Upute za laboratorijske vježbe: Hadoop[1]

Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu
Računarstvo zasnovano na uslugama

26. listopada 2018.

Sadržaj

1 Uvod **2**

2 Radni okvir Hadoop **3**

2.1 Osnovne značajke i namjena 3

2.2 Kratka povijest razvoja 6

3 Programski model MapReduce **7**

3.1 Oblikovanje MapReduce programa 7

3.2 Podsustav za izvođenje MapReduce programa 12

4 Zadatak za vježbu **15**

Literatura **16**

Poglavlje 1

Uvod

Količina podataka koju prikupljaju danas najpopularnije i najviše korištene web-usluge zahtijeva potpuno drukčije programske sustave i modele programiranja od onih na koje su programeri naviknuli. Nije neuobičajeno da volumen dnevnika velikih mrežnih sjedišta raste brzinom od `1TB` na dan. Prirodno je za očekivati da će ovaj trend još samo rasti s padom cijena sklopovlja i pojavom boljih i bržih načina stvaranja sadržaja.

Tradicionalne programske sustave nije moguće prilagoditi obradi nevjerojatnih količina podataka odjednom dostupnih programerima i analitičarima. Radni okvir *Hadoop* razvijen je za *ad-hoc* paralelnu obradu nestrukturiranih podataka koristeći programski model *MapReduce* i raspodijeljeni datotečni sustav *HDFS*. Ovaj dokument služi za upoznavanje s osnovnim načelima programiranja, postavljanja i izvršavanja *MapReduce* programa. Drugo poglavlje opisuje radni okvir Hadoop, njegove osnovne značajke i komponente. Treće poglavlje detaljno opisuje *MapReduce* programski model.

Poglavlje 2

Radni okvir Hadoop

U ovom poglavlju opisan je Hadoop radni okvir. Dana je kratka povijest nastanka radnog okvira, opisana je njegova struktura te najvažnije komponente.

2.1 Osnovne značajke i namjena

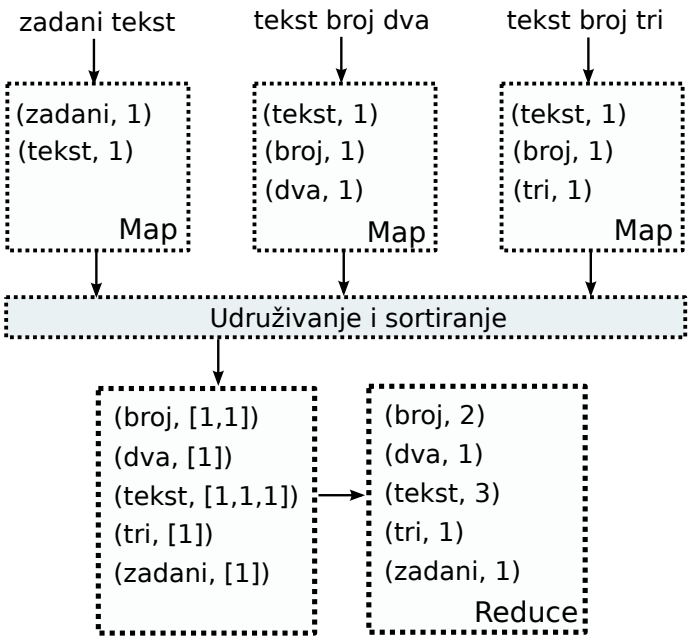
Radni okvir *Hadoop* je programski sustav za analiziranje velikih količina podataka koji se nalaze u pouzdanom i raspodijeljenom spremištu [2]. "Velike količine" podataka odnose se na baze podataka programskih sustava koji učinkovito upravljaju volumenom podataka na Internet razini (engl. *Internet scale*), npr. sustavi za web indeksiranje ili dubinsku analizu podataka (engl. *data mining*). Radni okvir Hadoop ostvaren je u programskom jeziku *Java*.

Jezgru radnog okvira čine programski model *MapReduce* i raspodijeljeni datotečni sustav *HDFS*. Koristeći osnovne Hadoop primitive, razvijeno je nekoliko programskih sustava koji olakšavaju korištenje Hadoopa ili ostvaruju potpuno nove funkcionalnosti. Sustav *Pig* olakšava pisanje MapReduce programa uvodeći poseban jezik *Pig Latin* te okolinu za izvršavanje programa. Pig prevodi programe iz jezika više razine u MapReduce programe koje izvodi na spletu računala. *HBase* je raspodijeljena baza podataka razvijena po uzoru na *BigTable* bazu podataka koja se koristi u tvrtci Google. HBase je namijenjen za čitanje i pisanje zapisa u stvarnom vremenu za vrlo velike skupove podataka. *ZooKeeper* je pouzdani sustav za koordinaciju raspodijeljenih primjenskih sustava.

Programski model MapReduce

MapReduce je programski model za raspodijeljenu obradu podataka sa svojstvom linearnog razmjernog rasta. MapReduce se može promatrati kao sustav za izvršavanje upita u pozadini (engl. *batch query processor*). Bez obzira na količinu podataka, sustav obrađuje cijeli skup podataka za svaki upit. Obrada se definira dvjema funkcijama:

- *Map* – transparentno čitanje "sirovih" podataka iz raspodijeljenog datotečnog sustava, filtriranje te generiranje parova *ključ-vrijednost*
- *Reduce* – obrada udruženih i sortiranih parova generiranih *Map* funkcijama te generiranje izlaza u obliku parova *ključ-vrijednost*.



Slika 2.1: Kanonski primjer MapReduce programa

Kanonski primjer MapReduce programa je program za brojanje ponavljanja riječi u zadanom tekstu (slika 2.1). Zadani tekst podijeljen je u tri particije. Funkcija *Map* paralelno čita odgovarajuće particije i za svaku riječ unutar teksta generira par $(rijec, 1)$ koji označava da je pročitana promatrana riječ. Nakon što su pročitane sve particije, sustav MapReduce udružuje

i sortira generirane parove u obliku para $(rijec, [1, ..., 1])$. Ulaz u funkciju *Reduce* su udruženi i sortirani parovi iz prethodnog koraka. Broj ponavljanja pojedine riječi predstavlja duljina liste u drugom elementu para.

Za razliku od sustava za upravljanje relacijskim bazama podataka (RDBMS), MapReduce model pokazuje svojstvo razmjernog rasta jer funkcije *Map* i *Reduce* ne ovise o veličini ulaznog skupa podataka niti o veličini spleta računala na kojem se sustav izvršava. MapReduce model obrađuje cijeli skup podataka za vrijeme izvršavanja upita, dok RDBMS sustavi obično održavaju dodatne strukture podataka (npr. B-stabla) koje ubrzavaju izvršavanje upita ili ažuriranje manje količine zapisa, ali značajno usporavaju ažuriranje većine zapisa u bazi. Dodatno, MapReduce programski model zamišljen je za obradu nestrukturiranih (ili polustrukturiranih) podataka kao što je tekst ili binarni podatci.

Raspodijeljeni datotečni sustav HDFS

HDFS (engl. *Hadoop Distributed Filesystem*) je raspodijeljeni datotečni sustav za spremanje i slijedno čitanje vrlo velikih datoteka u spletu računala [2]. HDFS je oblikovan za brzo slijedno (engl. *streaming*) čitanje, jer je za programski model MapReduce važnija optimizacija čitanja cijelog skupa podataka od optimizacije vremena pristupa prvom zapisu u skupu podataka. Dodatno, HDFS se izvršava na spletu običnih računala (engl. *commodity hardware*) gdje je vjerojatnost kvara vrlo velika. HDFS osigurava ispravan rad sustava bez obzira na prisutnost kvarova ili grešaka u komunikaciji.

HDFS sprema datoteke u raspodijeljeno spremište koristeći apstrakciju blokova. Pretpostavljena veličina bloka u HDFS datotečnom sustavu je 64MB. Veličina bloka je velika (u odnosu na veličinu bloka običnih datotečnih sustava) zbog optimizacije vremena čitanja – nakon što je blok pozicioniran, cijeli blok se može učitati deklariranom brzinom čitanja. Prednosti korištenja blokova fiksne veličine u raspodijeljenom datotečnom sustavu su jednostavnost ostvarenja te mogućnost spremanja datoteka čija veličina nadmašuje kapacitet bilo kojeg diska u spletu računala.

Dostupnost i otpornost sustava na pogreške osigurava se replikacijom blokova u spletu računala. Ako promatrani blok postane nedostupan, HDFS održava replike promatranog bloka i njihovo čitanje i održavanje je transparentno korisnicima datotečnog sustava. Pretpostavljeni broj replika bloka je 3. Interno Hadoop osigurava konzistentnost datoteka koristeći dvije vrste čvorova u spletu: *čvor imenik* za održavanje prostora imena te *podatkovni*

čvor za spremanje blokova. HDFS klijenti komuniciraju s imenikom koristeći standardno POSIX sučelje.

Budući da Hadoop koristi datotečni sustav kroz apstraktno sučelje, osim datotečnog sustava HDFS, Hadoop podržava i druge datotečne sustave kao što su lokalni datotečni sustav ili S3 – datotečni sustav za uslugu *Amazon S3*.

2.2 Kratka povijest razvoja

Hadoop je nastao 2004. godine kao otvoreno ostvarenje *MapReduce* radnog okvira razvijenog u tvrtci Google [1]. Prvu inačicu razvio je *Doug Cutting* za potrebe ostvarenja web-pretraživača *Nutch*. MapReduce i HDFS se 2006. godine odvajaju od projekta Nutch u projekt Hadoop kao dio projekta *Apache Lucene*. U isto vrijeme autori radnog okvira Hadoop zapošljavaju se u tvrtci *Yahoo!* koja stvara posebnu grupu razvojnih inženjera posvećenu isključivo razvoju Hadoopa. 2008. godine Hadoop postaje projekt prve razine u sklopu Apache grupe.

Danas je Hadoop prihvaćen u industriji kao *de facto* standard pozadinske obrade podataka sa svojstvom razmjernog rasta. Između ostalih, koriste ga Yahoo!, Facebook, Twitter, Last.fm i drugi [3].

Poglavlje 3

Programski model MapReduce

Ovo poglavlje detaljno opisuje MapReduce programski model kao temeljni koncept obrade podataka u Hadoop radnom okviru. Osim samog načina oblikovanja i osvarenja MapReduce programa, opisana je i arhitektura sustava za izvođenje MapReduce programa.

3.1 Oblikovanje MapReduce programa

Map funkcija MapReduce programskog modela može se prikazati kao:

`map:` `(K1, V1)` $\rightarrow$ `list(K2, V2)`,

tj. *Map* funkcija za ulaz prima par ključ-vrijednost `(K1, V1)`, a kao izlaz daje listu parova `(K2, V2)`. Udruženi i sortirani parovi `(K2, list(V2))` ulaz su u funkciju *Reduce*:

`reduce:` `(K2, list(V2))` $\rightarrow$ `list(K3, V3)`.

Funkcija *Reduce* ponovo kao izlaz daje listu parova koji mogu predstavljati ulaz u drugi MapReduce program, čime se jednostavno ostvaruje cjevovod MapReduce programa. Kompleksna obrada podataka koristeći MapReduce programski model obično zahtijeva oblikovanje većeg broja manjih MapReduce poslova, za razliku od oblikovanja jednog kompleksnog MapReduce posla.

Primjer MapReduce programa

Kao primjer MapReduce programa poslužit će program za računanje broja pogleda video isječaka na zamišljenoj web-usluzi za prikazivanje video isječaka. Web-usluga bilježi u tekstualni dnevnik (engl. *log*) svako prikazivanje pojedinog video isječka koristeći jednostavan zapis: `VideoID Postotak`, gdje je `VideoID` jedinstveni identifikator video isječka, a `Postotak` označava koliko je korisnik vremenski stvarno pogledao video isječak (npr. ako je video isječak pogledan samo do polovice ukupnog trajanja, onda je vrijednost `Postotak` polja jednaka 50). Primjer dnevnika prikazuje sljedeći isječak tekstualne datoteke:

```
00001 100
00001 95
00002 100
00001 2
00002 100
00003 100
00001 100
```

Zamišljena web-usluga broji samo one poglede u kojima je korisnik pogledao više od 5% video isječka. Iz jednostavnog dnevnika lako je iščitati da je video isječak `0001` pogledan ukupno 4 puta, ali je jedan od tih pogleda ispod granice gledanosti po trajanju pa web-usluga broji 3 pogleda za promatrani video isječak.

MapReduce program koji računa najgledaniji video isječak sastoji se od *Map* funkcije koja čita parove `(VideoID Postotak)` te jednostavno proslijedi pročitani par ukoliko vrijednost `Postotak` polja nije manja od 5:

```
1  import java.io.IOException;
2  import org.apache.hadoop.io.IntWritable;
3  import org.apache.hadoop.io.LongWritable;
4  import org.apache.hadoop.io.Text;
5  import org.apache.hadoop.mapred.MapReduceBase;
6  import org.apache.hadoop.mapred.Mapper;
7  import org.apache.hadoop.mapred.OutputCollector;
8  import org.apache.hadoop.mapred.Reporter;
9
10 public class VideoCountMap extends MapReduceBase
```

```
11   implements Mapper<LongWritable, Text, Text, IntWritable> {
12
13   private static final int THRESHOLD = 5;
14
15   public void map(LongWritable key, Text value,
16                   OutputCollector<Text, IntWritable> output,
                        Reporter reporter)
17     throws IOException {
18
19     String[] parts = (value.toString()).split("␣");
20     String id = parts[0];
21     int percent = Integer.parseInt(parts[1]);
22
23     if (percent > THRESHOLD) {
24       output.collect(new Text(id), new IntWritable(percent));
25     }
26   }
27 }
```

Linije 19-21 čitaju identifikator video isječka i odgovarajući postotak gledanosti iz dnevničkog zapisa. Ako je postotak veći od definiranog praga (`THRESHOLD`), *Map* funkcija generira par ključ-vrijednost sadržaja `(id, percent)` (linija 24). Hadoop radni okvir prikuplja generirane parove, udružuje ih te sortira. Ulaz u funkciju *Reduce* je:

```
(00001, [100, 95, 100])
(00002, [100, 100])
(00003, [100])
```

Zadatak *Reduce* funkcije je jednostavno brojanje članova liste spremljene u drugom elementu pročitanih parova:

```
1 import java.io.IOException;
2 import java.util.Iterator;
3 import org.apache.hadoop.io.IntWritable;
4 import org.apache.hadoop.io.Text;
5 import org.apache.hadoop.mapred.MapReduceBase;
6 import org.apache.hadoop.mapred.OutputCollector;
7 import org.apache.hadoop.mapred.Reducer;
8 import org.apache.hadoop.mapred.Reporter;
```

```
9
10 public class VideoCountReduce extends MapReduceBase
11   implements Reducer<Text, IntWritable, Text, IntWritable> {
12   public void reduce(Text key, Iterator<IntWritable> values,
13                      OutputCollector<Text, IntWritable> output,
                           Reporter reporter)
14     throws IOException {
15
16     int count = 0;
17     while (values.hasNext()) {
18       values.next();
19       count += 1;
20     }
21     output.collect(key, new IntWritable(count));
22   }
23 }
```

Varijabla `count` služi kao brojač elemenata spremljenih u drugom elementu para ključ-vrijednost (`values`). Broj elemenata liste se emitira ponovo kao drugi element para ključ-vrijednost (linija 21), dok je ključ pročitani identifikator video isječka.

Program koji postavlja i pokreće MapReduce prikazan je sljedećim isječkom:

```
1  import java.io.IOException;
2  import org.apache.hadoop.fs.Path;
3  import org.apache.hadoop.io.IntWritable;
4  import org.apache.hadoop.io.Text;
5  import org.apache.hadoop.mapred.FileInputFormat;
6  import org.apache.hadoop.mapred.FileOutputFormat;
7  import org.apache.hadoop.mapred.JobClient;
8  import org.apache.hadoop.mapred.JobConf;
9
10 public class VideoCount {
11
12   public static void main(String[] args) throws IOException {
13     if (args.length != 2) {
14       System.err.println("Usage:␣VideoCount␣<input␣path>␣<
             output␣path>");
15       System.exit(-1);
16     }
```

```
17      JobConf conf = new JobConf(VideoCount.class);
18      conf.setJobName("Video count");
19      FileInputFormat.addInputPath(conf, new Path(args[0]));
20      FileOutputFormat.setOutputPath(conf, new Path(args[1]));
21      conf.setMapperClass(VideoCountMap.class);
22      conf.setReducerClass(VideoCountReduce.class);
23      conf.setOutputKeyClass(Text.class);
24      conf.setOutputValueClass(IntWritable.class);
25      JobClient.runJob(conf);
26    }
27  }
```

Prije izvođenja programa potrebno je program prevesti i zapakirati u `jar` arhivu:

```
1$ javac -classpath
hadoop-common-2.7.4.jar:hadoop-mapreduce-client-core-2.7.4.jar
-d classes *.java
2$ jar -cvf VideoCount.jar -C classes/ .
```

Samo pokretanje Hadoop sustava, pokretanje programa te izlistavanje rezultata ostvaruje se sljedećim naredbama[1]:

```
1$ start-dfs.sh
2$ hdfs dfs -put log log1
3$ hadoop jar VideoCount.jar VideoCount log1 out1
4$ hdfs dfs -cat out1/part-00000
00001 3
00002 2
00003 1
5$ hdfs dfs -get out1/part-00000
```

Prve naredba pokreće HDFS i MapReduce podsustave, dok druga naredba kopira primjer dnevnika (`log`) iz lokalnog datotečnog sustava u raspodijeljeni datotečni sustav (`log1`). Treća naredba pokreće program `VideoCount` sa ulazom `log1` i direktorijem za rezultate `out1`. Po završetku izvođenja programa, rezultati se nalaze u datoteci `part-00000` direktorija `out1`.

[1] pretpostavlja se da je radni okvir Hadoop ispravno postavljen na računalu

3.2 Podsustav za izvođenje MapReduce programa

Podsustav za izvođenje MapReduce programa u radnom okviru Hadoop čini glavni čvor naziva `JobTracker` te skup čvorova radnika naziva `TaskTacker`. MapReduce program poslan na izvođenje naziva se *posao* (engl. *job*). Hadoop dijeli posao u skup *zadataka* (engl. *task*). Ulaz u MapReduce program je skup podataka spremljenih na raspodijeljeni datotečni sustav. Hadoop dijeli ulazne podatke u particije jednake veličine koje se zatim dodjeljuju *Map* funkcijama. *Map* funkcije emitiraju parove *ključ-vrijednost* koje sustav Hadoop udružuje i sortira prema ključu. Kad sve *Map* funkcije završe s zadatkom, *Reduce* funkcije obavljaju zadatke na sortiranim i udruženim parovima.

Slanje poslova na izvođenje

MapReduce posao šalje se na izvođenje koristeći Hadoop klijentsku aplikaciju `JobClient`. Klijentska aplikacija traži od glavnog čvora (`JobTracker`) novi jedinstveni identifikator posla i izračunava particije ulaznih podataka. Nakon što su ulazni podaci podijeljeni u particije i nakon što su provjereni parametri posla (npr. postojanje izlaznog direktorija), `JobClient` kopira komponente posla u raspodijeljeni datotečni sustav u direktorij naziva jednakog identifikatoru posla generiranom u prvom koraku. Komponente posla uključuju `JAR` arhivu sa samim programom, konfiguracijske datoteke te particije ulaznih podataka.

Nakon što su komponente posla postali dostupni u raspodijeljenom datotečnom sustavu, posao se sprema u interni red poslova (engl. *job queue*). Raspoređivač poslova (engl. *job scheduler*) zatim dohvaća posao te inicijalizira zadatke potrebne za njegovo izvođenje. Inicijalizirani zadatci uključuju *Map* funkcije (po jednu *Map* funkciju za svaku particiju ulaznih podataka) i *Reduce* funkcije (broj *Reduce* funkcija definiran je u konfiguracijskoj datoteci).

Izvođenje zadataka

Izvođenje zadataka je u potpunosti orkestrirano glavnim čvorom. Prije samog izvođenja pojedinih zadataka, `JobTracker` mora izabrati kojem poslu

pripadaju zadatci koje će izvoditi. Pretpostavljeni raspoređivač bira posao koji je prvi stigao u red poslova. Nakon što odabere posao, `JobTracker` dodijeljuje zadatke koji čine odabrani posao slobodnim radnicima.

Radnik (`TaskTracker`) periodički javlja svoje stanje glavnom čvoru. Stanje uključuje informaciju o slobodnim "utičnicama" (engl. *slots*) za *Map* i *Reduce* zadatke. Važna optimizacija događa se kod dodijeljivanja *Map* zadataka. *Map* zadatci pokušavaju se dodijeliti čvorovima radnicima na kojima se nalaze podatci koje obrađuje upravo dodijeljeni zadatak. Na taj način se izbjegava skupa mrežna komunikacija jer su podatci lokalni *Map* zadatku.

S ciljem optimizacije preklapanja čitanja i obrade podataka, Hadoop radni okvir pokreće više *Map* i *Reduce* zadataka konkurentno na čvorovima radnicima. Pretpostavljeni broj utičnica za zadatke je 4 (dva za *Map* i dva za *Reduce* zadatke).

Nakon što je čvoru radniku dodijeljen zadatak, radnik dohvaća `JAR` arhivu s programom, pokreće posebnu instancu *Java* prividnog stroja (JVM) za izvođenje dodijeljenog zadatka. *MapReduce* programi mogu trajati satima, stoga čvorovi radnici periodički dojavljuju napredak izvršavanja zadatka. Posao je završio kada čvor radnik koji izvršava posljedni zadatak u poslu javi glavnom čvoru da je završio sa izvršavanjem dodijeljenog zadatka.

Oporavak od pogrešaka

Izvođenje zadataka je podložno programskim ili sklopovskim pogreškama koje se mogu manifestirati na razne načine. Jedna od prednosti radnog okvira Hadoop je praćenje stanja poslova i zadataka, rukovanje pogreškama i zastojima u radu te omogućavanje izvršavanja posla u potpunosti u nesigurnoj okolini računalnog spleta.

Pogrešku u čvoru radniku može uzrokovati iznimka za vrijeme izvođenja *Map/Reduce* zadatka ili neka druga pogreška u JVM sustavu. Čest slučaj je i pojava sporih radnika ili radnika u zastoju. Kada glavni čvor primi periodičku poruku o stanju radnika s dojavom pogreške, on ponovo pokreće neuspješan zadatak (izbjegavajući pokretanje zadatka na istom čvoru radniku gdje se dogodila pogreška). Ako glavni čvor uopće na primi periodičku poruku o stanju radnika, glavni čvor briše radnika u kvaru iz skupa radnika za raspoređivanje poslova (*tasktrackers pool*).

Dodatno, dostupnost sredstava u sustavu je vrlo visoka koristeći raspodijeljeni datotečni sustav s replikacijom blokova. npr. stupanj replikacije sred-

stava poslova je 10. Na taj način osim očitih prednosti smanjenja mrežne komunikacije tijekom izvođenja zadataka, povećava se i pouzdanost sustava redundancijom podataka.

Hadoop trenutno ne razmatra pogrešku u glavnom čvoru koji predstavlja jedinstvenu točku pada sustava (engl. *single point of failure*). Jedna od mogućnosti zaštite je uvođenje redundancije korištenjem sustava *ZooKeeper* koji bi upravljao spletom čvorova i određivao primarni (glavni) čvor.

Poglavlje 4

Zadatak za vježbu

Vaš zadatak je:

1. proučiti radni okvir Hadoop i postaviti ga na vlastitom računalu (za potrebe ove vježbe nije potrebno postavljati splet računala, dovoljno je koristiti jedno računalo u *pseudocluster* načinu rada) [1]

2. pokrenuti *MapReduce* primjer programa iz uputa (VideoCount) (izlaz je lista s elementima (*video id, broj-pogleda*))

3. ostvariti *MapReduce* program koji generira silazno sortiranu listu gledanosti video sredstava (izlaz je sortirana lista s elementima (*video id, broj-posjeta*))

4. Korištenjem *MapReduce* programskog okvira ostvariti množenje dviju matrica. Matrice su zapisane u jednoj datoteci, svaka vrijednost matrice predstavljena je uređenom četvorkom (M, r, c, v) gdje je M ime matrice, r je redak matrice, c je stupac matrice a $M[r, c] = v$. Rezultat množenja treba zapisati u izlaznu datoteku kao listu uređenih trojki (r, c, v), gdje je r redak rezultirajuće matrice, c je stupac rezultirajuće matrice a v je pripadajuća vrijednost za redak r i stupac c.

   Npr. neka je zadan sljedeći ulaz:

```
a 0 0 2
a 0 1 -1
```

[1] Više o to me možete pročitati na poveznici `https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html`

```
a 1 0 1
a 1 1 3
b 0 0 1
b 0 1 2
b 1 0 3
b 1 1 4
```

Za zadani ulaz program treba ispisati sljedeći izlaz u izlaznu datoteku:

```
0 0 -1
0 1 0
1 0 10
1 1 14
```

Što je potrebno znati na labosima?

Očekuje se razumijevanje MapReduce paradigme i Hadoop radnog okvira (ovaj dokument je dovoljna literatura). Na samoj predaji demonstrirati ispravnost svih zadataka koje je potrebno ostvariti u okviru vježbe.

Literatura

[1] Jeffrey Dean and Sanjay Ghemawat. Mapreduce simplified data processing on large clusters, 2004.

[2] Tom White. *Hadoop: The Definitive Guide, Second Edition*. O'Reilly, 2011.

[3] Wikipedia. Apache hadoop framework, 2014.