RDD, DataFrame, Dataset

RDD

- Resilient Distributed Dataset
- Koncepcja RDD pojawiła się w artykule: Matei Zharia, et al. *Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing*
- Podstawowa abstrakcja danych oraz rdzeń Sparka
- Niemutowalna rozproszona kolekcja rekordów, rozdzielona na jednej lub więcej partycji (węzłach klastra)
- Obsługiwana przez dwa rodzaje operacji: akcje i transformacje

RDD

- **Resilent** – odporny na uszkodzenia dzięki lineage graph, który pozwala na odtworzenie utraconych lub uszkodzonych partycji powstałych w wyniku awarii węzłów
- **Distributed** – dane znajdują się na wielu węzłach w klastrze
- **Dataset** – kolekcja podzielonych danych z prymitywnymi wartościami

RDD

- In-Memory – dane w RDD są przechowywane w pamięci
- Immutable – RDD nie można zmienić, stworzone RDD może zostać jedynie przekształcone w nowe RDD z wykorzystaniem transformacji
- Lazy evaluated – dane w RDD nie są dostępne/przetransformowane do czasu aż nie zostanie wywołana akcja uruchamiająca egzekucję
- Cacheable – RDD może zostać zapisane w pamięci lub na dysku
- Parallel – przetwarzane równolegle
- Typed – rekordy w RDD mają przypisane typy np. RDD[String]
- Partitioned – rekordy są podzielone na logiczne partycje i rozproszone na węzłach klastra

RDD Persistence

- Metody *cache* i *persist* pozwalają na zachowanie RDD w pamięci
- Zachowanie RDD w pamięci ma duży wpływ na szybkość działania programu jeśli do danego RDD odołuje się wielokrotnie, jak to ma miejsce w algorytmach iteracyjnych
- 3 podstawowe poziomy składowania (storage levels) RDD:
  - MEMORY_ONLY
  - MEMORY_AND_DISK
  - DISK_ONLY
- Usunięcie RDD z pamięci możliwe jest dzięki metodzie *unpersist*

RDD - operacje

Akcje:

- Operacje uruchamiające egzekucję transformacji na RDD
- Powodują ewaluację RDD lineage graph
- Funkcje przyjmujące RDD jako input i zwracające wartość nie będącą RDD
- Przykłady: reduce, collect, count, first, take, saveAsTextFile, takeSample

Transformacje:

- Leniwe operacje, wykonywane dopiero po wywołaniu akcji
- Funkcje przyjmujące RDD jako input i zwracające jedną lub więcej nowych RDD
- Dokonując transformacji przyrostowo budowany jest RDD lineage graph
- Przykłady: map, filter, flatMap, union, groupByKey, reduceByKey, distinct

RDD lineage graph

- DAG (Directed Acyclic Graph)
- Logiczny plan egzekucji
- Graf zawierający nadrzędne (parent) RDD danego RDD
- Metoda *toDebugString* pozwala zobaczyć lineage graph

```
scala> val test = sc.parallelize(1 to 10).map(_+1).filter(_>3)
test: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[2] at filter at
<console>:24
scala> test.toDebugString
res0: String =
(8) MapPartitionsRDD[2] at filter at <console>:24 []
 |  MapPartitionsRDD[1] at map at <console>:24 []
 |  ParallelCollectionRDD[0] at parallelize at <console>:24 []
```

RDD - tworzenie

Lokalne kolekcje:

- SparkContext.parallelize
- SparkContext.makeRDD

Pliki:

- SparkContext.textFile
- SparkContext.hadoopFile
- SparkContext.pickleFile
- ...

Inne RDD:

- Transformacje

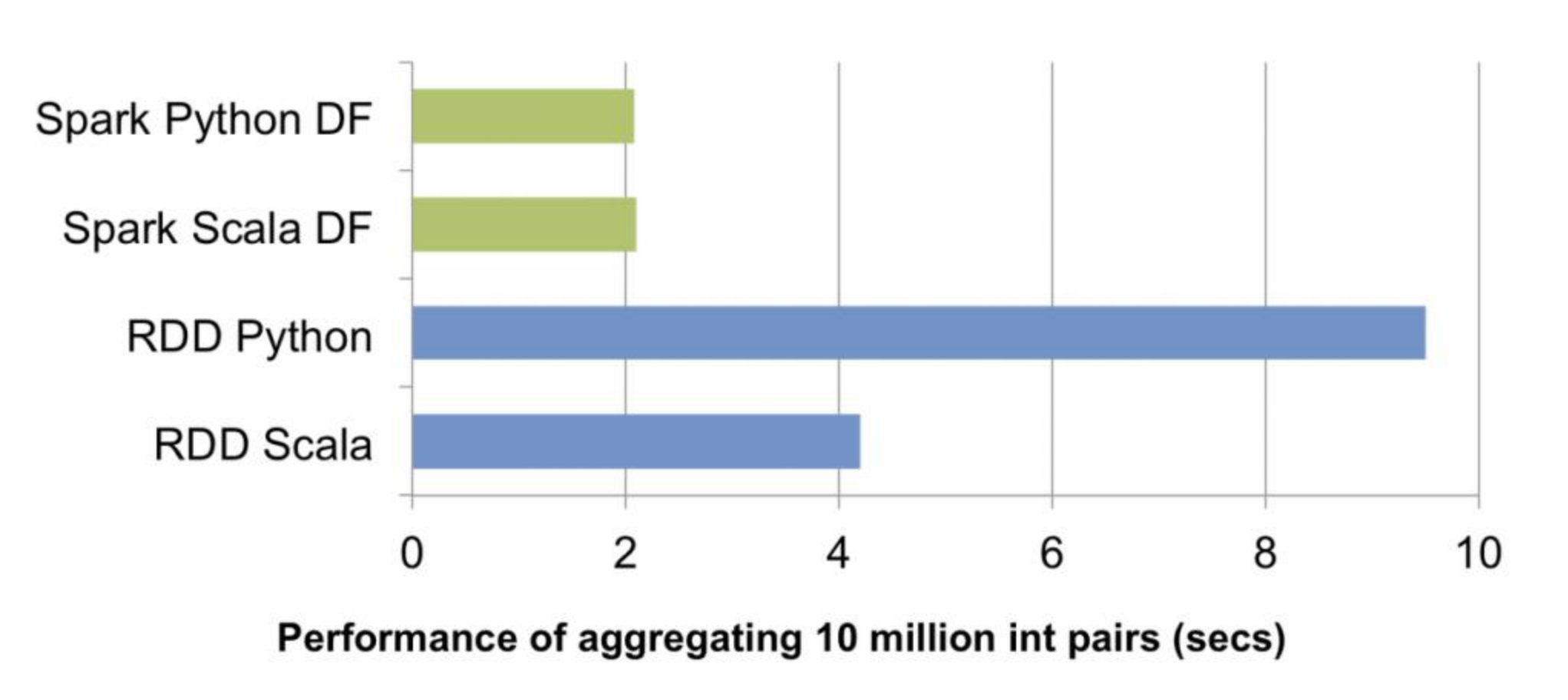
DataFrame/Dataset

- Część modułu Spark SQL
- DataFrame od wersji 1.3
- Dataset od wersji 1.6
- Zawierają więcej informacji o strukturze danych i wykonywanych obliczeń
- Optymalizowane wewnętrznie przez Spark SQL niezależnie od używanego języka
- Rozproszona kolekcja danych
- Dataset -> Scala i Java
- DataFrame -> Scala, Java, Python i R

Dataset

- Silnie typowany zbiór obiektów specyficznych dla domeny, które można przekształcać równolegle przy użyciu operacji funkcyjnych lub relacyjnych
- Dostępne są operacje podobne jak przy RDD: akcje i transformacje
- Dostępne są również operacje analogiczne do tych występujących w „klasycznych” ramkach danych np.:
  - dane("col1") + 10
  - people.filter("age > 30").join(department, people("deptId") === department("id")).groupBy(department("name"), people("gender")).agg(avg(people("salary")), max(people("age")))
- Rozszerzenie DataFrame, łączy cechy DF i RDD

DataFrame – wydajność



DataFrame – zwięzły kod

Średnia wieku dla departamentów:

dept	age
A	34
B	27
B	40
…	…
A	42

DataFrame

```
dataDF.groupBy("dept").avg("age").show()
```

RDD

```
dataRDD.map(lambda line: line.split(",")).map(lambda line: (line[0], (int(line[1]), 1))).reduceByKey(lambda x,y: (x[0] + y[0], x[1] + y[1])).map(lambda line: (line[0], line[1][0]/line[1][1]))
```

DataFrame

- Tabelaryczna abstrakcja danych
- Służy do pracy z ustrukturyzowanymi i semi-ustrukturyzowanymi danymi
- Kolekcja wierszy
- Posiada następujące właściwości RDD: niemutowalność, in-memory, odporność na uszkodzenia, rozproszenie, przetwarzanie równoległe
- Do tych właściwości dodaje strukturę (schemat) danych (schemat można zobaczyć używając metody *printSchema*)
- Rozproszona kolekcja tabelarycznych danych zorganizowanych w wiersze i nazwane kolumny

DataFrame

- Podobnie jak w tabelach relacyjnych baz danych dostępne są tu operacje takie jak: select, filter, intersect, join, groupBy, orderBy, sort, agg, union
- Metoda *registerTempTable* pozwala na zapisanie DataFramu jako tabeli co umożliwia operowanie zawartymi w niej danymi za pomocą selectów

```
>>> df.registerTempTable("dummy")
>>> df2 = spark.sql("select * from dummy")
>>> sorted(df.collect()) == sorted(df2.collect())
True
>>> spark.catalog.dropTempView("dummy")
```

DataFrame - tworzenie

SparkSession pozwala na tworzenie DataFrame:

- Na podstawie istniejących RDD
- Z tabel w Hive
- Źródeł danych Sparka:
  - json
  - parquet
  - jdbc
  - orc
  - libsvm
  - csv
  - text