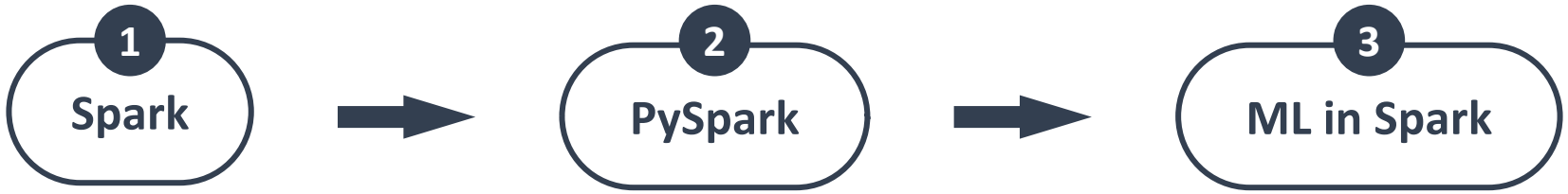
Spark :: PySpark

Dayoung Kang, Jaehyeon Kim

Table of Contents



Reference

- Spark: Cluster Computing with Working Sets
- PySpark: High-performance data processing without learning Scala
- MLlib: Machine Learning in Apache Spark
- Scalable machine-learning algorithms for big data analytics: a comprehensive review

Emergence of Spark

Large-scale data processing

→ **Map Reduce**

For acyclic data flow model

Map Reduce

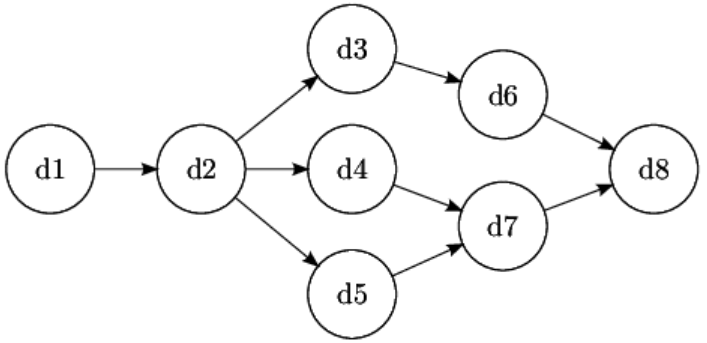
Step1. Map

- Split input data to number of slices
- Apply specific function to each to generate intermediate results

Step2. Reduce

- Combine the intermediate results to make the final result.

Acyclic data flow model



- By representing data processing tasks as a graph
- Data can only flow in one direction
  ---> preventing loops

Emergence of Spark

Not suitable for applications

"reuse datasets through **parallel operations**"

=

repeatedly use the same data set in multiple parallel operations

⇓

Spark

- one variant of Map reduce
- introduces the concept of a Resilient Distributed Dataset (RDD)

Resilient Distributed Dataset(RDD)

- a collection of read-only objects that are partitioned and stored on multiple computers
- if one of the partitions is lost, it can be rebuilt

Resilient Distributed Dataset(RDD)

in Spark, each RDD is represented by a Scala object

Methods

1. **Generated from the file** – from Shared file systems
2. **Parallelizing Scala collections** – split number of slices ---> nodes
3. **Transformation RDD** – convert type of component
4. **Changing persistence** – Cache & Save

Resilient Distributed Dataset(RDD)

RDD performs a variety of parallel tasks

Works

1. **Reduce** – combine operation
2. **Collect** – send to driver program
3. **Foreach** – function pass

Spark

Spark supports two types of shared variables

Variables

1. **broadcast** – cache values in memory
2. **accumulator** – counter & sum

Spark vs Hadoop : Logistic regression

when comparing the performance of logistic regression,
Spark ran up to 10 times faster than Hadoop.

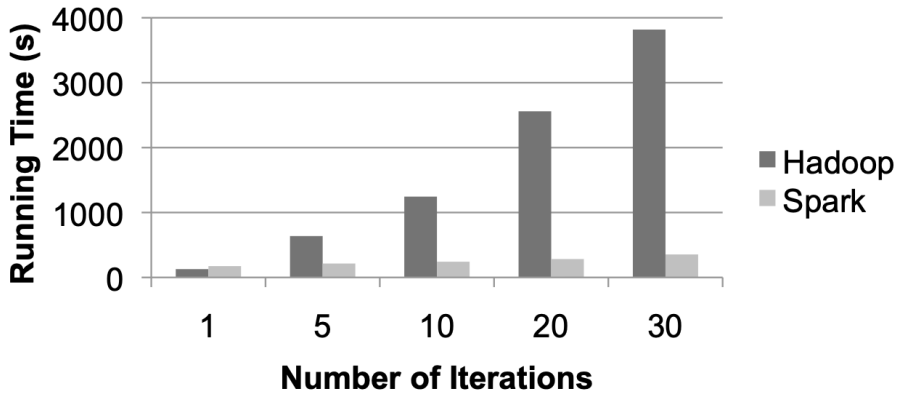


Figure 2: Logistic regression performance in Hadoop and Spark.

- reuse cached data
- shared variables
- parallel processing
- recovery function

About PySpark

PySpark

The python API for Apache Spark

PySpark or pandas

PySpark

- Large datasets

pandas

- Easy to analyze data in local
- Limitations for very large datasets

PySpark with pandas

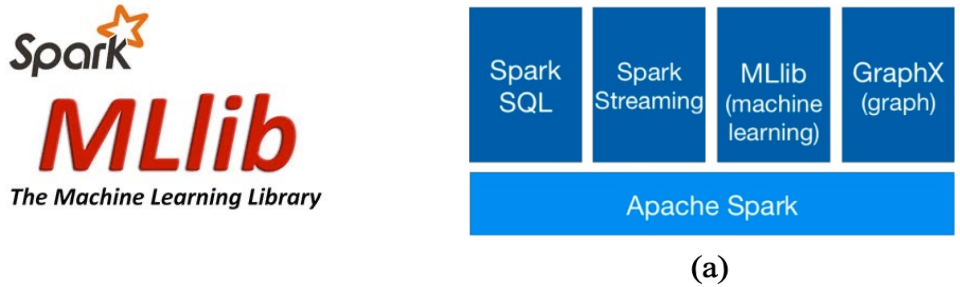
PySpark to aggregate and transform data and pull the results into a DataFrame in pandas for further analysis in local environment

"an effective approach to processing and analyzing large amounts of data"

- The key point is that PySpark will not necessarily replace the pandas

MLlib: Machine Learning in Apache Spark

Spark's distributed machine learing library, and the largest such library



- Targets large-scale learning settings that benefit from data-parallelism or model-parallelism

MLlib: Algorithms

MLlib provides fast, distributed implementations of common learning algorithms.

Learning

1. **Classification**
2. **Regression**
3. **Clustering**
4. **Recommendation**

MLlib: Classification Algorithms

Algorithms

1. Logistic Regression with SGD
2. Logistic Regression with LBFGS
3. SVM with SGD
4. Naïve Bayes
5. Decision Tree
6. Gradient Boosted Trees

MLlib: Regression Algorithms

Algorithms

1. Linear Regression with SGD
2. Ridge Regression with SGD
3. Lasso with SGD
4. Elastic Net Regression
5. Isotonic Regression
6. Gradient Boosted Trees
7. Random Forest

MLlib: Clustering Algorithms

Algorithms

1. K-means
2. Streaming K-Means
3. Gaussian Mixture
4. LDA
5. Power Iteration Clustering

MLlib: Recommendation Algorithms

Algorithms

1. **ALS**

MLlib: Algorithmic Optimization

MLlib include many optimizations to support efficient distributed learning land prediction

- JVM GC
- Reduce Communication Costs
- Parallelize Learning within Trees