Apache Spark

Dr. Mihail

Content derived from:

Ankam, Venkat. Big Data Analytics. Packt Publishing, 2016.

July 9, 2019

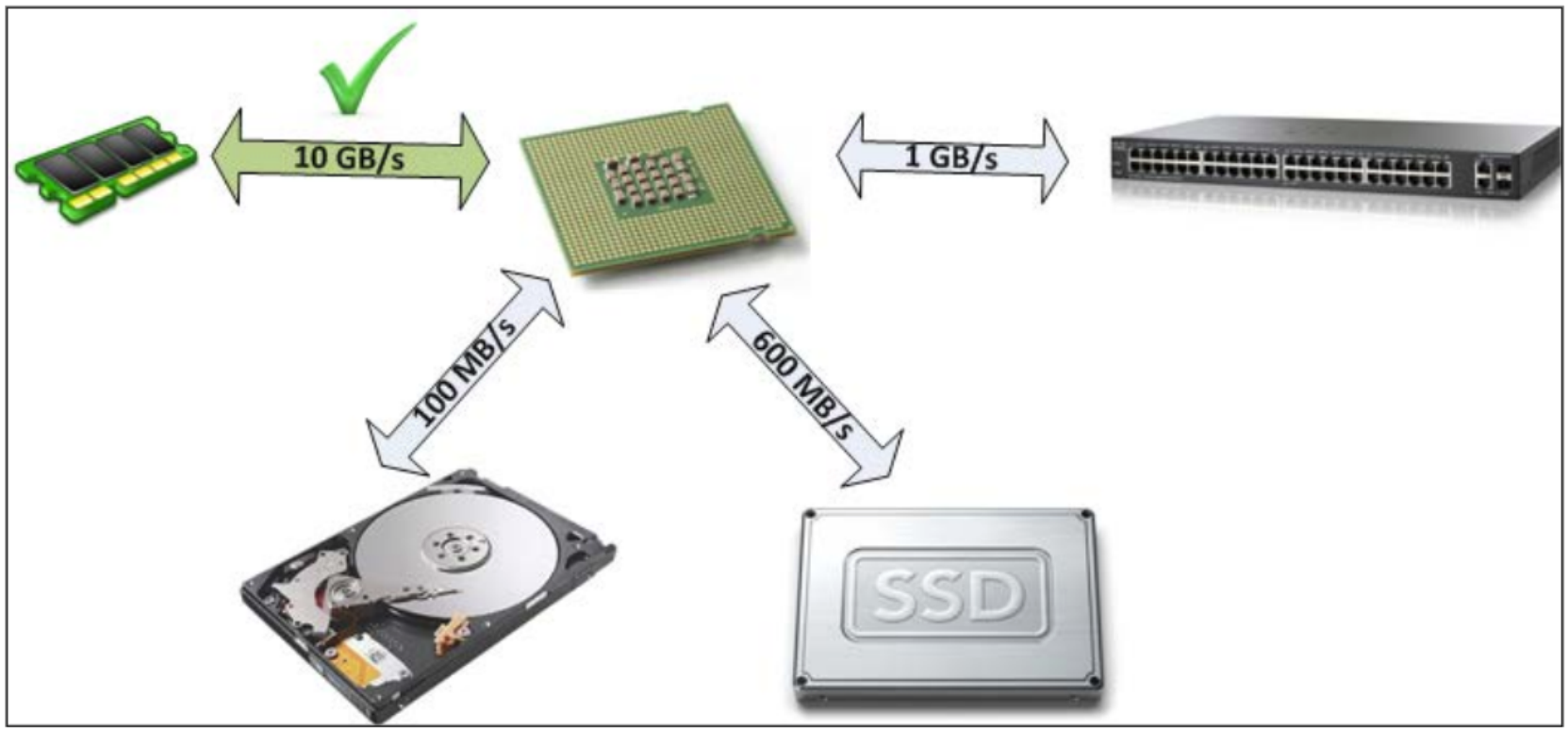
Apache Spark

Why

- Hadoop and MapReduce have been around for 10 years and proven to be best to process massive amounts of data with high performance.
- MR lacked performance in **iterative** computing: output between multiple MR jobs had to be dumped to disk (bottleneck)
- Focus on all-in-memory-compute.

Memory

Figure 2.6 indicates the data transfer rates from various mediums to the CPU. Disk to CPU is **100 MB/s**, SSD to CPU is **600 MB/s**, and over the network to CPU is **1 MB** to **1 GB/s**. However, RAM to CPU transfer speed is astonishingly fast, which is **10 GB/s**. So, the idea is to cache all or partial data in-memory so that higher performance can be achieved:



Spark

Why Spark?

- Designed to be interoperable with Hadoop
- Enables applications to distribute data reliably in-memory during processing. This is key to its performance and allows applications to avoid expensive disk access.
- Suitable for iterative algorithms
- Spark programs run up to 100x faster in-memory
- Provides native support for Java, Scala, Python and R
- Spark powers a stack of libraries, including Spark SQL, DataFrames (for interactive analytics), MLib (for machine learning) GraphX (for graph processing)
- Spark runs on Hadoop, Mesos, standalone cluster managers, on-premise hardware, or in the cloud

MR vs. Spark

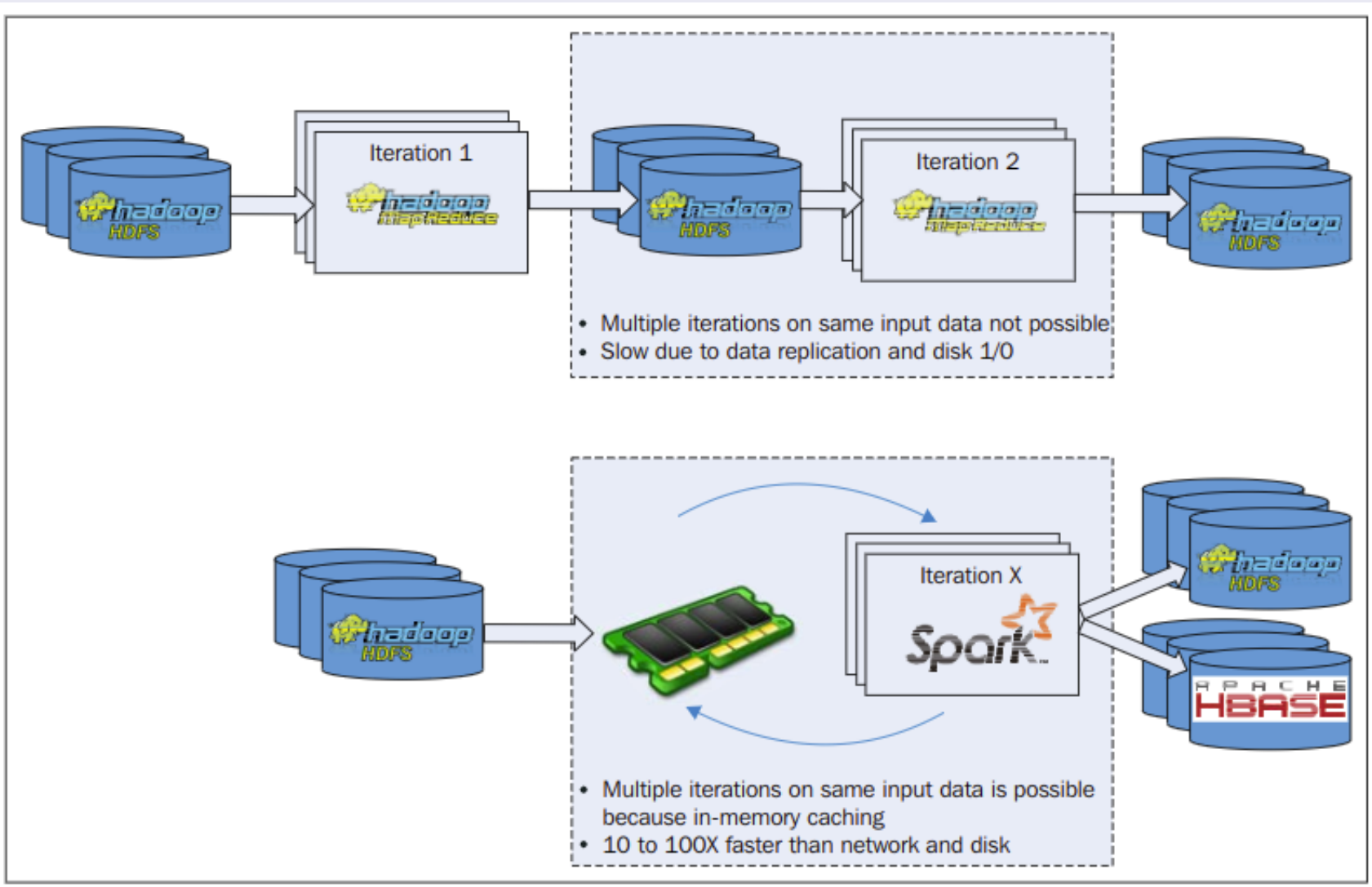


Figure 2.8: MapReduce versus Apache Spark