Obrada velikih količina podataka u Hadoop ekosistemu

Ljubomir Lazić
Računarski fakultet, Beograd, Srbija
ljlazic@raf.rs

Sažetak—**U ovom radu se opisuje jedno rešenje obrade velikih količina podataka pomoću Hadoop ekosistema u odnosu na tradicionalne baze podataka na primeru fiktivnih transakcija sajta za e-kupovinu. U radu se najpre teorijski obrađuje skladištenje i obrada velikih količina podataka, arhitektura i komponente Hadoop klastera, njegova konfiguracija i primena MapReduce tehnike na njemu. Zatim se, na praktičnom primeru prikazuje nadmoć Hadoop sistema prilikom izvlačenja statističkih podataka iz 3 tabele različitih veličina (1, 10 i 100 miliona transakcija) koje su smeštene u okviru jedne MySQL baze podataka. Na kraju su prikazani rezultati dobijeni merenjima i njihova analiza, kao i zaključci eksperimenta.**

Ključne Big data; Hadoop; MapReduce; ETL proces; MySQL

I. UVOD

Od početka razvoja računarstva, jedan od glavnih problema bilo je povećanje brzine izvršavanja programa. Strogo sekvencijalnom obradom, mnogi resursi su ostajali neiskorišćeni, pa se težilo što efikasnijem iskorišćavanju resursa u svakom trenutku. [1], [2]. Sa razvojem Internet mreže, javila se mogućnost da se paralelizam sa jednog računara prenese na više njih. Tako je nastala distribuirana obrada podataka, gde su se podaci vezani za jedan zadatak izvršavali na više mašina unutar jedne mreže. Međutim, glavni problem velikih kompanija u današnje vreme jeste skladištenje i obrada velikih količina podataka (*Big data*).

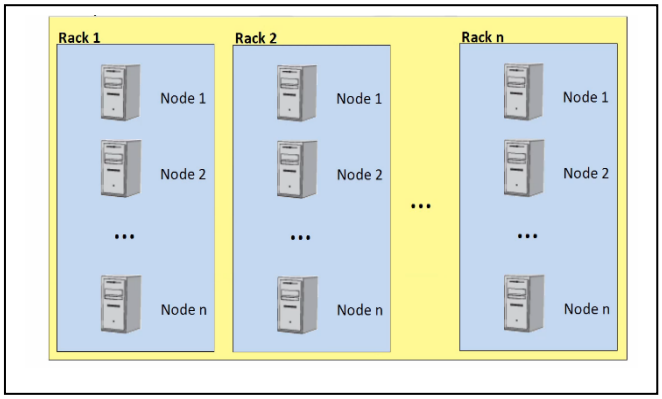
Ljudi svakodnevno upload-uju slike, video materijale, razmenjuju tekstualne poruke, itd. Mašine, sa druge strane, takođe generišu i skladište sve više i više podataka. Ovaj eksponencijalni rast podataka je predstavljao veliki izazov za velike kompanije kao što su Google, Yahoo, Amazon i Microsoft. Oni su morali da istražuju terabajte i petabajte podataka kako bi otkrili koji web sajtovi su popularni, koje knjige su najtraženije, i koje reklame privlače korisnike Interneta. Postojeći alati su postajali neadekvati za procesiranje velikih količina podataka [1].

Cilj ovog rada je da pokaže prednost obrade velikih količina podataka pomoću *Hadoop ekosistema* u odnosu na tradicionalne baze podataka na primeru fiktivnih transakcija sajta za e-kupovinu. U radu će najpre biti teorijski obrađeno sladištenje i obrada velikih količina podataka, biće predstavljena arhitektura i komponente Hadoop klastera, kao i njegova konfiguracija i primena *MapReduce* tehnike na njemu. Zatim će, na praktičnom primeru, biti prikazana nadmoć Hadoop sistema prilikom izvlačenja statističkih podataka iz podataka u tabeli. Na kraju će biti prikazani rezultati dobijeni merenjima i njihova analiza, kao i zaključci koji su usledili iz nje.

II. HADOOP ARHITEKTURA

A. Hadoop klaster

Centralni deo Hadoop arhitekture jeste klaster. Klaster predstavlja veliku kolekciju umreženih računara (čvorova - *Nodes*) od kojih svaki predstavlja *commodity* hardver koji sadrži podatke. Kolekcija od 20-30 čvorova koji su fizički blizu jedan drugoga i povezani na isti mrežni switch se naziva rek (*Rack*). Dakle, Hadoop klaster čini kolekcija rekova (Sl. 1).



Slika 1. Hadoop klaster

Postoje tri vrste čvorova u odnosu na uloge koje imaju:

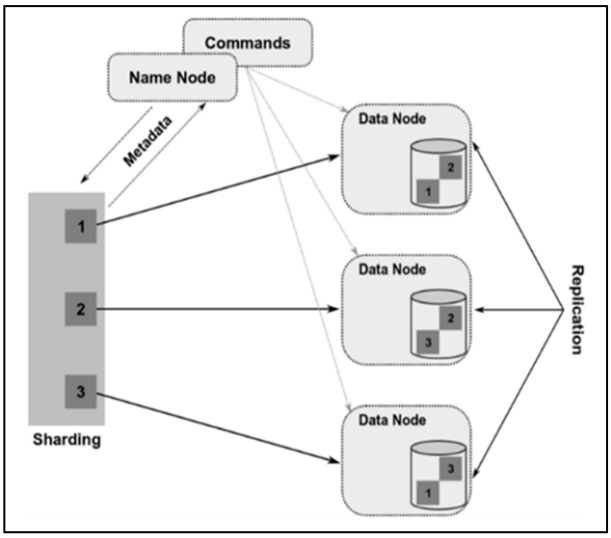
1. *klijent mašina* - *load*-uje podatke u klaster, preda alatu *MapReduce* posao (*job*) koji opisuje kako treba obraditi podatke i dobije ili pogleda rezultate obavljenog posla na kraju.

2. *master* (gospodar) čvor - nadgleda dve ključne komponente Hadoop-a; skladištenje velike količine podataka i izvršavanje paralelnih izračunavanja nad svim tim podacima.

3. *slave* (sluga) čvor - obavlja skladištenje podataka i izvršava izračunavanja.

Hadoop sistem sadrži tri ključne komponente: 1. Distribuiran sistem datoteka (HDFS - Hadoop Distributed File System) tj. fajl sistem koji je zadužen za upravljanje podacima u okviru klastera, 2. MapReduce implementaciju koja omogućava paralelno izvršavanje svih zadataka i 3. YARN menadžer koji je zadužen za koordinisanje i upravljanje resursima u klasteru [1]. HDFS je razvijen po ugledu na Google-ov GFS distribuirani fajl sistem [2].

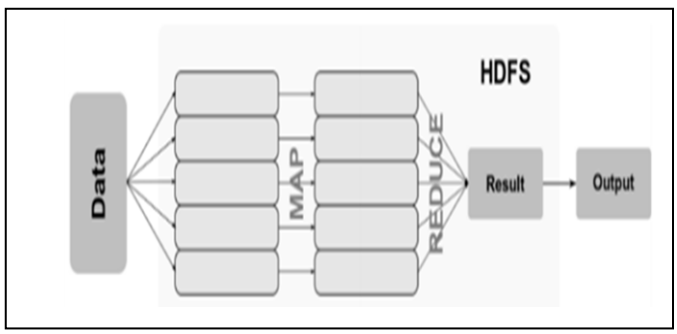
Arhitektura ovog rešenja je osmišljena kao master-slave arhitektura gde je glavni čvor nazvan *NameNode*, a čvor kojim on upravlja je nazvan *DataNode*. Hadoop klaster sadrži samo jedan *NameNode* koji radi u sprezi sa jednim *SecondaryNameNode*-om, kako bi omogućio brz oporavak klastera. U *NameNode*-u se nalaze svi meta podaci koji su vezani za klaster. Broj *DataNode*-ova je praktično neograničen, a može dostići i više desetina hiljada. Na slici 2 je prikazana arhitektura HDFS-a. Za upravljanje podacima koristi sharding mehanizam.



Slika 2. HDFS – Distribuirani Hadoop fajl sistem

B. MapReduce mehanizam

Osnova *Hadoop MapReduce* implementacije počiva u Google-ovom *MapReduce* mehanizmu [2]. Ovaj mehanizam omogućava *Hadoop*-u paralelizaciju izvršenja zadataka i distribuiranje paralelizovanih delova svim čvorovima u klasteru. Izvršenje zadatka počinje deljenjem ulaznih podataka na blokove koji su predmet obrade *Map* dela programa. Rezultati ove faze izvršenja se klasifikuju i predstavljaju ulaz u drugu, *Reduce* fazu. Na slici 3 je prikazan tok podataka kroz *MapReduce* mehanizam. HDFS izvršava prvi korak i reguliše distribuciju, smeštanje i replikaciju podataka. Nakon toga se, na svakom čvoru na kome se podaci nalaze, pokreće *Map* zadatak koji podatke transformiše u niz ključ-vrednost parova koji predstavljaju ulaz u sledeći deo procesa. Pre početka izvršenja *Reduce* zadataka svi ključ-vrednost parovi generisani u prethodnoj fazi se grupišu po vrednosti ključa i tako grupisani predstavljaju ulaz u *Reduce* fazu.



Slika 3. MapReduce tok podataka

MapReduce tehnika se sastoji iz dve vrste transformacija koje mogu biti primenjene više puta na ulazne podatke [3]:

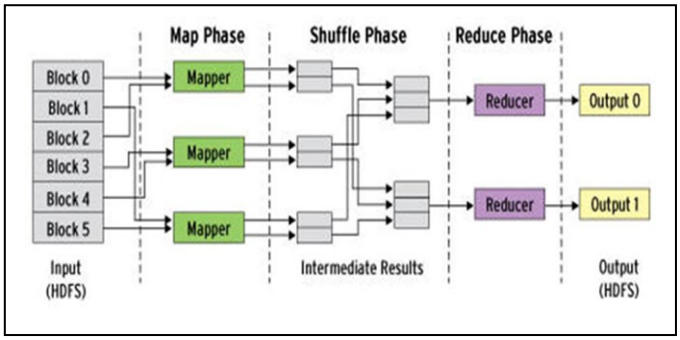
- *Map* transformacija, i
- *Reduce* transformacija.

Kod *Map* komponente *master* čvor deli ulazni fajl na manje fajlove i distribuira ih *slave* čvorovima. Svaki od *slave* čvorova obrađuje manji deo problema i prosleđuje rezultate nazad *master* čvoru. Nakon toga, u *Reduce* komponenti, *master* čvor prikuplja rezultate *slave* čvorova i kombinuje ih kako bi se dobio izlaz, tj. krajnji rezultat.

Tačnije, map funkcija transformiše ulazni skup ključ-vrednost u skup izlaznih vrednosti (skup međupodataka), nakon toga se grupišu međupodaci sa istim ključem i prosleđuju istoj *Reduce* funkciji, dok *Reduce* funkcija vrši sumiranje svih podataka sa istim ključem kako bi se dobio krajnji rezultat. Pseudo-kod map i reduce funkcije dat je na sledećem listingu.

```
Map:      (k1, v1) → list(k2, v2)
Reduce:   (k2, list(v2)) → list(v2)
```

Treba napomenuti da se između *map* i *reduce* funkcije javljaju funkcije *shuffle* i *sort* koje olakšavaju posao *reducer*-u prilikom prikupljanja svih elemenata sa istim ključem. Šematski prikaz *MapReduce* tehnike dat je na slici 4.



Slika 4. MapReduce tehnika

C. Tok obrade

Tipični tok obrade u Hadoop distribuiranom sistemu se sastoji iz 4 dela:

- unos podataka u HDFS (*HDFS* upis),
- obrada i analiza podataka (*MapReduce*),
- skladištenje rezultata u HDFS (*HDFS* upis),
- čitanje podataka iz HDFS (*HDFS* čitanje).

Dakle, jedna *MapReduce* aplikacija počinje sa velikom količinom podataka. Na početku se vrši obilazak podataka, koristi se *map* funkcija da bi se izdvojili podaci od interesa i kreiraju se izlazni podaci iz map funkcije (međupodaci). Zatim se vrši organizacija međupodataka radi optimizacije dalje obrade, nakon čega se koristi *Reduce* funkcija koja izračunava skup rezultata. Na kraju se generiše konačni izlaz.

D. ETL proces i Hadoop alati

ETL je skraćenica za *Extract – Transform - Load* što u prevodu znači *izvlačenje, transformacija i učitavanje*. ETL

proces zapravo predstavlja temeljno skladištenje podataka. Dobro dizajniran ETL proces uzima podatke sa različitih izvora, uvodi kvalitetne podatke, standarde konzistentnosti, prilagođava podatke tako da se odvojeni izvori podataka mogu koristiti zajedno, i na kraju dostavlja iste u format spreman za prezentaciju, a tako da korisnici mogu graditi aplikaciju i donositi odluke.

Potreba za kvalitetnim ETL procesom je velika jer bez kvalitetnih podataka izrađeno skladište podataka neće imati dobre rezultate. ETL nije vidljiv krajnjim korisnicima, ali on predstavlja veliki deo projekta izrade svakog skladišta podataka. Kvalitet podataka u skladištu stvara dodatnu vrednost tih podataka. Osnovne funkcije ETL-a su [4]:

- briše i ispravlja loše podatke,
- pruža dokumentovano merenje poverljivosti podatka,
- priprema tok transakcije za sigurnu pripremu,
- usklađuje podatke sa različitih izvora radi njihovog zajedničkog korišćenja i
- struktuira podatke radi optimalnijeg korišćenja.

Potreban je kvalitetan zaseban proces koji će omogućavati navedene funkcije. ETL se, naravno, može izraditi i unutar samog sistema za upravljanje relacionim bazama podataka (eng. *Relational Database Management System*). Svaki *RDBMS* ima mogućnost korišćenja procedura, okidača, pogleda i funkcija čijom se kombinacijom može izraditi ETL proces. Međutim, takva primena ETL-a ubrzo dostiže svoje granice jer, iako je logika primjenljiva za neki drugi izvor, potrebna je ponovna implementacija.

E. Sqoop alat

Apache Sqoop je alat koji služi za prenos podataka u i iz *Hadoop* klastera. Njegova posebna namena je da radi sa relacionim bazama podataka. Osim transfera podataka u *HDFS*, *Sqoop* može direktno da radi sa *Hive*-om ili *HBase*-om. Prednost *Sqoop*-a je i što brzo prenosi podatke unutar HDFS-a [5].

Sqoop se koristi iz komandne linije, premda postoje i posebni alati sa grafičkim interfejsima u sklopu nekih *Hadoop* distribucija, kao što je alat *Hue* iz *Hadoop* distribucije *Cloudera*, o kojoj će kasnije biti reči. Ime je dobio od kombinacije imena *SQL* i *Hadoop*, i nastao je kao Apache projekat u martu 2012. godine.

Osim unosa kompletne baze podataka, *Sqoop* podržava i unos pojedinačnih tabela, ali i inkrementalna učitavanja, kao i učitavanje delova tabela pomoću *SQL* upita. Takođe, može izvršavati snimljene poslove koji će se izvršavati prilikom svakog ažuriranja baze, i tako periodično osvežavati tabele u *Hadoop* fajl sistemu. Sa druge strane, isto to može obavljati i u službi *SQL* baze podataka, tj. periodično eksportovati podatke iz *Hadoop* fajl sistema u *SQL* tabele.

F. Priprema podataka za obradu u Hadoop-u

Trenutno stanje sistema je da se u bazi podataka sajta za e-kupovinu nalaze 3 tabele sa različitim brojem transakcija. U cilju poboljšanja performansi analize transakcija, podaci će biti importovani u *Hadoop* distribuirani fajl sistem (HDFS) alatom *Sqoop*, gde će biti kreirane tabele i izvršeni identični upiti kao u MySQL bazi podataka alatom *Impala*. Podaci koji se budu našli u HDFS-u moraju biti identični podacima iz MySQL baze podataka, baš kao i rezultati izvršenih upita.

Najpre je potrebno kreirati direktorijume u HDFS-u za smeštanje podataka iz tabela MySQL baze podataka. Trenutna pozicija u HDFS-u je direktorijum za root korisnika, tj. `/user/root/`. Sve komande vezane za *Hadoop* počinju sa `hdfs`, a ukoliko se komanda izvršava nad HDFS-om dodaje se `dfs`. Nakon toga se zadaje konkretna komanda koje su dosta slične sa osnovnim *Linux* komandama. Za kreiranje direktorijuma komanda je `mkdir`. Poslednji parametar komande jeste naziv direktorijuma koji se kreira.

Izvršen je import sve tri tabele, sa 1, 10 i 100 miliona transakcija, iz MySQL baze podataka u HDFS, uz očuvanje integriteta podataka.

Da bi se podaci koji se nalaze u HDFS-u mogli obrađivati SQL upitima, potrebno ih je dovesti u oblik pogodan za to, tj. neophodno je kreirati tabele pomoću alata *Impala* [6]. Pozivanjem komande `impala-shell` iz komandne linije master noda pristupa se komandnoj liniji alata *Impala* bez autentifikacije.

G. Obrada podataka u Hadoop-u

Dakle, u ovom trenutku se nalaze identične tabele u MySQL bazi podataka koja nije povezana sa *Hadoop* ekosistemom, i u HDFS-u u okviru *Hadoop* ekosistema. Tek sada je moguće obraditi podatke u Hadoop-u, izvršavanjem upita nad tabelama u HDFS-u identičnih onim koji su izvršavani nad tabelama u MySQL bazi podataka.

Nakon zadavanja komandi u *Impala* alatu dobijeni su rezultati izvršavanja upita nad tabelom od 1 miliona transakcija koja se nalazi u HDFS-u. Upoređivanjem rezultata sa onima koji su dobijeni izvršavanjem upita nad tabelama u MySQL bazi podataka moguće je utvrditi njihov integritet. Ovo sve je ponovljeno i za ostale 2 tabele sa kojima je eksperimentusano da bi utvrdili njihov integritet.

Jedino što se razlikuje kod rezultata iz *HDFS*-a u odnosu na rezultate iz *MySQL* baze podataka jesu *vremena izvršavanja upita,* i oni će biti detaljno analizirani u narednom poglavlju. Upoređivanje ovih vremena zapravo predstavlja upoređivanje distribuirane obrade koju predstavlja *Hadoop* ekosistem, i obrade nad tradicionalnom bazom podataka kakva je MySQL.

III. UPOREDNA ANALIZA REŠENJA SA MYSQL BAZOM PODATAKA I NA HADOOP KLASTERU

U ovom poglavlju će biti prikazana i upoređena vremena izvršavanja upita nad MySQL bazom podataka i na *Hadoop* klasteru [7]. Takođe, biće izvedeni zaključci iz odnosa distribuirane obrade i obrade sa tradicionalnom bazom podataka.

A. Poređenje vremena izvršavanja upita

U tabeli I prikazana su uporedna vremena izvršavanja identičnih upita nad identičnim tabelama sa 1 milionom transakcija. U prvoj koloni se nalaze vremena izvršavanja upita nad tabelom koja se nalazi u MySQL relacionoj bazi podataka, i predstavlja tradicionalnu obradu podataka na jednoj mašini. U drugoj koloni se nalaze vremena izvršavanja upita nad tabelom koja se nalazi u Hadoop distribuiranom fajl sistemu, i predstavlja distribuirani način obrade podataka na više mašina. Nad ove tri tabele izvršeni su sledeći upiti:

1. **koji korisnik je najaktivniji na sajtu** – koji korisnik je obavio najveći broj transakcija,
2. **koji proizvod je najkupovaniji na sajtu** – nad kojim prozvodom je obavljen najveći broj transakcija,
3. **u kom delu godine su korisnici najaktivniji na sajtu** – u kom mesecu je obavljen najveći broj transakcija,
4. **u kom delu dana su korisnici najaktivniji na sajtu** – u kom času je obavljen najveći broj transakcija.

TABELA I. VREMENA IZVRŠAVANJA UPITA – 1 MILION TRANSAKCIJA

Upit – 1M (55MB)	MySQL	Hadoop
Najaktivniji korisnik	0,41s	0,53s
Najkupovaniji proizvod	0,36s	0,43s
Najaktivniji deo godine	0,20s	0,23s
Najaktivniji deo dana	0,25s	0,33s

Primećujemo da nema značajne razlike u vremenima izvršavanja upita. U tabeli II prikazana su uporedna vremena izvršavanja identičnih upita nad identičnim tabelama sa 10 miliona transakcija.

TABELA II. VREMENA IZVRŠAVANJA UPITA – 10 MILIONA TRANSAKCIJA

Upit – 1M (55MB)	MySQL	Hadoop
Najaktivniji korisnik	1m11s	1,84s
Najkupovaniji proizvod	4,82s	0,87s
Najaktivniji deo godine	2,46s	0,88s
Najaktivniji deo dana	2,55s	0,88s

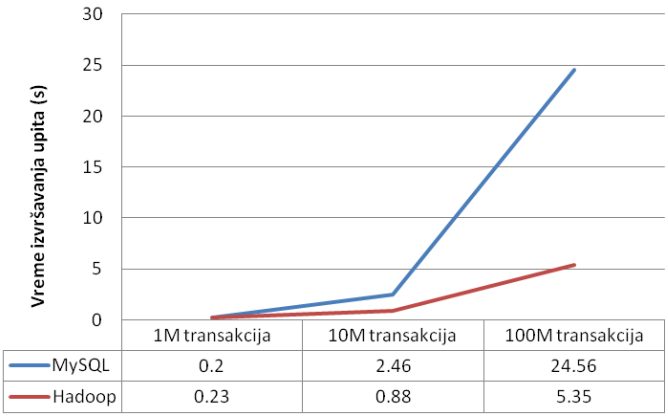
Primećujemo da ima značajne razlike u vremenima izvršavanja upita. Naredna tabela III prikazuje uporedna vremena izvršavanja identičnih upita nad identičnim tabelama sa 100 miliona transakcija.

TABELA III. VREMENA IZVRŠAVANJA UPITA – 100 MILIONA TRANSAKCIJA

Upit – 1M (55MB)	MySQL	Hadoop
Najaktivniji korisnik	45m52s	23,60s
Najkupovaniji proizvod	21,44s	14,07s
Najaktivniji deo godine	24,56s	5,35s
Najaktivniji deo dana	25,68s	5,16s

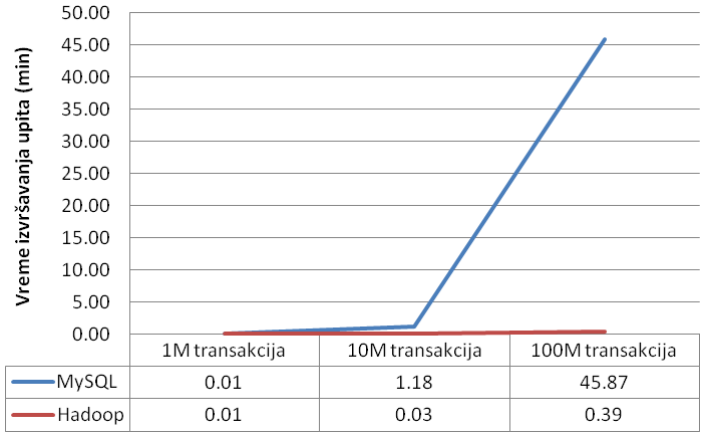
Odnos vremena izvršavanja će se najbolje videti ukoliko se oni prikažu na dijagramu. Dijagrami će biti prikazani za dva tipa upita, jedan koji se izvršava najbrže – **najaktivniji deo godine na sajtu**, i drugi koji se izvršava najsporije – **najaktivniji korisnik**.

Na sledećem dijagramu (Sl. 5) su prikazana vremena izvršavanja upita za najaktivniji deo godine na sajtu u tabelama sa 1, 10 i 100 miliona transakcija, nad MySQL bazom podataka i HDFS-om, izražena u sekundama.



Slika 5. Najaktivniji deo godine na sajtu

Na slici 6 su prikazana vremena izvršavanja upita za najaktivnijeg korisnika u tabelama sa 1, 10 i 100 miliona transakcija, nad MySQL bazom podataka i HDFS-om, izražena u sekundama.



Slika 6. Najaktivniji korisnik

B. Analiza i zaključci izvedeni iz eksperimenta

Prvo što se može primetiti iz tabela jesu različita vremena izvršavanja za različite upite u okviru istog sistema. Dakle, kako u *MySQL* bazi podataka, tako i u *Hadoop*-u, vremena izvršavanja upita variraju, iako se izvršavaju nad istim setom podataka, tj. tabelom sa istim brojem transakcija. To je zbog toga što svi upiti koriste *Group by SQL* funkciju, koja grupiše redove iz tabele po određenoj koloni, pa tako za najsporiji upit – **najaktivnijeg korisnika**, imamo 100 hiljada grupa, za **najkupovaniji proizvod** 10 hiljada, za **najaktivniji deo dana** 24, a za **najbrži upit** – najaktivniji deo godine, imamo 12 grupa. Što se tiče upita, iz dijagrama se može primetiti da, kako raste količina podataka tabele, najbrže napreduju vremena za upit sa najvećim brojem grupa, tj. upit za najaktivnijeg korisnika, dok najsporije napreduju vremena za upit sa najmanjim brojem grupa, tj. upit za najaktivniji deo godine. To je posledica činjenice da se sa porastom količine podataka tabela za najaktivnijeg korisnika, osim broja redova u tabeli,

povećava i broj grupa (za 1 milion transakcija – 100 hiljada korisnika, za 10 miliona transakcija – 1 milion korisnika, i za 100 miliona transakcija – 10 miliona korisnika). Sa druge strane, za najaktivniji deo godine se sa porastom količine podataka tabela i broja redova u tabeli ne povećava broj grupa, već je on uvek isti – 12.

Iz prve tabele se može primetiti da se sva četiri upita *izvršavaju brže nad MySQL bazom* podataka nego nad *Hadoop*-om. Iz druge tabele se primećuje da *Hadoop* preuzima primat nad vremenima izvršavanja, ali da razlika nije toliko velika. U trećoj tabeli se primećuje osetna razlika *Hadoop*-a nad *MySQL* bazom podataka. Dakle, za tabelu koja "teži" 55MB brži je *MySQL*, kod tabele od 550MB brži je *Hadoop*, ali ne dominantno, dok se kod tabele od 5,5GB javlja nadmoć *Hadoop*-a u odnosu na *MySQL* bazu podataka. Ovakvo variranje odnosa je posledica distribuirane prirode *Hadoop* ekosistema.

Naime, *Hadoop* klaster sadrži, u ovom slučaju, 3 mašine koje učestvuju u podeli poslova. *Master nod* najpre podeli zadatke *slave* nodovima, zatim *slave* nodovi izvršavaju upite, i na kraju *slave* nodovi vraćaju rezultate *master* nodu. *Dakle, za razliku od monolitne baze podataka, kakva je MySQL, koja se nalazi na jednoj mašini, kod Hadoop-a je prisutna razmena podataka između nodova, na šta dosta utiče brzina lokalne mreže.*

Čak i kod mreža velikih brzina, sama razmena podataka između nodova uzima određeni vremenski period koji je potreban i za tabele najmanjih dimenzija. Tačnije, kod malih tabela, *Hadoop* ne dolazi do prednosti zbog kratkih vremena izvršavanja samih upita. Sa povećanjem veličine tabela i broja redova u njima, vreme izvršavanja upita postaje duže od vremena potrebnog za razmenu podataka između nodova klastera, pa samim tim i *Hadoop* postaje brži od *MySQL* baze podataka.

Kod tabela ogromnih količina podataka i sa velikim brojem redova, vreme za razmenu podataka između podataka je gotovo zanemarljivo u odnosu na vreme za obradu podataka, te *Hadoop pokazuje svu svoju snagu i nadmoć nad MySQL bazom podataka.*

Kao što i sama namena *Hadoop*-a govori, *to je sistem za obradu velikih količina podataka*, i samo u takvim uslovima on dobija svoju pravu primenu. U konkretnom slučaju tabela iz ovog rada, *Hadoop* dobija svoju primenu tek sa fajlovima koji su veći od **500MB**, a osetnu razliku pravi sa fajlom koji "teži" **5,5GB**. U idealnom slučaju veličine fajlova se moraju izražavati u gigabajtima kako bi *Hadoop ekosistem, koji predstavlja distribuiranu obradu podataka, pokazao apsolutnu nadmoć nad tradicionalnom obradom podataka kakvu pruža MySQL baza podataka.*

IV. ZAKLJUČAK

U radu su predstavljene osnove *Hadoop* ekosistema i njegova arhitektura. Takođe, detaljno su objašnjene komponente *Hadoop* klastera, kako se na njemu primenjuje *MapReduce* tehnika, a ukratko je opisan tipičan tok obrade podataka prilikom izvršavanja jedne *MapReduce* funkcije.

Zatim su opisani koraci *ETL (Extract, Transform, Load)* procesa, kao i *Hadoop* alati koji se u današnjici koriste za izvršavanje pomenutih procesa. Naime, objašnjena je upotreba dva alata: *Sqoop* koji služi za unošenje podataka u *Hadoop* fajl sistem i izvlačenje informacija iz njega, kao i *Impala* koja služi za obradu podataka koji se nalaze u *Hadoop* fajl sistemu. Posle je detaljno opisana implementacija *Hadoop* ekosistema na klasteru od 3 virtuelne mašine korišćenjem distribucije *Cloudera*, koja u sebi sadrži sve neophodne, prethodno pomenute alate. *ETL proces* je predstavljen kroz primer fiktivnih transakcija sajta za e-kupovinu, čime je prikazana nadmoć *Hadoop* sistema prilikom izvlačenja statističkih podataka. Tri tabele različitih veličina (1, 10 i 100 miliona transakcija) su smeštene u okviru jedne *MySQL* baze podataka. Koji je proizvod najkupovaniji, koji je korisnik najaktivniji na sajtu, i u kom delu dana i godine su korisnici najaktivniji na sajtu, pitanja su na koje je odgovor dobijen izvršavanjem upita nad tabelama u bazi.

Realizovan je i testiran transakcioni sistem sajta za e-kupovinu, uz prikaz nadmoći *Hadoop* sistema prilikom izvlačenja statističkih podataka. U okviru budućih istraživanja trebalo bi razmotriti najznačajnije izazove sa kojima se suočavaju provajderi servisa prilikom primene koncepta *Big data*, poput kvantifikovanja vrednosti podataka, konvergencije sa drugim tehnologijama, troškova i tarifiranja.

ZAHVALNICA

Ovaj rad je realizivan u sklopu istraživanja koje delimično finansira Ministarstvo prosvete i Tehnološkog razvoja, Srbija na projektu 'Softversko okruženje za optimalni razvoj kvalitetnog softvera', Projekat br.TR-35026.

LITERATURA

[1] C. Lam, Hadoop in Action, Manning Publications, 2010.
[2] D. Ghemawat, MapReduce: a flexible data processing tool, Communications of the ACM 53.1, 2010.
[3] T. White, Hadoop: The Definitive Guide, O'Relly Media, 2012.
[4] R. Kimball, The Data Warehouse ETL Toolkit, Wiley Publishing, 2004.
[5] K. Ting, Apache Sqoop Cookbook, O'Relly Media, 2013.
[6] J. Russell, Getting Started with Impala, O'Reilly Media, 2016.
[7] N. Spasić, Obrada velikih količina podataka u Hadoop ekosistemu, FIT, METROPOLITAN Univerzitet, master rad, 2018.

ABSTRACT

This paper presents the basics of the Hadoop ecosystem and its architecture. Also, the components of the Hadoop cluster are explained in detail, how the MapReduce technique is applied, and a typical flow of data processing when performing a MapReduce function is briefly described. After a comparative analysis of the solutions with the MySQL database and on the Hadoop cluster over the traditional solutions, the advantage of the proposed solution in this paper over the traditional solutions was unambiguously shown.

PROCESSING LARGE AMOUNTS OF DATA IN THE HADOOP ECOSYSTEM

Ljubomir Lazić, The School of Computing, Belgrade, ljlazic@raf.rs