Concepts and Models of Parallel and Data-centric Programming

Apache Spark – Introduction

Lecture, Summer 2020

Simon Schwitanski
Dr. Christian Terboven

i12 High Performance Computing
RWTH AACHEN UNIVERSITY

Outline

0. Organization
1. Foundations
2. Shared Memory
3. GPU Programming
4. Bulk-Synchronous Parallelism
5. Message Passing
6. Distributed Shared Memory
7. Parallel Algorithms
8. Parallel I/O
9. MapReduce
10. **Apache Spark**

a. Spark Programming Model
b. Resilient Distributed Datasets (RDDs)
c. Job Scheduling and Fault Tolerance
d. Streaming and Applications
e. Concluding Remarks

Motivation

- MapReduce: Abstractions for large-scale data processing
  - Load balancing, fault tolerance etc. handled by framework
- **Problem:** MapReduce jobs only disk-based
  - Inefficient when reusing intermediate results, especially for iterative algorithms
  - Interactive querying of data slow
- **Result:** *Iterative jobs* and *interactive analyses* lead to enormous disk I/O
- **Solution:** Implement systems caching data in memory for reuse
  - Specialized frameworks: Pregel, HaLoop, Apache Storm, …
  - More general framework: **Apache Spark**

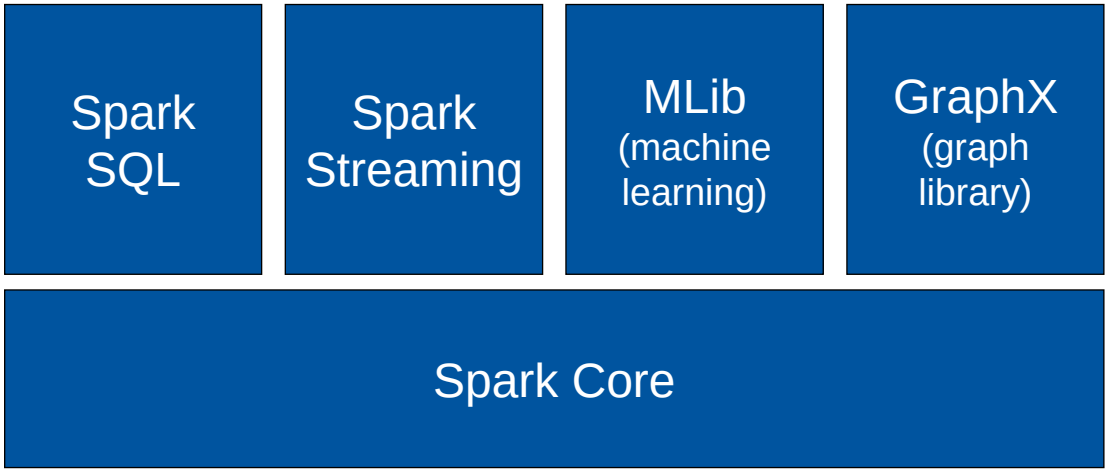
Brief History

- 2010: Spark source published by Matei Zharia at UC Berkeley's AMPLab
- 2010: Paper introducing Spark by Matei Zharia (UC Berkeley)
- 2012: Paper describing Resilient Distributed Datasets (RDDs)
- 2014: Spark gets top-level Apache project
- 2016: More than 1000 organizations using Spark in production [1]

[1] https://spark.apache.org/faq.html

Framework Overview

- Spark Core provides fundamental functionalities (RDDs, transformations, …)
- Spark Core written in Scala (~ 14.000 LOC)
- Several abstractions build upon the core functionalities
- Spark Core and abstractions form the Apache Spark project



Goals

- **Speed:** Programs up to 100 times faster than Hadoop MapReduce (in-memory computing)
- **Ease of Use:** Java, Scala, Python and R bindings
- **Generality:** Integrating a stack of different libraries (SQL, Streaming, Machine Learning, …)
- **Runs Everywhere**
  - Supports YARN and other cluster frameworks as well as a standalone mode
  - Supports various data sources like HDFS, Cassandra, HBase and S3