An Introduction to Spark and to its Programming Model

Rosa Filgueira

Introduction to Apache Spark

- Fast, expressive cluster computing system compatible with Apache Hadoop
- It is much faster and much easier than Hadoop MapReduce to use due its rich APIs
- Large community
- Goes far beyond batch applications to support a variety of workloads:
  - including interactive queries, streaming, machine learning, and graph processing

Figure: *Real Time Processing In Spark*

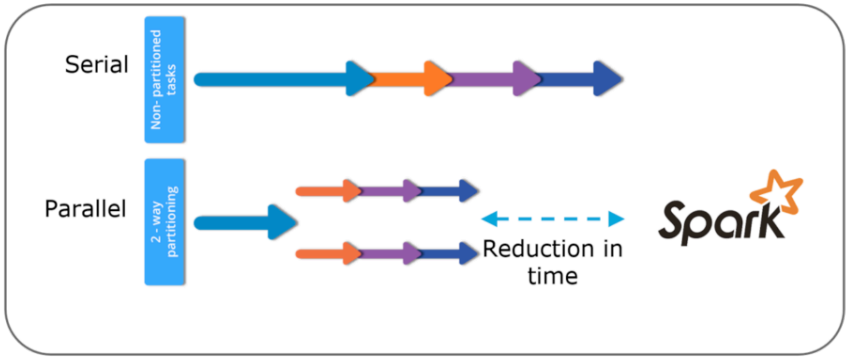
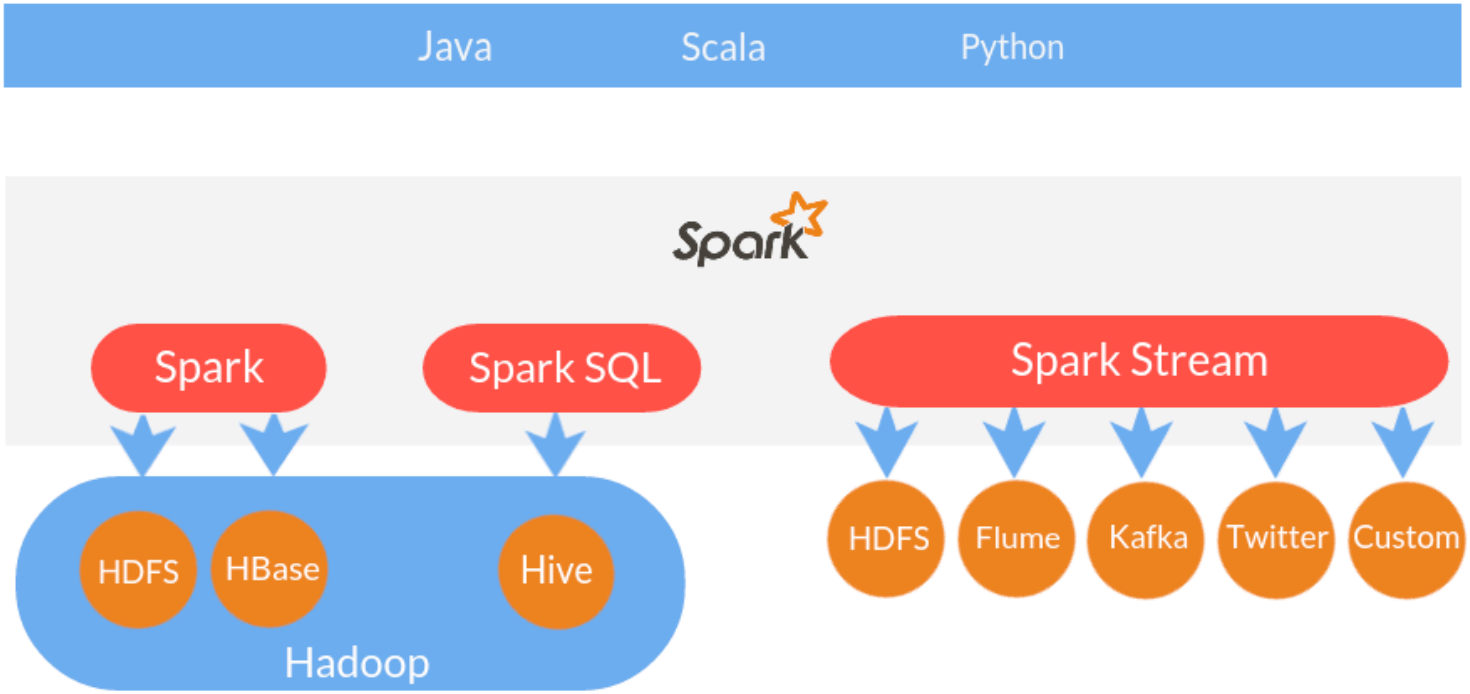


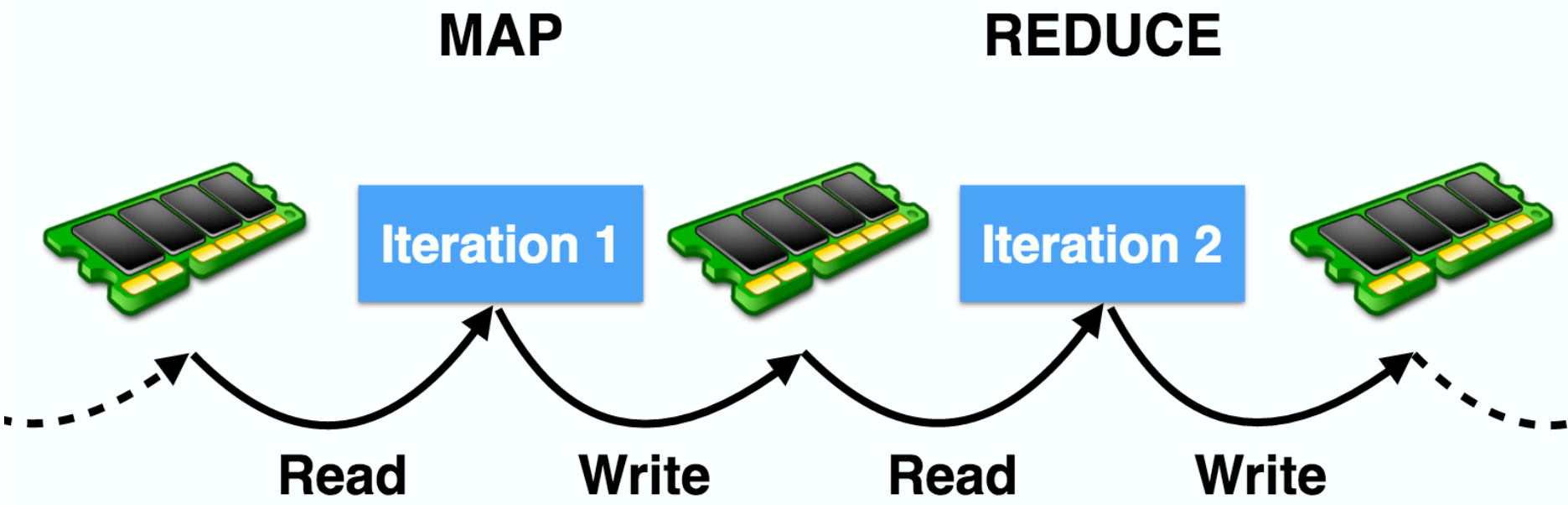
Figure: *Data Parallelism In Spark*

Introduction to Apache Spark

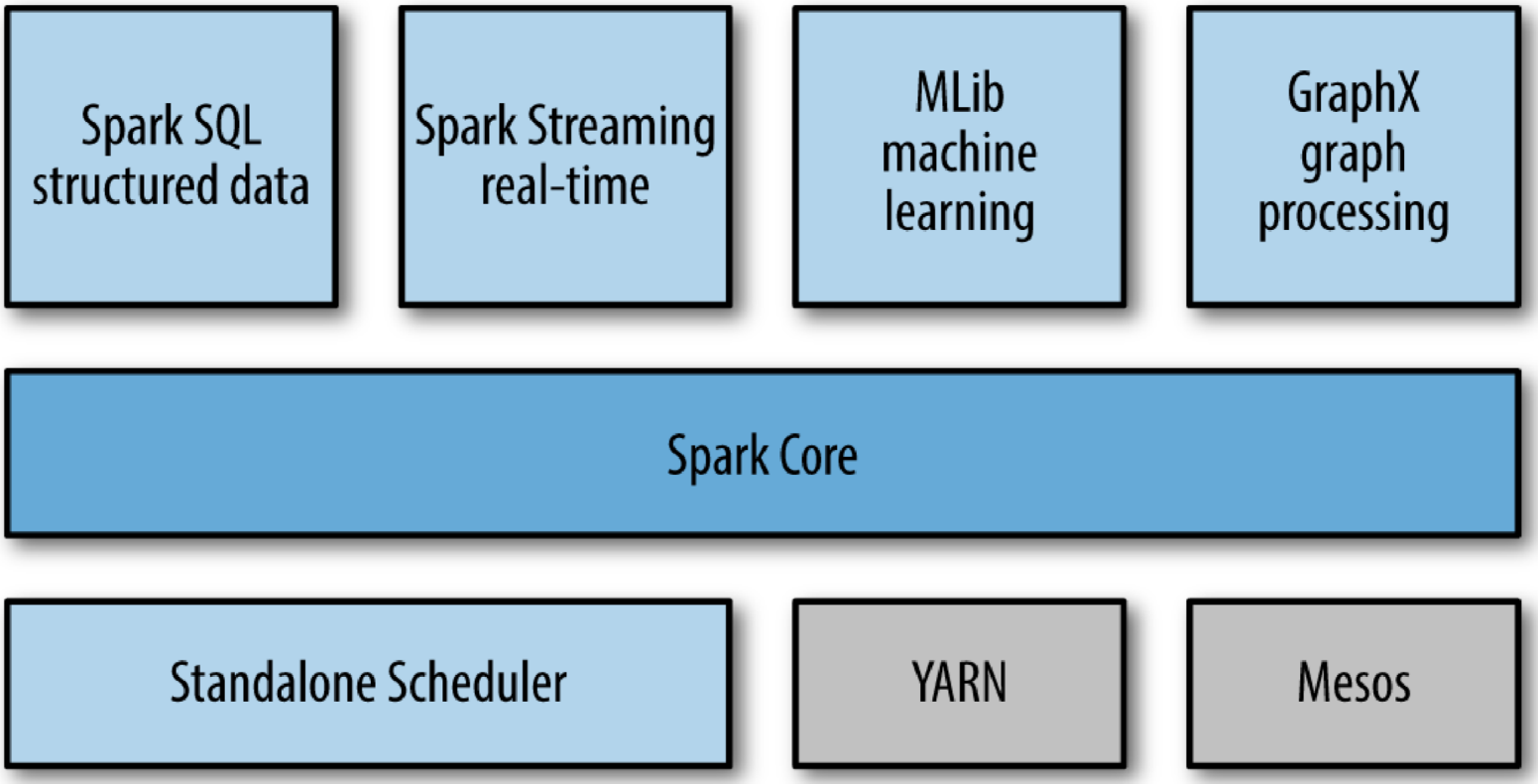
- General-purpose cluster in-memory computing system
- Provides high-level APIs in Java, Scala, python



Spark



Spark Ecosystem



Spark Core

- Contains the basic functionality for
  - task scheduling,
  - memory management,
  - fault recovery,
  - interacting with storage systems,
  - and more.
- Defines the Resilient Distributed Data sets (RDDs)
  - main Spark programming abstraction.

Spark SQL

- For working with structured data.
- View datasets as relational tables
- Define a schema of columns for a dataset
- Perform SQL queries
- Supports many sources of data
  - Hive tables, Parquet and JSON
- DataFrame

Spark Streaming

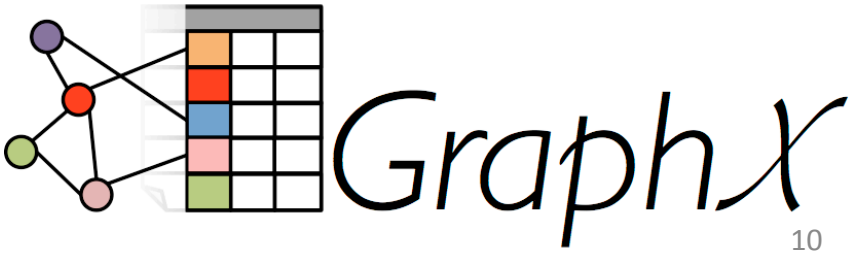
- Data analysis of streaming data
  - e.g. log files generated by production web servers
- Aimed at hight-throughput and fault-tolerant stream processing
- Dstream → Stream of datasets that contain data from certal interval

Spark MLlib

- MLlib is a library that contains common Machine Learning (ML) functionality:
  - Basic statistics
  - Classification (Naïve Bayes, decision tress, LR)
  - Clustering (k-means, Gaussian mixture, …)
  - And many others!
- All the methods are designed to scale out across a cluster.

Spark GraphX

- Graph Processing Library
- Defines a graph abstraction
  - Directed multi-graph
  - Properties attached to each edge and vertex
  - RDDs for edges and vertices
- Provides various operators for manipulating graphs (e.g. subgraph and mapVertices)



Programming with Python Spark (pySpark)

- We will use Python's interface to Spark called **pySpark**
- **A driver program** accesses the Spark enviroment throught a **SparkContext** objet
- They **key concept** in Spark are datasets called **RDD**s (**R**esilient **D**istributed Dateset)
- Basic idea: We load our data into RDDs and perform some **operations**

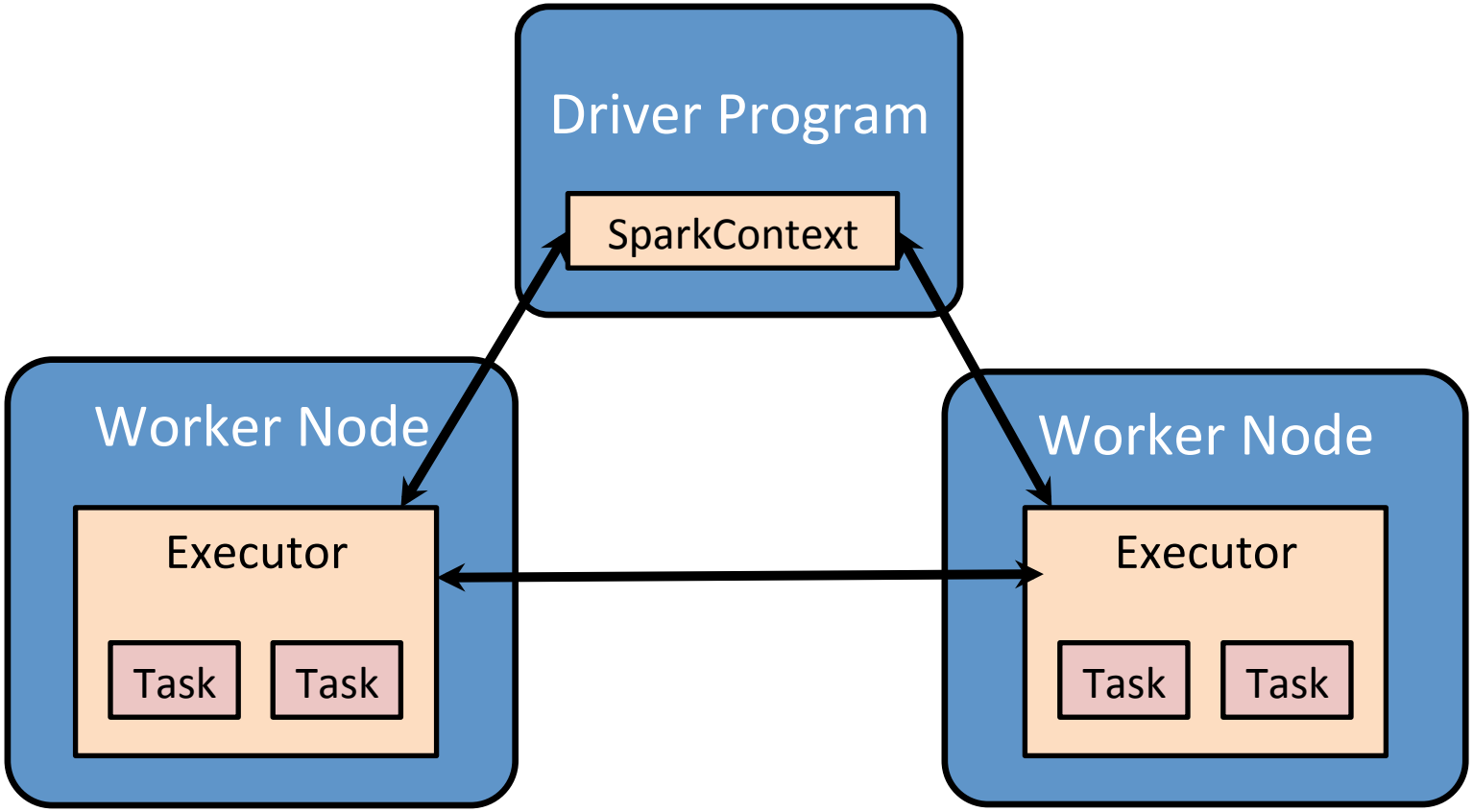
Deployment options

- Interactive mode for testing and development
  - **On local machine using shared memory and one or more cores**
  - **Or interacting with cluster**
- Job submission to a cluster manager
  - **Spark Standalone cluster**
  - **Hadoop YARN**
  - Apache Mesos
  - Amazon EC2

Programming environment
Spark concepts

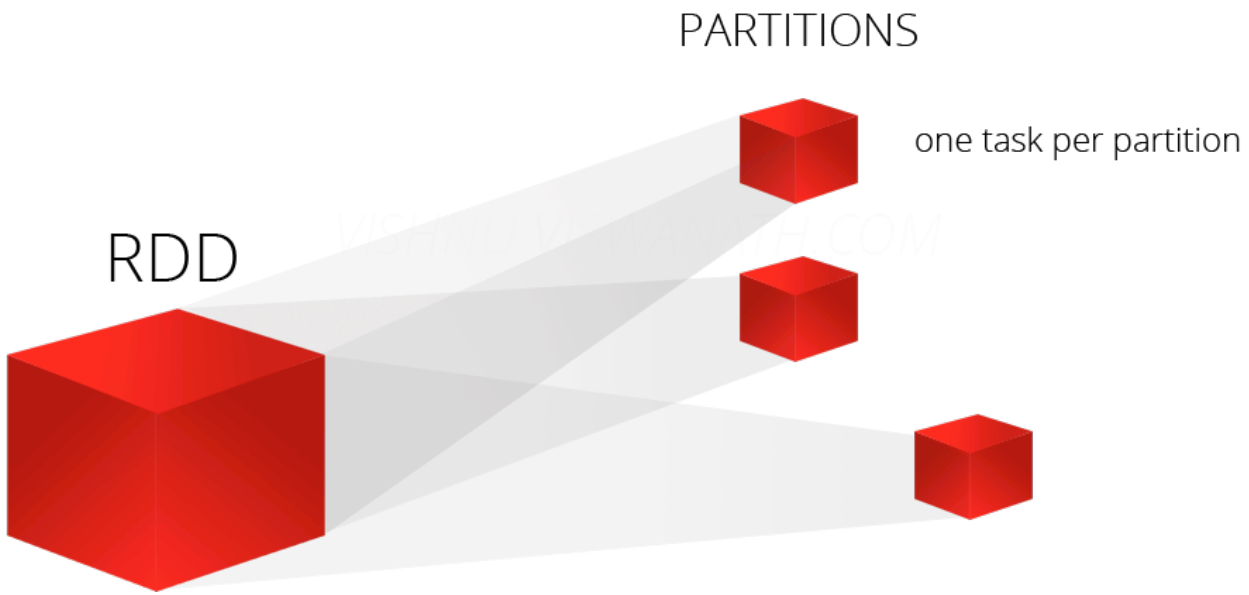
- Driver programs access Spark through a **SparkContext** object which represents a connection to the computing cluster.
- In a *shell* the **SparkContext** is created for you and available as the variable **sc**.
- You can use it to build **Resilient Distributed Data (RDD)** objects.
- Driver programs manage a number of *worker nodes* called *executors.*

Programming environment
Spark concepts



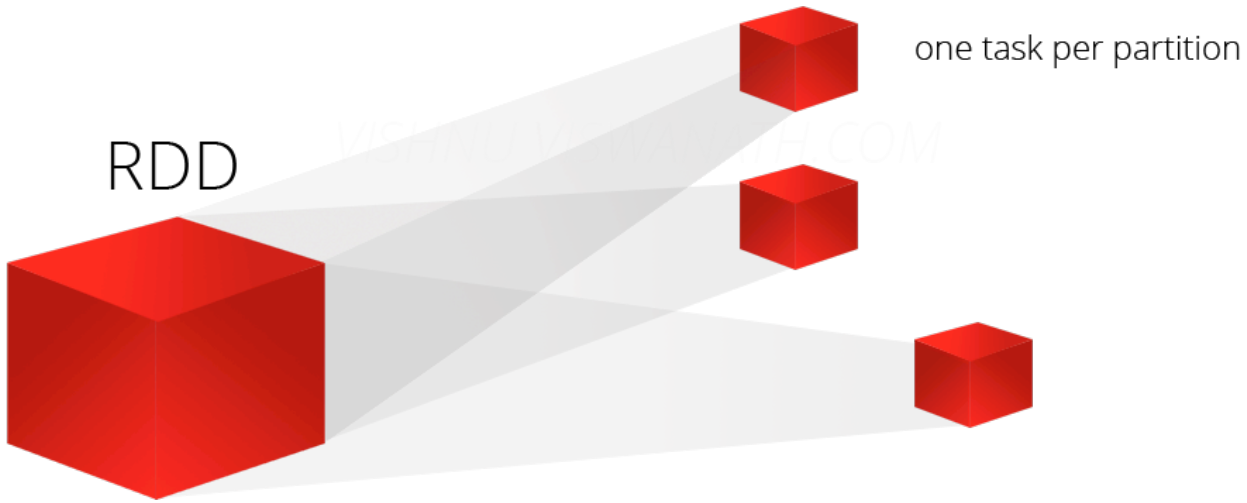
RDD abstraction

- Represent data or transformations on data
- It is distributed collection of items - partitions
- Read-only → they are immutable
- Enables operations to be performed in parallel



RDD abstraction

- Fault tolerant:
  - Lineage of data is preserved, so data can be re-created on a new node at any time
- Caching dataset in memory
  - different storage levels available
  - fallback to disk possible



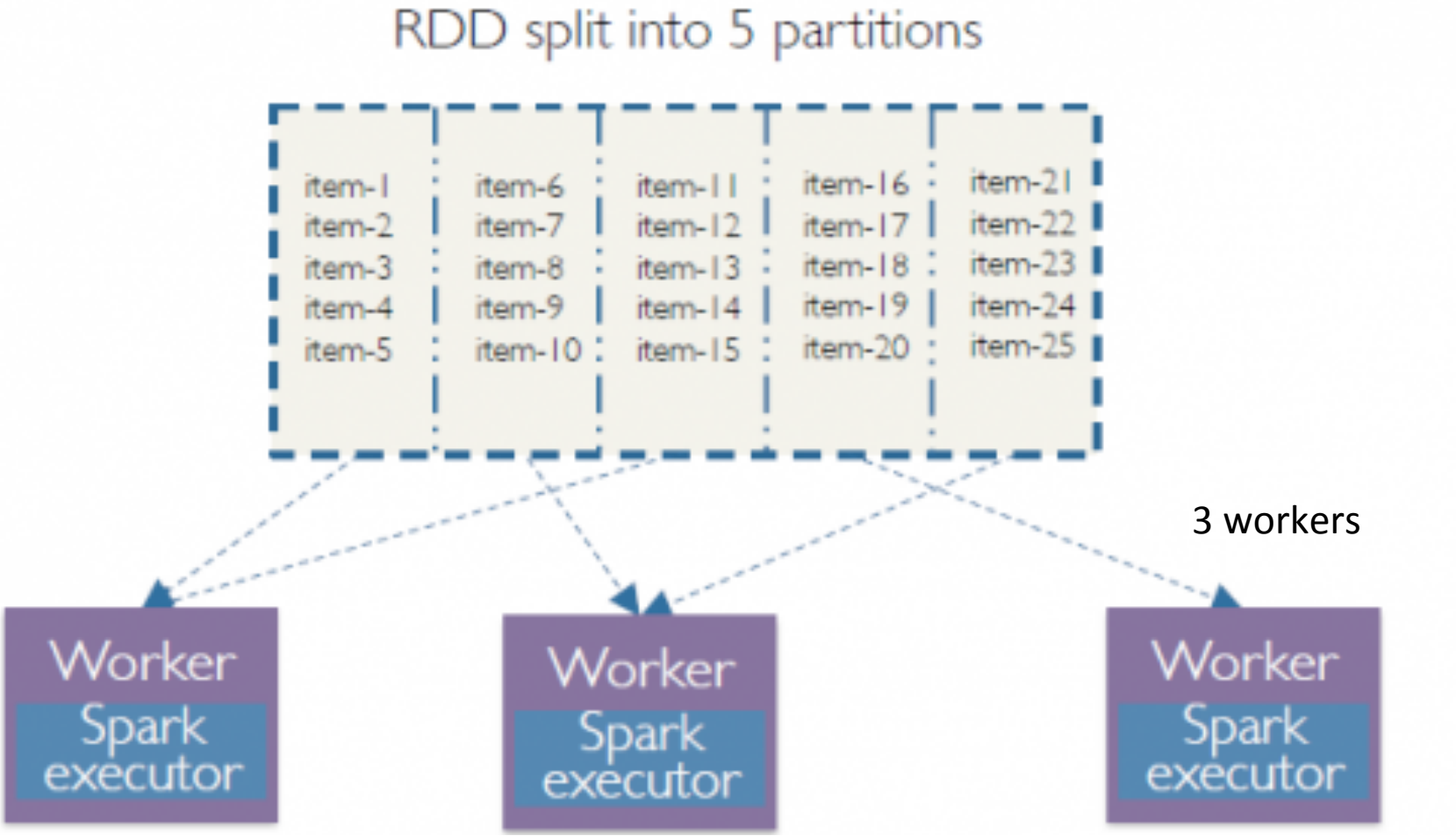
Programming with RDDs

- All work is expressed as either:
  - creating new RDDs
  - transforming existing RDDs
  - calling operations on RDDs to compute a result.

- Distributes the data contained in RDDs across the nodes (executors) in the cluster and parallelizes the operations.

- Each RDD is split into multiple **partitions**, which can be computed on different nodes of the cluster.

Partition


RDD split into 5 partitions
item-1
item-2
item-3
item-4
item-5
item-6
item-7
item-8
item-9
item-10
item-11
item-12
item-13
item-14
item-15
item-16
item-17
item-18
item-19
item-20
item-21
item-22
item-23
item-24
item-25
3 workers
Worker
Spark executor
Worker
Spark executor
Worker
Spark executor

Note about partitions

- By default, a partition is created for each HDFS partition, which by default is 64MB (from Spark's Programming Guide).

- RDDs get partitioned automatically without programmer intervention.

- But they can be adjusted → Optimization techniques

  – When the data is key-value oriented → If similar keys or range of keys are stored in the same partition then the shuffling is minimized and the processing becomes substantially fast.

First Program!

```
sc = SparkContext(master="local[*]")

lines = sc.textFile("README.md", 4)

lines.count()

pythonLines = lines.filter(lambda line : "Python" in line)

pythonLines.first()
```

Context

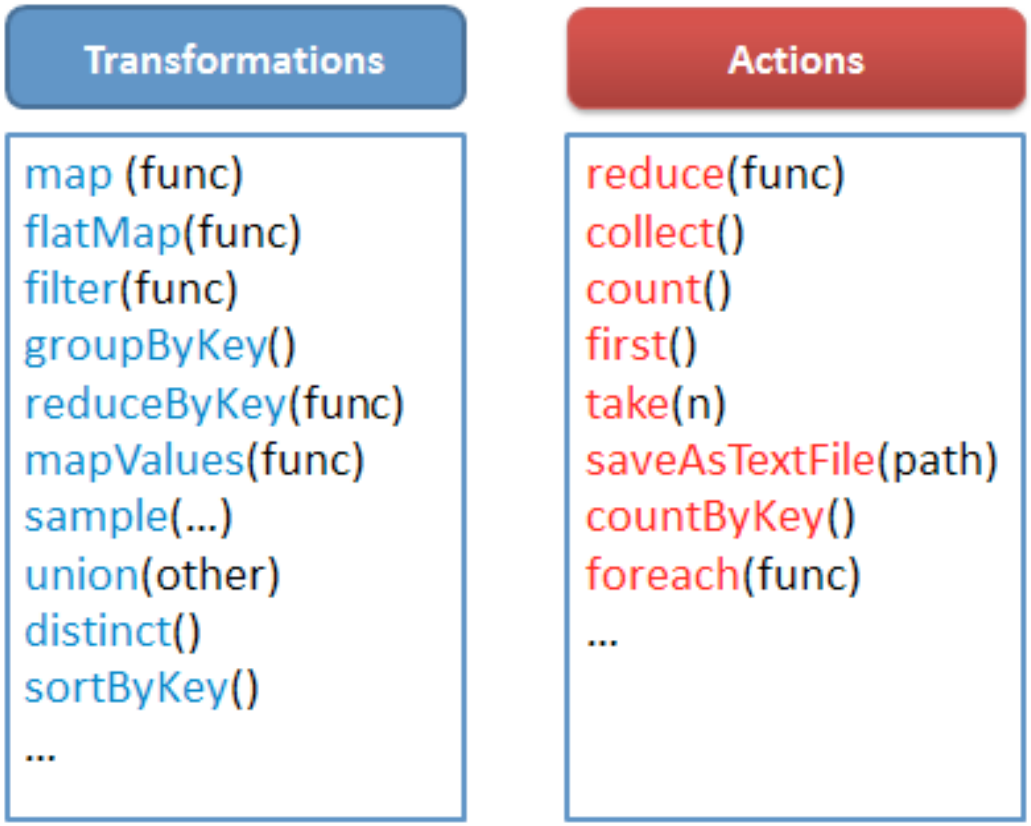
RDD-1

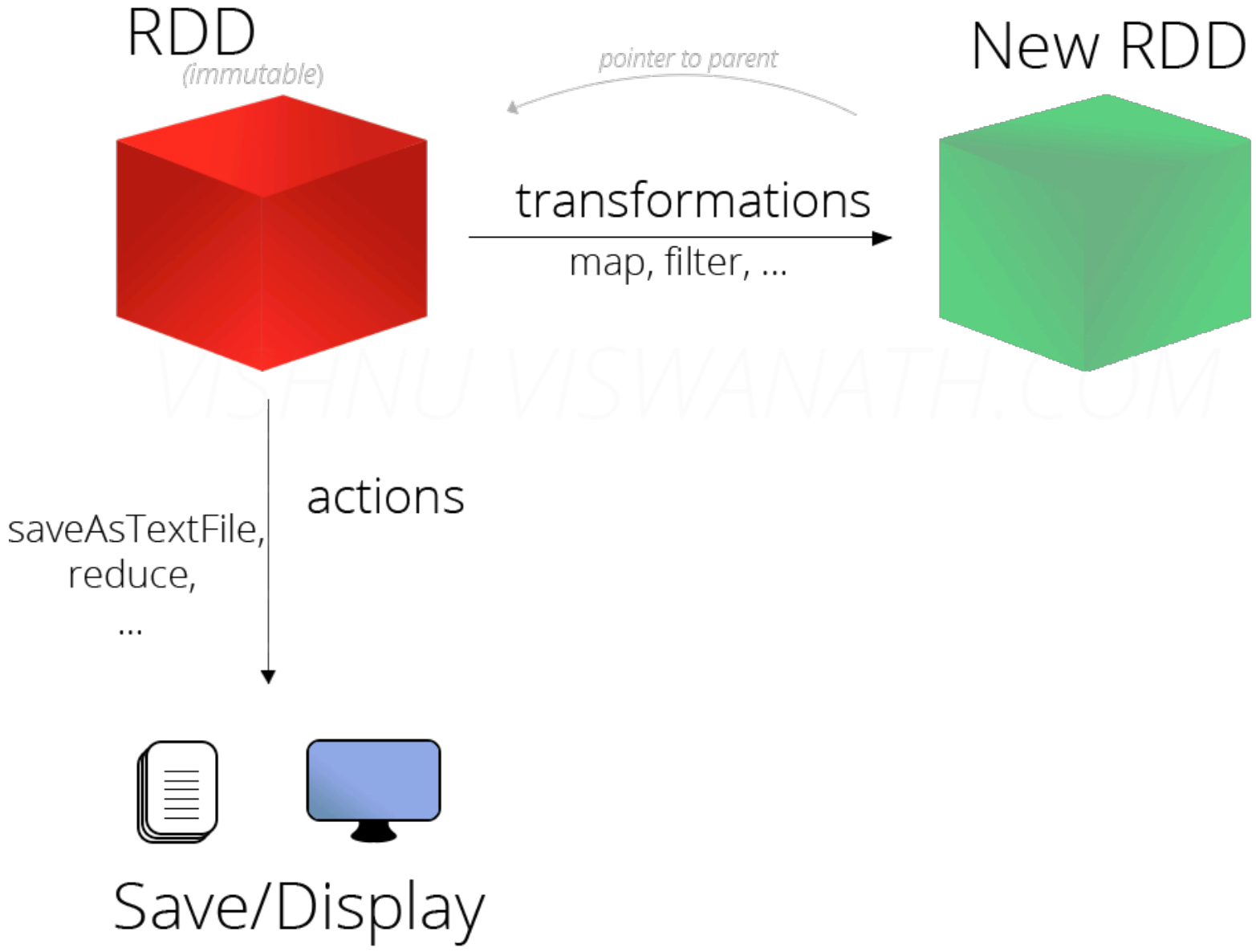
RDD-2

RDD operations

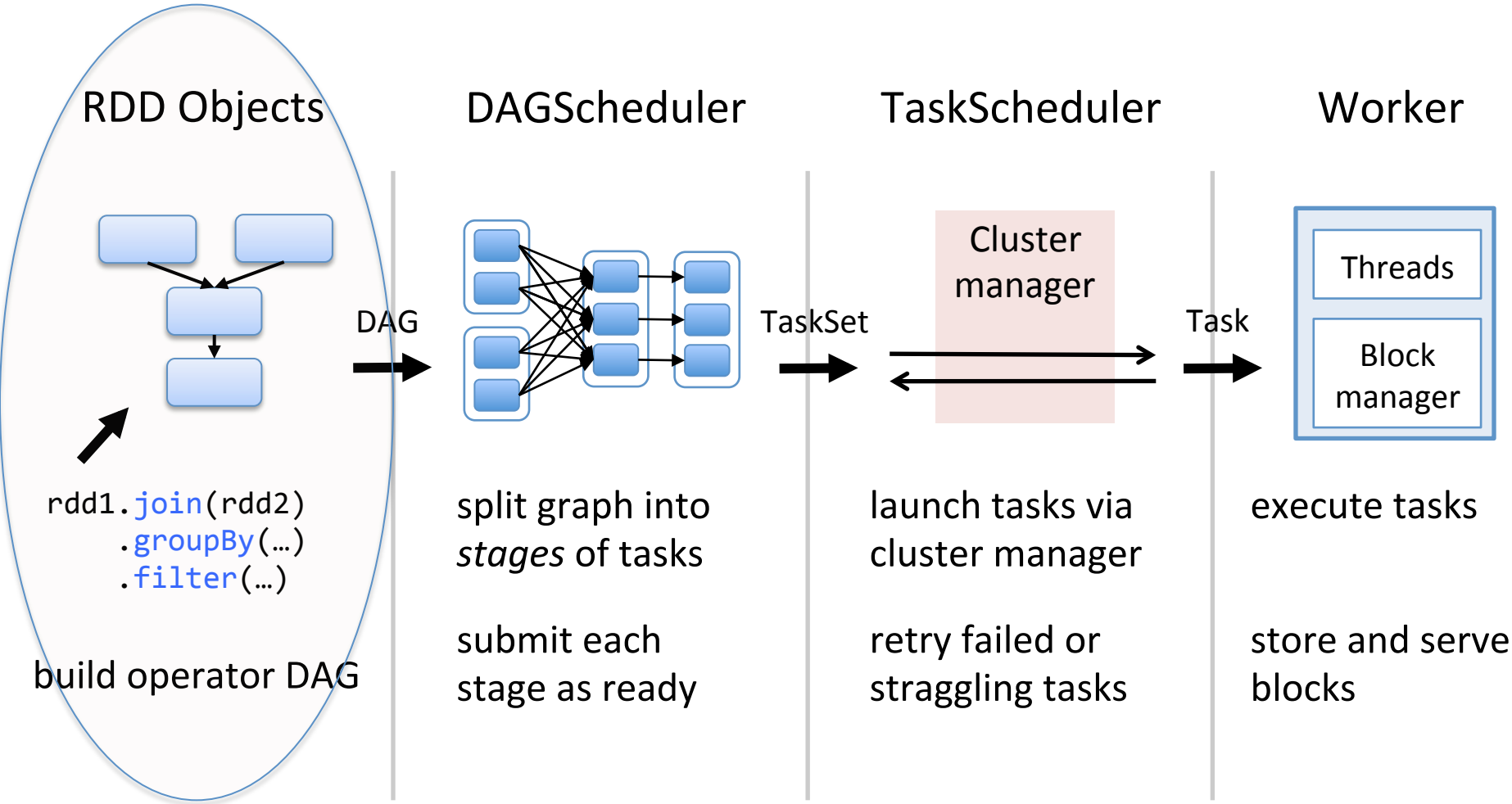
- Once created, RDDs offer two types of operations:
  - **transformations**
    - transformations include *map*, *filter*, *join*
    - lazy operation to build RDDs from other RDDs
  - **actions**
    - actions include *count*, *collect*, *save*
    - return a result or write it to storage

Transformation vs Actions





Job scheduling



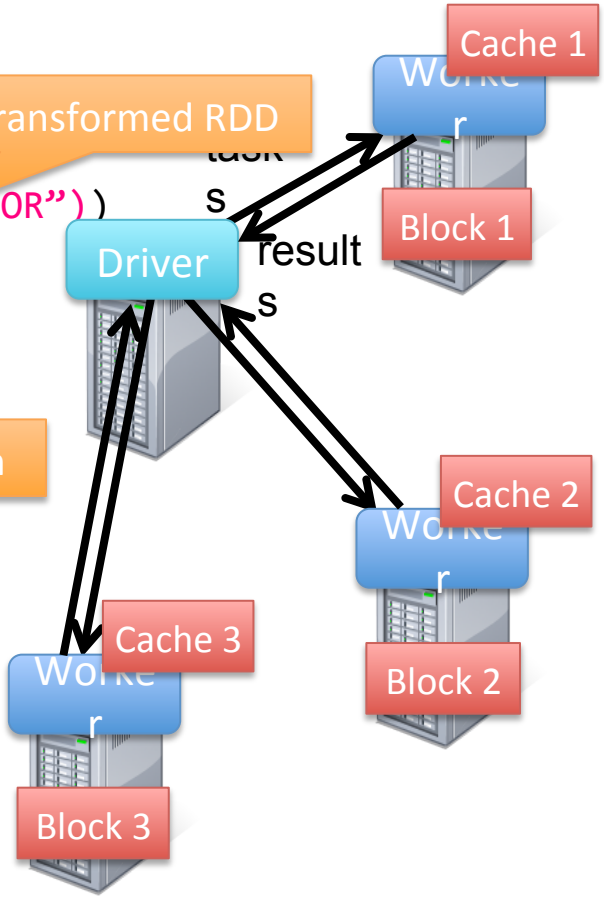
source: https://cwiki.apache.org/confluence/display/SPARK/Spark+Internals

Example: Mining Console Logs

- Load error messages from a log into memory, then interactively search for patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(‘\t’)[2])
messages.cache()


messages.filter(lambda s: “foo” in s).count()
messages.filter(lambda s: “bar” in s).count()
. . .
```



Some Apache Spark tutorials

- https://www.cloudera.com/documentation/enterprise/5-7-x/PDF/cloudera-spark.pdf
- http://www.bigdata-toronto.com/2016/assets/getting_started_with_apache_spark.pdf
- https://stanford.edu/~rezab/sparkclass/slides/itas_workshop.pdf
- http://ictlabs-summer-school.sics.se/2015/slides/spark.pdf