HW#3 - Tutorial: Running MapReduce on Amazon Elastic Mapreduce (EMR)

18-645: How to write fast code

Due: March 10, 2014 (Monday), 11:00pm EDT, 8:00pm PDT

This tutorial will go through the following tasks:

- Task 0: Create Amazon AWS account
- Task 1: Installing Hadoop on local machine
- Task 2: Compiling and running Project 3 on local machine
- Task 3: Running Project 3 on Amazon EMR cluster

Submission:

To submit your homework, you must carefully follow these instructions:

1. Use the text-based answer template attached for filling in your answers.

2. To get credit for the homework, please submit your answer sheet to the Acatar system.

with the answer sheet : "**hw3_<yourFirstName>_<yourLastName>.txt**"

As you work through the homework and the project, you will realize that the homework is designed to help you get started on the project.

Task 0: Create Amazon AWS account

In this task you'll be setting up Amazon Web Service (AWS) account, which is required to run Amazon Elastic MapReduce and store data in Amazon S3.

1. One team member: Create an AWS (Amazon Web Services) account for your team (Credit card required) – no charge incurred.

2. E-mail 18645@sv.cmu.edu requesting a $100 AWS credit

> From: your e-mail address
> To: 18645@sv.cmu.edu
> Subject: "Credit Code Request - TeamXX (your name)"
> Body: ... empty ...

After receiving this email, we'll send out the code for your team to redeem the $100 credit.

3. Visit http://aws.amazon.com/awscredits to redeem credit

Only one $100 credit will be sent to each team. 10 extra credits for the team member that performs this task. You will need to put a credit card on file for AWS.

NOTE: Please remember to stop or terminate your instances after a session – your credit card will be charged if you don't terminate the process.

We only have limited credits for the class. $100 should be enough for this project. For more credits, you will have to show us that you have used your credits wisely – i.e. show us the optimizations you were already able to achieve with $100 of credit.

NOTE: For Task 1, Task2, you can use your local machine instead of ghc servers. Either Ubuntu or mac will work smoothly with the instruction here.

Task 1: Installing Hadoop on local machine

Step 1: Install prerequisites

To be able to run Hadoop on local machine we need the following:
1. **Java 1.7** – Hadoop MapReduce runs on Java Virtual Machine
2. **Ant** – Project management tool. Used to build the Hadoop project.

Here we give instructions on installing the above assuming you're using Ubuntu Linux. For other platforms, you can go to the their home pages (http://java.com, http://ant.apache.org) for installation instructions.

Note that we need to install JDK (the Java development kit) instead of JRE (Java Runtime Environment), because we will need to compile a jar file that's only available with JDK.

```
# Install JDK 1.7
$ sudo apt-get install openjdk-7-jdk

# Install ant
$ sudo apt-get install ant
```

After installation, verify they're installed by typing the following in your command line:

```
$ java -version
$ javac -version
$ ant -version
```

If all of the above commands are found and java / javac shows a version number of 1.7.xxx, you're ready to proceed to next step. Ant may have a version number of 1.8.x.

Step 2: Install Hadoop

Follow the commands below to install Hadoop.

```
# Download latest stable hadoop distribution.
# Here we are installing it to your home directory, but you can install it anywhere you like.
$ cd ~/
$ wget http://apache.claz.org/hadoop/common/hadoop-2.3.0/hadoop-2.3.0.tar.gz
$ tar xzvf hadoop-2.3.0.tar.gz

#Add hadoop executables into PATH (bash)
$ export PATH=$PATH:$HOME/hadoop-2.3.0/bin

# Add hadoop executables into PATH (csh)
$ setenv PATH "${PATH}:${HOME}/hadoop-2.3.0/bin"
```

Add the PATHs to your shell login file to avoid typing it every time.

Type the following to verify it's successfully installed:

```
$ hadoop
```

If it shows help information of the command, you're ready to proceed.
If you're getting error, it may be possible that the Hadoop command requires JAVA_HOME environment variable to be set. Usually the Java JDK is installed under /usr/lib/jvm/, depending on the version that's installed, you may need to do the following:

```
# Set JAVA_HOME environment variable
$ export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

Task 2: Compiling and running Project 3 on local machine

Step 1: Download Project 3 package in team repo and download data

```
$ cd ~/645/fastcode
$ wget http://fast645.info/course/data/Proj3.tar.gz
$ tar –zxvf Proj3.tar.gz
$ git add Proj3
# The project 3 package will be included to your repo.
$ cd Proj3
$ mkdir –p data/tweets10m
$ cd data/tweets10m
$ wget http://fast645.info:/course/data/tweets10m.txt.bz2 # Note: It's huge (300MB)
$ bunzip2 tweets10m.txt.bz2
# Split out a small trunk of data
$ mkdir ../tweets1m
$ head  -1000000  tweets10m.txt > ../tweets1m/tweets1m.txt
```

Now we'll use **$PROJ3_ROOT** to denote the location of project 3 files. For example, $PROJ3_ROOT/data/tweets10m is where we just downloaded the data.

We have two different sizes of data: tweets10m and tweets1m. "tweets1m" is simply a dataset with first 1 million rows from tweets10m.

Step 2: Check $PROJ3_ROOT/lib/ directory

```
$ cd $PROJ3_ROOT
$ ls –l lib/
-rwxr-xr-x 1 saintkyumi users   41123 Dec  3  2012 commons-cli-1.2.jar
-rw-r--r-- 1 saintkyumi users 3928345 Dec  3  2012 hadoop-core-1.0.3.jar
```

Ant build system can recognize the location of Hadoop jar, and integrate into our project jar file that will be used to run Hadoop locally and on EMR cluster.

Step 3: Build the project with ant

```
$ cd $PROJ3_ROOT
$ ant
```

On a successful build, it will show "BUILD SUCCESSFUL". Under $PROJ3_ROOT there will be a new file, **18645-proj3-0.1-latest.jar.**

Step 4: Run Project 3

There are two programs in proj3 package:
1. **NgramCount**: Counts the number of occurrences of ngrams in a corpus.
2. **HashtagSim**: Computes similarities between hashtags in a twitter corpus.

You'll be running both programs on your local machine. (There will be tasks to run them on EMR in later section.)

Run ngramcount (assuming you're currently in $PROJ3_ROOT)

```
$ hadoop jar  18645-proj3-0.1-latest.jar –program ngramcount -input data/tweets10m/tweets10m.txt -output
data/ngram10m
```

Run hashtagsim

```
$ hadoop jar  18645-proj3-0.1-latest.jar -program hashtagsim -input data/tweets1m/tweets1m.txt -output data/hashtag1m -
tmpdir tmp
```

In both programs, "-input" flag specifies where the input data is located, "-output" specifies where the final output will be stored, and "-tmpdir" specifies where the intermediate files of the map-reduce steps will be stored. In NgramCount, there's only 1 map-reduce step, so we don't need to specify a temporary directory.

Note that we're using difference input sizes for NgramCount and HashtagSim (10 million vs 1 million). This is because later in the project we'll be extending the HashtagSim program, that will be running much slower than NgramCount.

Proceed to next Task while the jobs are running. It may take hours for one job to finish.

Answer the following questions in answer sheet:

Question 1: How long does each map-reduce step take, for NgramCount and HashtagSim? (Hint: See Hadoop output)
Question 2: How many output files are there for NgramCount and HashtagSim? (Only count part-r-*)
Question 3: What is the N of the ngrams, for the current implementation?
Question 4: What are the top 5 most occurring ngrams? (Hint: sort the output in data/ngram10m/)
Question 5: What are the top 5 pairs of hashtags that have highest similarity score? (Hint: sort output in data/hashtag1m, remove pairs that are duplicated)

Task 3: Running Project 3 on Amazon EMR cluster

Now we turn the attention to the publicly available cloud computing platform – Amazon Elastic MapReduce (EMR). For full documentation, visit http://aws.amazon.com/elasticmapreduce/ .

Step 1: Install & configure command line tools

There are multiple ways of using Amazon EMR:
1. Web UI
2. Command line tool
3. Web service

Using Web UI is easiest if you're the first time to use EMR, but it can hardly be automated. Command line tools provide full operations available in Web UI while giving you power of running the program in command line. Web service is the most flexible (after all both the Web UI and command line tools are built on top of raw Web service), but requires too much expert knowledge and is tedious to program with. Here we'll guide you through steps of using EMR with command line tools.

Tools we need:

1. **Elastic MapReduce Ruby Client** (http://aws.amazon.com/developertools/2264): Command line tool to access Amazon EMR.
2. **S3 CLI** (http://timkay.com/aws): Command line tool to access Amazon S3 cloud storage platform. We'll be using S3 to upload the Hadoop jar file, data file and store output files.

First, visit the webpage http://docs.amazonwebservices.com/ElasticMapReduce/latest/GettingStartedGuide/SignUp.html for a full documentation of how to install **EMR Ruby client**. I find the tutorial pretty amazing, because it not only shows you how to install the ruby client, but also instructions on how to sign up for Amazon Web Services, and how to get security credentials.

After following instructions on this page, you should add the tool to your path to be able to call it from anywhere. Assuming you downloaded the tool to $HOME/elastic-mapreduce-cli, do the following:

```
$ export PATH=$PATH:$HOME/elastic-mapreduce-cli

# Verify you can access the tool
$  elastic-mapreduce
```

If the above command shows help information, you can proceed.

Note: The **"credentials.json"** file in $HOME/elastic-mapreduce-cli/ mentioned in the tutorial looks like the following after modification (replace entries with your own):

```
{
"access_id": "1B5JYHPQCXW13GWKHAG2",
"private_key": "2GAHKWG3+1wxcqyhpj5b1Ggqc0TIxj21DKkidjfz",
"keypair": "dpiao.18645",
"key-pair-file": "/home/dpiao/dpiao.18645.pem",
"log_uri": "s3n://dpiao.log/",
"region": "us-east-1"
}
```

[TIP0] Read elastic-mapreduce-cli/README
[TIP1] To obtain "access-id" and "private_key", you may go to https://console.aws.amazon.com/iam and explore "Users" menu.
[TIP2] To obtain "key-pair" and "key-pair-file", you may go to https://console.aws.amazon.com/ec2/home#c=EC2&s=KeyPairs

Next we need to install **S3 CLI**. Visit the page http://timkay.com/aws, follow instruction in "Download" section to install the tool. Verify the success of installation by typing the following:

```
$ s3put
$ s3mkdir
```

Step 2: Spin up an EMR cluster

We'll be using a small cluster (5 machines) for the purpose of the homework. Run the following command to start the EMR cluster:

```
$ elastic-mapreduce --create --alive --num-instances 5 --instance-type  c1.medium
```

This will create a EMR cluster with 1 master node and 4 slave nodes, all of which are c1.medium instances. For details of configurations of various EMR instances, visit http://aws.amazon.com/ec2/instance-types/. For pricing of the instances, visit http://aws.amazon.com/elasticmapreduce/pricing/. To see what the code numbers for various instances are, visit http://docs.amazonwebservices.com/ElasticMapReduce/latest/DeveloperGuide/Intro_AWSConcepts.html. For the evaluation of Project 3, we'll be using the exact configuration as described above.

The above command will return a jobflow id, something similar to "j-2PS7D4FCM65S3". We'll denote this id with **$JID**. We'll be using $JID later when running a job. The cluster takes several minutes to start, you can see the status of the cluster either by using the "elastic-mapreduce" command line tool (visit http://docs.amazonwebservices.com/ElasticMapReduce/latest/GettingStartedGuide/Essentials.html for details), or go to the WebUI (https://console.aws.amazon.com/elasticmapreduce/).

Step 3: Upload compiled jar file and data file to Amazon S3

With all the preparation work, now we can enjoy the fun part - asking computers to run jobs for you. There are several ways to run Hadoop map-reduce on EMR, such as using Streaming, Pig, Cascading (full documentation is here: http://docs.amazonwebservices.com/ElasticMapReduce/latest/GettingStartedGuide/WhereGoFromHere.html), among which using custom jar file is the most powerful and flexible way. It's also the most traditional way of using Hadoop.

We'll need to upload the project jar file to Amazon S3 for EMR to run. Run the following command to create a bucket to store your jar file and upload to the bucket:

```
$ s3mkdir dpiao.18645       # Change to your own bucket name!
$ cd $PROJ3_ROOT
$ s3put dpiao.18645 18645-proj3-0.1-latest.jar

# Now the jar file is available at s3n://dpiao.18645/18645-proj3-0.1-latest.jar

# Note: We're NOT doing this:
$ s3put dpiao.18645 $PROJ_ROOT/18645-proj3-0.1-latest.jar    # WRONG
# Because it will upload the file to
# s3n://dpiao.18645/$PROJ_ROOT/18645-proj3-0.1-latest.jar

# Verify the jar file is successfully uploaded:
$ s3ls dpiao.18645
```

You can also visit the Amazon S3 WebUI to "visually" verify the upload: https://console.aws.amazon.com/s3/

Next we upload the data file.

```
$ s3mkdir dpiao.tweets10m       # Change to your own bucket names!
$ s3mkdir dpiao.tweets1m

$ cd $PROJ3_ROOT/data/tweets10m
$ s3put dpiao.tweets10m tweets10m.txt

$ cd $PROJ3_ROOT/data/tweets1m
$ s3put dpiao.tweets1m tweets1m.txt

# Verify the jar file is successfully uploaded:
$ s3ls dpiao.tweets10m
$ s3ls dpiao.tweets1m
```

Finally, we need to create a bucket to store the output file.

```
$ s3mkdir dpiao.output     # Change to your own bucket name! (NOTE: bucket names must be globally unique)
```

Step 4: Run NgramCount on EMR

Do the following to send NgramCount job to EMR:

```
# Run NgramCount on EMR
$ elastic-mapreduce --jobflow $JID --jar s3n://dpiao.18645/18645-proj3-0.1-latest.jar --arg -program --arg ngramcount --arg -input
--arg s3n://dpiao.tweets10m/tweets10m.txt --arg -output --arg s3n://dpiao.output/ngram10m
```

This tells the EMR to run the jar saved in bucket "dpiao.18645/", with the input from s3 bucket "dpiao.tweets10m", and outputs to bucket "dpiao.output/ngram10m".

Step 5: See log output and result file

The result file will be uploaded to your designated bucket (dpiao.output/ngram10m) upon the finish of the jobs. You can see the status of the job running by using "elastic-mapreduce" command line tool (visit http://docs.amazonwebservices.com/ElasticMapReduce/latest/GettingStartedGuide/Essentials.html for details), or go to the WebUI (https://console.aws.amazon.com/elasticmapreduce/).

The runtime of the NgramCount can be found in the S3 bucket specified in **credentials.json** file, under entry "**log_uri**". In the case of this tutorial, it's "s3n://dpiao.log/". Click on the jobflow that you started, then click "steps", and choose the latest step. The standard output from the program, including runtime for each step and the **total runtime**, is stored in file "**stdout**". If there's any error, such as the job failed, the error messages are in file "**stderr**".

Note: The log will be uploaded **5 minutes** after the job is finished.

Troubleshooting: If there's any abnormal activities, such as the job failing, check the following webpage to troubleshoot the problem:
http://docs.amazonwebservices.com/ElasticMapReduce/latest/DeveloperGuide/DebuggingJobFlows.html.

Step 6: Run HashtagSim on EMR

```
# Run HashtagSim on EMR
$ elastic-mapreduce --jobflow $JID --jar s3n://dpiao.18645/18645-proj3-0.1-latest.jar --arg -program --arg hashtagsim --arg -input
--arg s3n://dpiao.tweets1m/tweets1m.txt --arg -output --arg s3n://dpiao.output/hashtag1m --arg -tmpdir --arg tmp
```

Similar to NgramCount, retrieve the result from bucket “dpiao.output/hashtag1m”, and log file (runtime) from bucket “dpiao.log”.

Step 7: Terminate the EMR cluster

Once all jobs are finished, terminate the cluster. **Your credit card could be charge hundreds of dollars if you leave it running for days and run out of the $100 credits**.

```
$ elastic-mapreduce --terminate $JID
```

Answer the following questions in answer sheet:

Question 1: How long does each map-reduce step take, for NgramCount and HashtagSim? (Hint: See log file in S3)
Question 2: How many output files are there for NgramCount and HashtagSim? (Only count part-r-*) Is it the same as when running both programs locally? If not, what’s the reason?
Question 3: What are the top 5 most occurring ngrams? (Hint: retrieve all output files and sort)
Question 4: What are the top 5 pairs of hashtags that have highest similarity score? (Hint: retrieve all output files and sort, remove pairs that are duplicated)