

Uvod

Apache Spark

Apache Spark

- Apache Spark je “*fast, general-purpose, distributed big data processing platform*”.
- Efikasna upotreba memorije
 - Do 100 puta brži od sličnih platformi
- API zasnovan na skupu collection-based procedura koji sakriva da se radi sa klasterom mašina

Osobine

- Distribuirana platforma, ali sa mogućnošću da se velike količine podataka u memoriji
- Spark kolekcija sakriva činjenicu da se referenciraju podaci sa različitih čvorova
- Podržava
 - Batch programiranje, real-time procesiranje, SQL-like upite, algoritme na grafovima, algoritme mašinskog učenja
 - Python, Java, Scala, R
- Nije pogodan za OLTP obradu ili kada dataset može da obradi jedna mašina

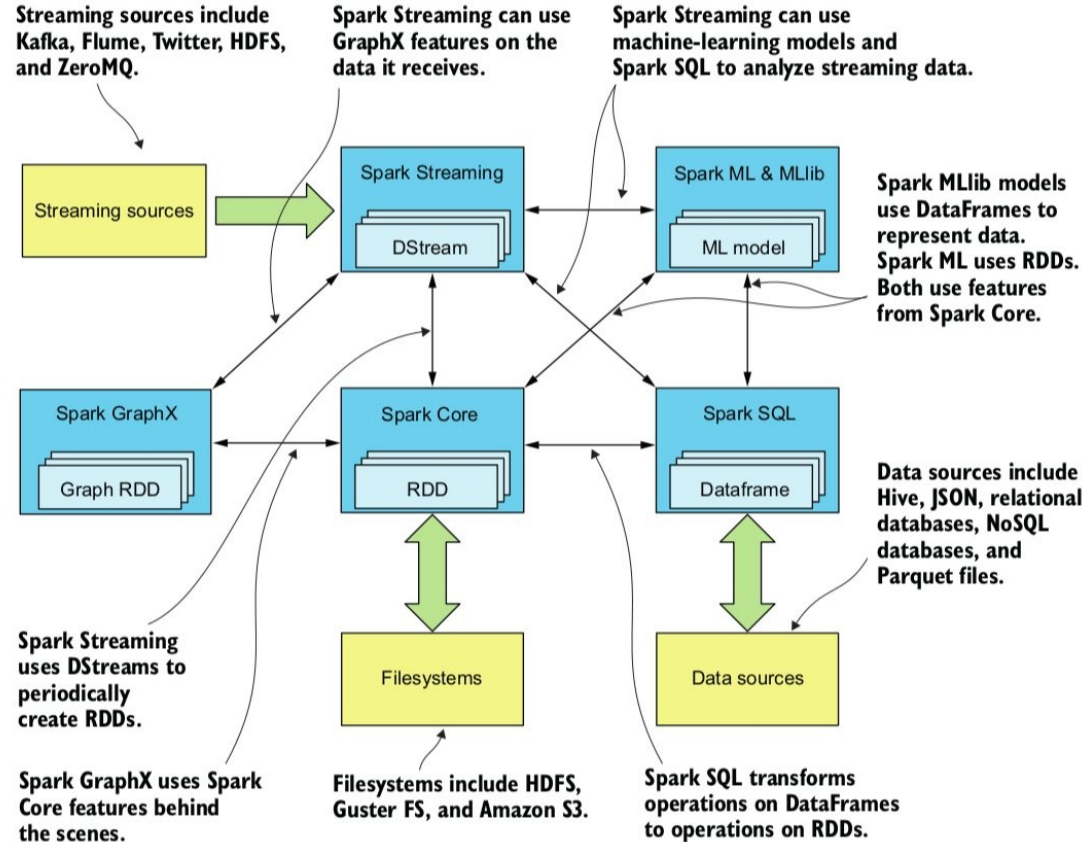
Hadoop

- Osobine
 - Paralelizam
 - Distribuiranost
 - Fault tolerance
- Hadoop =
 - HDFS (Hadoop Distributed File System)
 - MapReduce data-processing engine, pri čemu rezultat jednog koraka mora da bude sačuvan u HDFS da bi ga upotrijebio sljedeći korak u postupku obrade

Spark + Hadoop

- Jedinstvena platforma uz bogat API
- Keširanje podataka koji nastaju kao rezultat međukoraka
 - In-memory execution model
- Sortiranje 100TB podataka za manje od 1.5 sekundi (sortbenchmark.org)

Spark komponente



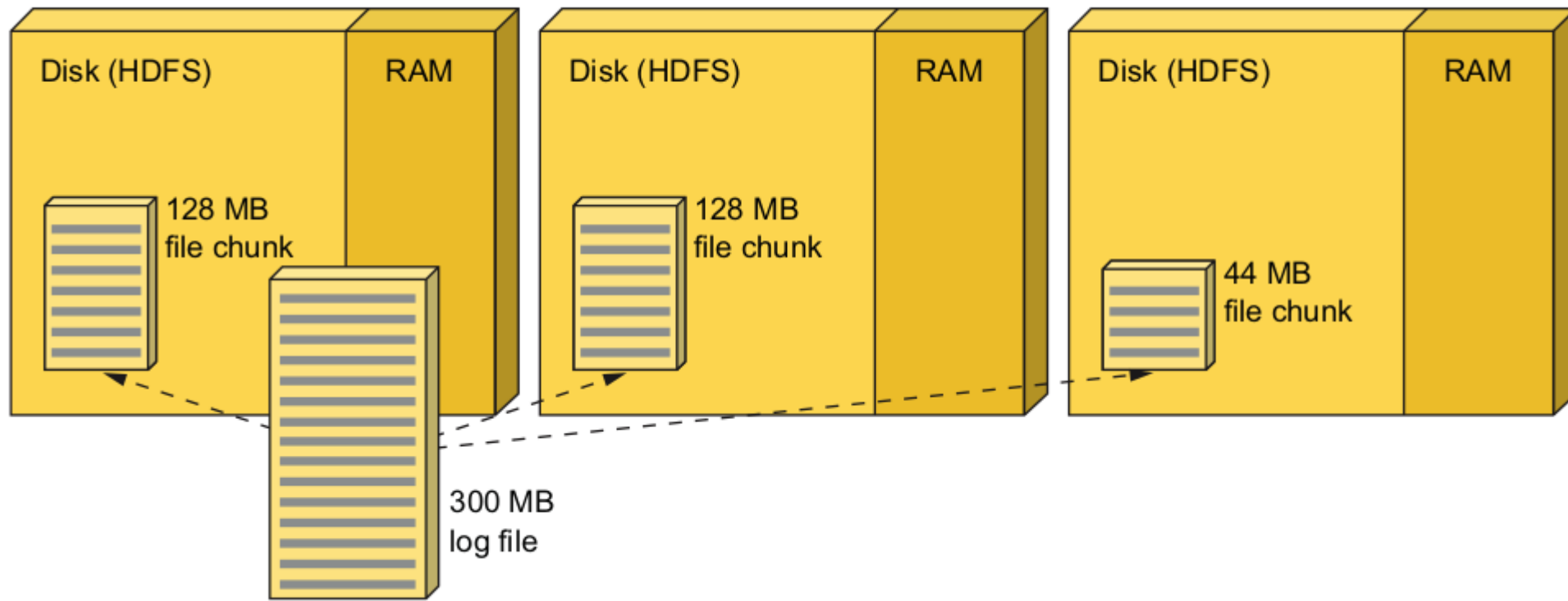
Spark komponente 2

- Spark Core, Resilient Distributed Dataset – RDD je apstrakcija distribuirane kolekcije
 - Immutable, resilient, distributed
 - Transformacije i akcije
- Spark SQL sa Dataframe-ovima omogućava manipulisanje velikim skupovima distribuiranih strukturiranih podataka. Operacije nad DataFrame-ovima mapiraju se na RDD operacije

Spark komponente 3

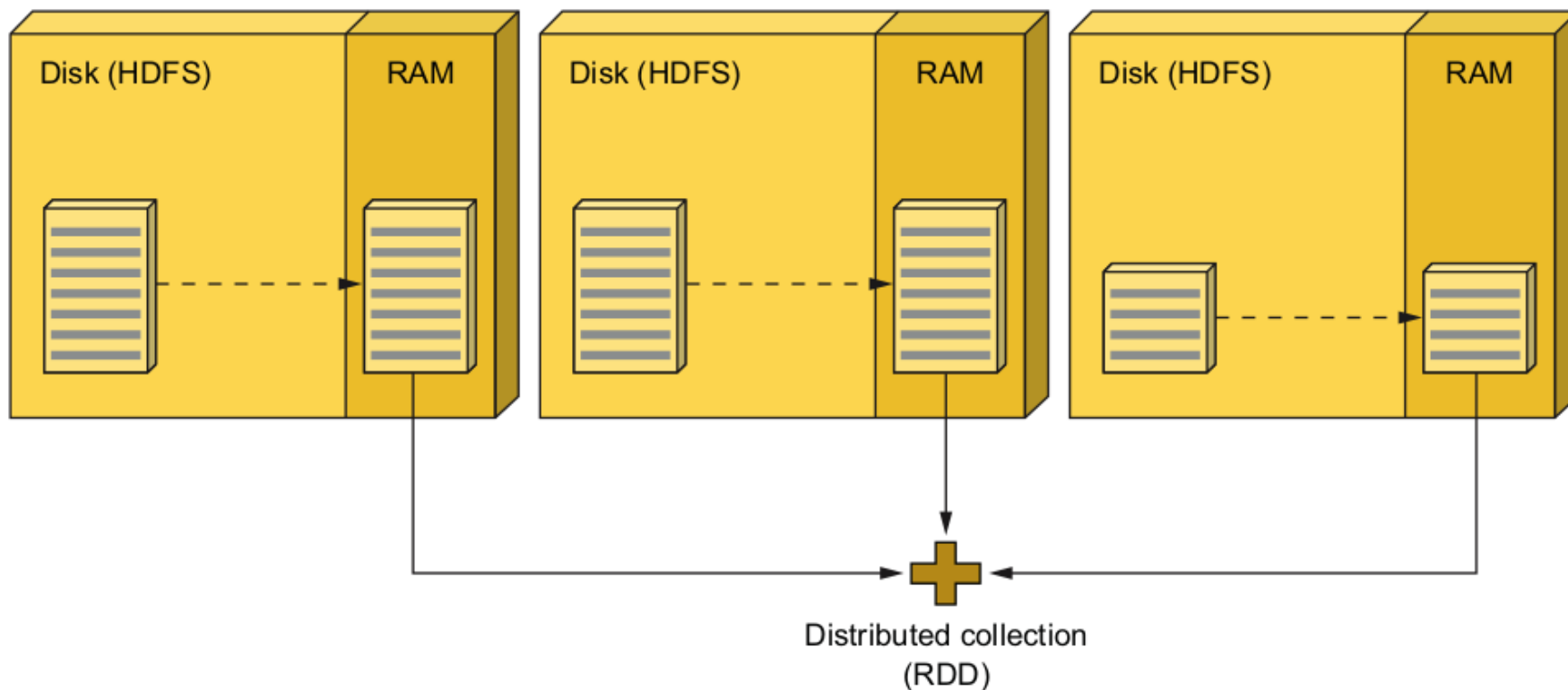
- Spark streaming – procesiranje real-time strimova podataka. Izvori mogu da budu razni, HDFS, ZeroMQ, Twitter itd. Dstream je RDD koji sadrži podatke iz posljednjeg “prozora”
- Spark Mllib je biblioteka za algoritme mašinskog učenja
- Spark GraphX sadrži metode za kreiranje i rad sa graph RDD-ovima.

Spark program flow



Spark program flow 2

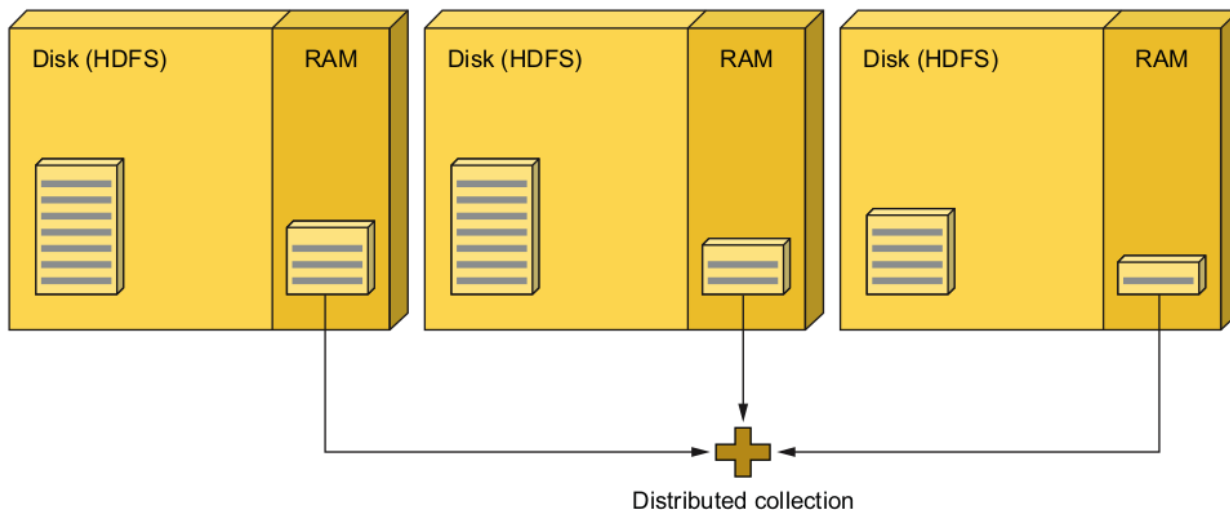
```
val lines = sc.textFile("hdfs://path/to/the/file")
```



Spark program flow 3

- Koliko se grešaka tipa *OutOfMemory* generisalo tokom posljednje dvije sedmice?

```
val oomLines = lines.filter(l => l.contains("OutOfMemoryError")).cache()
```



```
val result = oomLines.count()
```

Spark shell

- pip install pyspark

- pyspark

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

      /_/_/  _/_/  _/_/  _/_/  _/_/
     /_/_/  /_/_/  /_/_/  /_/_/  /_/_/
    /_/_/  /_/_/  /_/_/  /_/_/  /_/_/
   /_/_/  /_/_/  /_/_/  /_/_/  /_/_/
  /_/_/  /_/_/  /_/_/  /_/_/  /_/_/
 /_/_/  /_/_/  /_/_/  /_/_/  /_/_/
/_/_/  /_/_/  /_/_/  /_/_/  /_/_/

version 3.1.2

Using Python version 3.8.10 (default, Jun  2 2021 10:49:15)
Spark context Web UI available at http://192.168.1.33:4040
Spark context available as 'sc' (master = local[*], app id = local-1633970851451).
SparkSession available as 'spark'.
>>> lines = spark.read.text('/home/hadoop_log.log')
>>> lines.count()
377
>>> 
```

Primjer

- Fajl `clients_ids.log` sadrži ID klijenata koji su pokrenuli neku transakciju tokom dana, jedan red u fajlu odnosi se na jedan dan
- Potrebno je naći ID klijenata koji su realizovali bar jednu transakciju tokom posljednje sedmice

```
15, 16, 20, 20  
77, 80, 94  
94, 98, 16, 31  
31, 15, 20
```

Rješenje

```
/clients_ids.log  
>>> clients = spark.read.text('clients_ids.log')  
>>> clients.show()
```

```
+-----+  
|      value|  
+-----+  
|15, 16, 20, 20|  
|   77, 80, 94|  
|94, 98, 16, 31|  
|   31, 15, 20|  
+-----+
```

```
>>> █
```

```
>>> df1 = clients.rdd.flatMap(lambda line: line)  
>>> df1.collect()  
['15, 16, 20, 20', '77, 80, 94', '94, 98, 16, 31', '31, 15, 20']  
>>> df2 = df1.flatMap(lambda item: item.split(", "))  
>>> df2.collect()  
['15', '16', '20', '20', '77', '80', '94', '94', '98', '16', '31', '31', '15', '20']  
>>> unique = df2.distinct()  
>>> unique.count()  
8  
>>> unique.collect()  
['15', '16', '20', '77', '80', '94', '98', '31']  
>>>
```

Kreiranje uzorka

```
[ 0.0]
>>> sample = unique.sample(False, 0.3)
>>> sample.count()
4
>>> sample.collect()
['80', '94', '98', '31']
>>> sample = unique.sample(False, 0.3)
>>> sample.collect()
['20', '94']
>>>
```

```
>>> sample3 = unique.takeSample(False, 5)
>>> sample3
['20', '98', '94', '16', '77']
>>> sample4 = unique.takeSample(True, 5)
>>> sample4
['16', '98', '98', '94', '31']
>>>
```

Osnovne statističke funkcije

```
>>> idsInt.collect()
[15, 16, 20, 77, 80, 94, 98, 31]
>>> intIds = unique.map(lambda el: int(el))
>>> intIds.max()
98
>>> intIds.mean()
53.875
>>> intIds.sum()
431
>>>
```

Histogram

```
>>> intIds.histogram([10, 20, 40, 100])  
([10, 20, 40, 100], [2, 2, 4])  
>>> intIds.histogram(3)  
([15.0, 42.66666666666667, 70.33333333333334, 98], [4, 0, 4])  
>>>
```

Spark Core API

<https://github.com/spark-in-action/first-edition>

Zadatak

- Potrebno je da se generiše izvještaj o broju operacija PUSH za svakog korisnika
- Dataset
`wget https://data.gharchive.org/2015-01-01-{0..23}.json.gz`

Dataset

```
object {7}
  id : 2489368070
  type : PushEvent
  actor {5}
    id : 9152315
    login : davidjhulse
    gravatar_id : {value}
    url : https://api.github.com/users/davidjhulse
    avatar_url : https://avatars.githubusercontent.com/u/9152315?
  repo {3}
  payload {7}
    push_id : 536740396
    size : 1
    distinct_size : 1
    ref : refs/heads/master
    head : a9b22a6d80c1e0bb49c1cf75a3c075b642c28f81
    before : 86ffa724b4d70fce46e760f8cc080f5ec3d7d85f
    commits [1]
  public : true
  created_at : 2015-01-01T00:00:00Z
```

Dataframe API

- Dataframe je RDD sa šemom, slično tabeli iz relacionog modela
- Kreiranje dataframe iz JSON fajla – automatsko izvođenje šeme
- Sintaksa je SQL-like i omogućava deklarativni način zadavanja upita
 - Razlika u odnosu na Spark Core API

Dataframe API 2

- <http://spark.apache.org/docs/latest/api/python/reference/pyspark.sql.html#dataframe-apis>
 - `pyspark.sql.Session.read`
 - Moguće je pročitati više fajlova u jedan dataframe
 - `pyspark.sql.DataFrame.printSchema`
 - `pyspark.sql.DataFrame.count`
 - `pyspark.sql.DataFrame.show`
 - `pyspark.sql.DataFrame.filter`

Primjer rješenja

- pyspark < main.py
ili
- spark-submit main.

```
main.py x
0 results
1 from pyspark.context import SparkContext
2 from pyspark.sql.session import SparkSession
3
4 if __name__ == '__main__':
5     sc = SparkContext('local')
6     spark = SparkSession(sc)
7
8     inputPath = "../GitHub archive dataset/2015-01-01-0.json"
9     ghLog = spark.read.json(inputPath)
10    ghLog.printSchema()
11    print(ghLog.count())
12
13    pushes = ghLog.filter(ghLog.type == "PushEvent")
14    print(pushes.count())
15    pushes.show()
16
17    print('Published')
```

Agregiranje podataka

- `pyspark.sql.DataFrame.groupBy`
- `pyspark.sql.DataFrame.orderBy`

```
grouped = ghLog.groupBy(ghLog.actor.login).count()
grouped.show(5)

ordered = grouped.orderBy(grouped["count"].desc())
ordered.show(5)
```