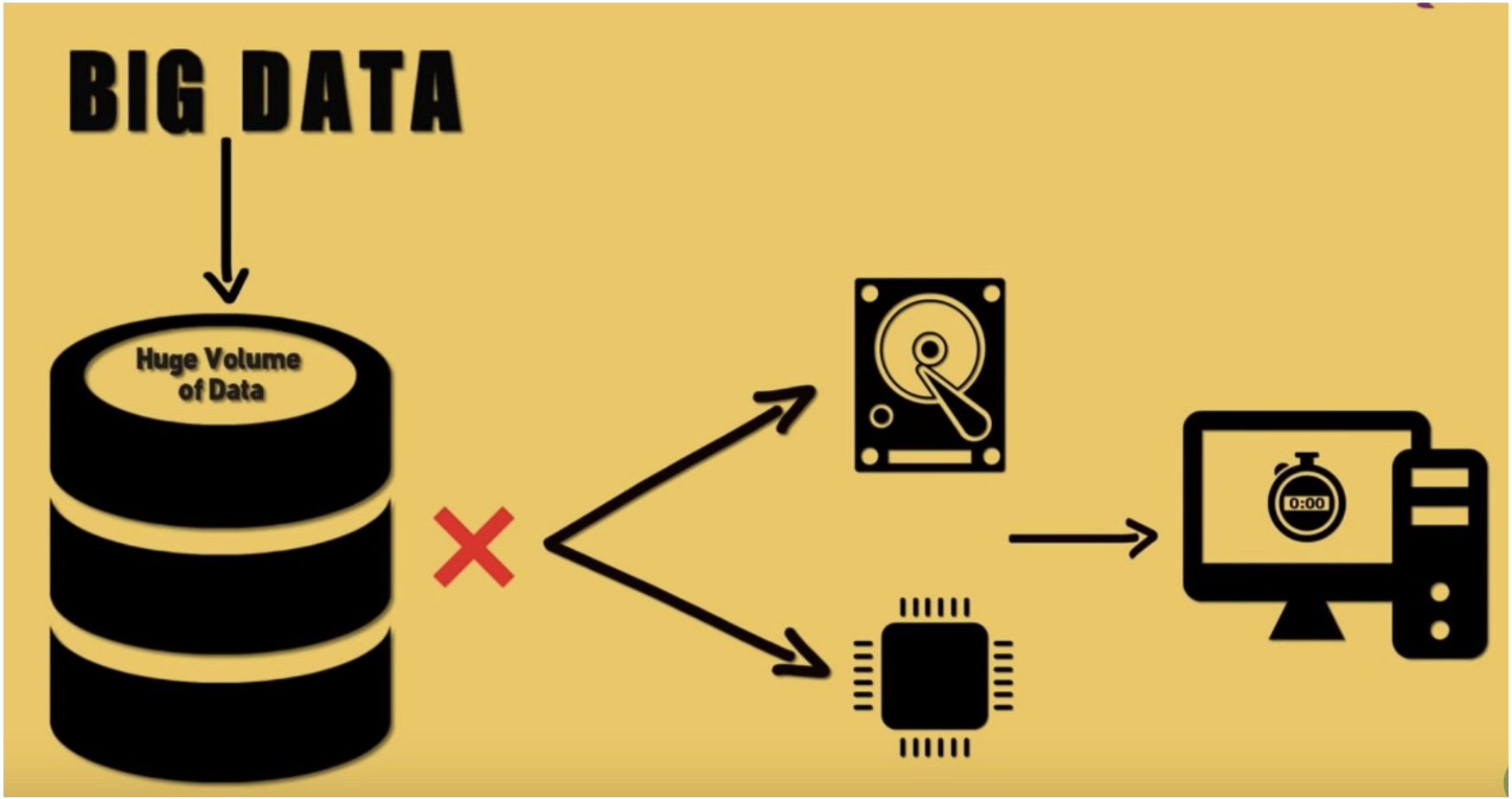
Big data & PySpark

Agenda

- Introduction to Big data
- Hadoop- MapReduce
- Pros & Cons
- Apache Spark overview
- Python code implementation


BIG DATA
Huge Volume of Data
0:00

How huge is the big data?

- Volume
- Variety
- Velocity

Big data & Hadoop

Traditional Scenario:

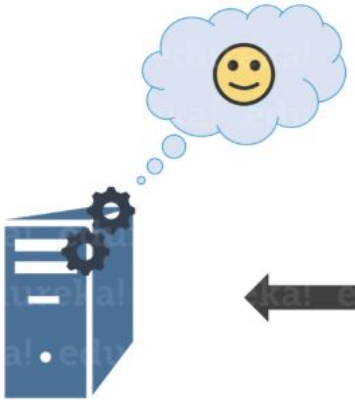
2 orders per hour

Single Cook

Food Shelf

Traditional Scenario:

Data is generated at a steady rate and is structured in nature



Traditional Processing System

RDBMS

Scenario 2:

- They started taking Online orders
- 10 orders per hour

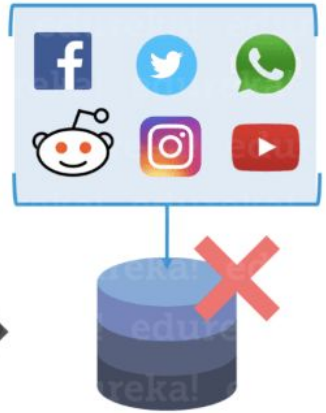
Single Cook
(Regular Computing System)

Food Shelf
(Data)

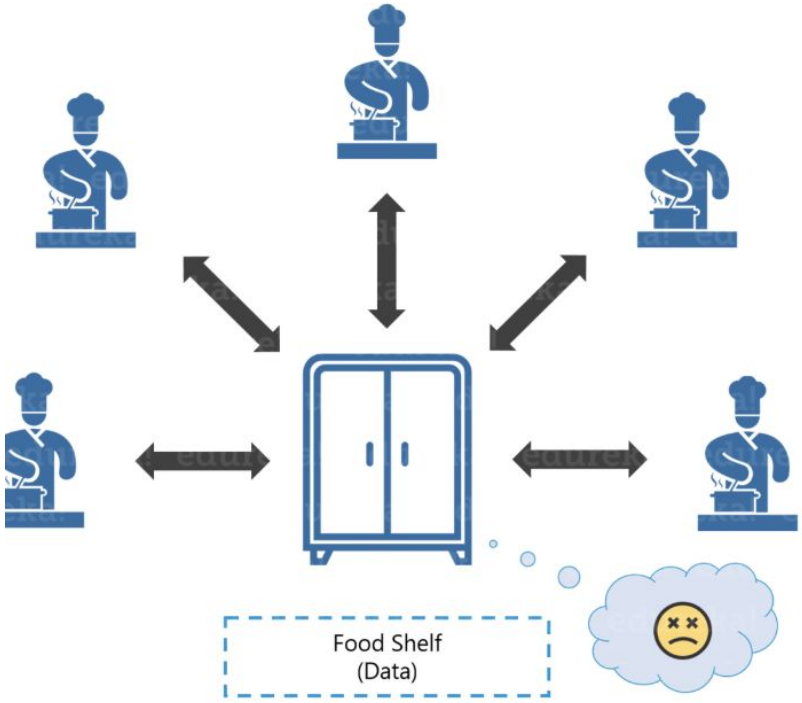
Big Data Scenario:

Heterogenous data is being generated at an alarming rate by multiple sources

Traditional Processing System



RDBMS

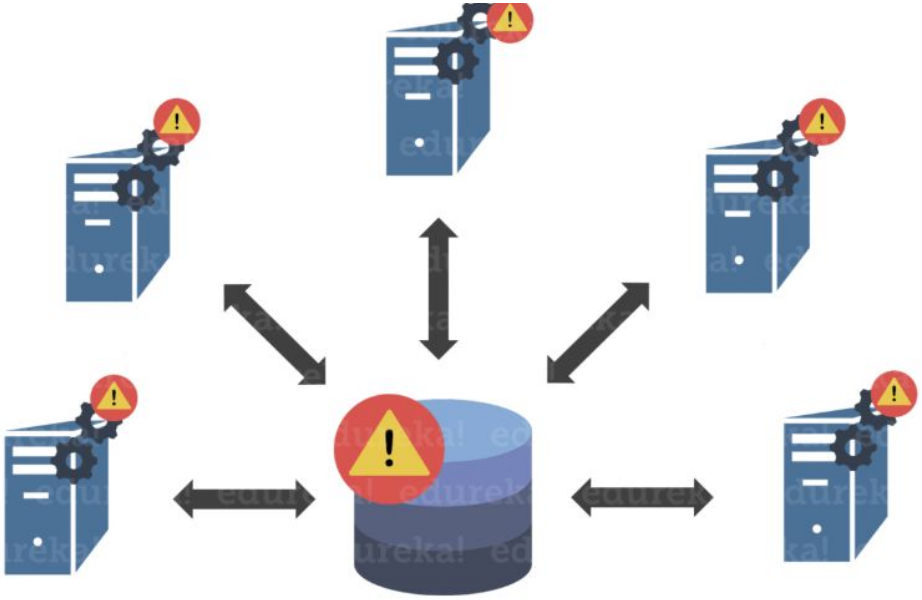


Scenario:

Multiple Cook cooking food

Issue:

Food Shelf becomes the BOTTLENECK



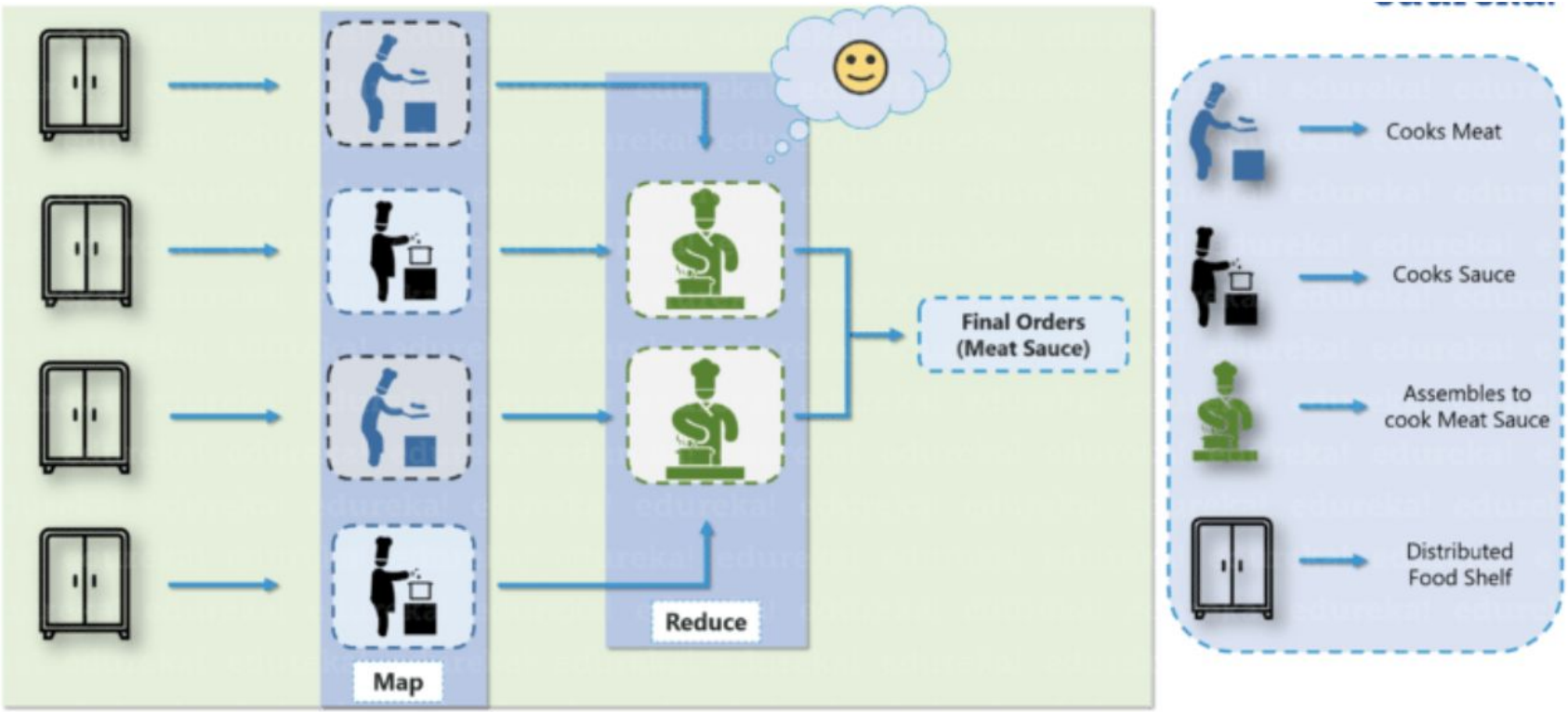
Data Warehouse

Scenario:

Multiple Processing Unit for data processing

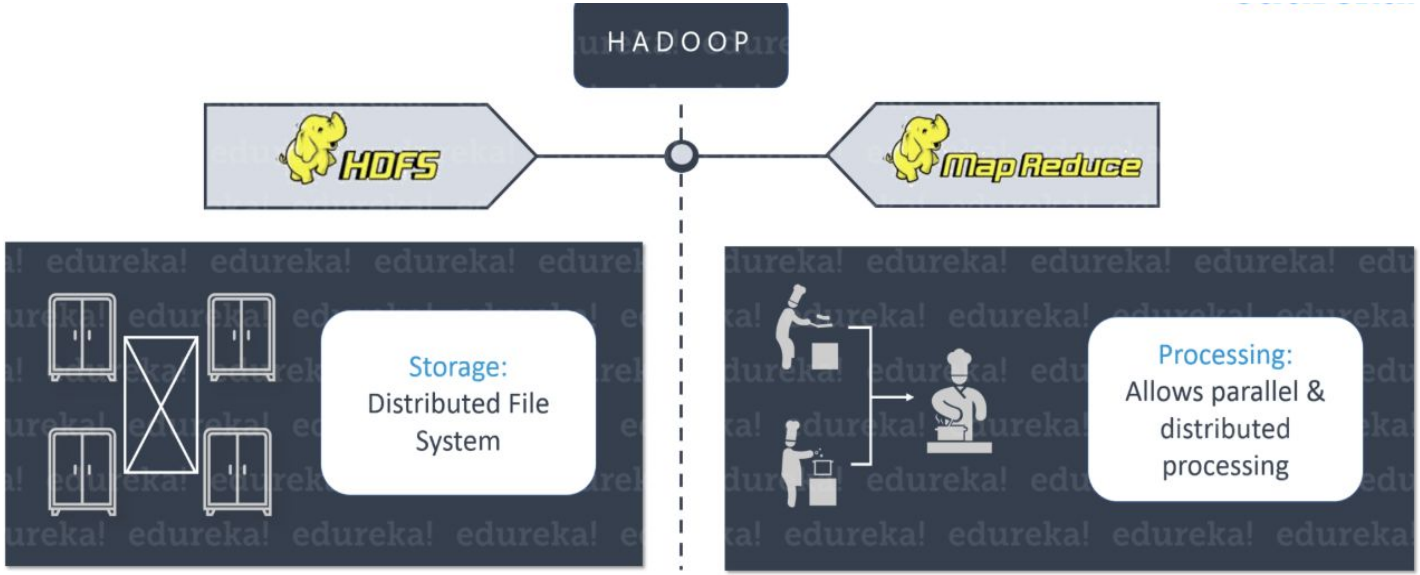
Issue:

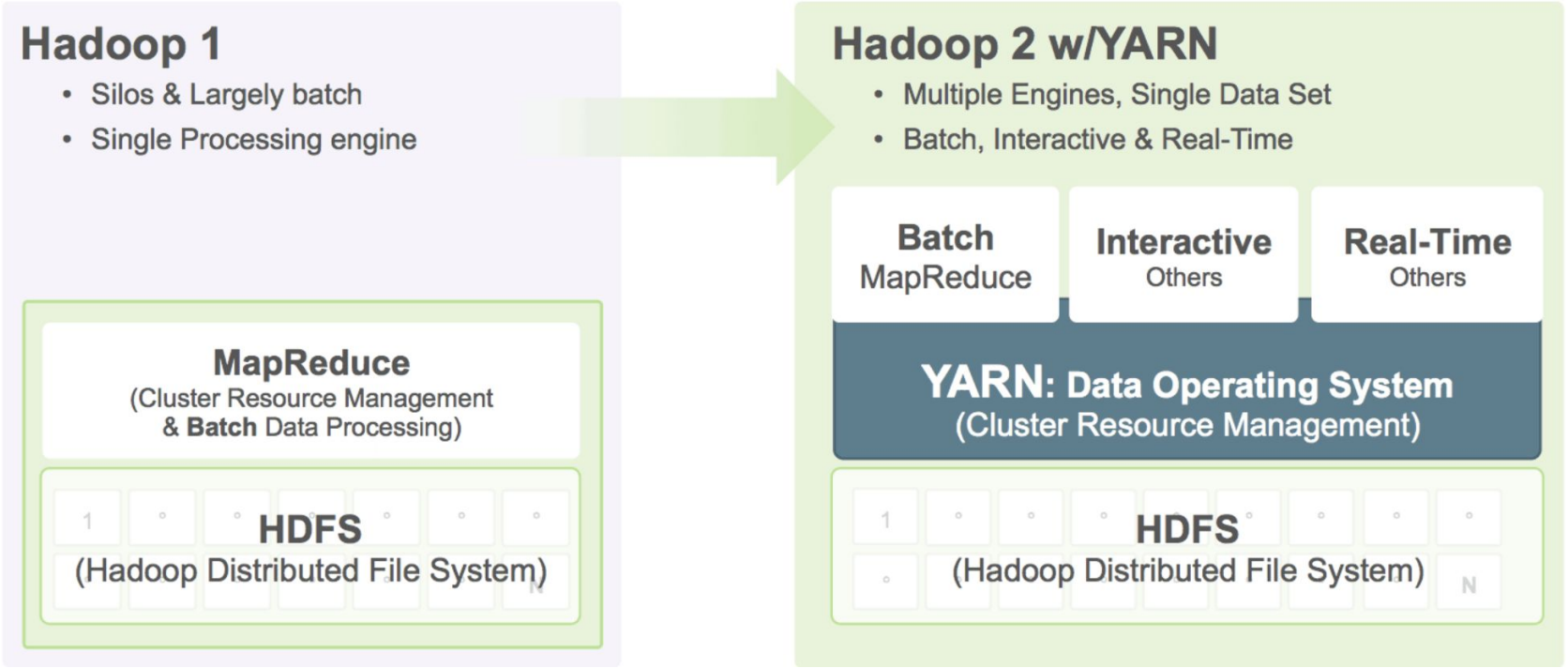
Bringing data to processing generated lot of Network overhead


Map
Reduce
Final Orders
(Meat Sauce)
Cooks Meat
Cooks Sauce
Assembles to
cook Meat Sauce
Distributed
Food Shelf

Hadoop

Hadoop is an open-source software framework used for storing and processing Big Data in a distributed manner on large clusters of commodity hardware



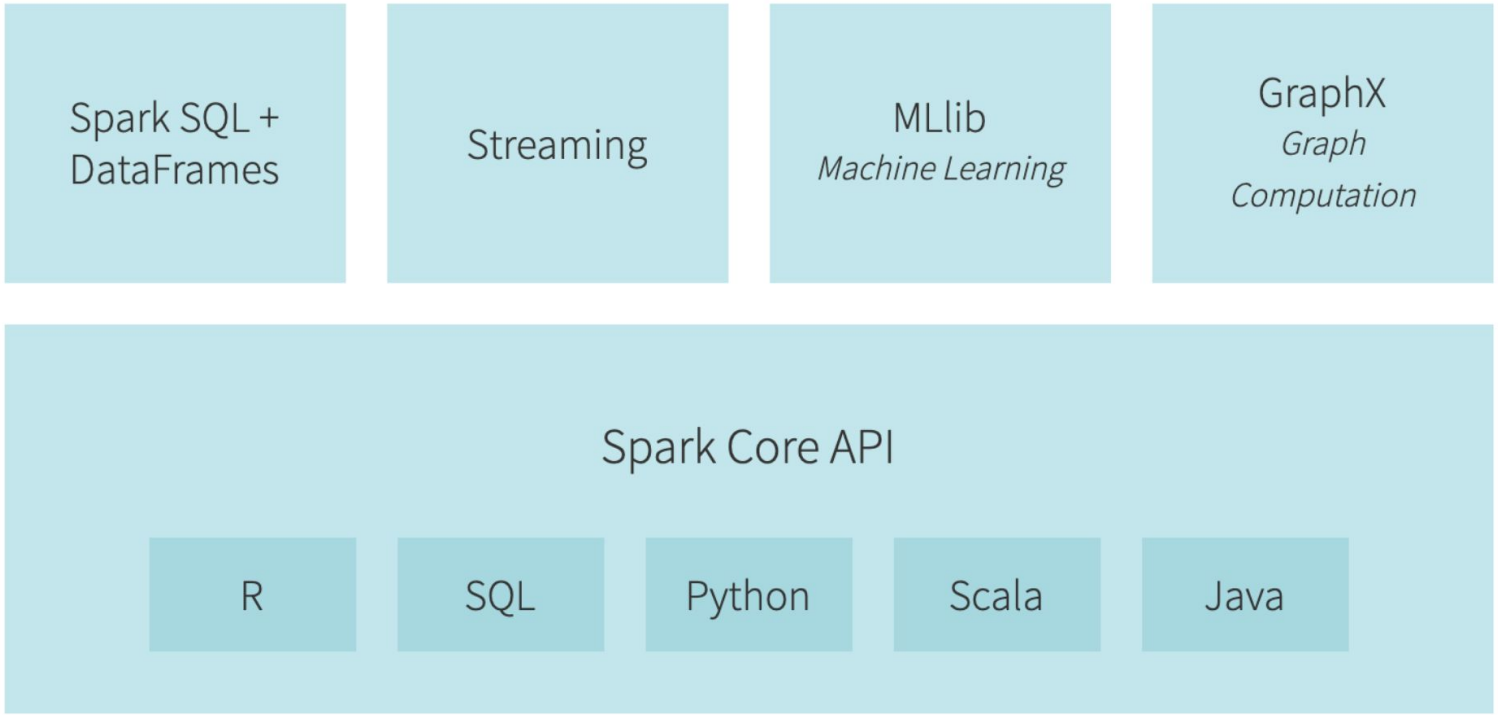

Hadoop 1
• Silos & Largely batch
• Single Processing engine
MapReduce
(Cluster Resource Management
& Batch Data Processing)
HDFS
(Hadoop Distributed File System)
Hadoop 2 w/YARN
• Multiple Engines, Single Data Set
• Batch, Interactive & Real-Time
Batch
MapReduce
Interactive
Others
Real-Time
Others
YARN: Data Operating System
(Cluster Resource Management)
HDFS
(Hadoop Distributed File System)

Spark Overview

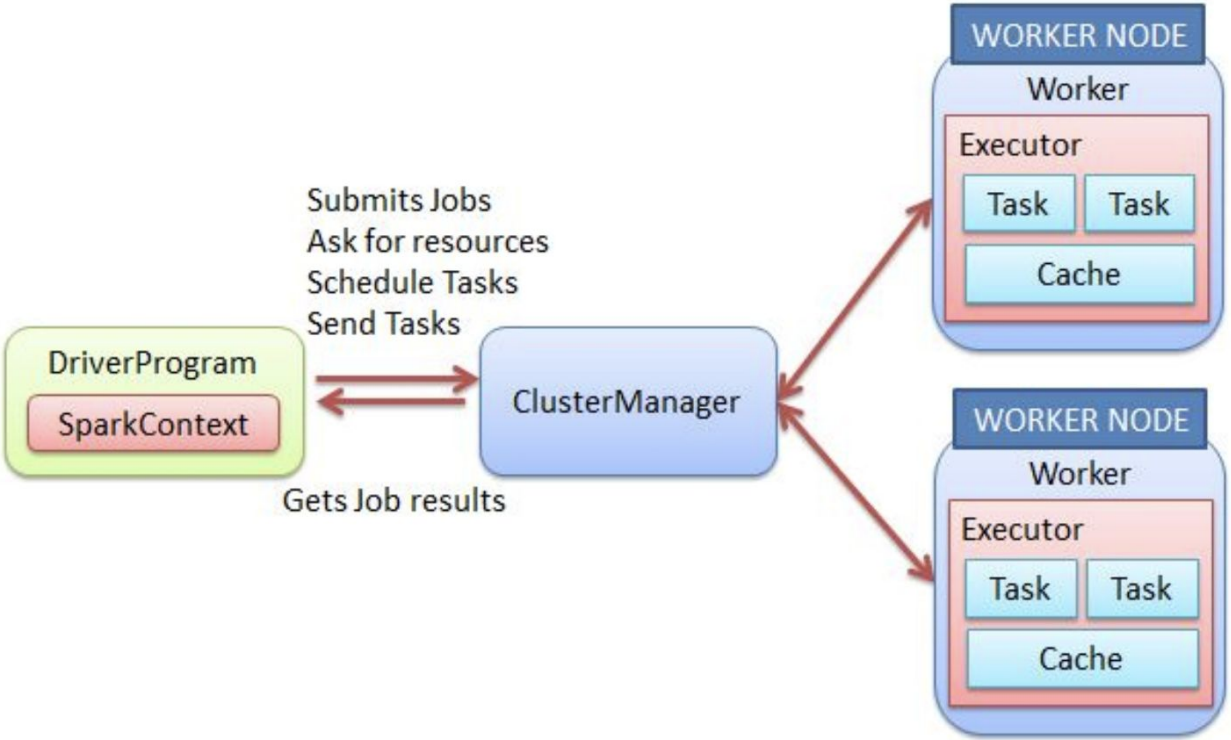
- Apache Spark is a fast, general-purpose engine for large-scale data processing.
- Spark enables in-memory distributed data sets. Data sets are cached in memory to reduce latency of access:
- 100 times faster than MapReduce in memory
- 10 times faster on disk

The Spark framework can be deployed on its own cluster (standalone) or on Apache Hadoop via YARN.

Spark ecosystem



How does it work?



Why use spark ? And why not Hadoop?

- Mapreduce does only Batch-processing
- Spark solves general purpose cluster computing systems (Real time data and Batch processing)

Applications

Twitter Sentiment Analysis With Spark

Trending Topics can be used to create campaigns and attract larger audience

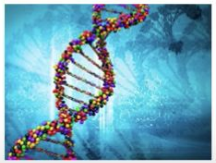
Sentiment helps in crisis management, service adjusting and target marketing

NYSE ICE

NYSE: Real Time Analysis of Stock Market Data

Banking: Credit Card Fraud Detection

Genomic Sequencing



Spark objects

Distribute collection of JVM objects

Functional Operators (map, filter, etc.)

Distribute collection of Row objects

Expression-based operations and UDFs

Logical plans and optimizer

Fast/efficient internal representations

Internally rows, externally JVM objects

Almost the "Best of both worlds": **type safe + fast**

But slower than DF
Not as good for interactive analysis, especially Python

When to use RDD

- you want low-level transformation and actions and control on your dataset;
- your data is unstructured, such as media streams or streams of text;
- you want to manipulate your data with functional programming constructs than domain specific expressions;
- you don’t care about imposing a schema, such as columnar format, while processing or accessing data attributes by name or column

When to use Dataframe

- you want rich semantics, high-level abstractions, and domain specific APIs, use DataFrame
- your processing demands high-level expressions, filters, maps, aggregation, averages, sum, SQL queries, columnar access and use of lambda functions on semi-structured data, use DataFrame
- you want higher degree of type-safety at compile time, want typed JVM objects, take advantage of Catalyst optimization, and benefit from Tungsten's efficient code generation, use Dataset.
- you want unification and simplification of APIs across Spark Libraries, use DataFrame or Dataset.
- If you are a R user, use DataFrames.
- If you are a Python user, use DataFrames and resort back to RDDs if you need more control.

You have an RDD of data that you wish to use to build a predictive model. Should you leave it as an RDD or transform it to a DataFrame?

Thank You!