**Distributed file system**

**Modern data mining applications**

Primjeri: Rangiranje web stranica po važnosti

Zajedničko: zahtjeva procesiranje velike količine podataka, podaci su često regularni, ideja je iskoristiti paralelnost

**Datotečni sustav**

Trajna pohrana podataka

Smješteni na vrhu niske fizičke pohrane

Organizirani u file-ove - .txt, podržava hijerarhijsku organizaciju

Putanja=relativni direktorij+ ime file-a

DirB/dirB/file.txt

**Viša razina datotečnog sustava**

Podržava pristup datotekama na udaljenim sustavima

Mora podržavati poklapanje?

Upravlja znatno večim količinama podataka nego OS

Pruža replikaciju/redundanciju podataka:

* Česti kvarovi
* Jeftini kompjuterski čvorovi
* Velika kolekcija čvorova

Većina obrade je napravljena na jednom „čvoru“

**Paralellizam danas**

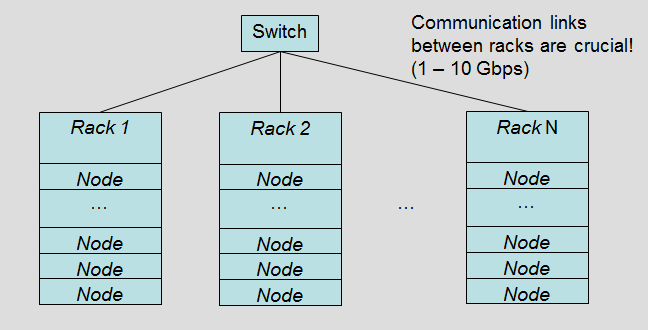
* Rasprostranjenost Web-a
* Obrada je napravljena na instalacijama velikih količina jednostavnih kompjuterskih čvrova
* Tisuće čvorova koji rade nezavisno
* Troškovi su uvelik smanjeni u usporedbi s uporabom posebnih paralelnih računala

Paralelna računalna arhitektura – grupno računanje

8-64 računalnih čvorova unutar jednog „postolja.

Večinom su povezani sa gigabit eternetskim kablom

Komunikacija između postolja 1-10 Gbps – vrlo važno!



**Problemi**

Gubitak pojedinog čvora( zbog rušenja lokalnog diska)

Gubitak cijelog „postolja(rušenje mrežne veze)

**Rješenje**

Podaci se moraji spremati redundantno

Izračuna mora biti podijeljen na zadatke

**DFS(large-scale fyle system organization)**

Fajlovi su ogromne veličine

Rijetko su ažurirani, dodatni podaci su dodani fajlovima

Fajlovi su podijeljeni na komade(64 MB)

Komadi su replicirani na nekoliko čvorova koji su u odvojenim postljima

Za svaki fajl, postoji master čvor, sa svojom replikacijom

Primjeri DFS: Google file system, hadoop, cloudstore

**Google file system**

Dijeli mnoge zahtjeve sa drugim DFS

* Performanse
* Skalabilnost
* Dostupnost
* Pouzdanost

**Pretpostavke dizajna**

Visoka stopa urušavanja čvorova – velika kolekcija jeftinog i nepouzdanog hardvera

Umjeren broj izuzetno velikih fajlova – nekoliko milijuna fajlova, svaki 100 MB ili veći, uobičajeno gigabajti

Velik broj streaming čitanja, i mal broj random čitanja? What?

**Dizajn odluke**

Podaci su spremljeni u komade – 64MB

Pouzdanost se postiže replikacijom – svaki komad je na 3 servera

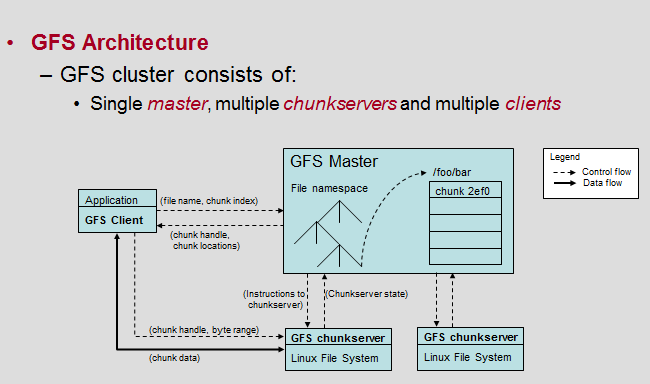
Jedan „vlasnik“ odgovoran za pristup i metapodatke

Nema „keširanja“ podataka

Poznato sučelje datotečnog sustava + Google API – podržava operacije create, delete, open, close, read, vrite

Custon google api uvodi snapshot i record append operacije

GFS ARHITEKTURA



Veličina „komada“ je 64 MB

**Prednosti**

Reducira klijent-maser komunikaciju

Reducira mrežno opterećenje održavajući perzistentnu TCP vezu sa serverom tijekom dužeg razdoblja

Reducira veličinu metapodataka spremljenih na masteru

**Nedostatci**

Mal fajlvse sastoji od nekoliko komada(često 1)

Chunk serveri koji spremaju te podatke su potencijalno „usko grlo“

**Metapodaci**

Spremljeni na masteru

In-memory strukture podataka – 64 byte-a po komadu, brze operacije, jednostavnost performanse, fleksibilnost, pouzdanost

Checkpoint – b-stablo forma, može biti lako mapiran u memoriju, 1 minuta za napraviti potrebna, za nekoliko milijuna fajlova

**Master operacije**

Spremanje metapodataka

Upravljanje prostorom i zaključavanje

Periodička komunikacija sa chunk-serverima

Postavljanje replika

Kreiranje komada, re-replikacija, rebalansiranje

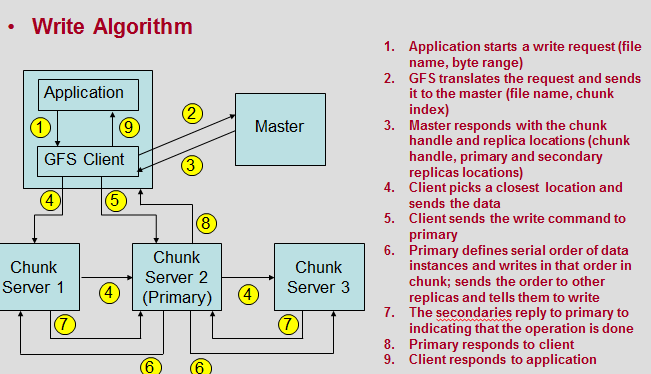
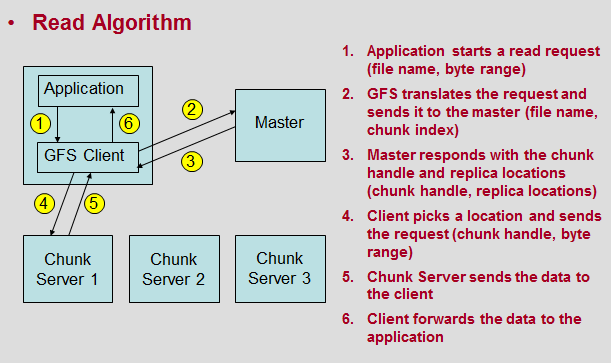
Skupljanje smeća

Detekcija starih replika

**Mutacije**

Zahtjevi Random zapisivanja ili dodavanja zapisa

Generalna ideja je smanjiti-minimizirati uključenost master čvora

Protok podataka je odvojen od kontrolnog protoka

**Atomic Record Append**

GFS dodaje podatke u fajl atomički barem jednom

Klijent push-a upisane podatke na sve lokacije

Primarno se provjerava jel podatak odgovara određenom komadu podataka

Ako ne:

Postavlja chunk na maximalnu veličinu

Kaze drugom da isto to napravi

Informira klijenta da proba na sljedećem chunku

Ako da:

Dodaje podataka na taj chunk

Govori sekundarnom da napravi isto

Prima odgovor od sekundarnog i odgovara klijentu

**Relaxed Consistency Model**

Svi klijenti vide iste podatke neovisno iz koje replike čitaju – consistent region

Defined region – consistenti I klijenti vide što je mutacija upisala

Stanje određene regije ovisi o tipu mutacije I ako postoji konkurentnih mutacija ili ne

**Zaključak**

Postoji mnogo GFS clustera

Stotine/tisuće pohranjivanja na svakom čvoru

GFS podržava large-scale procesiranje

**Hadoop file system**

Namenodes I datanodes

Postoje: klijent, namenode(master), datanodes – pohranjuju I vraćaju blokove podataka, prijavljuju namenode-u koje sve blokove podataka sadrže

**Razlike od GFS**

Samo jedan zapis po fajlu

128 MB blok podataka

2 fajla po zapisu(data+checksum)

Nema record append operaciju

Hadoop je open source, omogućava mnoga sučelja i libraryje za raličite sustave