećemo u obzir činjenicu da krajnjeg korisnika, klijenta, ne zanima gde se podaci nalaze i da želi potpunu transparentnost sistema. Ovo je upravo i razlog pravljenja finalne liste. Ovim omogućujemo da krajnji korisnik na bilo kom od nodova vidi sve fajlove, one koji se nalaze na svim nodovima. Izgled jedne finalne liste:

```
[ /rootdir file,alias,node1,node2;file2,alias,node1;
                                        dir1;dir2; ]
[ /rootdir/dir1 0;          dir3;    ]
```

Mogućnost poklapanja imena fajlova eliminisana je upotrebom aliasa. Zapravo se korisniku ne daje pravo ime fajla osim ako nije unikatno.

```
final list algoritam:
Search (ime fajla) (tražimo od trenutne pozicije)
  ako nema → dodaj fajl → exit.
  ako ima (alias flag = true)
    proveri md5 checksum i size
      isti → dodaj u listu (fajl, alias) → exit;
      nije → traži dalje → next;
```

Generisanje finalne liste je jedan od najsloženijih procesa koje obavlja ovaj sistem. U apstraktnu strukturu podataka, tj. listu učitaju se sve liste koje nod poseduje kao hash tabele. Kad su svi podaci na jednom mestu primenjuje se gore pomenuti algoritam i na kraju se nova lista izvozi u fajl.

FTP VFS (FTP Virtual File System)

FTP VFS je posrednik između fajl sistema i FTP Daemona. Ovaj fajl sistem se snabdeva podacima iz finalnih lista preko FinalListFetcher modula. FTP VFS sesija je jedna lista u koju se stavljaju hash tabele sa 2 ključa. Prvi ključ nazvan je "id" i njegova vrednost ekvivalentna je vrednosti id-a FTPD sesije. Drugi ključ ima ime "dir" i predstavlja trenutni direktorijum u kom se klijent nalazi. Prednost upotrebe FTP VFS-a je što se sve posmatra "virtuelno", čak se i lokalni FS posmatra virtuelno.

```
my @session = ( {   id   = "",
                    dir  = ""  } );
```

FTP VFS služi za isčitavanje podataka iz finalnih lista, beleženje trenutnog virtuelnog direktorijuma u kom se klijent nalazi, kao i obaveštavanje sistema o događajima.

Uzmimo u razmatranje slučaj da se korisnik konektovao na jedan od nodova distribuiranog sistema i da je zatražio fajl koji se nalazi na lokalnom fajl sistemu tog noda. Sistem će pristupiti lokalnom fajl sistemu, uzeti taj fajl i isporučiti ga korisniku. U slučaju da je korisnik zatražio fajl sa udaljenog noda, nod na kom je izvršio zahtev povezaće se na udaljeni nod i dobaviti traženi fajl i isporučiti ga klijentu. Fajl se isporučuje direktno po prispeću sadržaja korisniku.

Load balancer

Kao što je već rečeno, load balancing je veoma bitna osobina ovog sistema. Load balancing se vrši stalno u ListFetcheru. ListFetcher periodično dovlači liste. Tokom tog procesa on meri vreme koje je potrebno da bi se dobavila lista. Ova vremena se beleže lokalno na svakom serveru i vremenom se generiše lista prioriteta. Prvi u listi je najmanje op-

terećen. Sa porastom rednog broja noda u load balansing listi raste i njegova opterećenost.

Load balancing algoritam:

Učitaj load balancing listu
Proveri na kojim nodovima fajl postoji
Proveri u load balancing listi koji od njih je najažurniji i od njega zatraži fajla
Ako je konekcija nemoguća ka tom nodu pređi na sledeći u listi

Pri svakom zahtevu za dobavljanje fajla sistem učitava prethodno generisanu load balansing listu. Load balancing lista se generiše pri dobavljanju lista. Prvi član liste jeste onaj nod sa najvećim prioritetom pristupa. To je nod kojem se najbrže pristupa sa lokalnog noda. Klijent je zatražio fajl, lista je učitana, sledeći korak je provera gde se sve nalazi traženi fajl. Kada je sistem odredio moguće lokacije traženog fajla kreće se sa isporučivanjem fajla. Ukoliko je isporučivanje sa te lokacije nemoguće, bilo da je lokacija nedostupna ili je fajl izbrisan ili pomeren, prelazi se na dobavljanje fajla sa sledećeg noda u novoj load balansing listi koja sadrži samo nodove na kojima se nalaze kopije originala tog fajla.

Cache proxy

Uloga cache proxyja je da čuva podatke dobavljene sa drugih servera i da proverava jesu li oni sveži. U slučaju da se lokalni fajl razlikuje od fajla na lokaciji odakle je dobavljen, cache proxy dobavlja udaljeni fajl. Fajlovi sa lokalnog noda se ne keširaju.

Kad se prvi put izvrši keširanje fajla na lokalni fajl sistem, sistem beleži odakle je fajl dobavljen, sa kog od nodova. Svi keširani fajlovi se smeštaju u jedan poseban direktorijum koji je isključen iz ListFetchera, što znači da server ne popisuje sadržaj cache direktorijuma, niti klijent ima bilo kakvu mogućnost da tom direktorjiumu direktno pristupi. Pristup sadržaju tog direktorijuma moguć je samo preko cache proxyja. Kad klijent zatraži fajl:

proverava se postojanje keširanog fajla istog imena
ako postoji → isporučuje se klijentu
ako ne postoji → prelazimo na sledeći uslov u algoritmu

proverava se da li fajl postoji na lokalnom fajl sistemu uzimajući u obzir identičnost fajla koji se nalazi na udaljenom serveru i lokalnog fajla (prethodno je objašnjeno kako se proverava identičnost fajlova):

ako postoji → fajl se isporučuje
ako ne postoji → fajl se dobavlja sa jednog udaljenih servera za čiji izbor je zadužen Load balancer

Kad korisnik zatraži fajl koji se nalazi na udaljenom nodu, sistem će prvo proveriti da li ima taj fajl na lokalnom fajl sistemu. Ako fajl postoji, server proverava md5 sume i veličine fajlova da bi odredio da li su fajl na lokalnom fajl sistemu i udaljeni fajl isti. Ako su fajlovi isti, počeće se sa isporučivanjem odmah po završetku provere. Ako se razlikuju, server će početi sa dobavljanjem i paralelnim isporučivanjem fajla klijentu.

Ograničenja i mogućnosti proširivosti

Jedna od mana ovog sistema je komplikovana konfiguracija nodova. Mada postoji mogućnost implementacije jednostavnog alata koji bi služio za generisanje konfiguracionih fajlova za sve nodove pojedinačno, taj alat bi trebao imati mogućnost uploadovanja konfiguracionih fajlova na svaki nod. Gui alat pisan uz pomoć QT biblioteke mogao bi da olakša administratorima posao. Konfiguracioni fajlovi bi mogli da se skladište u .xml formatu, čime bi se dodatno dobilo na skalabilnosti, prenosivosti kao i na mogućnosti upotrebe softvera "sa strane" za generisanje konfiguracionih fajlova.

Vreme sinhronizacije između nodova je relativno i zavisi od same konfiguracije distribuirane mreže.

Opisan način rada cache proxyja je jednostavan. On bi se mogao unaprediti tako što bi se fajlovi dobavljali parcijalno. To znači da bi se jedan fajl dobavljao iz 5 celina paralelno, čime bismo dobili efekat transprentnosti rada sistema za kranjeg korisnika.

Zaključak

Upotreba ovog distribuiranog sistema pogodna je kako za male, tako i za veoma obimne mreže jer sistem poseduje veliku skalabilnost u pogledu infrastrukture. Sistem je dizajniran da opslužuje fajlovima, a ne da se na servere uploaduju podaci.

U odnosu na klasičan FTP server, upotreba distribuiranog FTP servera za krajnjeg korisnika je daleko bolja, jer korisnik ima sve fajlove na "jednom mestu".

Dušan Panić

DFTPD – Distributed FTP Daemon

DFTPD is an implementation of the distributed FTP server offering great scalabilbity of its infrastructure. The system can be set-up to be centralized or decentralized. Data synchronization is achieved through the developed two-way synch-mechanism called ListFetcher, which offers additional system integrity. Servers synchronize their file and directory lists among each other.

This system, from the users point of view, offers faster file distribution in comparison with the plain FTP server. Use of the distributed FTP server eliminates the need for FTP mirrors. Implementation of the Load balancer into the DFTPD system produces an effective FTP routing – bandwidth based.