Sviluppo di sistemi scalabili con Apache Spark

Alessandro Natilla - 22/10/2016

Outline

- Big Data
- Cosa è Apache Spark
- Storia di Spark
- Spark vs MapReduce
- Componenti di Apache Spark
- Foundations: RDD e operazioni
- Modello di esecuzione
- Esempi
- Deploying
- Riferimenti

Big Data

- Velocity
- Volumes
- Variety
- Value

Fonti

- Social Media
- GPS data
- IoT
- Sensors Networks

Cosa è Apache Spark

- Framework per massive parallel computing
- Basato su Direct Acyclic Graph (DAG) computing engine
- in-memory computation
  - Hadoop MapReduce svolge operazioni su disco
- Apache Project (spark.apache.org)

Storia

- Progetto nato presso l'Università di Berkeley nel 2009
- Progetto Apache dal 2013
- Progetto top-level dal 2014
- I creatori hanno fondati databricks.com
- Giunto alla versione 2.0.1 (last stable)

Spark vs Hadoop Mapreduce

- Graysort benchmark, http://sortbenchmark.org/
- Hadoop - 72 minutes / 2100 nodes / datacentre
- Spark - 23 minutes / 206 nodes / AWS

Esempio classico: Word Count on Hadoop

Obiettivo

Contare il numero di occorrenze di ciascuna pa
in un testo

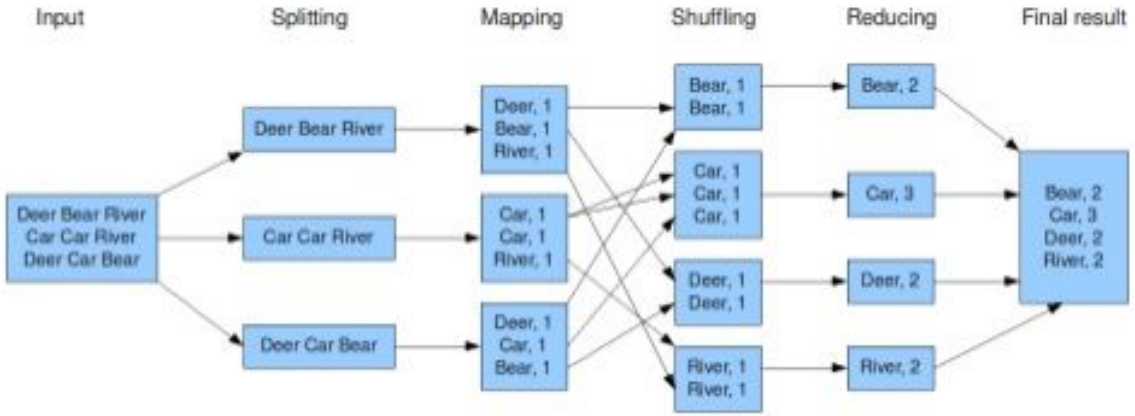
Logica

Per ogni parola, associarvi il valore intero 1.

Si ottiene una lista di coppie (parola, 1).

Aggregazione delle coppie in base alla parola chiave utilizzando una funzione associativa (somma).

Si ottiene una lista di coppie dove il primo elemento corrisponde alla parola, il secondo elemento coincide con il numero totale di occorrenze nel testo



Esempio classico: Word Count su Spark (Python API)

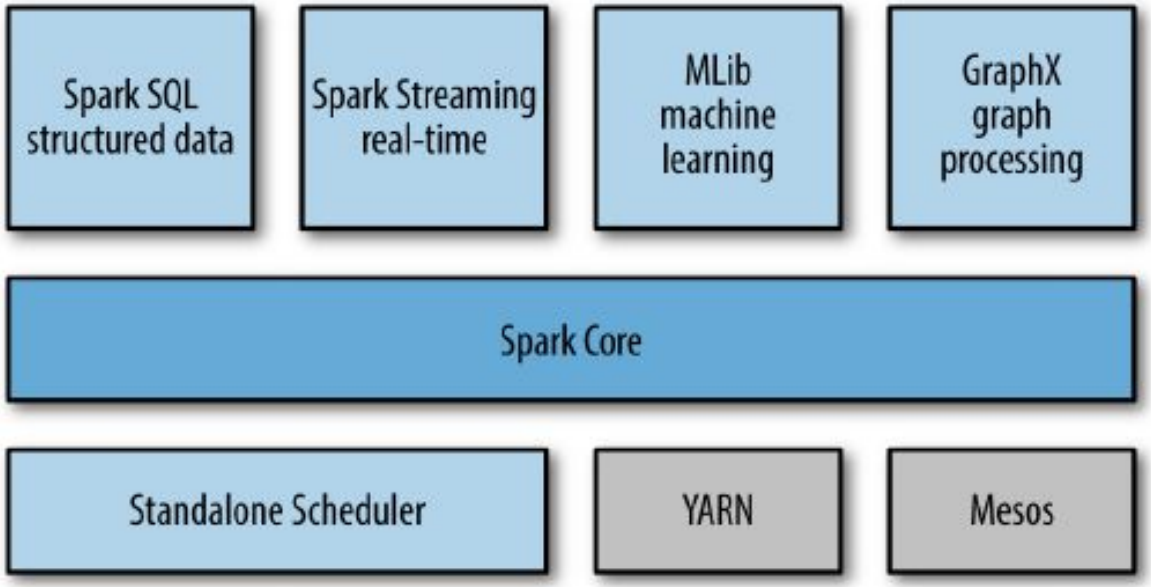
```
from pyspark import SparkContext

logFile = "hdfs:///input"
sc = SparkContext("spark://spark-m:7077", "WordCount")
textFile = sc.textFile(logFile)

wordCounts = textFile.flatMap(lambda line: line.split())
                     .map(lambda word: (word, 1))
                     .reduceByKey(lambda a, b: a+b)

wordCounts.saveAsTextFile("hdfs:///output")
```

Componenti di Spark



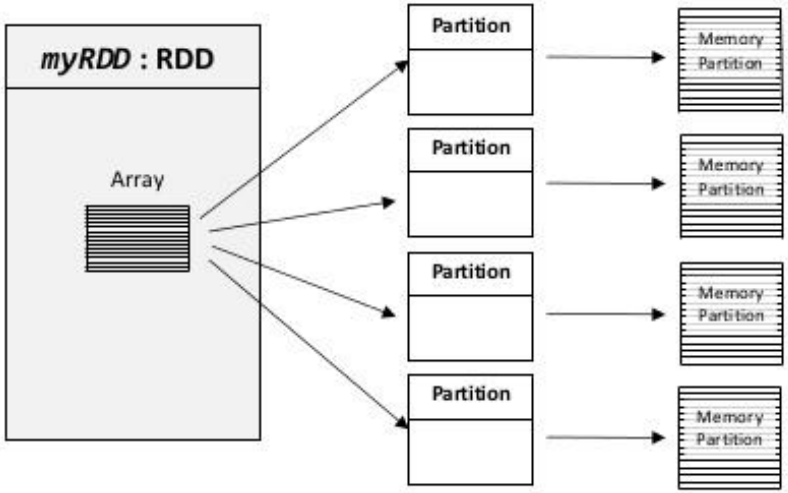
Applicazioni concrete

- Analytics (batch / streaming)
- Machine Learning
- ETL (Extract - Transform - Load)

- Datawarehousing

Fondamenti: Resilient Distributed Datasets (RDD)

- RDD = Resilient Distributed Dataset
- Collezione di dati immutabile
- Fault-tolerant
- Parallel



Some RDD Characteristics

- Hold references to Partition objects
- Each Partition object references a subset of your data
- Partitions are assigned to nodes on your cluster
- Each partition/split will be in RAM (by default)

Fondamenti: operazioni

- Trasformazioni
- Azioni
- Le trasformazioni sono operazioni *lazy*
- Le trasformazioni vengono compiute dalle azioni

PRO
No problemi di concorrenza in contesti di elaborazione distribuiti
Tutti i nodi lavorano su partizioni differenti dei dati

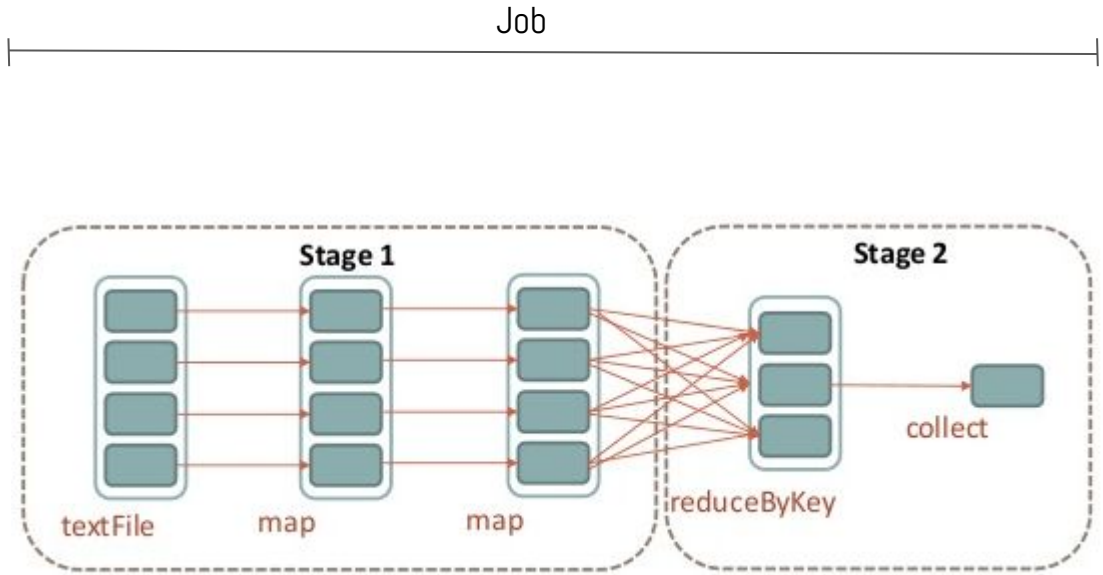
RDDs – Trasformazioni vs Azioni

- map(): trasformazione
- filter(): filtraggio
- flatMap(): trasformazione dati
- sample(): campionamento
- …

- reduce(): applicazione operazione associativa
- count(): conteggio
- saveAsTextFile()
- …

Modello di esecuzione di Spark

- I jobs sono cittadini di prima classe
- L'invocazione di una azione causa l'esecuzione di un job per evadere una richiesta
- Spark esamina il grafo degli RDD, producendo un piano di esecuzione che tenga conto delle risorse disponibili



Streaming

- Micro-batches (DStreams of RDDs)
- Disponibile per tutte le componenti (MLLib, GraphX, Dataframes, Datasets)
- Fault-tolerant
- Connettori per TCP Sockets, Kafka, Flume, Kinesis, ZeroMQ, ...

Spark SQL

- Libreria di astrazione dati
- Idea presa in prestito da Python/R
- Supporto per JSON, Cassandra, HBase, Hive, SQL databases, etc.
- Sintassi più semplice rispetto agli RDD

- Datasets vs Dataframes
  - type-safe, interfaccia object-oriented programming
  - utilizzano l'ottomizzare nativo
  - elaborazione dei dati in-memory

Dataframe vs Dataset: contare numero di persone con età > 35

Dataframe

```
val sqlContext = new org.apache.spark.sql.SQLContext(new SparkContext())
val df = sqlContext.read.json("people.json")

df.show()
df.filter(df("age") >= 35).show()

df.groupBy("age").count().show()
```

Datasets

```
case class Person(name: String, age: Long)

val people = sqlContext.read.json("/people.json").as[Person]

people.filter(_.age >= 35).show()

people.groupBy(_.age).count().show
```

MLLib / ML

- Componente di Machine Learning
- Include implementazioni per algoritmi quali for NaiveBayes, logistic regression, k-means clustering, ALS, word2vec, random forests, etc.
- ML introduce il concetto di pipelines
- Operazioni su matrici (dense / sparse), fattorizzazioni matriciali, etc.
- Basic statistics

http://spark.apache.org/docs/latest/mllib-guide.html

GraphX

- Contiene algoritmi su grafi
- Operazioni su vertici e archi
- Include l'algoritmo PageRank
- Combinabile with Streaming/SparkSQL/MLLib/ML

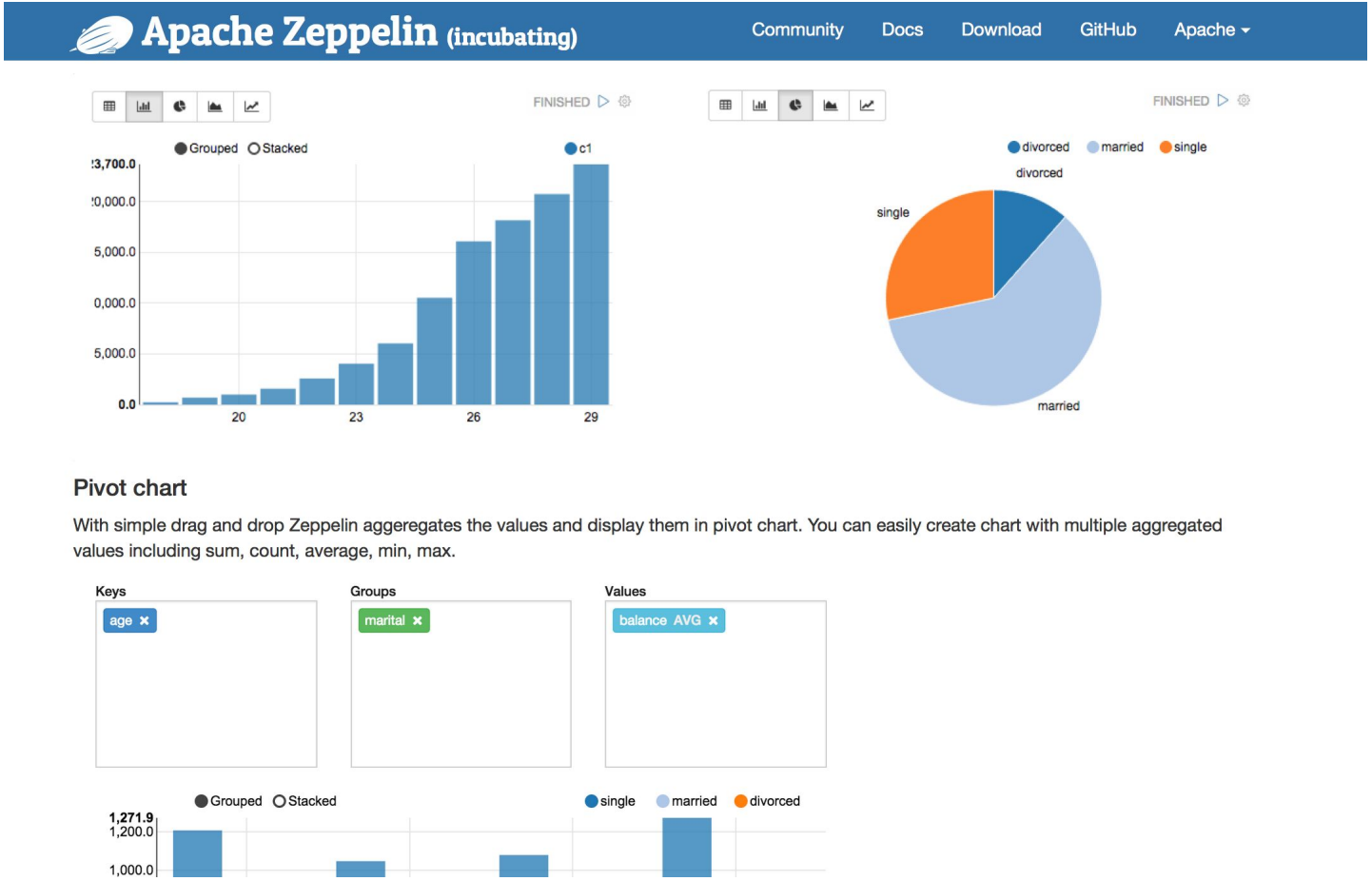
Deploying Spark

- Standalone
- YARN (Hadoop ecosystem)
- Apache Mesos

Using Spark

- Traditional (write code, submit to cluster)
- REPL (write code interactively, backed by cluster)
- Interactive Notebooks (iPython/Zeppelin)

Interactive Notebooks



References

- spark.apache.org
- databricks.com
- zeppelin.incubator.apache.org
- mammothdata.com/white-papers/spark-a-modern-tool-for-big-data-applications