15-319 / 15-619 Cloud Computing

Recitation 2

September 4 & 6, 2018

Accessing the Course

- Open Learning Initiative (OLI) Course
  - Access via canvas.cmu.edu
- http://theproject.zone (access through canvas)
  - choose CMU as the identity provider
  - AWS Account Setup
  - Azure Account Setup
  - GCP Account Setup
  - Update your TPZ profile with AWS, Azure & GCP info
  - Complete the Primers on AWS, Azure and GCP
- Piazza

Piazza

- Suggestions for using Piazza
  - Discussion forum, contribute questions and answers
  - Read the Piazza Post Guidelines (@6) before asking
- When you have a (project-specific) problem, follow the order below!
  - Try to solve the problem by yourself (Search, Stack Overflow)
  - Read Piazza questions & answers carefully to avoid duplicates
    - Visit TA OHs: TA office hours are posted on Piazza and Google calendar
    - Create a piazza post
- Please note:
  - **Show the effort you have done first**
  - **Give us context** and as much information as possible!
  - Try to ask a public question if possible
  - Provide your andrewID privately if you think we need it to help
  - Don't ask a public question about a quiz question

Reflecting on Last Week

- AWS, Azure and GCP
  - Create accounts
  - Use web consoles or APIs to launch VMs on AWS, Azure and GCP
  - (AWS) Spot instances and S3
- In P0, run a web server, test to access the server over a browser
  - Launch, connect to and terminate VMs
  - Install & run software on a VM
  - Vertical scaling
- Basic SSH skills
- In OMP primer, set up configuration for AWS Cloud 9
  - Read it ASAP if you have not done so
- Terraform primers
  - Read it if you have not done so

Programming Experience Expected

- **Strong proficiency** in at least one of the following, with some fair comprehension of the others:
  - **Java 8**
  - **Python 2/3**
  - **Bash**
- **Java is required** to complete parts of Projects.
- Use the time now to brush up
- Please read Maven primer!
- Do not fear bash/python scripting, it will make your life easier!

Completing Projects in this Course

- Provision AWS, Azure or GCP Resources
  - Use the AMIs/VHDs/OS Images we provide for the project
  - Tag all instances!
- Monitor your cost
  - Calculate costs before you provision!
- Complete tasks for each project module
- Project writeup has several sections unlocked by AssessMe
- Submit your work
  - Pledge of integrity
  - Results in scoreboard
- Terminate all resources when you have verified your score and kept a copy of your work (e.g. git **private** repo)

Tagging

- Tag *all* tag-able resources on AWS
  - Before you make a resource request, read the docs/specifications to find out if tagging is supported
  - We will specify which resources are required to be tagged in each project
  - Apply the tags during resource provisioning
  - We need tags to track usage, a grade penalty will be applied automatically if you do not tag!

- Tagging Format
  - `Key:  Project`
  - Value: 0, 1.1, 1.2....etc.

Budgets and Penalties

- No proper tags → 10% grade penalty
- Provision resources in regions other than us-east-1 → 10% grade penalty
- Budget
  - For P1.1, each student's budget is **$20**
  - Exceeding Budget → 10% project penalty
  - Exceeding Budget x 2 → 100% project penalty (no score)
  - You can see Cost and Penalties in TPZ.
- No exceptions.
- We will enforce these penalties automatically starting from Project 1.1

Academic Integrity Violation

- Cheating ➔ the lowest penalty is a 200% penalty & or R in the course
  - Other students, previous students, Internet (e.g. Stackoverflow), etc.
  - Do not work on code together
  - This is about you struggling with something and learning
  - Penalty for cheating is SEVERE – don’t do it!
  - Ask us if you are unsure

Compromised Accounts

- If you put any of your credentials in files on
  - Github, Dropbox, Google Drive, Box, etc.
  - You are vulnerable to getting your account compromised.
  - Going over 2x the project budget ⇒ 100% penalty!
- People are scanning publicly available files for cloud credentials.
  - They compromise your account and launch resources in other regions.

DO NOT SAVE YOUR CLOUD CREDENTIALS IN FILES!

Deadlines!

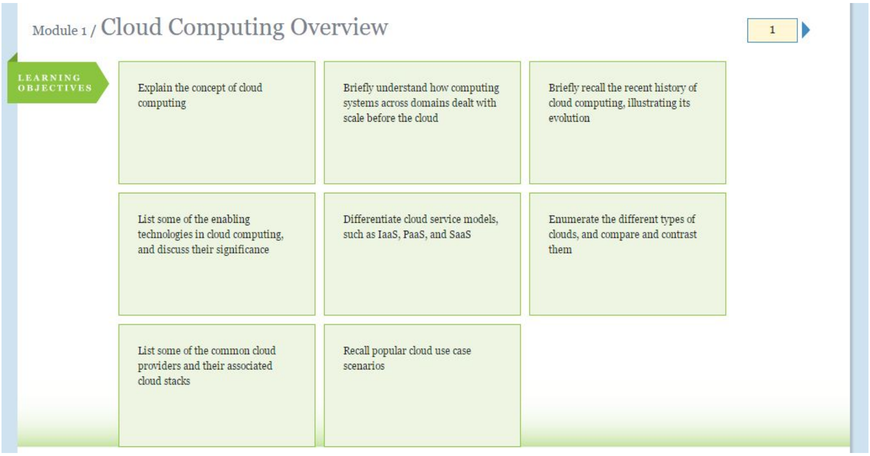
- **Hard** Deadlines
  - No late days, no extensions
  - Start early!
  - Plan your activities, interviews and other commitments around the deadlines.
  - **No exceptions!**
- Project modules are typically due on Sundays at 23:59 ET
- Quizzes are typically due on Fridays at 23:59 ET

Deadlines!

- **Project deadlines**
  - **On TheProject.Zone**
- **Quiz deadlines**
  - **On OLI**

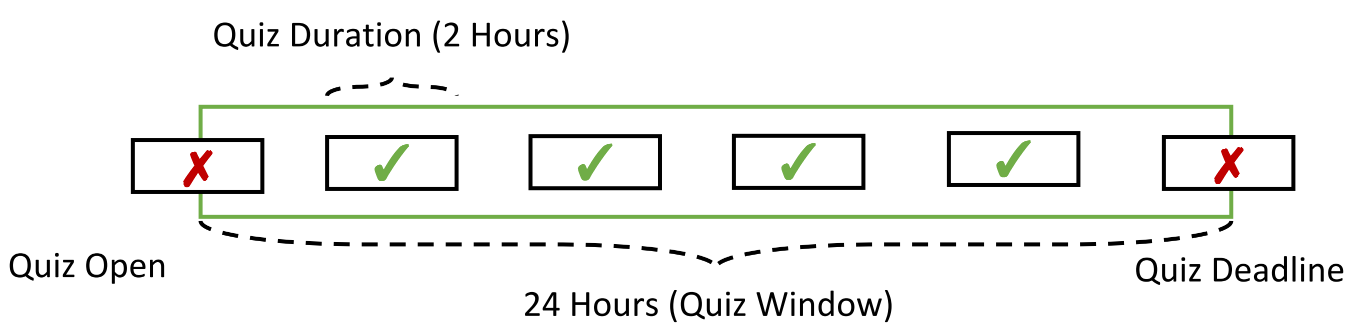
OLI: Quiz 1 Preparation

- Tests your understanding in Modules 1 and 2
  - Cloud computing fundamentals, service models, economics, SLAs, security
  - Use the activities in each page for practice
  - You will be tested on you ability to perform the stated learning objectives on OLI:



OLI: Quiz 1 Logistics

- Quiz 1 will be open on OLI for 24 hours, Friday, Sep 7
  - Quiz 1 becomes available on **Sep 7, 00:01 AM ET.**
  - Deadline for submission is **Sep 7, 11:59 PM ET.**
  - Once open, you have **120 min** to complete the quiz.
  - You may not start the quiz after the deadline has passed.
  - Every 15 minutes you will be prompted to save.
  - **Maintain <u>your own timer</u> from when you start the quiz.**
  - **Click <u>submit</u> before deadline passes. No Exceptions!**



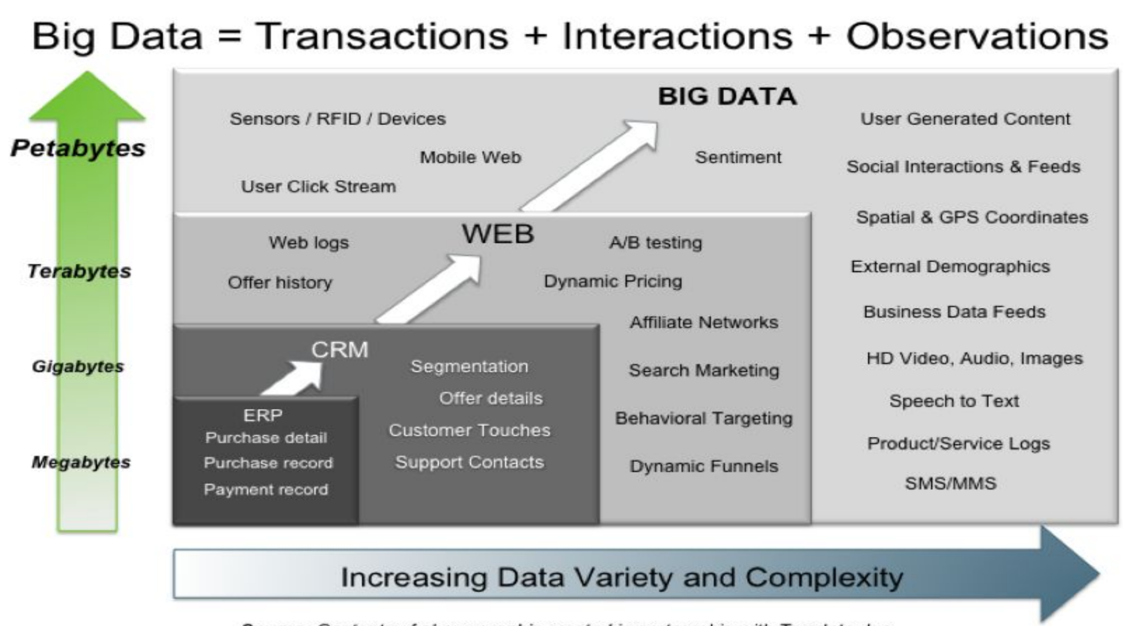
Submit Before Deadline

- When you start the Quiz, you cannot stop the clock
  - You have 120 minutes to click on submit
  - You have to keep track of the time yourself
  - If you don’t click on submit you will not receive a grade

YOU MUST SUBMIT
WITHIN 120 MINUTES
AND
BEFORE THE DEADLINE

Project 1 Motivation: Big Data

- What is Big Data?
  - It is high volume, high velocity, and/or high variety information assets.
  - There is a lot of value in the analysis of big data for organizations



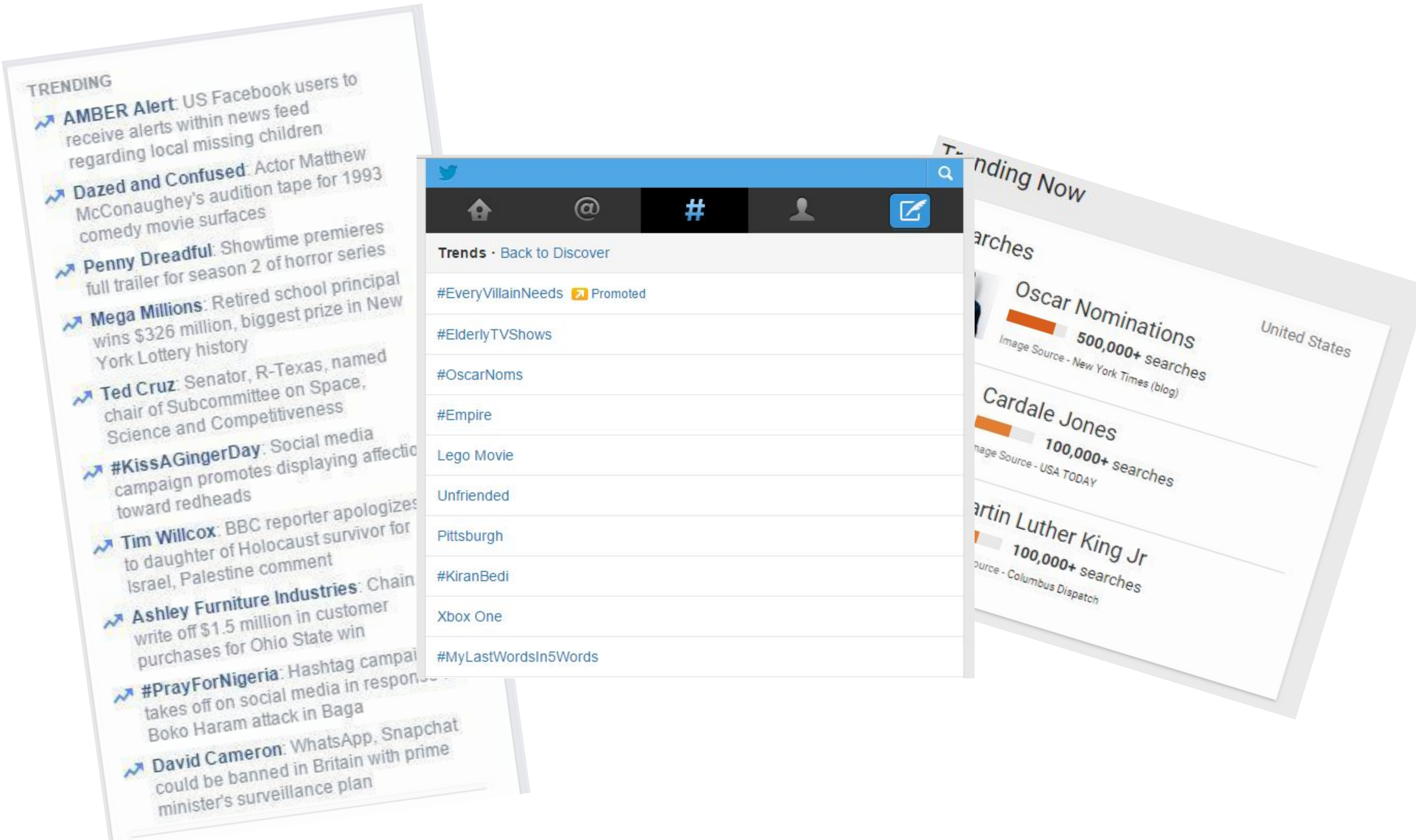
Source: *Contents of above graphic created in partnership with Teradata, Inc.*

Use Cases: Big Data Analysis

- Online retailers are analyzing consumer spending habits to learn trends and offer personalized recommendations and offers to individual customers

- Companies such as Youtube, etc. are using big data to track media consumption habits of their subscribers and trends to provide value-added information to advertisers and customers

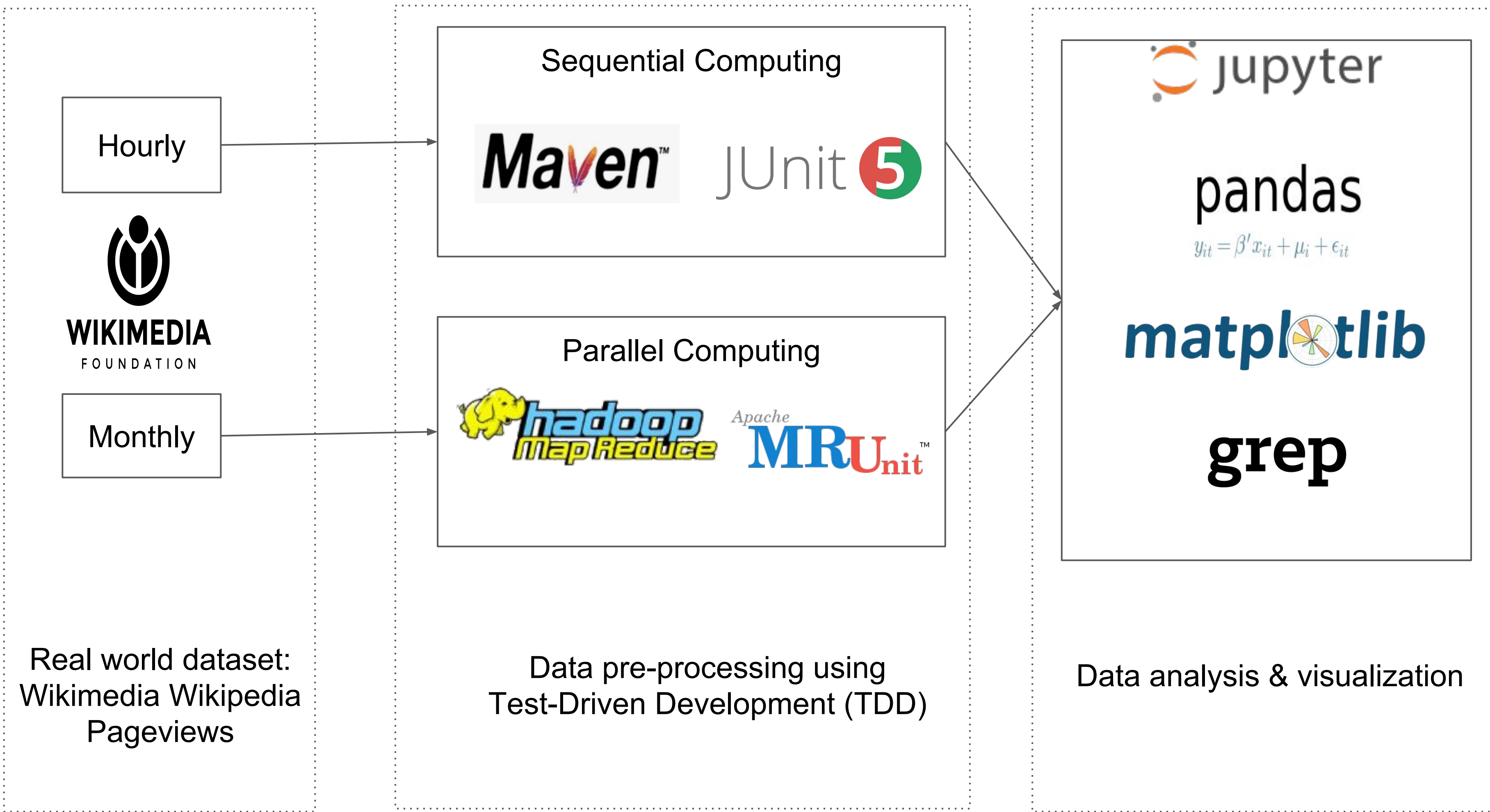
Trending Topics are Everywhere!



Why Trending Topics?

- Identify trends and viral content
- Maximize advertisement placement opportunities
- Search Engine Optimization (SEO)
- And more....

P1.1: Big Data Analytics



Project 1.1 Datasets

- Wikimedia Wikipedia pageviews dataset
  - Hourly: Start with filtering/pre-processing the hourly pageviews dataset
  - Monthly: Implement the filtering/pre-processing of the 30-day dataset using MapReduce
    - Data from March 8 to April 6 in 2018

The Wikimedia Dataset

- Data set
  - Wikimedia page views dataset
  - One file per hour
  - One month (30 days) = 720 files

- Data format:

<domain code> <page title> <number of accesses> <total data returned>

<Language>.<ProjectName>
en = English Wikipedia (Desktop)
en.b = English Wikibooks
fr.v = French Wikiversity

Project 1.1 Tasks

- Task 1: Sequential data filtering or pre-processing
  - Implement sequential data filter in Java with JUnit
- Task 2: Practice parallel processing with MapReduce
  - Implement Word Count as an example in MapReduce
- Task 3: Data Preprocessing with MapReduce
  - Implement the filter and keep the popular articles from Wikipedia pageviews in MapReduce using AWS EMR
- Task 4: Data analysis
  - Use Jupyter Notebook and Pandas library to analyze the output of Task 3

Data Pre-processing is Important

- Impossible: Raw Dataset → Data analysis
- Raw Dataset → Data pre-processing → Data analysis

raw data: after data pre-processing

reference: Nishant Neeraj

Task 1: Data Pre-processing

- We are only interested in English Wikipedia desktop/mobile pages (**`<domain code>:`** en, en.m)
- This dataset is raw, real-world
  - Never assume that the dataset is perfectly clean and well formed
- Use the filtering rules specified in the writeup
- If there are records from both desktop and mobile sites for the same page title, sum the accesses into one record
- Sort the pages by number of pageviews, break ties by ascending lexicographical order
- Output: **`<page title> <number of accesses>`**

Bad Coding Practices!

No modularity in code, hard to debug:

```
public static void main(final String[] args) {
    read the records from the input
    for record in records:
        if it violates the rule A: (20 lines)
            continue
        if it violates the rule B: (20 lines)
            continue
        … 5 other rules (100 lines)
        put record into a map <title, pageview>
    sort the map
    print output
}
```

Good Coding Practice: Test-Driven Development (TDD)

What is TDD?

- divide the problem into a series of small steps
- start by writing test cases
- then refactor the code to pass the test
- and repeat
    - Test case ⇒ code ⇒ pass ⇒ another test case ⇒ ...

TDD emphasizes writing unit tests ahead of writing the code.

TDD lets you treat failures as a norm instead of an exception.

Test-Driven Development (TDD)

Why TDD?

- Helps to structure your code in a way that easily facilitates testing
- Separates the concerns and makes your code clean, easy-to-read and robust
- Ensures that your changes won’t break existing functionality
- Achieves safer refactoring, increasing returns and effective collaborations
- TDD is an industry best practice!!!

TDD w/ JUnit 5

```java
public class Filter {

    public static boolean containsCloud(final String record) {
        // it is okay to start with an incorrect solution
        // the method signature is what matters
        return false;
    }
}
```

TDD w/ JUnit 5

```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class FilterTest {
    @Test
    void testContainsCloud() {
        // positive
        assertTrue(Filter.containsCloud("cloud computing"));
        // negative
        assertFalse(Filter.containsCloud("mapreduce"));
    }
}
```

TDD w/ JUnit 5

```
[INFO] Running FilterTest

[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.021 s <<< FAILURE! - in FilterTest

[ERROR] testContainsCloud  Time elapsed: 0.017 s  <<< FAILURE!

org.opentest4j.AssertionFailedError: expected: <true> but was: <false>

        at FilterTest.testContainsCloud(FilterTest.java:9)

[INFO] Results:

[ERROR] Failures:

[ERROR]   FilterTest.testContainsCloud:9 expected: <true> but was: <false>

[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] BUILD FAILURE

[INFO] ------------------------------------------------------------------------
```

TDD w/ JUnit 5

```
public class Filter {
    public static boolean containsCloud(final String record) {
        return record.contains("cloud");
    }
}
```

TDD w/ JUnit 5

```
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running FilterTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.012 s - in FilterTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

P1.1 Task 1: Data Pre-processing Code Template

- In this task, we provide a code template:
  - Merges both desktop and mobile sites for the same page title if any
  - Sorts the output in descending numerical order of the number of accesses and break ties by ascending lexicographical order
  - Outputs the results into a file named as exactly "output"
- It also defines a set of filter methods that you need to implement
- We provide you with a set of test cases for the first several filter methods
- Your task is to:
  - add the test cases for the rest of the required methods
  - implement the methods and pass the test

Limitations of sequential programs

- Your data-preprocessing program might work well with an hourly dataset, but will fall short to process a large dataset
- Methods to scale your solution
  - Sequential program might not scale ⇒ a parallel solution
  - A single EC2 machine might not have adequate memory and computational capabilities either ⇒ a large distributed cluster
- Challenges to overcome for your program to work in a distributed system
  - How would you partition and distribute the tasks and data?
  - How would the nodes communicate?
  - What if a node fails?

The MapReduce programming model

- The MapReduce programming model simplifies parallel processing by abstracting away the complexities involved in working with distributed systems
  - parallel computing
  - work distribution
  - dealing with unreliable hardware and software
- MapReduce allows the programmers to focus on writing and running the code rather than implementing your own distributed system framework

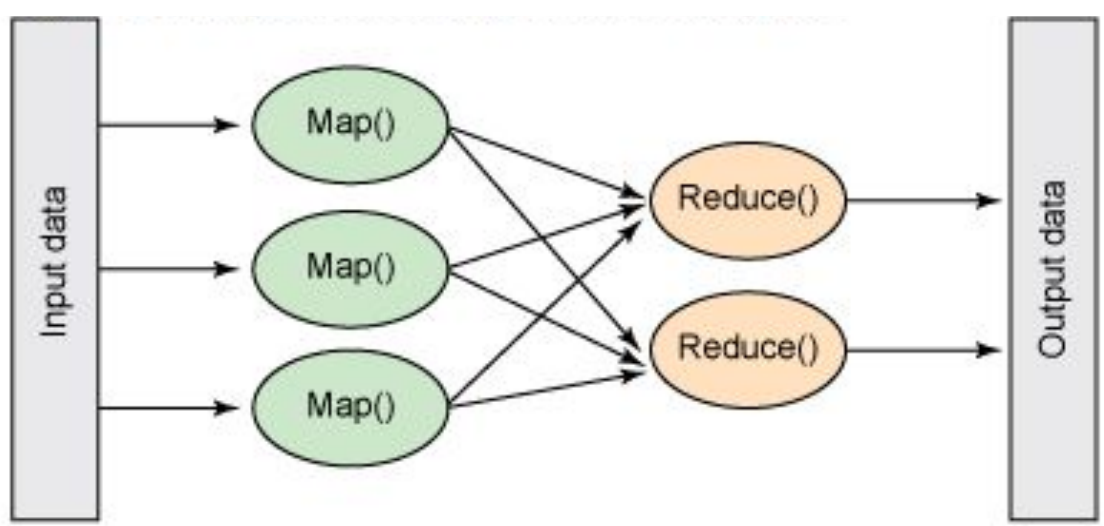
Motivation for MapReduce

- How do you perform *batch processing* of *large data sets* using *low cost clusters* with *thousands of machines* which *frequently experience partial failure or slowdowns*?

- The MapReduce programming model is designed for processing large data sets with a parallel, distributed algorithm on a cluster

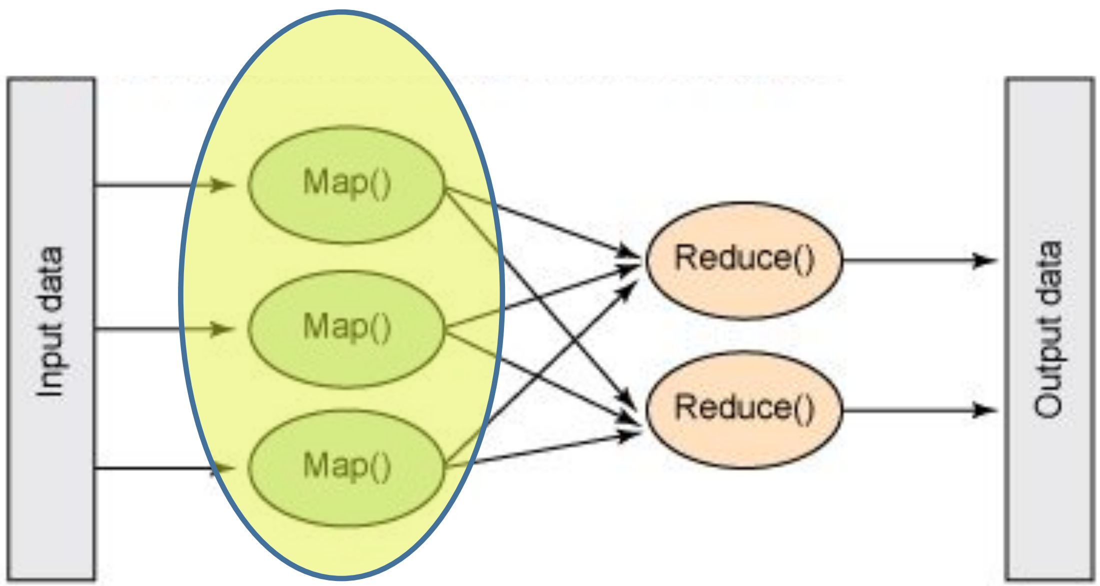
Introduction to MapReduce

- Map: Process the input data in chunks **in parallel**
- Shuffle and sort
- Reduce: Aggregate or summarize intermediate data **in parallel** and output the result



Introduction to MapReduce

- Map `map(k1,v1) --> list(k2,v2)`
  - Map function takes input as Key-Value pairs `k1,v1`.
  - The map function produces zero or more output Key-Value pairs for one input pair. `list(k2,v2)`



Introduction to MapReduce

- Map
  - `map(k1,v1) --> list(k2,v2)`

If the input is a file, the input Key-Value pair could represent a line in the file

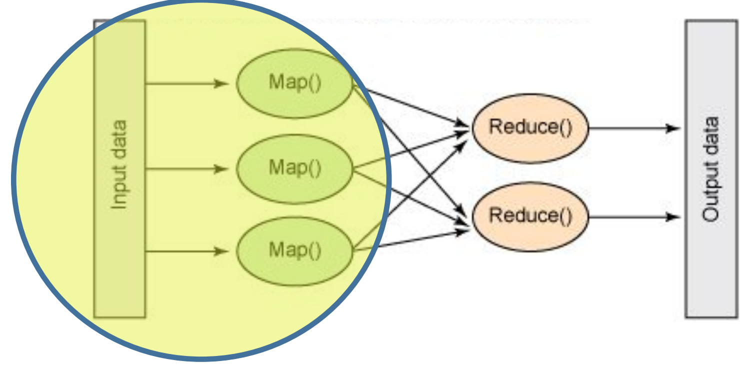
- keys are the position in the file
- values are the line of text

Word Count Example:

- Input ⇒ Word Count ⇒ output

- Content of one or more input files:
  - cat cow
  - duck
  - dog cat
  - cat
- Output:
  - cat, 3
  - cow, 1
  - dog, 1
  - duck, 1

Introduction to MapReduce



- Map in the Word Count Example

Input:
file1.txt

cat cow

duck

dog cat

`k1,v1 pairs:`

(pos, “cat cow”)
(pos, “duck”)
(pos, “dog cat”)

Mapper1

`k2,v2 pairs:`

(cat, 1)
(cow, 1)
(duck, 1)
(dog, 1)
(cat, 1)

Input:
file2.txt

cat

`k1,v1 pairs:`

(pos, “cat”)

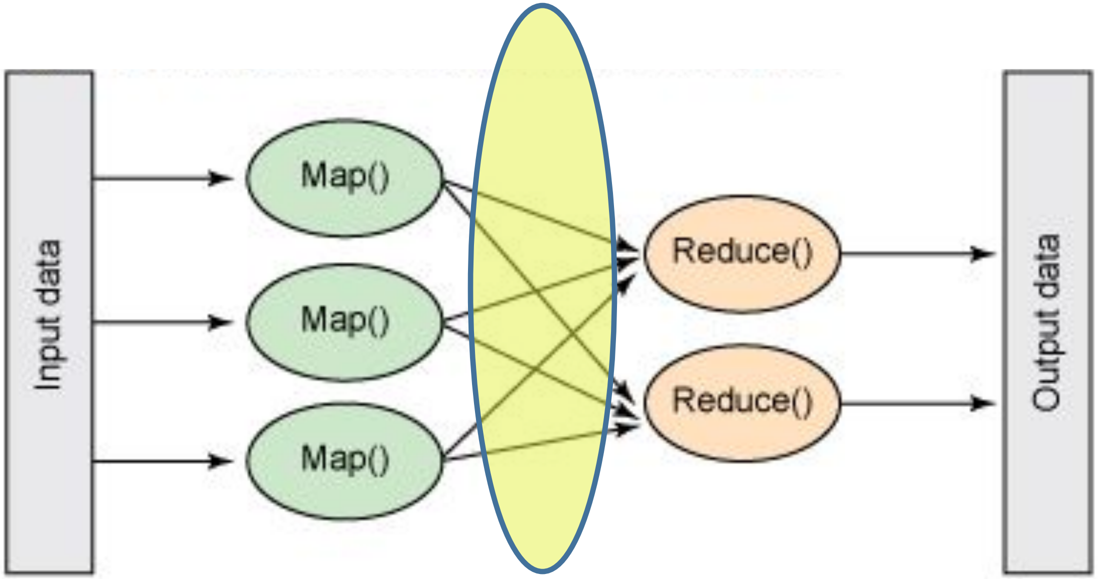
Mapper2

`k2,v2 pairs:`

(cat, 1)

Introduction to MapReduce

- Shuffle and sort

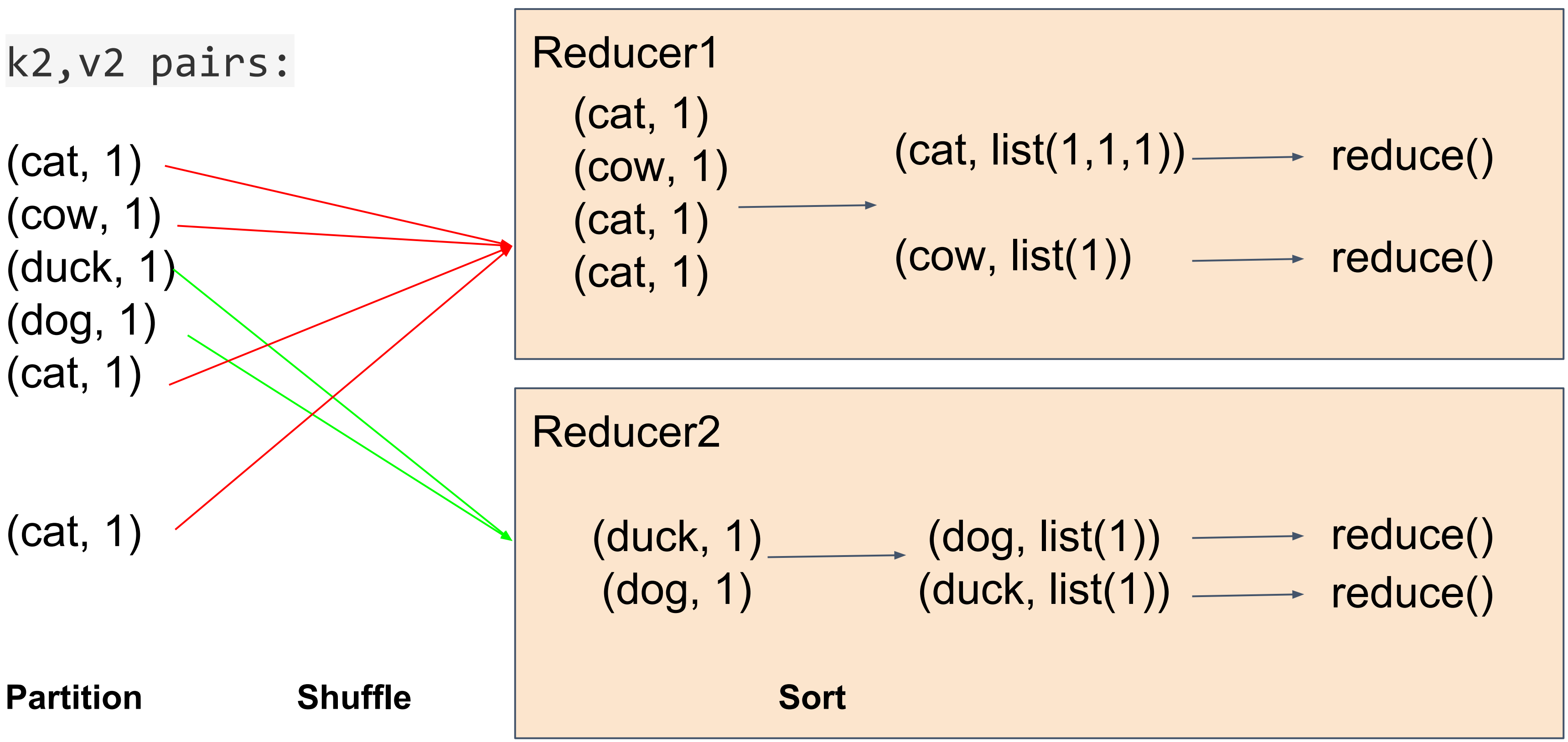


Introduction to MapReduce

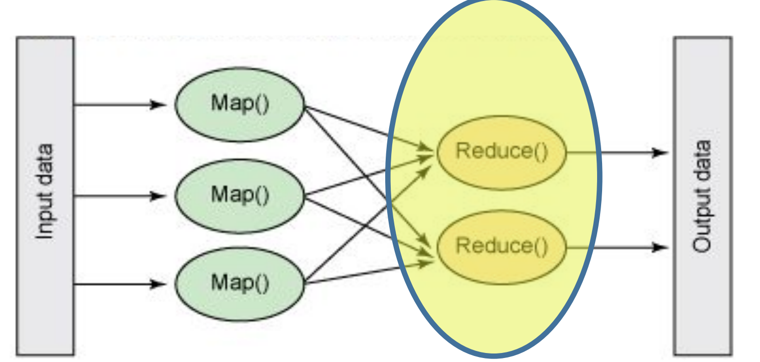
- Shuffle and sort
  - shuffle: transfers data from the mappers to the reducers
  - sort: ensures that all the input keys for a given reducer are sorted

Introduction to MapReduce

- Shuffle and sort in the Word Count Example



Introduction to MapReduce



- Reduce:
  - `reduce(k2, list(v2)) --> list(v3)`
- The reduce function is called **once for each unique key** emitted from the Mapper.
- The Reducer **has an iterator for all values for each key**.
- Produce the output to the output directory defined by the MapReduce job.

Introduction to MapReduce

- Reduce in the Word Count example

Reducer1

(cat, list(1,1,1)) → reduce() → (cat, 3)

(cow, list(1)) → reduce() → (cow, 1)

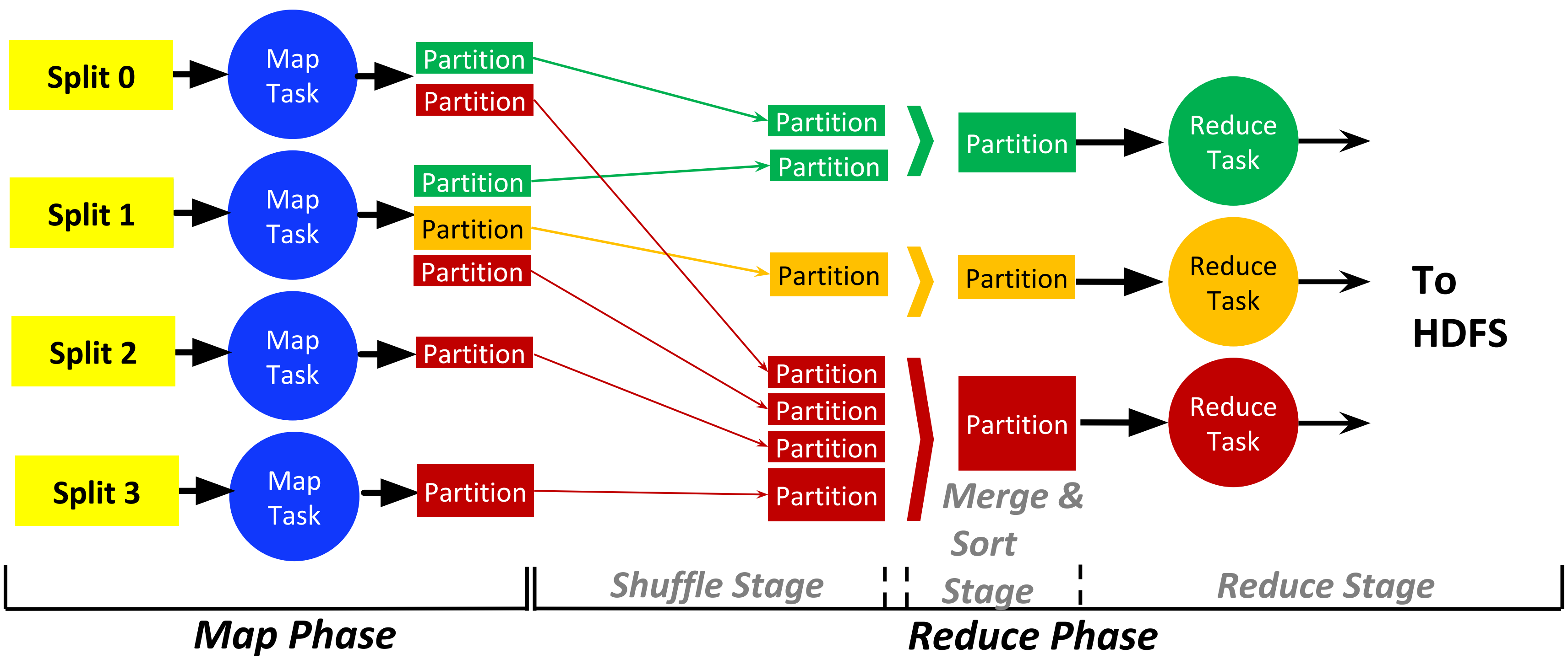
Reducer2

(dog, list(1)) → reduce() → (dog, 1)

(duck, list(1)) → reduce() → (duck,1)

MapReduce In a Nutshell

- MapReduce incorporates two phases
  - Map Phase
  - Reduce phase



Parallelism in MapReduce

- Mappers run in parallel, processing different input splits and creating intermediate Key-Value pairs
- Reducers also run in parallel, each working on a set of keys based on the partitioning function
    - By default, the partitioning function is a hash function
- Although the shuffle can start early, however, the reduce function cannot start until all mappers finish and all data is shuffled, i.e. barrier

MRUnit: TDD for MapReduce

- MRUnit is a unit test framework for MapReduce
- Allows you to define your input and expected output for the map and reduce functions
- This will allow you to test your map and reduce functions
- The MRUnit test cases are provided for the Word Count task

Using MRUnit

- Tests supported
  - Map Test to test map()
  - Reduce Test to test reduce()
  - MapReduce Test to test both
- Steps to Map Test
  - Step 1: Create your Mapper
  - Step 2: Create map test using MRUnit
  - Step 3: Set the input and output records
  - Step 4: Implement your map function
  - Step 4: Run locally to evaluate the test

MRUnit: Example map() test

```
// the test code is under the test source folder, similar to JUnit 5 test code
// run “mvn test” to run the test
public class WordCountMapTest extends TestCase {
    @Test
    public void testWordCountMapper() throws IOException {
        driver.withInput(new Text(""), new Text("cat cat dog"))
                .withOutput(new Text("cat"), new VIntWritable(1))
                .withOutput(new Text("cat"), new VIntWritable(1))
                .withOutput(new Text("dog"), new VIntWritable(1))
                .runTest(false);
    }
}
```

MRUnit: Test the MR workflow

- Use LocalJobRunner to test the whole MR workflow
  - Runs the MapReduce workflow in memory
- Steps to follow:
  - Define the configurations similar to the configurations of a real MapReduce job
    - Input path, output path
    - Mapper class, reducer class
    - etc.
  - Test if the job can be successful

Task 2: Word Count in MapReduce

- Implement Word Count as an example in MapReduce using MRUnit
- We provide you with the MRUnit test cases
  - to test the implementation of map and reduce functions
  - to test if the MapReduce application can run successfully w/ LocalJobRunner on a local dataset
- Your task is to pass the test cases
- If you can pass the test cases, the LocalJobRunner will generate the output to a local path

Running a Hadoop MR Job from the Command Line

- Create a cluster as per the AWS EMR section
  - created via Terraform (recommended) or web console
- SSH into the master node
- Run the MapReduce job in hadoop

```
> hadoop jar project1.jar
edu.cmu.scs.cc.project1.WordCount input-path
output-path
```

Troubleshooting EMR and MapReduce

- As you run the jobs with the large dataset, you can still run into errors despite the tests because of:
  - Resource limit, e.g., OutOfMemory
  - Malformed input data
- Aggregate the distributed log chunks into a single file will enable you to search all logs at once
- To retrieve the aggregated logs, run the following command on the master node

```
yarn logs -applicationId <applicationId>
```

- The first 3 questions in runner.sh will help you practice how to use grep to search the log files

Task 3: Wikipedia MapReduce application

- Put what you have learned together
- Design and implement a MapReduce application to:
  - Filter out elements based on the filtering rules in the data filtering task. Reuse your code.
  - Get the input filename from within a Mapper
  - Aggregate the pageviews from hourly views to daily views
  - Calculate the total pageviews for each article
  - Print the popular article that has over 100,000 page-views (100,000 excluded)

Task 4: Data Analysis with Pandas

- Now that you have filtered the monthly data, we are ready to analyze the data to answer some interesting analytics questions.
- The questions are in file **`runner.sh`**, and to solve these analytics questions, you will need an effective tool named pandas which is a Python package providing fast data structures for data analysis.

Progressively Solve Data Science Problems with Jupyter Notebook

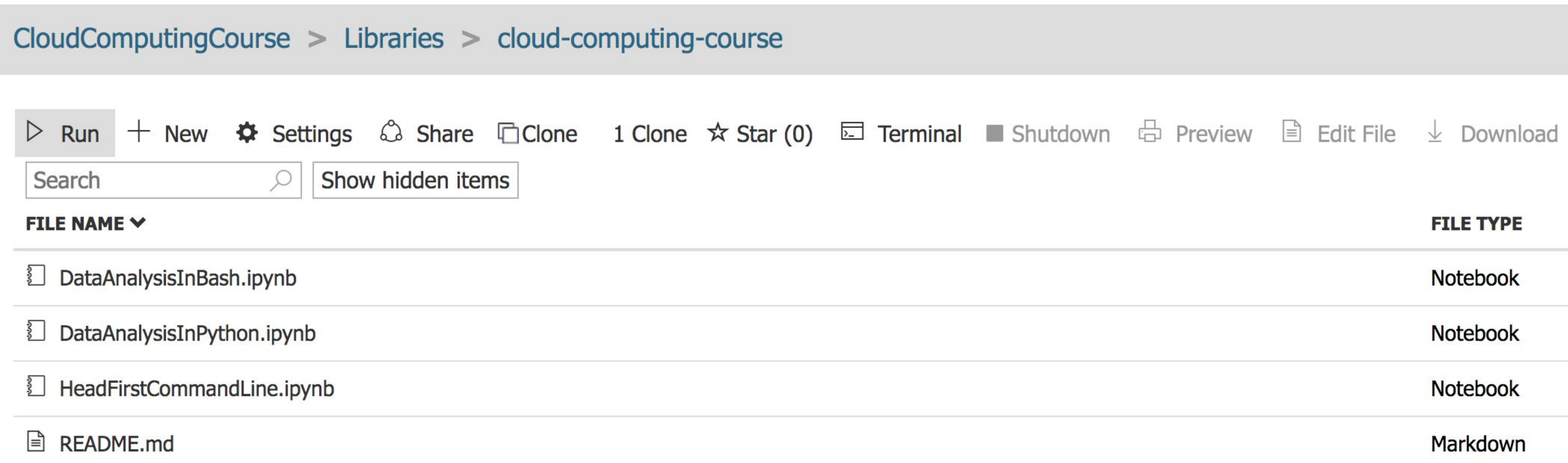
Why Jupyter Notebook?

- Interactive Computing
  - "save" your progress at the latest checkpoint

- Persisted Output and Reproducible Analysis
  - write data analysis reports and share with others

Progressively Solve Data Science Problems with Jupyter Notebook

- Finish the Jupyter Notebook primer
    - Practice tutorials in the Azure Notebooks library [15-319/15-619: Cloud Computing Course]



Project 1.1 Workflow

- Launch an EC2 instance with a specified AMI
  - We recommend using Terraform
- Finish tasks:
  - Data Pre-processing
  - Word Count in MapReduce
  - Wikipedia MapReduce
  - Data Analysis task
- Complete and run the script
  - /home/<andrew_id>/Project1_1/runner.sh
  - Answer a set of questions by providing the commands/code inside runner.sh
- Submit your code for grading
  - Complete the `references` file in JSON format
  - Execute `submitter` to submit your code
- Finish Project Reflection (graded) before the deadline
- Finish Project Reflection Feedback for 3 students
  - Within 7 days **after** the project deadline

Grading of Your Projects

- Code submissions are auto-graded
- Scores will be made available on http://theproject.zone
  - it may take several minutes for your score to show
  - the submissions table is updated with every submission
- We will grade all the code (both auto and manually)
- We use Checkstyle, PEP8 and shellcheck to check your coding style, which worths 5 points

Online Mob Programming

- Last week
    - Completed the OMP Primer
        - Created the AWS IAM role and assumed the role to get ready to access Cloud 9
    - Signed up with your available time slots
- This week
    - Mob Programming Training Session
        - Practice and get familiar with OMP workflow with an easy task
        - You should have received the email from us about your scheduled time to participate
        - Make sure that you set up Cloud 9 access successfully beforehand
            - Follow the steps outlined in the OMP Primer

Reminder: Deadlines

- Friday, September 7, 2018 at 23:59 ET
  - Quiz 1
- Sunday, September 9, 2018 at 23:59 ET
  - Project 1.1 (including Project Reflection)
- Sunday, September 16, 2018 at 23:59 ET
  - Project 1.1 Reflection Feedback
- ASAP, at the latest 9/10/2018 at 23:59 ET
  - Academic Integrity Course Quiz