15-319: PROJECT PHASE I-A – INTRODUCTION TO AMAZON EC2 AND HADOOP

Project Start Date: Jan 16, 2012
Project Due Date: Feb 15, 2012

LEARNING OBJECTIVES

This project is meant to get your feet wet with cloud computing using Hadoop as well as Amazon EC2 and S3 web services. You will configure and provision a Hadoop cluster on a number of leased Amazon EC2 instances and use S3 to backup your data before you terminate your cluster instances. You will develop a MapReduce program to solve a classical string matching problem and deploy it on your provisioned Amazon Hadoop cluster. When done with the project, you will understand how to provision and use various Amazon web services as well as how to create and deploy MapReduce programs on a public cloud.

PROJECT OBJECTIVES

This project has the following objectives:

1) Setup an Amazon Web Service (AWS) account with credit to enable you to work on the course project.
2) Get comfortable with the AWS web interface for VM management.
3) Provision a single instance manually and analyze instance performance: Run and analyze resource micro-benchmarks (CPU and Memory micro-benchmarks).
4) Provision and utilize a Hadoop cluster on Amazon using Apache Whirr and Amazon EC2 and S3 web services.
5) Learn how to develop and deploy MapReduce jobs on your Amazon Hadoop cluster.
6) Perform scalability (via scaling the number of EC2 instances) and sizing (via using different EC2 instance types) studies for a MapReduce application.

BACKGROUND

Amazon Web Services (AWS) is by far the most popular cloud computing provider and is widely credited with the introduction of innovate concepts in cloud computing. AWS offers various services such as Amazon Simple Storage Service (S3) and Amazon Compute Cloud (EC2). For this course, you will work on instances provisioned on Amazon EC2 and utilize S3 for persistent storage. For more information about AWS, please refer to: http://aws.amazon.com/

PROJECT GUIDELINES

This project phase, as well as others, requires access to a Linux machine as this allows you to install and run software such as Hadoop and Apache Whirr seamlessly. You can use any flavor of Linux as you prefer, however, the instructions in this document are suited for Fedora 15. If you currently do not have access to a Linux machine, your options are:

1) Install Linux on your own computer in a dual-boot configuration.
2) Install Linux as a Virtual Machine using Oracle's VirtualBox (http://www.virtualbox.com) or VMWare's VMWare Player (http://www.vmware.com/products/player/).
3) Get access to an existing Linux PC or Virtual Machine.

Amazon EC2 Guidelines

Amazon EC2 is a paid, public cloud service. Launch instances only when you are ready to run jobs. Please use the service responsibly, and always remember to terminate your EC2 instances as soon as you are done working on your projects.

PART 1: AMAZON EC2 ACCOUNT SETUP

Refer to the class Webpage for instructions on how to setup your Amazon AWS account and retrieve your AWS credits for your project. Please note that AWS user registration is known to take time, possibly a few hours to a day. After this step, please sign into AWS and access the AWS Management Console on your web browser. Click on the "Amazon EC2" tab in the console. You should be able to see a screen similar to Figure 1.

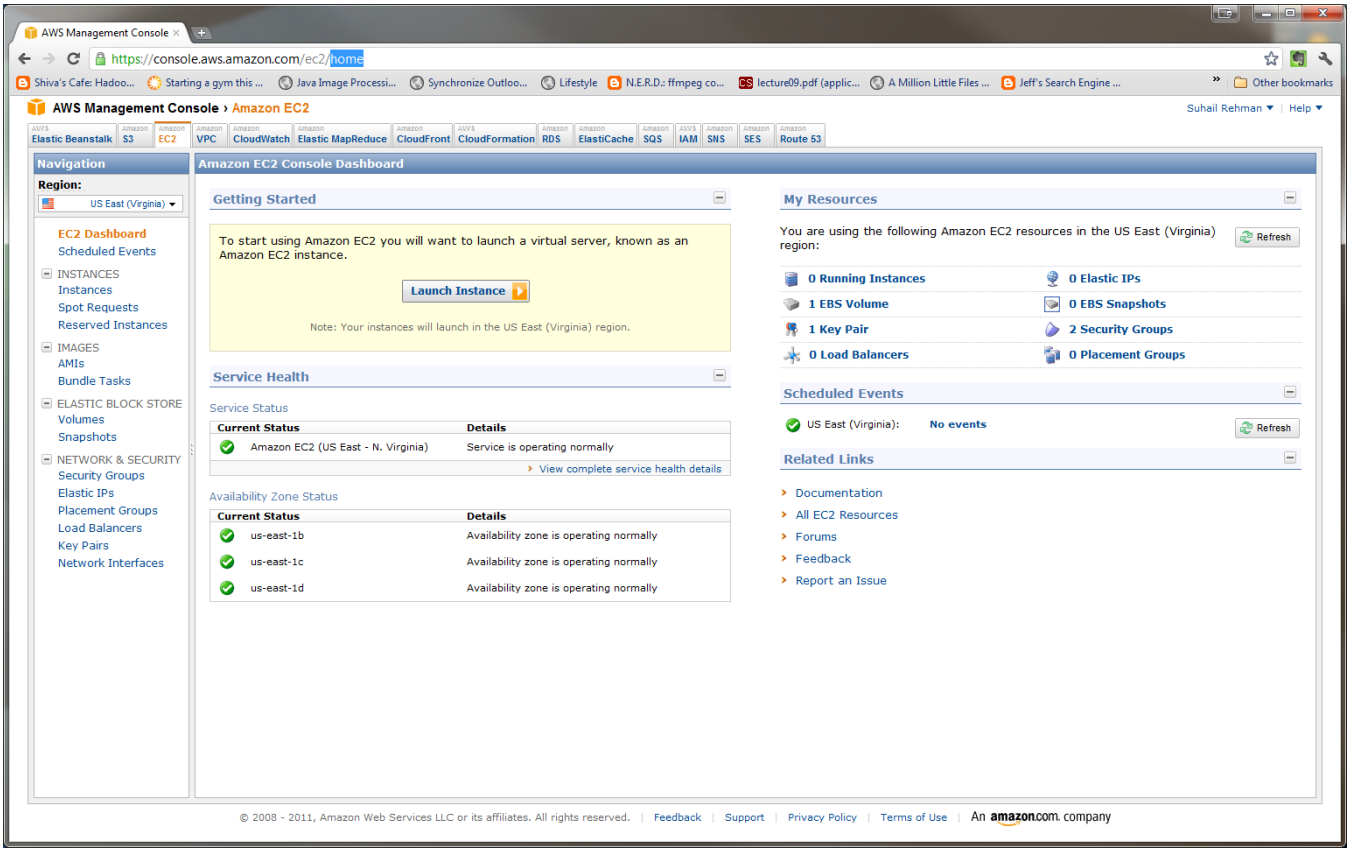

Figure 1 - The AWS Management Console

Deliverable 1: Screenshot of the AWS Management Console of your AWS account as a JPEG file named **Part1.jpg**

PART 2: YOUR FIRST AMAZON EC2 INSTANCE

Please read the following guide for reference:

Amazon EC2 Getting Started Guide (http://docs.amazonwebservices.com/AWSEC2/latest/GettingStartedGuide)

Once you have read the guide, please perform the following tasks:

1) Launch a Micro Instance (t1.micro) running the Basic 32-bit Amazon Linux AMI (AMI Id: ami-31814f58) using the AWS Management Console.
2) SSH to your Instance once it has been launched.
3) Poke around – Try out various linux commands that you are familiar with.

Tip: By default all ports on your Instance have been locked down. Ensure you have correctly setup the Firewall on your instance to allow SSH (Port 22) in case you are having trouble in connecting. Remember that you can always install software packages that you need using the command `sudo yum install <<package_name>>`. For example to install `emacs` simply type `sudo yum install emacs`.

Deliverable 2: Once you have successfully logged into your EC2 instance, run the `hostname` and `uname -a` commands and paste their outputs to a file named **Part2.txt**

PART 3: PERFORMANCE ANALYSIS OF YOUR INSTANCE

In this part you will run a few simple benchmarks on your provisioned EC2 instance.

1) Attempt to identify the physical CPU on the machine.
2) Download the benchmark applications from: http://www.qatar.cmu.edu/~msakr/15319-s12/projects/project1/files/ using the `wget` or `curl` command.
3) Unpack and install the `systester-cli` and `stream` benchmarks. For detailed instructions please refer to their respective websites (http://systester.sourceforge.net/ and http://www.cs.virginia.edu/stream/).
4) Run the `systester-cli` benchmark 10 times – record the last time reported in each run and plot them on a Line graph using Excel.
5) Run the `stream` benchmark 10 times – record the Rate (MB/s) for the Triad function for each run and plot them on a Line graph using Excel.

Deliverable 3: Once you have run the benchmarks and collected all the required information, store the data along with the line plots as described in steps 3 and 4 above in an excel file named **Part3.xlsx**

Note: Please terminate your instance once you have completed this part – Do not leave your instance running idle for more than a few hours.

PART 4: YOUR FIRST AMAZON S3 STORAGE

In this part, you will use Amazon Simple Storage Service (S3). Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. Amazon S3 provides a functionality to write, read, and delete files (objects in Amazon's parlance) of sizes 1 byte to 5 terabytes of data each. The number of objects that you can store is unlimited. Each object is stored in a ***bucket*** and retrieved via a unique, developer-assigned key. For more information on S3, please refer to: http://aws.amazon.com/s3/.

To upload, retrieve and manage data in Amazon S3, you will use the command line tool named s3cmd. Specifically, you will use s3cmd to build a bucket, remove a bucket, list buckets/objects, put files into a bucket, get files from a bucket, delete files from a bucket, and synchronize a directory tree as S3. For more information on s3cmd, please refer to: http://s3tools.org/s3cmd.

Once you have familiarized yourself with Amazon S3 and s3cmd, please perform the following tasks:

1) Launch a Micro Instance (t1.micro) running the Basic 32-bit Amazon Linux AMI (AMI Id: ami-31814f58) using the AWS Management Console.
2) SSH to your Instance once it has been launched.
3) Download the s3cmd tool and unzip it.
4) Configure the s3cmd tool:
    a. Go into the s3cmd directory and run **./s3cmd --configure**
    b. Enter your Amazon Web services Access key (can be obtained from http://aws-portal.amazon.com/gp/aws/developer/account/index.html?action=access-key)
    c. Enter your Access Secret Key (can be obtained from http://aws-portal.amazon.com/gp/aws/developer/account/index.html?action=access-key)
    d. Enter your Encryption password. Leave it blank if no password and press enter.
    e. Specify the path to your GPG program. The default path is /usr/bin/gpg (You can leave it blank and press enter- GPG encryption key is typically used for encrypting your files before sending them to Amazon. By using this, you can protect your data against unauthorized users.).
    f. Specify whether to use the HTTPS protocol and press enter.
    g. Specify whether to use an HTTP proxy server name and press enter.
    h. Check access with supplied credentials (press y and enter)
5) Build a bucket from your EC2 terminal by typing: **./s3cmd mb s3://my-new-bucket-name**
6) List all the buckets by typing: **./s3cmd ls**
7) Upload a file into your bucket by typing:
   ./s3cmd put <<file_name>> s3:// my-new-bucket-name/file_name
8) Download a file from your bucket by typing:
   ./s3cmd get s3:// my-new-bucket-name/file_name new_file_name
9) You can also create a bucket by using the AWS Management Console (go to https://console.aws.amazon.com/s3/home, log in, and create a bucket in the region of your EC2 instance.).

Deliverable 4: Include a screenshot that shows the execution of the commands in Steps 5, 6, 7 and 8 as **Part4.jpg.**

Note: Please terminate your instance once you have completed this part – Do not leave your instance running idle for more than a few hours.

PART 5: PROVISIONING A HADOOP CLUSTER AND RUNNING A MAPREDUCE JOB

In this part, you will use Apache Whirr to automatically provision and configure a Hadoop cluster on Amazon EC2. Apache Whirr is a set of libraries for running cloud services.

Whirr provides:

- A cloud-neutral way to run services. You don't have to worry about the idiosyncrasies of each provider.

- A common service API. The details of provisioning are particular to the service.
- Smart defaults for services. You can get a properly configured system running quickly, while still being able to override settings as needed.

You can also use Whirr as a command line tool for deploying clusters. More information on Apache Whirr is available at http://whirr.apache.org.

Refer to the class Webpage and follow the steps outlined in the **Apache Hadoop and Whirr install guide.** Once you have provisioned a 4-node cluster in Amazon EC2 using Whirr, you will run the Wordcount application that comes with the Hadoop distribution. Please complete the following tasks:

1) Inspect the wordcount MapReduce code and describe its logic using pseudo-code.
2) Using the appropriate HDFS commands, create a directory under HDFS of your cluster called `/wordcount/`.
3) Run the `randomtextwriter` MapReduce example job to generate 12 GB of text into the directory `/wordcount/in`.
4) Run the `wordcount` MapReduce example job to count the number of occurrences of each word in the file generated by the program in step 2. The output of this MapReduce job should be stored in `/wordcount/out`.
5) Use the appropriate HDFS command to retrieve the first 10 lines of output from the file `/wordcount/out/part-00000`.
6) Open your web browser and access the Hadoop web interface to view the statistics of your job. (e.g., http://ec-50-17-101.139.compute-1.amazonaws.com:50030/; replace the address with the address you obtained while provisioning your cluster).
7) Build a bucket in S3 named PART5_Results using s3cmd in an EC2 instance.
8) Move your log files (found under $HADOOP_HOME/logs) and your wordcount output to PART5_Results for persistent storage (i.e., so that you can use them after you terminate your Hadoop cluster).

Tips: You may refer to the following guides to help complete this part of the project:
Hadoop Commands Guide: http://hadoop.apache.org/common/docs/r0.20.2/commands_manual.html
The HDFS Guide: http://hadoop.apache.org/common/docs/r0.20.2/hdfs_shell.html

Deliverable 5: A text file called **Part5a.txt** that contains the pseudo-code of the wordcount program you wrote in Step 1, the execution time of the `wordcount` job you ran in Step 3 as well as the output of the command you executed in Step 4. In addition, include a screenshot of the web interface you obtained in Step 5 as **Part5b.jpg.** Lastly, include a screenshot of the EC2 terminal you obtained in Steps 6 and 7 as **Part5c.jpg**.

Note: Please DO NOT terminate your instances once you have completed this part – Please move directly to part 5.

PART 6: SEQUENTIAL VERSUS PARALLEL WORDCOUNT

In this part, you will develop a sequential version of the wordcount program and compare against the MapReduce version which you ran in part 4. Please complete the following tasks:

1) Implement a sequential version of the wordcount problem using Java.
2) Run your implementation on 1 EC2 instance of the 4-node cluster you created in part 4. In addition, use the same 12 GB dataset you generated in part 5.
3) Report the execution time of your wordcount sequential program and compare against the execution time of the wordcount job that you ran in part 5.

4) Discuss your insights concerning the performance tradeoffs of sequential versus parallel wordcount application.

Deliverable 6: A word file called **Part6.docx**, in which the first line contains the execution time of the `wordcount` job that you ran in Step 3 as well as your discussion in Step 4.

Note: Please terminate your instances once you have completed this Part – Do not leave your instances running idle for more than a few hours.

PART 7: WORDCOUNT SCALING STUDY

1) Provision 3 different Hadoop Clusters of various sizes (other than 4 slave nodes. E.g., 8, 12, and 16 slave nodes) and run the `wordcount` job by repeating the steps 2-5 in Part 5 above.
2) Record the execution time of each wordcount run under each cluster size you have provisioned in Step 1.
3) Calculate the cost of running the job for each cluster that you have provisioned using the Amazon instance pricing chart (http://aws.amazon.com/ec2/pricing/).
4) Plot the collected numbers in Steps 3 and 4 using a Column chart in Excel.
5) Analyze and discuss the wordcount performance under different cluster sizes.

Deliverable 7: An excel file named **Part7a.xlsx** that contains a table with the execution time and pricing comparison of the `wordcount` job under each cluster size and the column chart as described in Steps 3 and 4 above. Furthermore, include a word file named **Part7b.docx** that contains your analysis and discussion as described in Step 5.

Note: Please terminate your instances once you have completed this Part – Do not leave your instances running idle for more than a few hours.

PART 8: INSTANCE SIZING STUDY

1) Provision 3 different Hadoop clusters using the following instance types:
    a. Each node in the cluster uses the Standard Large Instance type.
    b. Each node in the cluster uses the Standard Extra-Large Instance type.
    c. Each node in the cluster uses the High-CPU Medium Instance type.
2) Run the wordcount job by repearing the steps 2-5 in Part 5 above.
3) Record the execution time of each wordcount run under each cluster type you have provisioned in Step 1.
4) Calculate the cost of running the wordcount job under each cluster type you have provisioned in Step 1 using the Amazon instance pricing chart (http://aws.amazon.com/ec2/pricing/).
5) Plot the collected numbers in Steps 3 and 4 using a Column chart in Excel.
6) Analyze and discuss the wordcount performance under different cluster types.

Hint: You will need to modify the `hadoop-ec2.properties` file to instruct Whirr to provision different instance types. More information on the different instance types in Amazon EC2 can be found at http://aws.amazon.com/ec2/instance-types/

Deliverable 8: An excel file named **Part8a.xlsx** that contains a table with the execution time and pricing comparison of the `wordcount` job under each cluster type and the column chart as described in Steps 3 and 4 above. Furthermore, include a word file named **Part8b.docx** that contains your analysis and discussion as described in Step 5.

Note: Please terminate your instances once you have completed this part – Do not leave your instances running idle for more than a few hours.

PART 9: DEVELOPING A MAPREDUCE PROGRAM TO SOLVE A STRING MATCHING PROBLEM

In this part of the project, you will develop a MapReduce program to solve a simple string matching problem and deploy it on Amazon EC2. Given a ***text*** string (e.g., a document) and a ***pattern*** string (e.g., a word supplied by a user), the string matching problem is to find all the occurrences of the pattern in the text. Efficient algorithms for this problem are typically used in Bioinformatics to search for particular patterns in DNA sequences and in Internet search engines to find Web pages relevant to queries, among others. The string matching problem can be formalized as follows:

We assume that text, ***T***, is a file of ***N*** records, each referred to as ***R*** and has length ***n*** (a fixed length), and that pattern, ***P***, is a word of length ***m***, with $m \leq n$. We further assume that the elements of ***P*** and ***R*** are characters drawn from a finite alphabet set $\sum$ = {A, B, C}. We say that ***P*** occurs with shift ***s*** in record ***R*** if $0 \leq s \leq n - m$ and $R[s+1 \ldots s+m] = P[1 \ldots m]$ (see Figure 2). If ***P*** occurs with shift ***s*** in ***R***, then we call ***s*** a valid shift; otherwise, we call ***s*** an invalid shift. Our string matching problem will be the problem of finding all valid shifts with which a given pattern ***P*** occurs in each record ***R*** of text ***T***.

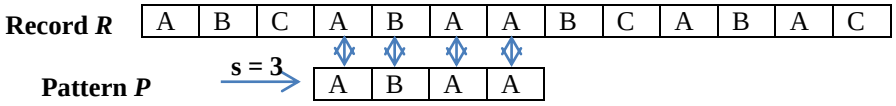

Figure 2: Pattern P occurs in R with shift s = 3

Please complete the following tasks:

1) Write your own data set generator that generates 3 files, file01, file02, and file03 of 1-million, 10-million, and 100-million 10-character records, respectively.
2) Write a MapReduce program that solves our string matching problem with P = {A, B, A, A}, s = 3, and input datasets file01, file02 and file03.
3) Package your MapReduce program and run it with the three different datasets on an 8-node cluster in Amazon EC2 using Whirr.
4) Plot the execution times of all your runs on an Excel sheet.
5) Analyze and discuss your results.
6) Discuss your experience in applying MapReduce to a string matching problem.

Deliverable 9: An excel file named **Part9a.xlsx** that contains a table and a plot with the execution times of all your string matching runs. Besides, include a word file named **Part9b.docx** that contains your analysis and discussion as required in Steps 5 and 6.

Note: Please terminate your instances once you have completed this part – Do not leave your instances running idle for more than a few hours.