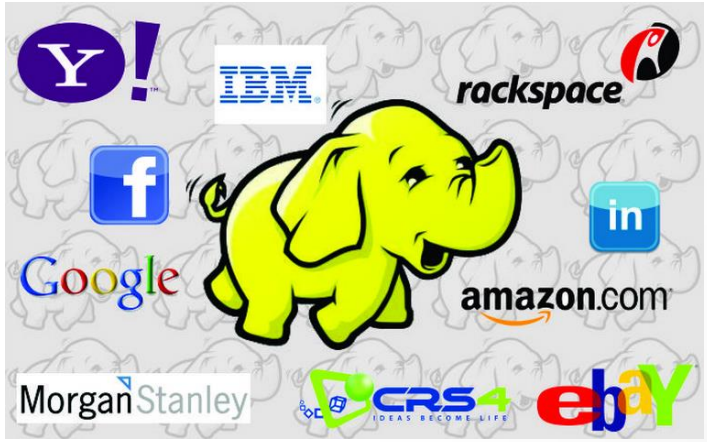
Apache hadoop

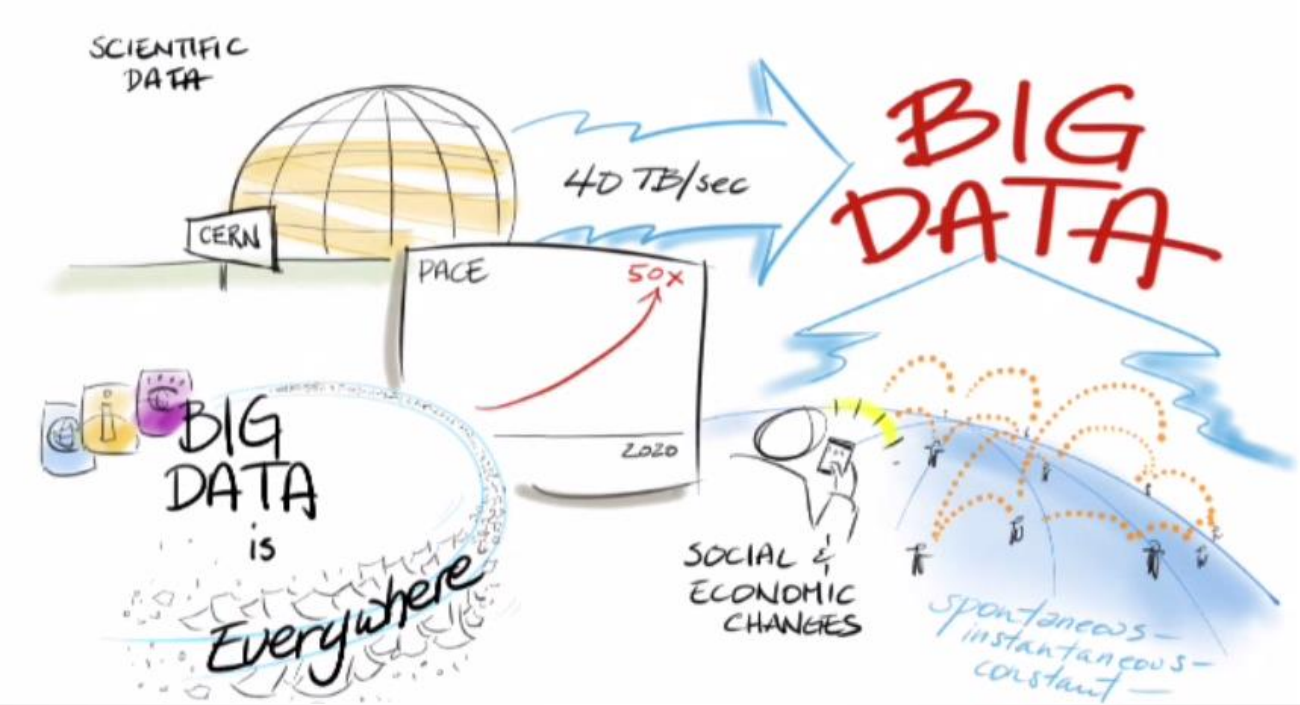
Skladištenje i obrada podataka

- Tradicionalni način : MySql, Oracle, MSSQL, ...
- Eksponencijalan rast količine podataka u svetu
- Big data
- Uticaj velikih količina podataka na vodeće svetske IT kompanije

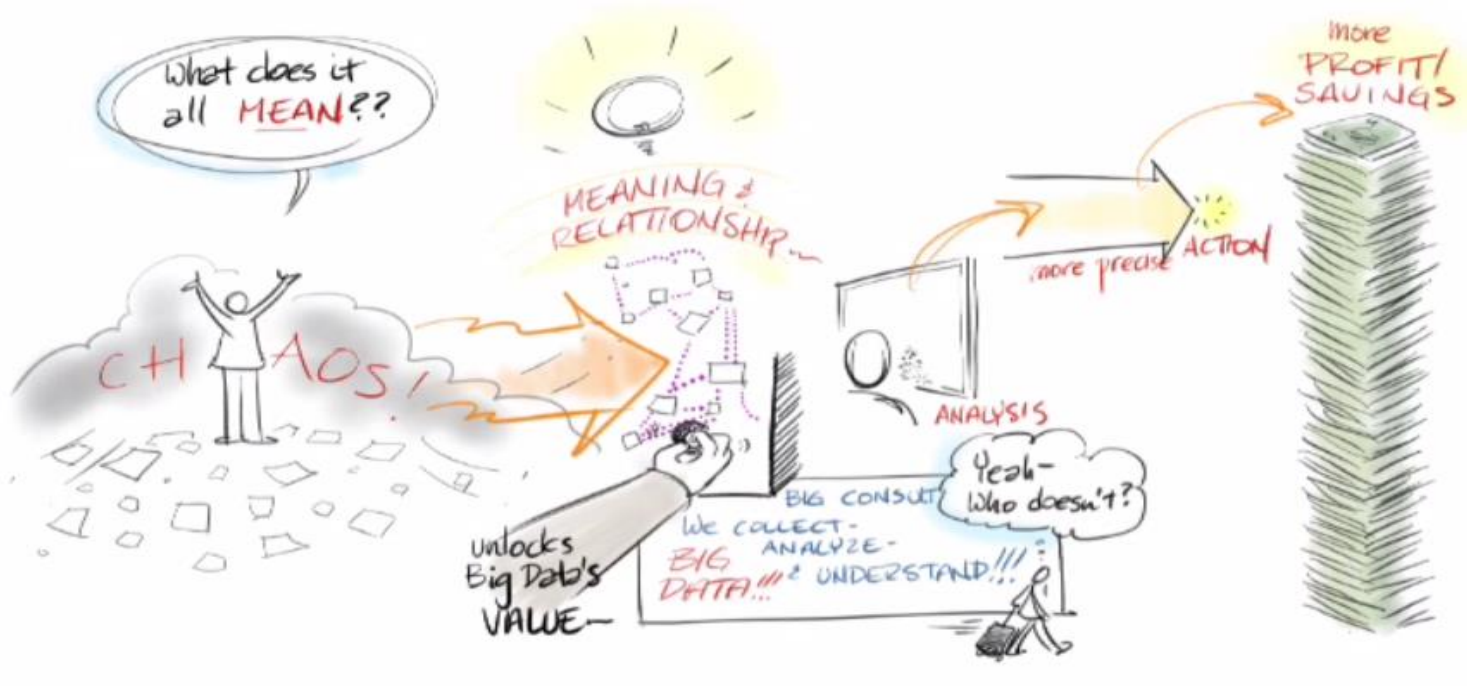


How big is Big data?

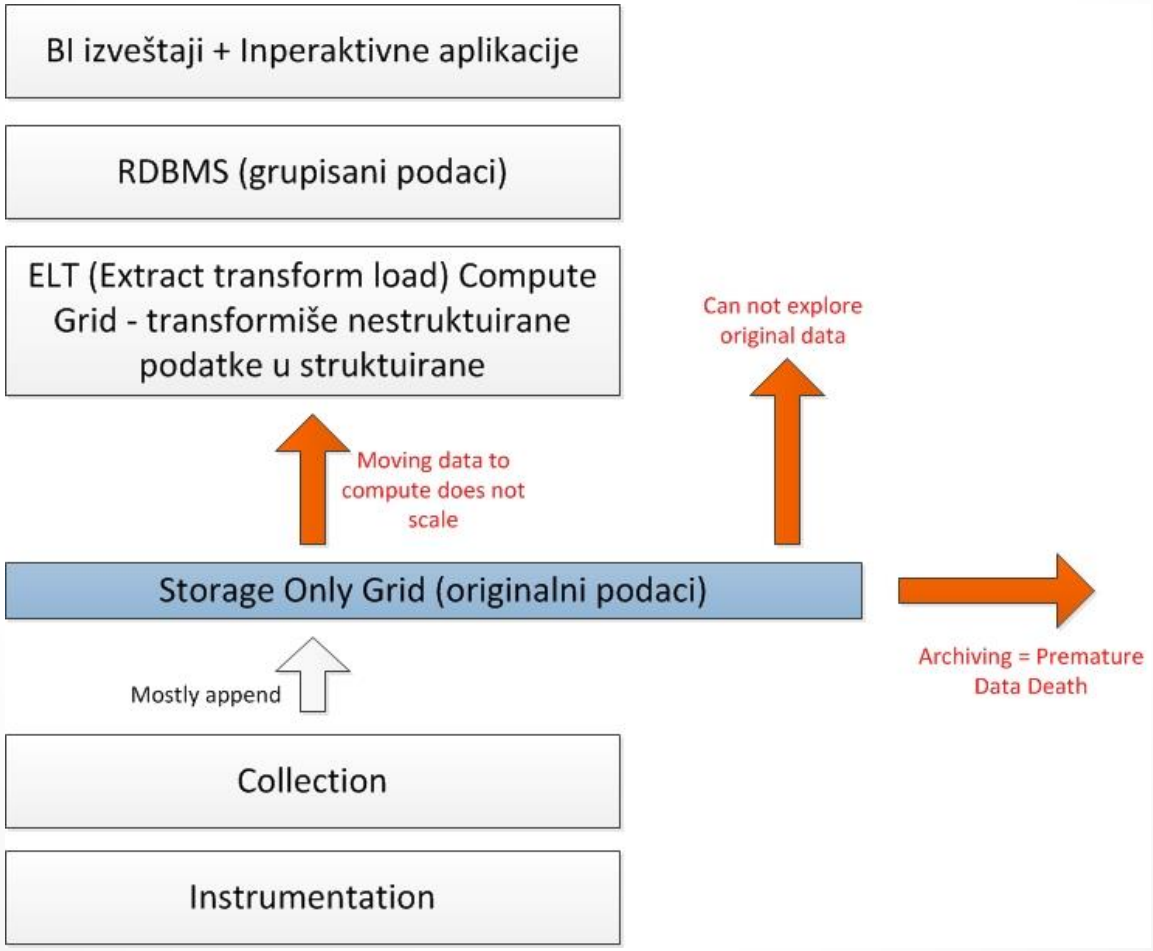
- **U svetu trenutno postoji 4*10^{21} bajtova podataka**
- **2020. godine 40% informacija će biti obrađivano na *cloud* platformi**



Značaj analize velikih podataka



Ograničenja tradisionalnog načina skladištenja podataka



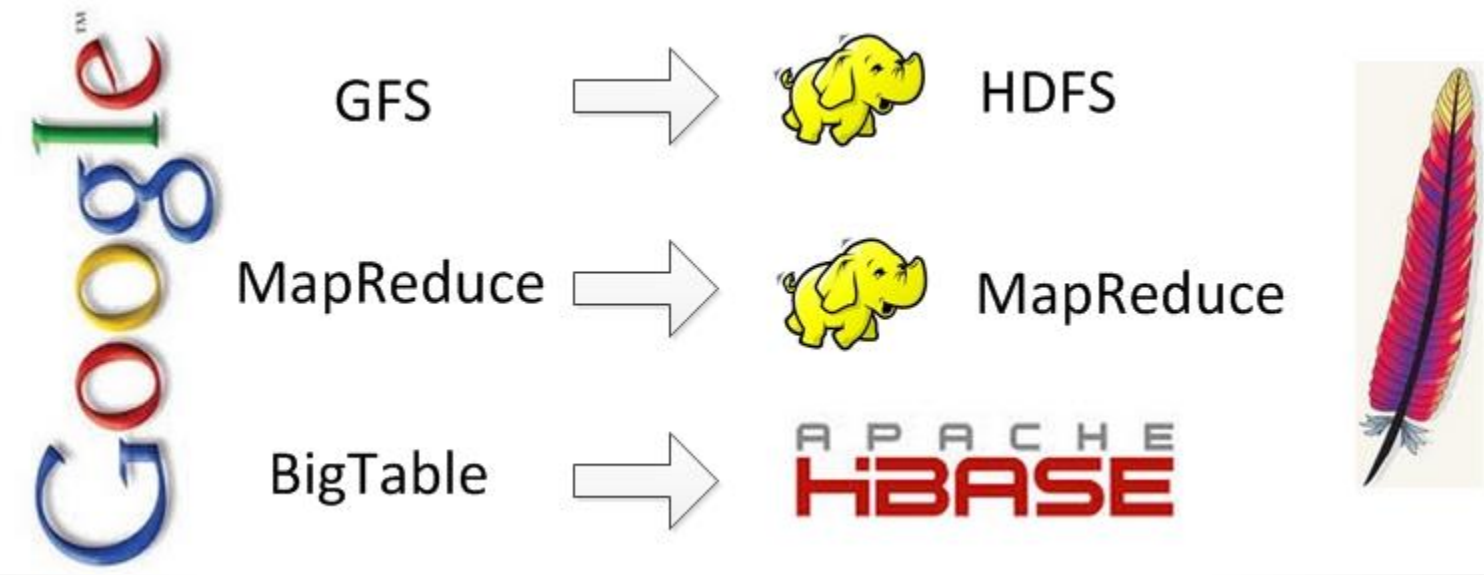
Frejmvork za obradu i skladištenje velikih količina podataka

- Potreban je distribuirani sistem koji ima sledeće osobine:
  - Ugrađeno čuvanje kopija podataka
  - Lak oporavak od pada sistema i kvarova komponenti
  - Jednostavan način pisanja koda
  - Jednostavno ispravljanje grešaka
  - Jednostavna administracija sistema

Apache hadoop

- Frejmvork otvorenog koda, pisan u JAVA programskom jeziku za pouzdane, skalabilne i distribuirane sisteme.
- Omogućava distribuiranu obradu velikih skupova podataka kroz klastere računara koristeći jednostavan programski model.
- Glavne osobine:
  - Skalabilnost,
  - Robusnost,
  - Jednostavnost.
- Dve ključne celine:
  - HDFS – distribuirani fajl sistem,
  - MapReduce – programski model za parelelnu obradu podataka.

Kako je sve počelo?



Schema on Write vs. Schema on Read

Schema on Write (SQL RDBMS)

Kreiranje šeme tabele

```
CREATE TABLE Customers (
 id int, name varchar(40), ...
)
```

Nemoguće je ubaciti podatke pre nego što se šema kreira.

Dodavanje podataka

```
BULK INSERT Customers
FROM 'c:\temp\customers.txt'
WITH FIELDTERMINATOR = '","'
```

Promena strukture fajla customers.txt zahteva promenu šeme tabele. Ukoliko želimo da čuvamo nove podatke u tabeli moramo dodati kolonu.

Upit nad podacima

```
SELECT * FROM Customers
```

Brzo čitanje podataka

Schema on Write vs. Schema on Read

Schema on Read (Hadoop)
-Dodavanje podataka hadoop dfs -copyFromLocal /temp/customers*.txt /user/hadoop/custs
Upit nad podacima hadoop jar wordcount.jar WordCount hdfs://input hdfs://output

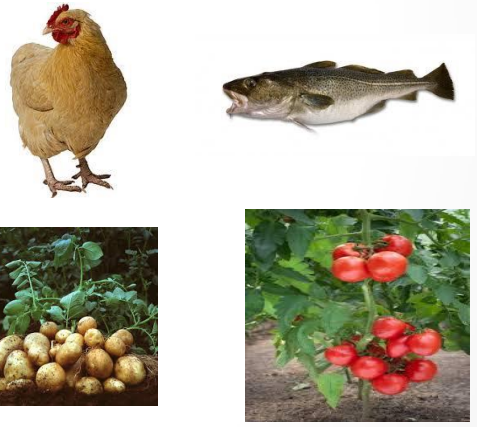
Potrebno je jednostavno kopirati fajlove na HDFS (Hadoop Distributed File System) bez prethodnog definisanja šeme.

Programskim modelom određujemo kako će šema podataka izgledati.

Istraživanje originalnih podataka

RDBMS

Hadoop



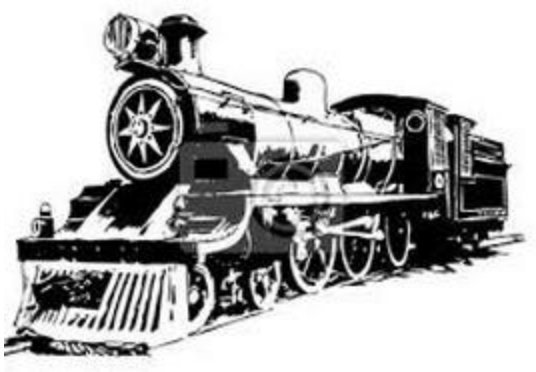
Ko ima više mogućnosti da napravi dobar obrok?

Pravi alat za prave stvari

Relational Databases:

Prednosti:
Interaktivna OLAP analiza (< 1 sec)
Multistep ACID transaction
100% SQL Compilance

Hadoop:



Prednosti:
Koristi se i kada su podaci struktuirani i kada nisu
Skalabilbost skladištenja podataka
Kompleksno procesiranje podataka

Apache hadoop projekti

- **Hadoop Common** - zajednički alati koje podržavaju drugi projekti.
- **Hadoop Distributed File System (HDFS)**: Distribuirani sistem datoteka koji obezbeđuje visok pristup podacima.
- **Hadoop MapReduce**: Biblioteka za distribuirane obrade velikih setova podataka na klasterima.
- **AvroTM**: Sistem za serilizaciju podataka.
- **CassandraTM**: Skalabilna multi-master baza podataka bez pojedinačnih tačaka neuspeha.
- **ChukwaTM**: Sistem za prikupljanje podataka za upravljanje velikim distribuiranim sistemima.
- **HBaseTM**: Skalabilna, distribuirana baza podataka za podšku kreiranja struktuiranog načina skladistenja velikih tabela podataka.
- **HiveTM**: Infrastruktura magacina za skladištenje podataka koja omogućava generalizovanje i ad hok pravljenje upita nad podacima.
- **MahoutTM**: Skalabilna mašna za učenje i „data mining“ biblioteka
- **PigTM**: Visoki data-ow jezik i biblioteka za paralelno programiranje.
- **ZooKeeperTM**: Harmonični servis sa visokim performansama za distribuirane aplikacije.

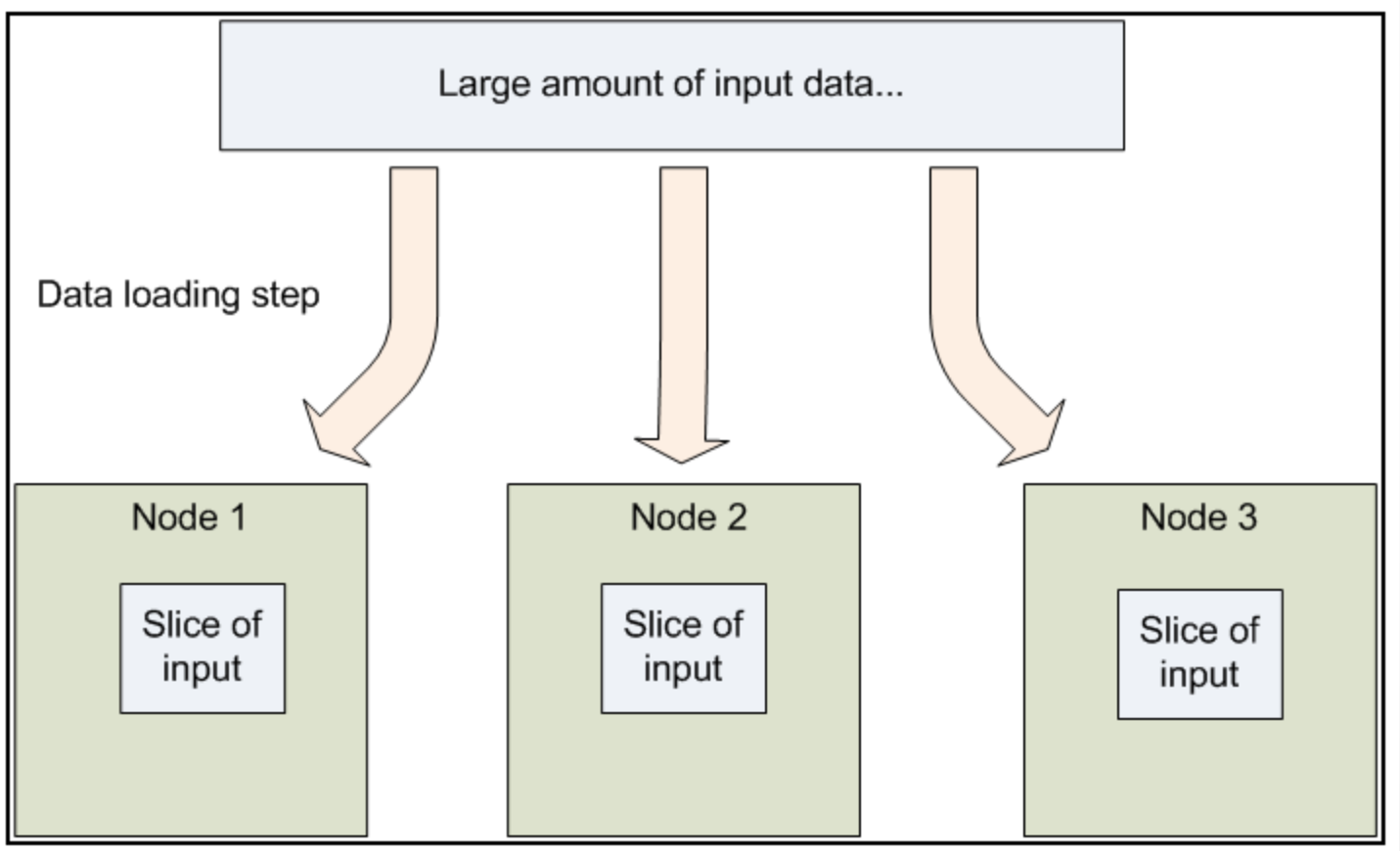
Hadoop Distributed File System

- Distribuirani fajl sistem napravljen da radi na **hardveru** koji je **lako dostupan.**
- Veoma je **otporan** na **greške** i dizajniran da bude raspoređen na **hardveru** koji **radi paralelno.**
- Obezbeđuje **visok nivo pristupa podacima** i **pogodan** je za **aplikacije** koje upravljaju **velikim skupom podataka.**
- HDFS instanca se može **sastojati** iz **stotina** ili **hiljada računara**.
- **Kvar** je izuzetak ili **pravilo**?
- **Aplikacije** na HDFS-u **nisu opšte namene** jer HDFS **nije** dizajniran za **interaktivni pristup korisnika.**

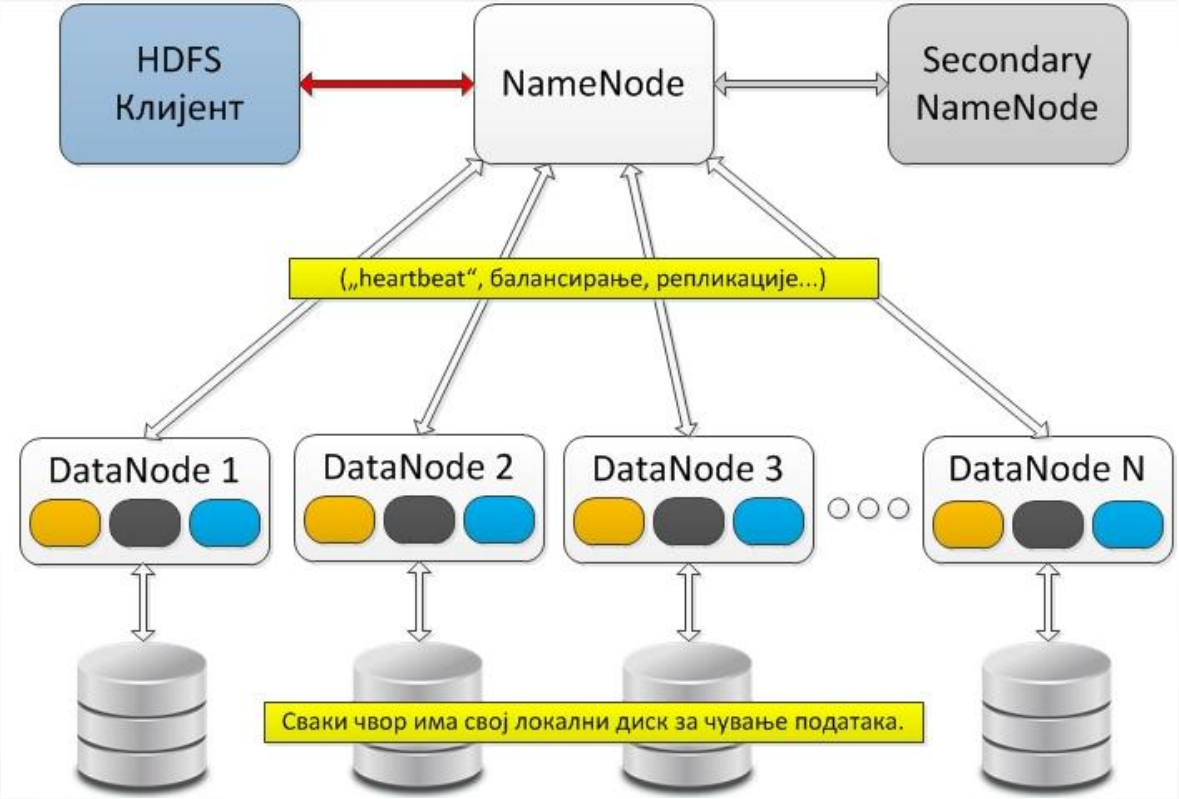
Opis modala podataka na HDFS-u

- **Write-once-read-many** – ova pretpostavka pojednostavljuje povezanost podataka što omogućuje visok propusni opseg protoka podataka.
- Postoji plan da se **podrži izmena fajlova**.
- HDFS omogućava **kretanje** same **aplikacije** do mesta gde su podavi koje ona obrađuje smešteni. Prednosti ovakvog pristupa?

Distribucija podataka



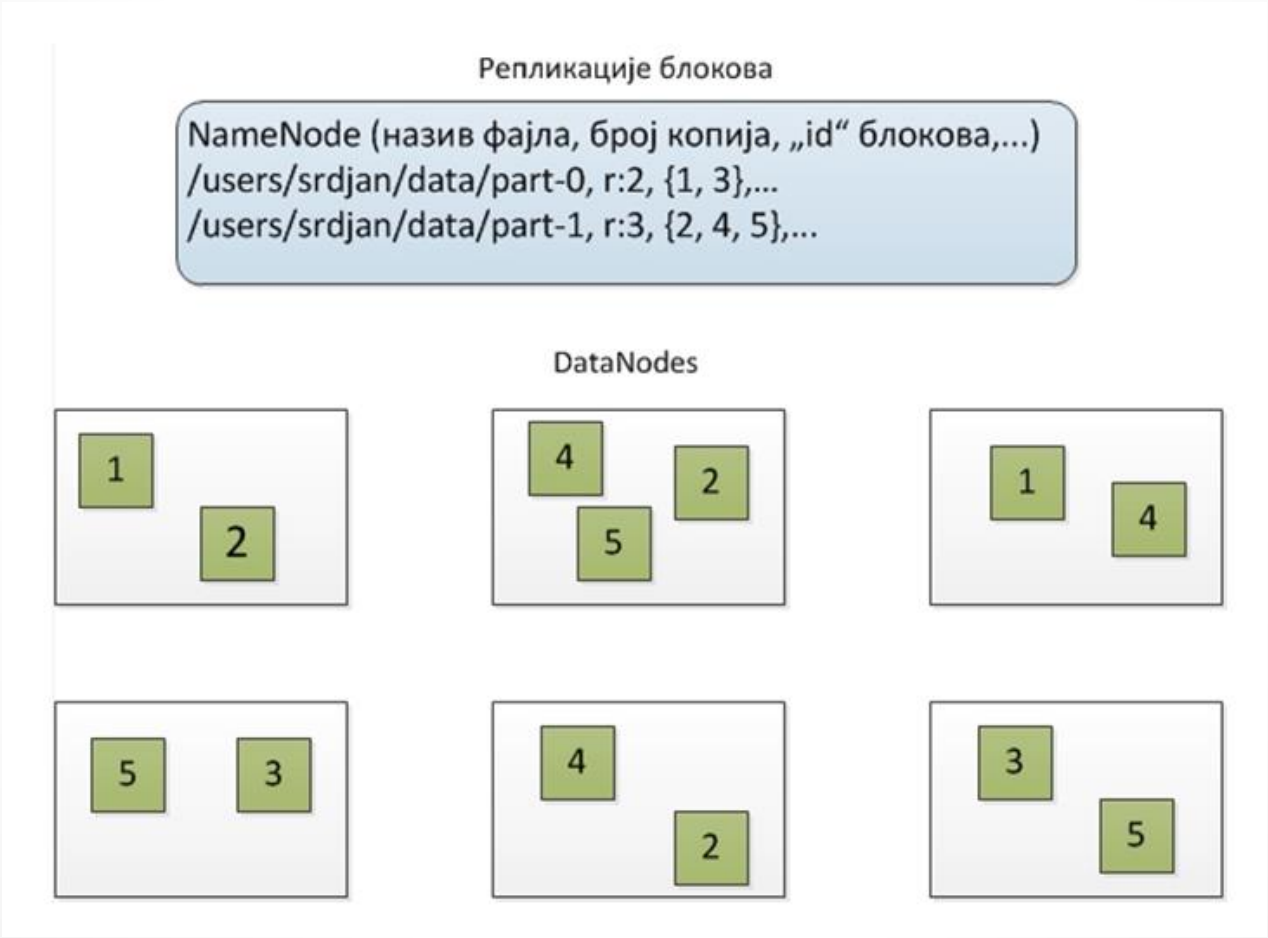
Arhitektura HDFS-a



Arhitektura HDFS-a

- HDFS ima master/slave arhitekturu.
- **NameNode** – master server
  - Upravlja fajl sistemom,
  - Reguliše pristup,
  - Izvršava komande kao što su otvaranje, zatvaranje i preimenovanje direktorijuma i fajova,
  - Fajlovi su interno podeljeni na jedan ili više blokova gde NameNode vrši mapiranje blokova za DateNode čvorove.
- **Niz DataNode-a** – čvorovi za podatke
  - Obično jedan po čvoru klastera,
  - Imaju ulogu da upravljaju skladištenjem podataka na čvoru na kome su pokrenuti,
  - Odgovorni su za opsluživanje zahteva za čitanje i pisanje od strane korisnika,
  - Kreiranje i brisanje blokova po nalogu iz NameNode-a.

Raplike blokova



DFSShell

- HDFS omogućava DFSShell interfejs da upravlja podacima koji se nalaze na HDFS-u.
- Većina komandi koje komuniciraju sa Hadoop-om izvršavaju se uz pomoć skripta **hadoop** (učitava Hadoop sistem sa java virtuelnom mašinom i izvršava komandu definisanu od strane korisnika)
  - **hadoop modulName -cmd args…**
    - **modulName –** program koji predstavlja podskup funkcionalnosti Hadoop-a
    - **-cmd** – definiše specifičnu komandu koju modul treba da izvrši

DFSShell primeri 1

cat

hadoop dfs -cat URI [URI . . .]

Kopira sadržaje navedenih fajlova na standardni izlaz.

Primer:

hadoop dfs -cat hdfs://host1:port1/file1 hdfs://host2:port2/file2

hadoop dfs -cat file:///file3 /user/hadoop/file4

chmod

hadoop dfs -chgrp [-R] GROUP URI [URI . . .]

Promena ovlašćenja nad fajlovima. Korisik mora biti vlasnik fajlova ili super-user

DFSShell primeri 2

chown

hadoop dfs -chown [-R] [OWNER][:[GROUP]] URI [URI]

Menja vlasnika fajlova. Može je pokrenuti samo super-user.

copyFromLocal

Usage: hadoop dfs -copyFromLocal <localsrc> URI

Kopira fajl(fajlove) sa lokalnog fajl sistema na HDFS.

mkdir

hadoop dfs -mkdir <paths>

Ponaša se nalik UNIX komande *mkdir -p*.

rm

hadoop dfs -rm URI [URI . . .]

Briše fajlove navedene koji su dati kao argumenti komande.

DFSShell primeri 3

put

hadoop dfs -put <localsrc> ... <dst>

Kopira fajl(fajlove) sa lokalnog fajl sistema na HDFS. Takođe čita ulaz sa stdin i upisuje ga na HDFS fajl sistem.

- hadoop dfs -put localfile /user/hadoop/hadoopfile
- hadoop dfs -put localfile1 localfile2 /user/hadoop/hadoopdir
- hadoop dfs -put localfile hdfs://host:port/hadoop/hadoopfile
- hadoop dfs -put - hdfs://host:port/hadoop/hadoopfile - **Čita podataka sa standardnog ulaza.**

HDFS Java 1

- **Napomena**: potrebno je uključiti odgovarajuće klase iz biblioteka **org.apache.hadoop.conf**, **org.apache.hadoop.fs** i **org.apache.hadoop.io**

Kopiranje fajla sa lokalnog fajla na HDFS fajl sistem:

```
Configuration config = new Configuration();
FileSystem hdfs = FileSystem.get(config);
Path srcPath = new Path(srcFile);
Path dstPath = new Path(dstFile);
hdfs.copyFromLocalFile(srcPath, dstPath);
```

Kreiranje fajla na HDFS fajl sistemu:

```
//byte[] buff - Sadržaj fajla
Configuration config = new Configuration();
FileSystem hdfs = FileSystem.get(config);
Path path = new Path(fileName);
FSDataOutputStream outputStream = hdfs.create(path);
outputStream.write(buff, 0, buff.length);
```

HDFS Java 2

Preimenovanje fajla na HDFS fajl sistemu:

```
Configuration config = new Configuration();
FileSystem hdfs = FileSystem.get(config);
Path fromPath = new Path(fromFileName);
Path toPath = new Path(toFileName);
boolean isRenamed = hdfs.rename(fromPath, toPath);
```

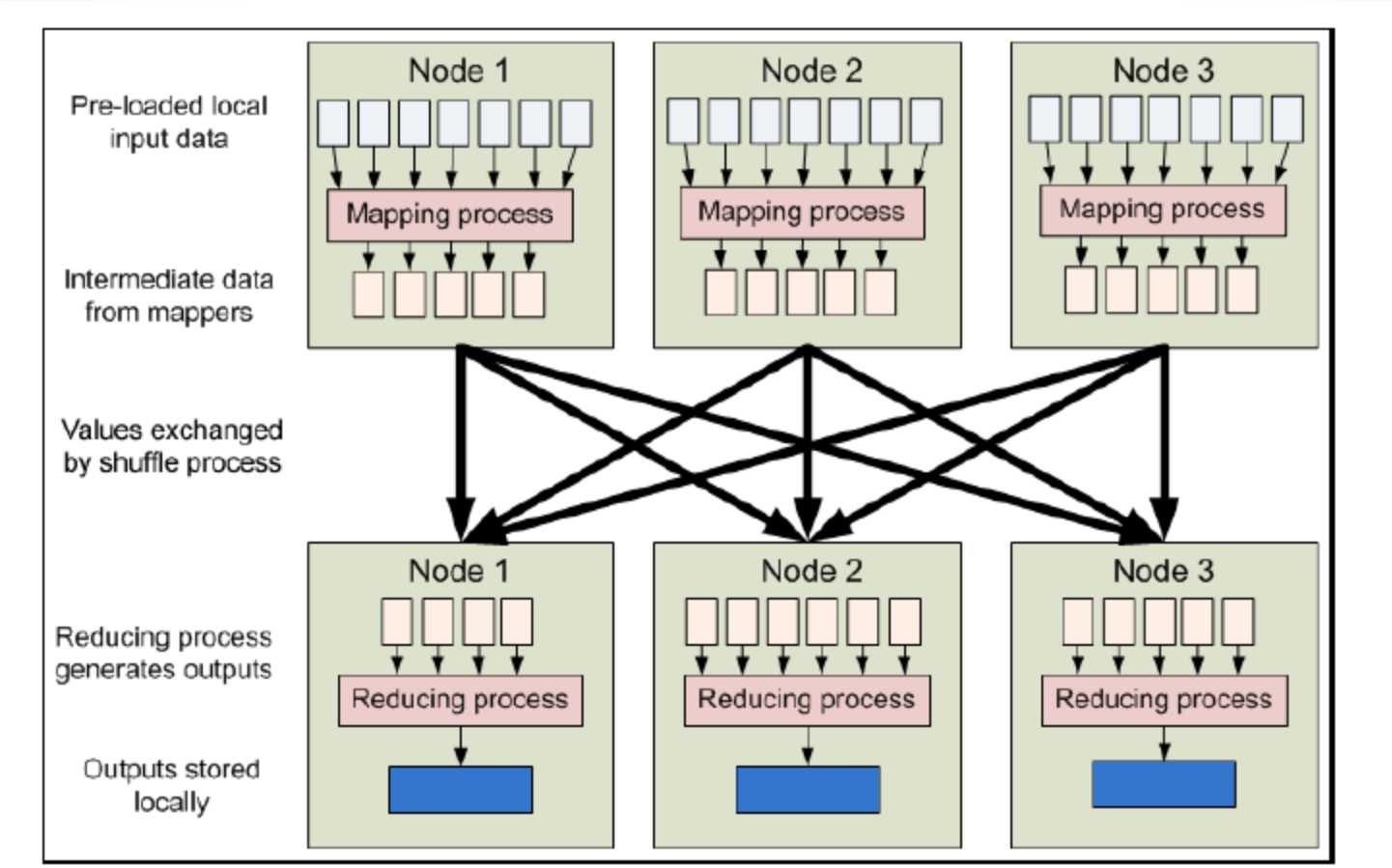
Brisanje fajla sa HDFS fajl sistema:

```
Configuration config = new Configuration();
FileSystem hdfs = FileSystem.get(config);
Path path = new Path(fileName);
boolean isDeleted = hdfs.delete(path, false);
```

Map/Reduce model

- Java, Ruby, Python, i C++.
- Model je baziran na dve posebna koraka:
  - **Mapiranje**: Početni korak obrade i transformacije ulaznih podataka, u kome se zapisi ulaznih podataka obrađuju paralelno,
  - **Redukcija**: Ovaj korak se radi posle mapiranja i predstavlja agregaciju, gde se svi povezani zapisi obrađuju kao jedan entitet(detaljnije objašnjenje u nastavku).
- Map/Reduce framework radi isključivo sa **(key, value)** parovima
- Glavni koncept Map/Reduce modela u Hadoop-u je da se ulaz može podeliti u više logičkih komada, i da se svaki od tih komada može obrađivati paralelno, od strane zadataka mapiranja. Rezultati ovih pojedinačnih obrada mogu biti podeljeni na različite skupove podataka koji se sortiraju. Svaki sortirani komad se šalje redukcionom zadatku.

Map/Reduce tok podataka



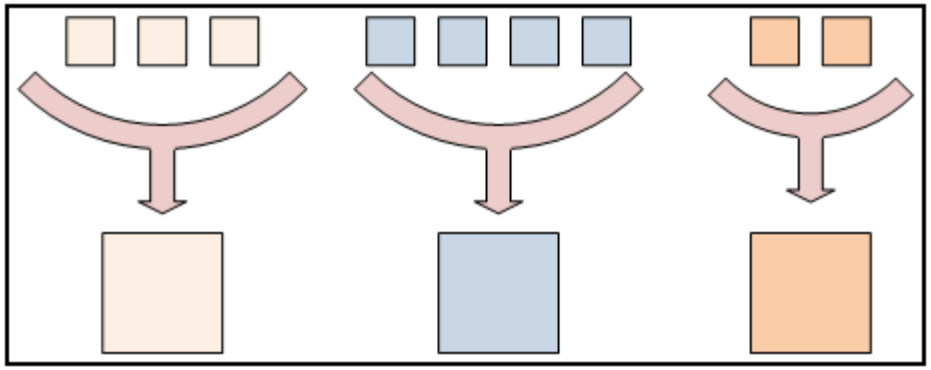
Ključevi i vrednosti 1

- Nijedna vrednost ne stoji sama, već svaka ima svoj ključ
- Ključevi identifikuju srodne vrednosti
- Mapiranje i redukcija ne primaju samo vrednosti, već parove <key, value>
- Primer:

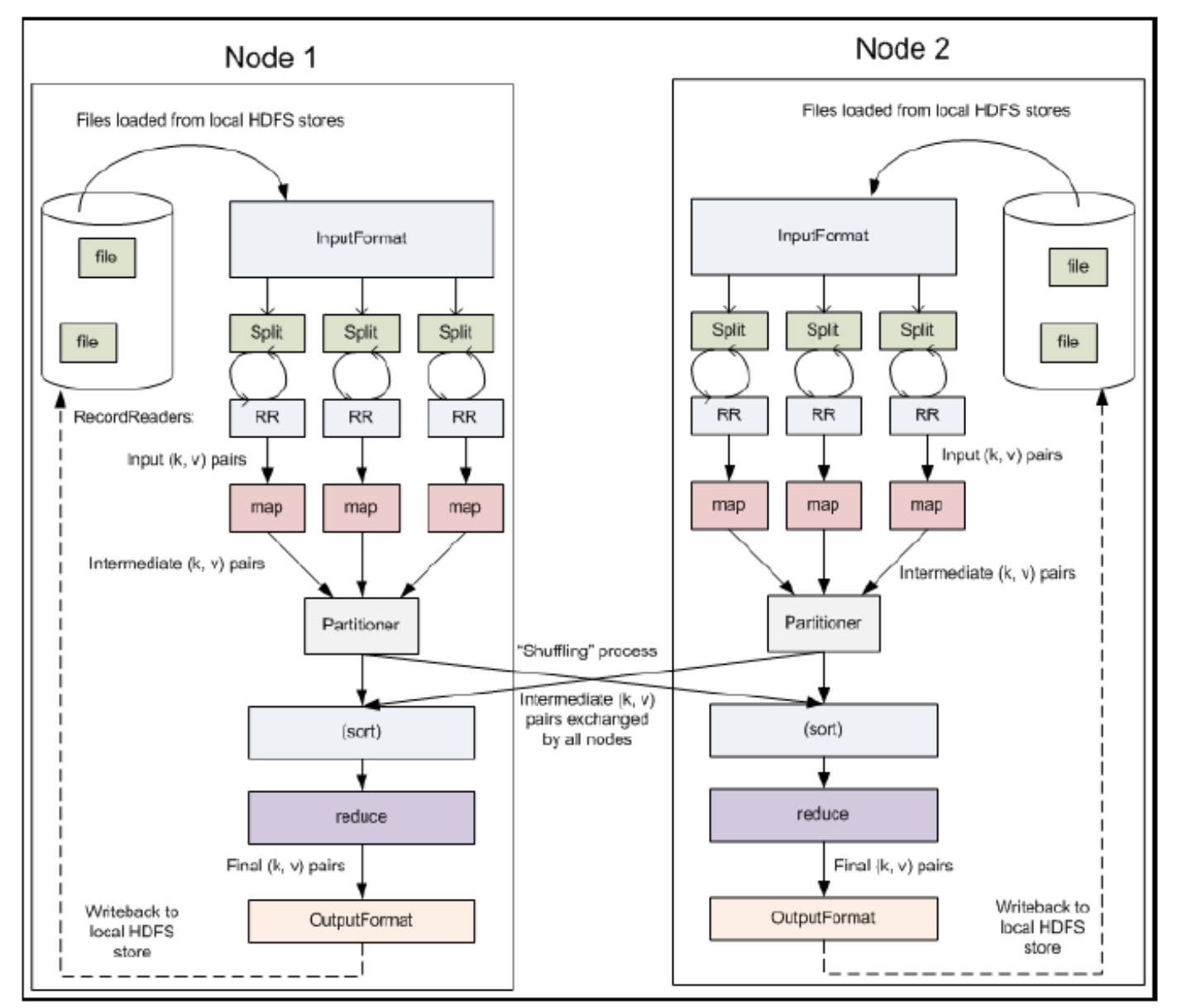
```
AAA-123    65mph, 12:00pm
ZZZ-789    50mph, 12:02pm
AAA-123    40mph, 12:05pm
CCC-456    25mph, 12:15pm
...
```

Ključevi i vrednosti 2

- Sve vrednosti sa istim ključem se šalju jednom reduktoru.
- Reduktor pretvara veliku listu vrednosti u jednu (ili nekoliko) izlaznih vrednosti.



Detaljniji opis toka podataka 1



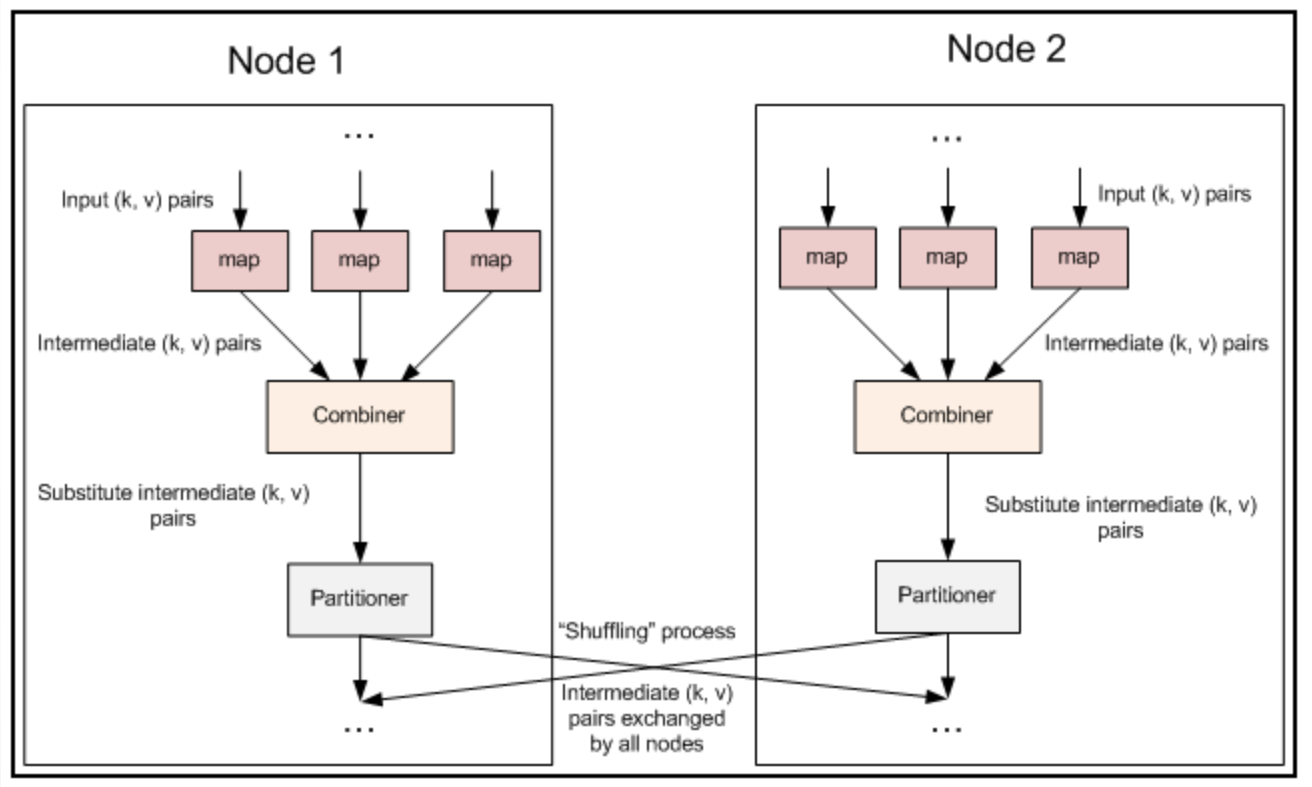
Detaljniji opis toka podataka 2

- **InputFormats**: Deniše kako su podeljeni fajlovi na manje komade. Postoji nekoliko tipova:
  - **TextInputFormat** - Ovo je osnovni format. Njegov ključ je bajt na početku linije, a vrednost sadržaj linije,
  - **KeyValueInputFormat** - Sastoji se iz (key, value) parova. Ključ je od početka linije do prvog tab-a, dok je vrednost ostatak linije,
  - **SequenceFileInputFormat** - Specijalni Hadoop binarni format. Ključ i vrednost se denišu od strane korisnika.
- **InputSplits**: Opisuje jedinicu posla koji svaki map zadatak izvršava. Map zadatak može čitati i ceo fajl, ali je najčešći slučaj da se čitaju samo delovi fajlova
- **RecordReader**: učitava podatke iz svog izvora i pretvara ih u parove (key, value) koji su pogodni za rad od strane klase Mapper.
- **Mapper**: Mapper obavlja korisnički denisan posao prve faze Map/Reduce programa. Klasa dobija ključ i vrednost a opciono može da prosledi par (ključ, vrednost) reduktorima

Detaljniji opis toka podataka 2

- **Partition and Shuffle**: Kada maperi završe obradu oni moraju razmeniti svoj izlaz sa reduktorima. Pre nego što reduktori dobiju ove podatke, oni se mešaju (shuffleing). Svi parovi sa istim ključem se prosleđuju istom reduktoru.
- **Sort**: Pre nego što se parovi (key, value) pošalju reduktoru, vrši se njihovo sortiranje
- **Reduce**: Reduce instanca se kreira za svaki redukcioni zadatak. Ova instanca pokreće kod koji je denisan od strane korisnika. Za svaku vrednost ključa reduce() funkcija se poziva se jednom poziva. Izlaz iz Reducer-a su parovi (key, value).
- **OutputFormat**: Definiše način na koji se upisuju izlazni parovi
- **RecordWriter**: Kao što InputFormat čita pojedinačne zapise pomoću klase RecordReader, OutputFormat klasa se služi klasom RecordWriter.

Dodatna Map/Reduce funkcionalnost (Combiner)



Svi delovi Map/Reduce posla

- Konguracija posla – **korisnik**
- Ulazna podela podataka (spliting) i njihova distribucija - Hadoop framework
- Startovanje pojedinačnih zadataka mapiranja za svaki deo fajla (split) - Hadoop framework
- Map funkcija, poziva se za svaki par (key, value) – **korisnik**
- Mešanje podataka po izlasku iz mapiranja - Hadoop framework
- Sortiranje izlaza posle mešanja podataka - Hadoop framework
- Startovanje pojedinačnih zadataka redukcije za svake sortirane delove fajla (split) – Hadoop framework
- Reduce funkcija, poziva se za sve parove (key, value) koji imaju istu vrednost key – **korisnik**
- Sakupljanje i skladištenje izlaznih podataka u direktorijom koji je naveden prilikom startovanja posla. Postojeća onoliko fajlova koliko je definisano zadataka redukcije - Hadoop framework

Primer

- WordCount.java
- Kompajliranje

```
mkdir wordcount_classes
javac -classpath   /var/lib/hadoop/hadoop-core-1.0.3.jar  -d wordcount_classes WordCount.java
jar -cvf wordcount.jar -C wordcount_classes/  .
```

- Pokretanje

```
hadoop  jar  wordcount.jar   WordCount   hdfs:/input   hdf:output
```