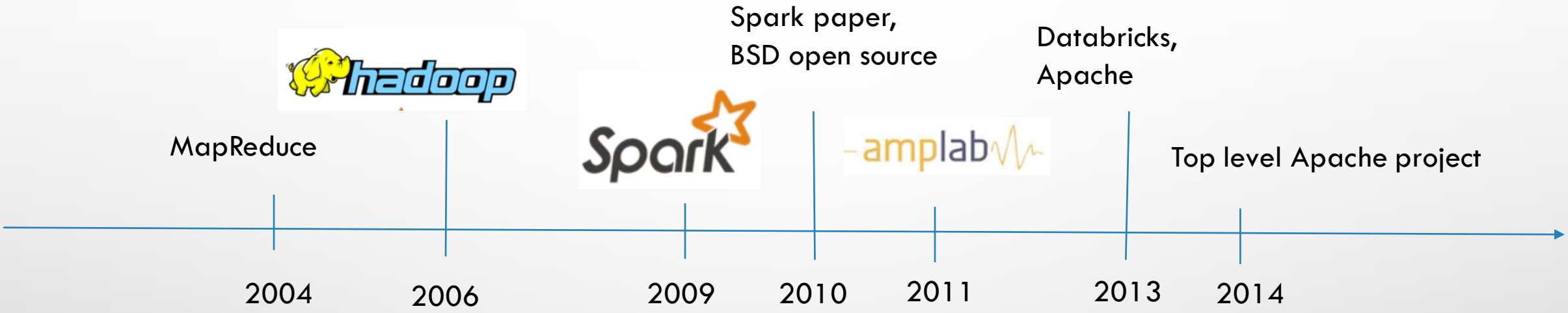
Introduction to Big Data with Apache Spark

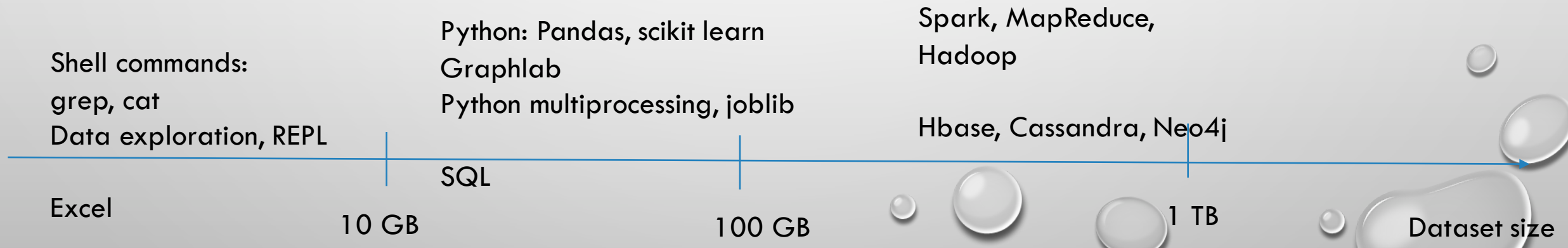
Alexey Svyatkovskiy

Princeton University

Matei Zakaria

HISTORY & DATA SCALE CONSIDERATIONS





Red color: Specific to Princeton clusters

SPARK SOFTWARE STACK

Spark R
Available starting Spark 1.5

Spark SQL structured data
Read: Distributed Pandas Dataframe

Spark Streaming real-time
Supports Kafka, Flume sinks

MLib machine learning

GraphX graph processing

Spark Core

Standalone Scheduler
Use SLURM on general purpose HPC clusters

YARN
Use on Hadoop cluster

Mesos

Storage:

Interconnect:

Infiniband

10g ethernet

GPFS

NFS

HDFS

Cassandra

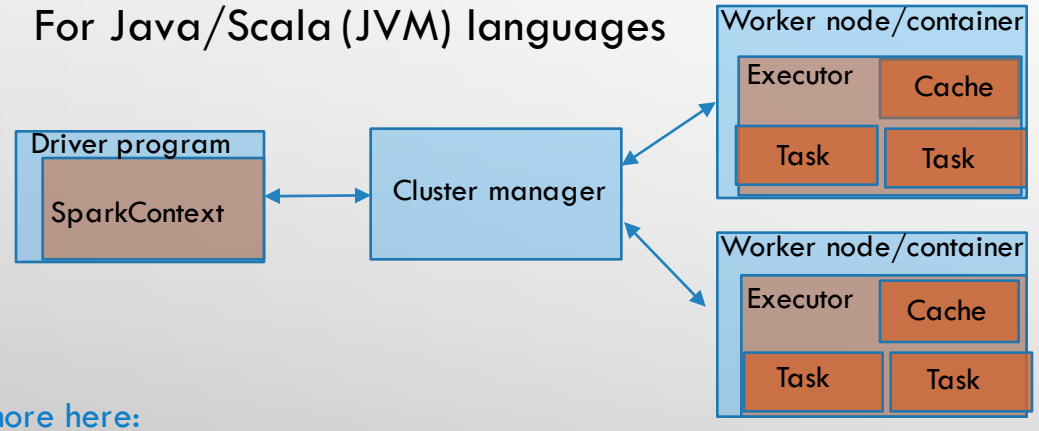
https://www.princeton.edu/researchcomputing/computational-hardware/hadoop/
https://www.princeton.edu/researchcomputing/computational-hardware/machine-2/

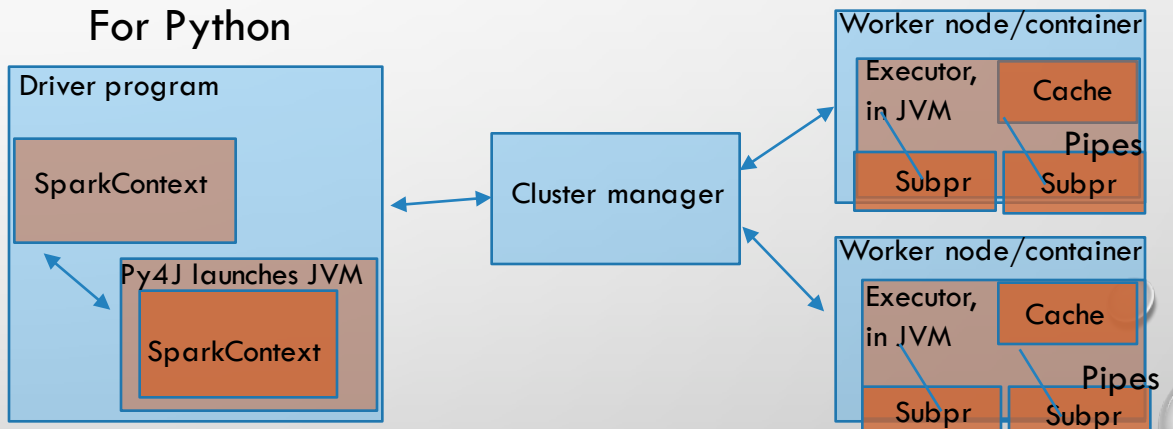
Spark installation and help pages

- To download spark follow the instructions on the following page: http://spark.apache.org/downloads.html for your operating system and setup
- To get started, and learn more about spark go to: http://spark.apache.org/docs/latest/
- Spark via slurm at Princeton:
  - https://www.princeton.edu/researchcomputing/faq/spark-via-slurm/
  - https://www.princeton.edu/researchcomputing/computational-hardware/hadoop/spark-memory/

Anatomy of a Spark applications

- Spark uses master/slave architecture with one central coordinator (driver) and many distributed workers (executors)
- Driver runs its own Java process, Executors each run their own java processes
- For PySpark, SparkContext uses Py4J to launch a JVM and create a JavaSparkContext
  - Executors all ran in JVMs, and Python subprocesses (tasks) which are launched communicate with them using pipes

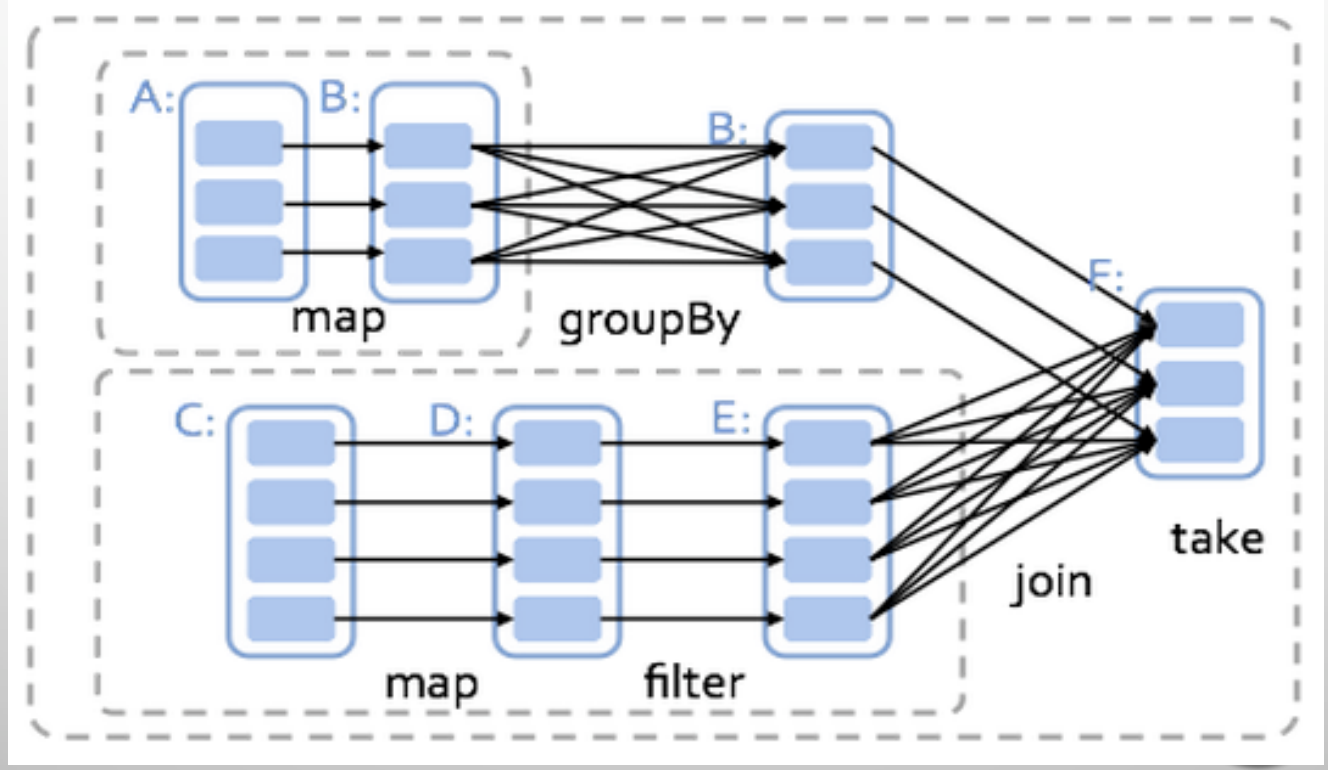




See more here:
https://cwiki.apache.org/confluence/display/SPARK/PySpark+Internals

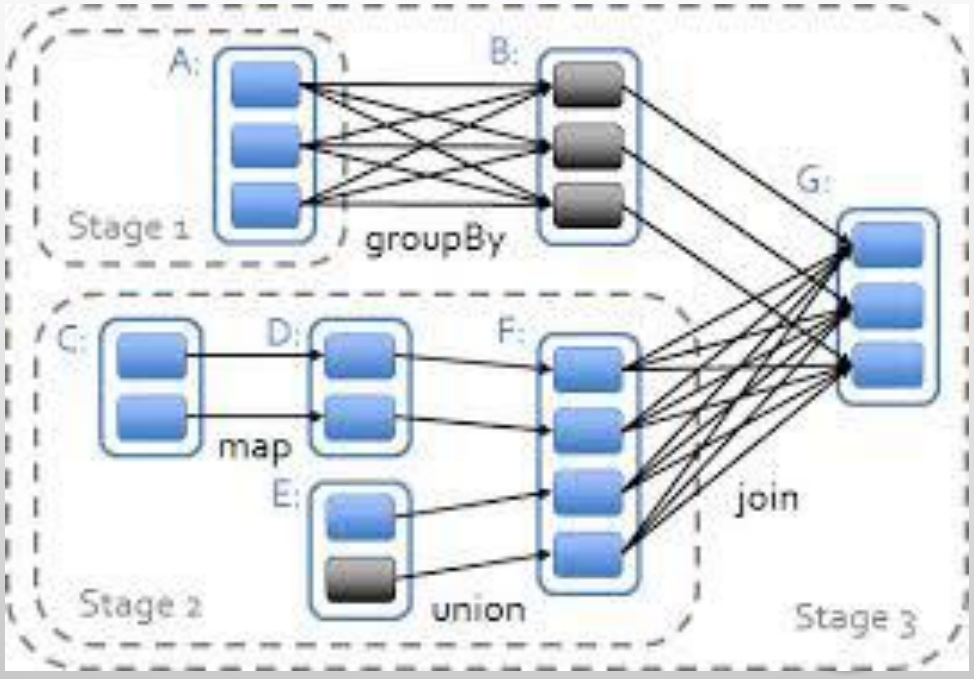
Spark programming model: DAG

- **Directed acyclic graph (DAG)** model used in Spark extends the mapreduce paradigm. the MapReduce model states that distributed computation on a large dataset can be boiled down to two kinds of computation steps - a map step and a reduce step
  - Complex computations typically require multiple map and reduce steps. When you have multiple such steps, it essentially forms a DAG of operations.



Spark RDDs, lineage

- **Resilient distributed datasets (RDDs):** distributed fault-tolerant collections of elements that can be operated on in parallel.
  - RDD is like a (smartly) distributed list, which is partitioned across the nodes of a cluster and can be operated on in parallel
- Besides RDDs there are higher level APIs: dataframes and datasets, which allow to take the most out of the Catalyst, have more relational primitives and allow operations directly on serialized dtaa



Transformations and actions

- **Transformations:** operations on RDD that return a new RDD. They do not mutate the old RDD, but rather return a pointer to it
- Examples of transformations are:

Transformation	Meaning
map(*func*)	Return a new distributed dataset formed by passing each element of the source through a function *func*.
filter(*func*)	Return a new dataset formed by selecting those elements of the source on which *func* returns true.

  - http://spark.apache.org/docs/latest/programming-guide.html#transformations
  - Transformations are **lazily evaluated,** meaning they do not compute the answer unless an action is called on it
  - intermediate results to computations do not necessarily get created and stored - if the output of your map is going straight into a reduce
  - results of transformations are, by default, recomputed each time they are needed - to store them one must explicitly persist them in memory or on disk

Actions

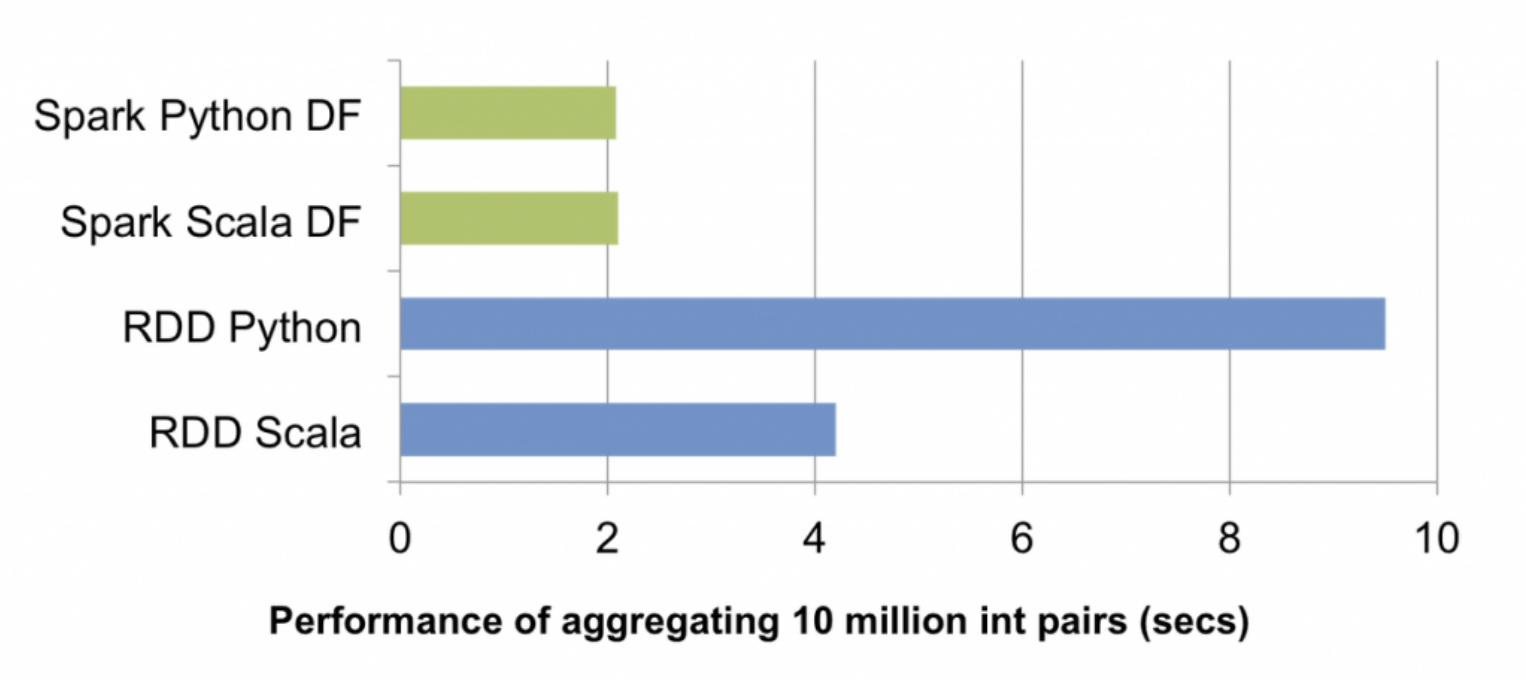
- **Actions:** force program to produce some outpout (note: RDDs are lazily evaluated)

Action	Meaning
reduce(*func*)	Aggregate the elements of the dataset using a function *func* (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.
collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.

- http://spark.apache.org/docs/latest/programming-guide.html#actions
- These actually return a value as a result of a computation. For instance, reduce is an action that combines all elements of an RDD using some function and then returns the final result. (Note that reduceByKey actually returns an RDD.)

Working with DataFrames

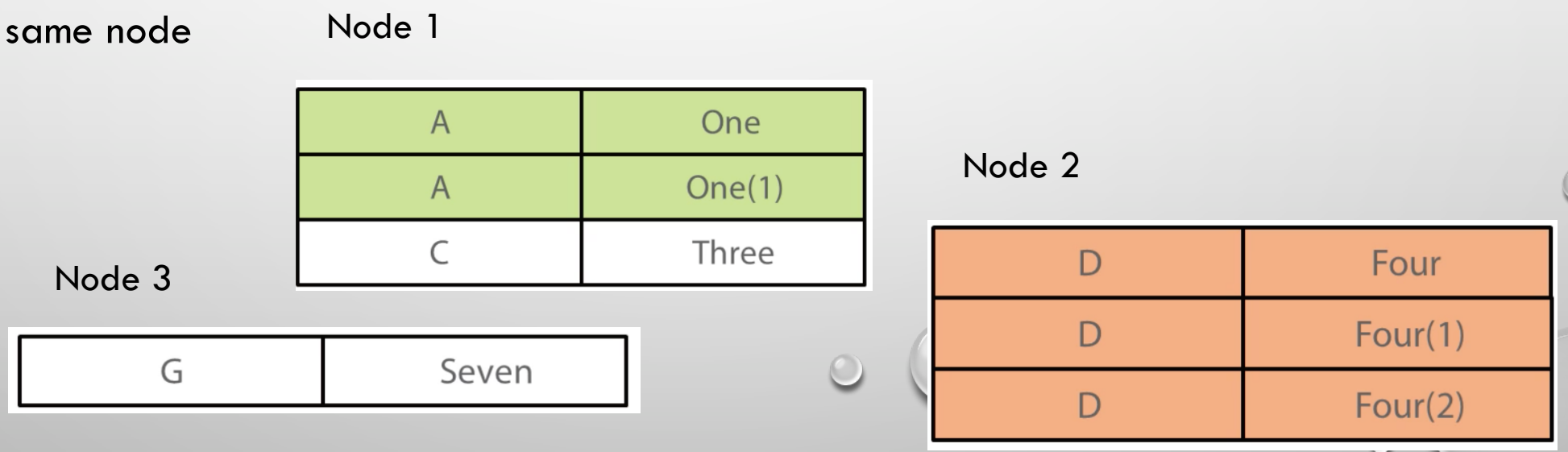
- When working with structured or semi-structured data provided as csv, JSON, Avro or other datatype having schema it is often better to use dataframes
- Take advantage of catalyst optimizer



- The above chart compares the runtime performance of running group-by-aggregation on 10 million integer pairs on a single machine
- Since both Scala and Python DataFrame operations are compiled into JVM bytecode for execution, there is little difference between the two languages, and both outperform the vanilla Python RDD variant by a factor of 5 and Scala RDD variant by a factor of 2.

Working with key-value pairs

- Some types of data are naturally stored in key-value pairs (tuples)
- A set of pair transformations and actions is supported:
  - collectAsMap
  - MapValues, flatMapValues
  - ReduceByKey
- Partitioning makes sure regions with same key are grouped on the same node



Input: ingest data into an RDD