18-645: How to Write Fast Code

HW#3 - Tutorial

Running MapReduce on Amazon Elastic Mapreduce (EMR)

Due: April 14th, 2017 (Friday), 11:59pm EST

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 Canvas.

with the answer sheet: "**18645_HW3_TeamXXX_yourAndrewID.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 (or use your existing AWS account)
2. E-mail 18645@sv.cmu.edu using your CMU email account to request AWS credit

From: your **CMU** e-mail address To: 18645@sv.cmu.edu Subject: "Credit Code Request - TeamXX (your name & Andrew ID)"

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

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

Only one $50 credit will be sent to each team member. You might need to put a credit card on file for AWS.

NOTE: Please remember to stop or terminate your instances after you finish your work – your credit card will be charged once your AWS coupons run out if you don't terminate the cluster!

We only have limited credits for the class. $50 per student 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 have achieved with $50 of credit.

NOTE: For Task 1, Task2, you can use your local machine instead of ghc servers. Either Ubuntu or mac should work well with the instructions here. Using your local machine is recommended due to the disk quota restriction on GHC machines.

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 (Please follow the version requirement)
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 Java and Ant's 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 with JDK. On Ubuntu, all JDK installed can be found under /usr/lib/jvm/ by default. If you are using GHC machines, they may already have the JDK installed. Check it out at /usr/lib/jvm/.

Notice: You might see some strange characters if you copy the commands directly from this handout and paste it to the terminal. We suggest you to type your own commands, which will greatly facilitate your understanding.

```
# Install JDK 1.7 (You may see UnsupportedClassVersionError later on EC2 if you compile your code locally using higher version of JDK)

$ sudo apt-get install openjdk-7-jdk

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

After installation, verify they are installed properly 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. Set your JAVA_HOME properly to if your installed openjdk-7-jdk is not being applied.

Ant may have a version number of 1.7.x and above.

If you have troubles with installing multiple versions of jdks on your Mac OS X, this is a useful tool you can use: https://andrew-jones.com/blog/managing-multiple-versions-of-java-on-os-x/.

Step 2: Install Hadoop

Follow the commands below to install Hadoop:

```
# Download stable hadoop distribution.
# Here we are installing it to your home directory, but you can install it anywhere you like.
$ cd ~/
$ wget https://archive.apache.org/dist/hadoop/core/hadoop-2.4.0/hadoop-2.4.0.tar.gz
$ tar xzvf hadoop-2.4.0.tar.gz

#Add hadoop executables into PATH (if you are using bash)
$ export PATH=$PATH:$HOME/hadoop-2.4.0/bin

# Add hadoop executables into PATH (if you are using csh)
$ setenv PATH "${PATH}:${HOME}/hadoop-2.4.0/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. You may need to do the following (update the JAVA_HOME directory according to your environment setup):

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

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              # or any project root directory on your local machine
$ wget https://cmu.box.com/shared/static/2fwqjutnf8u299sbotkgqamoc7iweawx.gz -O Proj3.tar.gz
$ tar -zxvf Proj3.tar.gz

# As said, you can run task 1& 2 on any machine, but do commit a copy of your project in the end to the repo.
# Commit the Project only; Do NOT commit the "data" folder below in the repo – it is huge.
$ git add Proj3

$ cd Proj3                  # we refer to this directory as $PROJ3_ROOT
$ mkdir -p data/tweets10m
$ cd data/tweets10m
$ wget https://cmu.box.com/shared/static/701hqlebgkpek9n9qtcd2vqzt6k98x84.bz2 -O tweets10m.txt.bz2
# data size is around 300MB
$ bunzip2 tweets10m.txt.bz2
# Split out a small trunk of data
$ mkdir ../tweets1m
$ head  -1000000  tweets10m.txt > ../tweets1m/tweets1m.txt
```

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

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 lib/
commons-cli-1.2.jar  hadoop-core-1.0.3.jar
```

Step 3: Build the project with ant

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.

```
$ 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 -n 1 -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 mini project 3 we will extend the HashtagSim program, and 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 AWS Client

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

Using Web UI is easiest if it is your first time to use EMR, but it is not the best option to automate job deployment. 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 additional knowledge and is tedious to program with. Here we'll guide you through steps of using EMR with command line tools.

- **AWS Command Line Interface** (http://aws.amazon.com/cli): The AWS Command Line Interface is a unified tool to manage AWS services. You can control multiple AWS services from the command line and automate them with scripts.

The AWS CLI documentation provides comprehensive information on how to use this command line tool. For concepts of AWS CLI, take a look at the document here: http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html. For CLI syntax and options, this page (http://docs.aws.amazon.com/cli/latest/reference/) provides you easy reference. Below instructions should be sufficient for you to complete this homework. Review the above CLI documentations if you want to explore the tool further. Let's start.

A. **Install the AWS CLI Using Pip**
pip is a Python-based tool that offers convenient ways to manage Python packages and their dependencies. Pip is the recommended method of installing the CLI on Mac and Linux.

```
$ pip install awscli
```

Detailed instructions can be found in:
http://docs.aws.amazon.com/cli/latest/userguide/installing.html#install-with-pip

B. **Configuring the AWS Command Line Interface**
aws configure command is the fastest way to setup your AWS CLI installation. You will see something like below when typing the *aws configure* command. Before that you will need to create an IAM user and assign permissions to the user by attaching IAM policies. **Create an IAM user for each team member if you want to develop or test independently.** Want to log

in to AWS web console with IAM user? See the below instructions:
http://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_how-users-sign-in.html

```
$ aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE          ## Replace it with yours
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
        ## Replace it with yours
Default region name [None]: us-west-2
        ## This is usually the region closest to you, but it can be any region
Default output format [None]: json
```

First let's create an IAM user and get the access key ID and secret access key. Follow the "To get your access key ID and secret access key" section here: http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-set-up.html#cli-signup. Record your AWS access key ID and AWS secret access key.

Next, we need to add permissions to the IAM user, otherwise the user won't be able to launch EMR jobs. To assign permissions to a user, group, role, or resource, you create a policy, which is a document that explicitly lists permissions. In the same IAM home page (https://console.aws.amazon.com/iam/home#home), click on "Policies", and look for policy "AmazonElasticMapReduceFullAccess". Click on it, and attach it to the IAM user you just created.

Now we are ready to configure AWS CLI,

```
$ aws configure
```

Detailed instructions here:
http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html#cli-quick-configuration

Step 2: Upload compiled jar file and data file to Amazon S3

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 S3 bucket:

```
$ aws s3 mb s3://bliu.fastcode          # Change to your own bucket name!
$ cd $PROJ3_ROOT
$ aws s3 cp 18645-proj3-0.1-latest.jar s3://bliu.fastcode

# Now the jar file is available at s3://bliu.fastcode/18645-proj3-0.1-latest.jar
# Verify the jar file is successfully uploaded
$ aws s3 ls s3://bliu.fastcode
```

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.

```
$ aws s3 mb s3://bliu.tweets10m      # Again, change to your own bucket names!
$ aws s3 mb s3://bliu.tweets1m

$ cd $PROJ3_ROOT/data/tweets10m
$ aws s3 cp tweets10m.txt s3://bliu.tweets10m

$ cd $PROJ3_ROOT/data/tweets1m
$ aws s3 cp tweets1m.txt s3://bliu.tweets1m

# Verify the data files are successfully uploaded:
$ aws s3 ls s3://bliu.tweets10m
$ aws s3 ls s3://bliu.tweets1m
```

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

```
$ aws s3 mb s3://bliu.output  # bucket names must be globally unique

# create bucket for job logs
$ aws s3 mb s3://bliu.log-uri.ngramcount
$ aws s3 mb s3://bliu.log-uri.hashtagsim
```

Step 3: Launch the cluster and run the job

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.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-get-started.html), among which using custom jar file is the one of the most powerful and flexible ways. It's also the traditional way of using Hadoop.

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

This will create an 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 in the example below.

Let's run the NgramCount program on EMR cluster:

```
aws emr create-cluster --name "Test cluster ngramcount" --ami-version 2.4.11 \
--service-role EMR_DefaultRole --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
--log-uri s3://bliu.log-uri.ngramcount --enable-debugging \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=c1.medium
InstanceGroupType=CORE,InstanceCount=4,InstanceType=c1.medium \
--steps Type=CUSTOM_JAR,Jar=s3://bliu.fastcode/18645-proj3-0.1-latest.jar,Args=["-
input","s3://bliu.tweets10m/tweets10m.txt","-output","s3://bliu.output/ngram10m","-
program","ngramcount"] \
--auto-terminate
```

Remember to set the s3 address to your own s3 bucket location before you press enter.

The above command will return a ClusterId, something similar to "j-2PS7D4FCM65S3". We will refer to this as **$CID**. The cluster takes several minutes to start, you can see the status of the cluster either by using the command line tool or go to the WebUI. In web UI, make sure you are looking at the same AWS region as you configured in AWS CLI. https://console.aws.amazon.com/elasticmapreduce/

```
# Check cluster status from command line
$ aws emr describe-cluster --cluster-id $CID
```

In above command, "aws emr create-cluster" tells AWS to create and start running an EMR cluster. "--steps" indicates a list of steps to be executed by the cluster. "--auto-terminate" tells AWS to terminate the cluster once all steps are completed. This option, which is off by default, helps to prevent your cluster from running idly when you forget to turn it off.

Note that the two actions above (creating EMR cluster & adding steps) can be separated, meaning you can start the cluster first without any job (or any step), and add new steps to it as long as the cluster is not in terminated status. This helps to avoid bringing up new cluster (which takes a few minutes' time) every time when you want to test a job. So feel free to run the jobs in your preferred manner! While, as always, we would recommend you to do development and testing on local environment first so as to control your AWS budget. For more details, check it here:
http://docs.aws.amazon.com/cli/latest/reference/emr/index.html

Similarly, run the HashtagSim job:

```
aws emr create-cluster --name "Test cluster hashtagsim" --ami-version 2.4.11 \
--service-role EMR_DefaultRole --ec2-attributes InstanceProfile=EMR_EC2_DefaultRole \
```

```
--log-uri s3://bliu.log-uri.hashtagsim --enable-debugging \
--instance-groups InstanceGroupType=MASTER,InstanceCount=1,InstanceType=c1.medium
InstanceGroupType=CORE,InstanceCount=4,InstanceType=c1.medium \
--steps Type=CUSTOM_JAR,Jar=s3://bliu.fastcode/18645-proj3-0.1-latest.jar,Args=["-
input","s3://bliu.tweets1m/tweets1m.txt","-output","s3://bliu.output/hashtag1m","-
program","hashtagsim","-tmpdir","tmp"] \
--auto-terminate
```

Step 4: See log output and result file

The result file will be uploaded to your defined S3 bucket (e.g. s3://bliu.log-uri.ngramcount) upon completing of the jobs. You can see the status of the job running by AWS CLI command line tool or go to the WebUI (https://console.aws.amazon.com/elasticmapreduce/).

Note: The log will show up there around **5 minutes** after the job is finished.

Troubleshooting: If you saw any abnormal activity, such as the job failing, check the following web page on troubleshooting guidance:
http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-troubleshoot.html

Step 5: Terminate the EMR cluster

In our above examples we enabled the "--auto-terminate" option when creating the EMR cluster. Thus you do not need to terminate the cluster explicitly. If this option is not enabled, please do remember to bring the cluster down once all your jobs (steps) are completed. You can either terminate the cluster via AWS CLI or the web UI.
Your credit card could be charge hundreds of dollars if you leave it running for days after using up all AWS credits.

```
$ aws emr terminate-clusters --cluster-ids $CID
```

Hooray, we are done! By now you should have a basic understanding of AWS EMR and the AWS CLI command line tool. Remember that command line tool is not the only way connecting us to EMR. Try to explore the EMR Web UI to see how you can map those command line options to the UI options. EMR Web UI provides an intuitive way for people to interact with the service (especially when one is new to EMR), but command line interfaces we introduced will make the job scheduling and automation a lot easier when running large complex jobs. You will like it more when you work on larger and more complex tasks one day.

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)