Big data architectures for Machine Learning and Data Mining

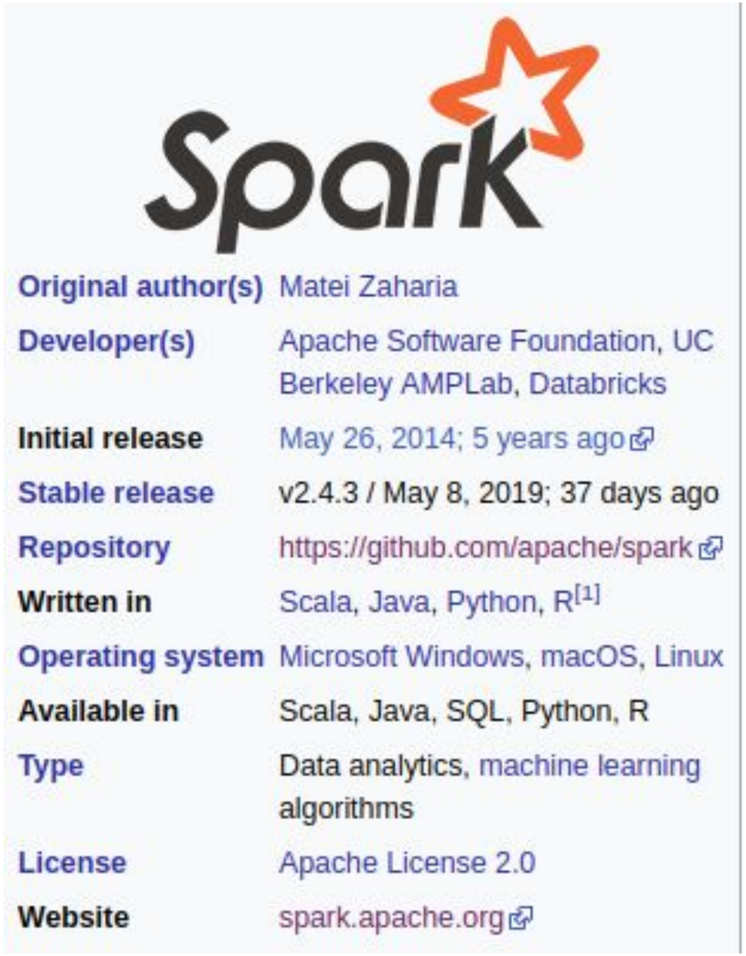
Apache Spark

Sagar Nagaraj Simha
Matrikel Number - 120797, sagar.nagaraj.simha@uni-weimar.de
Master’s Computer Science for Digital Media - SS2019
Bauhaus University Weimar

What is Apache Spark?

A unified computing engine and a set of libraries for parallel data processing on computer clusters

Origin



Original author(s)	Matei Zaharia
Developer(s)	Apache Software Foundation, UC Berkeley AMPLab, Databricks
Initial release	May 26, 2014; 5 years ago
Stable release	v2.4.3 / May 8, 2019; 37 days ago
Repository	https://github.com/apache/spark
Written in	Scala, Java, Python, R[1]
Operating system	Microsoft Windows, macOS, Linux
Available in	Scala, Java, SQL, Python, R
Type	Data analytics, machine learning algorithms
License	Apache License 2.0
Website	spark.apache.org

Why develop Apache Spark?

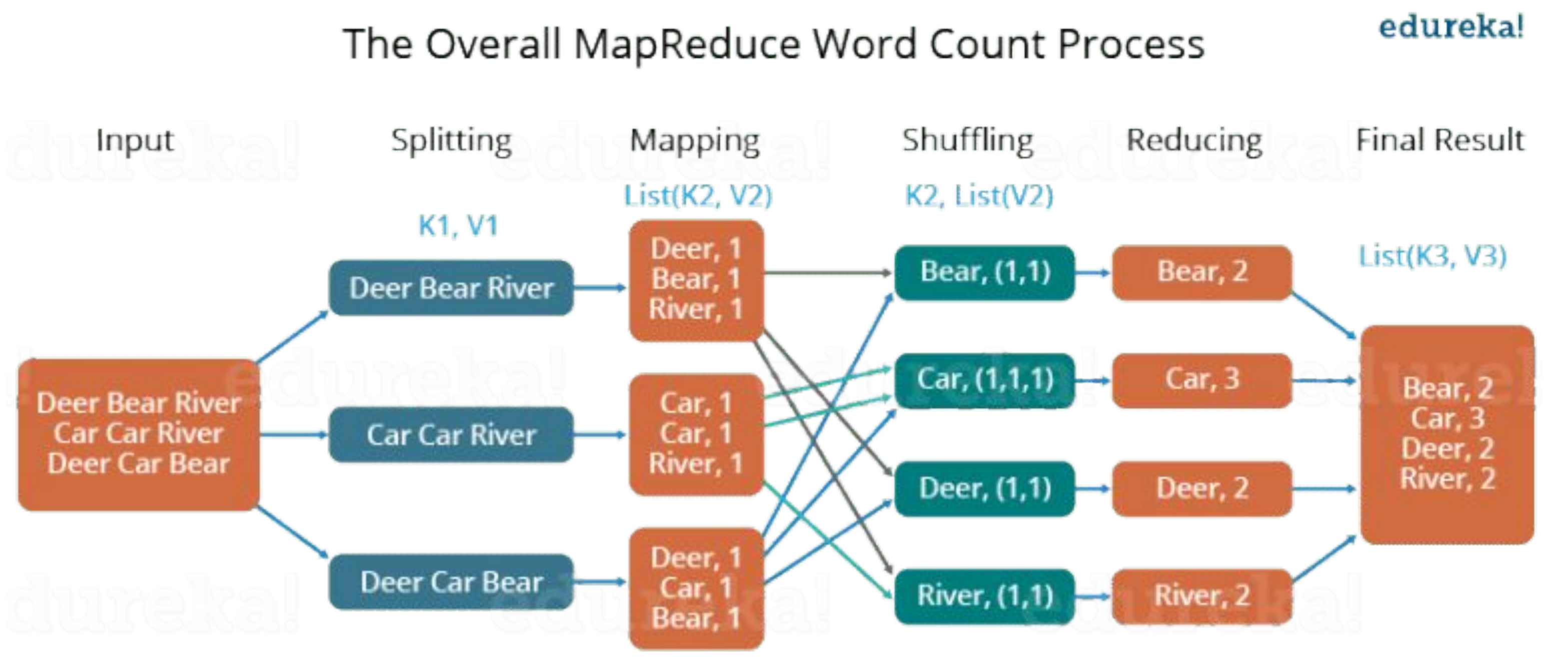
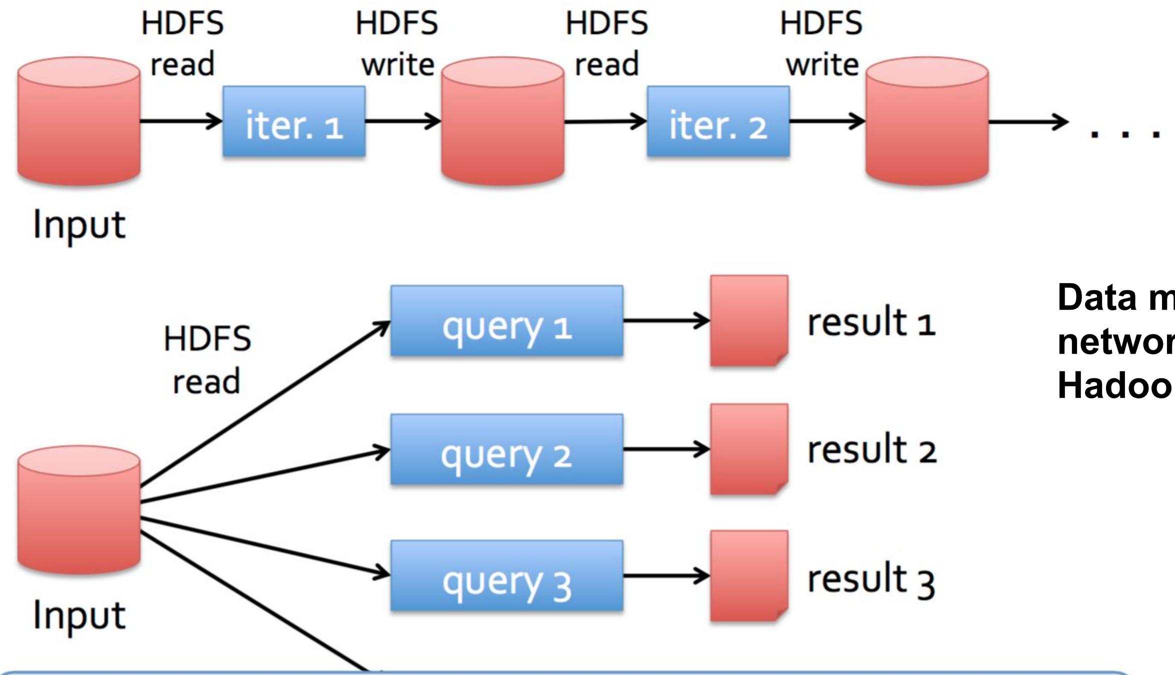


Image courtesy [7]

Apache Hadoop-MapReduce



Data moving over network in Hadoop is slow

Slow due to replication and disk I/O, but necessary for fault tolerance

Apache Hadoop-MapReduce

MapReduce is inefficient for *multi-pass* and low latency requirement applications

- *Iterative algorithms*
- *Interactive data mining*
- *Streaming applications*

Hadoop included the Hadoop file system and MapReduce.

- Hard to run one of the systems without the other
- Incompatible with other cloud storage services

Challenges

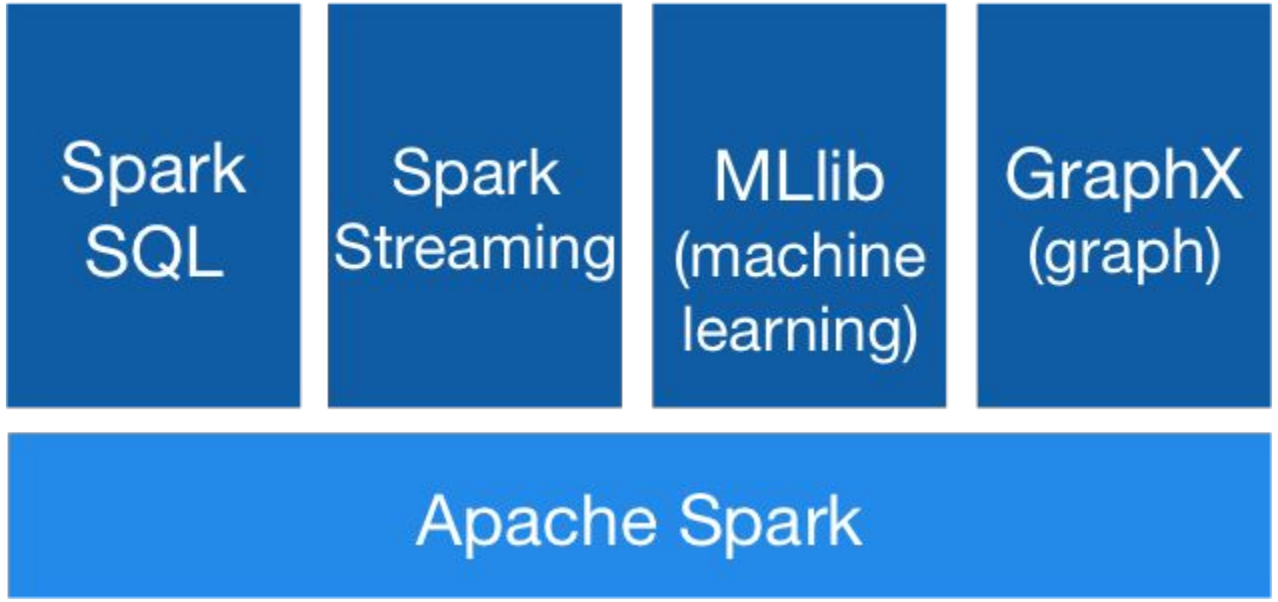
- Incorporate the previously mentioned features
- Have a distributed memory abstraction
- Fault tolerant and efficient

Apache Spark’s Philosophy

“A new engine and programming model for data analytics”

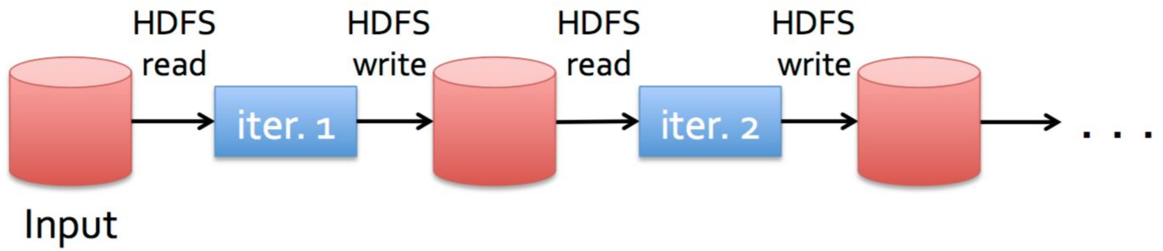
1. Unified
    a. Combines batch, interactive and streaming (incremental processing)
2. Computing Engine
    a. Supports HDFS, Cassandra, Kafka etc and focuses on Computing - Moving data is expensive.
3. Libraries
    a. Spark SQL - SQL and structured data
    b. MLlib - Machine learning
    c. GraphX - Graph analytics
    d. Spark Streaming and the newer Structured Streaming - Stream processing
    e. Many others on https://spark-packages.org/ (Ex: Connectors to cloud storages, ML algorithms etc)
4. Resilient Distributed Datasets at the Lowest level for efficient fault tolerance

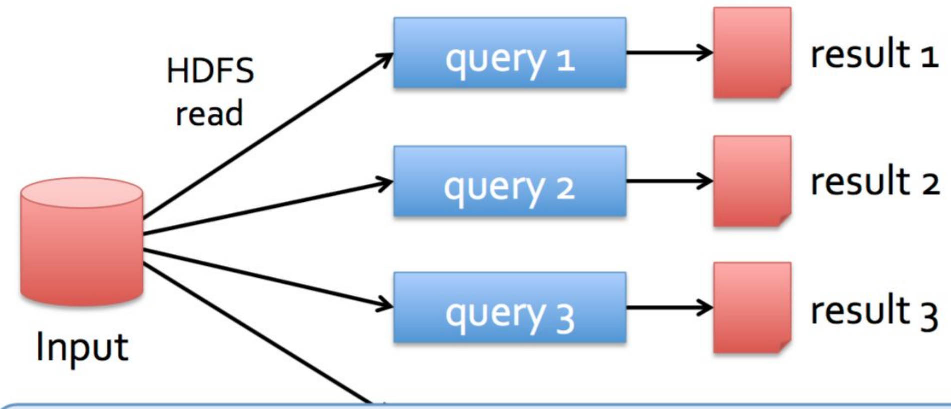
The Stack


Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

Resilient Distributed Datasets (RDDs)

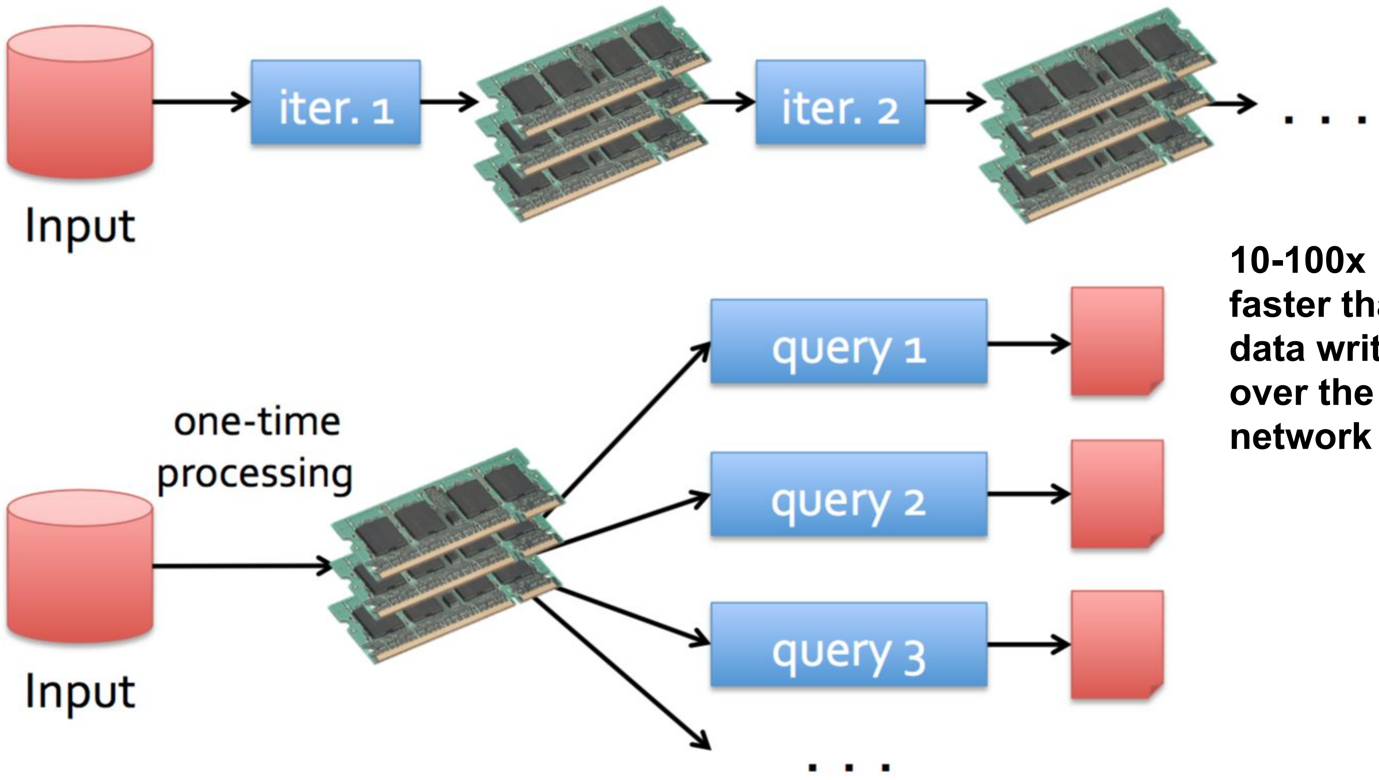
Apache Hadoop-MapReduce





Slow due to replication and disk I/O, but necessary for fault tolerance

Spark - In Memory Data Sharing using RDDs

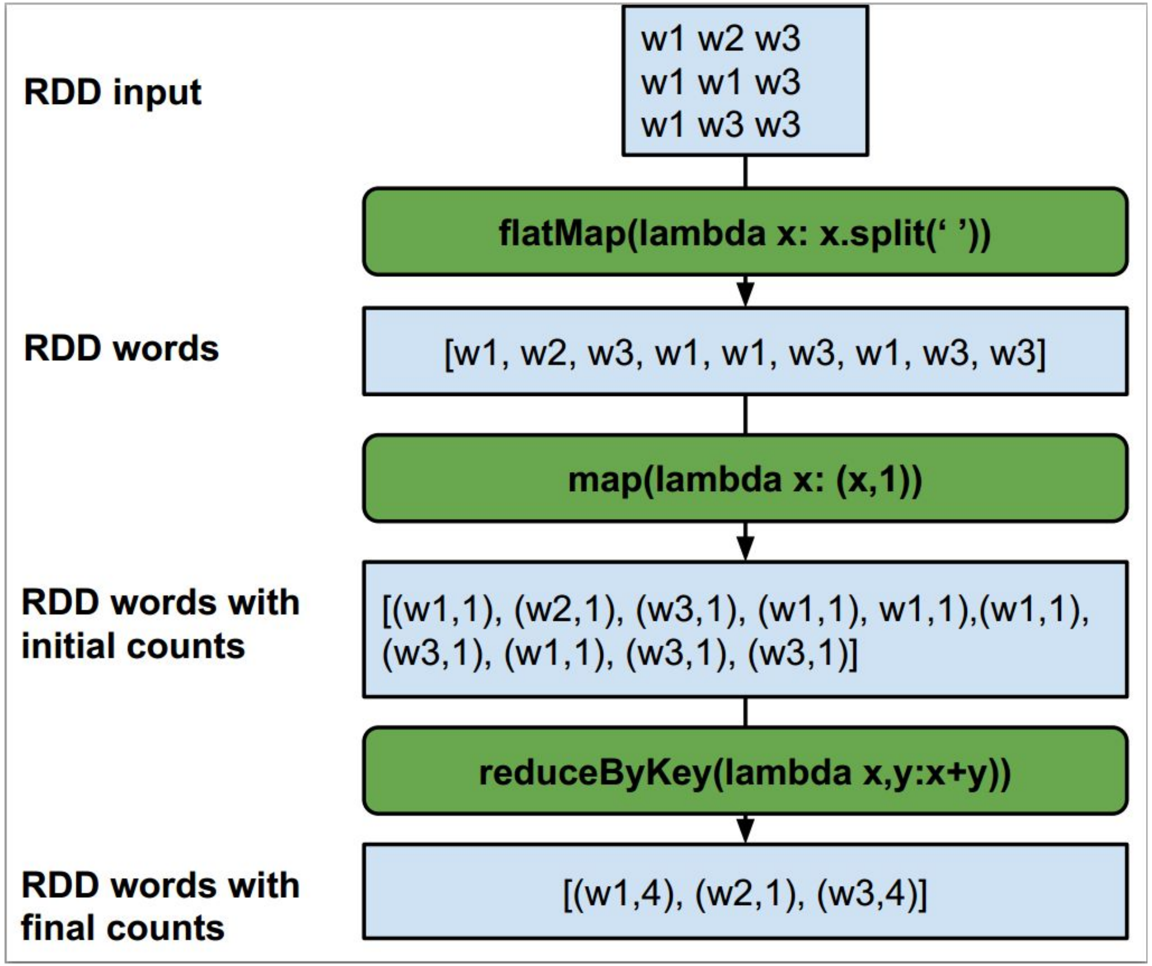


10-100x faster than data write over the network

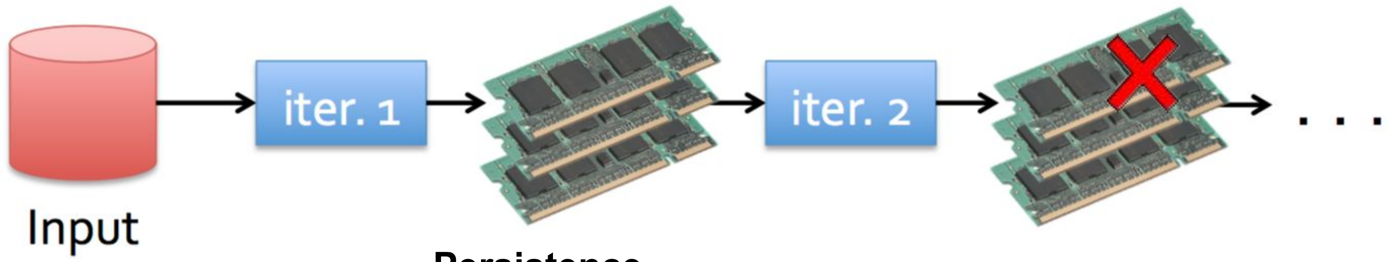
Fault Recovery

RDDs track the graph of transformations that built them (their lineage - map, filter, join etc…) to rebuild lost data

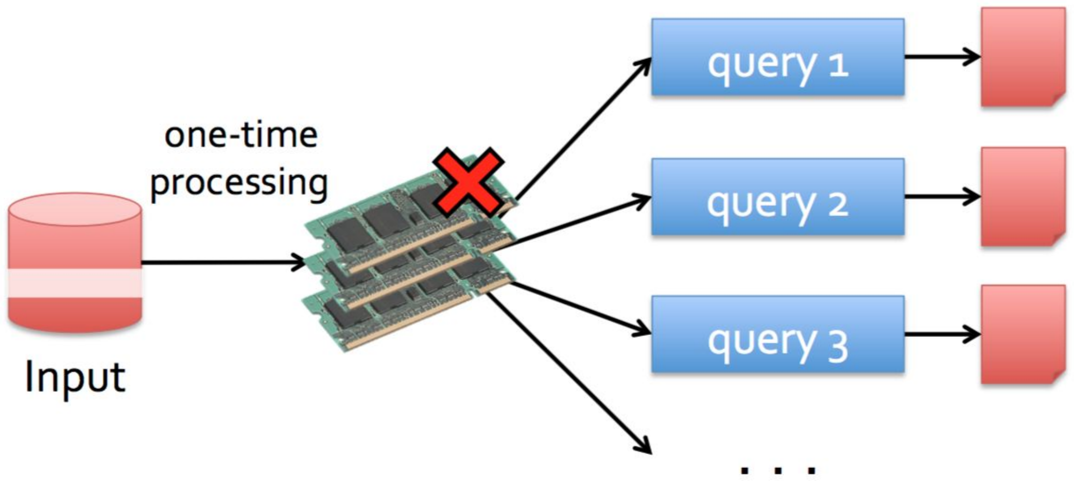
- Each row in the RDD input is a partition stored in a different machine in the cluster
- The transformations (in green box) are applied to each partition
- A batch data (blue box) may be recovered by running the corresponding transformation that produced it


RDD input
w1 w2 w3
w1 w1 w3
w1 w3 w3
flatMap(lambda x: x.split(‘ ’))
RDD words
[w1, w2, w3, w1, w1, w3, w1, w3, w3]
map(lambda x: (x,1))
RDD words with initial counts
[(w1,1), (w2,1), (w3,1), (w1,1), w1,1),(w1,1), (w3,1), (w1,1), (w3,1), (w3,1)]
reduceByKey(lambda x,y:x+y))
RDD words with final counts
[(w1,4), (w2,1), (w3,4)]

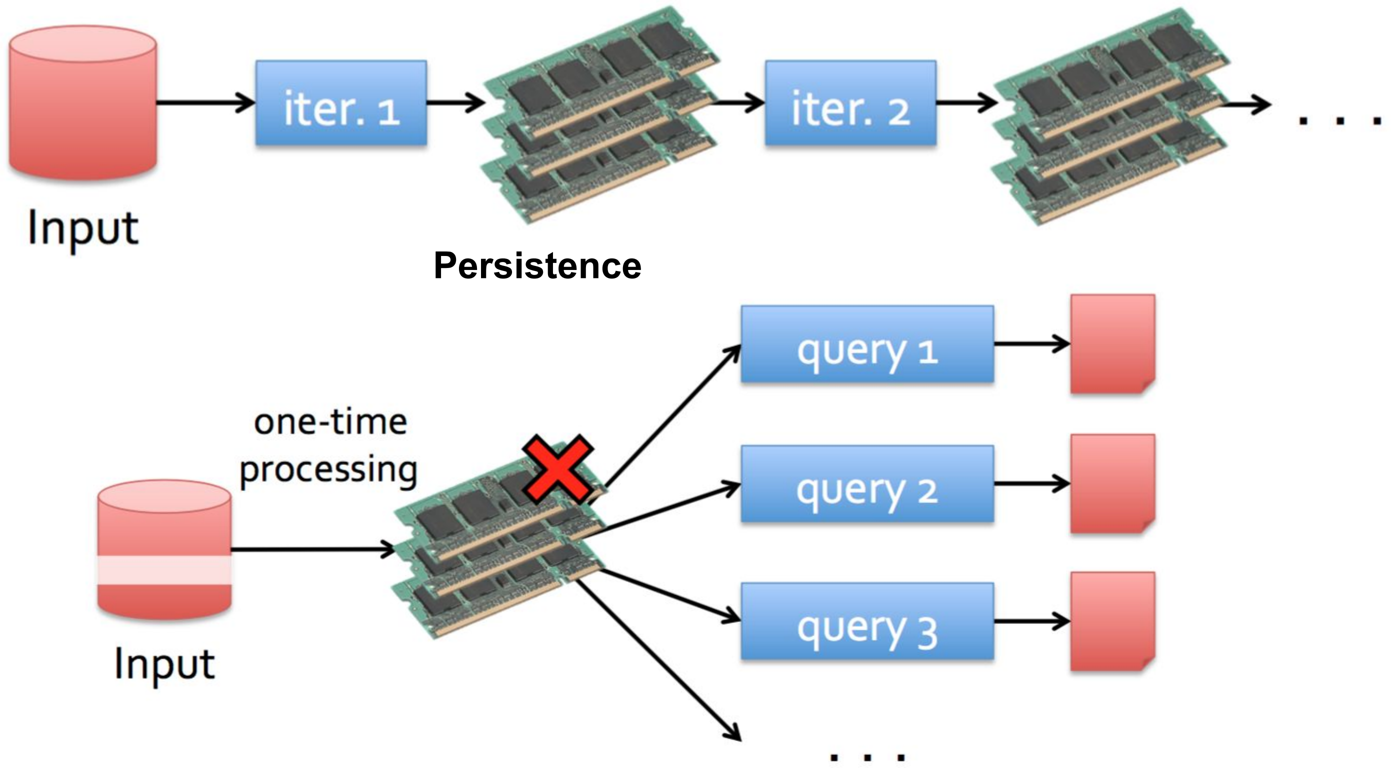
RDD Recovery - Fault Tolerance



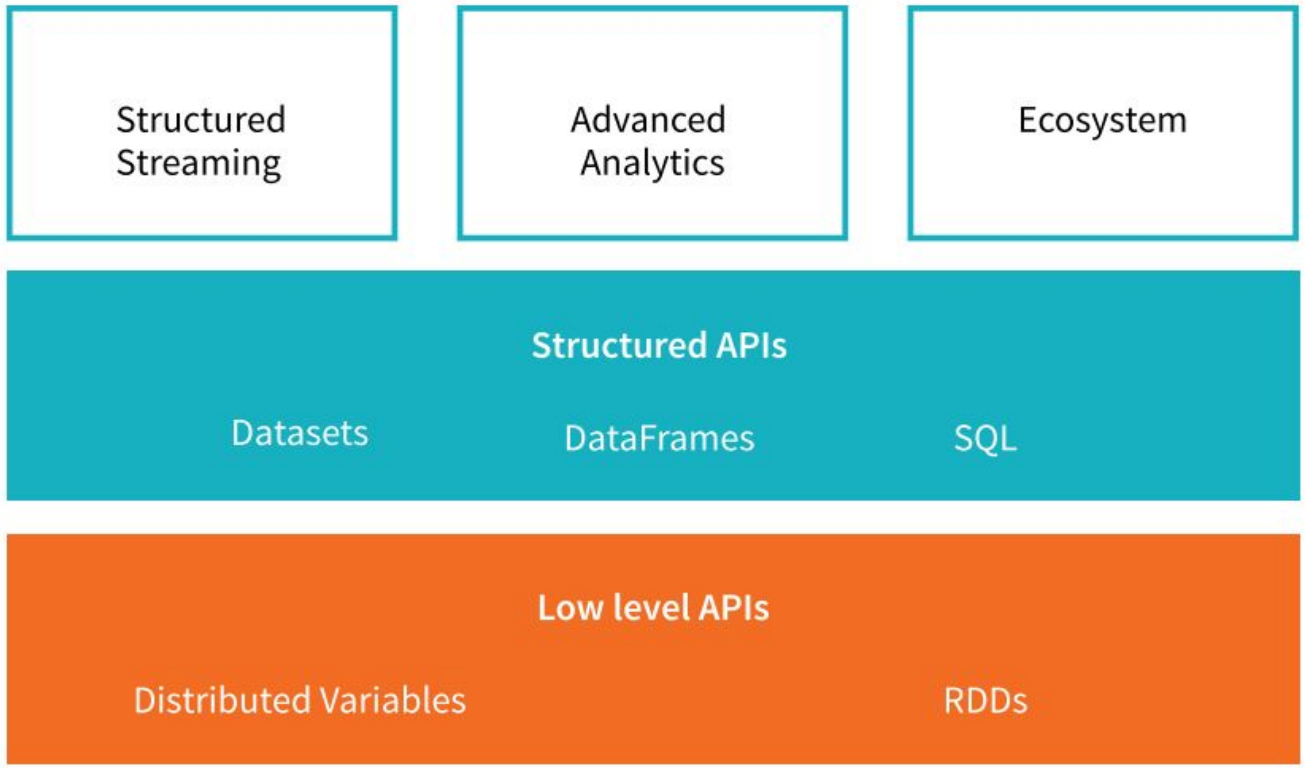
Persistence



RDD Recovery - Fault Tolerance

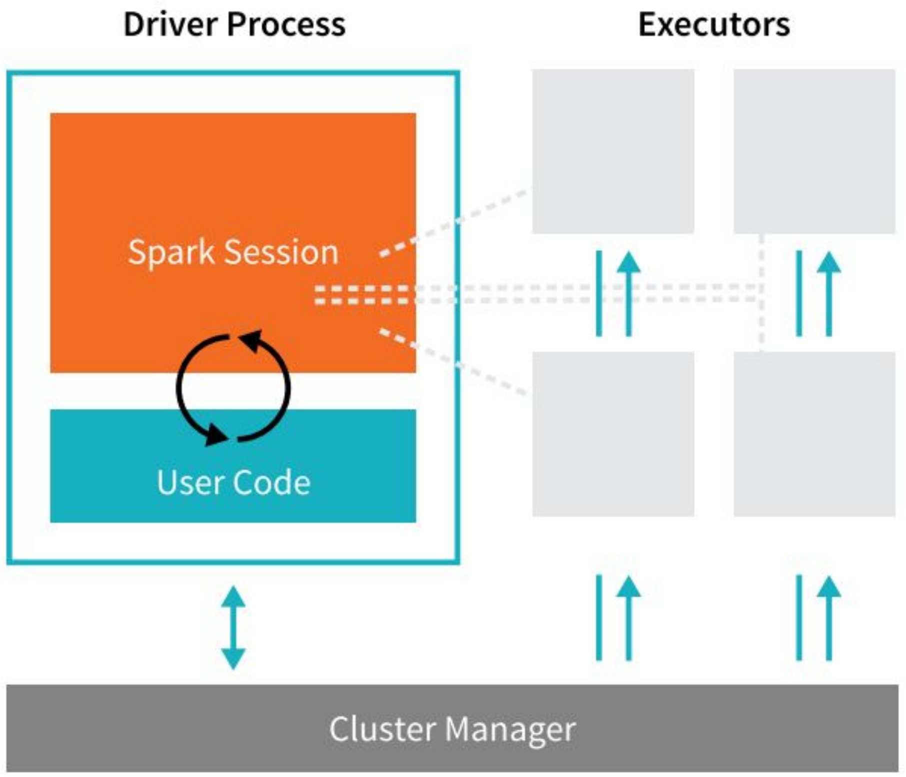


What Spark offers

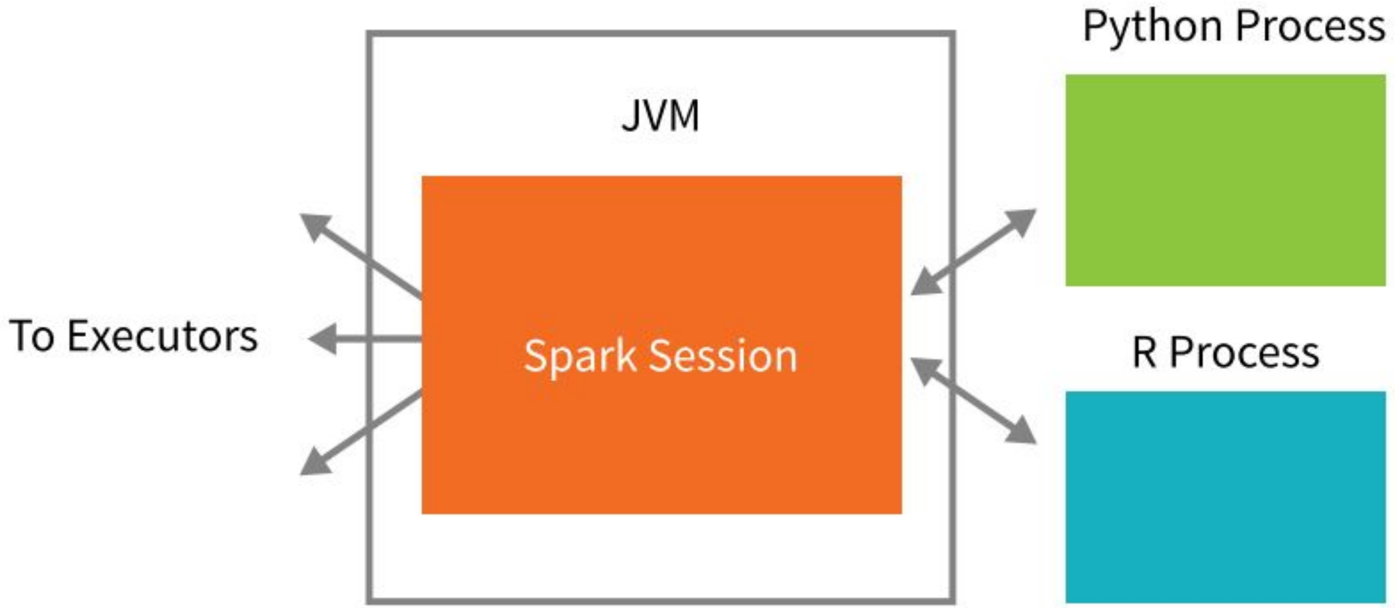


Architecture and workflow

Apache Spark architecture and workflow



Scala, Python & R hooks to launch a Spark Session



Spark Example: Log Mining

Load error messages from a log into memory and run interactive queries

lines = spark.textFile("hdfs://...")

base RDD

errors = lines.filter(startsWith("ERROR"))

transformation

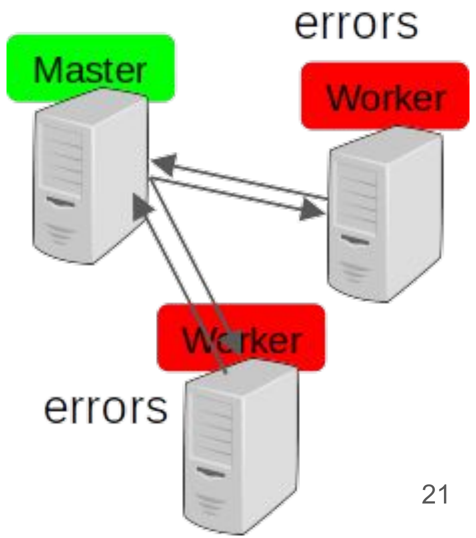
errors.persist()

errors.filter("404" in errors).count()

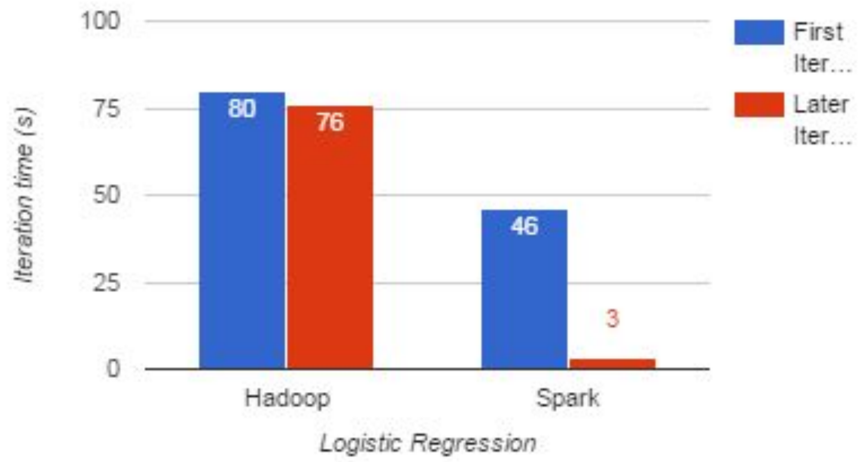
action!

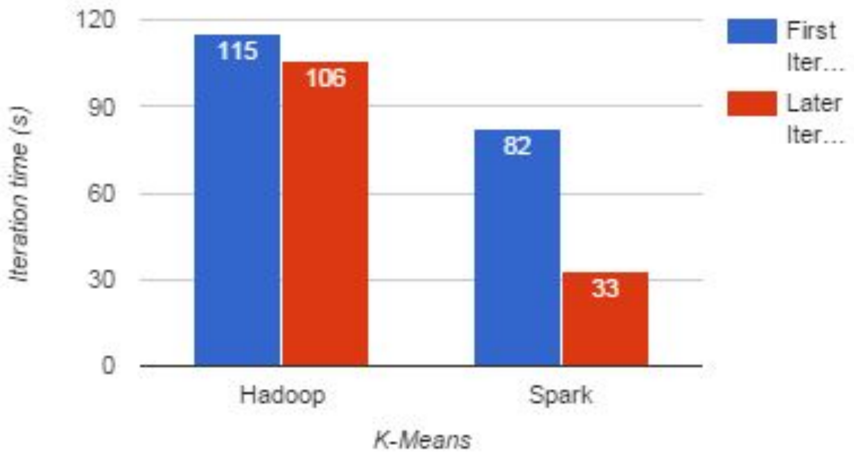
errors.filter(“405” in errors).count()

Result: full-text search on 1TB data on 100 machines in 5-7sec vs. 170sec with on-disk data!



Performance - 10 iterations on 100GB data using 25-100 machines





Features/Advantages

1. Open-source
2. Speed - 100x faster than Hadoop for large scale data processing
3. Automatic fault tolerance
4. Unified Engine
5. Usability with multiple languages
6. Lazy Evaluation - “predicate pushdown”
7. Compatibility with other ecosystems
8. Supports interactive and production applications
9. Easy-to-use APIs for operating on large datasets

Companies/Products that use Spark

- Amazon
- Uber
- Baidu
- eBay Inc.
  - Using Spark core for log transaction aggregation and analytics
- Yandex
- PanTera
  - PanTera is a tool for exploring large datasets. It uses Spark to create XY and geographic scatterplots from millions to billions of datapoints.
- Vistar Media
  - Location technology company enabling brands to reach on-the-go consumers

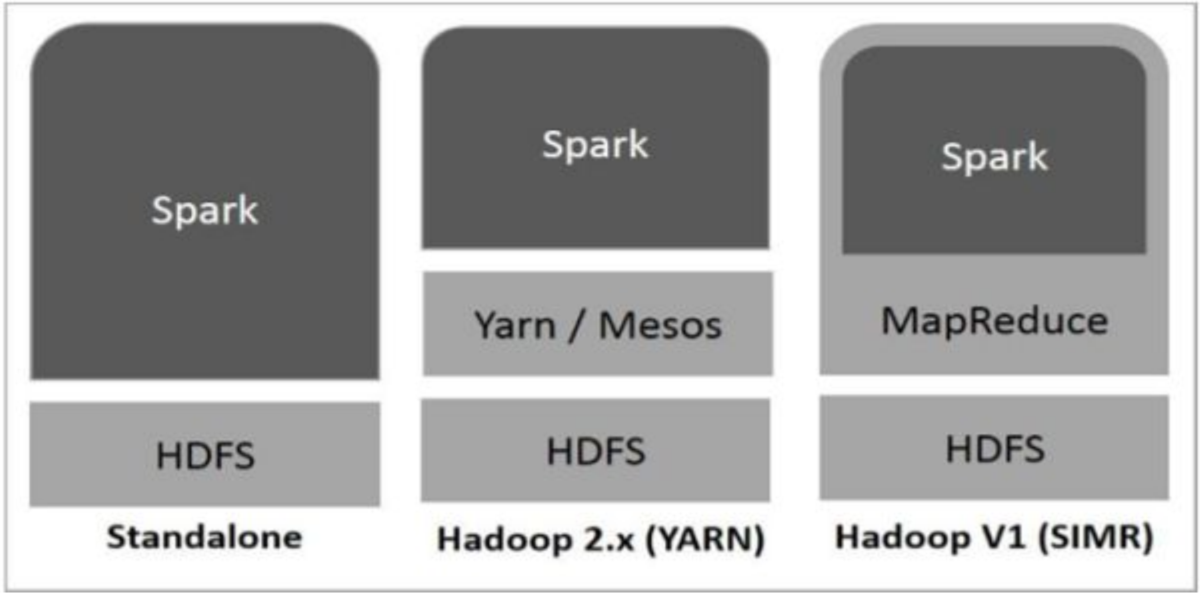
Research institutions that use Spark

- CERN Open Lab
    - From Collision to Discovery: Physics Analysis with Apache Spark
- European Gravitational Observatory
    - Machine Learning for Gravitational Wave signals classification in LIGO and Virgo
- Freeman Lab at HHMI
    - We are using Spark for analyzing and visualizing patterns in large-scale recordings of brain activity in real time
- NASA JPL - Deep Space Network
- Stanford DAWN
    - Research lab on infrastructure for usable machine learning, with multiple research projects that run over or accelerate Apache Spark.

Installation

For the Demo, we will run a Hadoop Compatible Apache Spark on a Vagrant VM machine with a driver(master) node and N(2) worker nodes

Apache Spark with other systems



We will use Apache Spark Standalone Cluster manager for demo

Apache Spark with other systems

Over Hadoop YARN Cluster,

https://www.linode.com/docs/databases/hadoop/install-configure-run-spark-on-top-of-hadoop-yarn-cluster/

https://databricks.com/blog/2014/01/21/spark-and-hadoop.html

http://www.datumly.com/2017/08/apache-spark-2-2-in-a-virtual-machine-simple-getting-started-guide-to-run-spark-on-your-laptop/

https://medium.com/explore-artificial-intelligence/downloading-spark-and-getting-started-with-python-notebooks-jupyter-locally-on-a-single-computer-98a76236f8c1

With Hadoop:

https://www.davidadrian.cc/posts/2017/08/how-to-spark-cluster/

With Jupyter Notebook:

https://opensource.com/article/18/11/pyspark-jupyter-notebook

DEMOS

Demo plan

1. Launching the VM cluster with Spark
2. Spark UI
3. Spark Shell (PySpark)
4. Features of Spark - RDDs, DataFrames, Lazy Evaluation
5. Flight Data Analysis
6. Word Count
7. Production Application - Calculation of Pi

Disadvantages

- No File Management System (Reliant on other storage)
- In-memory computations are expensive
- Spark MLlib has limited algorithms
- Manual Optimization
- Lower latency compared to Apache Flink

Research in Spark

https://spark.apache.org/research.html

Spark Summit - https://databricks.com/sparkaisummit

To deep dive further

- A-Gentle-Introduction-to-Apache-Spark (Databricks)
- Spark - The Definitive Guide - Big data processing made simple (O’Reilly)
- Learning Spark, Lightning-Fast Big Data Analysis (O’Reilly)

Dataset Used for Demo

https://github.com/databricks/Spark-The-Definitive-Guide/tree/master/data

References

[1] https://www.youtube.com/watch?v=dXG4yC8ICEI

[2] https://spark.apache.org/research.html

[3] https://www.whizlabs.com/blog/apache-spark-limitations/

[4] https://spark.apache.org/powered-by.html

[5] https://www.youtube.com/watch?v=L029ZNBG7bk&feature=youtu.be

[6] A Gentle Introduction to Apache Spark - By Databricks

[7] https://www.edureka.co/blog/mapreduce-tutorial/

"After careful consideration of all 437 charts, graphs, and metrics, I've decided to throw up my hands, hit the liquor store, and get snockered. Who's with me?!"

Vielen Dank!