

# MSDS 694

# Distributed Computing

---

DIANE WOODBRIDGE, PH.D



# Announcement

---

Feel free to dress-up for Oct 28th class for Halloween 🎃

# Contents

---

Class Intro

Environment Setup

Code Standard

Big Data and Distributed Computing

MapReduce Concept



# Contents

---

## Class Intro

Environment Setup

Code Standard

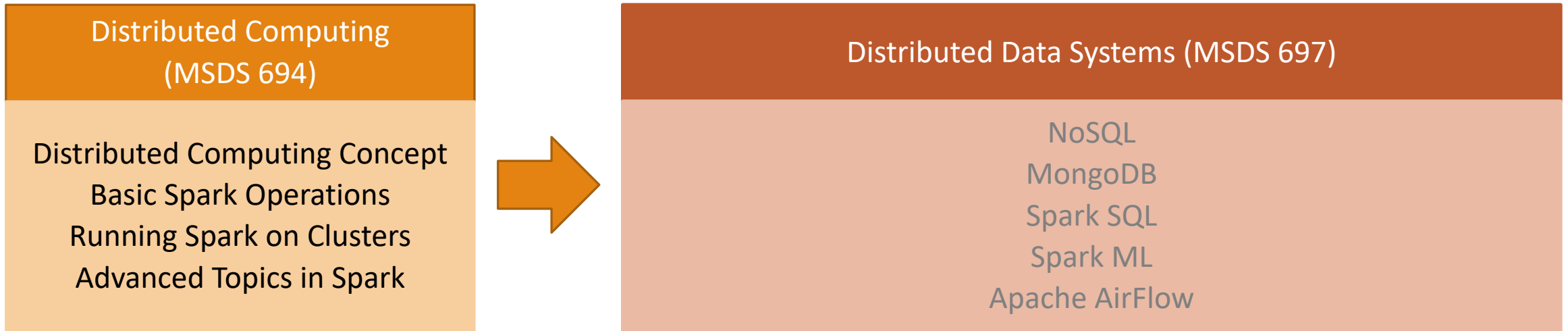
Big Data and Distributed Computing

MapReduce Concept



# About Distributed Computing(MSDS694)

---



Plus, some practicum companies require Spark.



# Possible Outcomes After MSDS697 (Spring)

---

2017

## Machine Learning-based Product Recommendation using Apache Spark

Lin Chen \*, Rui Li \*, Yige Liu \*, Ruixuan Zhang \*, Diane Myung-kyung Woodbridge  
{lchen74,rli33,yliau225,rzhang45,dwoodbridge}@usfca.edu  
*Master of Science in Analytics (MSAN) Program,  
University of San Francisco,  
San Francisco, California*

2018

## Forecasting Smart Meter Energy Usage using Distributed Systems and Machine Learning

Chris Dong\*, Lingzhi Du\*, Feiran Ji\*, Zizhen Song\*, Yuedi Zheng\*,  
Paul Intrevado, Diane Myung-kyung Woodbridge  
{cadong,ldu4,fji3,zsong11,yzheng41,pintrevado,dwoodbridge}@usfca.edu  
Data Science Program  
University of San Francisco

## Distributed Data Analytics Framework for Smart Transportation

Alexander J. Howard\*, Tim Lee\*, Sara Mahar\*, Paul Intrevado, Diane Myung-kyung Woodbridge  
{ajhoward7,semahar2,tlee,pintrevado,dwoodbridge}@usfca.edu  
Data Science Program  
University of San Francisco



# Possible Outcomes After MSDS697 (Spring)

---

2019

- A Scalable Smartwatch-based Medication Adherence Monitoring System using Distributed Machine Learning. Donya Fozoonmayeh, Hai Vu Le, Ekaterina Wittfoth, Chong Geng, Natalie Ha, Jingjue Wang, Maria Vasilenko, Yewon Ahn, Diane Myung-kyung Woodbridge. Journal of Medical Systems (JOMS)
- Predicting Unethical Physician Behavior At Scale: A Distributed Computing Framework. Anastasia Quinn Keck, Miguel Romero, Robert Sandor, Diane Myung-kyung Woodbridge, Paul Intrevado. IEEE Smart World Congress. August 2019.
- A Scalable and Reliable Model for Real-time Air Quality Prediction. Liying Li, Zhi Li, Lara G. Reichmann, Diane Myung-kyung Woodbridge. IEEE Smart World Congress. August 2019.
- The Impact of Bike-Sharing Ridership on Air Quality: A Scalable Data Science Framework. Nina Hua, Victoria Suarez, Rebecca Reilly, Philip Trinh, Paul Intrevado, Diane Myung-kyung Woodbridge. IEEE International Conference on Smart City Innovations. August 2019.
- Distributed Data Analytics Framework for Cluster Analysis of Parking Violation. Nan Lin, Evan Liu, Fiorella Tenorio, Xi Yang, Diane Woodbridge. IEEE International Conference on Smart City Innovations. August 2019.
- Scalable Real-time Prediction and Analysis of San Francisco Fire Department Response Times. Xu Lian, Sarah Melancon, Jon-Ross Presta, Adam Reevesman, Brian J. Spiering, Diane Myung-kyung Woodbridge. IEEE International Conference on Ubiquitous Intelligence and Computing. August 2019.
- Scalable Motor Movement Recognition from Electroencephalography using Machine Learning. Aditi Sharma, Shivee Singh, Brian Wright, Alan Perry, Diane Myung-Kyung Woodbridge, Abbie Popa. IEEE International Workshop on Integrated Smart Healthcare (WISH). July 2019.
- A Scalable Automated Diagnostic Feature Extraction System for EEGs. Prakhar Agrawal, Divya Bhargavi, Gokul Krishna G, Xiao Han, Neha Tevathia, Abbie Popa, Nicholas Ross, Diane Myung-Kyung Woodbridge, Barbie Zimmerman-Bier and William Bosl. IEEE International Workshop on Medical Computing (MediComp). July 2019.



# Possible Outcomes After MSDS697 (Spring)

---

## 2020

- A Machine Learning Approach to Detecting Low Medication State with Wearable Technologies. Andy Cheon, Stephanie Yeoju Jung, Collin Prather, Matt Sarmiento, Kevin Wong, Diane Woodbridge. International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). July 2020.
- Sensor Selection for Activity Classification at Smart Home Environments. Nithish Bolleddula, Geoffrey Hung, Daren Ma, Hoda Noorian, Diane Woodbridge. International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). July 2020.

## 2021

- Machine Learning-based Meal Detection Using Continuous Glucose Monitoring on Healthy Participants: An Objective Measure of Participant Compliance to Protocol. Victor Palacios, Diane Myung-kyung Woodbridge, Jean L. Fry. International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). November 2021.
- Depression Level Prediction in People with Parkinson's Disease during the COVID-19 Pandemic. Hashneet Kaur, Patrick Ka-Cheong Poon, Sophie Yuefei Wang, Diane Myung-kyung Woodbridge. International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). November 2021.

## 2022

- A Spotify Song and Playlist Recommendation Engine. Lucas De Oliveira, Chandrish Ambati, Anish Mukherjee. MongoDB Developer Code Example. <https://www.mongodb.com/developer/code-examples/python/song-recommendations-example-app/>

◦



# About Distributed Computing (MSDS694)

---

## Learning Outcomes

- Understand needs and concepts of distributed computing.
- Understand Apache Spark programming basics.
- Gain experience with Apache Spark data processing including Spark APIs.
- Be competent to work with Spark on a distributed computing environment including Google Cloud Services (GCP) and Databricks.



# About Distributed Computing (MSDS694)

---

## Class Schedule

- Session 01 - Friday 10:00 AM - 12:00 PM (Room 527)
- Session 02 - Friday 2:00 PM - 4:00 PM (Room 529)

## Office Hour

- Tuesday 9:15 - 10:00 AM (Room 606 (Or via Zoom, if you request in advance - Give me at least 12 hours.))



# About Distributed Computing (MSDS694)

---

## Class Overview

Week	Overview
Week 1 (October 21)	Class Intro, Environment Setup, What/Why Distributed Computing? Concepts of Distributed Computing
Week 2 (October 28)	Run Spark on a Single-node Environment RDD and Basic Spark Operations (1)
Week 3 (November 4)	Basic Spark Operations (2)
Week 4 (November 11)	Pair RDD Operations
Week 5 (November 18)	Run Spark on a Multi-node Environment
Week 6 (November 25)	No Class
Week 7 (December 2)	Advanced Topics - Improving Spark Performance



# About Distributed Computing (MSDS694)

Spark Practice	Running Spark on a Cluster (Databricks/GCP)
Understanding and Practicing Spark	Pair RDD Operations (Transformation and Action)
	RDD Operations (Transformation and Action)
	Create Resilient Distributed Dataset (RDD)
Understanding Distributed Computing	Spark Overview
	MapReduce Concept
	Concepts of Distributed Computing
Development Basic	Environment Setup



# About Distributed Computing (MSDS694)

## Evaluation Criteria

- Attendance and Professionalism - 10 %
- Individual Programming Assignment - 25 %
- [Group Project](#) - 20%
- Quiz - 45% (November 11, December 2nd)

Grader : Phillip Navo ([panavo@usfca.edu](mailto:panavo@usfca.edu))

<b>Oct 28</b>	Spark Installation
<b>Nov 4</b>	Programming Assignment
<b>Dec 3</b>	Programming Assignment

<b>Nov 5</b>	Select Data Sets
<b>Nov 28</b>	Loading and Saving Data
<b>Dec 14</b>	Data Processing and Visualization on Databricks



# Professionalism

---

## Class Attendance

- Please close your laptop other than the exercise sessions.
- No cell phones, social media, slack, or texting during the class .

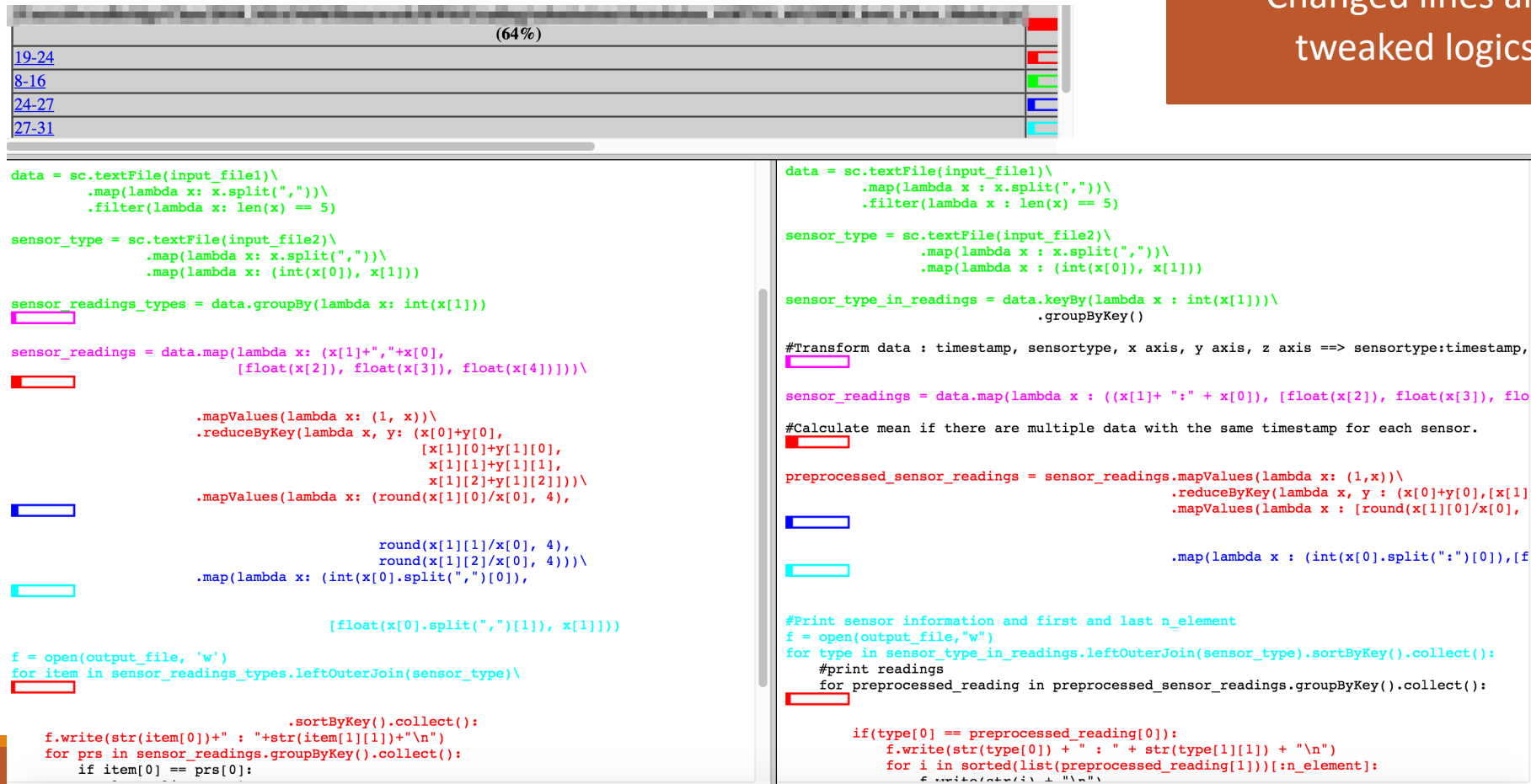
## Assignment

- No late submissions are allowed.
- Do not share any homework and exam files - All the codes including the last 6 years will be tested by Moss (Measure Of Software Similarity).
  - I caught 9 students plagiarized last year in this class and some had to leave the program.
- Make sure that your code runs in Python 3.10 and Pyspark 3.3.

# Course Evaluation

Please no plagiarism! – Zero tolerance.

Changed lines and  
tweaked logics



(64%)	
19-24	
8-16	
24-27	
27-31	

```
data = sc.textFile(input_file1)\
    .map(lambda x: x.split(","))\
    .filter(lambda x: len(x) == 5)

sensor_type = sc.textFile(input_file2)\
    .map(lambda x: x.split(","))\
    .map(lambda x: (int(x[0]), x[1]))

sensor_readings_types = data.groupBy(lambda x: int(x[1]))

sensor_readings = data.map(lambda x: (x[1]+","+x[0],
    [float(x[2]), float(x[3]), float(x[4])]))\

    .mapValues(lambda x: (1, x))\
    .reduceByKey(lambda x, y: (x[0]+y[0],
        [x[1][0]+y[1][0],
        x[1][1]+y[1][1],
        x[1][2]+y[1][2]]))\

    .mapValues(lambda x: (round(x[1][0]/x[0], 4),

        round(x[1][1]/x[0], 4),
        round(x[1][2]/x[0], 4))))\

    .map(lambda x: (int(x[0].split(",")[0]),

        [float(x[0].split(",")[1]), x[1]]))

f = open(output_file, 'w')
for item in sensor_readings_types.leftOuterJoin(sensor_type)\

    .sortByKey().collect():
    f.write(str(item[0])+" : "+str(item[1][1])+"\n")
for prs in sensor_readings.groupByKey().collect():
    if item[0] == prs[0]:
```

```
data = sc.textFile(input_file1)\
    .map(lambda x: x.split(","))\
    .filter(lambda x: len(x) == 5)

sensor_type = sc.textFile(input_file2)\
    .map(lambda x: x.split(","))\
    .map(lambda x: (int(x[0]), x[1]))

sensor_type_in_readings = data.keyBy(lambda x: int(x[1]))\
    .groupByKey()

#Transform data : timestamp, sensortype, x axis, y axis, z axis ==> sensortype:timestamp,

sensor_readings = data.map(lambda x: ((x[1]+ ":" + x[0]), [float(x[2]), float(x[3]), flo

#Calculate mean if there are multiple data with the same timestamp for each sensor.

preprocessed_sensor_readings = sensor_readings.mapValues(lambda x: (1,x))\
    .reduceByKey(lambda x, y: (x[0]+y[0],[x[1]

    .mapValues(lambda x: [round(x[1][0]/x[0],

        .map(lambda x: (int(x[0].split(":")[0]),[f

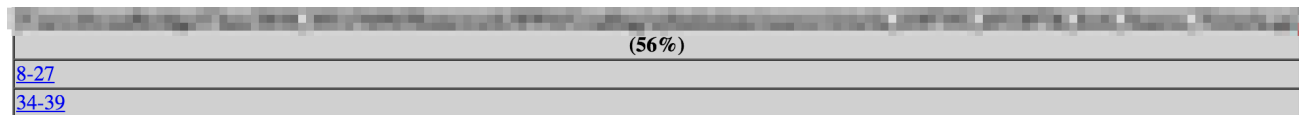
#Print sensor information and first and last n_element
f = open(output_file,"w")
for type in sensor_type_in_readings.leftOuterJoin(sensor_type).sortByKey().collect():
    #print readings
    for preprocessed_reading in preprocessed_sensor_readings.groupByKey().collect():

        if(type[0] == preprocessed_reading[0]):
            f.write(str(type[0]) + " : " + str(type[1][1]) + "\n")
            for i in sorted(list(preprocessed_reading[1]))[:n_element]:
                f.write(str(i) + "\n")
```

# Course Evaluation

Please no plagiarism! – Zero tolerance.

Changed lines and  
variable names



```
timeseries = sc.textFile(input_file1)
sensors = sc.textFile(input_file2)

timeseries = timeseries.filter(lambda x: len(x) > 0) # remove empty lines
timeseries = timeseries.map(lambda x: x.split(','))
timeseries = timeseries.map(lambda x: [float(num) for num in x])
# ((timestamp, sensorID), [x,y,z])
timeseries = timeseries.map(lambda x: ((x[0], x[1]), x[2:5]))

countbyTimeStamp = timeseries.countByKey()

timeseries = timeseries.reduceByKey(
    lambda x, y: [num_x + num_y for num_x, num_y in zip(x, y)])
timeseries = timeseries.map(
    lambda x: (x[0], [round(num/countbyTimeStamp[x[0]], 4) for num in x[1]]))

sensors = sensors.map(lambda x: x.split(','))
sensors = sensors.map(lambda x: (int(x[0]), x[1]))

timeseries = timeseries.map(
    lambda x: (x[0][1], [x[0][0], x[1]])) # (sensorID,[timestamp, [x,y,z]])

sensor_name_join = timeseries.leftOuterJoin(sensors)
# ((sensor_ID, Name),[timestamp,[x,y,z]])
sensor_name_join = sensor_name_join.map(
    lambda x: ((x[0], x[1][1]), x[1][0]))

sensor_name_join = sensor_name_join.groupByKey()
sensor_name_join = sensor_name_join.mapValues(
    lambda x: sorted(x, key=lambda y: y[0]))

with open(output_file, 'w') as f:
    for sensorID in sensor_name_join.sortByKey().collect():
        f.write(str(int(sensorID[0][0])) + ' : ' + str(sensorID[0][1]) + '\n')

from user_definition import *
# DO NOT ADD OTHER LIBRARIES/PACKAGES!

conf = SparkConf().setAppName(app_name)
sc = SparkContext(conf=conf).getOrCreate()

ts = sc.textFile(input_file1)
sensor = sc.textFile(input_file2)

ts = ts.filter(lambda x: len(x) > 1)
ts = ts.map(lambda x: x.split(","))
ts = ts.map(lambda x: [float(num) for num in x])
ts = ts.map(lambda x: ((x[0], x[1]), x[2:5]))
count = ts.countByKey()
ts = ts.reduceByKey(lambda x, y: [(x+y) for x, y in zip(x, y)])
ts = ts.map(lambda x: (x[0], [round(num/count[x[0]], 4) for num in x[1]]))
sensor = sensor.map(lambda x: x.split(","))
sensor_num = sensor.map(lambda x: (int(x[0]), x[1]))
ts_group_sensor = ts.map(lambda x: [x[0][1], (x[0][0], x[1])]).groupByKey()
ts_group_sensor = ts_group_sensor.map(lambda x: [x[0], list(x[1])])
full = ts_group_sensor.leftOuterJoin(sensor_num)

full = full.map(lambda x: [(x[0], x[1][1]), x[1][0]]).sortByKey()
full = full.mapValues(lambda x: sorted(x, key=lambda y: y[0]))

with open(output_file, "w") as f:
    for sensor in full.collect():
        f.write(str(int(sensor[0][0])) + " : " + str(sensor[0][1]) + '\n')
        for value in sensor[1][:-n_element]:
            f.write(str(list(value)) + '\n')
        f.write('...' + '\n')
        for value in sensor[1][n_element:]:
            f.write(str(list(value)) + '\n')

sc.stop()
```



# Textbook

---

Spark Online Documentation, <http://spark.apache.org/docs/latest/>

Google Cloud Online Documentation, <https://cloud.google.com/docs>

Databricks Online Documentation, <https://docs.databricks.com/>

Zecevic, Petar, et al. Spark in Action, Manning, 2016.



# Student Engagement

---

Example Data :

<https://github.com/dianewoodbridge/2022-msds694-example>

Poll : <https://pollev.com/msds>

Piazza : [piazza.com/usfca/fall2022/msds694](https://piazza.com/usfca/fall2022/msds694)

## Canvas

- Under each module (week), there are learning outcome, slides, homework, example tests, tests, etc.
- For some assignment, I am going to upload videos.
- For most of the lectures, I am going to publish recorded sessions – this does NOT mean that you can skip a class or not pay attention to the class.



# Student Engagement

---


## Canvas

### ▼ Week1 - Spark Installation and Introduction

 **Week 1 - Learning Outcome**

 **Week 1 - Slides**

 **HW1 - Code Style Exercise**  
Oct 25 | 3 pts

 **HW2 - Spark Installation**  
Oct 25 | 3 pts

 **Spark Installation**

 **HW 2 Instruction**

# Student Engagement

## Canvas

- For some assignment, I am going to upload videos.

[ISAN-694-02](#) > [Modules](#) > [Week1 - Spark Installation and Introduction](#) > Spark Installation

Spark Installation

Spark runs on Java 8+, Python 2.7+/3.4+ and R 3.1+. For the Scala API, Spark 2.2.0 uses Scala 2.11. You will need to use a compatible Scala version (2.11.x).

Install JDK

```
brew update
brew tap caskroom/versions
brew cask search java
brew cask install java8
```

UNIVERSITY OF SAN FRANCISCO  
CHANGE THE WORLD FROM HERE

00:32 / 07:48

Sources CC

# Any Questions?

# Contents

---

Class Intro

**Environment Setup**

Code Standard

Big Data and Distributed Computing

MapReduce Concept



# Collaboration

---

This class (and many other classes) requires collaborations.

- Things to consider..
  - Environment Setup
  - Code Standard
  - Documentation
  - And many more!



# Environment

---



```
import pyspark
```

```
-----  
ModuleNotFoundError                                Traceback (most recent call last)  
<ipython-input-2-49d7c4e178f8> in <module>  
----> 1 import pyspark
```

```
ModuleNotFoundError: No module named 'pyspark'
```





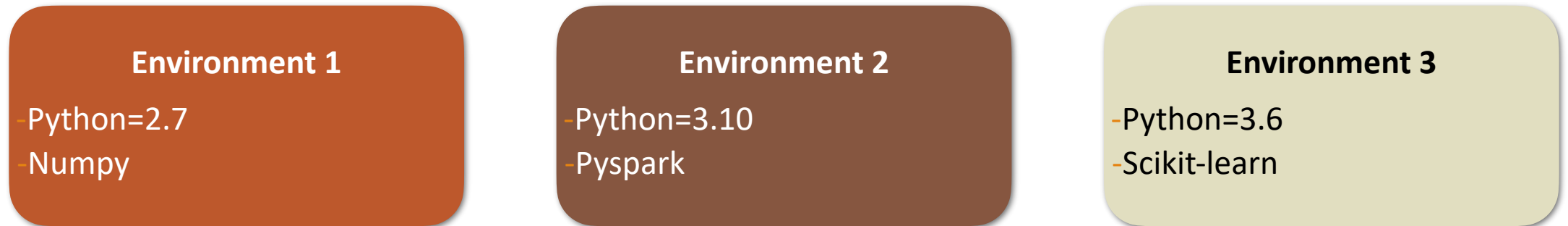
# Environment

---

A project team member might work on several projects requiring different versions of Python or other packages.

## Environment

- Keep dependencies required by different project in separate places.
- Easily switch to an environment that requires different dependencies.



# Create an Environment

## Environment

- **Create** an environment

```
conda create --name DistributedComputing python=3 -y
```

➤ `-y` : yes to all the questions.

➤ It will create "DistributedComputing" directory under `/opt/anaconda3/envs/`

```
ML-ITS-901885:2021_MSDS694_FALL dwoodbridge$ conda create --name DistributedComputing python=3 -y
Collecting package metadata (current_repodata.json): done
Solving environment: done
^[
## Package Plan ##
```

```
environment location: /Users/dwoodbridge/opt/anaconda3/envs/DistributedComputing
```

```
added / updated specs:
- python=3
```

The following packages will be downloaded:

package	build	
certifi-2020.6.20	pyhd3eb1b0_3	155 KB
python-3.7.4	py310h6cd8cb5_0	1.8 MB

# Create an Environment

## Environment

- **Activating** the environment

```
conda activate DistributedComputing
```

- **Checking** available environments and the current environment

```
conda info --envs
```

```
(DistributedComputing) ML-ITS-603436:2019_MSDS694 dwoodbridge$ conda info --envs
# conda environments:
#
base                        /Users/dwoodbridge/anaconda3
DL-2018                     /Users/dwoodbridge/anaconda3/envs/DL-2018
DistributedComputing *     /Users/dwoodbridge/anaconda3/envs/DistributedComputing
MSDS603                    /Users/dwoodbridge/anaconda3/envs/MSDS603
MSDS694                    /Users/dwoodbridge/anaconda3/envs/MSDS694
TOHP                       /Users/dwoodbridge/anaconda3/envs/TOHP
hawaii                    /Users/dwoodbridge/anaconda3/envs/hawaii
medhere                    /Users/dwoodbridge/anaconda3/envs/medhere
projectx                   /Users/dwoodbridge/anaconda3/envs/projectx
```

- **Deactivating** the environment

```
conda deactivate
```



# What is your env after deactivating the environment?

base

DistributedComputing

nothing

conda



# What is your env after deactivating the environment?

base

DistributedComputing

nothing

conda



# What is your env after deactivating the environment?

base

DistributedComputing

nothing

conda

# Example 1

---

Let's activate DistributedComputing again.

```
conda activate DistributedComputing
```



# Collaborating with Codes

---

When you push your code to the repository, it might cause errors on your collaborator's machine if she/he doesn't have all the libraries that you used.





# Create an Environment

When you're pushing your code to the git, you should add the updated .yaml file as well.

## Environment

- **Export** the environment file to share and reproduce the current environments including all the packages with corresponding versions.

```
conda env export > DistributedComputing_environment.yaml
```

```
name: DistributedComputing
channels:
  - defaults
dependencies:
  - ca-certificates=2019.8.28=0
  - certifi=2019.9.11=py37_0
  - libcxx=4.0.1=hcfea43d_1
  - libcxxabi=4.0.1=hcfea43d_1
  - libedit=3.1.20181209=hb402a30_0
  - libffi=3.2.1=h475c297_4
  - ncurses=6.1=h0a44026_1
  - openssl=1.1.1d=h1de35cc_2
  - pip=19.2.3=py37_0
  - python=3.7.4=h359304d_1
  - readline=7.0=h1de35cc_5
  - setuptools=41.4.0=py37_0
  - sqlite=3.30.0=ha441bb4_0
  - tk=8.6.8=ha441bb4_0
  - wheel=0.33.6=py37_0
  - xz=5.2.4=h1de35cc_4
  - zlib=1.2.11=h1de35cc_3
prefix: /Users/dwoodbridge/anaconda3/envs/DistributedComputing
```

**.yaml** : Data serialization language.  
Commonly used for configuration files.

Using Pip in a Conda Environment  
<https://www.anaconda.com/using-pip-in-a-conda-environment/>

# Create an Environment

---

## Environment

- **Remove** the environment (after deactivate)

```
conda remove --name DistributedComputing --all
```

- **Create or update** an environment using .yaml.

```
conda env create -f DistributedComputing_environment.yaml -n DistributedComputing
```

```
conda env update -f DistributedComputing_environment.yaml -n DistributedComputing
```

- -f : file name
- -n : environment name

# Example 2

---

Export your environment into DistributedComputing\_environment.yml.

Deactivate your environment.

Delete your environment.

Create an environment using DistributedComputing\_environment.yml

# Contents

---

Class Intro

Environment Setup

**Code Standard**

Big Data and Distributed Computing

MapReduce Concept



# Code Conventions

---

For collaborative environment, using a consistent style provides better code management, readability, understandability and maintenance.

Follow commonly used or internally agreed code conventions.

- Code Conventions : Commonly used and recommended choices as a more readable option.
- PEP (Python Enhancement Proposals)
  - PEP 8 : Style Guide for Python Code
  - PEP 20 : The Zen of Python

# Contents

---

Class Intro

Environment Setup

Code Standard

**Big Data and Distributed Computing**

MapReduce Concept



# Big Data

---

Size and structure is beyond the ability of traditional data-processing application software can adequately handle.

- Example domains
  - Internet of Things (IoT) – Smart homes, Electronic toll collection, Smart grid, Wearable, etc.
  - Health Care – Electronic health records (EHR)
  - Marketing – Consumer data
  - What else?
- Size : Constantly moving target - as of 2012 ranging from terabytes (TB) to exabytes (EB).
- Structure : Mostly unstructured.

➤ Storing and **Processing** Big Data became an important issue!

[https://en.wikipedia.org/wiki/Big\\_data](https://en.wikipedia.org/wiki/Big_data)

# Distributed Computing



For processing large volumes of data fast,

- “Scale out” instead of scale up.
- Cheaper : Run large data on clusters of many smaller and cheaper machines.
- Reliable (Fault Tolerant): If one node or process fails, its workload should be assumed by other components in the system.
- Faster : It parallelizes and distributes computations.





# Contents

---

Class Intro

Environment Setup

Code Standard

Big Data and Distributed Computing

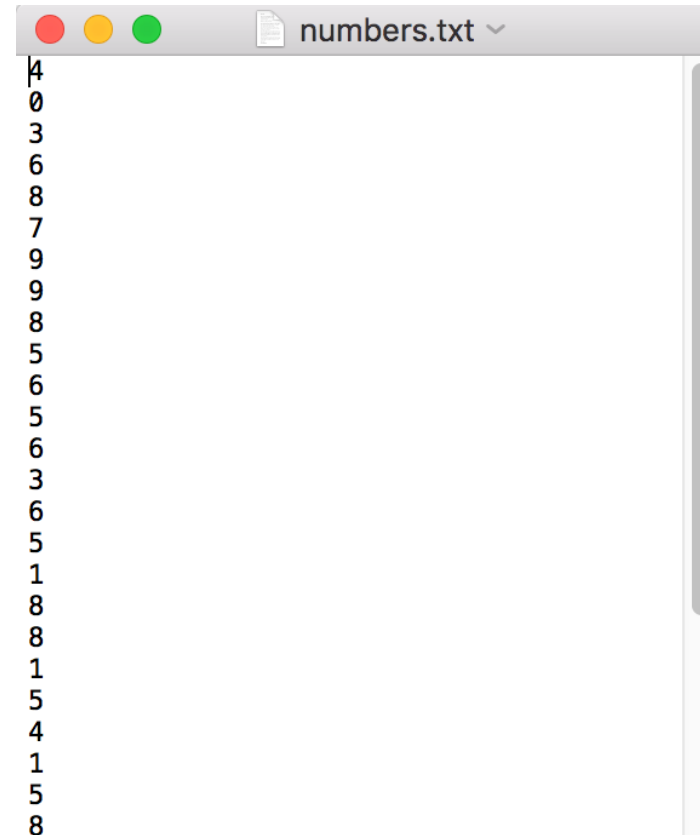
**MapReduce Concept**



# Discussion

In Data/numbers.txt, how to calculate the sum of numbers that are greater than or equal to 8 quickly?

- numbers.txt : 100 integers between 0 and 9.

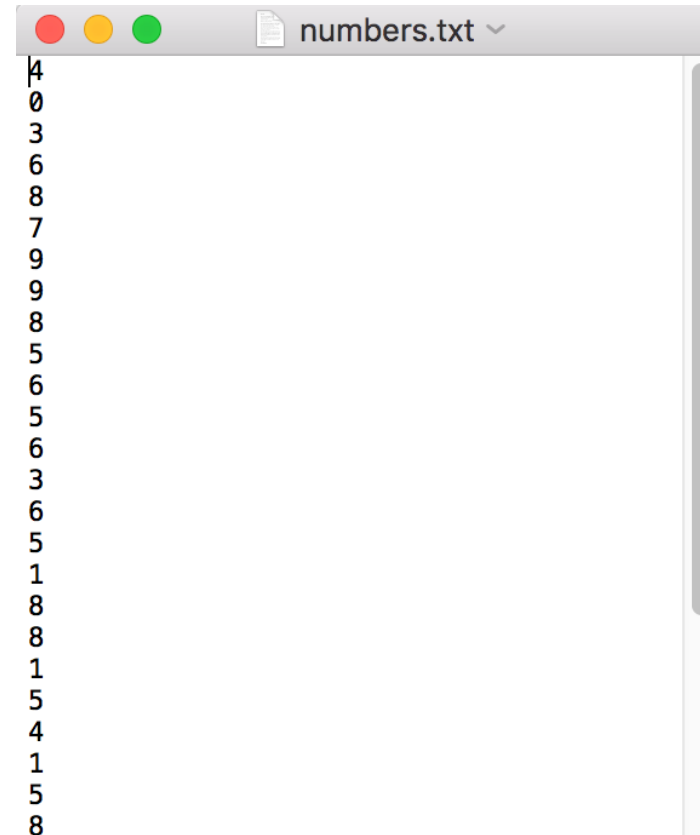


# Discussion

In Data/numbers.txt, how to calculate the sum of numbers that are greater than or equal to 8 quickly?

- numbers.txt : 100 integers between 0 and 9.

WHAT IF WE HAVE 1 TB Of DATA?



**How can we calculate the sum of the numbers that are greater/equal to 8 in Data/numbers.txt quickly?**

# Example

---

## My Computer

- Logical cores = (# of physical cores) x (# of threads that can run on each core)

```
sysctl hw.physicalcpu hw.logicalcpu
```

```
(DistributedComputing) ML-ITS-603436:Answer dwoodbridge$ sysctl hw.physicalcpu hw.logicalcpu  
hw.physicalcpu: 4  
hw.logicalcpu: 8
```



# Example

---

## My Computer

- Logical cores = (# of physical cores) x (# of threads that can run on each core)

```
sysctl hw.physicalcpu hw.logicalcpu
```

- This determines how many **partitions** (unit of work) can run.



# Example

Spark Way – **Utilize all the cores (or a few) !**

```
'4','0','3','6','8','7','9','9','8','5','6','5','6','3','6','5','1','8','8','1','5','4','1','5','8','2','6','8','8',  
'3','9','9','0','7','3','6','0','6','0','5','9','0','9','5','9','2','5','4','8','6','8','5','1','4','7','4','0','1',  
'9','5','3','1','1','0','0','9','3','0','5','4','0','8','0','1','8','5','7','2','1','9','3','4','2','0','2','8','1',  
      '0','0','8','4','1','0','0','7','3','8','3','7','3'
```

```
'4','0','3','6','8','7','9','9',  
'8','5','6','5','6'
```



# Example

## Spark Way

'4','0','3','6','8','7','9','9','8','5','6','5','6','3','6','5','1','8','8','1','5','4','1','5','8','2','6','8','8',  
'3','9','9','0','7','3','6','0','6','0','5','9','0','9','5','9','2','5','4','8','6','8','5','1','4','7','4','0','1',  
'9','5','3','1','1','0','0','9','3','0','5','4','0','8','0','1','8','5','7','2','1','9','3','4','2','0','2','8','1',  
'0','0','8','4','1','0','0','7','3','8','3','7','3'

'4','0','3','6','8','7','9','9',  
'8','5','6','5','6'

'3','6','5','1','8','8','1','5',  
'4','1','5','8','2'





# Example

## Spark Way

'4','0','3','6','8','7','9','9','8','5','6','5','6','3','6','5','1','8','8','1','5','4','1','5','8','2','6','8','8',  
'3','9','9','0','7','3','6','0','6','0','5','9','0','9','5','9','2','5','4','8','6','8','5','1','4','7','4','0','1',  
'9','5','3','1','1','0','0','9','3','0','5','4','0','8','0','1','8','5','7','2','1','9','3','4','2','0','2','8','1',  
'0','0','8','4','1','0','0','7','3','8','3','7','3'

'4','0','3','6','8','7','9','9',  
'8','5','6','5','6'

'3','6','5','1','8','8','1','5',  
'4','1','5','8','2'

'6','8','8','3','9','9','0','7',  
'3','6','0','6'



# Example

## Spark Way

'4','0','3','6','8','7','9','9','8','5','6','5','6','3','6','5','1','8','8','1','5','4','1','5','8','2','6','8','8',  
'3','9','9','0','7','3','6','0','6','0','5','9','0','9','5','9','2','5','4','8','6','8','5','1','4','7','4','0','1',  
'9','5','3','1','1','0','0','9','3','0','5','4','0','8','0','1','8','5','7','2','1','9','3','4','2','0','2','8','1',  
'0','0','8','4','1','0','0','7','3','8','3','7','3'

'4','0','3','6','8','7','9','9',  
'8','5','6','5','6'

'3','6','5','1','8','8','1','5',  
'4','1','5','8','2'

'6','8','8','3','9','9','0','7',  
'3','6','0','6'

'0','5','9','0','9','5','9','2',  
'5','4','8','6','8'



# Example

## Spark Way

'4','0','3','6','8','7','9','9','8','5','6','5','6','3','6','5','1','8','8','1','5','4','1','5','8','2','6','8','8',  
'3','9','9','0','7','3','6','0','6','0','5','9','0','9','5','9','2','5','4','8','6','8','5','1','4','7','4','0','1',  
'9','5','3','1','1','0','0','9','3','0','5','4','0','8','0','1','8','5','7','2','1','9','3','4','2','0','2','8','1',  
'0','0','8','4','1','0','0','7','3','8','3','7','3'

'4','0','3','6','8','7','9','9',  
'8','5','6','5','6'

'3','6','5','1','8','8','1','5',  
'4','1','5','8','2'

'6','8','8','3','9','9','0','7',  
'3','6','0','6'

'0','5','9','0','9','5','9','2',  
'5','4','8','6','8'

'5','1','4','7','4','0','1','9',  
'5','3','1','1'



# Example

## Spark Way

'4','0','3','6','8','7','9','9','8','5','6','5','6','3','6','5','1','8','8','1','5','4','1','5','8','2','6','8','8',  
'3','9','9','0','7','3','6','0','6','0','5','9','0','9','5','9','2','5','4','8','6','8','5','1','4','7','4','0','1',  
'9','5','3','1','1','0','0','9','3','0','5','4','0','8','0','1','8','5','7','2','1','9','3','4','2','0','2','8','1',  
'0','0','8','4','1','0','0','7','3','8','3','7','3'

'4','0','3','6','8','7','9','9',  
'8','5','6','5','6'

'3','6','5','1','8','8','1','5',  
'4','1','5','8','2'

'6','8','8','3','9','9','0','7',  
'3','6','0','6'

'0','5','9','0','9','5','9','2',  
'5','4','8','6','8'

'5','1','4','7','4','0','1','9',  
'5','3','1','1'

'0','0','9','3','0','5','4','0',  
'8','0','1','8','5'



# Example

## Spark Way

'4','0','3','6','8','7','9','9','8','5','6','5','6','3','6','5','1','8','8','1','5','4','1','5','8','2','6','8','8',  
'3','9','9','0','7','3','6','0','6','0','5','9','0','9','5','9','2','5','4','8','6','8','5','1','4','7','4','0','1',  
'9','5','3','1','1','0','0','9','3','0','5','4','0','8','0','1','8','5','7','2','1','9','3','4','2','0','2','8','1',  
'0','0','8','4','1','0','0','7','3','8','3','7','3'

'4','0','3','6','8','7','9','9',  
'8','5','6','5','6'

'3','6','5','1','8','8','1','5',  
'4','1','5','8','2'

'6','8','8','3','9','9','0','7',  
'3','6','0','6'

'0','5','9','0','9','5','9','2',  
'5','4','8','6','8'

'5','1','4','7','4','0','1','9',  
'5','3','1','1'

'0','0','9','3','0','5','4','0',  
'8','0','1','8','5'

'7','2','1','9','3','4','2','0',  
'2','8','1','0'



# Example

## Spark Way

'4','0','3','6','8','7','9','9','8','5','6','5','6','3','6','5','1','8','8','1','5','4','1','5','8','2','6','8','8',  
'3','9','9','0','7','3','6','0','6','0','5','9','0','9','5','9','2','5','4','8','6','8','5','1','4','7','4','0','1',  
'9','5','3','1','1','0','0','9','3','0','5','4','0','8','0','1','8','5','7','2','1','9','3','4','2','0','2','8','1',  
'0','0','8','4','1','0','0','7','3','8','3','7','3'

'4','0','3','6','8','7','9','9',  
'8','5','6','5','6'

'3','6','5','1','8','8','1','5',  
'4','1','5','8','2'

'6','8','8','3','9','9','0','7',  
'3','6','0','6'

'0','5','9','0','9','5','9','2',  
'5','4','8','6','8'

'5','1','4','7','4','0','1','9',  
'5','3','1','1'

'0','0','9','3','0','5','4','0',  
'8','0','1','8','5'

'7','2','1','9','3','4','2','0',  
'2','8','1','0'

'0','8','4','1','0','0','7','3',  
'8','3','7','3'



# Example

## Spark Way

```
rdd = sc.textFile("../Data/numbers.txt", 8)
```

```
rdd.glom().collect()
```

```
[['4', '0', '3', '6', '8', '7', '9', '9', '8', '5', '6', '5', '6'],  
 ['3', '6', '5', '1', '8', '8', '1', '5', '4', '1', '5', '8', '2'],  
 ['6', '8', '8', '3', '9', '9', '0', '7', '3', '6', '0', '6'],  
 ['0', '5', '9', '0', '9', '5', '9', '2', '5', '4', '8', '6', '8'],  
 ['5', '1', '4', '7', '4', '0', '1', '9', '5', '3', '1', '1'],  
 ['0', '0', '9', '3', '0', '5', '4', '0', '8', '0', '1', '8', '5'],  
 ['7', '2', '1', '9', '3', '4', '2', '0', '2', '8', '1', '0'],  
 ['0', '8', '4', '1', '0', '0', '7', '3', '8', '3', '7', '3']]
```

'4', '0', '3', '6', '8', '7', '9', '9',  
'8', '5', '6', '5', '6'

'3', '6', '5', '1', '8', '8', '1', '5',  
'4', '1', '5', '8', '2'

'6', '8', '8', '3', '9', '9', '0', '7',  
'3', '6', '0', '6'

'0', '5', '9', '0', '9', '5', '9', '2',  
'5', '4', '8', '6', '8'

'5', '1', '4', '7', '4', '0', '1', '9',  
'5', '3', '1', '1'

'0', '0', '9', '3', '0', '5', '4', '0',  
'8', '0', '1', '8', '5'

'7', '2', '1', '9', '3', '4', '2', '0',  
'2', '8', '1', '0'

'0', '8', '4', '1', '0', '0', '7', '3',  
'8', '3', '7', '3'



# Example

---

Spark Way

**String to  
Integer**

'4', '0', '3', '6', '8', '7', '9', '9', '8', '5', '6', '5', '6'

4,0,3,6,8,7,9,9,8,5,6,5,6





# Example

NOTE : This happens in parallel.

Spark Way

	String to Integer	
'4', '0', '3', '6', '8', '7', '9', '9', '8', '5', '6', '5', '6'	→	4,0,3,6,8,7,9,9,8,5,6,5,6
'3', '6', '5', '1', '8', '8', '1', '5', '4', '1', '5', '8', '2'	→	3,6,5,1,8,8,1,5,4,1,5,8,2
'6', '8', '8', '3', '9', '9', '0', '7', '3', '6', '0', '6'	→	6,8,8,3,9,9,0,7,3,6,0,6
'0', '5', '9', '0', '9', '5', '9', '2', '5', '4', '8', '6', '8'	→	0,5,9,0,9,5,9,2,5,4,8,6,8
'5', '1', '4', '7', '4', '0', '1', '9', '5', '3', '1', '1'	→	5,1,4,7,4,0,1,9,5,3,1,1
'7', '2', '1', '9', '3', '4', '2', '0', '2', '8', '1', '0'	→	0,0,9,3,0,5,4,0,8,0,1,8,5
'0', '0', '9', '3', '0', '5', '4', '0', '8', '0', '1', '8', '5'	→	7,2,1,9,3,4,2,0,2,8,1,0
'0', '8', '4', '1', '0', '0', '7', '3', '8', '3', '7', '3'	→	0,8,4,1,0,0,7,3,8,3,7,3



# Example

---

## Spark Way

```
converted_rdd = rdd.map(lambda x: int(x))
```

4,0,3,6,8,7,9,9,8,5,6,5,6

3,6,5,1,8,8,1,5,4,1,5,8,2

6,8,8,3,9,9,0,7,3,6,0,6

0,5,9,0,9,5,9,2,5,4,8,6,8

5,1,4,7,4,0,1,9,5,3,1,1

0,0,9,3,0,5,4,0,8,0,1,8,5

7,2,1,9,3,4,2,0,2,8,1,0

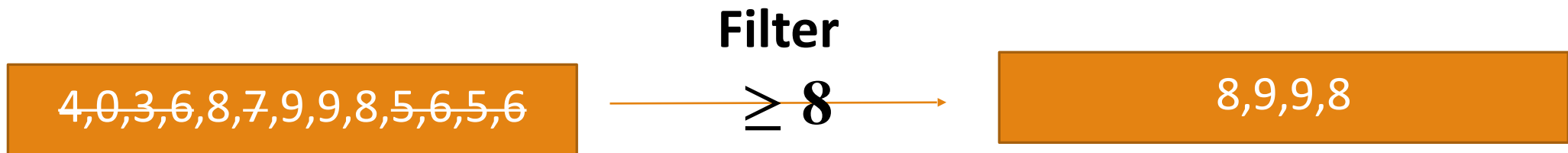
0,8,4,1,0,0,7,3,8,3,7,3



# Example

---

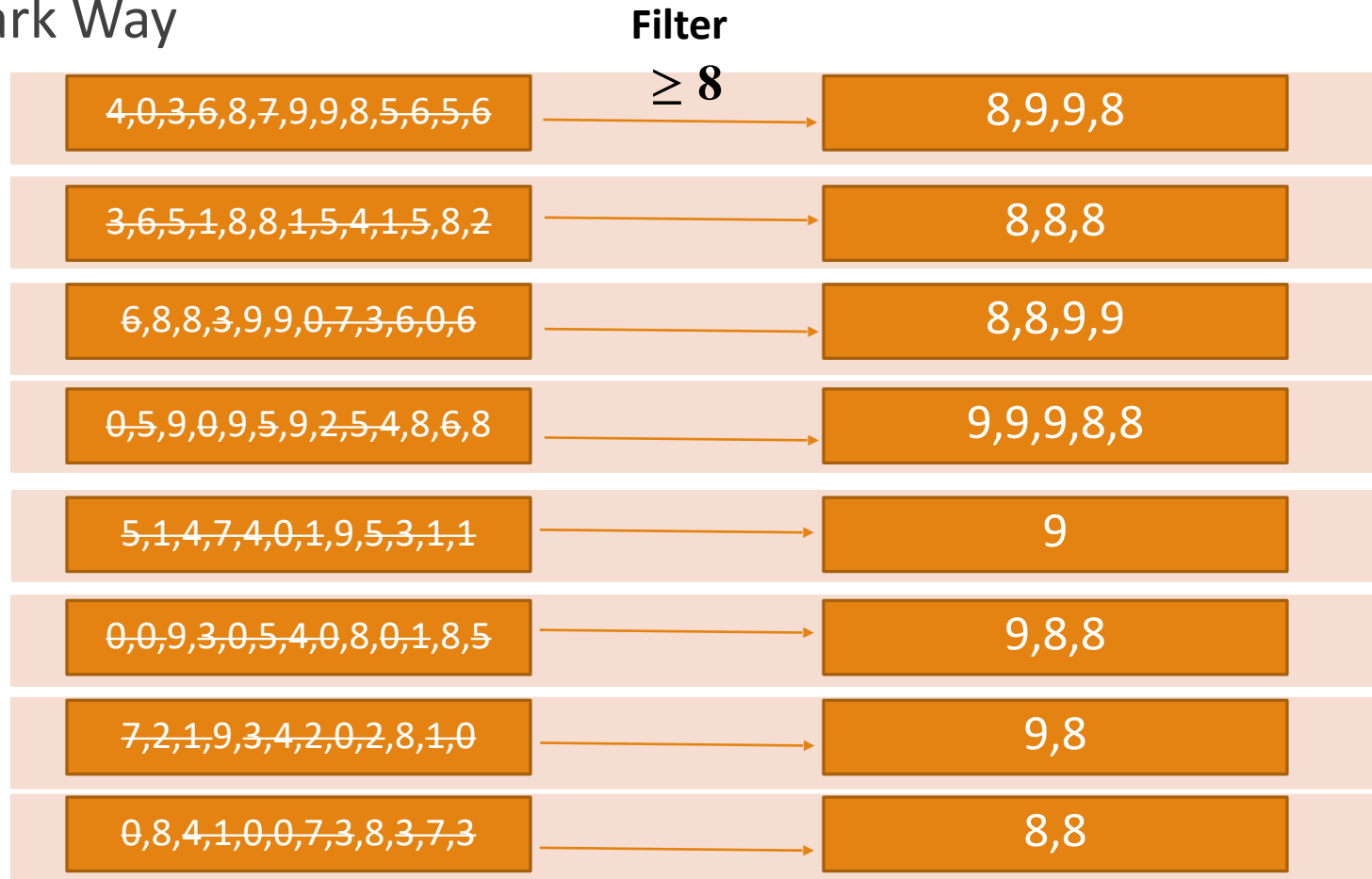
Spark Way



# Example

NOTE : This happens in parallel.

Spark Way



# Example

## Spark Way

```
filtered_rdd = converted_rdd.filter(lambda x : x >= 8 )
```

8, 9, 9, 8

8, 8, 8

8, 8, 9, 9

9, 9, 9, 8, 8

9

9, 8, 8

9, 8

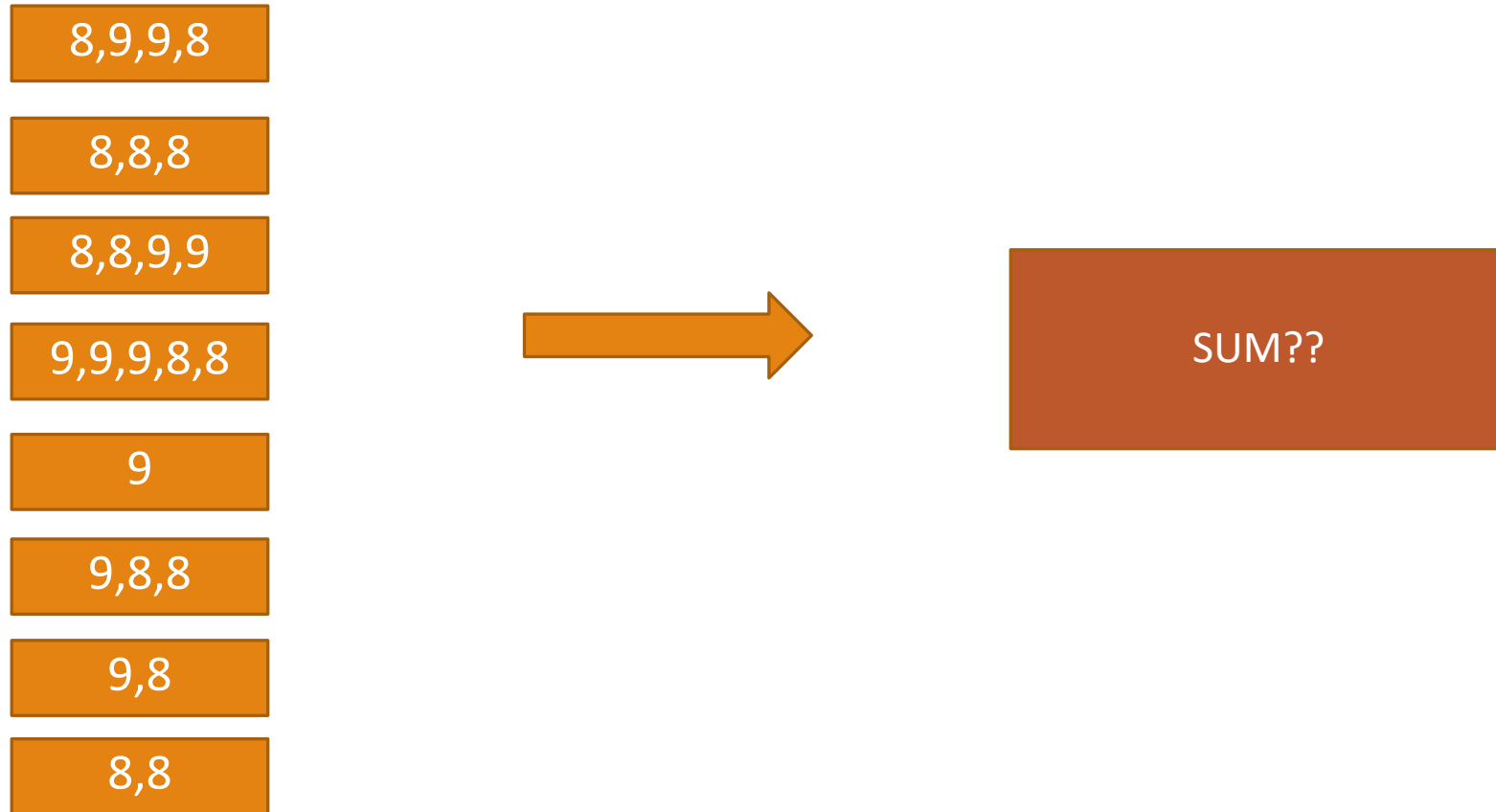
8, 8



# Example

---

Spark Way



# Example

---

Spark Way

8, 9, 9, 8

Add two numbers in the same partition until there is only one number left.

8+9,9,8

17+9,8

26+8

34



# Example

---

Spark Way

8, 9, 9, 8

Add two numbers in the same partition until there is only one number left.

8+9,9,8

17+9,8

26+8

34

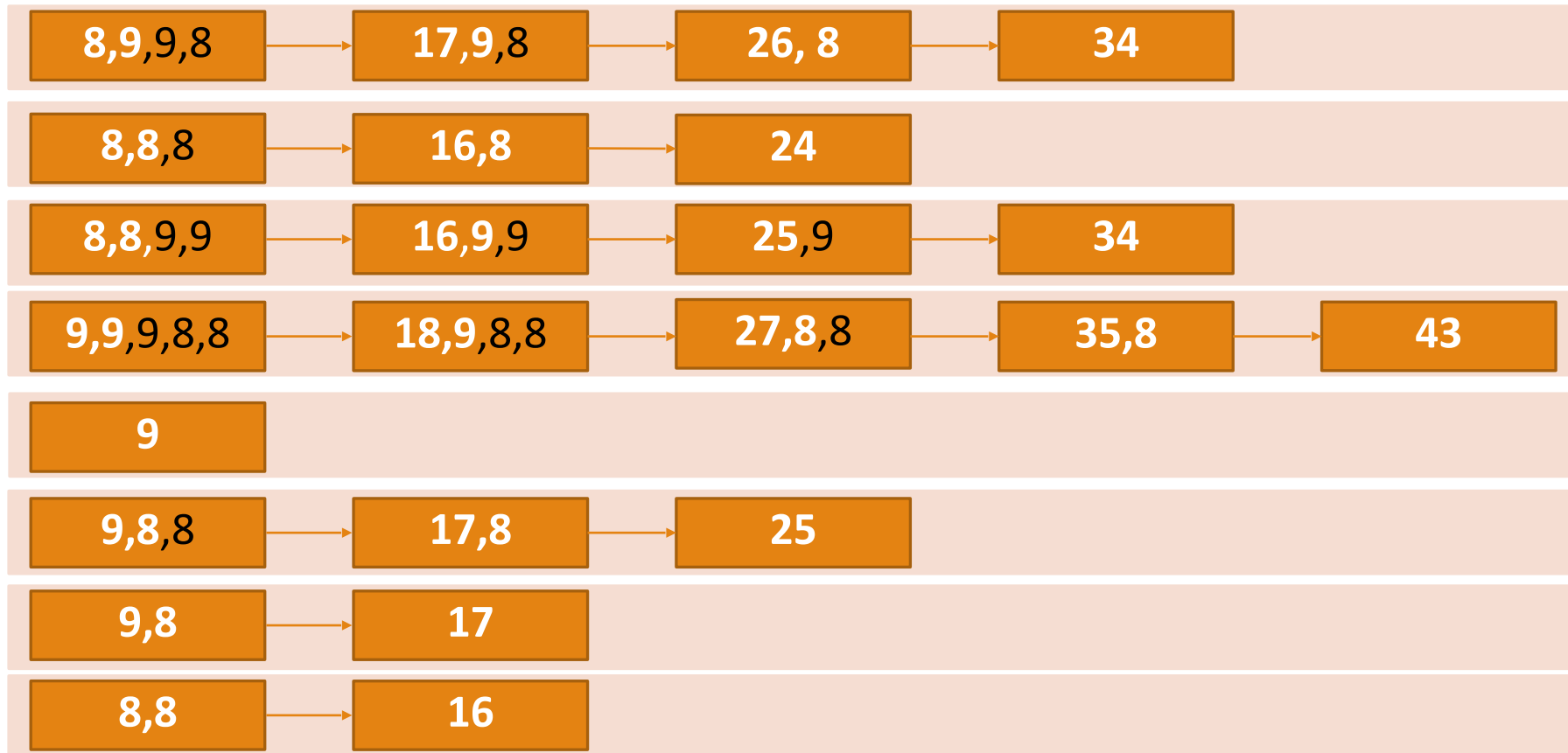




# Example

NOTE : This happens in parallel.

## Spark Way



# Example

NOTE : This happens in parallel.

## Spark Way



# Example

---

34

24

34

43

9

25

17

16

Add two numbers in different partitions until there is only one number left.



# Example

Add two numbers in different partitions  
until there is only one number left.  
- shuffling happens

34

58

24

34

77

43

9

34

25

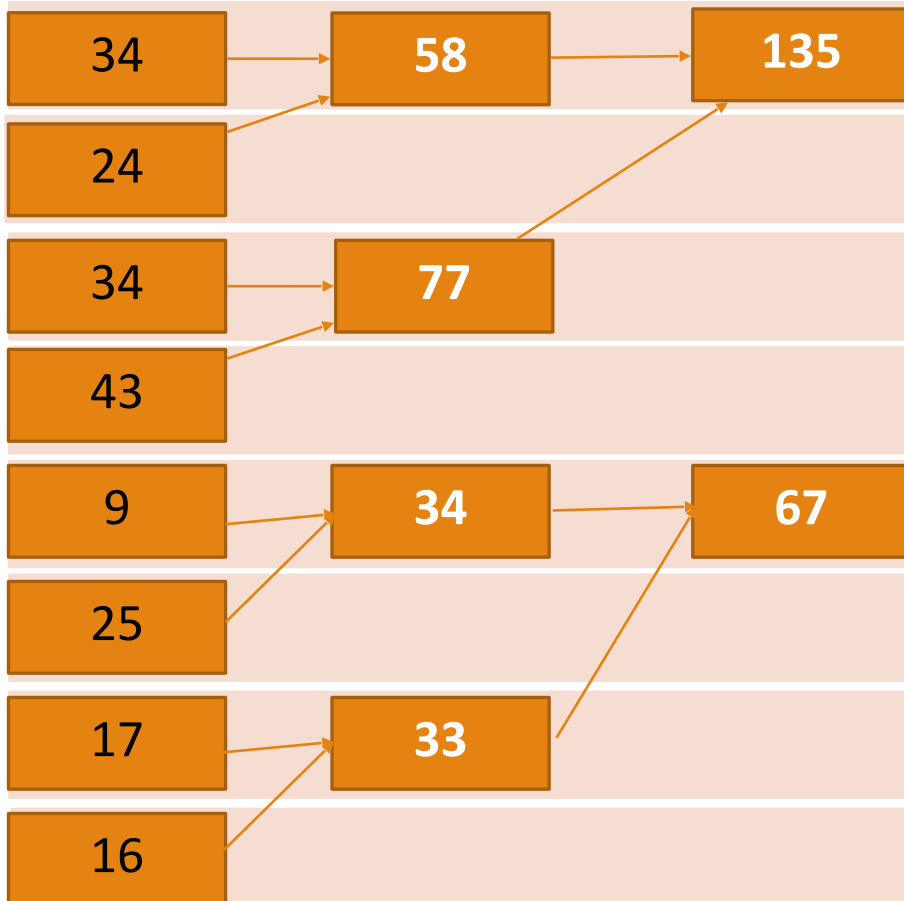
17

33

16



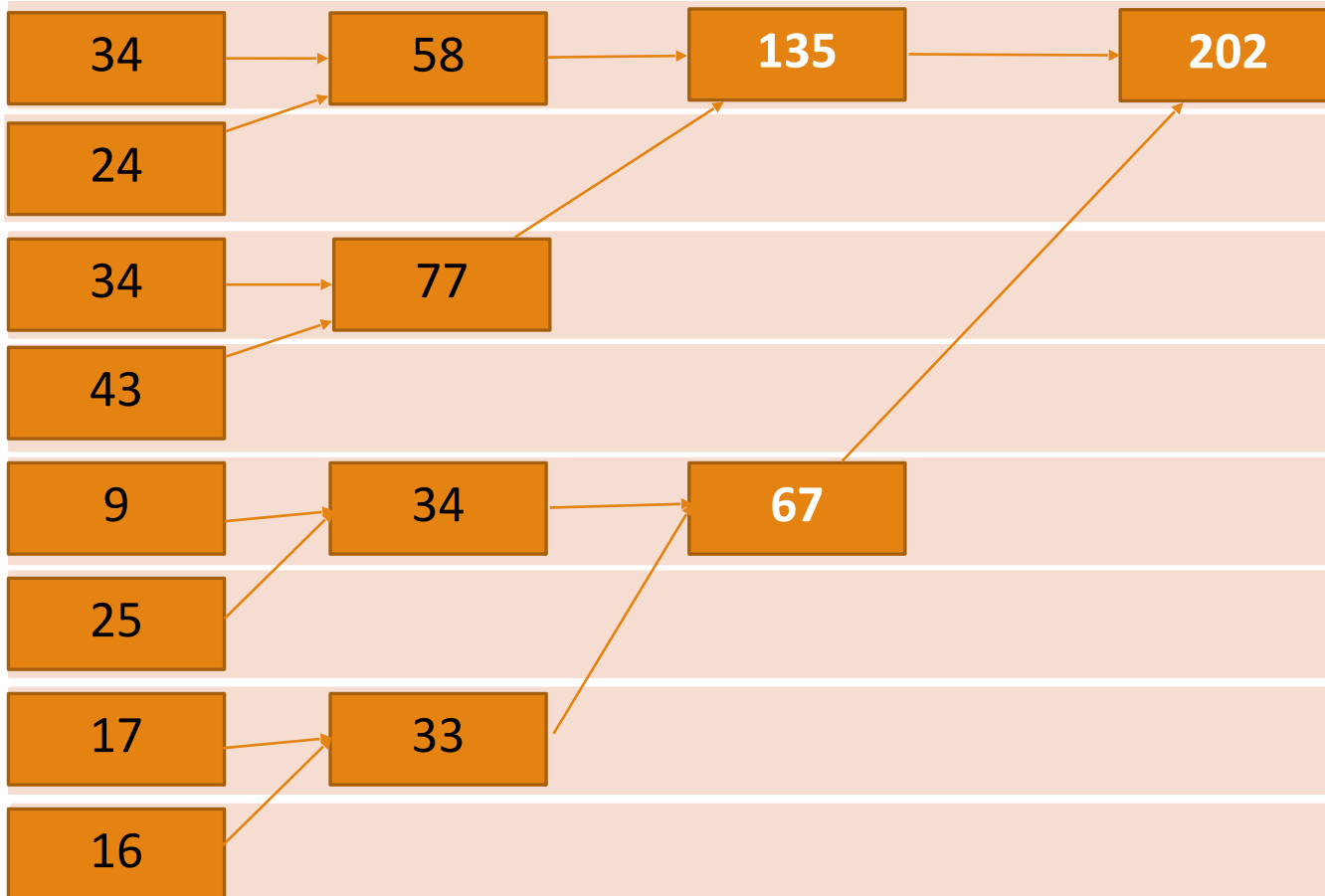
# Example



Add two numbers in different partitions until there is only one number left.  
- shuffling happens



# Example



Add two numbers in different partitions until there is only one number left.  
- shuffling happens



# Example 3

You don't need to understand this code yet.

1. Load data to 8 partitions.

```
In [2]: rdd = sc.textFile("../Data/numbers.txt", 8)  # Load data from the file.
```

2. Convert each data to integer.

```
In [3]: converted_rdd = rdd.map(lambda x: int(x))
```

3. Filter data.

```
In [4]: filtered_rdd = converted_rdd.filter(lambda x: x >= 8)
```

4. Calculate the sum.

```
In [5]: filtered_rdd.reduce(lambda x, y: x+y)
```



# MapReduce

---

Map-Reduce: Allow computations to be parallelized over a cluster.

- Basic Map-Reduce
  - The map-reduce framework plans map tasks to be run on the correct partitions and shuffle data for the reduce function.
  - **Map** : Apply a function to each data over a portion of data in parallel. ex. map(), filter()
  - **Reduce** : Return one value from multiple values. ex. reduce(), sum(), count()





# MapReduce

---

Map-Reduce: Allow computations to be parallelized over a cluster.

- Basic Map-Reduce
  - The map-reduce framework plans map tasks to be run on the correct nodes and shuffle data for the reduce function.
  - **Map** : Apply a function to each data over a portion of data in parallel. ex. filter(), map()

```
In [3]: converted_rdd = rdd.map(lambda x: int(x))
```

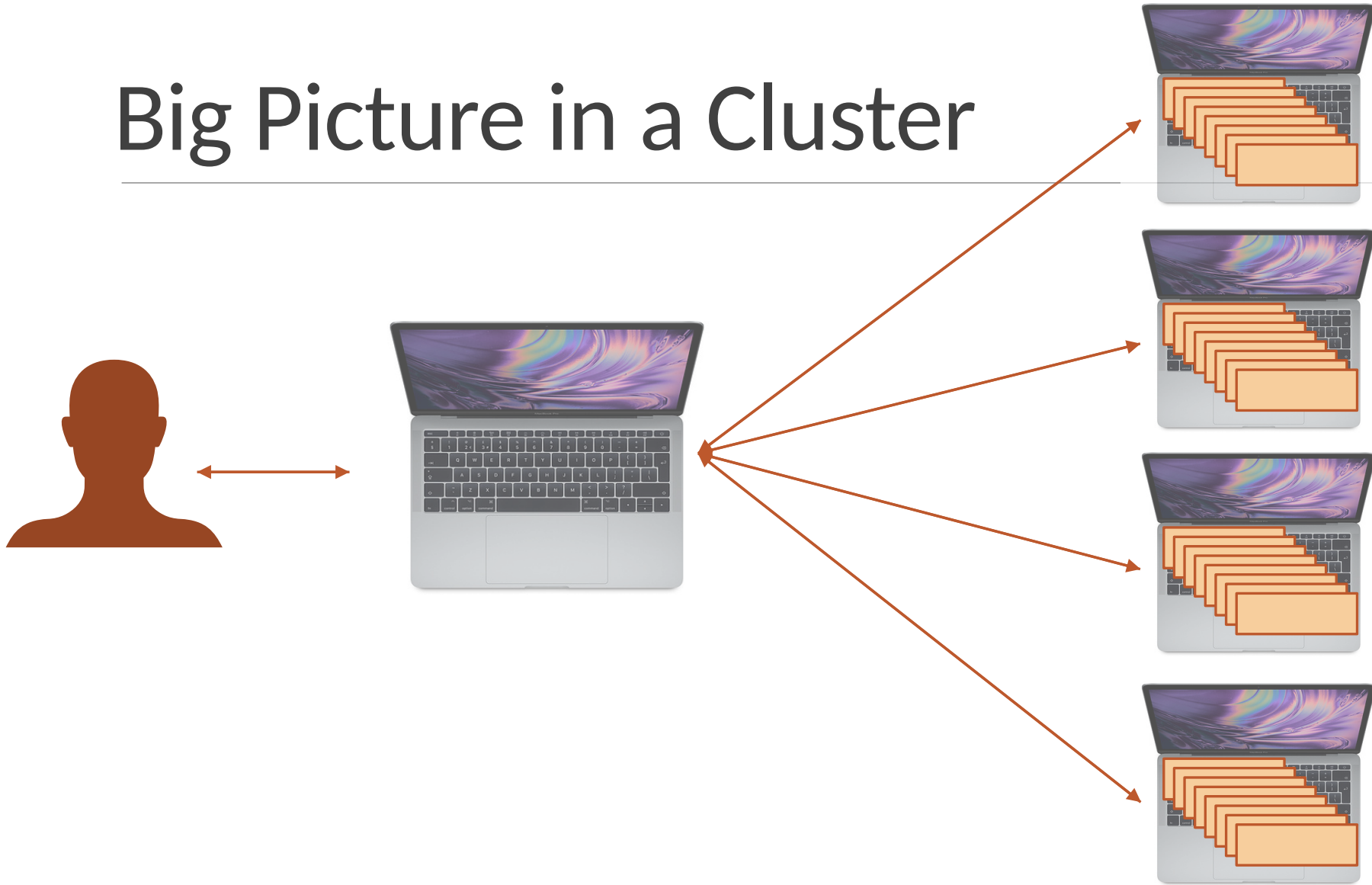
```
In [4]: filtered_rdd = converted_rdd.filter(lambda x: x >= 8)
```

- **Reduce** : Return one value from multiple values. ex. reduce(), sum(), count()

```
In [5]: filtered_rdd.reduce(lambda x, y: x+y)
```



# Big Picture in a Cluster



# Contents

---

Class Intro

Environment Setup

Code Standard

Big Data and Distributed Computing

MapReduce Concept



# **Week 1 - Comments (What you liked/disliked so far? What should I do for you?)**

# Reference

---

Spark Online Documentation, <http://spark.apache.org/docs/latest/>

Anaconda Online Documentation, <https://docs.anaconda.com/>

PEP 8, <https://peps.python.org/pep-0008/>

PEP 20, <https://peps.python.org/pep-0020/>

