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

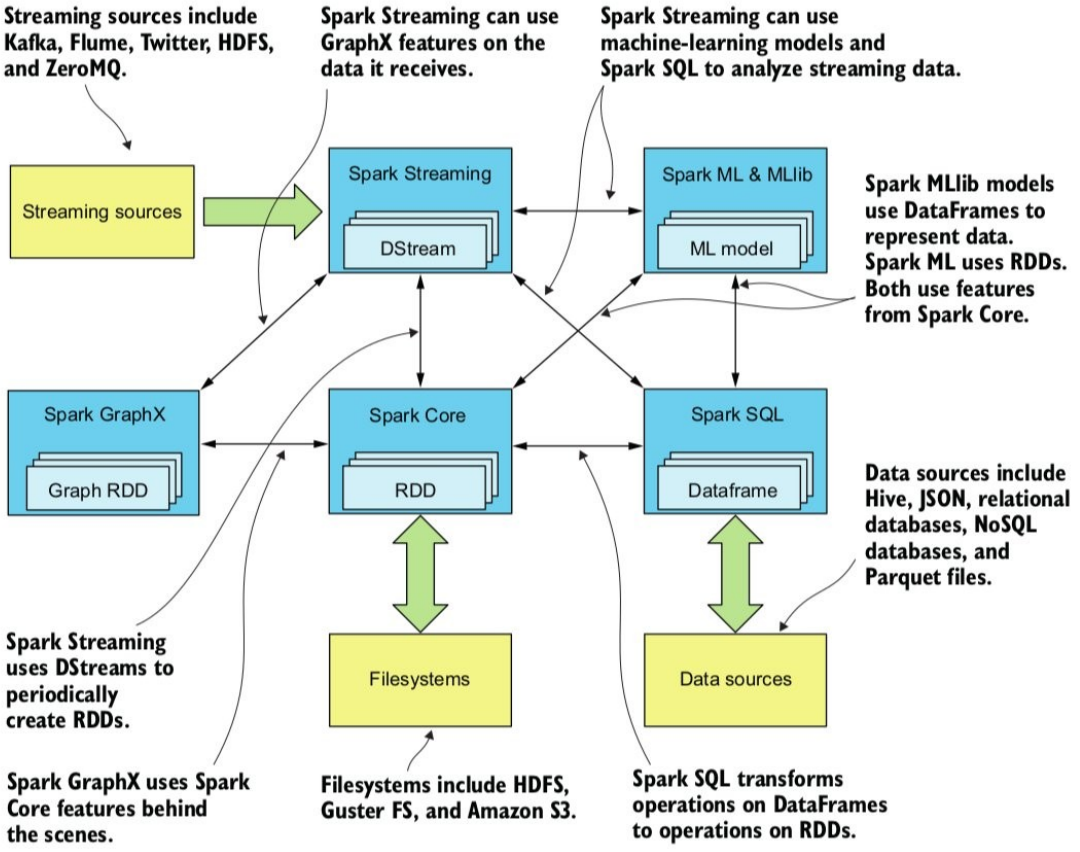
Hadoop

- 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



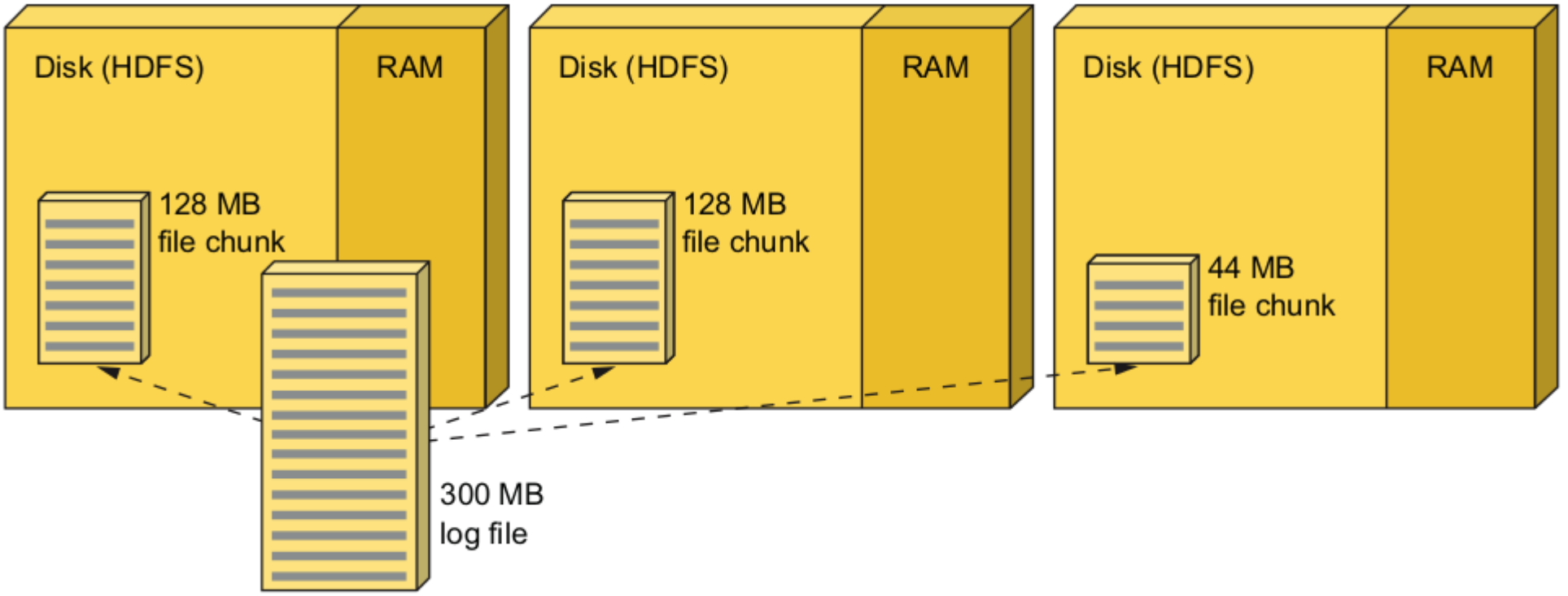
Spark komponente 2

- 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

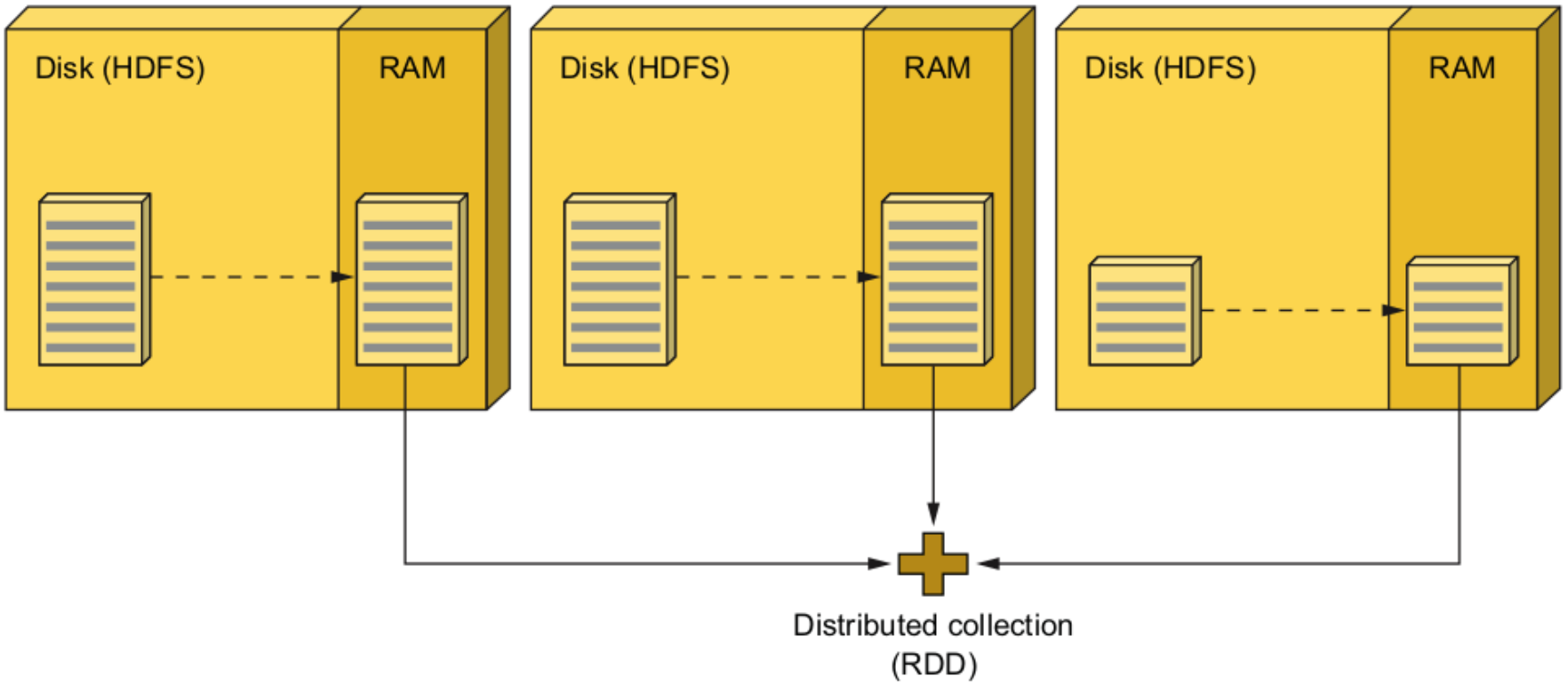
- Spark streaming – procesiranje real-time strimova podataka. Izvori mogu da budu razni, HDFS, ZeroMQ, Twitter itd. Dstream je RDD koji sadrži podatke iz posljednjeg “prozora”
- Spark Mllib je biblioteka za algoritme mašinskog učenja
- Spark GraphX sadrži metode za kreiranje I rad sa graph RDD-ovima.

Spark program flow



Spark program flow 2

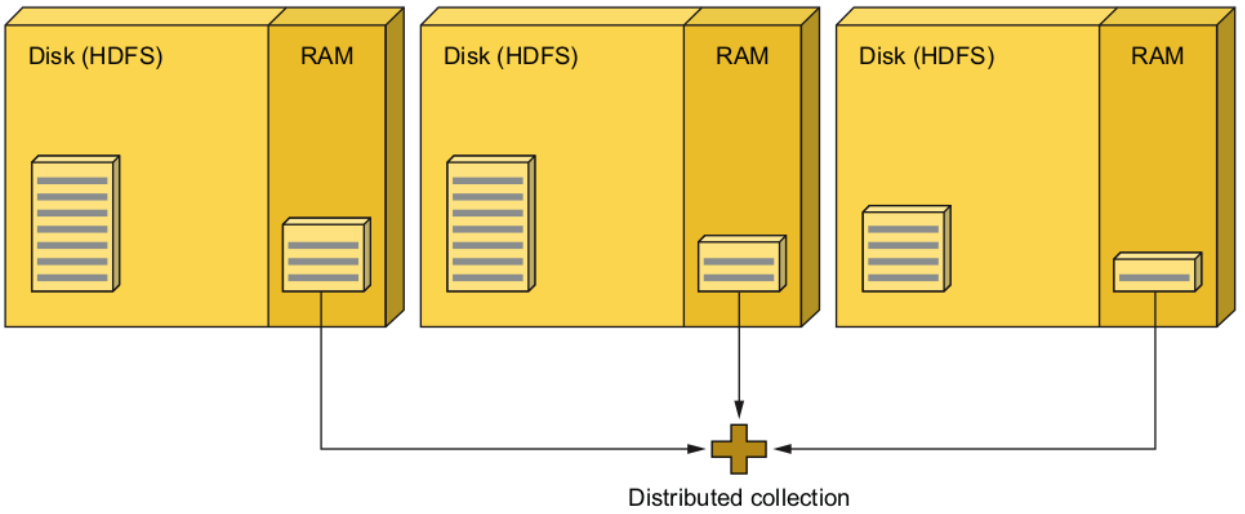
```
val lines = sc.textFile("hdfs://path/to/the/file")
```



Spark program flow 3

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

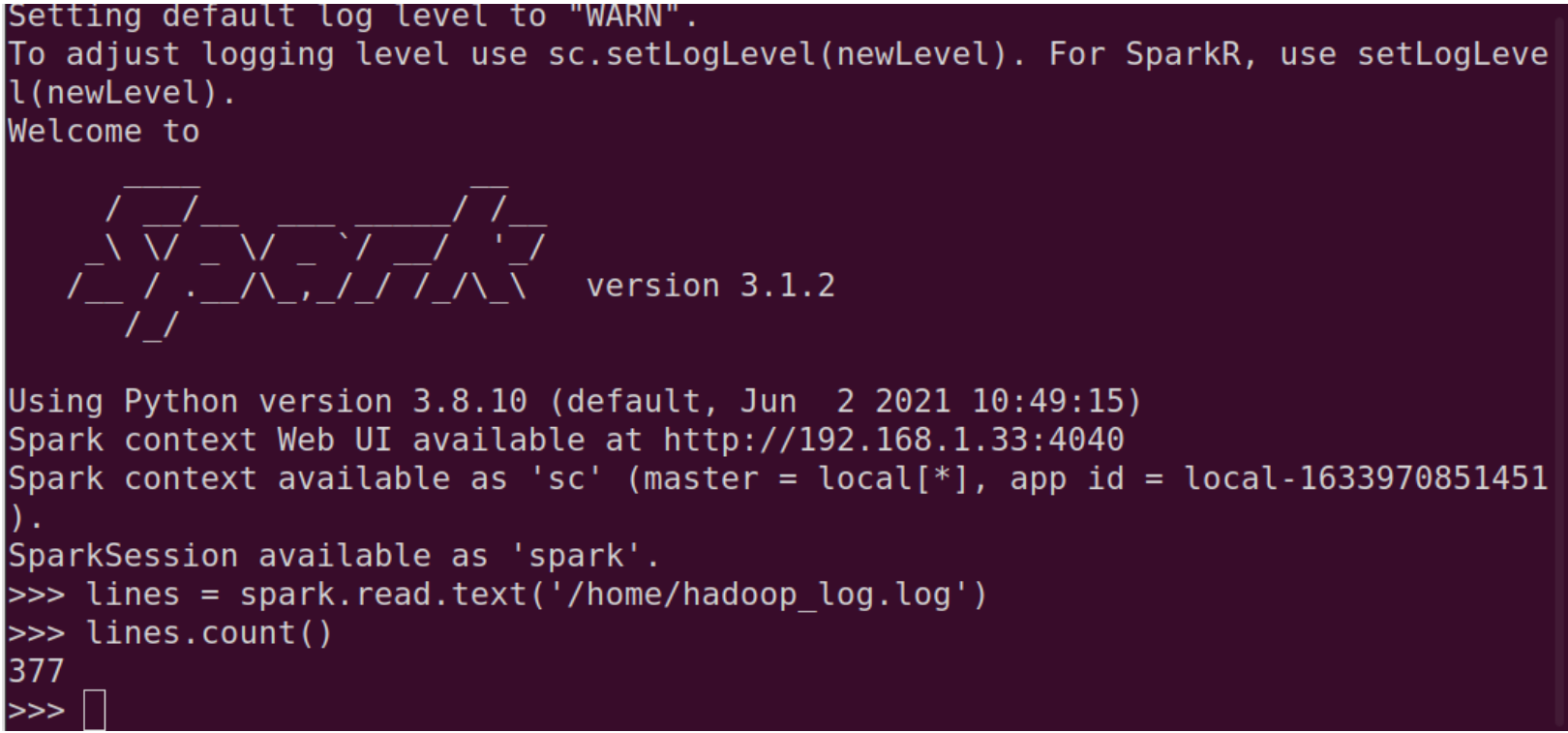
```
val oomLines = lines.filter(l => l.contains("OutOfMemoryError")).cache()
```




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

Spark shell

- pip install pyspark
- pyspark



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

```
>>> 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, 42.666666666666667, 70.333333333333334, 98], [4, 0, 4])
>>>
```