ISIT312 Big Data Management

Introduction to Spark

Dr Fenghui Ren

School of Computing and Information Technology -
University of Wollongong

Introduction to Spark

Outline

MapReduce Challenges

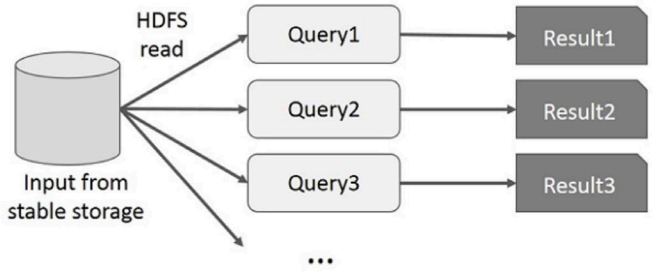
Google introduced MapReduce, which was a breakthrough in the history of big data technologies

- It makes the processing of big data feasible and practical

However, Hadoop MapReduce framework releases the developer from the distributing computing trickiness, its Java API is still too low-level

- Think how would you implement an inner join in MapReduce ?
- Hive, Pig and other frameworks based on MapReduce can help

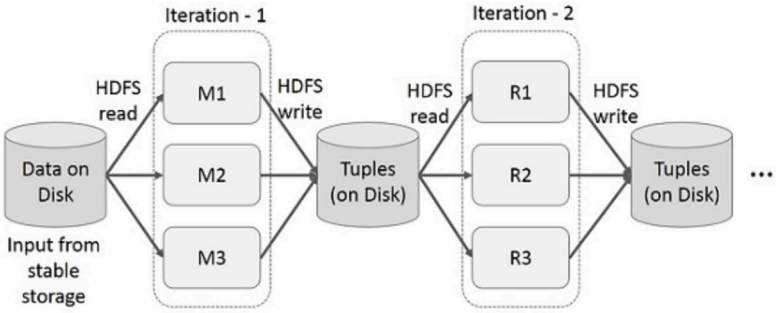
More importantly, the persistent storage I/O makes the interactive and iterative computation (two important forms of data processing) very inefficient and leads to to potential I/O latency



MapReduce Challenges

Data Sharing in Hadoop

- In the Hadoop MapReduce framework, the reuse data between computations (e.g., between two MapReduce jobs) is to write it to an external stable storage system, for example HDFS)
- Although Hadoop provides numerous abstractions for data access, data sharing is slow due to replication, serialisation and persistent storage IO
- More than 90% of the time for running a MapReduce job is doing HDFS read-write operations
- Substantial overhead is caused by data replication, persistent storage I/O, and serialization



Introduction to Spark

Outline

Meet Spark !

Apache Spark, as a cluster computing platform, is designed to be fast and general-purpose

On the speed, Spark extends the MapReduce model to efficient support more types of computations, for example, interactive and iterative computations

- This is mainly due to Spark's in-memory computing, although Spark is also more efficient than MapReduce for persistent storage complex applications

On the generality, Spark covers a wide ranges of workloads, including batch applications, iterative queries, streaming and advanced analytics

- It provides a unified environment makes the combination of those different engines easy and inexpensive
- It provides multiple language APIs, including Scala, Python, Java, SQL, and R, as well as a very rich (yet fast-growing) built-in libraries

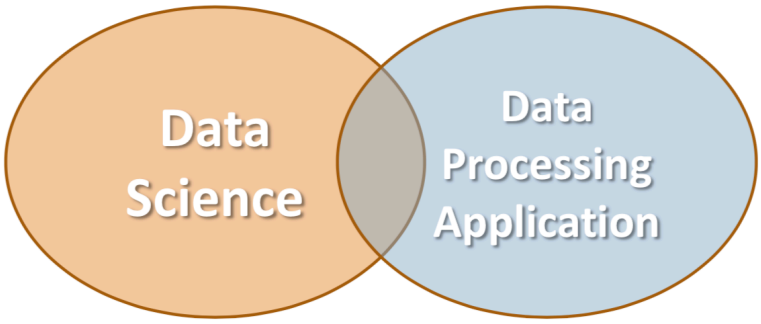
Meet Spark !

Data Scientists use Spark to:

- Analyse and model data
- Make prediction based on data
- Build data pipelines to fulfil certain tasks

Data Engineers use Spark to:

- Develop data processing applications
- Deploy the output of data scientists in production



Introduction to Spark

Outline

Features of Spark

As mentioned, Spark supports not only batch computing (as MapReduce does) but also interactive, iterative, and real-time computing

The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application

The programming model of Spark is based on Directed Acyclic Graphs (DAG) and it is more flexible than the MapReduce model

Unlike MapReduce, Spark has built-in high-level APIs to process structured data, e.g., tables in Hive

Spark reduces the management burden of maintaining separate tools

Features of Spark

Speed
Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk

- This is achieved by reducing number of read/write operations to persistent storage
- It stores the intermediate processing data in memory

Support for multiple languages
Spark provides built-in APIs in Java, Scala, or Python

Advanced Analytics
Spark not only supports MapReduce It also supports SQL queries, Streaming data, Machine Learning (ML), and Graph algorithms

Introduction to Spark

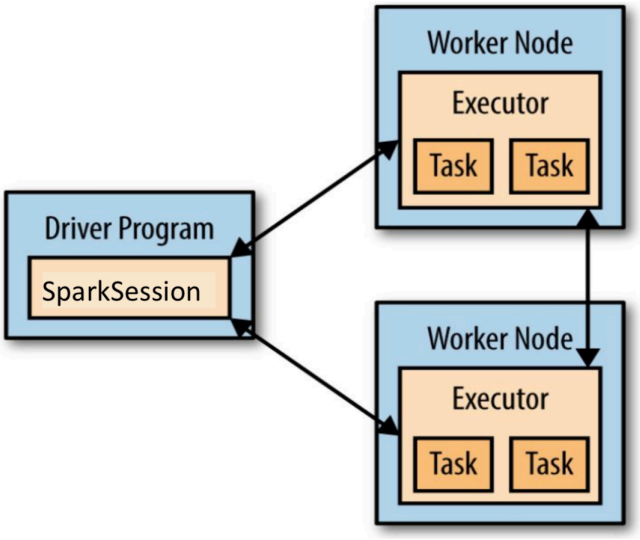
Outline

Spark Architecture

Driver program is the main Spark application that consists of the data processing logic

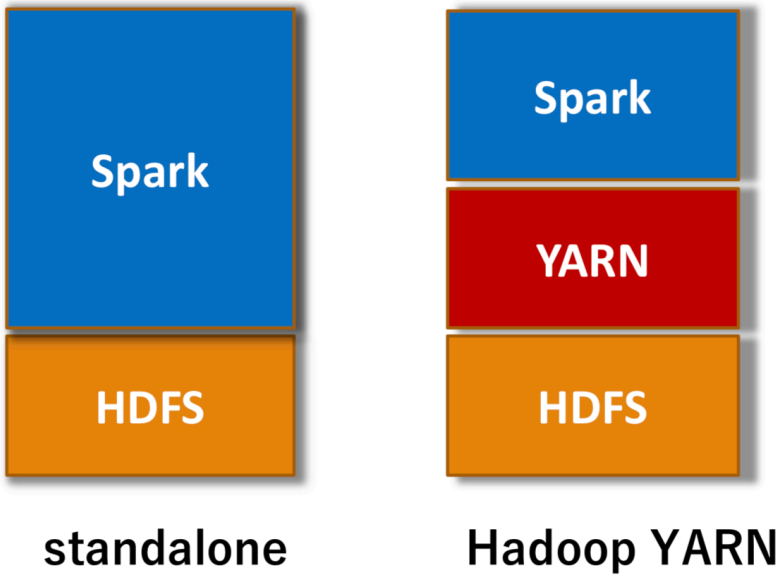
Executor is a JVM process that runs on each worker node and processes the jobs that the driver program submits

Task is a subcomponent of a data processing job



Spark Architecture

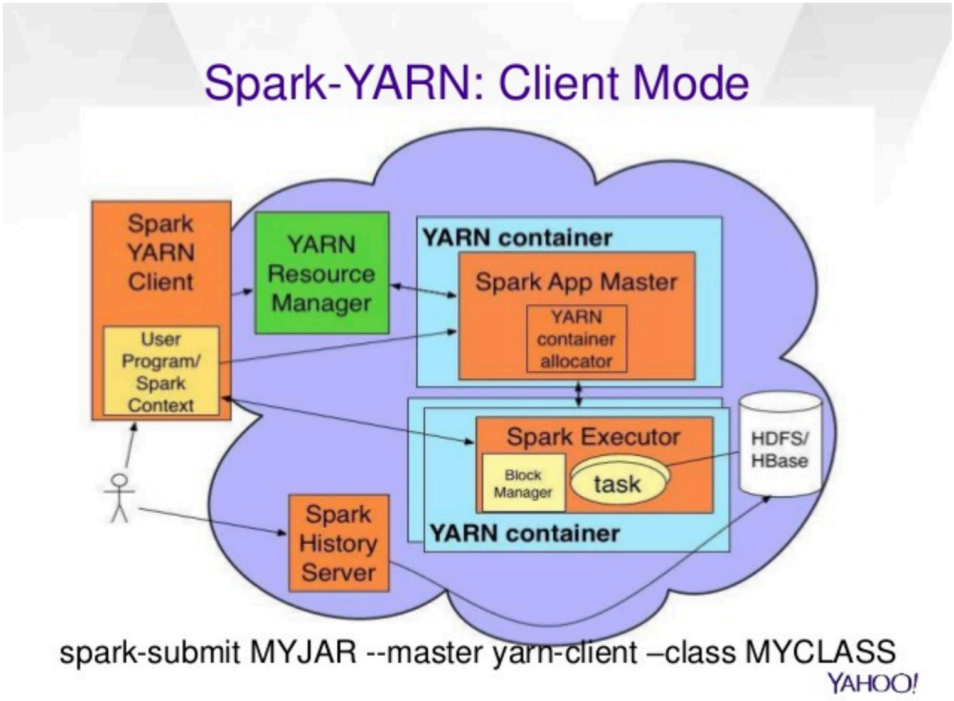
Two modes of Building Spark on Hadoop



These modes are different from full-distributed mode and pseudo-distributed mode

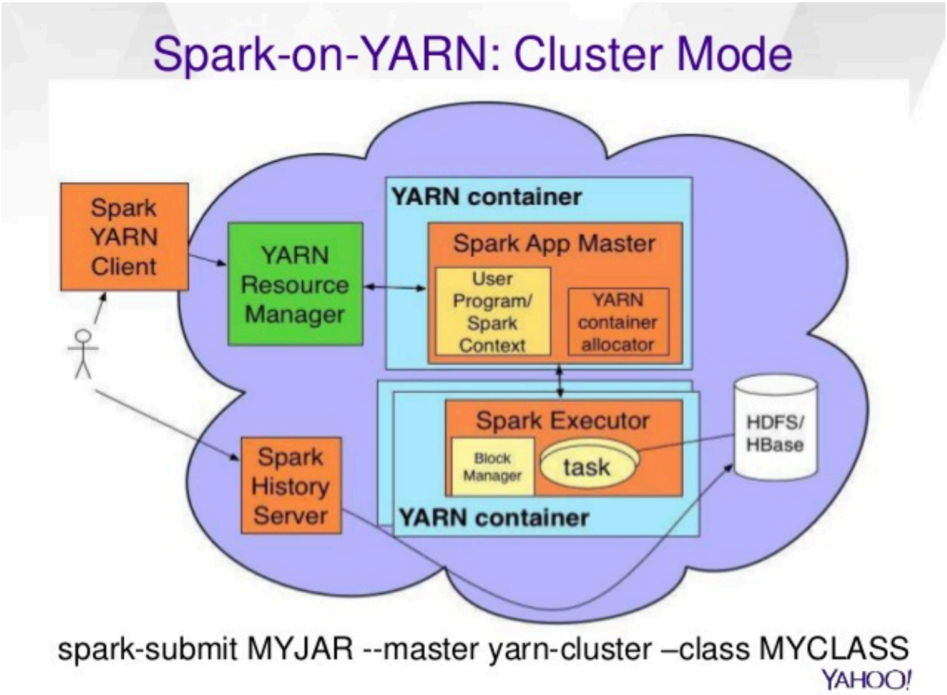
Spark Architecture

Spark on YARN: Client-Mode



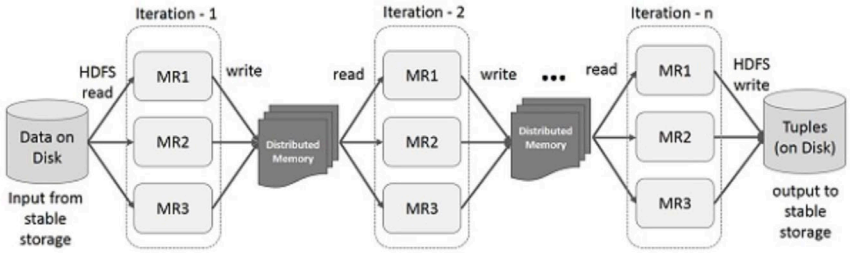
Spark Architecture

Spark on YARN: Cluster-Mode

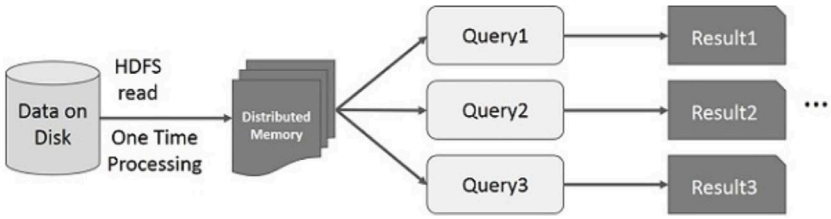


Spark Architecture

Writing to and reading from memory in Spark reduces I/O overhead



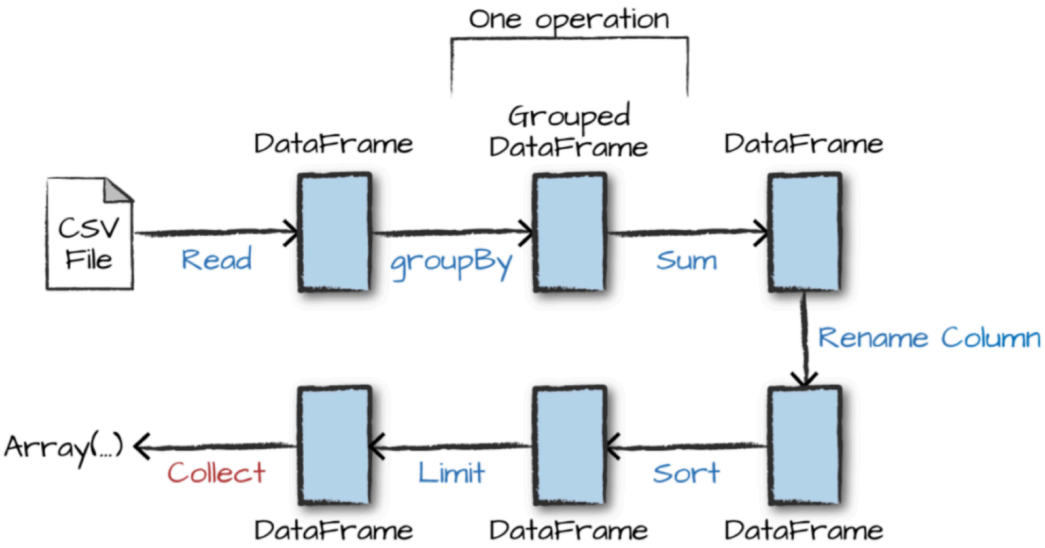
Data persists in memory and enables fast access



Spark Architecture

High-level API of Spark eases the development of a data processing pipeline

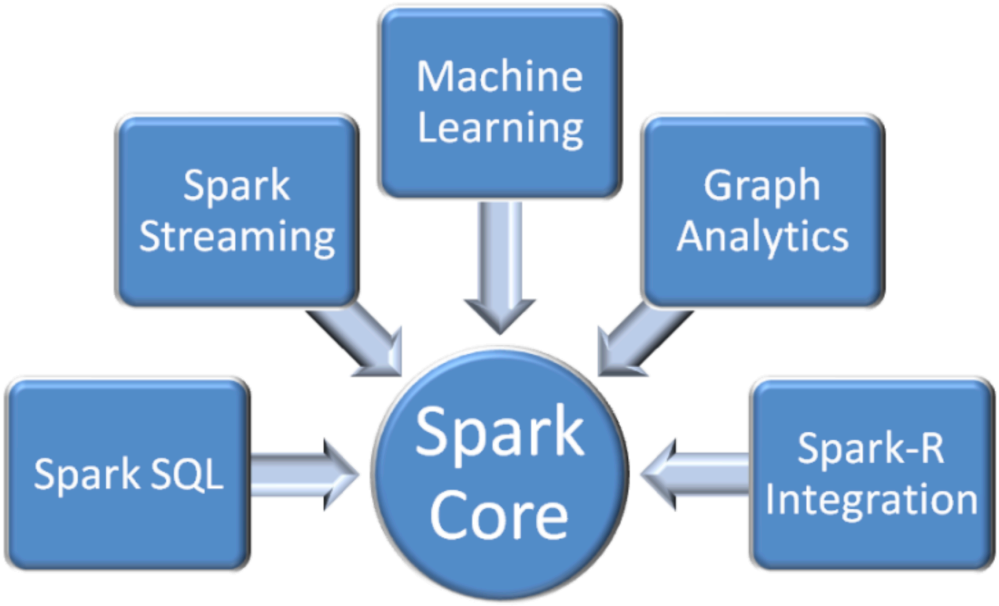
- No need to confirm to a specific pattern, for example MapReduce



- Under the hood, Spark determines the best logical plan and translate it to a physical plan

Spark Architecture

Components of Spark



Introduction to Spark

Outline

Overview of Spark Components

Spark Core

- Spark Core is the underlying general execution engine for Spark platform that all other functionality is built upon
- It provides In-Memory computing and referencing datasets in external storage systems

Spark SQL

- Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data

Spark Streaming

- Spark Streaming leverages Spark Core fast scheduling capability to perform streaming analytics

Overview of Spark Components

SparkML

- SparkML is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture

GraphX

- GraphX is a distributed graph-processing framework on top of Spark
- It provides an API for expressing graph computation and provides an optimized runtime for this abstraction

SparkR

- SparkR is an R package that provides a light-weight frontend to use Spark from R

Introduction to Spark

Outline

Spark Glossary

Term	Meaning
Application	User program built on Spark. Consists of a *driver program* and *executors* on the cluster.
Application jar	A jar containing the user's Spark application. Usually jar not include Hadoop/Spark libraries, which are added at runtime.
Driver program	The process running the main() function of the application and creating the SparkContext
Cluster manager	An external service for acquiring resources on the cluster (e.g. standalone manager, YARN, Mesos)
Worker node	Any node that can run application code in the cluster

Spark Glossary

Term	Meaning
Deploy mode	Distinguishes where the driver process runs. In "cluster" mode, the framework launches the driver inside of the cluster. In "client" mode, the submitter launches the driver outside of the cluster.
Executor	A process launched for an application on a worker node, that runs tasks and keeps data in memory or disk storage across them. Each application has its own executors.
Task	A unit of work that will be sent to one executor
Job	A parallel computation consisting of multiple tasks that gets spawned in response to a Spark *action* (e.g. save, collect)
Stage	A subset of tasks in a job that depend on each other (similar to the map and reduce stages in MapReduce)

References

A Gentle Introduction to Spark, Databricks, (Available in `READINGS` folder)

Spark Overview

Karau H., Fast data processing with Spark Packt Publishing, 2013 (Available from UOW Library)

Srinivasa, K.G., Guide to High Performance Distributed Computing: Case Studies with Hadoop, Scalding and SparkSpringer, 2015 (Available from UOW Library)

Chambers B., Zaharia M.,Spark: The Definitive Guide, O'Reilly 2017