GENERAL ASSEMBLY

Intro to Spark

Patrick Smith

OPENING

Intro to Spark

Intro to Spark

Today will be dedicated to Spark. Spark has brought a revolution in Big Data in the past few years and it is thus important to introduce it and explain how it differs from Hadoop.

Intro to Spark

Let’s recap MapReduce - how does it work?

Intro to Spark

What limitations have you encountered when processing data with Hadoop?

Spark: An introduction

Intro to Spark

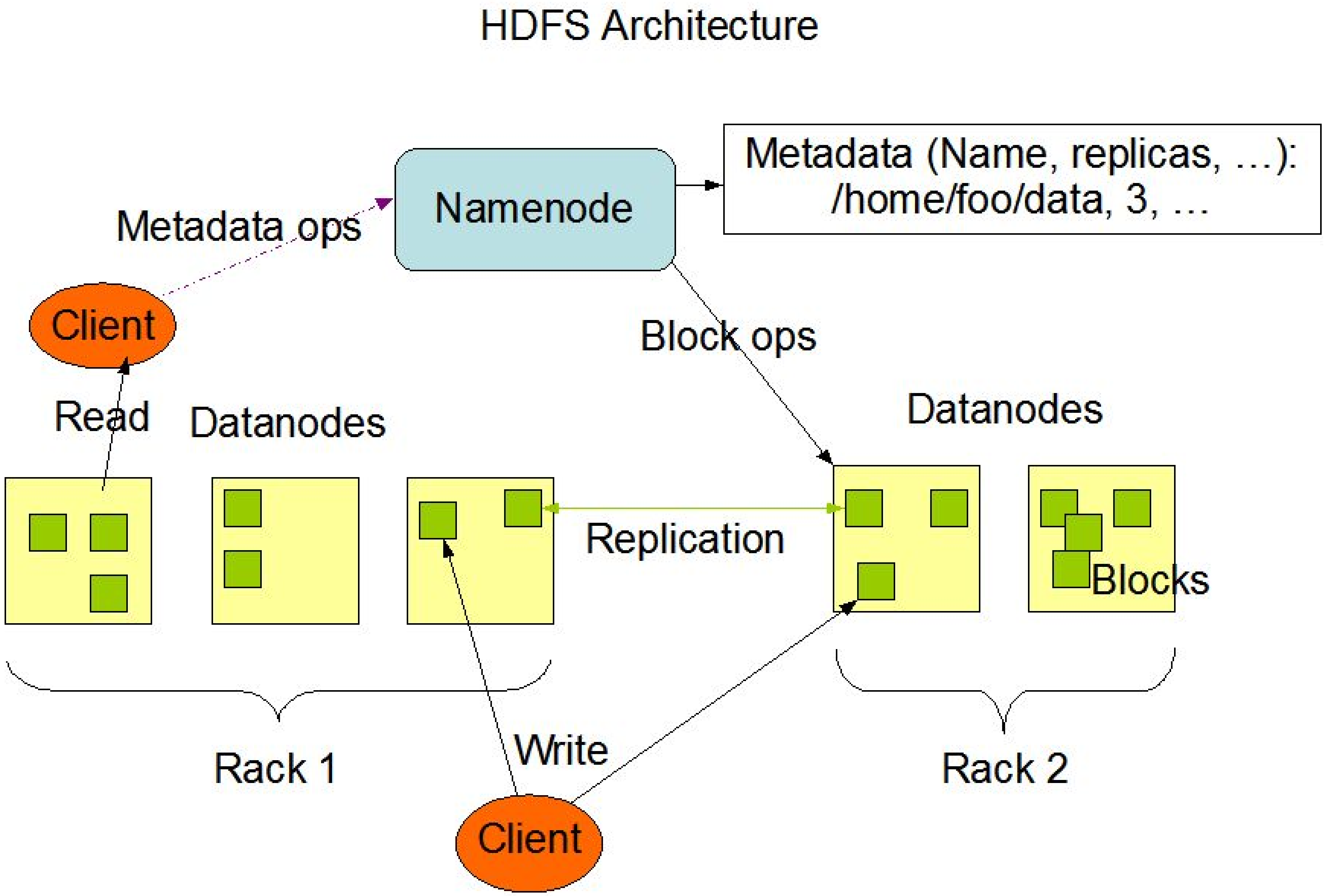
Apache Spark is an open source cluster computing framework.

- Originally developed at the University of California, Berkeley's AMPLab, Spark is now open source
- Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance.

Intro to Spark

- Spark has gained traction over the past few years because of its superior performance with respect to Hadoop-MapReduce.
- In MapReduce data is read from disk *and then* a function is mapped across the data. *Then* the reducer will reduce the results of the map and finally store reduction results back to HDFS.

Intro to Spark

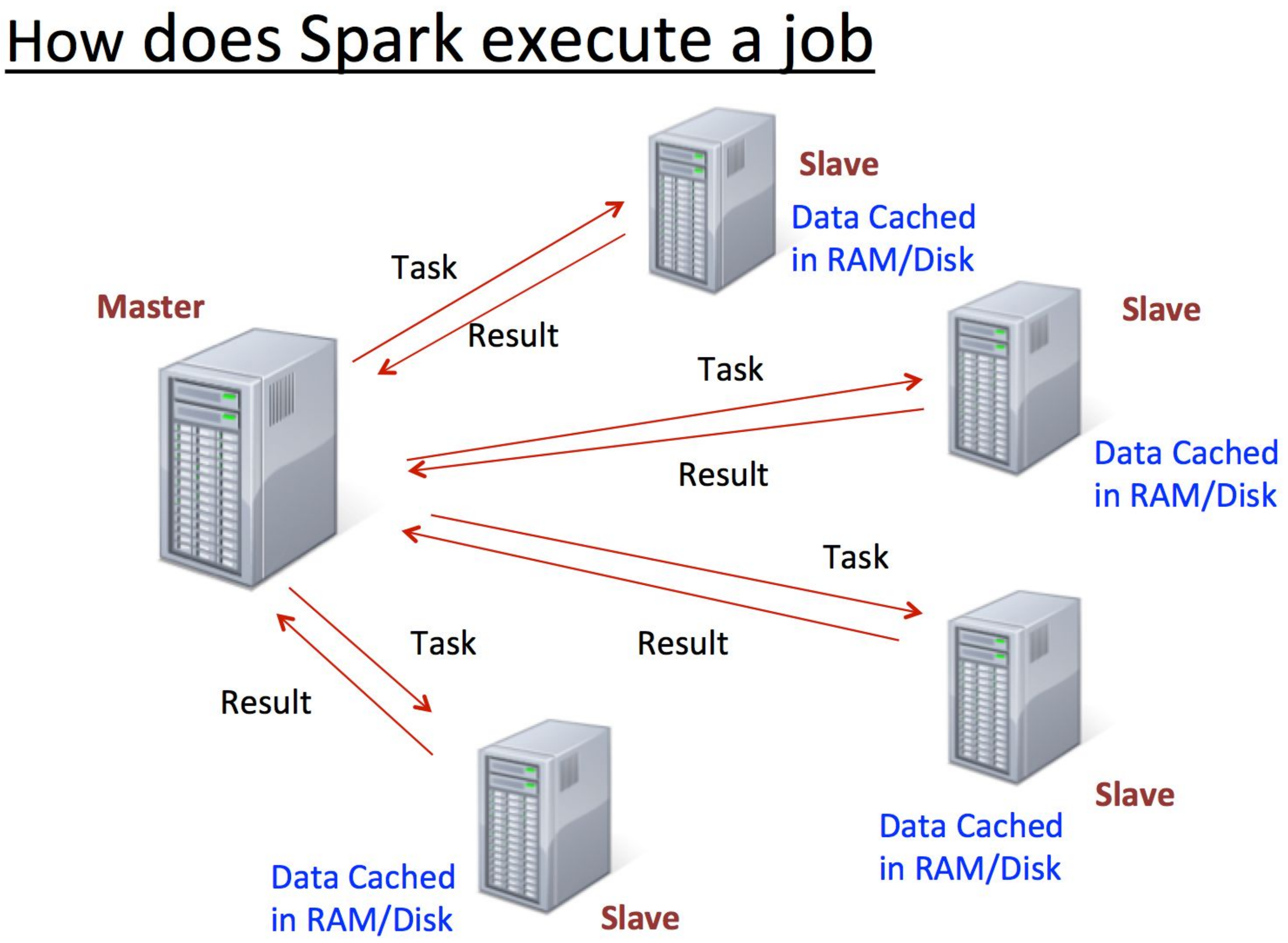


Intro to Spark

Spark relaxes the constraints of MR by doing the following:

- Generalizes computation from Map/Reduce only graphs to arbitrary Directed Acyclic Graphs (DAGs)
- Removes a lot of boilerplate code present in Hadoop allows to "tweak" parts that in Hadoop are not accessible, like for example the sort algorithm
- Allows to load data in a cluster memory, speeding up I/O enormously
- The two pillars on which Spark is based are RDDs and DAGs.

Intro to Spark



RDDs and DAGs

RDD

Apache Spark provides programmers with an application programming interface centered on a data structure called the **resilient distributed dataset (RDD)**, a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way.

RDD

Spark is keeping the data in memory, instead of on disk, thus making it easier to implement both iterative algorithms, that visit their dataset multiple times in a loop, and interactive/exploratory data analysis, i.e., the repeated database-style querying of data.

RDD

Apache Spark requires a cluster manager and a distributed storage system.

For cluster management, Spark supports standalone (native Spark cluster), Hadoop YARN, or Apache Mesos.

For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS), MapR File System (MapR-FS), Cassandra, OpenStack Swift, Amazon S3, Kudu, or a custom solution can be implemented.

DAG - Directed Acrylic Graph

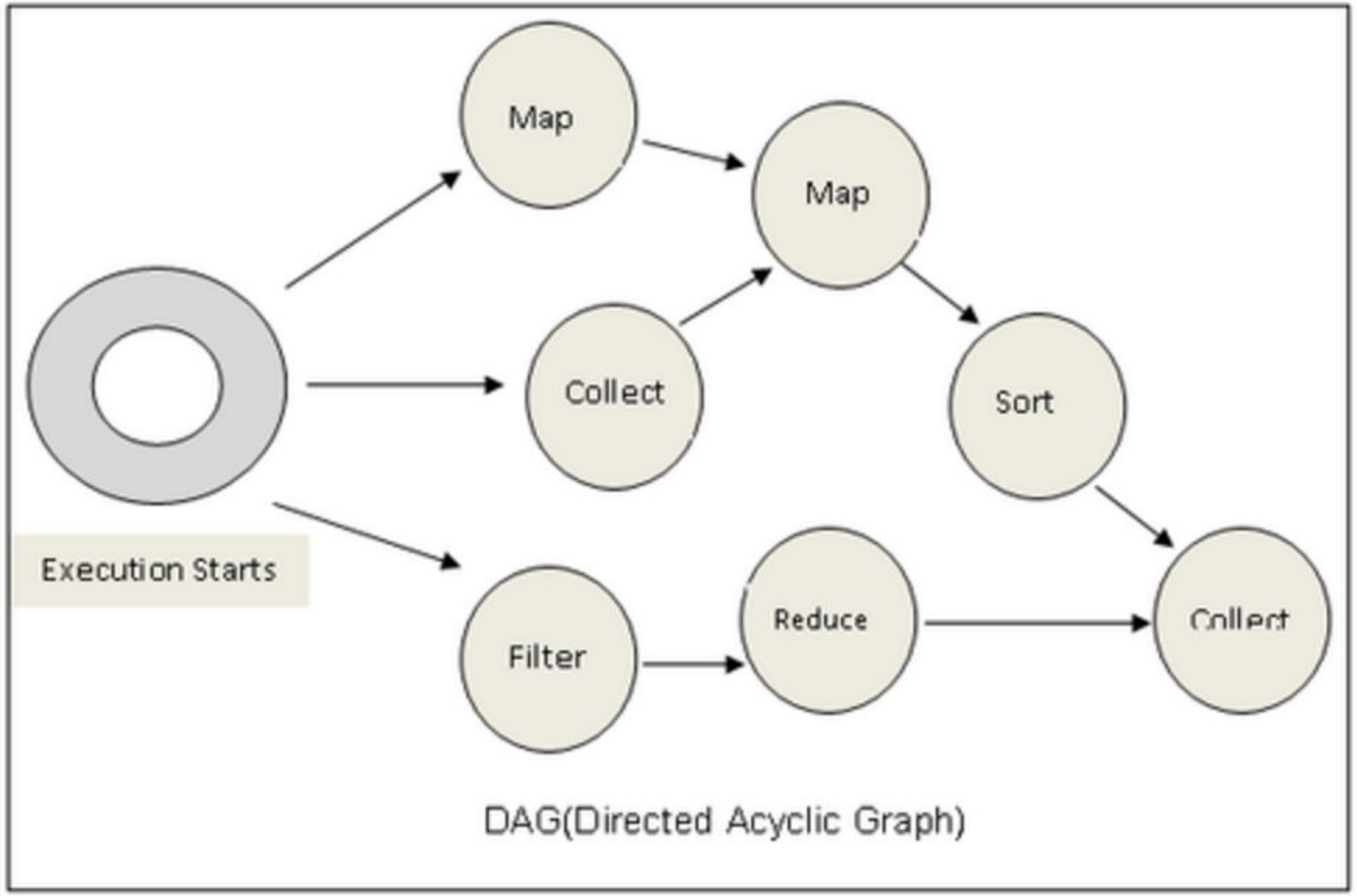
DAG (Directed Acyclic Graph) is a programming style for distributed systems - You can think of it as an alternative to Map Reduce.

DAG - Directed Acrylic Graph

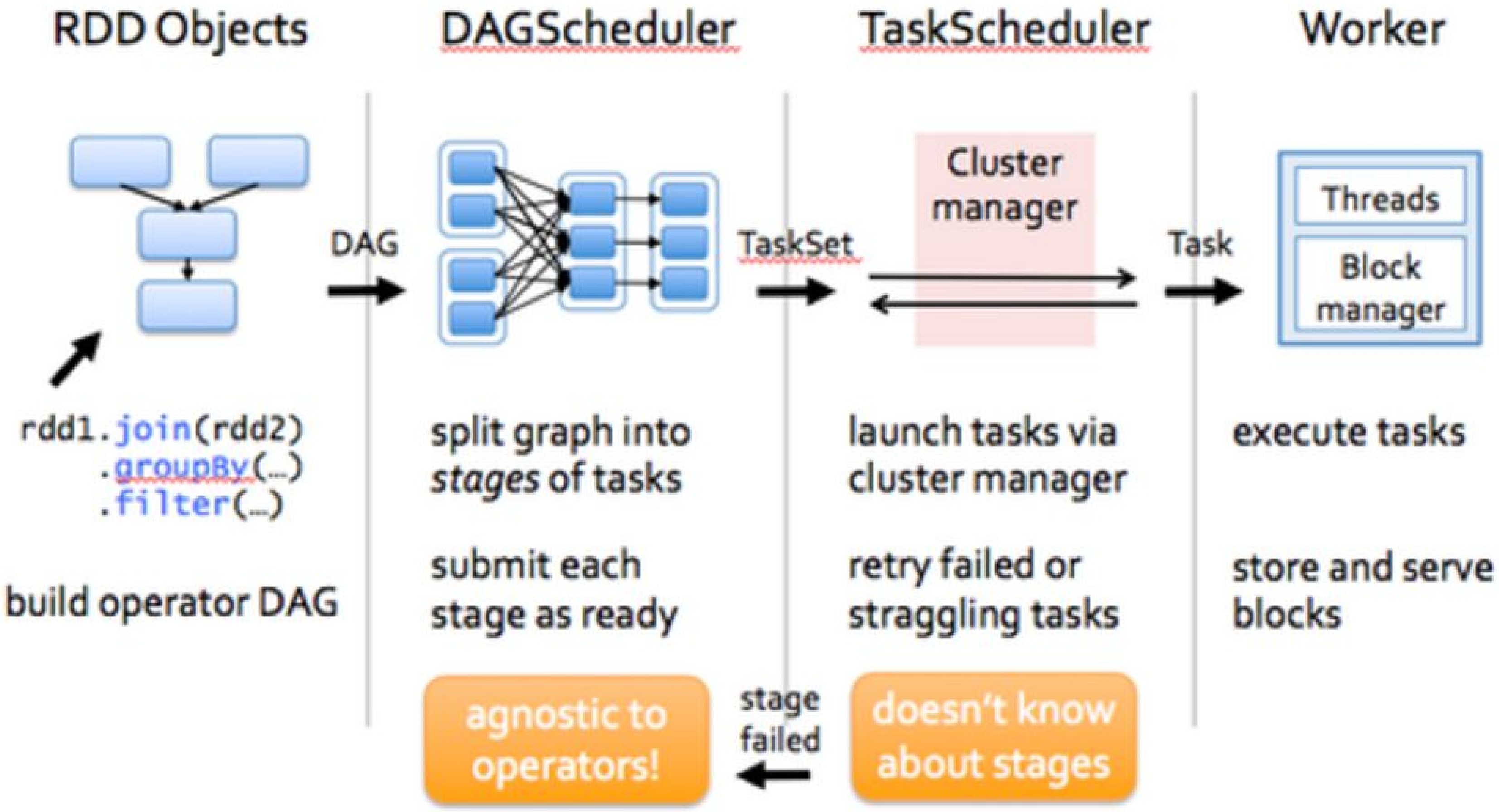
DAG (Directed Acyclic Graph) is a programming style for distributed systems - You can think of it as an alternative to Map Reduce.

While MR has just two steps (map and reduce), DAG can have multiple levels that can form a tree structure.

DAG - Directed Acrylic Graph

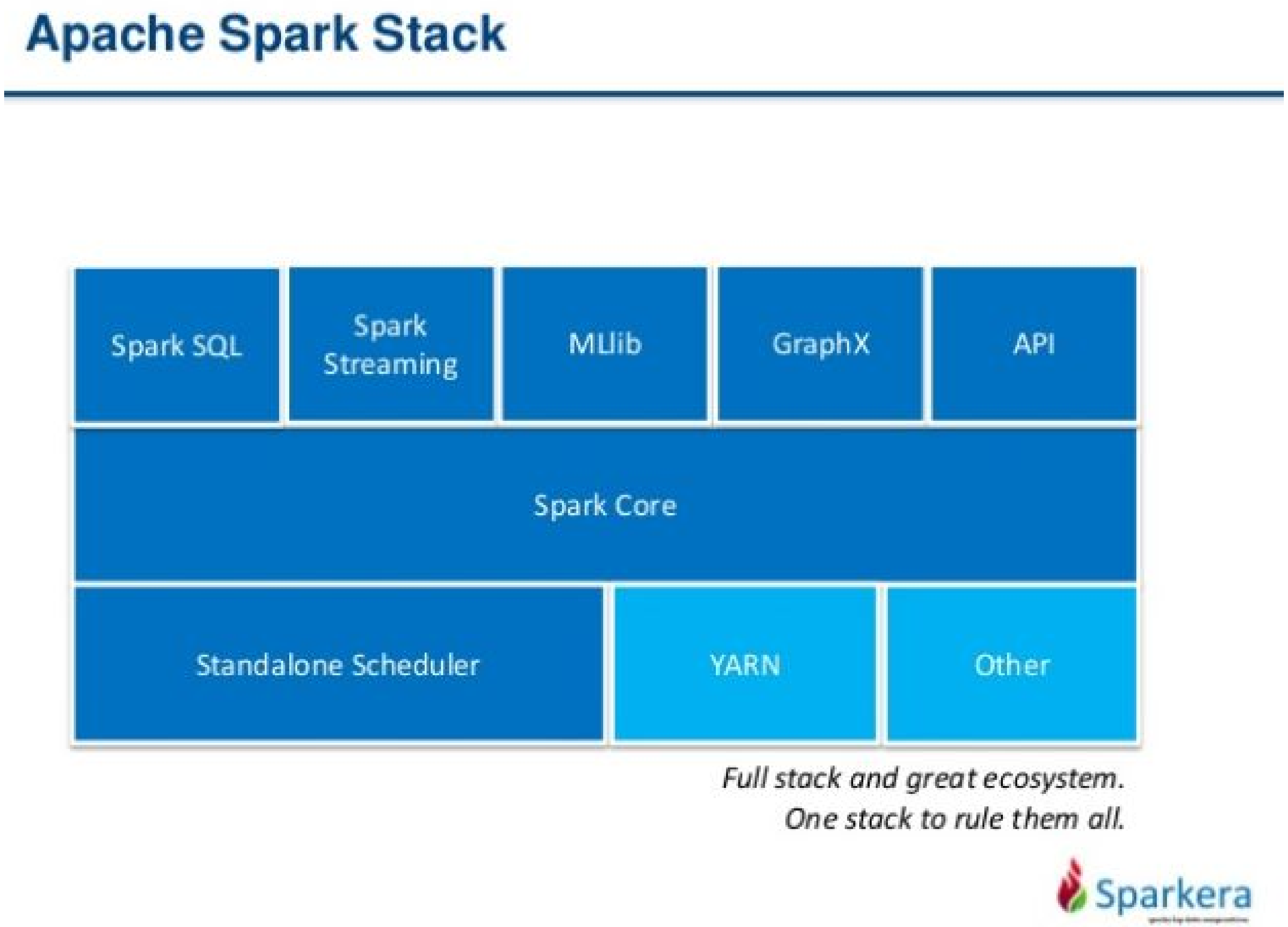


DAG - Directed Acrylic Graph



Spark Stack and API (10 min)

Intro to Spark



Apache Spark Stack

Spark SQL

Spark Streaming

MLlib

GraphX

API

Spark Core

Standalone Scheduler

YARN

Other

Full stack and great ecosystem.
One stack to rule them all.

Sparkera

Intro to Spark

The Spark Core is the foundation of the overall project. It provides distributed task dispatching, scheduling, and basic I/O functionalities, exposed through an application programming interface (for Java, Python, Scala, and R) centered on the RDD abstraction.

Intro to Spark

The Spark Core is the foundation of the overall project. It provides distributed task dispatching, scheduling, and basic I/O functionalities, exposed through an application programming interface (for Java, Python, Scala, and R) centered on the RDD abstraction.

Spark is built in Scala, a language derived from Java, that has all the support for functional programing languages, but also has support for OOP languages. Spark builds computation by concatenating functions in the DAG.

Spark Variables:

Spark provides two forms of shared variables:

- Broadcast variables: they reference read-only data that needs to be available on all nodes
- Accumulators: they can be used to program reductions in an imperative style

Spark Operations:

Spark provides two types of operations:

- Transformations: these are "lazy" operations that only return a result upon "collect"
- Actions: these are "non-lazy" operations that immediately return a result
- Using lazy operations, we can build a computation graph that only gets executed when we collect the result. This allows Spark to optimize the requested calculation by optimizing the underlying DAG of operations.

Spark MapReduce (15 min)

Guided Practice: Spark MapReduce (10 min)

Independent Practice: Explore the Spark Shell (15 min)

Intro to Spark

Conclusion

Q & A