

Apache Spark

Osnove, osobine, primeri

Apache Spark

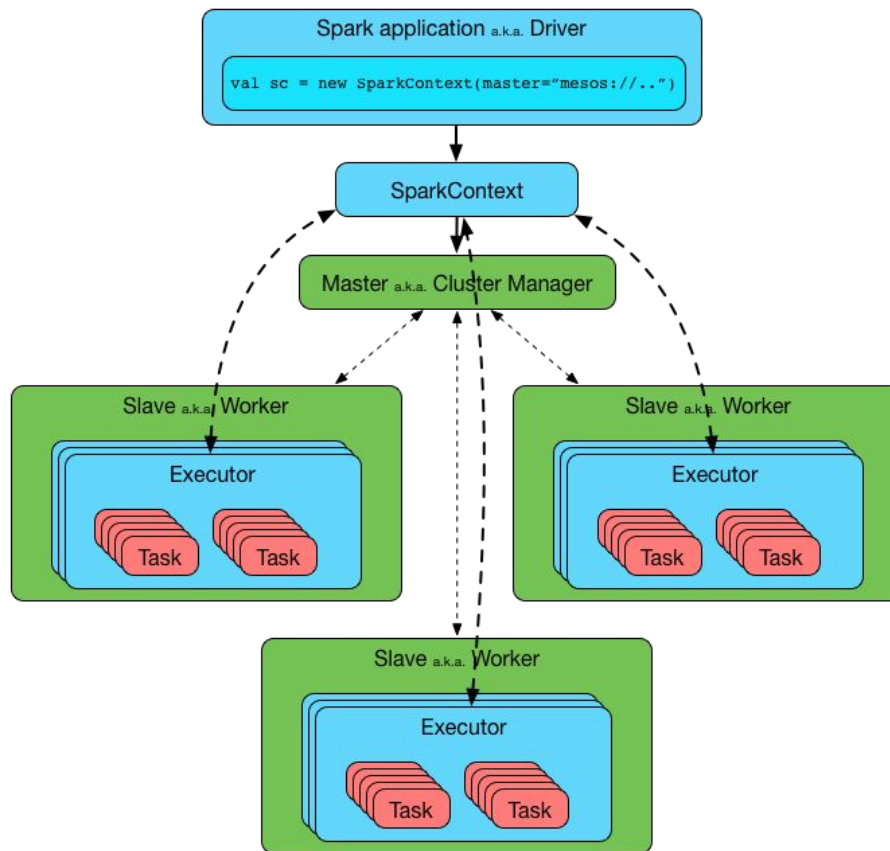
- Objedinjeno radno okruženje za analizu velike količine podataka
- Kombinuje SQL, tokove podataka i kompleksnu analitiku (mašinsko učenje)
- Radi nad različitim tipovima skladišta podataka
- Visoka performansa rada prilikom paketne i obrade tokova podataka
- Korisničke programe moguće je pisati u
 - Scala (izvorno)
 - Java
 - Python
 - SQL
 - R



Lightning-fast unified analytics engine

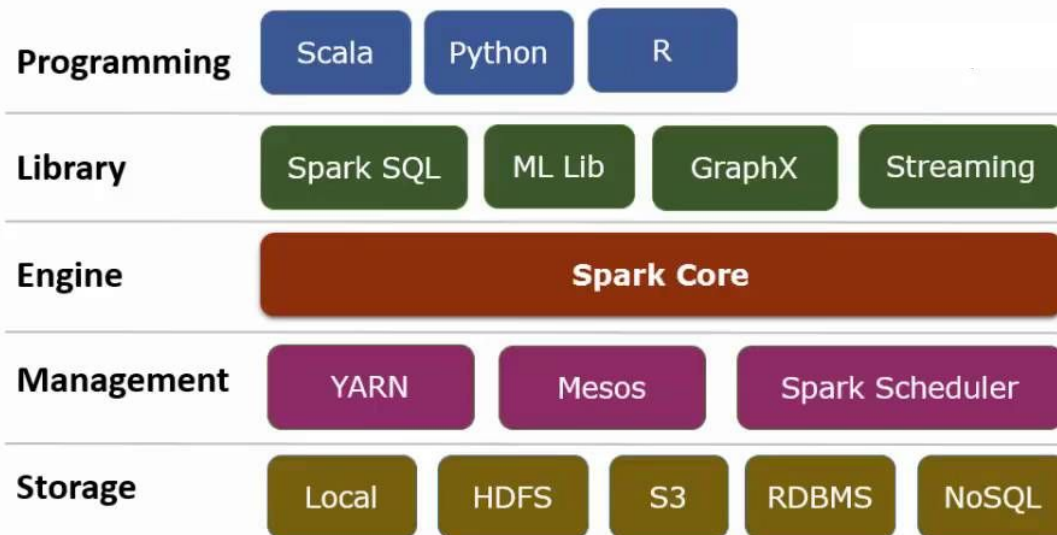
Spark - arhitektura

- Master/slave arhitektura
- Osnovni Spark moduli
 - Spark Core
 - Spark SQL
 - Spark Streaming
 - MLib
 - GraphX



Spark - arhitektura

Spark Framework

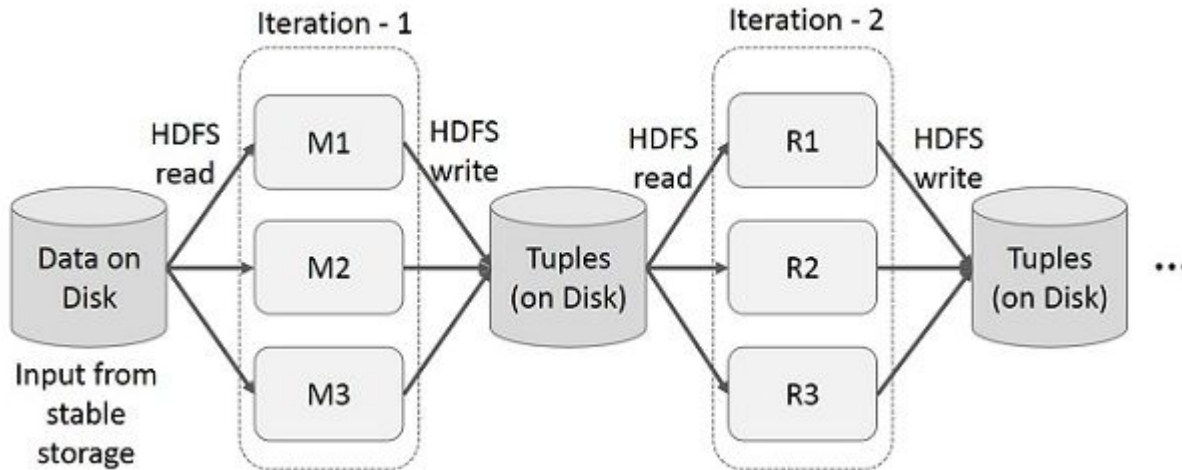


Docker compose

- Servisi
 - HDFS
 - Namenode
 - Datanode x 2
 - Spark
 - Spark-master
 - Spark-worker x 2
 - Opciono:
 - Hue - Web-bazirani SQL asistent (file-browser za HDFS, HIVE dashboard...)
 - Spark-notebook - Web-interfejs za pokretanje Spark aplikacija

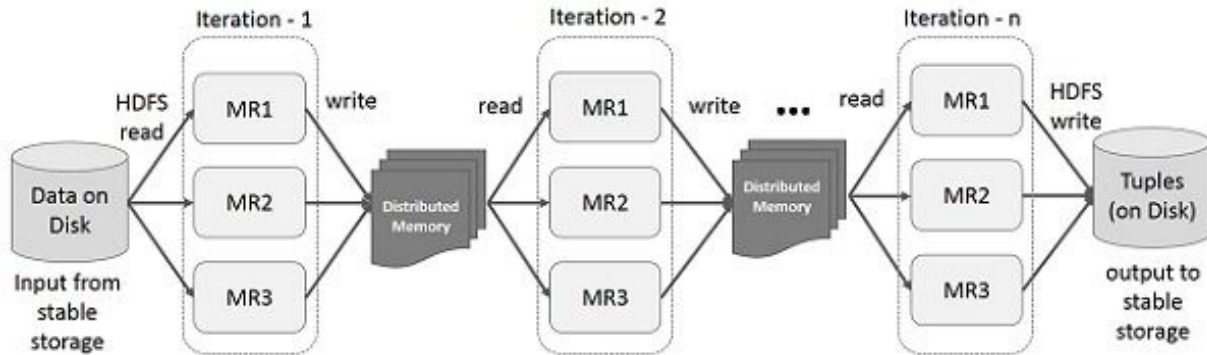
RDD

- Problem sa iterativnim *map-reduce* programima
 - Sporo deljenje memorije
 - Upis na disk nakon svakog *map-reduce* koraka
 - Previše (za krajnji rezultat) nepotrebnih I/O operacija



RDD

- *Resilient Distributed Datasets (RDD)*
 - Kolekcija elemenata distribuirana po čvorovima klastera koju je moguće obrađivati paralelno u radnoj memoriji
 - Eliminacija velikog broja I/O poziva značajno povećava brzinu izvršavanja iterativnih *map-reduce* programa



RDD - osnove

- Fundametalni koncept u Spark-u
- *Resilient Distributed Dataset*
 - *Resilient* - otporan na greške sa mogućnošću nadoknađivanja oštećenih/izgubljenih delova strukture podataka
 - *Distributed* - podaci su smešteni na više čvorova u klasteru
 - *Dataset* - predstavlja skup slogova (zapisa) u bilo kom formatu
 - najčešće JSON, CSV, tekstuelni format

RDD - mehanizmi

- Čitanje
 - Celokupnog dataseta (coarse-grained)
 - Pojedinačnih elemenata (fine-grained)
- Pisanje
 - Celokupnog dataseta (coarse-grained)
- Konzistentnost
 - Na visokom nivou s obzirom da je RDD nepromenljiv
- Oporavak od grešaka
 - Omogućen upotrebom usmerenih acikličnih grafova operacija (DAG)

RDD - osobine

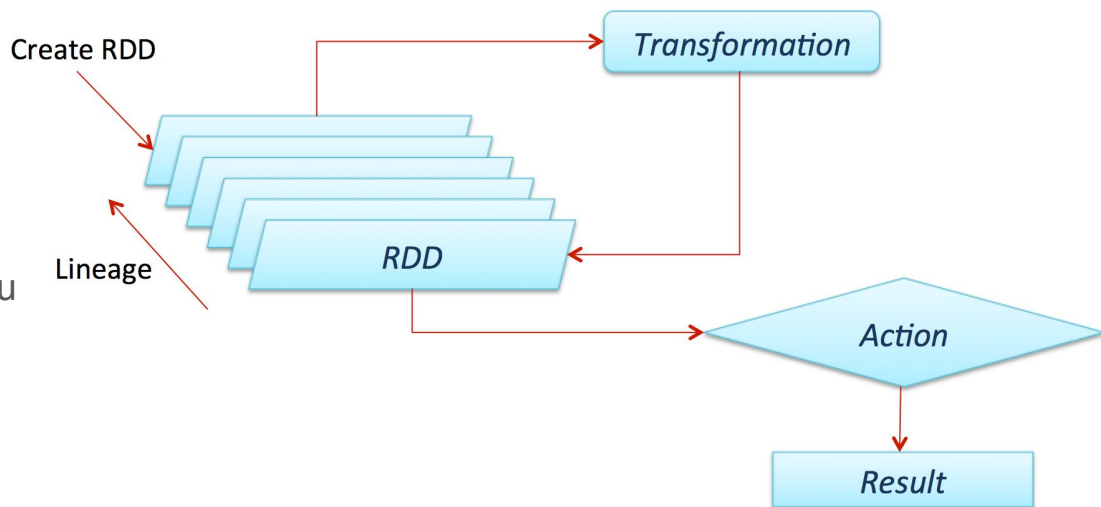
- *In-memory computation*
- Odložena (*lazy*) evaluacija
- Otpornost na greške
- Nepromenljivost
- Particionisanje
- Mogućnost perzistovanja
- Operacije nad celokupnim setom podataka

Deljene promenljive

- Spark nudi dva tipa deljenih promenljivih između zadataka koji se izvršavaju na različitim čvorovima klastera
 - *Broadcast* varijable
 - Keširane *read-only* promenljive na svakom čvoru
 - Često se koriste da svaki čvor dobije svoju kopiju nekog ulaznog skupa podataka
 - Akumulatori
 - Njihova vrednost može samo biti uvećana
 - Često se koriste za implementaciju agregacionih funkcija (npr. brojanje, suma)

RDD - operacije

- Transformacije
 - Od postojećeg RDD formiraju novi
 - Uske (*pipelining*)
 - Nad jednom particijom
 - Široke (*shuffle*)
 - Zahtevaju više particija
- Akcije
 - daju neki rezultat
 - Sve do nailaska na akciju Spark odlaže izvršenje transformacija



RDD - operacije

- Transformacije

- Uske

- Map
 - FlatMap
 - MapPartition
 - Filter
 - Sample
 - Union

- Široke

- Intersection
 - Distinct
 - ReduceByKey
 - AggregateByKey
 - SortByKey
 - Join
 - Cartesian
 - Repartition
 - Coalesce

- Akcije

- Count
 - CountByKey
 - Collect
 - First
 - Take
 - Top
 - CountByValue
 - Reduce
 - Fold
 - Aggregate
 - Foreach
 - SaveAsText
 - SaveAsSequenceFile
 - SaveAsObjectFile

RDD - ograničenja

- **Nepostojanje ugrađenog optimizatora operacija sa RDD**
 - Sva optimizacija prepuštena programeru
- **Obrada strukturiranih podataka**
 - RDD nema mehanizme da prepozna šemu ulaznog skupa podataka
- **Performansa**
 - RDD su JVM objekti i često uzrokuju preteranu upotrebu Garbage Collectora i Java serijalizacije
- **Skladištenje**
 - U slučaju da RDD ne može stati u radnu memoriju, sledi smeštanje jednog dela na disk

RDD - Primeri

- Brojanje reči
- Prosečna dužina reči
- Sve reči diže od X slova koje se pojavljuju više od Y puta
- Unija i presek
- Uklanjanje duplikata
- Grupisanje i sortiranje po ključu
- Spajanje
- Upotreba deljenih varijabli

Spark SQL

- Integriše relacioni način procesiranja podataka sa Spark funkcionalnom paradigmom
- Predstavlja Spark modul za procesiranje strukturiranih podataka
- Premošćava jaz između RDD koncepta i relacione tabele
- Deklarativni pristup na dva načina
 - DataFrame
 - Datasets API
 - Java i Scala



Spark SQL - osobine

- **Integrisanost**
 - SQL upiti sa Spark programima
- **Unificiran pristup podacima**
 - Izvori podataka mogu biti Hive, Parquet, JSON, JDBC, ...
- **Kompatibilnost**
 - Moguće pokretati Hive upite
- **Standardni mehanizmi konekcije**
 - Moguće se konektovati preko JDBC ili ODBC
- **Optimizacija**
 - Postoji ugrađeni optimizator upita

Spark SQL DataFrame

- Struktura podataka organizovana u imenovane tabele
 - Slično kao koncept R dataframe ili Python pandas dataframe
- Predstavlja reprezentaciju tabele iz relacione baze podataka u Spark SQL-u
- Posедуje šemu
 - Svaka kolona ima ime, tip i indikator dozvole NULL vrednosti
 - Moguće je programski definiti šemu
- Moguće je kreirati DataFrame od RDD
- DataFrame API se može koristiti u
 - Scala
 - Java
 - Python
 - R

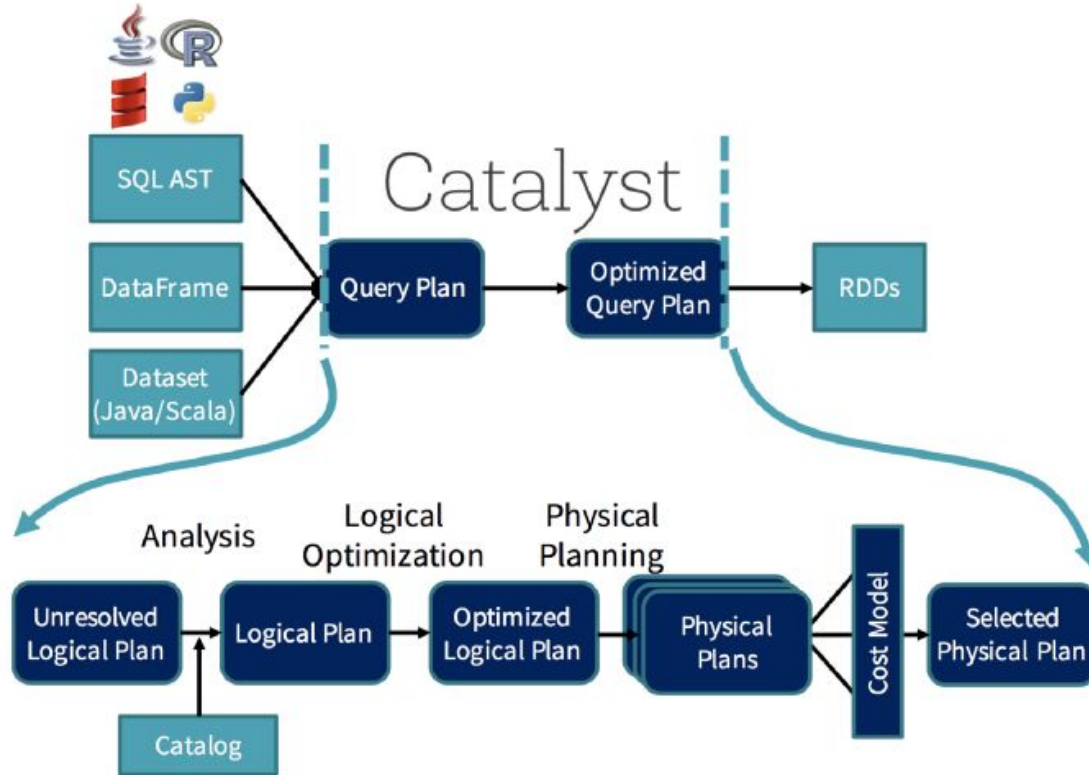
Spark SQL DataFrame - osobine

- Oslanja se na RDD i od njega nasleđuje
 - Nepromenljivost
 - Distribuirano izvršavanje u radnoj memoriji
 - Otpornost na otkaze
- Poboljšanja u odnosu na RDD
 - Efikasnije upravljanje memorijom (*Custom Memory Management*)
 - Optimizacija upita
- Daje se prednost *DataFrame*-u u odnosu na RDD prilikom obrade strukturiranih podataka

Spark SQL DataFrame - primeri

- <https://spark.apache.org/docs/latest/sql-getting-started.html>

Spark SQL DataFrame - Optimizator upita



DataFrame - kreiranje

- Od RDD-ova - `toDF()`
- Od listi - `toDF()`
- Od lokalnih ili HDFS datoteka
 - **CSV** - `spark.read.csv(path)`
 - **Text** - `spark.read.text(path)`
 - **JSON** - `spark.read.json(path)`
 - **Parquet** - `spark.read.parquet(path)`
- Preko JDBC konekcije
 - `spark.read.format("jdbc").option().load()`

DataFrame - dodavanje i uklanjanje kolona

- Dodavanje nove kolone

- `df.withColumn("Country", lit("USA"))`
- `df.withColumn("salary", col("salary")*100)`
- `df.withColumn("salary", col("salary").cast("Integer"))`

- Preimenovanje kolone

- `df.withColumnRenamed("gender", "sex")`

- Uklanjanje kolone

- `df.drop("CopiedColumn")`

DataFrame - filtriranje

- Filtriranje
 - `df.filter(df["state"] == "OH").show(truncate=False)`
 - `df.filter("gender == 'M'").show(truncate=False)`
 - `df.where("gender == 'M'").show(truncate=False)`
 - `df.filter(df("state") == "OH"
 & df("gender") == "M").show(truncate=False)`
- Uslov može biti zadat kao
 - Poređenje
 - String sa logičkim izrazom
 - Kompleksan logički uslov

DataFrame - tipovi podataka i konverzije

- Podržani tipovi podataka
 - Numeric tipovi
 - ByteType, ShortType, IntegerType, LongType
 - FloatType, DoubleType, DecimalType
 - Stringovi - StringType
 - Binarni - BinaryType
 - Logički - BooleanType
 - Datumski - TimestampType, DateType
 - Kompleksni tipovi
 - ArrayType
 - MapType
 - StructType

DataFrame - pivoting

- Rotiranje podataka iz jedne u više kolona po vrednostima
 - `df.groupBy("Product").pivot("Country").sum("Amount")`
 - Za svaku vrednost iz kolone Country napraviće se posebna kolona
- Depivotiranje
 - `pivotDF.select(col("Product"),
 expr("stack(3, 'Canada', Canada, 'China', China,
 'Mexico', Mexico) as (Country,Total)")).where("Total is not null")`
 - Stack funkcija od kolona i vrednosti u njima prvi redove

DataFrame - spajanje

- Tipovi spajanja
 - INNER, LEFT OUTER, RIGHT OUTER, LEFT ANTI, LEFT SEMI, CROSS, SELF
- Optimizacija spajanja
 - BROADCAST, MERGE, SHUFFLE_HASH and SHUFFLE_REPLICATE_NL
 - Hint
 - `SELECT /*+ COALESCE(3) */ * FROM t`
 - `SELECT /*+ REPARTITION(3) */ * FROM t`
 - `SELECT /*+ REPARTITION(c) */ * FROM t`
 - `SELECT /*+ REPARTITION(3, c) */ * FROM t`
 - `SELECT /*+ REPARTITION_BY_RANGE(c) */ * FROM t`
 - `SELECT /*+ REPARTITION_BY_RANGE(3, c) */ * FROM t`
 - “Zakrivljenost” (skewness) podataka - neuniformna distribucija vrednosti ključa
 - “Soljenje” (salting) ključa dodavanjem još jedne kolone u ključ radi ravnomernijeg particionisanja

DataFrame - perzistencija

- Čuvanje međurezultata
- Korišćenje međurezultata u uzastopnim operacijama
- `cache()`
 - `MEMORY_AND_DISK` - DataFrame
 - `MEMORY_ONLY` - RDD
- `persist()`
 - `MEMORY_ONLY`
 - `MEMORY_AND_DISK`
 - `MEMORY_ONLY_SER`
 - `MEMORY_AND_DISK_SER`
 - `DISK_ONLY`
 - `MEMORY_ONLY_2`
 - `MEMORY_AND_DISK_2`

DataFrame - agregacije

- Agregacione funkcije
 - sum, avg, min, max, count, countDistinct
 - first, last, mean, stddev, variance, approx_count_distinct
- Sa ili bez groupBy
 - ```
df.groupBy("department")
 .agg(
 sum("salary").as("sum_salary"),
 avg("salary").as("avg_salary"),
 sum("bonus").as("sum_bonus"),
 max("bonus").as("max_bonus"))
 .show(truncate=False)
```

# DataFrame - analitičke funkcije

- Definisanje prozora - Window
  - `windowSpec = Window.partitionBy("department").orderBy("salary")`
- Rangiranje
  - `row_number`, `rank`, `percent_rank`, `dense_rank`, `ntile`
- Analitičke funkcije
  - `cume_dist`, `lead`, `lag`
- Agregacione funkcije

# Zadatak

- Preuzeti CSV datoteku  
<https://data.gov.rs/sr/datasets/zagadjivachi-chvrstim-otpadom/>
- Kopirati je na proizvoljnu HDFS lokaciju
- Napisati Spark aplikaciju koja učitava sadržaj CSV datoteke u dataframe
  - Separator je ;
  - Encoding je Windows-1250
- Ispisati shemu učitano datafraame-a
- Ispisati prvih 5 redova
- Ispisati sve različite opštine

# Zadatak

- Ispisati preduzeća koja su 2015 proizvela preko 10 tona otpada
- Ispisati sve pretežne delatnosti sa preko 10 postrojenja
- Ispisati top 5 zagađivača u oblasti *Proizvodnja komunikacione opreme*
- Ispisati najveće zagađivače po gradovima
- Kolone sa 2010-2017 pretvoriti u jednu kolonu *godina*, a iznos količine otpada u kolonu *kolicina*
- Ispisati preduzeća (PIB + Postrojenje) sa najvećim skokom proizvodnje neke vrste otpada (Indeksni broj) u odnosu na prethodnu godinu
- Rezultat sačuvati u relacionu bazu podataka (npr. MySQL, PostgreSQL)
  - Proširiti docker-compose kontejnerom za bazu podataka
  - Koristeći [uputstvo](#), sadržaj dataframe-a upisati u tabelu u bazi podataka

# Spark MLlib

- MLlib - Sparkova biblioteka za mašinsko učenje zasnovana na RDD
  - RDD API danas zastareo pa ga nije preporučljivo koristiti
- DataFrame-based API - preporučeno za korišćenje - Spark ML
- Osnovni koncepti
  - DataFrame - ulazni dataset
  - Transformer - algoritam za transformaciju DataFrame-a (npr. dodavanje kolone sa predikcijama)
  - Estimator - algoritam za kreiranje transformatora (npr. fabrika modela)
  - Pipeline - estimatori i transformatori povezani u neki smislen tok
  - Parameter - podešavanja estimatora i transformatora

# Spark MLlib

- Osnovne statistike
  - Srednja vrednost, variјansa, kovariјsna, korelacija, testiranje hipoteze
- Klasifikacija i regresija
  - Linearni modeli, naivni Bayes, stabla odlučivanja, ansambli stabala
- Klasterovanje
  - k-means, Gaussian Mixture
- Kolaborativno filtriranje
- Redukcija dimenzionalnosti
  - SVD, PCA