Problem selection guidelines

Goal: A project of challenge to your group. No two groups should work on very similar projects.

A. Big data analytics related topics

Minimum requirement: Implement a big data analytics project of significance (meaningful problem – what conclusion(s) you'd get from the data analysis; large data set; cover all/most stages of the lifecycles; visualize and explain the results, …)

Challenges: (examples only, open to your choice)
(1) Perform big data analytics in parallel using Dask framework (dask.pydata.org/) or other software tools.
(2) Add machine learning components using SciKit-Learn or other software tools.

B. *Cloud Computing related topics*

Minimum Requirement: Implement a meaningful project on a cloud system

Suggestions:
(1) MapReduce tasks (e.g. Netflix Example as described below)
(2) Non MapReduce tasks (e.g. Elastic Search, build your own Dropbox – see description below.)

Note: More details will be provided during and after the topic selection stage.

Netflix Example: Instructions for writing MapReduce jobs and running it in a Docker container.

(1) *Problem statement: Processing the Netflix Dataset*
Netflix hosted a $1M competition a few years ago to find out the best movie recommendation algorithm (http://www.netflixprize.com/). We are going to use part of that dataset to practice the MapReduce.

(2) Dataset download: https://s3-us-west-1.amazonaws.com/cs499-cc/a3-dataset.zip which contains the users' rating on a set of movies. Each row represents a rating of a movie by some user. The dataset contains 1821 movies and 28978 users in all. Ratings are integers from 1 to 5. The training set has 3.25 million ratings.

(3) You need to write two MapReduce jobs to answer the following two questions:
1) What are the top 10 movies that have the highest average ratings? Tell us the titles (you can find the titles in a separated file).
2) Who are the top 10 users that posted the most reviews? Tell us their user ids.

Note that you don't have to output the final result directly from your Reducer. You can have a separate Java/python program to take the MapReduce output to figure out the rankings, the top 10, or find out the titles. You can do whatever you need to get the result, but you must have two MapReduce jobs written to accomplish the core tasks.

This is a good tutorial to use for the basis of MapReduce (in Java):
https://examples.javacodegeeks.com/enterprise-java/apache-hadoop/hadoop-hello-world-exam

ple/

(4) *Run Your MapReduce in a Docker Container (Optional)*
You can use Eclipse to test your MapReduce program. However, you may also use Docker Container to run your MapReduce jobs. You need to install and learn how to use Docker Container (https://www.docker.com/). There might be a little bit learning curve in the beginning, but it is worth the time, because Docker is very popular in the industry today.

This Docker Image (https://hub.docker.com/r/nagasuga/docker-hive/) has everything setup for you already. You need to put all your java programs and dataset into this container, and run your program using hadoop cluster and the HDFS . Look for online resources and tutorials on how to run a jar file with hadoop and hdfs.

Once you have done, you need to push the container that includes your own jar file and dataset to the Docker repository (https://docs.docker.com/engine/getstarted/step_six/). And of course, you will need to have a Docker account.

ElasticSearch and Kibana:

This example describes the basic steps of using ElasticSearch and Kibana to build a data analysis dashboard. This assignment is very open. Feel free to build with your own idea.

(1) Pull Your Data and Insert the Data to ElasticSearch
You can decide what kind of data you want to pull from the Internet. However, the data should be something that changes over the time. For instance, Universal Studio ride waiting time, Amazon product price, flight ticket price, stock price, etc.

You just need to focus on one domain and use any language or tools to pull this data periodically, followed by inserting those data into your ElasticSearch server. You can use any language (i.e., Java, Python, nodejs) as long as ElasticSearch supports it. For instance, you can pull the ride waiting time once every 10 mins; or you can pull the Amazon product price one every hour. You need to write this program and run it in EC2, so that it keeps sending the data to ElasticSearch. You don't need to pull all the data - just choose some interesting ones depending on the domain and building one index in ElasticSearch is fine.

You should use AWS ElasticSearch service, which has a free-tier service to use. You can just make the ElasticSearch open to the public so that it is easy to access and control from your program.

(2) Build a Dashboard using Kibana to Visualize the Data
The AWS ElasticSearch automatically comes with a Kibana service. You can try to load your ElasticSearch index into Kibana, and design some interesting visualizations using the tools provided in Kibana. Finally, you need to put all those visualizations into a Dashboard and save the dashboard in Kibana.

Again, this assignment is very open. You can decide the type of data, the number of data samples to send, the number of indices, the number of visualizations, and the type of visualizations to use. Make sure you explain your idea and domain in a README file.

Build your own Dropbox
In this project you will develop the **backend** for a Dropbox-like webservice. You will be working with AWS S3 API and a web service platform of your choice. Below are the functions you need to implement.

(1) Local Disk File Watcher
(2) Local to S3 Sync
(3) Wrap Your Program as a Web Service
(4) Configuring CORS
(5) Test your Endpoint with the Frontend

C. Additional Information

1. GitHub Account

To have a GitHub (https://github.com/) account, sign up the (free tier) account.

Once you got the GitHub account, please go through the following GitHub official tutorials (https://help.github.com/) and install Git in your machine. You can also google tons of tutorials and videos about how to use Git and GitHub. If you have never used Git before, it would be super helpful if you can watch some tutorials and do some exercises. One good interactive guide is: https://try.github.io
And a series of good Git video tutorials can be found at:
https://www.youtube.com/watch?v=r63f51ce84A

2. AWS Account
Create an account in AWS (https://aws.amazon.com/), and make sure that you can login to the AWS console. Make sure that you check out the free AWS credit at https://education.github.com/pack

3. Instructions for Running a web application in an AWS EC2 server.

E.g. A Web Service Program to Host A Simple Web Page
You need to build a simple web service using either Python or Node.js. For Python, a simple framework such as Python Flask (http://flask.pocoo.org/docs/0.12/) is recommended. For Node.js, express (http://expressjs.com/) is a good option.

The web service only needs to host a static HTML web page. Try a very simple HTML page (e.g. a personal webpage to talk about yourself. No need to build a fancy HTML page, nor any HTTP APIs.) Just want to make sure that you can build a simple web service with the new languages and tools quickly.

You should develop and test your web service locally. Then, deploy that to an EC2 instance in Amazon. Make sure you choose the free-tier EC2 instance type. If you need help on EC2, please see the official developer guide at https://aws.amazon.com/ec2/getting-started/

You need to check in your web service source code to a new repository in your GitHub account.