



Carnegie Mellon University

# Machine Learning with Large Datasets (Fall 2022)

---

Recitation 1

*September 2, 2022*

Special thanks to **Tian Li** and **Prof. Heather Miller** who developed some of the material covered today.

# Agenda

## 1. Introduction to Databricks

- Registration on Databricks (Community Edition)
- Uploading Notebooks
- Creation of Cluster

## 2. Spark Basics

- Spark Architecture
- Execution of Spark Program
- Spark APIs
- Lambda Functions
- Examples
- Cache/Persistence

## 3. Programming Demo

# Homework Setup

## Programming Assignments

- **5** programming assignments - **4** using PySpark, **1** using Tensorflow (Subject to Change)
- PySpark assignments can be completed on Databricks (Community).
- More details about the tensorflow assignments will be shared later.
- For autograding, submit assignments on Gradescope.
- Detailed instructions on how to submit assignment for grading will be in the writeup.

## Written Assignments

- Need to be directly submitted to Gradescope.

**More info on Course Website -** <https://10605.github.io/>

# To-Dos for students

- Register for a free community version of Databricks.
- Download the assignment and import the Jupyter Notebooks on Databricks.
- Configure the environment according to the instructions in the writeup.
  - Creating a cluster.
  - Installing a third-party package.
- After all *local* tests pass, hand-in the assignment on Gradescope for grading.
  - Please note that you get points if the tests on the grader pass.
  - The local tests exists to help you develop and test your code.

# Registration

<https://databricks.com/try-databricks>

## Try Databricks for free

An open and unified data analytics platform for data engineering, data science, machine learning, and analytics. From the original creators of Apache Spark™, Delta lake, MLflow, and Koalas.



### Databricks trial:

- Collaborative environment for data teams to build solutions together.
- Interactive notebooks to use Apache Spark™, SQL, Python, Scala, Delta Lake, MLflow, TensorFlow, Keras, Scikit-learn and more.
- Available as a 14-day full trial in your own cloud, or as a lightweight trial hosted by Databricks.

### Used by:



Please tell us about yourself

First Name: \*

Last Name: \*

Company \*

Company Email \*

Title \*

Phone Number

☐ Keep me informed with occasional updates about Databricks and related open source products

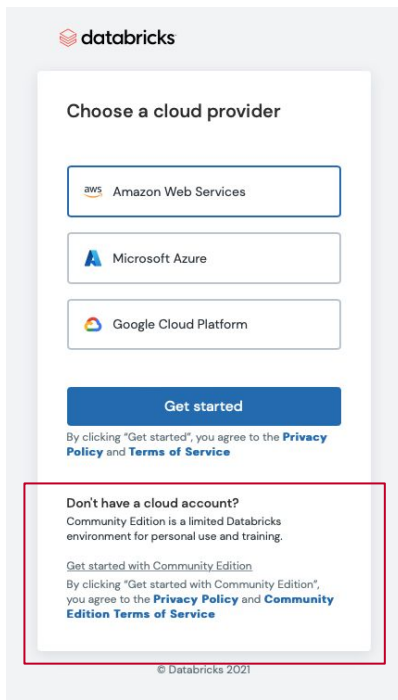
By Clicking "Get Started For Free", you agree to the [Privacy Policy](#).

GET STARTED FOR FREE

Carnegie  
Mellon  
University

# Registration

Please only choose the  
community edition and  
NOT a cloud provider.



The image shows a registration form for Databricks. At the top is the Databricks logo. Below it is the heading "Choose a cloud provider". There are three buttons for cloud providers: "Amazon Web Services", "Microsoft Azure", and "Google Cloud Platform". Below these is a blue "Get started" button. Under the button, it says "By clicking 'Get started', you agree to the Privacy Policy and Terms of Service". At the bottom, there is a section titled "Don't have a cloud account?" which describes the Community Edition as a limited environment for personal use and training. It includes a link "Get started with Community Edition" and states that by clicking it, the user agrees to the Privacy Policy and Community Edition Terms of Service. The entire form is enclosed in a light gray border, and the bottom section is highlighted with a red border.

databricks

Choose a cloud provider

aws Amazon Web Services

A Microsoft Azure

Google Cloud Platform

Get started

By clicking "Get started", you agree to the [Privacy Policy](#) and [Terms of Service](#)

**Don't have a cloud account?**  
Community Edition is a limited Databricks environment for personal use and training.

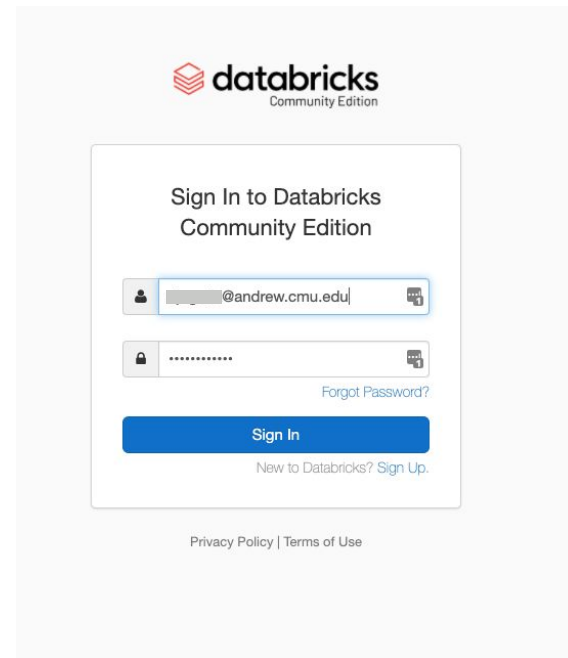
[Get started with Community Edition](#)  
By clicking "Get started with Community Edition", you agree to the [Privacy Policy](#) and [Community Edition Terms of Service](#)

© Databricks 2021

# Login

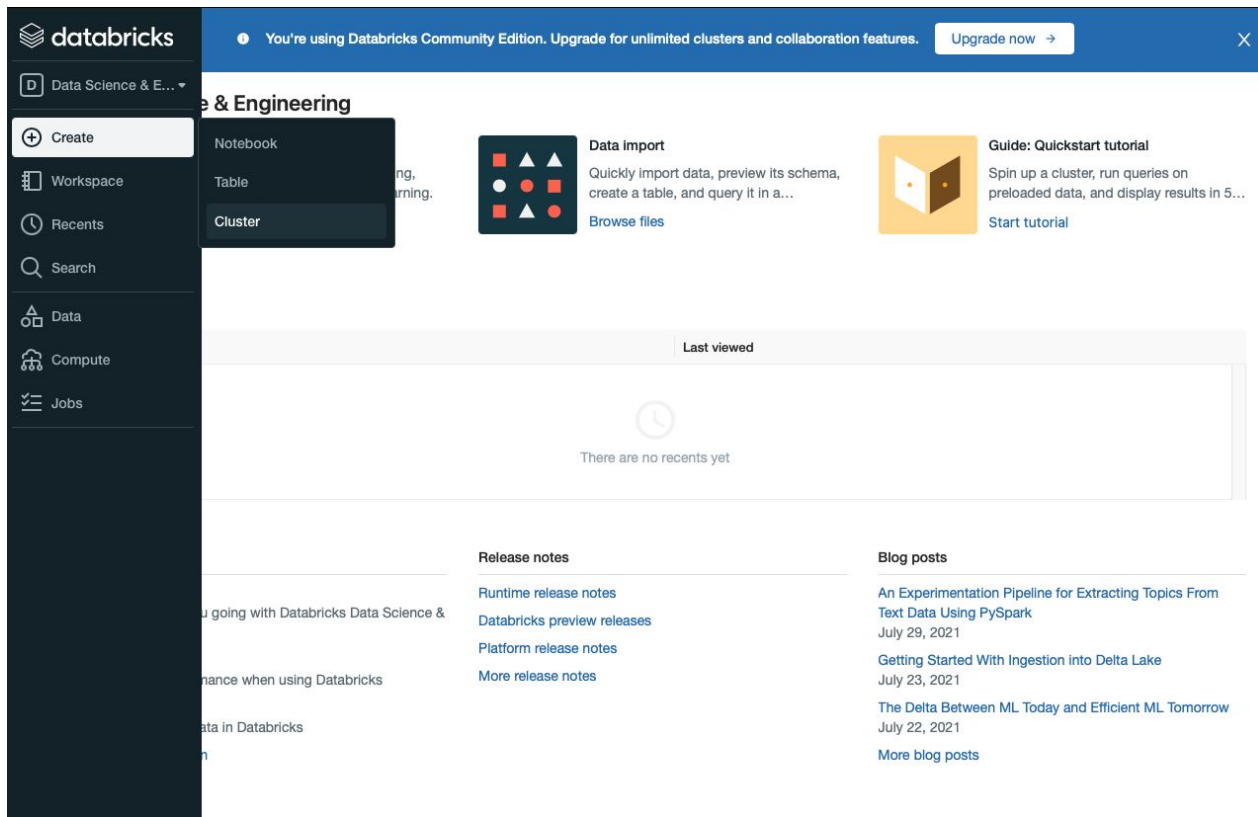
Login to the community edition at

<https://community.cloud.databricks.com/login.html>



The screenshot shows the login interface for Databricks Community Edition. At the top, the Databricks logo and 'Community Edition' text are displayed. The main heading is 'Sign In to Databricks Community Edition'. Below this, there are two input fields: the first for an email address (containing '@andrew.cmu.edu') and the second for a password (masked with dots). To the right of the password field is a 'Forgot Password?' link. A blue 'Sign In' button is positioned below the input fields. At the bottom of the form, there is a link for 'New to Databricks? Sign Up.' and a footer with 'Privacy Policy | Terms of Use'.

# Creating a Cluster



The screenshot displays the Databricks web interface. On the left is a dark sidebar with the Databricks logo and navigation links: Data Science & Engineering, Create, Workspace, Recents, Search, Data, Compute, and Jobs. The 'Create' menu is open, showing options for Notebook, Table, and Cluster, with 'Cluster' highlighted. The main content area has a blue header with a notification: 'You're using Databricks Community Edition. Upgrade for unlimited clusters and collaboration features.' with an 'Upgrade now' button. Below the header, there are three cards: 'Data import' (with a grid icon and a 'Browse files' link), 'Guide: Quickstart tutorial' (with a book icon and a 'Start tutorial' link), and a 'Last viewed' section showing a clock icon and the text 'There are no recents yet'. At the bottom, there are two columns of links: 'Release notes' (Runtime, Preview, Platform, and More) and 'Blog posts' (An Experimentation Pipeline..., Getting Started With Ingestion..., The Delta Between ML..., and More blog posts).



Lab 1 

Notebooks (0)

## Event log

## Driver logs

## Apps

Spark cluster UI - Master ▼

11.1 (includes Apache Spark 3.3.0, Scala 2.12)

Community Optimized

15.3 GB Memory, 2 Cores, 1 DBU

Free 15 GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For [more configuration options](#), please [upgrade your Databricks subscription](#).

## Spark

JDBC/ODBC

## Permissions

us-west-2b

# Installing Required Packages

Clusters / Lab 1

Lab 1

Edit Clone Restart Terminate Delete

Configuration Notebooks Libraries Event log Spark UI Driver Logs Metrics Apps

Uninstall Install new

<input type="checkbox"/>	Name	Type	Status	Source
--------------------------	------	------	--------	--------

## Create Library

Library Source

Upload DBFS/S3 PyPI Maven CRAN

Package

PyPI package (simplejson or simplejson==3.8.0)

nose

Repository ⓘ

Optional

Create Cancel

# Uploading/Importing the Homework

The screenshot displays the Databricks web interface. On the left is a dark sidebar with the 'databricks' logo and navigation links: 'Data Science & E...', 'Create', 'Workspace' (highlighted with a red bracket), 'Recents', 'Search', 'Data', 'Compute', and 'Jobs'. The main area is titled 'Workspace' and contains dropdown menus for 'Workspace', 'Shared', and 'Users'. Below these, a 'Users' dropdown menu is open, showing a list of users with 'arjain@andrew.cmu.edu' selected. A context menu is visible over the user list, with options: 'Create', 'Clone', 'Import' (highlighted by a red arrow), 'Export', 'Permissions', and 'Copy Link Address'. To the right, the 'Import Notebooks' dialog box is open. It has a title bar and a subtitle. Below the title, it says 'Import from:' followed by two radio buttons: 'File' (selected) and 'URL'. A large light gray box contains the text 'Drop file to upload, or [browse](#).' Below this box, it lists 'Accepted formats: .dbc, .scala, .py, .sql, .r, .ipynb, .Rmd, .html' and a note '(To import a library, such as a jar or egg, [click here](#))'. At the bottom right of the dialog are two buttons: 'Cancel' and 'Import'.

# Interacting with Notebooks

- Attach notebook to the cluster

Recitation\_1 Python



The screenshot shows the top toolbar of a Databricks notebook. The status is 'Detached'. The menu bar includes 'File', 'Edit', and 'View: Standard'. Below the toolbar, the 'Attach' section is visible, showing a green checkmark icon next to 'Lab 1'. The configuration details for 'Lab 1' are: 15.25 GB | 2 Cores | DBR 11.1 | Spark 3.3.0 | Scala 2.12. A share icon is located to the right of the configuration details.

Detached

File Edit View: Standard

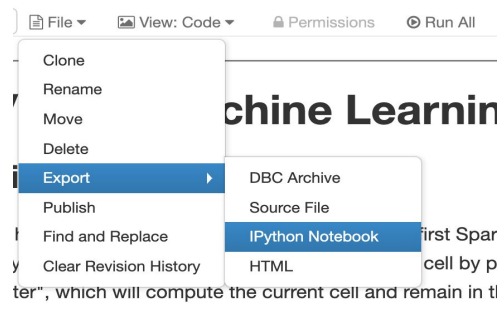
**Attach:**

✓ Lab 1

15.25 GB | 2 Cores | DBR 11.1 | Spark 3.3.0 | Scala 2.12

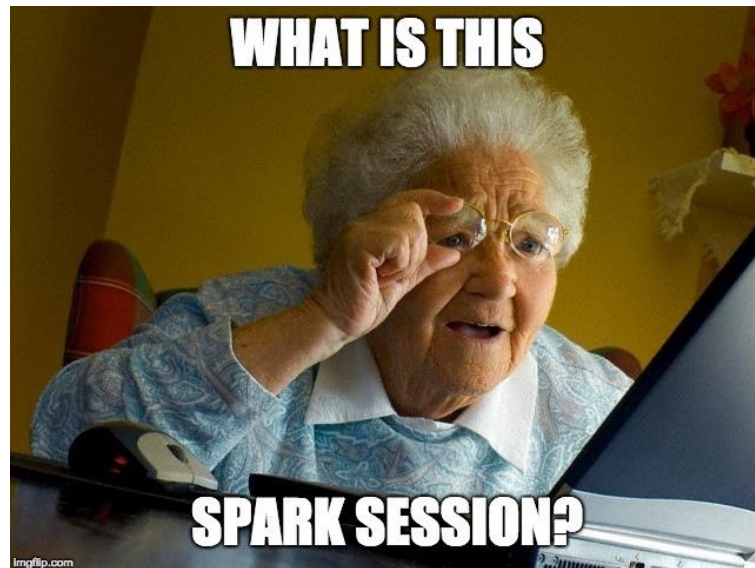
# Exporting the Homework

- After completion of assignment, export the notebook as an iPython Notebook and submit to Gradescope for grading.



# Important notes about clusters

- Please make sure that the following versions are used:
  - **Spark:** 3.3.0
  - **Scala:** 2.12
  - **Runtime:** 11.1
- For all coding, please use Python3 syntax.
- Launching a cluster can take some time (a few minutes).
- The cluster status should be “active” before running any cells.
- Community edition only allows for a single node cluster which should be sufficient for completion of the homeworks.
- Community edition clusters will automatically terminate after an idle period of two hours.



# PySpark

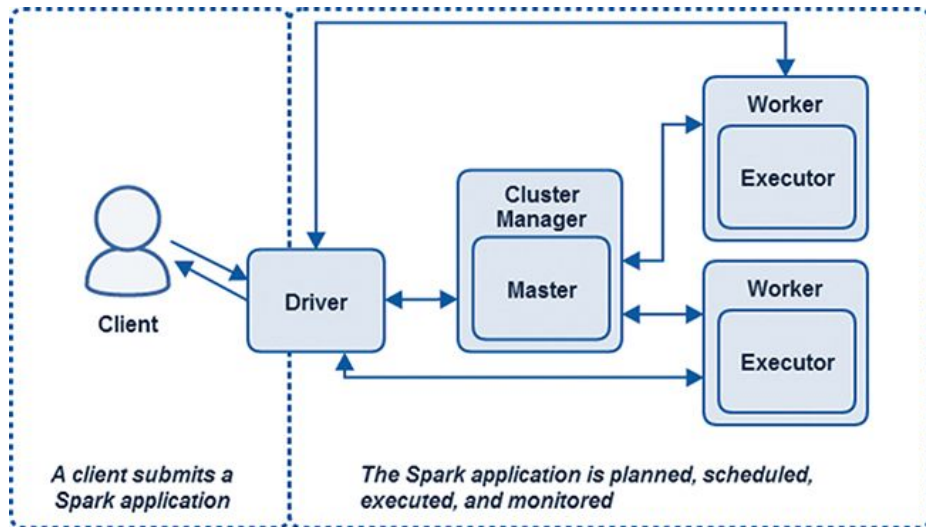
- We are using the Python programming interface to Spark (pySpark)
- Where can I run PySpark code?
  - Locally on your laptop!
  - Databricks
  - Zeppelin Notebooks
  - many more ...



# PySpark (local installation)

- To get HWs running locally, you'll need to install:
  - VSCode
    - Jupyter notebook extension
  - Python3
    - `pip install ipykernel`
    - `pip install pyspark==3.1.2`
    - `pip install findspark`
    - `pip install nose`
  - Java8
- Note: displaying math equations might be buggy
- We won't be able to support / debug local installations

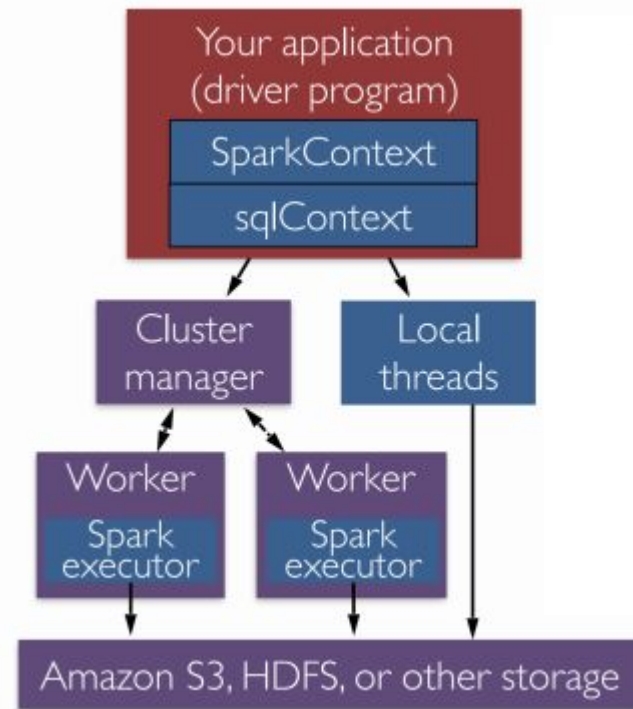
# Spark Architecture



- **Driver program:** This is the node you're interacting with when you're writing Spark programs. Creates Spark Context to establish the connection to Spark Execution Environment
- **Cluster Manager:** Allocates resources across cluster, manages scheduling. e.g., YARN/Mesos
- **Worker program:** The program that runs on the worker nodes.
- **Executor:** What actually does the work/tasks on each cluster node.

# Execution of a Spark Program

- The driver program runs the Spark application, which creates a SparkContext upon start-up.
- The SparkContext connects to a cluster manager (e.g., Mesos/YARN) which allocates resources.
- Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for your application.
- Next, driver program sends your application code to these executors.
- Finally, SparkContext sends tasks for the executors to run.



# Spark APIs

1. **RDD (Resilient Distributed Datasets):** Think distributed list.
  2. **DataFrames:** SQL-like structured datasets with query operations.
  3. **Datasets:** A mixture of RDDs and DataFrames in terms of the operations that are available on them.
- The homeworks in this class will focus on using the RDD and the Dataframe APIs.
  - The dataframe API has lots of parallels from Pandas Dataframes.

# RDDs vs Dataframes

## RDDs

- RDDs are **immutable distributed collection** of elements of your data that can be stored in memory or disk across a cluster of machines.
- The data is partitioned across machines in your cluster that can be operated in parallel with a low-level API that offers **transformations** and **actions**.
- RDDs are **fault tolerant** as they track data lineage information to rebuild lost data automatically on failure.

## DataFrames

- Like an RDD, a DataFrame is an immutable distributed collection of data.
- Unlike an RDD, data is organized into named columns, like a table in a relational database.
- Dataframes allow for a higher-level abstraction by imposing a structure onto a distributed collection of data.

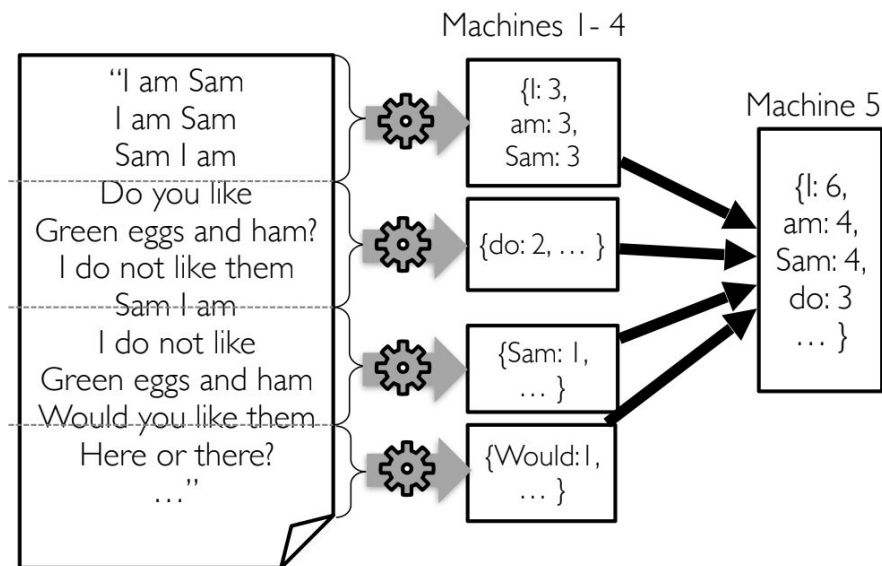
# Transformations vs Actions

- **Transformations:** Return new RDDs as a result.
  - They are lazy, their result RDD is not immediately computed.
- **Actions:** Compute a result based on an RDD, and either returned or saved to an external storage system (e.g., HDFS)
  - They are eager, their result is immediately computed.

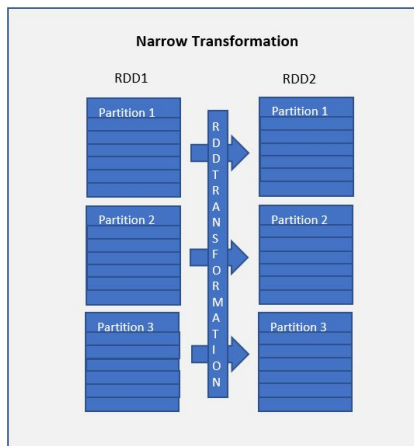
Transformation	Description
<code>map(func)</code>	return a new distributed dataset formed by passing each element of the source through a function <i>func</i>
<code>filter(func)</code>	return a new dataset formed by selecting those elements of the source on which <i>func</i> returns true
<code>distinct([numTasks])</code>	return a new dataset that contains the distinct elements of the source dataset
<code>flatMap(func)</code>	similar to map, but each input item can be mapped to 0 or more output items (so <i>func</i> should return a Seq rather than a single item)

Action	Description
<code>reduce(func)</code>	aggregate dataset's elements using function <i>func</i> . <i>func</i> takes two arguments and returns one, and is commutative and associative so that it can be computed correctly in parallel
<code>take(n)</code>	return an array with the first <i>n</i> elements
<code>collect()</code>	return all the elements as an array <b>WARNING: make sure will fit in driver program</b>
<code>takeOrdered(n, key=func)</code>	return <i>n</i> elements ordered in ascending order or as specified by the optional key function

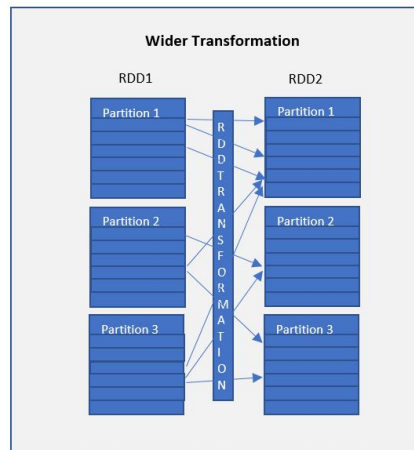
# Example: Map vs. Reduce



# Narrow vs Wide Transformations



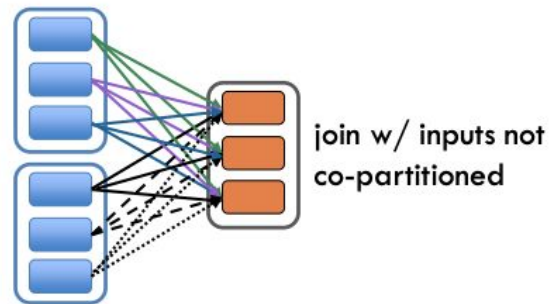
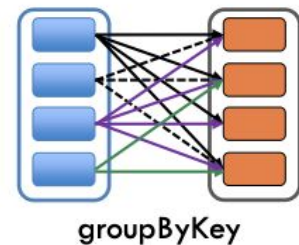
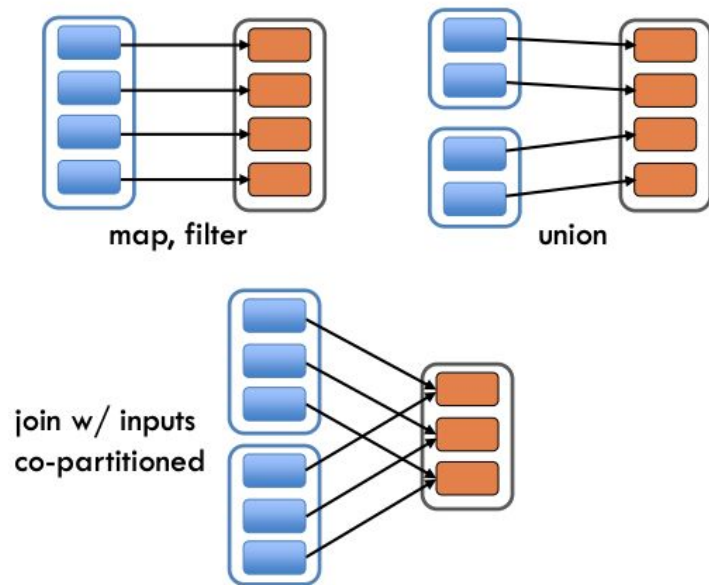
- Each partition of the parent RDD is used by at most one partition of the child RDD.
- Fast, No Data Shuffling



- Multiple child RDD partitions may depend on a single parent RDD partition.
- Slow, Data Shuffling Required



# Narrow vs Wide Transformations



# How can we create an RDD?

- From a SparkContext (or SparkSession)

```
# parallelizing data collection  
my_list = [1, 2, 3, 4, 5]  
my_list_rdd = sc.parallelize(my_list)
```

- Referencing to the external data file stored

```
## 2. Referencing to external data file  
file_rdd = sc.textFile("path_of_file")
```

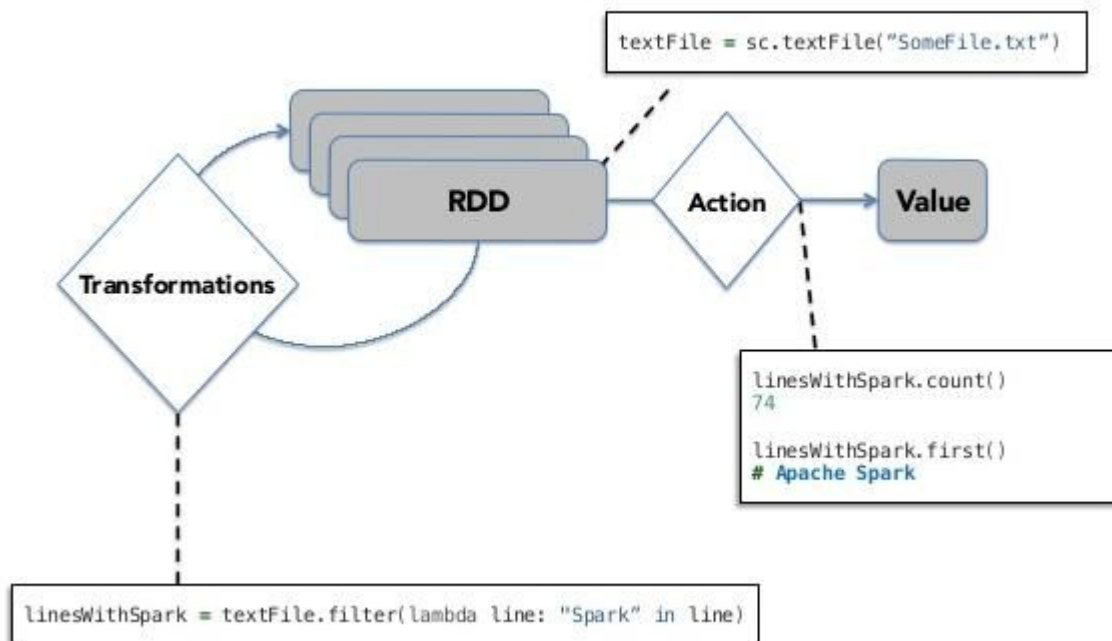
- Transforming an existing RDD

## What can you do when you have an RDD?

A small cheat-sheet can be found [here](#) with some nice examples.



# Interacting with RDDs



# Lambda vs Regular Functions

## Lambda Functions

- Evaluates only a single expression
- No name associated

```
lambda x : x + x
```

## Regular Function

- Can have multiple expressions
- Must have name associated

```
def adder (x, y):  
    return x + y
```

# Example

What does the following code snippet do?

```
peopleRdd = sc.parallelize (["bob", "alice", "bill"])
```

```
words_filter = peopleRdd.filter(lambda x: 'i' in x)
```

```
filtered = words_filter.collect()
```

# Example

What does the following code snippet do?

```
peopleRdd = sc.parallelize(["bob", "alice", "bill"]) ← creates the RDD.
```

```
words_filter = peopleRdd.filter(lambda x: 'i' in x) ← nothing, since this is a transformation.
```

```
filtered = words_filter.collect() ← the DAG is executed since an action is called.
```

## Example #2

What does the following code snippet do on the driver node?

```
peopleRdd = sc.parallelize (["bob", "alice", "bill"])
```

```
peopleRdd.foreach(print)
```

```
peopleRdd.take(2)
```

# Example #2

What does the following code snippet do on the driver node?

```
peopleRdd = sc.parallelize (["bob", "alice", "bill"])
```

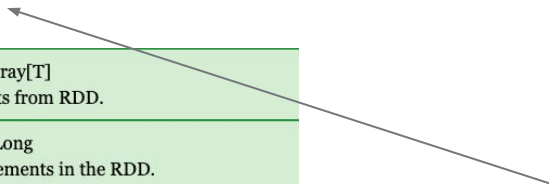
```
peopleRdd.foreach(print)
```

```
peopleRdd.take(2)
```



## On the driver: Nothing. Why?

*foreach is an action, which doesn't return anything. Therefore, it is eagerly executed on the executors, not the driver. Therefore, any calls to print are happening on the stdout of the worker nodes and are thus not visible in the stdout of the driver node.*



## What about when take is called? Where will the array of people end up?

*When an action returns a result, it returns it to the driver node.*

collect	collect(): Array[T] Return all elements from RDD.
count	count(): Long Return the number of elements in the RDD.
take	take(num: Int): Array[T] Return the first num elements of the RDD.
reduce	reduce(op: (A, A) -> A): A Combine the elements in the RDD together using op function and return result.
foreach	foreach(f: T -> Unit): Unit Apply function to each element in the RDD.



## Example #3

What does the following code snippet do on the driver node?

```
peopleRdd = sc.parallelize (["bob", "alice", "bill"])
```

```
newRDD = peopleRDD.foreach(print)  
print(type(newRDD))
```

```
newRDD = peopleRDD.map(print)  
print(type(newRDD))  
newRDD.take(2)
```

```
newRDD = peopleRDD.map(lambda x: "Dr." + x)  
newRDD.take(2)
```

## Example #3

What does the following code snippet do on the driver node?

```
peopleRdd = sc.parallelize(["bob", "alice", "bill"])
```

```
newRDD = peopleRDD.foreach(print)
print(type(newRDD))
```



**<class 'NoneType'>**

*foreach is an action, which doesn't return anything. Therefore the type of the returned is NoneType.*

```
newRDD = peopleRDD.map(print)
print(type(newRDD))
newRDD.take(2)
```

```
newRDD = peopleRDD.map(lambda x: "Dr." + x)
newRDD.take(2)
```

## Example #3

What does the following code snippet do on the driver node?

```
peopleRdd = sc.parallelize(["bob", "alice", "bill"])
```

```
newRDD = peopleRDD.foreach(print)
print(type(newRDD))
```



**<class 'NoneType'>**

*foreach is an action, which doesn't return anything. Therefore the type of the returned is NoneType.*

```
newRDD = peopleRDD.map(print)
print(type(newRDD))
newRDD.take(2)
```



**<class 'pyspark.rdd.PipelinedRDD'>**  
**[None, None]**

*map is a transformation, and the type of the returned is another RDD where the values are modified. However, since the print() function doesn't return anything, the output will be [None, None].*

```
newRDD = peopleRDD.map(lambda x: "Dr." + x)
newRDD.take(2)
```

## Example #3

What does the following code snippet do on the driver node?

```
peopleRdd = sc.parallelize (["bob", "alice", "bill"])
```

```
newRDD = peopleRDD.foreach(print)  
print(type(newRDD))
```



**<class 'NoneType'>**

*foreach is an action, which doesn't return anything. Therefore the type of the returned is NoneType.*

```
newRDD = peopleRDD.map(print)  
print(type(newRDD))  
newRDD.take(2)
```



**<class 'pyspark.rdd.PipelinedRDD'>**

**[None, None]**

*map is a transformation, and the type of the returned is another RDD where the values are modified. However, since the print() function doesn't return anything, the output will be [None, None].*

```
newRDD = peopleRDD.map(lambda x: "Dr." + x)  
newRDD.take(2)
```



**['Dr. bob', 'Dr. alice']**

*map is a transformation, so it returned a new RDD with "Dr." added before each name.*

# Spark Execution

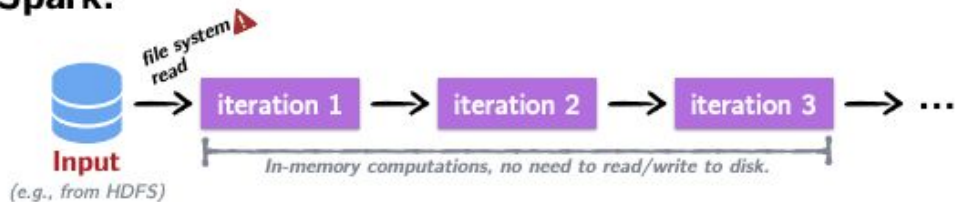
- Spark uses lazy evaluation!
  - Lazy evaluation means nothing executes. Spark saves recipe for transforming source.
  - Results are not computed right away – instead, Spark remembers set of transformations applied to base data set.
  - The way Spark “remembers” is by creating a DAG and then it tries and optimises the DAG to optimize the use of resources (like bandwidth, memory etc).
- It enables Spark to optimize the required operations.
- It enables Spark to recover from failures and slow workers.
- Spark supports in-memory computation which enables it to be a lot ***faster for iterations.***

# Spark Execution

## Iteration in Hadoop:



## Iteration in Spark:



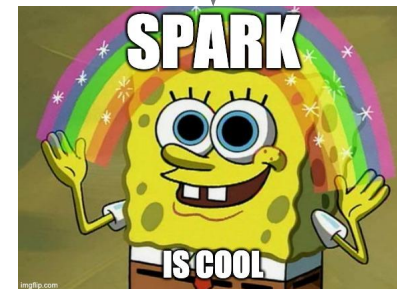
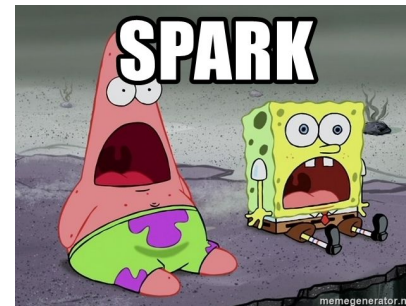
# Cache/Persistence

- By default, RDDs are recomputed each time you run an action on them. This can be expensive (in time) if you need to use a dataset more than once.
- Spark allows you to control what can be cached into memory (using the `.cache()` or the `.persist()` API).
- One of the most common performance bottlenecks arises from unknowingly re-evaluation several transformations.  
This will be tested in some of the assignments.

Reference Link - <https://data-flair.training/blogs/apache-spark-rdd-persistence-caching/>

# FAQs/Helpful Debugging Instructions

- In case tests don't pass, try running all the cells in the notebook.
- If you see a *name-not-defined error*, It could be because of your code timing out on the grader.
- Timeouts can occur due to inefficient code (for example, calling `collect()` on the entire dataset).
- Local tests are *not* comprehensive. They are there to assist you in building your code. You only get points if the assignment runs on the autograder.





# Appendix

## RDDs vs Dataframes vs Datasets

	RDDs	Dataframes	Datasets
<b>Data Representation</b>	RDD is a distributed collection of data elements without any schema.	It is also the distributed collection organized into the named columns	It is an extension of Dataframes with more features like type-safety and object-oriented interface.
<b>Optimization</b>	No in-built optimization engine for RDDs. Developers need to write the optimized code themselves.	It uses a catalyst optimizer for optimization.	It also uses a catalyst optimizer for optimization purposes.
<b>Projection of Schema</b>	Here, we need to define the schema manually.	It will automatically find out the schema of the dataset.	It will also automatically find out the schema of the dataset by using the SQL Engine.
<b>Aggregation Operation</b>	RDD is slower than both Dataframes and Datasets to perform simple operations like grouping the data.	It provides an easy API to perform aggregation operations. It performs aggregation faster than both RDDs and Datasets.	Dataset is faster than RDDs but a bit slower than Dataframes.

# Group By Key Example - PySpark

