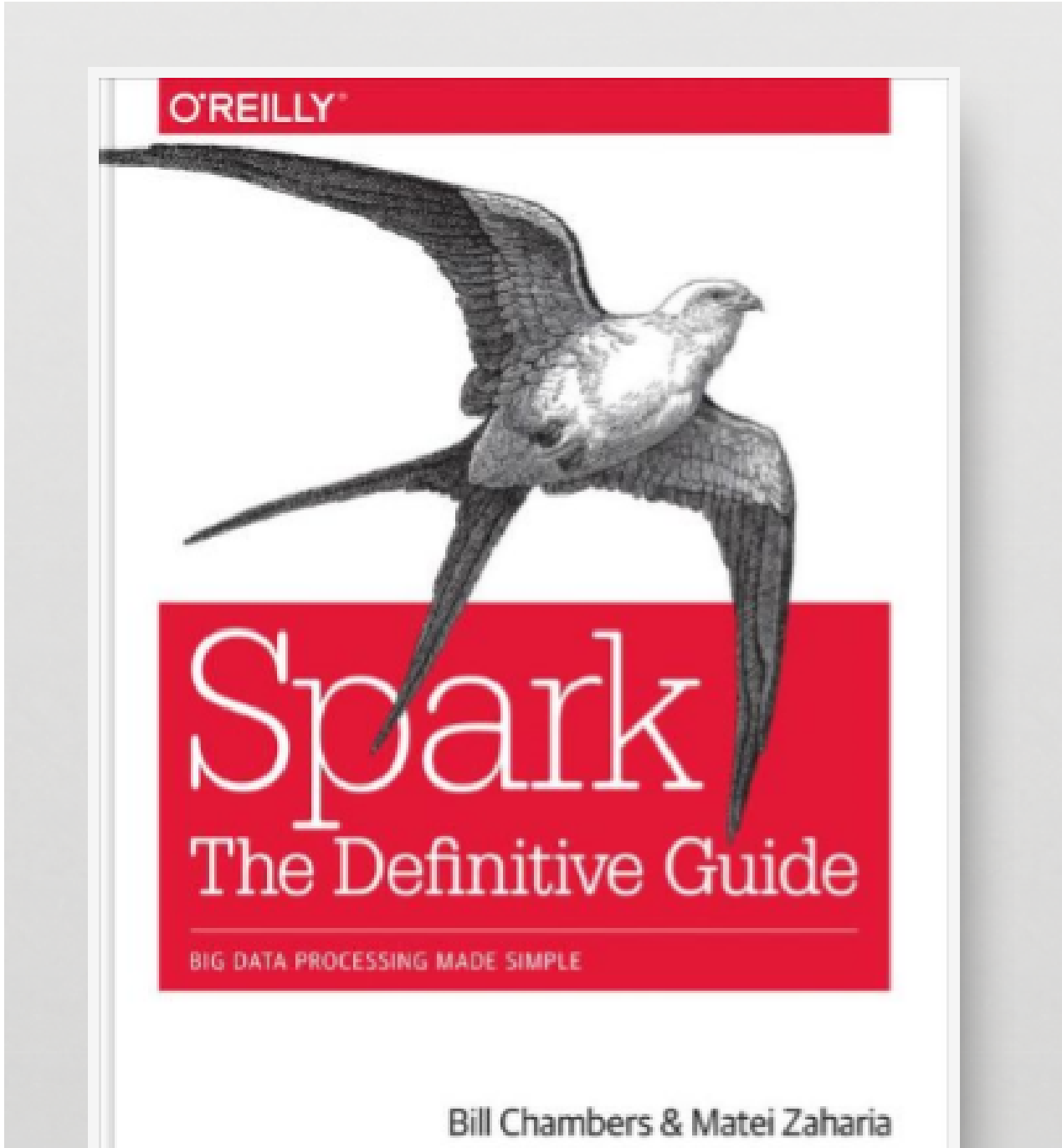
Spark the Definitive Guide 2nd Edition

Chapter 02

A Gentle Introduction to Spark

A Gentle Overview

Text Book


O'REILLY
Spark
The Definitive Guide
BIG DATA PROCESSING MADE SIMPLE
Bill Chambers & Matei Zaharia

Spark's Basic Architecture 22

- Single Computers work pretty well
- Powerful
- But only one machine
- This limits what can be done
- Single machines don't have the necessary power or the parallel ability
- Multiple computers alone are not enough – you need a framework to control the data
  - To schedule data movement and data processing

Spark Cluster Manager

- Spark has its won software based cluster manager.
- Configurable out of the box
  - Simple config file denoting if the node is a slave or master
- Spark can also use existing cluster managers:
  - YARN from Hadoop 2.x/3.x
- Mesos
  - Cluster scheduler created by Twitter
  - Still in use, we won't focus on Mesos in this class
- We will work initially with the built in Spark cluster manager
- YARN later in the semester when we move to cluster work

Core Architecture

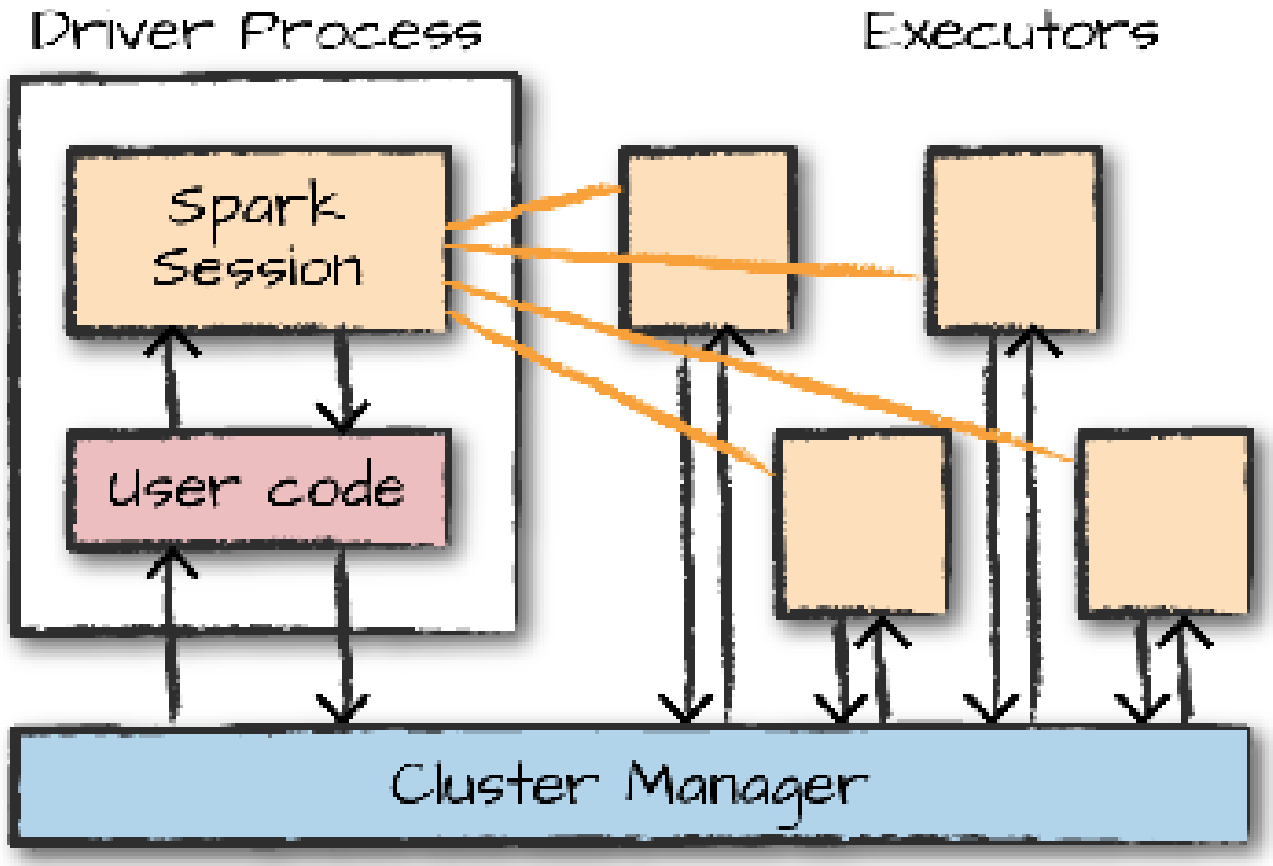


Figure 2-1. The architecture of a Spark Application

Figure 2: *Spark Core Architecture*

Spark Applications

- What makes up a Spark application?
  - Magic
- It is two things
  - A single **driver process** (like a main process in Java or Python)
  - A **set** of *executor processes*

More Application

- A Driver runs the Spark Applications main() function
- This process sits on a node in the cluster
  - Remember Spark is always assumed to be an 2+ node cluster with an additional master node
- The Main function does 3 things:
  - Maintain information about the running process
  - Respond a user's program or input
  - Analyzing, distributing, and scheduling work across the executor processes
- Driver process is essential to the running of the application (can't crash!)

Executors

- Responsible for carrying out the work that the Driver assigns them
- Executor then is responsible for two things:
  - Executing the code assigned by the Driver
  - Reporting the state of the execution back to the driver node

Architecture

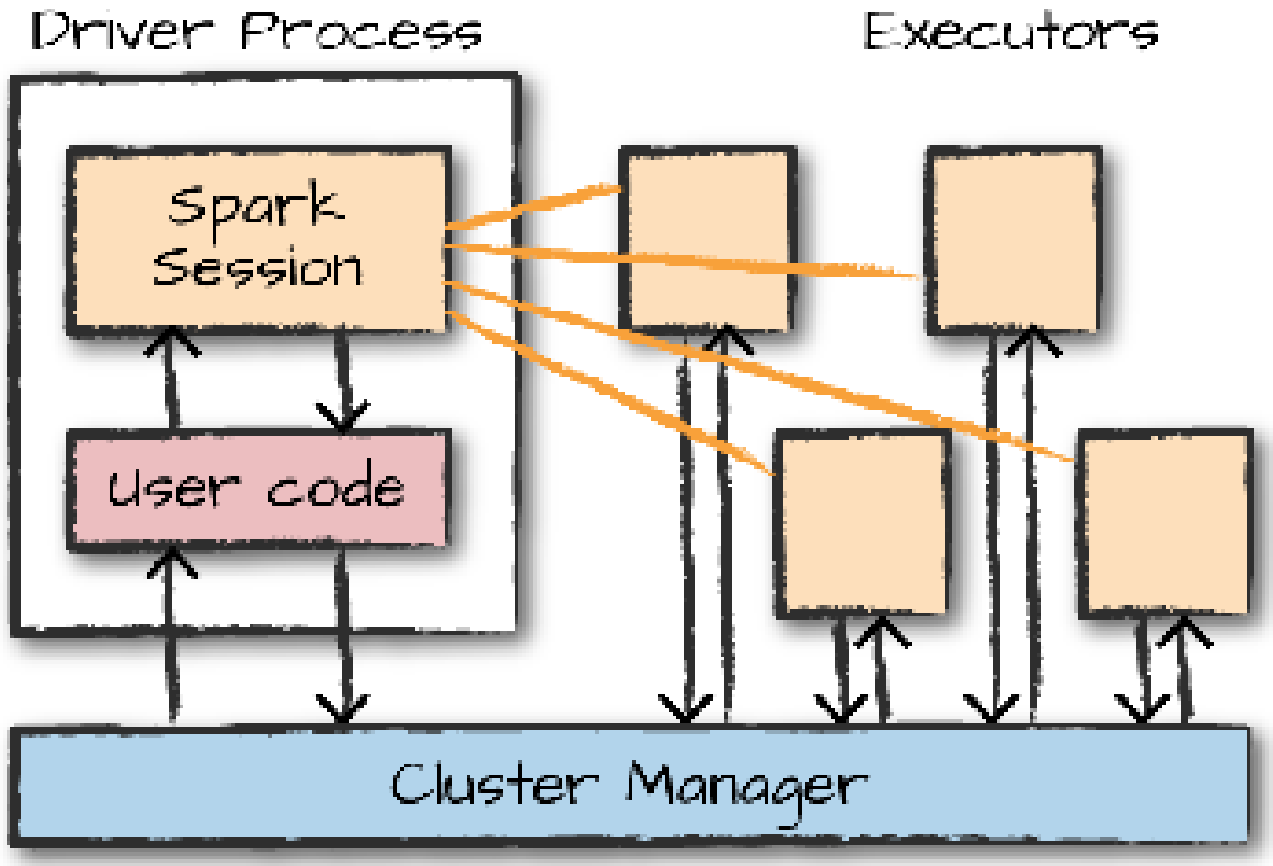


Figure 2-1. The architecture of a Spark Application

Figure 3: *Spark Core Architecture*

How Many Executors

- User specifies how many **executor** processes should fall on each cluster node
  - This can be declared at run time
  - This can be declared in the code
- There is a Spark mode called *local*
  - This runs both the driver and executors as local CPU threads and not distrubuted
  - Good for a quick test mode

Spark Application Have

- Spark Applications have:
  - A Cluster Manager
  - Driver process
  - Executors
  - Code that is executed across executors

Spark Language APIs

- Spark takes your logic in different languages
  - Translates it to the Core Spark language
  - Everything in Spark runs and computes in the Core Spark Language
- Scala is the default shell
  - You can launch this by typing from the command line:
  - `spark-shell`
  - This assumes you already installed Spark
- Spark runs on the JVM
  - Only requirement is Java 8 JDK
  - OpenJDK works fine

Languages

- We have said this a few times but again, Spark supports natively:
  - Scala
  - Java
  - Python
  - SQL, ANSI 2003 standard
  - R though the SparkR package

API Architecture

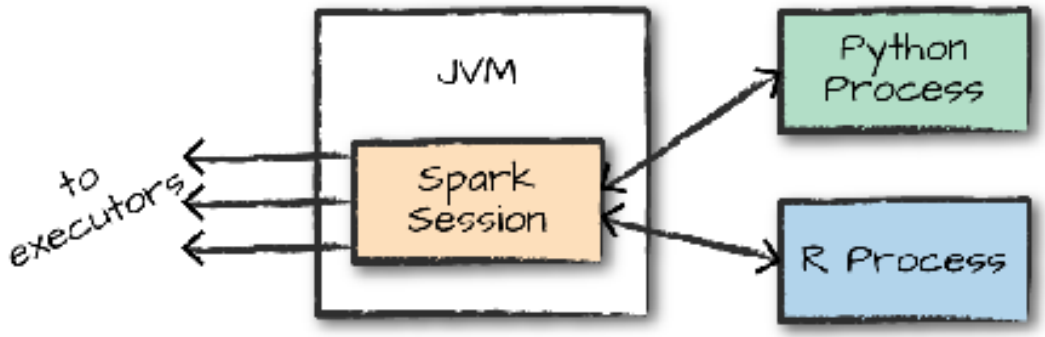


Figure 2-2. The relationship between the SparkSession and Spark's Language API

Figure 4: *Spark Executor Architecture*

How to interact with the Spark Session

- Every compiled spark code interacts through a `SparkSession()` object
  - `spark-submit` is for running batch jobs
  - Each Spark application has only 1 `SparkSession()`

Code

- Open the CLI in your Ubuntu Virtual machine
  - type: `spark-shell` or `pyspark`
  - For Scala, type:
  - `val myRange = spark.range(1000).toDF("number")`
  - For Python, type:
  - `myRange = spark.range(1000).toDF("number")`
- The text offers both languages, I will tend to use Python more

Conclusion

- Spark is great