

15-319 / 15-619 Cloud Computing

Weekly Overview 3
February 16th, 2021



Agenda

- Reflection from last week
- OLI content
- MapReduce programming model
- P1.2 tasks overview
- P1.2 grading
- Logistics update



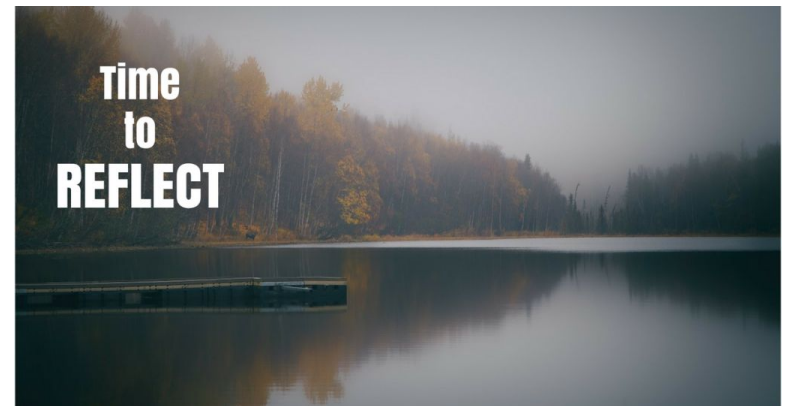
Reflection

- Conceptual content on OLI
 - Modules 1, 2, Quiz 1
- Project theme - **Sequential Analysis**
 - Wiki Data Processing Task
 - Sequential Analysis of **100s MB** of wikipedia data, adopt **Test Driven Development (TDD)**
 - Data Analytics Task
 - Write analytics code on Jupyter Notebook using the pandas library
 - Identify the limitations of sequential programs



Guideline for Project Reflection

- Describe your approach in solving each task in this project
 - Please share your
 - approach, challenges faced, how you overcame issues, and lessons learned
 - If you came up with a novel solution!
- However, please:
 - Do not share your code or pseudocode
 - Do not share details about your solution



Module 3: Data Center Trends

- Definition & Origins
 - Infrastructure dedicated to housing computer and networking equipment, including power, cooling, and networking
- Growth
 - Size (No. of racks and cabinets)
 - Density
- Efficiency
 - Servers
 - Server Components
 - Power
 - Cooling



Facebook data center

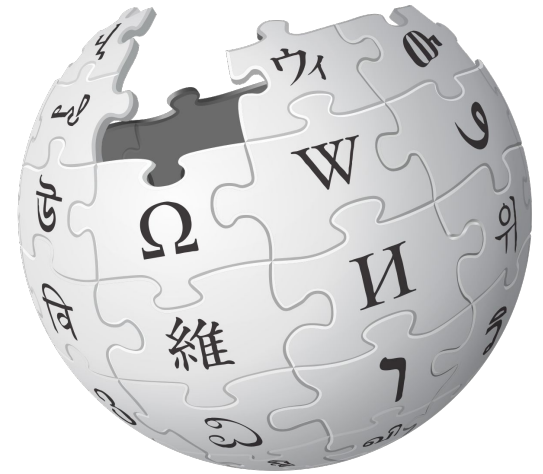


Module 4: Data Center Components

- IT Equipment
 - Servers : rack-mounted
 - Motherboard
 - Expansion cards
 - Types of Storage
 - Direct attached storage (DAS)
 - Storage area network (SAN)
 - Network attached storage (NAS)
 - Networking
 - Ethernet, protocols, etc.
- Facilities
 - Server room
 - Power (distribution)
 - Cooling



Project 1



- Identify Trending Topics on Wikipedia
 - Use the hourly pageviews dataset.
- Project 1.1: (Last Week)
 - Find trends from a single hour of data.
- Project 1.2: (This Week)
 - Find trends with a 30-day dataset using MapReduce.
 - Data collected from March 8th to April 6th in 2018.



Limitations of sequential programs

- Your data-preprocessing program works well with an hour's worth of data, what if you had 744 times that amount?
 - Would it just take 744 times as long?
 - Can your solution actually work with that much data?
- Methods to process a large dataset
 - Sequential program might not scale \Rightarrow a parallel solution
 - One EC2 instance might lack capacity \Rightarrow a large distributed cluster
- Challenges with distributed solutions
 - How to partition and distribute the tasks and data?
 - How would the nodes communicate and collaborate?
 - 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
 - Data partition and distribution
 - Management of communication across nodes
 - Deal with unreliable hardware and software



The MapReduce programming model

- Handling failure gracefully
 - Failure of a single machine will not cause the failure of the whole job
 - A task failure on one node can be resolved by rerunning the task on other nodes
- Reduce the communication cost
 - Data is stored in a distributed manner with replication
 - Exploiting data locality
- Easy to program
 - The minimal code you need to implement is only the map and reduce functions

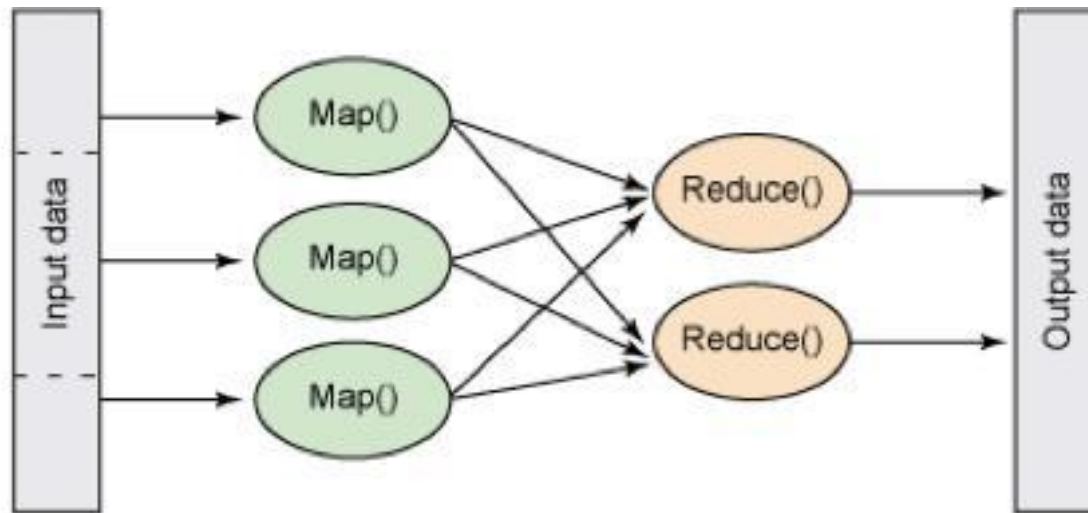
https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html



Overview of MapReduce

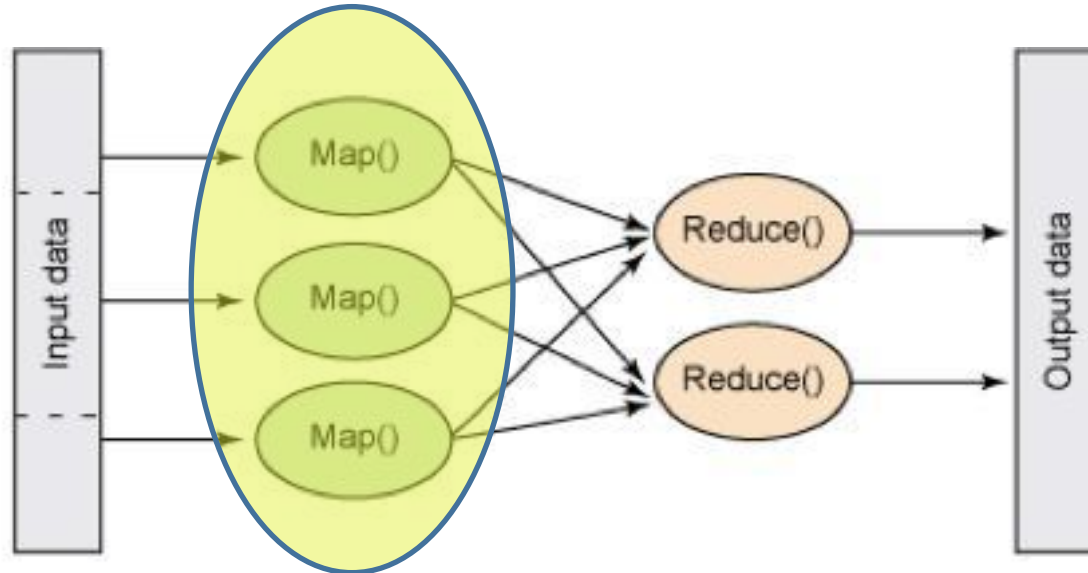
- Map: Apply some function across all data points
- Shuffle and sort
- Reduce: Aggregate this data and format for output

IN PARALLEL, ON A DISTRIBUTED FILE SYSTEM



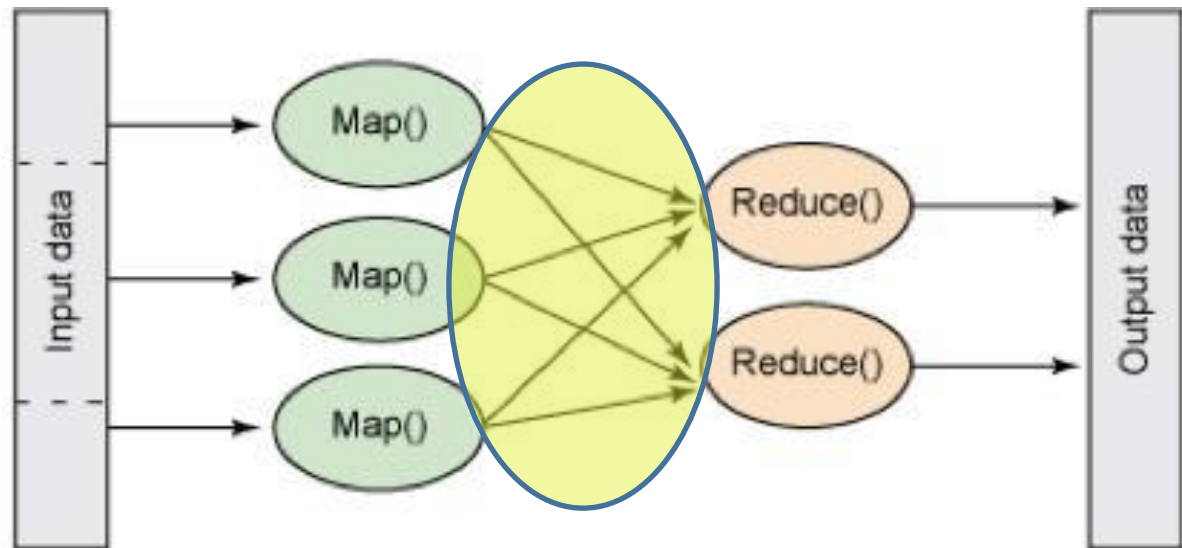
The Map phase in MapReduce

- 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)`



Intermediate Data

- Map
 - `map(k1,v1) --> list(k2,v2)`
- `k2,v2` is an “intermediate key-value pair” because it is
 - the output of the Mapper
 - the input of the Reducer

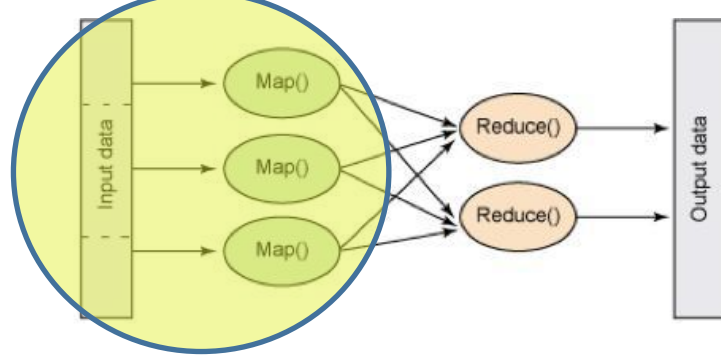


Word Count Example

- Input \Rightarrow Word Count \Rightarrow output
- Content of an input file:
 - cat cow
 - duck
 - dog cat
 - cat
- Output:
 - cat, 3
 - cow, 1
 - dog, 1
 - duck, 1



Map in Word Count



INPUT DATA

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

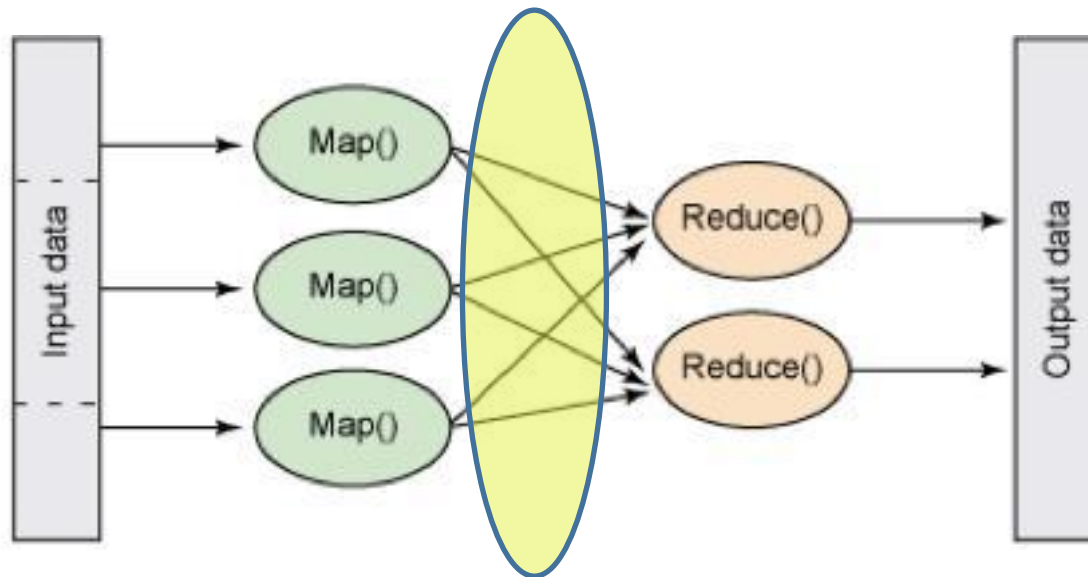
k2,v2 pairs:

(cat, 1)



The Shuffle and Sort in MapReduce

- Shuffle: transfers data from the mappers to the reducers
- Sort: sort intermediate data (k2, v2) by key



Shuffle and sort in the Word Count Example

k2,v2 pairs:

(cat, 1)

(cow, 1)

(duck, 1)

(dog, 1)

(cat, 1)

(cat, 1)

Partition

Shuffle

Reducer1

(cat, 1)

(cow, 1)

(cat, 1)

(cat, 1)

(cat, list(1,1,1)) → reduce()

(cow, list(1)) → reduce()

Reducer2

(duck, 1)

(dog, 1)

(dog, list(1))

(duck,

list(1))

→ reduce()

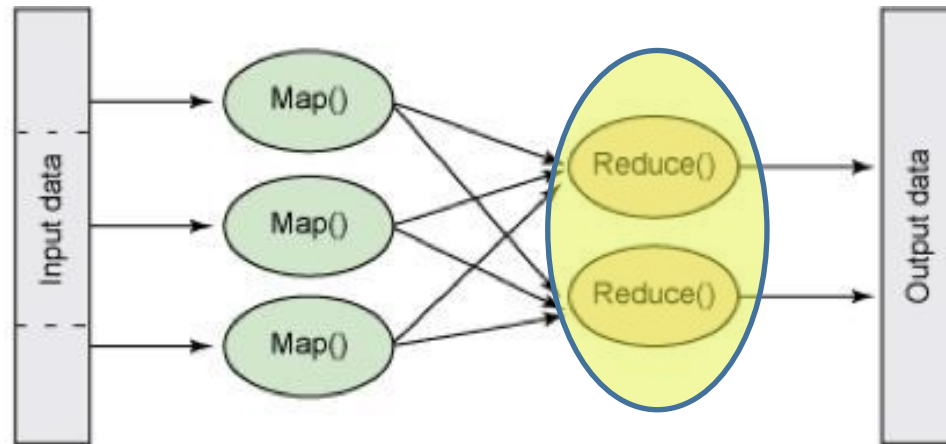
→ reduce()

Sort



The Reduce Phase in MapReduce

- Reduce:
 - `reduce(k2, Iterator(v2)) --> Iterator(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 directory defined by the MapReduce job.



Continuing the Word Count Example

Reducer1

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

(cow, list(1)) → reduce() → (cow, 1)

Reducer2

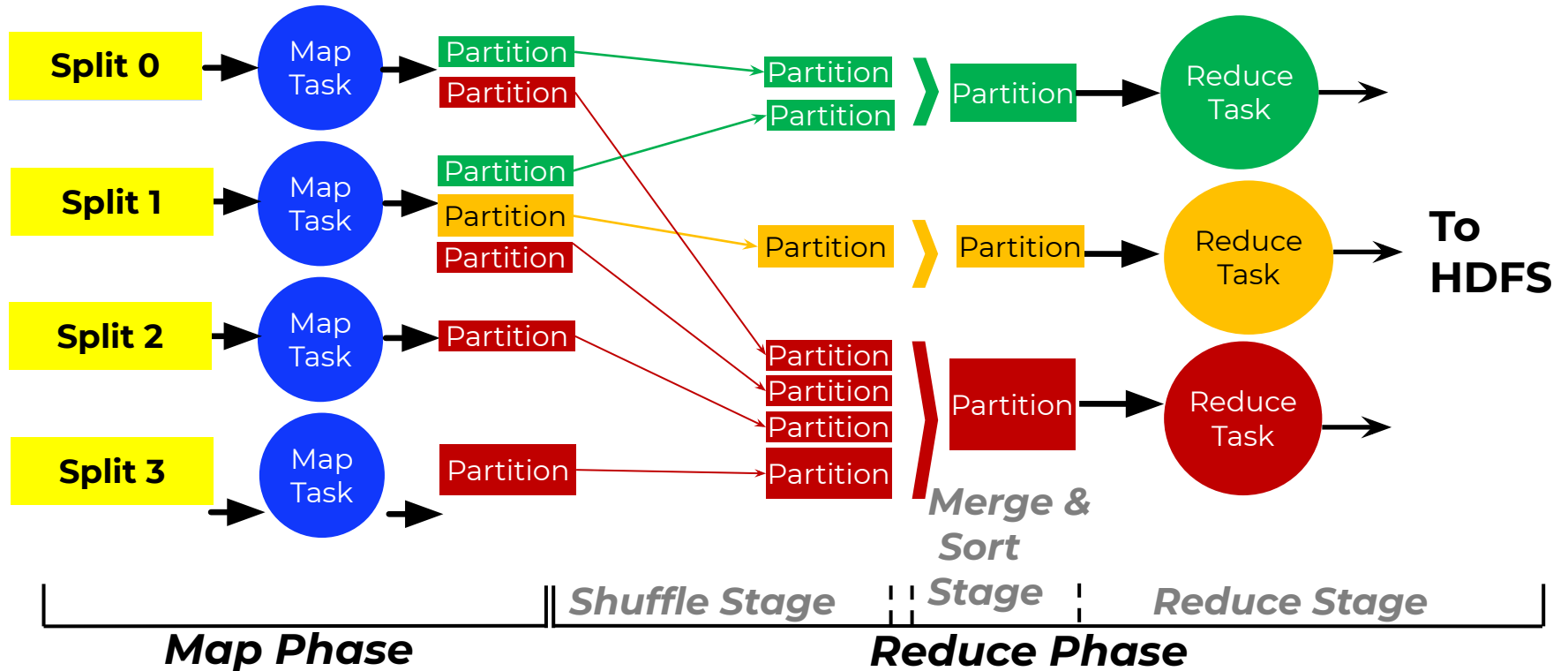
(dog, list(1)) → reduce() → (dog, 1)

(duck, list(1)) → reduce() → (duck, 1)



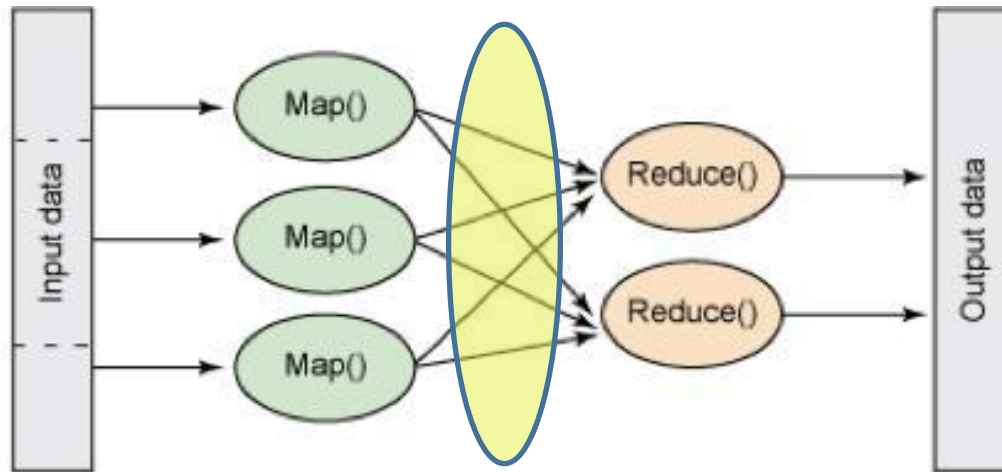
MapReduce In a Nutshell

- MapReduce incorporates two phases
 - Map Phase
 - Reduce phase



How about Combiner?

- What is a combiner?
 - An intermediate Reducer!
 - Runs on the intermediate data of the mapper.
- It can improve performance on large datasets, **but** your Reduce method must be both:
 - Commutative
 - Associative



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

TEST LOCALLY!
TEST SMALL!



Using MRUnit

- Tests supported
 - Map Test to test map()
 - Reduce Test to test reduce()
 - MapReduce Test to test both
- Steps to create Map Test
 - Step 1: Create map test using MRUnit
 - Step 2: Create your Mapper
 - Step 3: Set the input and output records
 - Step 4: Implement your map function
 - Step 5: 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);
    }
}
```



Test the MR workflow

- Use LocalJobRunner to test the MR jobs
 - Runs the MapReduce workflow in memory locally
- 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 will be successful

TEST LOCALLY!
TEST SMALL!



Troubleshooting EMR and MapReduce

- During a full run, you can still have errors!
 - Resource limit, e.g., OutOfMemory
 - Malformed input data
- Aggregating distributed log chunks into a single file will help 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

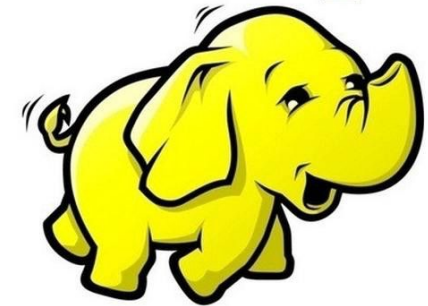


Security Exploit: Visit the YARN UI without opening unsafe ports to the public

- In the previous offerings of our course, students' Hadoop clusters were **compromised and implicated in DDOS attacks**.
- Follow the instructions in "Visit the YARN UI without opening unsafe ports to the public" in the writeup.
- Do not expose the port 8088 to the public.
- Otherwise, your Hadoop cluster will be vulnerable to a security exploit that may easily make you exceed the project budget due to heavy malicious egress data traffic and you will incur a **100% penalty**.
- **This applies to any Hadoop cluster, for example, EMR, GCP Dataproc, etc.**



hadoop



MapReduce Questions?

- Review: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
- Reach out to us on Piazza, or come by Office Hours and ask!



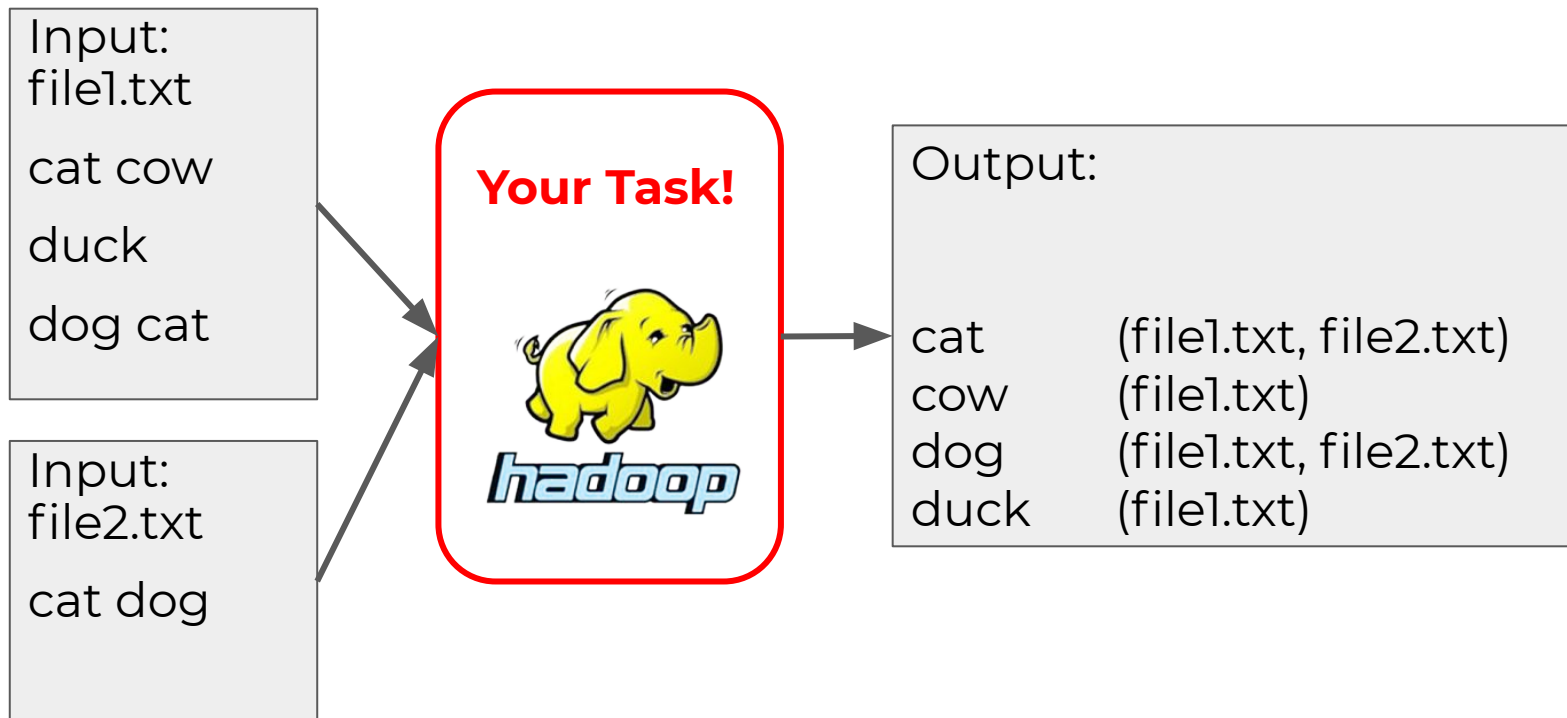
Project 1.2: Big Data Analytics

- Expand upon the analysis conducted in Project 1.1 to now process an entire month's worth of data
- Continue learning about using Test Driven Development, now with MRUnit
- Experiment with developing, testing, deploying, and debugging applications running on a cluster
- Practice writing robust, defensive programs
- Continue learning about Data Analysis using some more Linux and Python skills



Task 1: Inverted Index in MapReduce

- An index that maps words to the file where they appear



Your Task

Implement Inverted Index with MapReduce using TDD

- A worked example of WordCount and the test cases are provided for you to learn from
- We provide you with the test cases for Inverted Index
 - to test the implementation of map and reduce functions
 - to test if the MapReduce application can run successfully w/ LocalJobRunner on a local dataset
- You must pass these 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
 - provision via Terraform
 - SSH into the master node
- Run the MapReduce job in hadoop

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

- NOTE: input-path and output-path are HDFS file paths



Task 2: Wikipedia MapReduce application

- Put everything together into a single application
- Design and implement a MapReduce application to:
 - Filter out records based on the filtering rules in the data filtering task (Reuse your code from P1.1 here!)
 - 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 popular articles that has over 100,000 page-views (100,000 excluded)



Task 3: Data Analysis with Pandas

- Now that you have filtered and aggregated the monthly data, you are ready to analyze the data to answer some interesting analytics questions.



Project 1.2 Workflow

- Launch an EC2 instance with a specified AMI
 - Develop and test InvertedIndex locally!
- Provision EMR cluster(s) and practice:
 - Inverted Index in MapReduce
 - Wikipedia MapReduce
 - **Do not expose port 8088 to the public.**
- Complete the data analysis steps very similarly to Project 1.1
 - /home/clouduser/Project1/runner.sh
 - Answer a set of questions by providing the code inside data_analysis.ipynb
- Submit your code for grading
 - Complete the references file in JSON format
- Finish Project Reflection (graded) before the deadline



Grading of Your Project

- Code submissions are auto-graded
- We will grade all the code (both auto and manually)
- We auto grade your coding style, which is worth 5 points
- Coding style will also be manually graded
 - high quality code
 - sufficient comments
 - self-explanatory and modularized code
- We will also test the quality and coverage of the MRUTests you write
 - If your tests are really good, there are bonuses.



Course Logistics and Advice

- How to ask good questions on Piazza
- “A way” to structure your work in this course
 - Write-ups & documentation are your friends
- Developing for a remote environment
- Start early and explore!



amazon
EMR



Important Notice

- **DO NOT EXPOSE YOUR AWS CREDENTIALS!**
- **DO NOT EXPOSE YOUR GCP CREDENTIALS!**
- **DO NOT EXPOSE YOUR Azure CREDENTIALS!**
 - ApplicationId, ApplicationKey
 - StorageAccountKey, EndpointUrl
 - Github
 - Bitbucket
 - Anywhere public...
- **-100% penalty if you expose any of your credentials**



Logistics

- Quiz 2 (OLI Modules 3 & 4)
 - Due on Friday, Feb. 19th, 2021, 11:59 PM ET
- Project 1.2
 - Due on Sunday, Feb. 21st, 2021, 11:59 PM ET
- **Project 1.1 Reflection Feedback (graded)**
 - Due on Sunday, Feb. 21st, 2021, 11:59 PM ET



Best wishes. You got this!

