Apache-Spark Ecosystem and Key Innovations

Linxiao Bai
Nilesh Patil

Introduction

Spark was designed for massive parallel computation on large-scale distributed cluster. It is currently the most popular open-source project of Apache Software Foundation, and the first choice of “big-data engine” for cluster-computation.

Although Spark Core learns from Hadoop MapReduce framework at lower level, with extensive optimizations and design choices, it quickly outperformed its predecessor, providing faster, more resilient, more convenient cluster-computation experience. Also, with extensive modules and libraries like Spark SQL, Spark Streaming, GraphX, and MLlib, it is a powerful platform to cater different computation needs, and different environments.

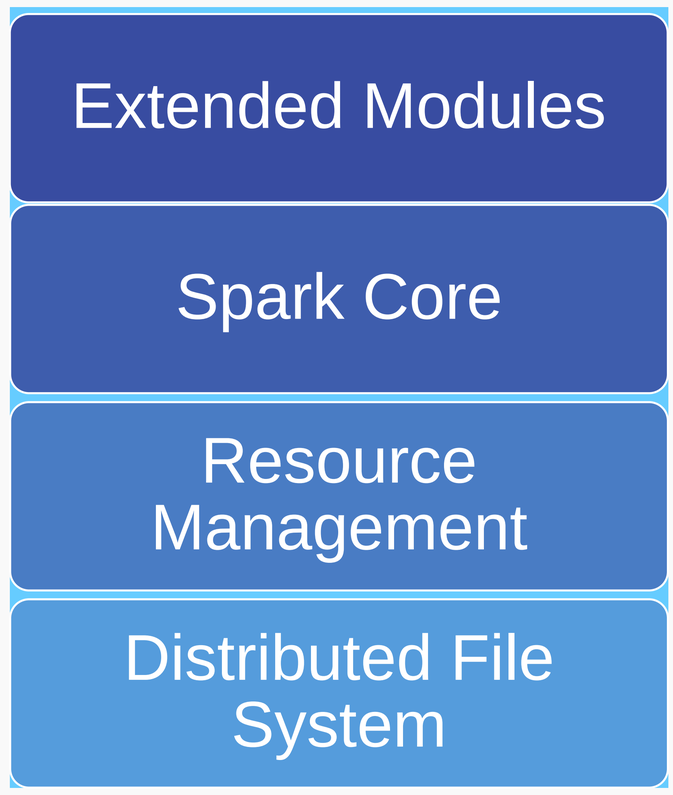
These powerful capabilities of Spark, understanding its key features and advantages is important for anyone who works in data-driven fields.

Objective

Our goal is to provide an overview of Apache Spark ecosystem and several key Spark modules. Including lower-level interfaces to machine and higher-level interfaces to users.

Also, we will compare Spark with its predecessor Hadoop, and review its advantages. The intention is to provide Spark intermediate learners with a better understanding of this powerful platform and offer a different perspective.

Spark Ecosystem



Modules and Innovations

Other Extended Modules

- GraphX — API for graphs and large scale graph-parallel computation.
- MLlib — A Spark scalable machine learning library.
- Spark Streaming — API for scalable streaming applications
- Zeppelin — Notebook like interactive kernel

User

Spark SQL

The most popular Spark extension.
Powerful relational data processing capability

- Implement DataFrame like in Python and R
- Uses schema at each data partition to keep track of the relational transformation.
- Uses “Catalyst” to optimize the query logic.

Resilience

The resilience of Spark relies on DAGScheduler, a process that record the lineage of all RDD partitions.

Compared to Hadoop:

Hadoop:
- One executor fails, all fail.
- Computation start-over.
- Relies on saving result to HDFS.
- Huge disk I/O, waste of communication.

Spark:
- One executor fails, others move on
- Only the faulty partition needs re-compute
- Relies on lineage to trace-back for computing the faulty partition.
- Minimum I/O, Maximum efficiency
- Allocate help to struggling executors.

In-Memory

Intermediate result of computations will not be serialized and saved to disk. RDD stays in memory as a reusable object.

Compared to Hadoop:

Hadoop:
- Result of map-reduce save to disk, and make multiple copies.
- Serialization/deserialization repetitively between computations
- Huge disk I/O, waste of communication.

Spark:
- Lazy execution, execute tasks batch-wisely
- Result saved as reusable object in memory for next computation.
- No serialization/deserialization.
- Minimum disk I/O. Lightning fast !

Upper-level

Spark Core

Encapsulation

Spark encapsulates parallelism to friendly APIs.
User programs without worrying about too much details.

Compared to Hadoop:

Hadoop:
- Users are responsible for details like exception handling.
- Massive declaration before start.
- Redundant coding, similar modules appears every time. “WordCount” takes 120 lines of code

Spark:
- Minimum declarations, usually only **SparkContext**.
- Friendly interfaces, users only worry about job itself.
- Parallelisms are well-encapsulated. “WordCount” only 5 lines of code.

Generality

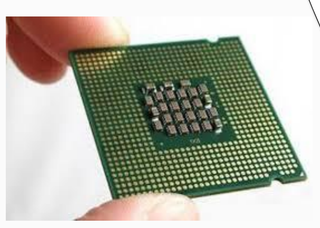
Spark offers great generality for application development and cluster deploying.

Compared to Hadoop:

Hadoop:
- Needs to be deployed on Hadoop cluster.
- Heartbeat-based communication, Slow.
- Non-universal API.
- Java indigenous.

Spark:
- Support platforms of different kinds and sizes. From local PC to large-scale cluster.
- Multiple language kernels
- Universal API. Akka fast communication.
- Powerful extensions and libraries.

Lower-level



Resource Manager

Wide-raged support includes:

- Hadoop YARN
- Apache MESOS
- Amazon EC2

Spark-Standalone **Spark-Local**

File System

Wide-raged support includes:

- HDFS
- Apache HBase
- Hive
- cassandra
- TACHYON

Machine