# Apache Spark

In this lesson, you learned that:

Spark is an open-source, in-memory application framework for distributed data processing and iterative analysis on massive data volumes. Both distributed systems and Apache Spark are inherently scalable and fault-tolerant.

Apache Spark solves the problems encountered with MapReduce by keeping a substantial portion of the data required in memory, avoiding expensive and time-consuming disk I/O.

Functional programming follows a declarative programming model that emphasizes what instead of how-to and uses expressions.

Lambda functions or operators are anonymous functions that enable functional programming. Spark parallelizes computations using lambda calculus, and all functional Spark programs are inherently parallel.

Resilient distributed datasets, or RDDs, are Spark’s primary data abstraction consisting of a fault-tolerant collection of elements partitioned across the cluster’s nodes, capable of accepting parallel operations.

You can create an RDD using an external or local Hadoop-supported file, from a collection, or from another RDD. RDDs are immutable and always recoverable, providing resilience in Apache Spark. RDDs can persist or cache datasets in memory across operations, which speeds up iterative operations in Spark.

Apache Spark architecture consists of components data, compute input, and management. The fault-tolerant Spark Core base engine performs large-scale big data–worthy parallel and distributed data processing jobs, manages memory, schedules tasks, and houses APIs that define RDDs.

Spark SQL provides a programming abstraction called DataFrames and can also act as a distributed SQL query engine.

Spark DataFrames are conceptually equivalent to a table in a relational database or a data frame in R or Python, but with richer optimizations.