

# Resilient Distributed Datasets

...

The Fundamental Data Structure in Spark

# Outline

(of this segment)

- Common Data Structures in Statistical Programming
- Basic Concepts in Spark
- RDDs
- The Spark Workflow
- Workflow Template

---

# Common Data Structures

# Vectors

And vectorized computing

- What is vectorization?
- Why is it fast?

---

# Vectors

And functional programming

- How is it related to functional programming?
- Why Functional Programming matters?

---

# Functional Programming

What is it?

Here are some things we care about:

- Functions as first-class objects
  - Anonymous Functions
- Side-effects (or not)

---

# Functions as first-class objects

One of my goals for Python was to make it so that all objects were "first class." By this, I meant that I wanted all objects that could be named in the language (e.g., integers, strings, functions, classes, modules, methods, etc.) to have equal status. That is, they can be assigned to variables, placed in lists, stored in dictionaries, passed as arguments, and so forth.

-- Guido van Rossum

(<http://python-history.blogspot.in/2009/02/first-class-everything.html>)

# First-class Functions

Functions as first-class objects

- being expressible as an anonymous literal value
- **being storable in data structures having an intrinsic identity (independent of any given name)**
- being passable as a parameter to a procedure/function
- being returnable as the result of a procedure/function
- being constructible at runtime
- being transmissible among distributed processes
- being storable outside running processes

---



# Anonymous Functions

A function has no name

- Why?
- Lambda

---

# Side-effects

Thinking about parallelism

- Functions in Programming
- Functions in Mathematics
- Functions in Distributed or Parallel Programming

Generating Random Numbers?

---

# Where does this leave us?

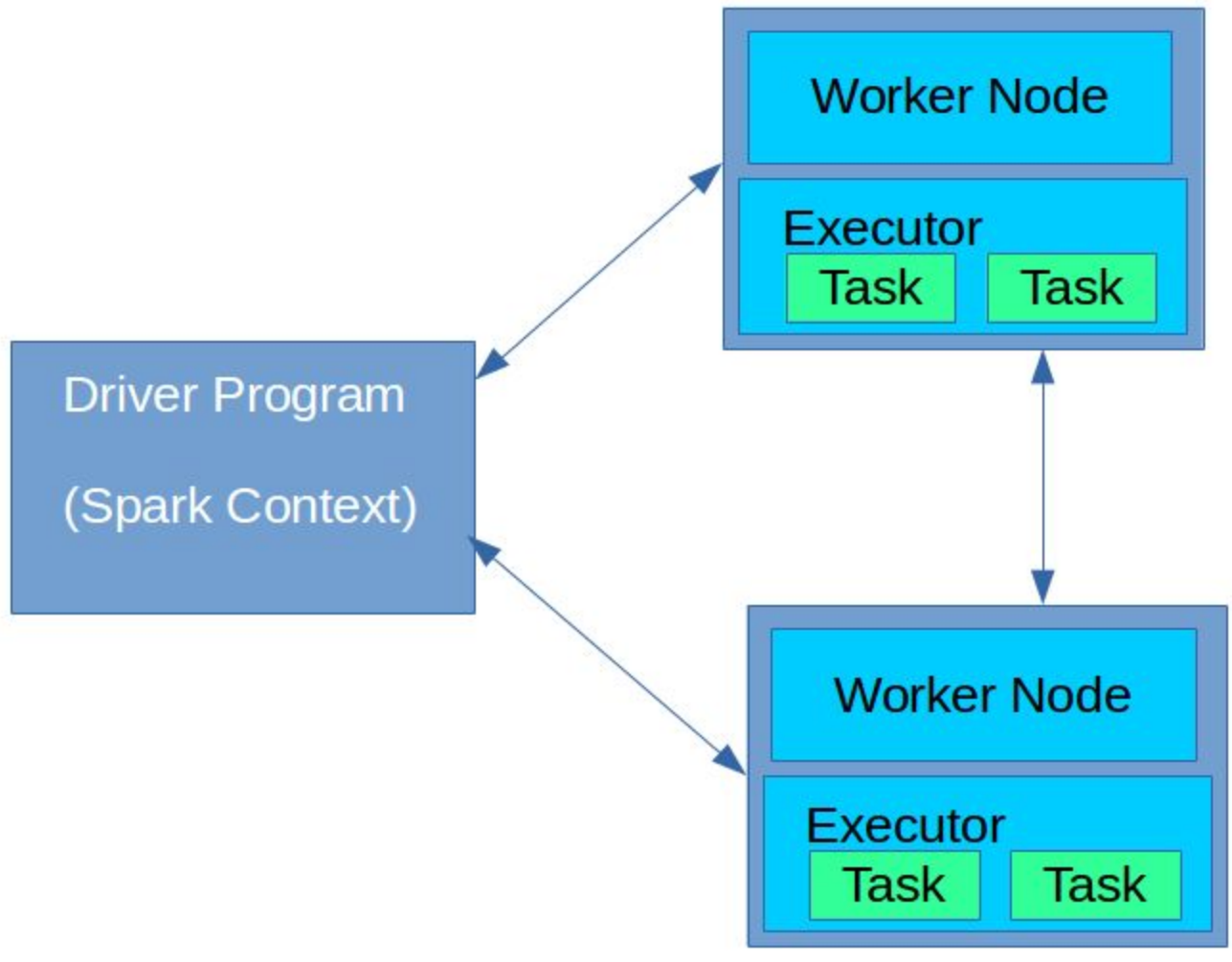
- Is Python a Functional Programming language?
- Is Scala a Functional Programming language?
- Is R a Functional Programming language?
- Is Spark a functional programming language?

---

# Where does this leave us?

- Is Python a Functional Programming language?
  - Is Scala a Functional Programming language?
  - Is R a Functional Programming language?
  - Is Spark a functional programming language?
    - No, it's not even a programming language?
    - But functions are first class objects in Python and Scala
-

# Some Basic Concepts



# Spark Context

- Represents a connection to a computing cluster
- In pySpark shell, a sparkContext is automatically created
  - A variable called `sc`
- In other modes (say Jupyter notebooks), we have to create it ourselves

---

# Driver Program

Every Spark application consists of a driver program

- launches various parallel operations on a cluster
  - defines distributed datasets on the cluster, then applies operations to them
- contains application's *main* function
- accesses Spark through a `SparkContext` object

---



# Executor

## Worker Nodes

- The typical worker nodes of the cluster
- Managed by the Driver Program

Let's simulate counting words in a text file.

---

# RDDs

# What are RDDs

Can read from and written to distributed storages like HDFS or S3

- Resilient
    - Can be rebuilt from a lineage (fault tolerant)
  - Distributed
    - partitioned across machines  
accessed via parallel operations (fast)
  - Vectorised
  - Immutable
    - read-only collection of objects
  - Cached
    - Because RDDs can be cached in memory of worker nodes, they can be good at iterative applications
-

# What are RDDs

A broader look

- Vectorization as a common pattern in other statistical programming languages
  - R
  - Python
- Functional Programming
- We just discussed these ideas

---

# Let's see it in practice

Let's use lambda to create an  
anonymous function to count the  
number of lines in a file  
containing Python

Spark\_Activities\_01\_Basics.ipynb

Activity 2: Using Anonymous  
Functions

---

# Let's see it in practice

We'll go ahead and calculate the  
number of primes less than a  
given large number

Spark\_Activities\_01\_Basics.ipynb

Activity 3: Counting Primes

---

# The Spark Workflow

# Working with RDDs

The simple, wrong and useful  
description of the process

- RDDs are created and distributed across worker nodes
- The Driver distributes code (transformations) across the cluster to be executed by workers on their partitions of the RDD
- Results are then sent back to the driver for aggregation (Actions)

Laziness

---



# Working with RDDs

Spark applications are  
manipulation of RDDs through  
transformations and actions

- Define RDD(s) by
    - accessing data stored on disk (HDFS, Cassandra, HBase, Local Disk)
    - parallelizing some collection in memory
    - transforming an existing RDD
    - caching or saving
  - Invoke operations on the RDD(s) by passing closures (functions) to each element of the RDD
  - Use the resulting RDDs with actions (e.g. count, collect, save, etc.). Actions kick off the computing on the cluster.
-

# Working with RDDs

## Variable and Scoping

When Spark runs a closure on a worker, any variables used in the closure are copied to that node, but are maintained within the local scope of that closure.

Spark provides two types of shared variables that can be interacted with by all workers in a restricted fashion

---

# Working with RDDs

## Broadcasts and Accumulators

- *Broadcast* variables are distributed to all workers, but are read-only. Broadcast variables can be used as lookup tables or stopword lists.
- *Accumulators* are variables that workers can "add" to using associative operations and are typically used as counters.

---

# Workflow Template

# More examples

Word and Line counting

Spark\_Activities\_01\_Basics.ipynb

Activity 4: Word and Line Counting

---

# The Template

Spark\_Activities\_01\_Basics.ipynb

Activity 5: Workflow Template

- `main()`
- `if __name__ == "__main__":`

---

# Sample Application

Spark\_Activities\_01\_Basics.ipynb

Activity 6: Sample Application

---