



Introduction to Apache Spark

Lightening fast cluster computing

Contents



Evolution of distributed systems

Need of new generation distributed system

Hardware/Software evolution in last decade

Apache Spark

Why Spark?

Who are using Spark?

Evolution of distributed systems

- First Generation
- Second Generation
- Third Generation

First Generation

- Proprietary
- Custom Hardware and Software
- Centralized data
- Hardware based fault recovery

Ex: Teradata, Netezza etc.

Second Generation

- Open source
- Commodity hardware
- Distributed data
- Software based fault recovery

Ex: Hadoop, HPCC

Why we need new generation?

- Lot has been changed from 2000
- Both Hardware and Software gone through changes
- Big data has become necessity now
- Let's look at what changed over decade

State of hardware in 2000

- Disk was cheap so disk was primary source of data
- Network was costly so data locality
- RAM was very costly
- Single core machines were dominant

State of hardware now

- RAM is the king
- RAM is the primary source of data and we use disk for fallback
- Network is speedier
- Multi core machines are commonplace

State of hardware now

- RAM is the king
- RAM is the primary source of data and we use disk for fallback
- Network is speedier
- Multi core machines are commonplace

Software in 2000

- Object orientation was the king
- Software optimized for single core
- No open frameworks for creating
 - Distributed storage
 - Distributed processing
- SQL was the only dominant way for data analysis

Software now

- Functional programming is on rise
- Software needs to exploit multiple cores on single node
- There are good frameworks to create distributed systems
 - HDFS for storage
 - Apache Mesos / YARN to create distributed processing
- NoSQL is real alternative now

Big Data processing needs in 2000

- Very few companies had big data issue
- Batch processing system ruled the world
- Volume was big concern compare to Velocity.
- Mostly used for
 - Search
 - Log analysis

Big Data processing needs now

- All companies use big data
- Velocity is as much concern as volume
- Needs of real time are as much important as batch processing.
- Use cases are not just limited to search

Shortcomings of Second generation

- Batch processing is primary objective
- Not designed to change depending upon use cases
- Tight coupling between API and the run time
- Do not exploit new hardware capabilities
- Too much complex

Third generation distributed systems

- Handle both batch processing and real time
- Exploit RAM as much as disk
- Multiple core aware
- Do not reinvent the wheel
- They use
 - HDFS for storage
 - Apache Mesos / YARN for distribution
- Plays well with Hadoop

Apache Spark

- A fast and general engine for large scale data processing
- Created by AMPLab now Databricks
- Written in Scala
- Licensed under Apache
- Lives in Github

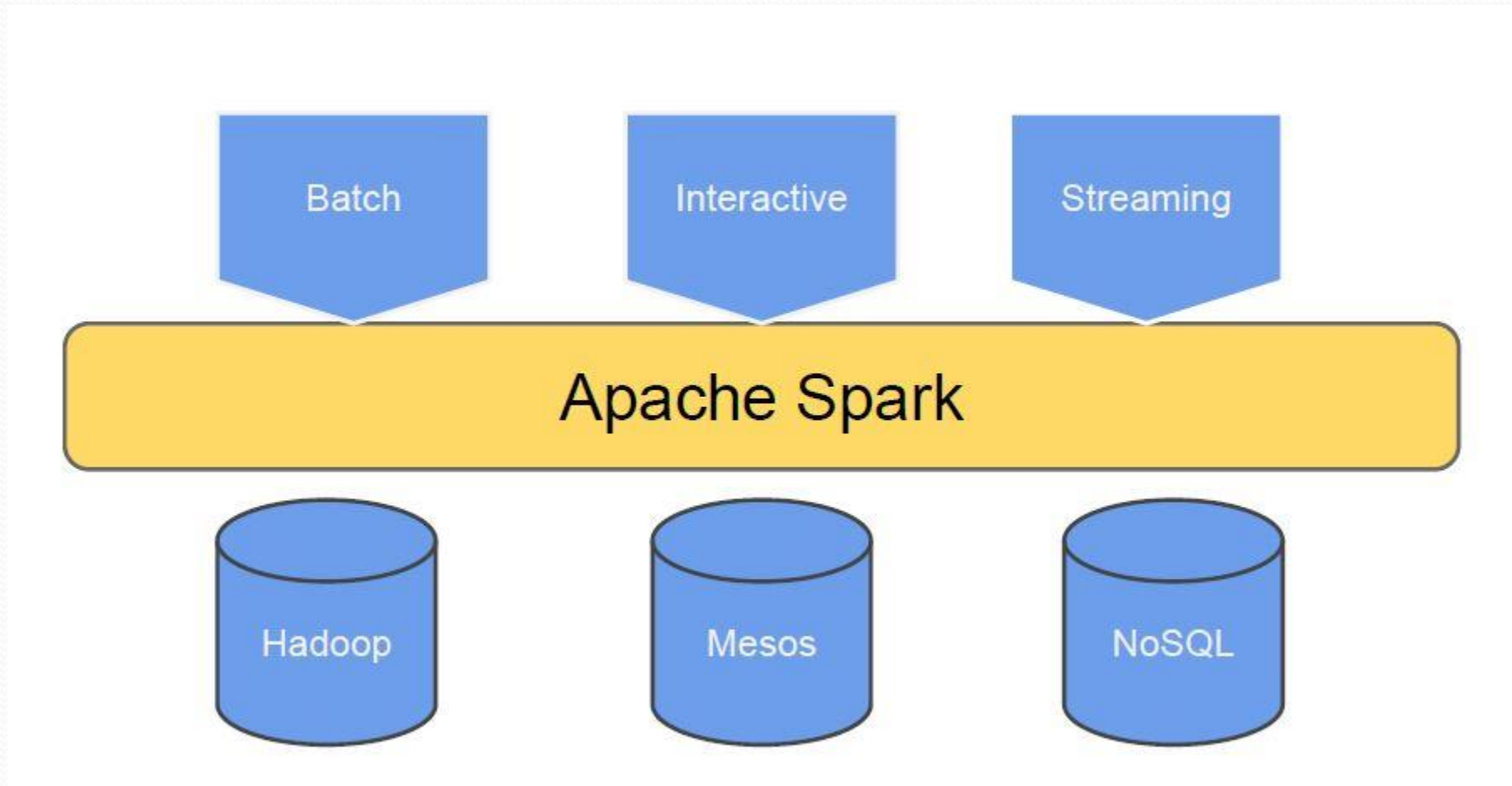
History of Apache Spark

- Mesos, a distributed system framework as class project in UC Berkeley in 2009.
- Spark to test how Mesos works
- Focused on
 - ❑ Iterative programs (ML)
 - ❑ Interactive querying
 - ❑ Unifying the real time and batch processing
- Open sourced in 2010



Why Spark?

Unified Platform for Big Data Ap



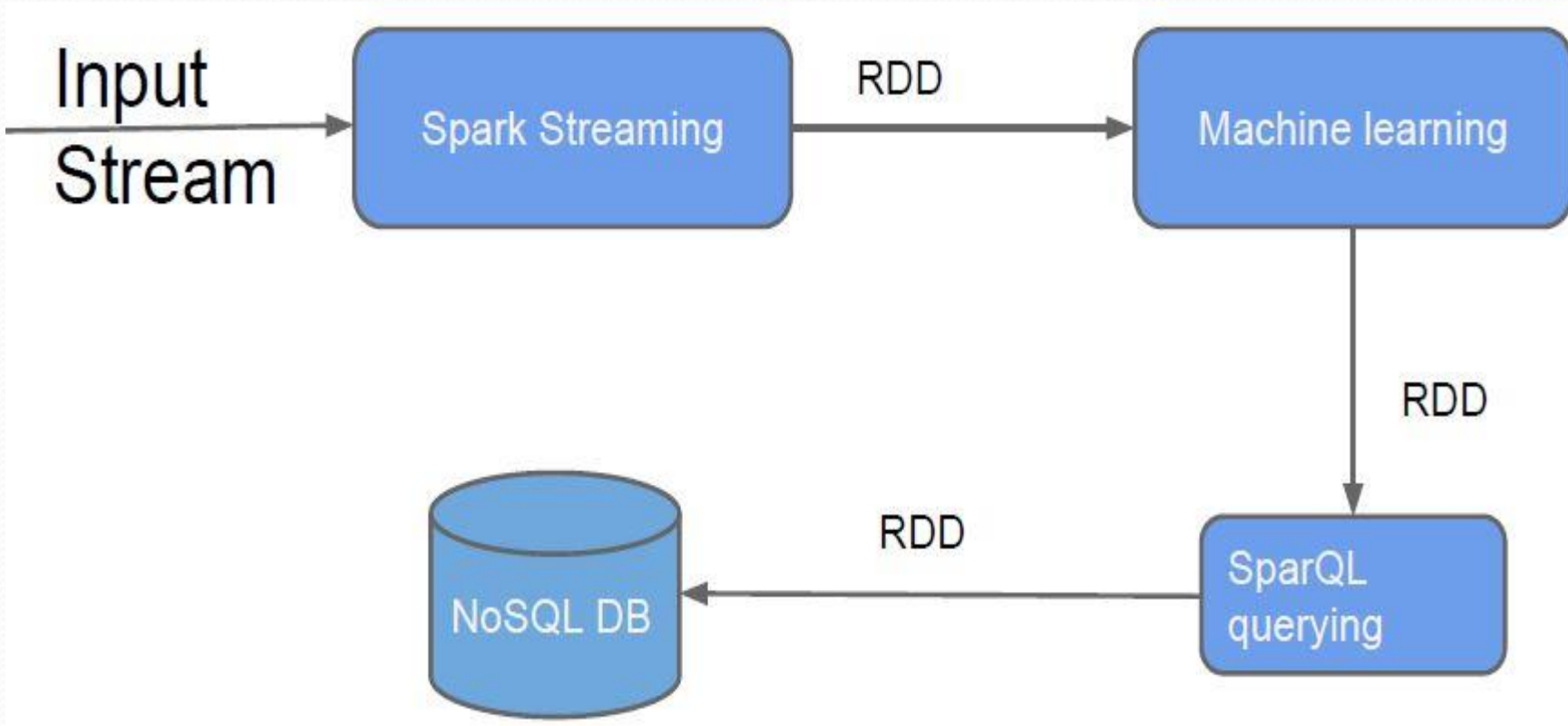
Why unification matters?

- Good for developers : One platform to learn
- Good for users : Take apps every where
- Good for distributors : More apps

Unification brings one abstraction

- All different processing systems in Spark share same abstraction called RDD.
- RDD is Resilient Distributed Dataset
- As they share same abstraction you can mix and match different kind of processing in same application

Spam detection



Boxes indicate different API calls not different processes

Apache Spark is not a one-trick pony : Going beyond in-memory processing

- One platform for all Big data loads
- One abstraction to rule them all
- Runs everywhere
 - ❑ You can run spark on top of any distributed system
 - ❑ It can run on
 - Hadoop 1.x
 - Hadoop 2.x
 - Apache Mesos
 - Its own cluster
 - ❑ Its just a user space library

Apache Spark is not a one-trick pony : Going beyond in-memory processing

- Small and Simple
 - Apache Spark is highly modular
 - The original version contained only 1600 lines of scala code
 - Apache Spark API is extremely simple compared to Java API of M/R
 - API is concise and consistent

Apache Spark is not a one-trick pony : Going beyond in-memory processing

- Prospering Ecosystem

Hadoop	Spark
Hive	SparkSQL
Apache Mahout	MLLib
Impala	SparkSQL
Apache Giraph	GraphX
Apache Storm	Spark Streaming

Apache Spark is not a one-trick pony : Going beyond in-memory processing

- Multi language API
 - ❑ Written in Scala but API is not limited to it
 - ❑ Offers API in
 - Scala
 - Java
 - Python
 - ❑ You can also do SQL using SparkSQL

In-memory aka Speed

- In Spark, you can cache hdfs data in main memory of worker node
- Spark analysis can be executed directly on in memory data
- Shuffling also can be done from in memory
- Fault tolerant

Integration with Hadoop

- No separate storage layer
- Integrates well with HDFS
- Can run on Hadoop 1.0 and Hadoop 2.0 YARN
- Excellent integration with ecosystem projects like Apache Hive, HBase etc.

Who are using

