CS15-319 / 15-619
Cloud Computing

Recitation 12
April 8th, 2014
Announcements

• Encounter a general bug:
  – Post on Piazza

• Encounter a grading bug:
  – Post Privately on Piazza

• Don’t ask if my answer is correct

• Don’t post code on Piazza

• Search before posting

• Post feedback on OLI
Piazza Questions

•STDOUT, STDERR redirection
  – ./run.sh 1> result.out 2>error.out

•Question 10
  – Some students have longer latency on Q10, this will be regarded manually.

•Security group
  – Both launch instance and HBase master node should be configured.
DynamoDB vs. HBase

• Data Model
  – Key-value vs. Column oriented Key-value

• Proprietary vs. Open source

• Cost
  – DynamoDB: Provisioned Throughput Capacity
  – HBase: Instance + EMR

• Limitations:
  – DynamoDB:
    • Item size: 64 KB
    • Query result: 1 MB
Project 3, Module 5 Reflections

• When to use DynamoDB:
  – Required throughput is determined
    • e.g. steady arrival rate
  – Easier to implement and scale
  – Enough budget
    • Charged by provisioned throughput capacity

• When to use HBase:
  – Low cost
  – Less constraints (Item size, query result)
  – Open source
Module to Read

• UNIT 5: Distributed Programming and Analytics Engines for the Cloud
  – Module 16: Introduction to Distributed Programming for the Cloud
  – Module 17: Distributed Analytics Engines for the Cloud: MapReduce
  – Module 18: Distributed Analytics Engines for the Cloud: Pregel
  – Module 19: Distributed Analytics Engines for the Cloud: GraphLab
Project 4

• MapReduce
  – Hadoop MapReduce

• Input Text Predictor: NGram Generation
  – NGram Generation

• Input Text Predictor: Language Model and User Interface
  – Language Model Generation
Google

- Inverted index
  - Word -> {doc1, doc2, ...}

- Ranking ...
Google

• Google Instant
  – Input text predictor
  – Generate a list of phrases in a text corpus with their corresponding counts
  – Rank the probability
MapReduce Reflection on Project 1

- The idea of MapReduce

How many times does the word “Apple” appear in these books?

I heard 6 “Apple”s!
MapReduce Reflection on Project 1

• The idea of MapReduce

How Do I know Who is the “Apple” Man?

You Don’t!
MapReduce Reflection on Project 1

• The idea of MapReduce

<table>
<thead>
<tr>
<th>Mapper</th>
<th>Blueberry, 1</th>
<th>Blueberry, 1</th>
<th>Apple, 1</th>
<th>Orange, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blueberry, 1</td>
<td>Apple, 1</td>
<td>Apple, 1</td>
<td>Apple, 1</td>
</tr>
<tr>
<td></td>
<td>Apple, 1</td>
<td>Apple, 1</td>
<td>Orange, 1</td>
<td>Blueberry, 1</td>
</tr>
</tbody>
</table>

Map Phase

Reducer

Magic Box (Shuffle, sort, merge)

Reduce Phase
MapReduce This Week

• The idea of MapReduce

Map Phase

Black Box (Shuffle, sort, merge)

Reduce Phase

Jar instead of streaming
MapReduce

• Mapper
  – Input: **key-value pairs**
    • lines in files in our project
  – Output: **key-value pairs**
    • **Keys** are used in Shuffling and Merge to find the Reducer that handles the intermediate output for that specific key. (in our example, Apple, Orange and Blueberry are keys)
    • **Values** are messages sent from mapper to reducer (in our case it is always 1)
    • Mappers’ output is intermediate because reducers will receive the key-value pairs and take them as input.
MapReduce

• Reducer
  – Input: **key-value pairs**
  – Output: **key-value pairs**
    • the final result we need
    • Depends on what we want, our code should process the value in the key-value pairs that we got accordingly (in the word count example, we just add up all the values).
HDFS

• Hadoop Distributed File System
• Open source version of Google File System
MapReduce and HDFS

• Workflow
Project 4 Module 1

• Write a MapReduce program that will build an inverted index of documents

• Have to use EMR Custom Jar
  – CANNOT use EMR streaming
# Upcoming Deadlines

## Project 4:

<table>
<thead>
<tr>
<th>Project 4</th>
<th>MapReduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop MapReduce</td>
<td>Checkpoint</td>
</tr>
</tbody>
</table>

Available Now
Due 4/13/14 11:59 PM

## Unit 5:

**UNIT 5: Distributed Programming and Analytics Engines for the Cloud**

- **Module 16: Introduction to Distributed Programming for the Cloud**
- **Module 17: Distributed Analytics Engines for the Cloud: MapReduce**
Demo Outline

• Introduction to Hadoop & HDFS
• Code for MapReduce example
• Demo of using custom Jar
Hadoop

• Apache Hadoop
  – A framework for running applications on a large cluster of commodity hardware
  – Implements the MapReduce computational paradigm
  – Uses HDFS for data storage
  – Engineers with little knowledge of distributed computing can finish the code in a short period

• MapReduce
  – A programming model for processing large data sets using a parallel distributed algorithm
HDFS

• Paper

• Purpose
  – Implemented for running Hadoop’s MapReduce applications with distributed storage
  – An open-source framework which can be used by different clients with different needs
Custom Jar

• What is custom Jar
  – Customize your java MapReduce program

• Why custom Jar
  – More resources: HDFS/HBASE/S3
  – More job configuration flexibility
  – More control of how the resources are utilized
Demo

• WordCount program demo
  – Code review
  – Launch EMR Cluster
  – Compile Java code
  – Generate WordCount input
  – Run WordCount program
Recommendations

• Test for correctness with small datasets first
• DO NOT need to restart a new cluster
  – EMR will charge you one hour of usage for instances even though your EMR job failed to start
• Pay attention to your code efficiency
• Version of Hadoop
  – should match the version of your API
• Start early
• Thanks