Apache Spark is an analytics engine to process data at a large scale. To accomplish high speed while analysing, it provides an interface for programming clusters, so that data can be processed simultaneously. It also provides fault tolerance through Resilient Distributed Datasets (RDDs) at its architectural foundation. Spark itself needs a Cluster Manager and a Distributed Storage System to work with.

Supported programming languages
- Python
- Scala
- Java
- R

Distributed Data Storage Systems
- Alluxio
- MAPR
- Amazon Web Services
- S3
- Cassandra
- Hadoop

Cluster Managers
- Hadoop
- Kubernetes

introduces Data Frames on top of Spark API, so that structured and semi-structured data can be analysed through a DSL for Python, Java or Scala.

provides an API for building scalable applications for stream processing. It supports Kafka, Flume, Twitter, Zero MQ, Kinesis and TCP/IP-Sockets. Alternatives are Storm and Apache Flink Streaming.

includes functions for machine learning while it leverages the speed of Spark for iterative algorithms to run faster than e.g. Apache Mahout or Vowpal Wabbit.

is a distributed graph processing framework, providing two API’s for the implementation of parallel algorithms, but only capable of processing immutable graphs. A similar framework is Apache Giraph, which uses Hadoop’s MapReduce algorithm.

Turning a Spark into a Flare

Spark was originally designed to scale-out on clusters and though it might scale well, it creates an overhead that makes executing a simple query 20 times slower in Spark than executing it in C. To speed up Spark, Flare provides a compiler for Catalyst query plans, turning them to native code. This achieves a similar performance as the C code.

Lecture: "A Programming Language and Compiler View on Data Management and Machine Learning Systems" by Tiark Rompf