Scala and Spark Overview

Scala and Spark

- In this lecture we will give an overview of the Scala programming language
- Then we will discuss the general Big Data Ecosystem
- Afterwards we will show how Apache Spark fits into all of this.

Scala

- Scala is a general purpose programming language
- It was designed by Matrin Odersky in the early 2000s at EPFL (École Polytechnique Fédérale de Lausanne)
- It was designed to overcome criticism of Java's shortcomings.

Scala

- Scala source code is intended to be compiled to Java bytecode to run on a Java Virtual Machine (JVM)
- Java libraries may be used directly in Scala
- Unlike Java, Scala has many features of functional programming

Scala

- A large reason Scala demand has dramatically risen in recent years is because of Apache Spark.
- Let’s discuss what Spark is in the context of Big Data.
- We’ll begin with a general explanation of what Big Data is and related technologies.

Big Data Overview

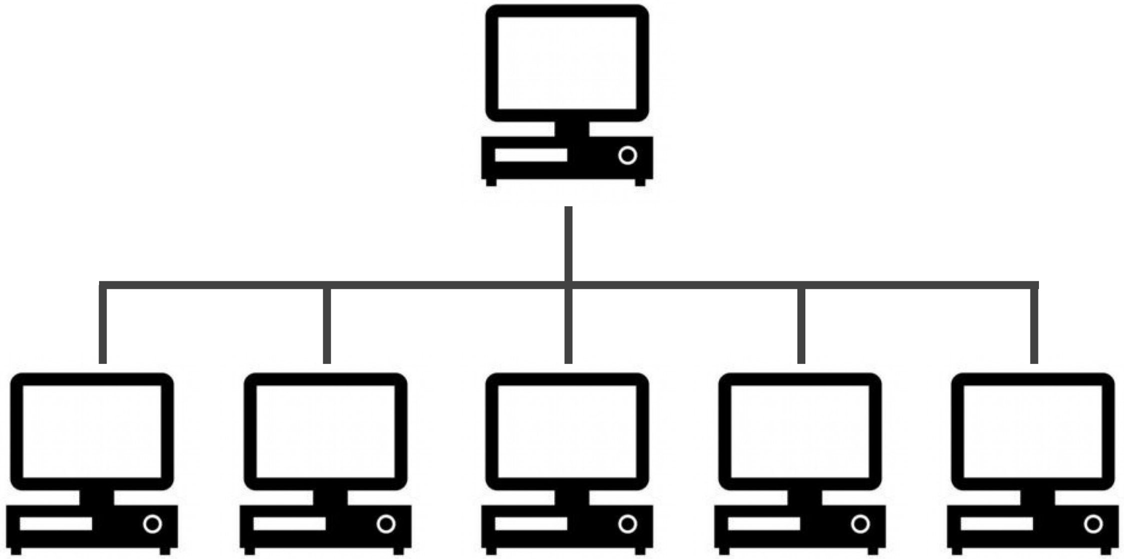
- Explanation of Hadoop, MapReduce,and Spark
- Local versus Distributed Systems
- Overview of Hadoop Ecosystem
- Overview of Spark

Big Data

- Data that can fit on a local computer, in the scale of 0-32 GB depending on RAM.
- But what can we do if we have a larger set of data?
    - Try using a SQL database to move storage onto hard drive instead of RAM
    - Or use a distributed system, that distributes the data to multiple machines/computer.

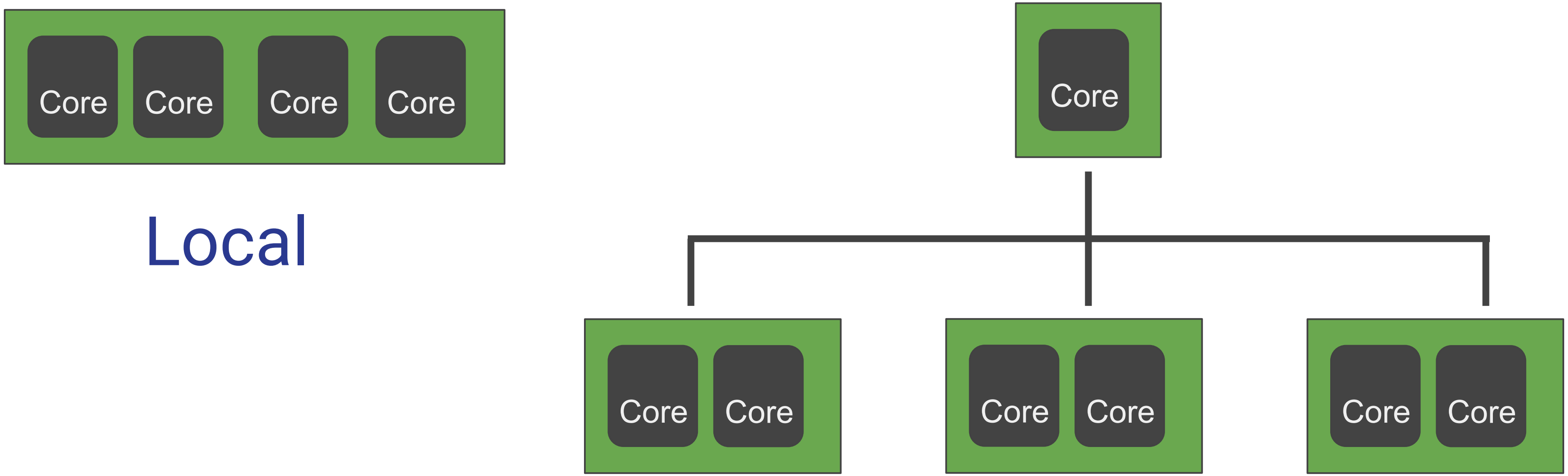
Local versus Distributed

Local



Distributed

Local versus Distributed



Local

Distributed

Big Data

- A local process will use the computation resources of a single machine
- A distributed process has access to the computational resources across a number of machines connected through a network
- After a certain point, it is easier to scale out to many lower CPU machines, than to try to scale up to a single machine with high a CPU

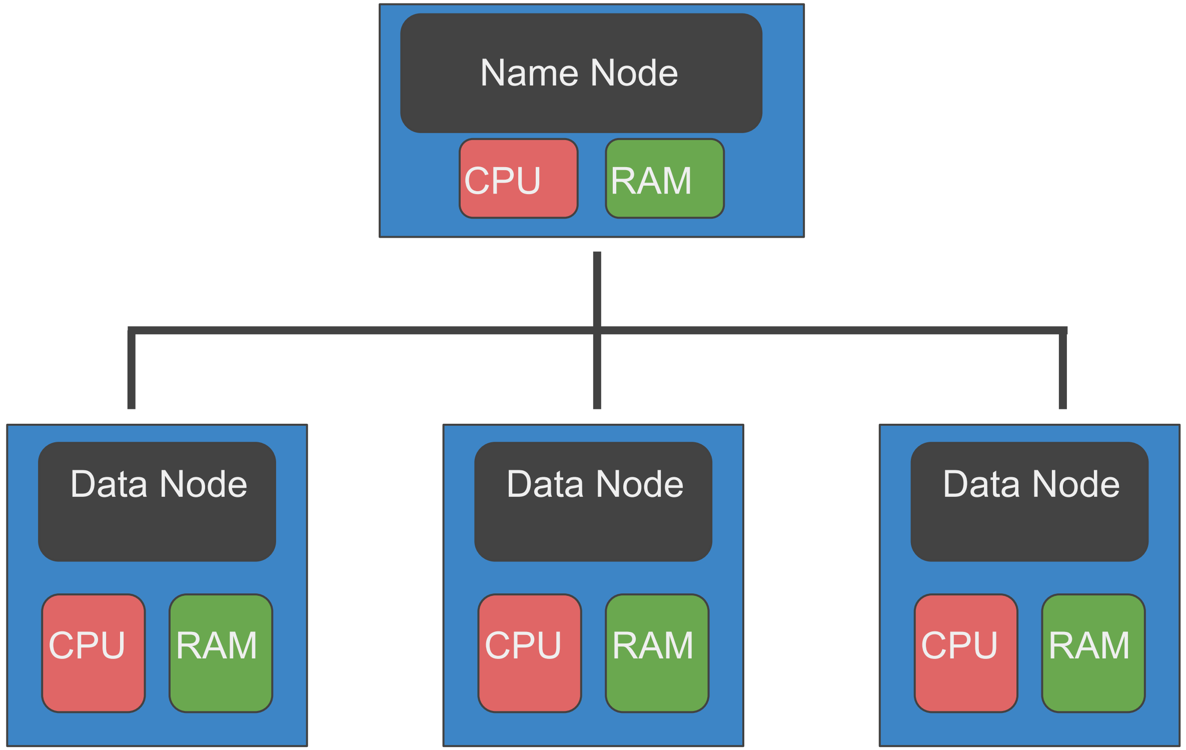
Big Data

- Distributed machines also have the advantage of easily scaling, you can just add more machines
- They also include fault tolerance, if one machine fails, the whole network can still go on.
- Let’s discuss the typical format of a distributed architecture that uses Hadoop

Hadoop

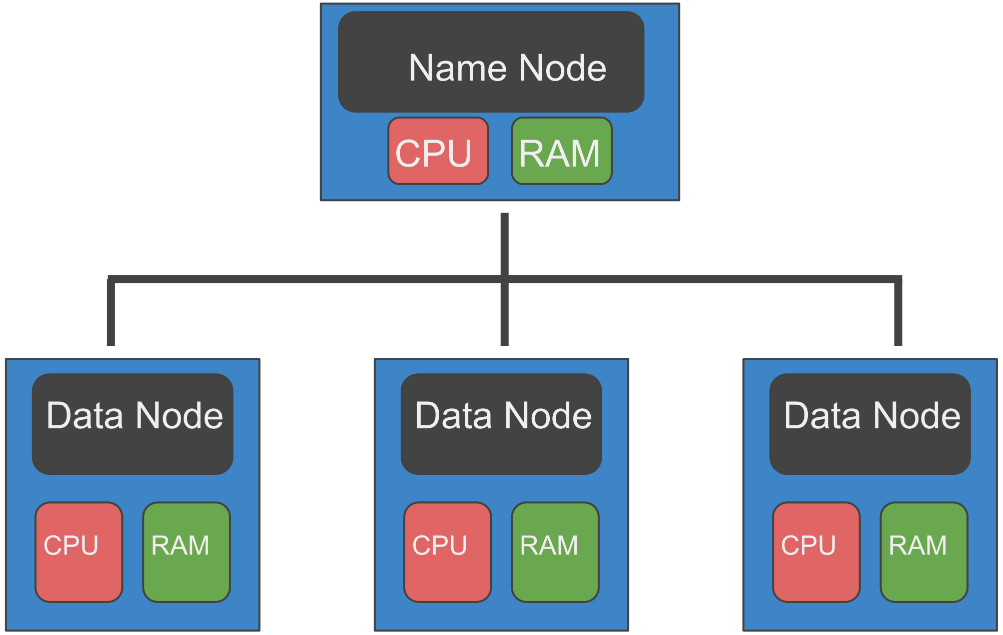
- Hadoop is a way to distribute very large files across multiple machines.
- It uses the Hadoop Distributed File System (HDFS)
- HDFS allows a user to work with large data sets
- HDFS also duplicates blocks of data for fault tolerance
- It also then uses MapReduce
- MapReduce allows computations on that data

Distributed Storage - HDFS



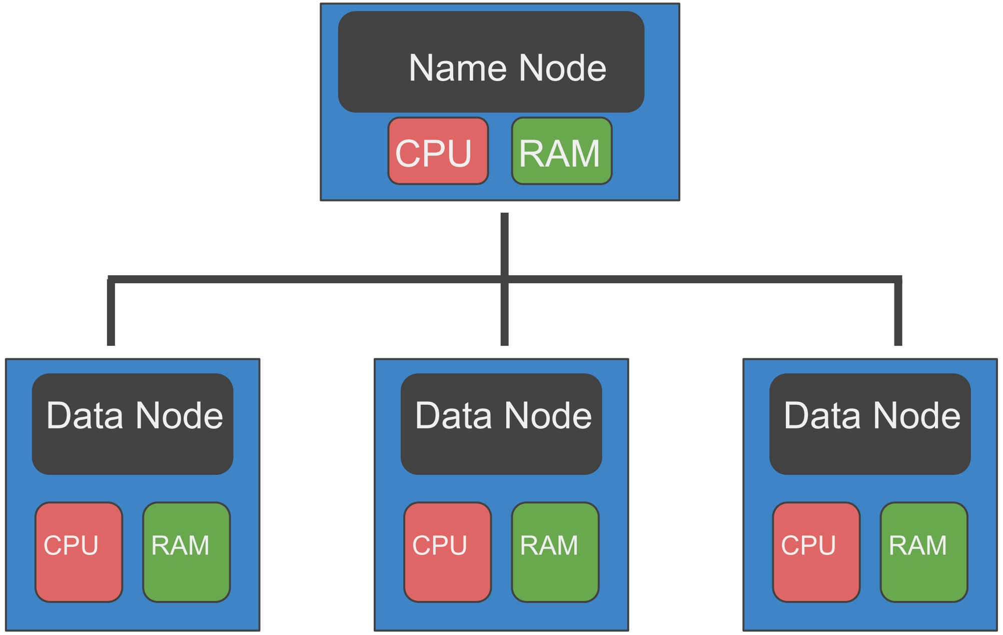
Distributed Storage - HDFS

- HDFS will use blocks of data, with a size of 128 MB by default
- Each of these blocks is replicated 3 times
- The blocks are distributed in a way to support fault tolerance



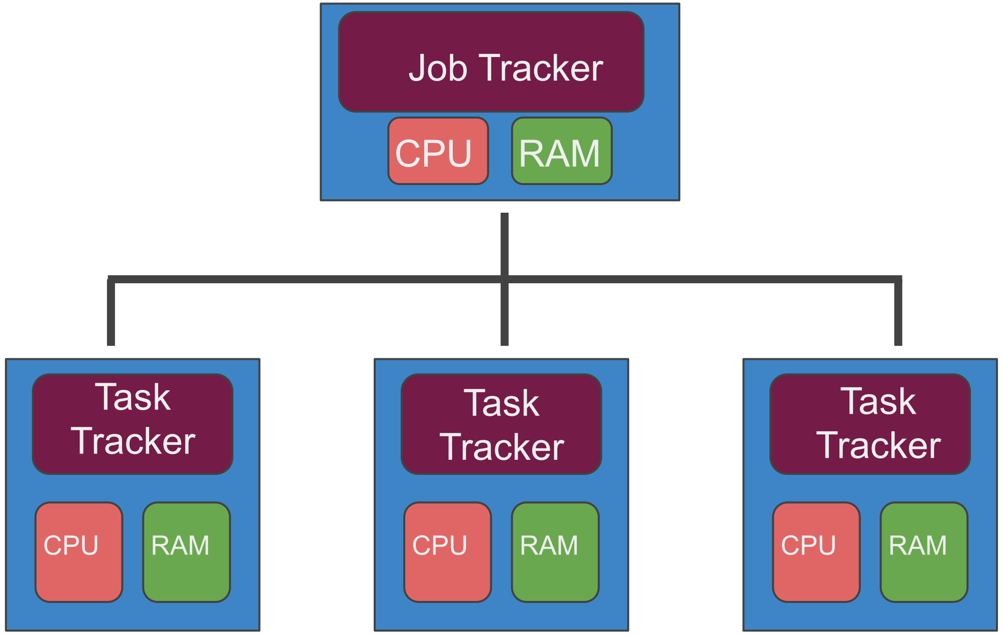
Distributed Storage - HDFS

- Smaller blocks provide more parallelization during processing
- Multiple copies of a block prevent loss of data due to a failure of a node



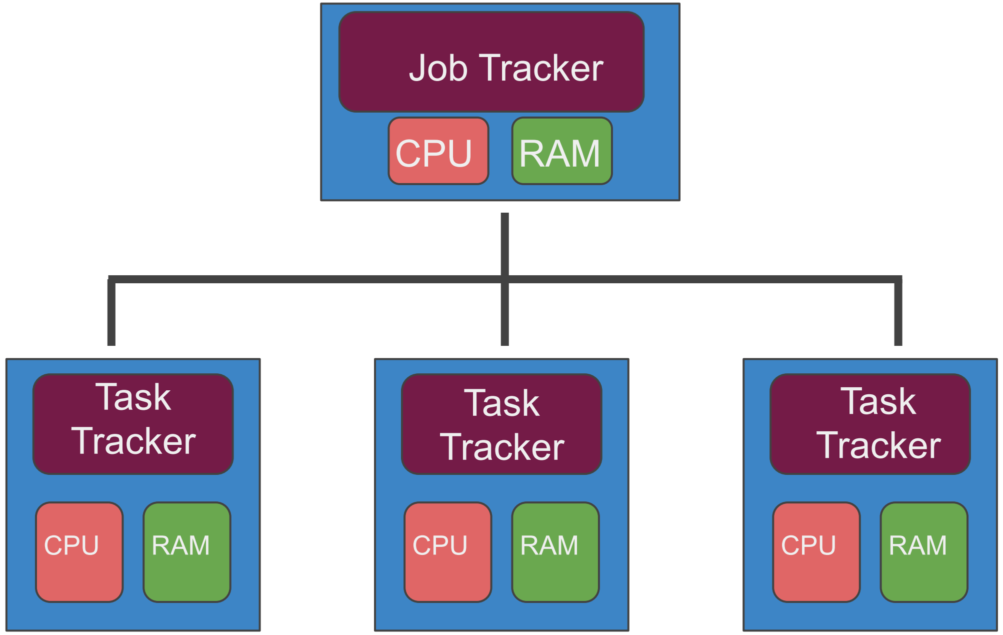
MapReduce

- MapReduce is a way of splitting a computation task to a distributed set of files (such as HDFS)
- It consists of a Job Tracker and multiple Task Trackers



MapReduce

- The Job Tracker sends code to run on the Task Trackers
- The Task trackers allocate CPU and memory for the tasks and monitor the tasks on the worker nodes



Big Data

- What we covered can be thought of in two distinct parts:
  - Using HDFS to distribute large data sets
  - Using MapReduce to distribute a computational task to a distributed data set
- Next we will learn about the latest technology in this space known as Spark.
- Spark improves on the concepts of using distribution

Spark

- This lecture will be an abstract overview, we will discuss:
    - Spark
    - Spark vs MapReduce
    - Spark RDDs
    - Spark DataFrames

Spark

- Spark is one of the latest technologies being used to quickly and easily handle Big Data
- It is an open source project on Apache
- It was first released in February 2013 and has exploded in popularity due to it’s ease of use and speed
- It was created at the AMPLab at UC Berkeley

Spark

- You can think of Spark as a flexible alternative to MapReduce
- Spark can use data stored in a variety of formats
  - Cassandra
  - AWS S3
  - HDFS
  - And more

Spark vs MapReduce

- MapReduce requires files to be stored in HDFS, Spark does not!
- Spark also can perform operations up to 100x faster than MapReduce
- So how does it achieve this speed?

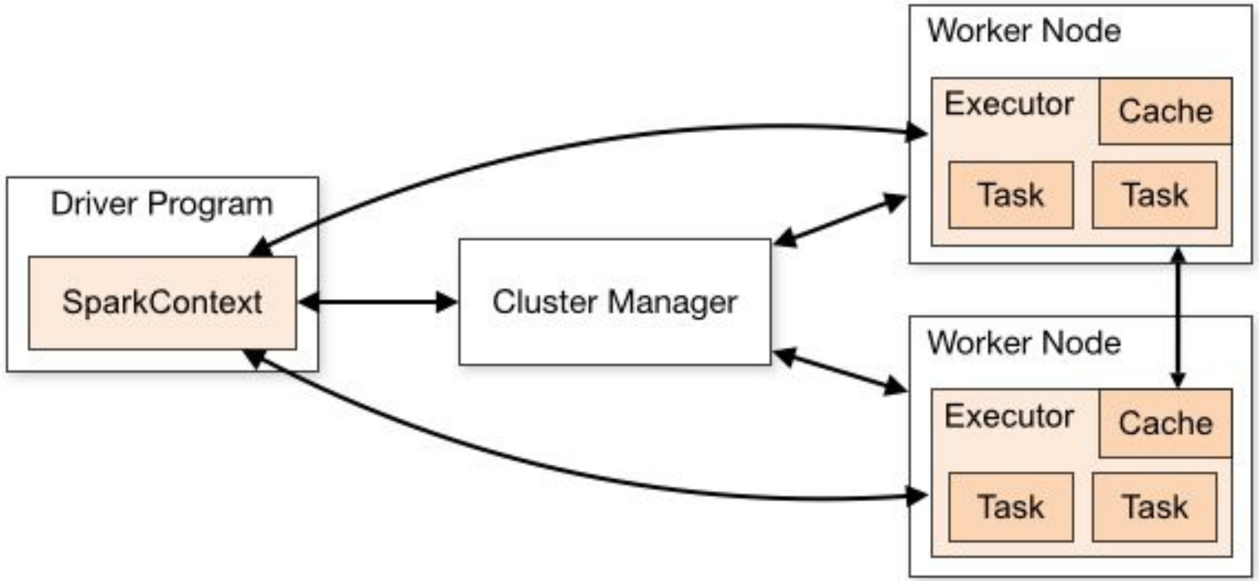
Spark vs MapReduce

- MapReduce writes most data to disk after each map and reduce operation
- Spark keeps most of the data in memory after each transformation
- Spark can spill over to disk if the memory is filled

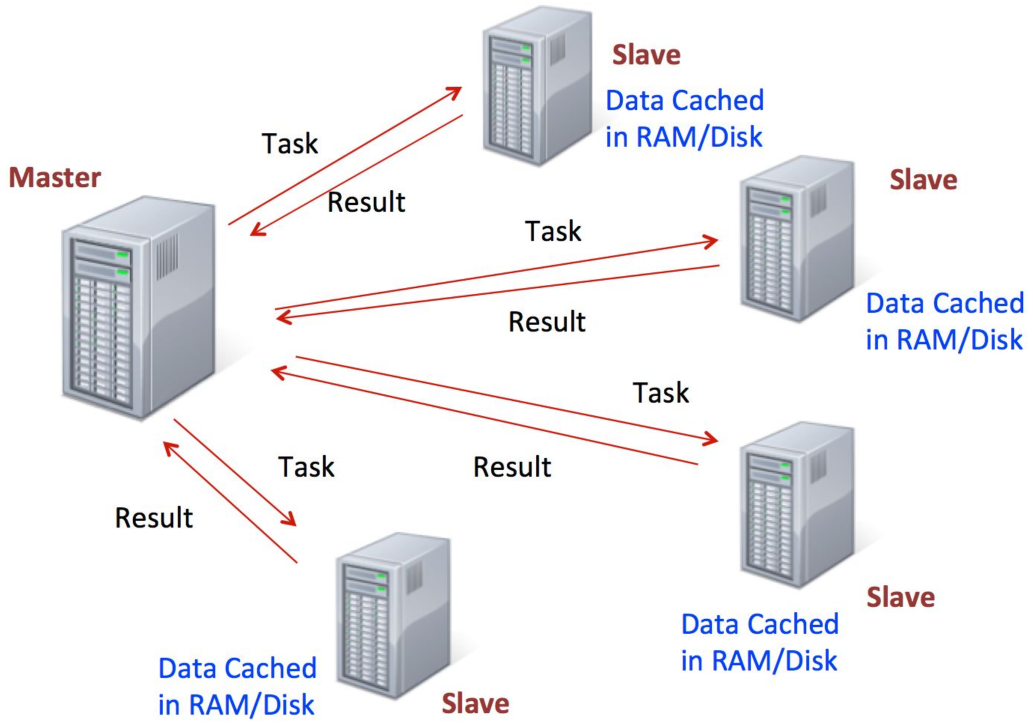
Spark RDDs

- At the core of Spark is the idea of a Resilient Distributed Dataset (RDD)
- Resilient Distributed Dataset (RDD) has 4 main features:
  - Distributed Collection of Data
  - Fault-tolerant
  - Parallel operation - partioned
  - Ability to use many data sources

Spark RDDs


Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task
Worker Node
Executor
Cache
Task
Task

Spark RDDs


Master
Slave
Data Cached
in RAM/Disk
Task
Result
Slave
Task
Result
Data Cached
in RAM/Disk
Task
Result
Slave
Data Cached
in RAM/Disk
Task
Result
Data Cached
in RAM/Disk
Slave

Spark RDDs

- RDDs are immutable, lazily evaluated, and cacheable
- There are two types of RDD operations:
  - Transformations
  - Actions
- Transformations are basically a recipe to follow.
- Actions actually perform what the recipe says to do and returns something back.

Spark RDDs

- When discussing Spark syntax you will see RDD versus DataFrame syntax show up.
- With the release of Spark 2.0, Spark is moving towards a DataFrame based syntax, but keep in mind that the way files are being distributed can still be thought of as RDDs, it is just the typed out syntax that is changing

Spark RDDs

- We’ve covered a lot!
- Don’t worry if you didn’t memorize all these details, a lot of this will be covered again as we learn about how to actually code out and utilize these ideas!

Spark RDDs

- Basic Actions
  - First
  - Collect
  - Count
  - Take

Spark RDDs

- Collect - Return all the elements of the RDD as an array at the driver program.
- Count - Return the number of elements in the RDD
- First - Return the first element in the RDD
- Take - Return an array with the first n elements of the RDD

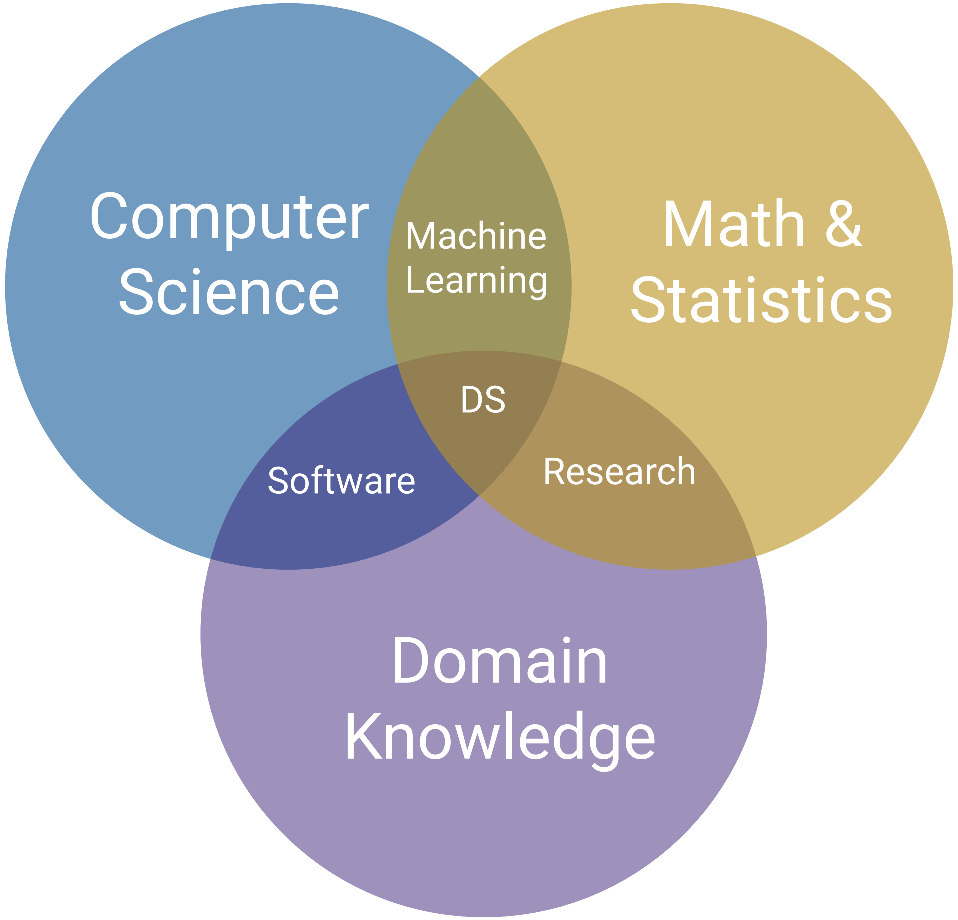
Spark RDDs

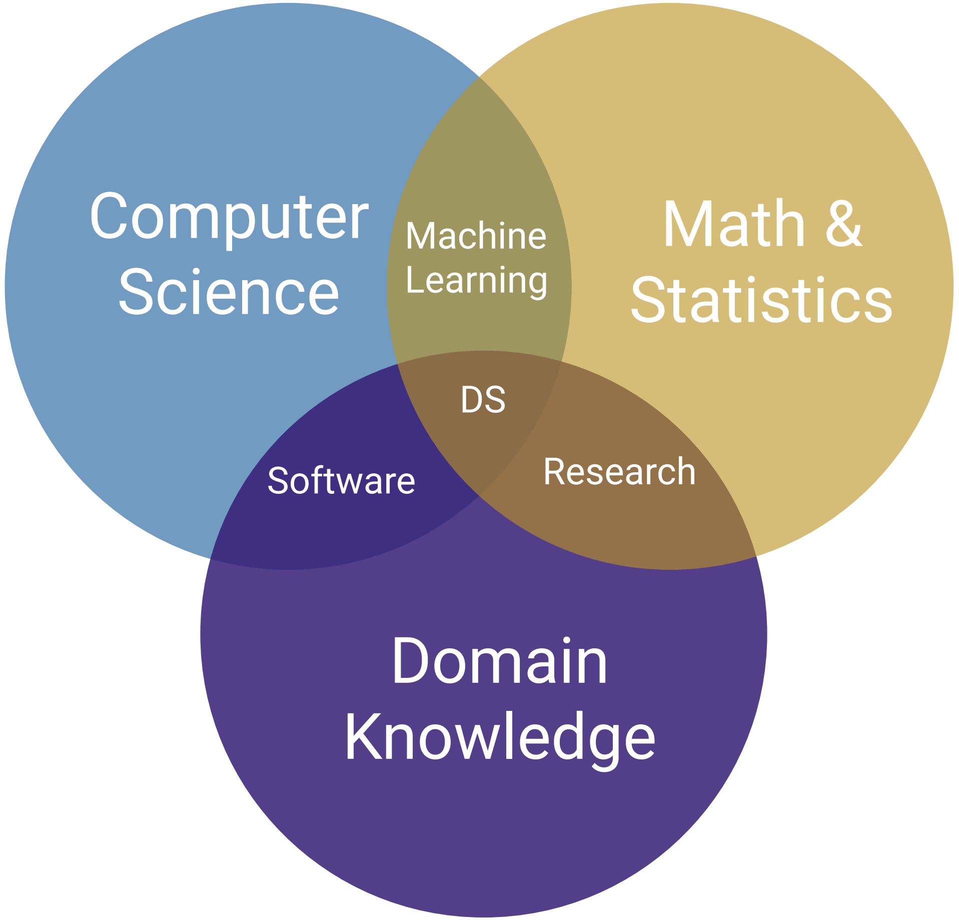
- Basic Transformations
  - Filter
  - Map
  - FlatMap

Spark DataFrames

- Spark DataFrames are also now the standard way of using Spark’s Machine Learning Capabilities.
- Spark DataFrame documentation is still pretty new and can be sparse.
- Let’s get a brief tour of the documentation!

http://spark.apache.org/


Computer Science
Machine Learning
Math & Statistics
DS
Software
Research
Domain Knowledge


Computer Science
Machine Learning
Math & Statistics
DS
Software
Research
Domain Knowledge