Big Data i jezici za upite

Zekić, Martin

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:868877

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-06-23**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Sveučilište Juraj Dobrila, Pula

Fakultet informatike

MARTIN ZEKIĆ

BIG DATA I JEZICI ZA UPITE

Završni rad

Rujan, 2021.

Sveučilište Jurja Dobrile u Puli

Fakultet informatike

MARTIN ZEKIĆ

BIG DATA I JEZICI ZA UPITE

Završni rad

JMBAG: 0112064482

Studijski smjer: Informatika

Predmet: Baze podataka 2

Znanstveno područje: Društvene znanosti

Znanstveno polje:

Znanstvena grana:

Mentor: doc. dr. sc.Siniša Miličić

Rujan, 2021.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Martin Zekić, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraţivanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli,_____,_____godine

IZJAVA

o korištenju autorskog djela

Ja, Martin Zekić dajem odobrenje Fakultetu informatike u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom Big data i jezici za upite koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjiţnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli,_____(datum)

Potpis

_____

Sažetak

U ovom će se radu prezentirati Big Data te alati MongoDB, Apache Hadoop, Hive, JAQL, Bigtable, Cassandra i Pig koji se koriste za manipulaciju sa tako ogromnom količinom strukturiranih I nestrukturiranih podataka. Prikazat će se funkcije Map i Reduce, odnosno MapReduce po jezicima za upite. Na kraju rada će se prikazati zadatak sa korištenjem indeksa, te MapReduce funkcije koji je djelo autora rada

Ključne riječi: Big Data, strukturirani podaci, nestrukturirani podaci, MapReduce, MongoDB, Cassandra, Apache Hadoop, Hive, Pig latin, JAQL, Bigtable, Apache Sparks

Abstract

This thesis presents BigData together with the tools MongoDB, Apache Hadoop, Hive, JAQL, Bigtable, Cassandra and Pig which are used for manipulating big amounts of structured and unstructured data. Functions Map and Reduce will be presented, that is, MapReduce by query languages. At the end of this thesis, the task with the use of the index and the MapReduce function will be presented, which is the work of the author.

Key words: Big Data, structured data, unstructured data, MapReduce, MongoDB, Cassandra, Apache Hadoop, Hive, Pig latin, JAQL, Bigtable, Apache Sparks

Sadržaj

Uvod

Upravljanje i analiza podataka oduvijek je predstavljala najveći izazov za sve organizacije u svim poljima industrije. Poduzeća su se dugo borila da pronađu najbolji način za sakupljanje informacija o svojim klijentima, proizvodima i uslugama. Nekada kada su kompanije imale mali broj kupaca koji su kupovali isti proizvod stvari su bile jasne i jednostavne. Vremenom, kompanije su porasle pa je stvar postala kompleksnija i teža nego prije.

Problemi oko podataka nisu ograničeni samo na polju poduzeća. Na primjer, organizacije koje se bave razvojem i istraživanjem imale su problem zbog nedovoljno financijske moći da bi pokrenuli sofisticirane modele ili obradile slike i druge izvore znanstvenih podataka. Neki podaci su strukturirani i sačuvani u relacijskim bazama podataka dok su drugi podaci, uključujući dokumente, slike i videozapise, nestrukturirani. Kompanije također moraju razmotriti nove izvore podataka koje generiraju uređaji poput senzora, kamera, i ostalog. Druge ogromne količine podataka su one koje generira gotovo svaka osoba, a to su podaci iz društvenih mreža i click-stream podaci sa raznih web stranica i aplikacija. Pored svega toga, dostupnost moćnijih mobilnih uređaja, uz stalan pristup internetu donosi velik izvor novih podataka.

Iako se svaki izvor podataka može nezavisno upravljati i pretraživati, trenutno je najveći izazov pronaći nekakav smislen presjek svih tih podataka koji su različitog tipa. Kada imamo toliko informacija u toliko različitih oblika, nemoguće je upravljati podacima na tradicionalan način. Organizacije sada, više nego ikada prije, pronalaze način da iskoriste sve informacije. Odnosno, o upravljanju podacima mora se misliti drugačije i to je izazov, a ujedno i šansa za big data-u. Big data se može definirati kao bilo koja vrsta izvora podataka koja ima najmanje tri zajedničke karakteristike:

1. Izuzetno velika količina podataka
2. Izuzetno velika brzina podataka
3. Izuzetna raznolikost podataka

Big data je važna zato što omogućava organizacijama da sakupljaju, upravljaju i obrađuju velike količine podataka velikom brzinom. To nije samostalna tehnologija, nego je to kombinacija posljednjih 50 godina evolucije tehnologije.

Big data

Big data ne predstavlja jedinstvenu tehnologiju, već kombinaciju novih i starih tehnologija koje pomažu kompanijama steći djelotvoran uvid u obrađene podatke. Odnosno, to je mogućnost upravljanja velikim količinama različitih podataka razumnom brzinom i u odgovarajućem vremenskom okviru da bi se omogućila analiza tih podataka u realnom vremenu. Big data tehnologije se ne razmatraju bez osvrta na V-ove. Različiti autori preferiraju različite V-ove, ali zajednički elementi su ovih pet V:

Količina (Volume):

Sam naziv 'Big Data' povezan je s veličinom koja je ogromna. Volumen je ogromna količina podataka. Da bi se utvrdila vrijednost podataka, veličina podataka igra vrlo presudnu ulogu. Ako je opseg podataka jako velik, onda se to smatra „velikim podacima“. To znači može li se određeni podatak zaista smatrati velikim podatkom ili ne, ovisi o količini podataka. Stoga, dok se bavimo velikim podacima, potrebno je razmotriti karakteristični "volumen".

Primjer: Procijenjeni globalni mobilni promet u 2016. godini iznosio je 6,2 egzabajta (6,2 milijarde GB) mjesečno. Također, u 2020. godine imamo gotovo 40000 egzabajta podataka.

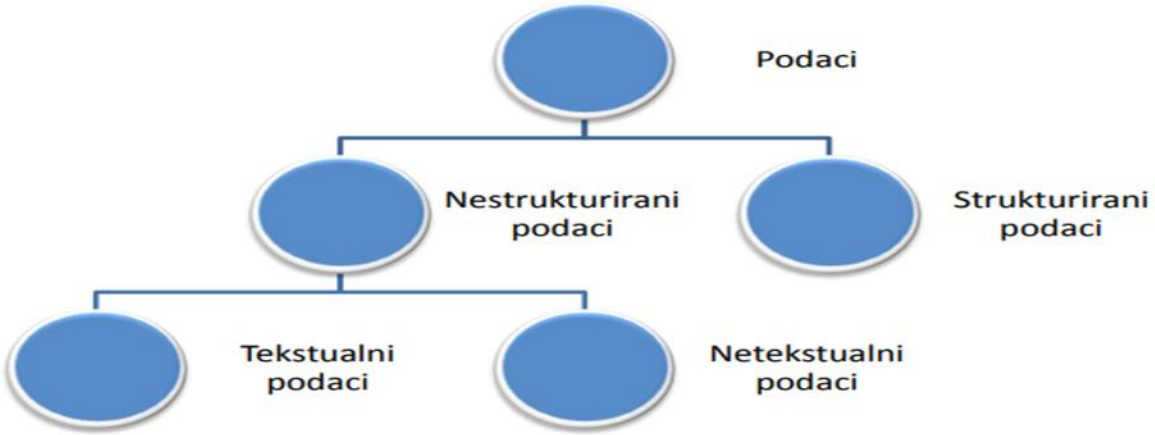
Nitko zapravo ne zna koliko se novih podataka generira, ali količina prikupljenih informacija je ogromna.

Raznolikost (Variety):

Raznolikost je jedno od najzanimljivijih dostignuća u tehnologiji jer se sve više i više informacija digitalizira. Tradicionalne vrste podataka (strukturirani podaci) uključuju stvari na bankovnom izvodu poput datuma, iznosa i vremena. To su stvari koje se lijepo uklapaju u relacijsku bazu podataka.

Strukturirani podaci povećavaju se nestrukturiranim podacima, gdje se stavljaju stvari poput Twitter feed-ova, audio datoteka, MRI slika, web stranica, web dnevnika. Sve što se može uhvatiti i pohraniti, ali nema meta model (skup pravila za uokvirivanje koncepta ili ideje - on definira klasu informacija i način na koji ih treba izraziti) koji ih uredno definira.

Nestrukturirani podaci temeljni su pojam velikih podataka. Najbolji način za razumijevanje nestrukturiranih podataka je usporedba sa strukturiranim podacima. Strukturirane podatke smatramo podacima koji su dobro definirani u nizu pravila. Na primjer, novčanice će uvijek biti brojevi i imati barem dvije decimale, imena su izražena kao tekst, a datumi slijede određeni obrazac. S druge strane, s nestrukturiranim podacima ne postoje pravila. Slika, snimka glasa, tweet - svi oni mogu biti različiti, ali izražavaju ideje i misli temeljene na ljudskom razumijevanju. Jedan od ciljeva Big data-e je korištenje tehnologije za uzimanje i razumijevanje tih nestrukturiranih podataka.



Slika 1: Prikaz vrste podataka

Brzina (Velocity):

Brzina je učestalost dolaznih podataka koje treba obraditi. U Big data podaci o brzini dolaze iz izvora kao što su strojevi, SMS poruke, društvene mreže, mobiteli, tableti i tako dalje. Postoji ogroman i kontinuiran protok podataka. To određuje potencijal podataka, odnosno koliko brzo se podaci generiraju i obrađuju kako bi udovoljili zahtjevima. Podaci uzorkovanja mogu pomoći u rješavanju problema poput "brzine". Amazon Web Kinesis primjer je aplikacije koja obrađuje brzinu podataka.

Primjer: Dnevno se izvrši više od 3,5 milijardi pretraživanja na Google-u. Također, korisnici Facebook-a iz godine u godinu povećavaju se za 22% (približno).

Istinitost (Veracity):

Istinitost se odnosi na neurednost ili pouzdanost podataka. S različitim oblicima velikih podataka, kvaliteta i točnost se mogu manje kontrolirati (sjetimo se samo postova na Twitter-u s hash oznakama, kratica, tipkarskih pogrešaka i razgovornog govora, kao i pouzdanosti i točnosti sadržaja), ali tehnologija velikih podataka i analitike sada nam omogućuje rad s ovom vrstom podataka. Količine često nadoknađuju nedostatak kvalitete ili točnosti u svim prikupljenim podacima.

Vrijednost (Value):

I zadnji V koji se mora uzeti u obzir prilikom promatranja velikih podataka je vrijednost.

Ukoliko je sve u redu s pristupom velikim podacima, ali ih ne možemo pretvoriti u vrijednost, postaju nam beskorisni. Dakle, možemo sa sigurnošću tvrditi da je 'vrijednost' najvažniji V velikih podataka.

Važno je da tvrtke dokažu poslovni slučaj za svaki pokušaj prikupljanja i iskorištavanja velikih podataka. Lako je upasti u zamku i započeti inicijativu za velike podatke bez jasnog razumijevanja troškova i koristi.

Upotreba

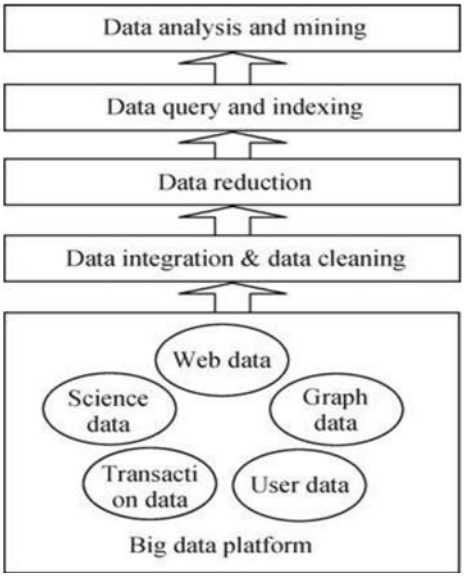
Razvoj tehnologija koje se koriste za obradu velike količine podataka pridonio je razvoju za određena područja u kojima se takve analize mogu koristiti. Na primjer, može se vidjeti velik napredak u prometu i zdravstvu. U zdravstvu se može pratiti broj prijevremeno rođene djece i u zavisnosti od dobivenih podataka procjenjivati kada je potrebna neka određena intervencija. U prometu je analizom velike količine podataka koje generiraju kamere sa autocesti, moguće predvidjeti i regulirati gužvu. Također, moguće je smanjiti broj prometnih nesreća, štedjeti gorivo te voditi računa o zagađenju.

Ipak, glavni problem ne predstavlja prikupljanje velikih količina podataka, već izvlačenje korisnih informacija iz tih podataka. Današnje tehnologije ne samo da podržavaju spremanje ovih podataka, već daju mogućnost da se iskoristi vrijednost tih podataka. To pomaže organizacijama da poprave svoje poslovanje i profit. Na primjer, uz pomoć ovih tehnologija moguće je:

- Analizirati milijune proizvoda da bi se odredila neka optimalna cijena, povećao profit ili oslobodilo skladište;
- Preračunavati rizike u minuti i na taj način se prilagođavati promjenama;
- Slati adekvatne ponude mobilnih operatera u pravom trenutku (kada bi ih korisnik mogao iskoristiti na najbolji način);
- Istraživanje podataka vezanih za potrošačke navike i potrebe i na taj način povećanje profita, podrške u izbornim kampanjama.

Klasični primjeri generiranja velikih količina podataka:

- Više od 5 milijardi ljudi zove, šalje poruke, tweet-a i surfa internetom preko mobilnih uređaja;
- Facebook ima više od 900 milijuna aktivnih korisnika koji svakodnevno generiraju podatke međusobnom komunikacijom i statusima;
- Svake sekunde obavi se 10 000 transakcija plaćanja kreditnom karticom u svijetu
- Dnevno se pošalje 340 milijuna tweet-ova, što je 4000 tweet-a po sekundi.



Slika 2: Tok podataka

Alati koji se koriste za upravljanje Big Datom

MapReduce

MapReduce je visoko skalabilna programska paradigma sposobna za obradu ogromnih količina podataka. Nedavno ga je popularizirao Google, iako je danas MapReduce implementiran u puno open source projekata, a najistaknutiji je Apache Hadoop. Popularnost MapReducea temelji se na visokoj skalabilnosti, toleranciji na grešku, jednostavnosti i neovisnosti o programskom jeziku ili sustavu za pohranu podataka.

U MapReduce paradigmi, funkcija Map izvodi filtriranje i sortiranje, dok funkcija Reduce vrši grupiranje i operacije agregacije.

Primjer MapReduce

Brojanje riječi jednostavna je aplikacija MapReduce-a koja se obično koristi u demonstracijske svrhe. Map objedinjuje popis nizova (po jedan u retku) kao mape i svakom ključu dodjeljuje proizvoljnu vrijednost. Funkcija Reduce ponavlja se tijekom ovog skupa, povećavajući brojač za svaku pojavu ključa.

```
map( 'input.dat' ){
 Tokenizer tok <- file.tokenize();
   while (tok.hasMoreTokens){
     output(tok.next(),"1"); // list(k2,v2)
   }
}

reduce( word, values ){
 Integer sum = 0;
   while(values.hasNext()){
     sum += values.next();
   }
 output(word,sum); // list(v3)
}
```

Slika 3: Primjer funkcija map i reduce

Hadoop

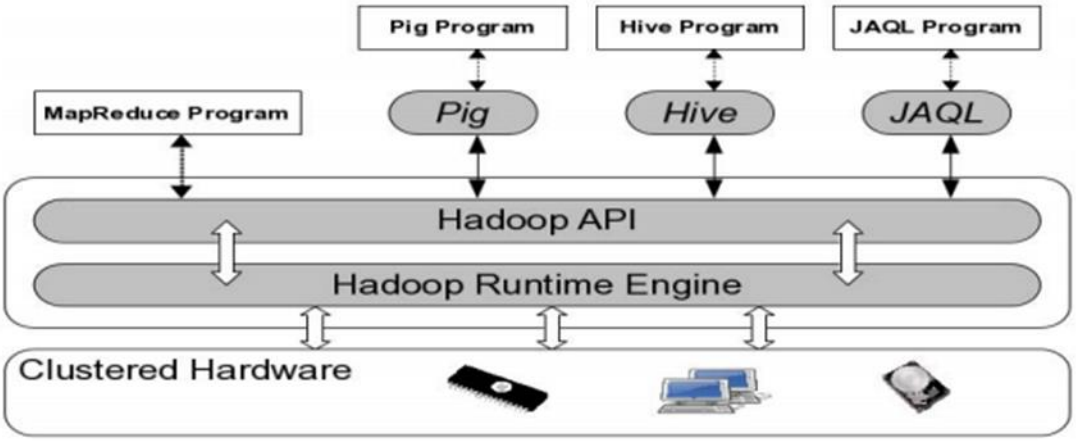
Hadoop je Apache-ova open source MapReduce implementacija, koja je vrlo pogodna za upotrebu u velikim skladištima podataka. Hadoop je opremljen kompletom besplatnih usluga i apstrakcijama više razine iz MapReduce-a. Ključni elementi Hadoopa su MapReduce – distribuirani model obrade podataka i okruženja izvršenja i Hadoop Distributed Filesystem (HDFS) – distribuirani sustav datoteka koji radi na velikim klasterima. HDFS omogućuje pristup protokolu podataka velike propusnosti, pogodan je za aplikacije koje imaju velike skupove podataka,a oporavak od kvarova je primarni cilj dizajna HDFS-a. Osim toga Hadoop nudi sučelje za aplikacije za premještanje zadataka bliže prema podacima, jer je premještanje računanja jeftinije od premještanja velikih količina podataka. Open source je i temelji se na Java-i te pruža podršku na više platformi. Neka od velikih imena koja korise Hadoop su Amazon Web usluge, IBM, Microsoft te mnogi drugi.

Prednosti:

- Osnovna snaga Hadoop-a je HDFS (Hadoop distribuirani sustav datoteka) koji ima mogućnost pohrane svih vrsta podataka - videozapisa, slika, JSON-a, XML-a i običnog teksta u isti datotečni sustav;
- Izuzetno koristan za potrebe istraživanja i razvoja;
- Omogućuje brzi pristup podacima;
- Vrlo skalabilno;
- Visoko dostupna usluga koja počiva na grupi računala.

Mane:

- Ponekad se mogu suočiti s problemima s prostorom na disku zbog toga što zauzimaju 3 puta veći prostor;
- Ulazno-izlazne operacije mogle su se optimizirati za bolje performanse.



Spark

Apache Spark započeo je kao istraživački projekt na UC Berkeley u AMPLabu, istraživanje je započeto s ciljem dizajniranja programskih model koji podržavaju širu klasu aplikacija nego MapReduce, uz zadržavanje svoje automatske tolerancije na pogrešku. Spark nudi apstrakciju nazvanu Elastically Distributed Datasets (RDD -ovi) za učinkovitu podršku ovim aplikacijama. RDD -ovi se mogu pohraniti u memoriji između upita bez potrebne replikacije. Umjesto toga, obnavljaju izgubljene podatke o neuspjehu koristeći liniju: svaki RDD pamti kako je izgrađen drugi skup podataka (transformacijama poput map, pridruživanja ili grupe) da se obnovi. RDD -ovi omogućuju Spark -u da nadmaši postojeć MapReduce model do 100x u višeprolaznoj analitici. RDD -ovi mogu podržavati veliki broj iterativnih algoritama, kao i interaktivno rudarenje podacima i visoko učinkovit SQL stroj Shark.

Pig

Pig je platforma koja služi za analizu velikih setova podataka te se radi o višem jeziku kojem je svrha upravo takva analiza podataka, no potrebna mu je i infrastruktura koja mu pomaže u procesu. Paralelizacija je temelj Piga-a te mu pomaže u ophođenju s velikim podacima. Pig-ov kompajler proizvodi MapReduce zadatke koje smo već spominjali.

Pig latin (Pig-ov jezik) značajke :

- Jednostavnost - bilo da se govori o jednostavnijim ili nešto složenijim zadacima koji se pretvaraju u „data flow sequences“, dobivamo jednostavnost pisanja, razumijevanja te održavanja;

- Optimizacija – načinom na koji se zadaci rade dopušta se sustavu da automatski optimizira izvršenje te se tako pruža korisnicima veći fokus na semantiku umjesto na efikasnost;

- Proširivost – korisnik može kreirati svoje funkcije koje imaju posebnu namjenu.

Njegov cilj je pojednostavljenje procesa i izvršavanja Map i Reduce zadataka. Dakle, za Pig su bitni PigLatin (njegov jezik) te okruženje koje mu omogućava izvršavanja. Neki koraci koji mu omogućuju navedeno su : LOAD (učitava podatke), TRANSFORM (manipulacija podacima), DUMP (prikaži rezultat na ekranu) te STORE (spremi rezultat za daljnju analizu).

```
myinput = LOAD 'input.dat' USING PigStorage();
grouped = GROUP myinput BY \$0;
counts =  FOREACH grouped GENERATE group,
          COUNT(myinput) AS total;
STORE counts INTO 'PigOutput.dat' USING PigStorage();
```

Slika 5: MapReduce u Pig latin jeziku za upite

Hive

Hive je poznato mjesto za analitičare podataka zbog podržavanja upita u jeziku koji su jako slični SQL-u. No ipak postoje određene razlike i ograničenja ovog jezika. Hive-ov jezik za upite naziva se Hive Querly Language (HQL), te je osmišljen kako bi olakšao posao ljudima koji se već godinama bave SQL-om, kako bi se lakše snašli u ovoj Big Data okolini.

Neke od značajki iz SQL-a: from, join, group by, agregacije ,create table, select all.

Hive strukturira podatke u dobro razumljive koncepte baze podataka poput tablica, stupaca, redaka i particija. Podržava sve glavne primitivne tipove: int, float, double i

string-ove, kao i tipove kolekcija kao što su mape, liste i strukture. Hive također sadrži sistemski katalog, meta-trgovinu koja sadrži scheme i statistike koje su korisne u istraživanju podataka i optimizaciji upita. Kao i Pig, Hive uključuje kompajler upita koji kompajlira HQL u graf zadatka MapReduce-a.

```
CREATE EXTERNAL TABLE Text(words STRING)
LOCATION 'input.dat';
FROM Text INSERT OVERWRITE DIRECTORY 'HiveOutput.dat'
SELECT words, COUNT(words) as totals
GROUP BY words;
```

Slika 6: MapReduce u Hive jeziku za upite

JAQL

JAQL – JSON sučelje za MapReduce. JAQL je funkcijski tip jezika za upite koji se temelji na JavaScript Object Notation Language-u (JSON). JAQL je jezik protoka podataka opće namjene koji manipulira polustrukturiranim infomacijama u obliku apstraktnih JSON vrijednosti. Pruža framework za čitanje i pisanje podataka u prilagođeni format, i pruža podršku za uobičajene ulazno/izlazne formate poput CSV-a. Isto kao Pig i Hive sadrži operatore za filtriranje, sortiranje, group by, agregacije i join. JSON podržava vrijednosti poput brojeva i nizova, te sadrži dva spremnika.

Baze podataka i programski jezici trpe zbog neusklađenosti jer su njihovi računski modeli i i modeli podataka toliko različiti. Kako JSON model omogućuje laku migraciju podataka na neke i iz nekih popularnih skriptnih jezika poput JavaScript-a i Python-a, JAQL je proširen sa operacijama iz više programskih jezika pošto JSON ima puno neusklađenosti sa na primjer XML-om, ali je puno bogatiji sa tipovima podataka nego relacijskih tablicama.

```
$input = read(lines('input.dat'));
$input -> group by $word = $
into { $word, num: count($) }
->write(del('JAQLOutput.dat',{fields:[''words'',''num'']}));
```

Slika 7: MapReduce u JAQL jeziku za upite

Bigtable

Bigtable je dizajniran za podršku aplikacijama koje zahtijevaju veliku skalabilnost. Od svoje prve interakcije, tehnologija je namijenjena za korištenje sa petabajt podacima. Baza podataka je dizajnirana za implementaciju na klasterirane sustave i koristi jednostavan model podataka koji je Google opisao kao "rijetku, distribuiranu i postojanu višedimenzionalnu sortiranu kartu". Podaci su sastavljeni redoslijedom po retku ključa, a indeksiranje karte je uredeno prema redoslijedu, tipkama stupaca i vremenskim oznakama. Algoritmi kompresije pomažu u postizanju velikog kapaciteta. Google Bigtable služi kao baza podataka za aplikacije kao što su Google App Engine Datastore, Google personalizirano pretraživanje, Google Earth i Google Analytics. Google je zadržao softver kao svoju tehnologija. Bez obzira na to, Bigtable je imao veliki utjecaj na dizajn SQL (Structured Query Language) baze podataka. Google-ovi programeri softvera javno su objavili detalje Bigtable-a u tehničkom radu predstavljenom na USENIX simpoziju o operativnim sustavima i implementaciji dizajna 2006. godine. Google-ov temeljit opis unutarnjih funkcioniranja Bigtable-a omogućio je drugim organizacijama i razvojnim timovima open source-a da stvore derivate Bigtable-a, uključujući bazu Apache HBase, koja je izgrađena za pokretanje na vrhu Hadoop distribuiranog sustava datoteka (HDFS). Drugi primjeri uključuje Cassandra-u, koja je nastala na Facebook Inc i Hypertable, tehnologiji open source koja se prodaje u komercijalnoj verziji kao alternativa HBase-u.

Primjer MapReduce zadatka:

Uzorak koda nudi jednostavno sučelje naredbenog retka koje za unos uzima jednu ili više tekstualnih datoteka i naziv tablice, pronalazi sve riječi koje se pojavljuju u datoteci i broji koliko se puta svaka riječ pojavi. Logika MapReduce pojavljuje se u klasi WordCountHBase.

Prvo, Map tokenizira sadržaj tekstualne datoteke i generira parove ključ/vrijednost, gdje je ključ riječ iz tekstualne datoteke, a vrijednost je 1.

```java
public static class MyTableReducer extends
    TableReducer<ImmutableBytesWritable, IntWritable, ImmutableBytesWritable> {

  @Override
  public void reduce(ImmutableBytesWritable key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
    int sum = sum(values);
    Put put = new Put(key.get());
    put.addColumn(COLUMN_FAMILY, COUNT_COLUMN_NAME, Bytes.toBytes(sum));
    context.write(null, put);
  }

  public int sum(Iterable<IntWritable> values) {
    int i = 0;
    for (IntWritable val : values) {
      i += val.get();
    }
    return i;
  }
}
```

Slika 8: Prikaz reduce funkcije

Reduce zatim zbraja vrijednosti za svaki ključ i zapisuje rezultate u tablicu Bigtable koju smo naveli. Svaka ključna riječ retka dolazi iz tekstualne datoteke. Svaki redak sadrži stupac cf: count koji sadrži broj pojavljivanja ključa retka u tekstualnoj datoteci.

```java
public static class TokenizerMapper extends
    Mapper<Object, Text, ImmutableBytesWritable, IntWritable> {

  private final static IntWritable one = new IntWritable(1);

  @Override
  public void map(Object key, Text value, Context context) throws IOException,
      InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    ImmutableBytesWritable word = new ImmutableBytesWritable();
    while (itr.hasMoreTokens()) {
      word.set(Bytes.toBytes(itr.nextToken()));
      context.write(word, one);
    }
  }
}
```

Slika 9: Prikaz map funkcije

Cassandra

Apache Cassandra je besplatan i distribuiran NoSQL DBMS otvorenog koda izrađen za upravljanje ogromnim količinama podataka raširenih na brojnim serverima, pružajući visoku dostupnost. Za interakciju s bazom podataka koristi CQL (Cassandra Structure Language).

CQL nudi model sličan SQL-u. Podaci se pohranjuju u tablice koje sadrže redove stupaca. Iz tog razloga, kada se koriste u ovom dokumentu, ti izrazi (tablice, retci i stupci) imaju istu definiciju koju imaju u SQL-u. Široko je naklonjena svojim poslovnim značajkama, poput skalabilnosti i velike dostupnosti, koje omogućuju za obradu velikih količina podataka i za analizu u stvarnom vremenu. Napisana na Java-i, Cassandra nudi sinkronu i asinhronu replikaciju za svako ažuriranje. Njegova izdržljivost i mogućnosti otpornosti na greške čine ga idealnim za aplikacije koje uvijek rade.

Prednosti i primjene Cassandre

Za razliku od MongoDB-a, Cassandra koristi "ring" arhitekturu bez mastera koja pruža nekoliko prednosti u odnosu na naslijeđene arhitekture poput master-slave arhitekture. To znači da se svi čvorovi u klasteru tretiraju jednako.

Iako Cassandra pohranjuje podatke u stupce i retke poput tradicionalnog RDBMS-a, pruža agilnost u smislu da dopušta retcima različite stupce, pa čak i promjenu formata stupaca.

Preporučena konfiguracija klastera prekriva Hadoop preko Cassandra-e. To uključuje instaliranje Hadoop TaskTracker-a na svaki čvor Cassandra-e. Također jedan server u klasteru trebao bi biti posvećen sljedećim Hadoop komponentama:

- JobTracker,
- Datanode,
- Namenode.

Ovaj namjenski server potreban je jer Hadoop koristi HDFS za pohranu JAR ovisnosti za naš posao, statičke podatke i druge potrebne podatke. U cjelokupnom kontekstu našeg klastera, ovo je vrlo mala količina podataka, ali je ključna za pokretanje MapReduce.

Hadoop TaskTrackers i Cassandra nodes

Pokretanje Hadoop TaskTracker-a na čvoru Cassandra zahtijeva da ažurirate HADOOP_CLASSPATH u <hadoop> /conf/hadoop-env.sh kako biste uključili biblioteke Cassandra-e. Na primjer, dodajte unos poput ovoga u hadoop-env.sh na svaki od čvorova za pracenje funkcije:

export HADOOP_CLASSPATH = / opt / cassandra / lib / *: $ HADOOP_CLASSPATH

Kada Hadoop TaskTracker radi na istim poslužiteljima kao i čvorovi Cassandra-e, svakom TaskTracker-u se šalju zadaci samo za podatke koji pripadaju rasponu tokena u lokalnom čvoru Cassandra-e. To omogućuje ogroman dobitak u učinkovitosti i vremenu obrade, budući da čvorovi Cassandra-e primaju samo upite za koje su primarna replika, izbjegavajući dodatne troškove protokola Gossip (“Peer to peer” protokol koji se bazira na tome kako se epidemija širi).

U procesu MapReduce, Cassandra-ini retci ili fragmenti redaka (parovi ključeva + SortedMap stupaca) mogu se unijeti u funkciju Map za obradu, kako je navedeno u SlicePredicate koji opisuje koje stupce treba dohvatiti iz svakog retka. Na primjer:

```
ConfigHelper.setColumnFamily(job.getConfiguration(), KEYSPACE, COL
SlicePredicate predicate = new SlicePredicate().setColumn_names(Ar
ConfigHelper.setSlicePredicate(job.getConfiguration(), predicate);
```

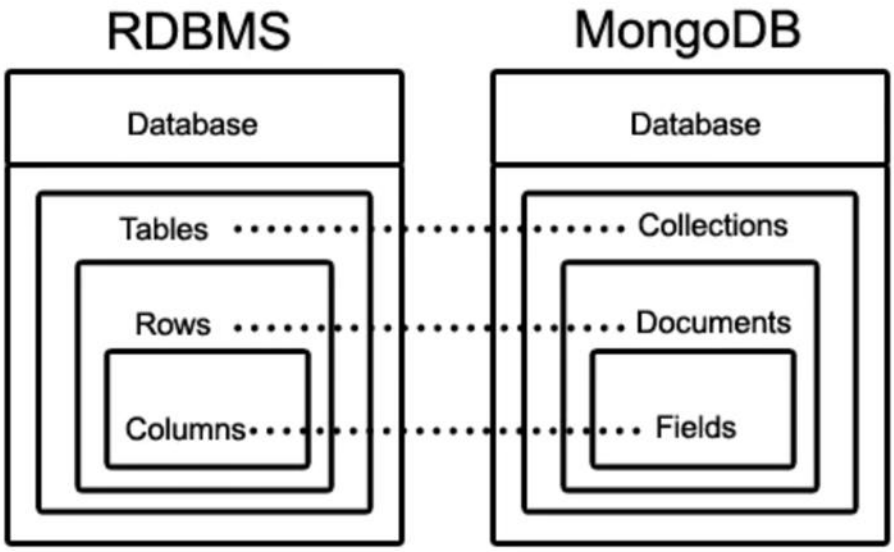
Slika 10: Dohvačanje stupaca

Neke od istaknutih tvrtki koje koriste Cassandra-u uključuju Netflix, American Express, Facebook, General Electric, Honeywell, Yahoo i mnogi drugi.

MongoDB

MongoDB je jedan od vodećih sustava za upravljanje bazama podataka. Stvorio ga je 2007. godinne tim iza DoubleClick-a (trenutno u vlasništvu Google-a) za rješavanje problema skalabilnosti i agilnosti u posluživanju DoubleClick internetskih oglasa. Postoji društvena i poslovna verzija softvera.

MongoDB je baza podataka bez scheme i pohranjuje podatke kao dokumente slične JSON-u (binarni JSON odnosno BSON). To pruža fleksibilnost i agilnost u vrsti zapisa koji se mogu pohraniti, pa čak i atributi mogu varirati od dokumenta do dokumenta. Također, ugrađeni dokumenti pružaju podršku za brže upite putem indeksa i uvelike smanjuju preopterećenje.



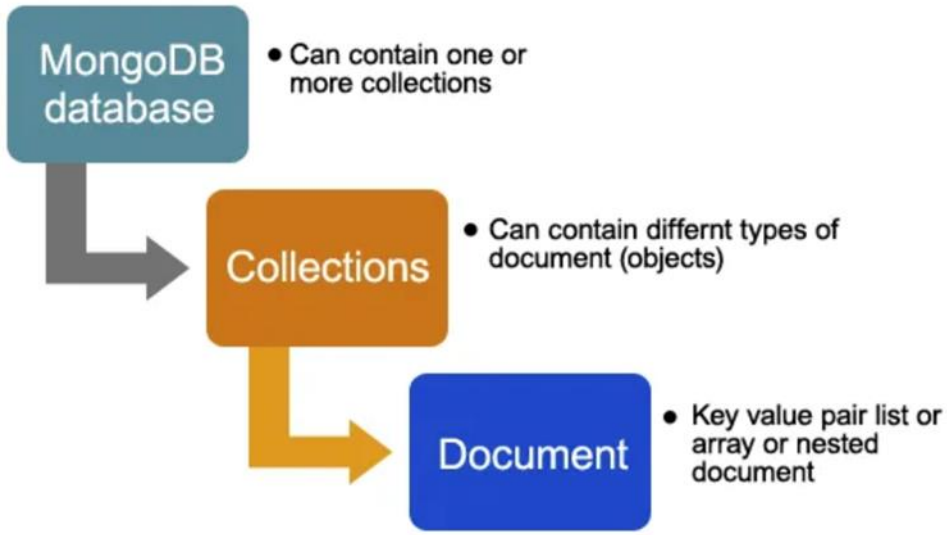
Slika 11: Usporedba relacijske baze podataka sa mongovom bazom

Pojmovi vezani za MongoDB koji se koriste tijekom uporabe:

- Kolekcija: skupina dokumenata u MongoDB bazi podataka. Ovo se može smatrati tablicama u RDBMS-u. Ova kolekcija ne nameće nijednu strukturu. Stoga je MongoDB bez schema, odnosno svaki dokument moze biti jedinstven.

- Dokument : dokument je zapis u MongoDB kolekciji. Sastoji se od atributa.

• Atribut: ime i vrijednosti u dokumentu. Dokument ima nula ili više atributa. Atributi su poput stupaca u relacijskim bazama podataka.



Hijerarhija zbirke

Slika 12: Hijerarhija MongoDB baze podataka

Primjer pretraživanja preko indexa

Primjer pretraživanja preko indeksa započinjemo

```
test> for(i= 0; i < 500000; i++){
      db.testdb.insert(
      {"ime":"ime"+i,
      "godine":Math.floor(Math.random() * 90)});}
```

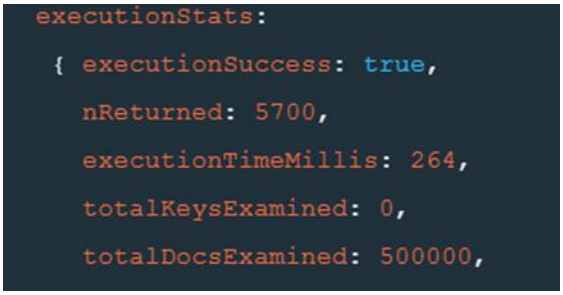
Slika 13: for petlja i insert funkcija

Preko for petlje smo unijeli 500 tisuca datoteka u našu kolekciju testdb:

```
db.testdb.find({"godine": 25}).explain("executionStats")
```

Slika 14: find i explain funkcije

Nakon toga radimo upit koji nam vraća dokumente u kojima su godine postavljene na 25, sa explain-om provjeravamo koliko je milisekundi trebalo našem upitu da se izvrši, koliko je dokumenata sa tim brojem godina pronašao i najvažnije koliko dokumenata se moralo pregledati kako bi se izvršio upit:



Slika 15: statistike gore navedenog upita

Na ovoj slici vidimo da su sve datoteke pregledane, odnosno svih 500 tisuća što nije najoptimalnije riješenje. Vidimo da mu je za izvrsavanje upita trebalo 264 milisekundi.

Kako bi riješili taj problem MongoDB nam nudi odlično riješenje koje nam omogućuje da preko indeksa dohvaćamo podatke. Indeks radi na način da pohranjuje vrijednost odredenog polja poredanih prema vrijednosti polja:

```
db.testdb.ensureIndex({"godine" : 1, "ime" : 1})
[ 'godine_1_ime_1' ]
```

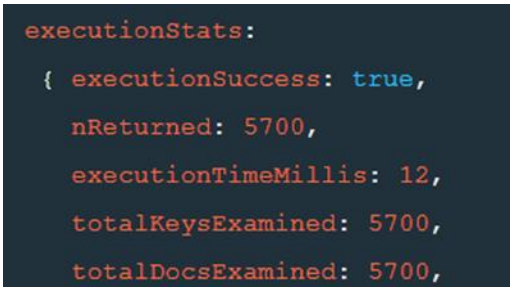
Slika 16: unošenje indexa

Sada kako bismo to testirali, dodijeliti cemo kolekciji indeks baziran na godinama, a zatim na imenu. Indexe dodajemo na atribute za koje želimo izvršiti upite:

```
db.testdb.find({"godine": 25}).explain("executionStats")
```

Slika 17: find i explain nakon indexiranja

Nakon što smo dodijelili index ponovno pokrećemo isti upit:



Slika 18: statistika upita nakon indexiranja

Vidimo da je u ovom slučaju trebalo samo 12 milisekundi da se izvrši i kako je mongo pregledao samo onoliko dokumenata koliko ih postoji sa godinama koje imaju vrijednost 25.

MapReduce primjer

Za početak ćemo unijeti tri objekta koja se sastoje od naziva kolegija te niza studenti koji sadrži objekte sa informacijama o studentima:

```
db.classes.insert({
class : "Baze podataka 1",
students : [
        {ime : "Jure", prezime : "Juric", godine : 22},
        {ime : "Ivan", prezime : "Ivic", godine : 25},
        {ime : "Martin", prezime : "Martinovic", godine : 25},
        {ime : "Ive", prezime : "Ivanovic", godine : 25},
        {ime : "Sandra", prezime : "Sandric", godine : 25},
        {ime : "Monika", prezime : "Monjikic", godine : 25},
        {ime : "Stjepan", prezime : "Stjepanovic", godine : 24}
],
})
```

Slika 19:Objekt kolegija Baze podataka 1

```
db.classes.insert({
class : "Dinamicke web aplikacije",
students : [
        {ime : "Jure", prezime : "Juric", godine : 22},
        {ime : "Ivan", prezime : "Ivic", godine : 25},
        {ime : "Monika", prezime : "Monjikic", godine : 25},
        {ime : "Stjepan", prezime : "Stjepanovic", godine : 24}
],
})
```

Slika 20: Objekt kolegija Dinamičke web aplikacije

```
db.classes.insert({
class : "Baze podataka 2",
students : [
        {ime : "Jure", prezime : "Juric", godine : 22},
        {ime : "Ivan", prezime : "Ivic", godine : 25},
        {ime : "Martin", prezime : "Martinovic", godine : 25},
],
})
```

Slika 21:Objet kolegija Baze podataka 2

Zatim kreiramo Map funkciju koja preko for petlje prolazi kroz sve elemente niza students, te sprema svaki od tih elemenata u varijablu student.

Zatim pozivamo funkciju emit koja prima dva argumenta koja su ključ po kojem želimo grupirati podatke:

```
var mapFunc = function() {
        for(i = 0; i < this.students.length; i++){
                var student = this.students[i];
                emit(student.ime + " " + student.prezime, 1);
        }
};
```

```
var reduceFunc = function(student, broj_predmeta) {
        brojac = 0;

        for(i = 0; i < broj_predmeta.length; i++){
                brojac += broj_predmeta[i];
        }
        return brojac;
};
```

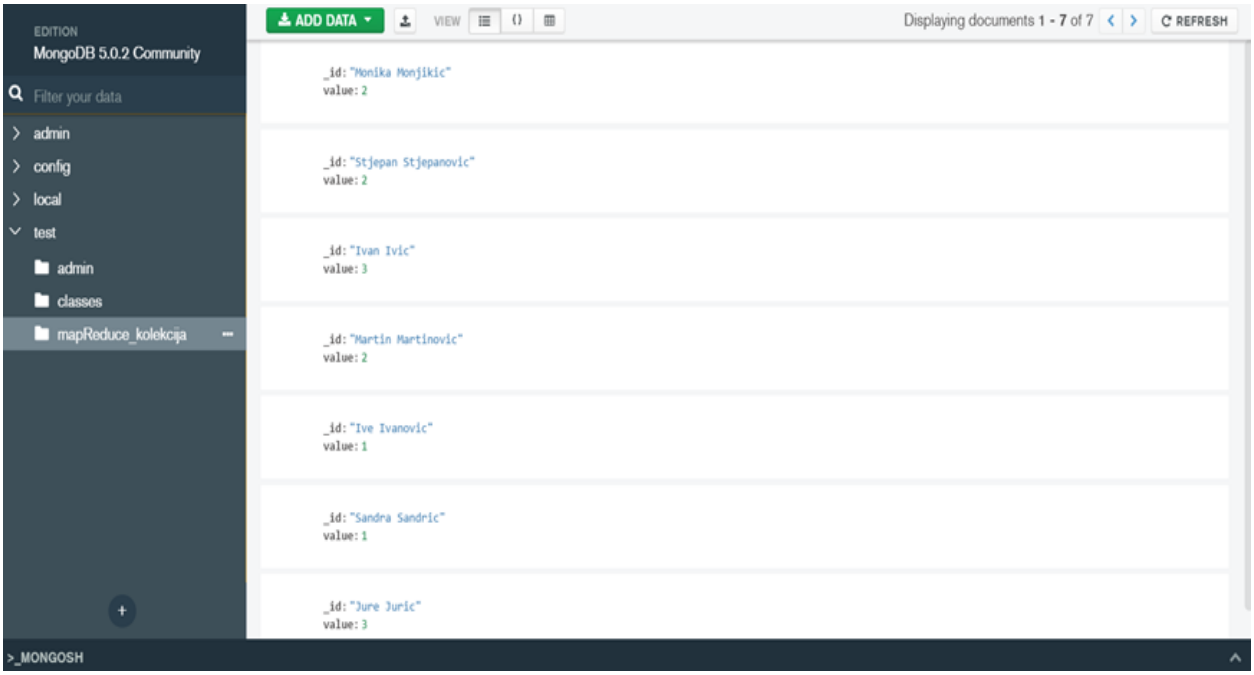
Slika 22: Map funkcija

Kreiramo reduce funkciju koja se poziva po mapu i prima sve vrijednosti po zadanom ključu. Zatim dodajemo koliko se puta ime studenta pojavilo u nizu students:

```
db.classes.mapReduce(
        mapFunc,
        reduceFunc,
        { out: "mapReduce_kolekcija" }
)
```

Slika 24: MapReduce funkcija

Za kraj definiramo mapReduce funkciju prema mongovoj dokumentaciji i dodajemo ime nove kolekcije gdje zelimo ove podatke pohraniti



Slika 25: Prikaz kolekcije MapReduce gdje value označava u koliko je kolegija student upisan

Zaključak

Kako se sve više podataka generira i prikuplja, za analizu podataka potrebni su skalabilni, fleksibilni i visoko učinkoviti alati za pravodoban uvid. Međutim, organizacije se suočavaju sa sve većom količinom podataka. Trenutno postoji na desetke alata za manipuliranje sa toliko velikom količinom podataka. Svaki od alata ima svoje prednosti i mane, a nažalost jos uvijek nemamo alat koji može savršeno baratati sa ogromnom količinom podataka. Razlog tome je što nam je Big data još uvijek „novost“ i podaci se eksponencionalno povečavaju, a alati jako brzo zastarijevaju. Alati su primorani funkcionirat na temelju ne relacijskih baza podataka. Kako nam takve baze pružaju prednosti poput rada sa nestrukturiranim podacima, njihova sigurnost je još uvijek upitna. Alati su u opticaju tek 15-ak godina zbog čega ima prostora za boljim rješenjima. Na svu sreću dolazimo sa sve genijalnijim idejama. Jedan or primjera je MongoDB koji je 2017. godini u jednoj od svojih inačica predstavio „Aggregation Pipline“ koji se čini kao dobra alternativa MapReduce-u.

Literatura

1. Bertoša, G. (2015). Big Data (Doctoral dissertation, University of Rijeka. Faculty of Humanities and Social Sciences).
2. Chickerur, S., Goudar, A., & Kinnerkar, A. (2015, November). Comparison of relational database with document-oriented database (mongodb) for big data applications. In 2015 8th International Conference on Advanced Software Engineering & Its Applications (ASEA) (pp. 41-47). IEEE.Chen, M., Mao, S., & Liu, Y. (2014). *Big Data: A Survey. Mobile Networks and Applications*
3. Chodorow, K. (2013). MongoDB: the definitive guide: powerful and scalable data storage. " O'Reilly Media, Inc.".
4. Gudivada, V. N., Rao, D., & Raghavan, V. V. (2014, June). NoSQL systems for big data management. In 2014 IEEE World congress on services (pp. 190-197). IEEE.
5. Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. International journal of information management, 35(2), 137-144.

Gopalani, S., & Arora, R. (2015). Comparing apache spark and map reduce with performance analysis using k-means. *International journal of computer applications*, *113*(1).

6. Pavlović, M. (2018). MongoDB (Doctoral dissertation, University of Pula. Faculty of Informatics in Pula).
7. https://www.datastax.com/
8. https://inteligencija.com/tehnologije/big-data-tehnologija/
9. https://cloud.google.com/bigtable/docs/samples-map-reduce
10. https://docs.mongodb.com/manual/