Architecture

Overview

This project consists of three parts:

- Splunk installation
- Splunk S3 integration
- Spark classifier in Splunk machine learning app

This project integrates Spark's ML Logistic Regression functionality into the Splunk Machine Learning Toolkit and Showcase App. Because Spark logistic regression runs up to 100x times faster than MapReduce logistic regression and Splunk runs on MapReduce, replacing Splunk's MapReduce backend with a Spark backend could dramatically increase Splunk's performance.

Installation

Splunk Installation

In order to deploy this application, you will need to set up both Splunk and Spark on a single machine. The following is how I did it:

1. Provision a new instance using "ucbw205_complete_plus_postgres (AMI ID: ami-bff58fda)." Please make it m2.medium or larger.
2. Open up ports 8000, 8080, 8089, and 9997 on your UCB instance's security group.
3. Note the Availability Zone of your UCB instance. We need to make sure Splunk is in the same region.

As Splunk is commercial software, you will need a Splunk license to use it. Fortunately, you can sign up for a pay-as-you-go license by provisioning your own Splunk instance.

4. On Amazon Web Services, provision a new instance using the "Splunk Enterprise (HVM)" AMI. Since we won't actually run Splunk from this instance for very long, you can use a t1.micro instance. Be sure to provision in the same region as your "Easy Button" instance.
5. At Step 4 in the Launch Instance Wizard, please add a 50 GB or more EBS volume to the instance.
6. Launch and start your instance.

The default Splunk password is the Instance ID of your new Splunk Enterprise (HVM) instance. To avoid confusion, let's change this password to something easier to remember before we copy the Splunk installation to another instance.

7. Once your instance is running, use a web browser to go https://<your instance address here>:8000. You will see the Splunk login screen
8. Login using the default username and password. At the time of writing, the default username is "admin" and the default password is the Instance ID of your newly provisioned Splunk AMI.
9. After you login, Splunk will ask you to make a new password. Be sure to remember your new password.

10. After you change your password and reach the Splunk home screen, you may close the browser window. There's no need to follow any of the tutorials or on-screen set up instructions.

Next, let's copy the Splunk installation to your EBS volume. In general, this is a good way to give Splunk plenty of storage space without paying for an expensive instance (such as c3.large).

11. Now SSH into your Splunk instance. Note that you will need to login as "ec2-user"
12. We will need read and write access to the root directory. You will need to switch to the root user

    sudo su - root

13. As the root user, shutdown Splunk.

    /opt/splunk/bin/splunk stop

14. Mount your EBS volume onto your machine:

    mkdir /splunkdata
    mkfs -t ext4 /dev/xvdb
    mount -t ext4 /dev/xvdb /splunkdata

15. Copy or move the Splunk installation folder

    mv /opt/splunk /splunkdata

16. Unmount your EBS volume

    umount /splunkdata

17. Stop, **but don't terminate**, your Splunk instance. Although you will no longer be using it, your Splunk instance is tried to your Splunk license. Please keep you Splunk instance around until you are ready to stop using the Splunk software application.

Now let's move Splunk over to your UCB instance.

18. Detach your Splunk EBS volume from your Splunk instance
19. Attach your Splunk EBS volume to your UCB instance
20. Start your UCB instance
21. SSH into your UCB instance
22. Mount your Splunk EBS volume

    mkdir /splunkdata
    mount -t ext4 /dev/xvdf /splunkdata

23. Create a Splunk user account and give it access to your Splunk EBS volume

    useradd splunk -p splunk
    chown -R splunk /splunkdata

24. To give the Splunk user access to the necessary Spark modules, we need to copy the some files:

    cp -P /home/w205/spark15 /home/splunk
    cp /usr/lib64/python2.6/lib-dynload/_lsprof.so /splunkdata/splunk/lib/python2.7/lib-dynload

25. Splunk is now installed on your UCB instance. To start the Splunk program, run the following command.

sudo –u splunk /splunkdata/splunk/bin/splunk restart

26. To stop your Splunk instance, run the following command:

    sudo –u splunk /splunkdata/splunk/bin/splunk stop

27. Remember to unmount your Splunk EBS volume before shutting your machine down

    umount /splunkdata

App Installation

Now that Splunk is running on your UCB instance, we now need to install the Apps that this project will use.

1. If you don't have one already, go to www.splunk.com and sign up for a free account.
2. If you haven't already, start your UCB instance, mount your Splunk EBS, and start Splunk:

   mount -t ext4 /dev/xvdf /splunkdata
   sudo –u splunk /splunkdata/splunk/bin/splunk restart

3. In a web browser, go to https://<your instance address here>:8000
4. Login in with the username "admin" and the password that you set previously
5. You should now be at your Splunk installation's home page. You can ignore any on-screen tutorials or setup instructions.
6. To install new apps, click on the white dotted plus sign on the left sidebar on the Splunk homepage. It should be right underneath the green box that says "Search & Reporting"
7. Install the following two free apps. Note that you will need to login with your Splunk.com account
   a. Splunk Add-on for Amazon Web Services <Version 2.0.1>
   b. Python for Scientific Computing (for Linux 64-bit) <Version 1.0.0>
8. In the upper left hand corner of the Splunk page, click on Apps -> Manage Apps.
9. Click the button that says "Install app from file." Browse to the file Jackson_Splunk_ML_Toolkit.spl located in this GitHub repo.
10. Once all three apps have finished installing, please restart Splunk either from the Web GUI or from the command line.

    sudo –u splunk /splunkdata/splunk/bin/splunk restart

11. Reload your browser page and login again
12. Go to your Splunk homepage. There should now be three apps on the left sidebar.
    a. Search & Reporting
    b. Jackson Splunk Machine Learning Toolkit
    c. Splunk Add-on for AWS

Usage

The first step is to get your data into your Splunk installation. For this project, I've put some server logs from one of my company's development domains onto an S3.

1. From the homepage, click on the box for the Splunk Add-on for AWS.

2. Click on the link that says “Create Account”
3. Click on the button that says “Create New Account”
4. Give your account a name. To access the server logs, put in the following
    a. Access Key: AKIAJFDYESBPEASRPMEQ
    b. Secret Access Key: qOt7S1RXkUsrikHat+jbhtFiX2D64MCZeP14+M39
5. Beneath the upper left corner of the page, click on the “Inputs” tab.
6. Click “Create New Input” -> “S3”
7. Select the newly created account and then the S3 bucket “ucb-w205-jackson-lane-data”
8. Click “Update”

Events will begin coming into Splunk. You will get a Daily Usage Limit warning, but don’t worry. The company won’t charge you for going over your license’s quota just once. Let’s now do a search of the incoming data.

9. Go to Apps -> Search & Reporting
10. Click on Data Summary. Under the host’s tab, you should see an IP address. (This is okay. All the logs are coming from a single host). Click on that IP address to start running a search.

Notice how the most recent events look broken and cut off at odd places. Splunk automatic event parser is generally not as robust when processing more than 500MB of event data.

To investigate this further, let’s go use the Machine Learning App to build a classifier

11. Go to Apps -> Jackson ML Toolkit and Showcase
12. Click on the “Showcase” tab -> “Predict Binary Fields With Spark”
13. After the screen loads, put in the following search query

    *host = * | head 50000| eval brokenevent = if(timestartpos >= 0 AND random() % 3 > 0, "false","true")*

    a. Ideally, we’d used manually compiled data where we ask senior level programmers to classify each event in a random sample as correctly or incorrectly parsed by Splunk. But for now, this search query should simulate that process.
    b. If you have less than 50000 events, please use the replay function in the ML app to generate more events. It’s used as follows

    |replay query=’search host = | head 5000’ repeats=10 maxwait=0 | | eval brokenevent = if(timestartpos >= 0 OR random() % 8 == 0, ‘False’,’True’)*

    However, it is not recommended to run this query if you have more than 50000 events, as doing so might cause a Splunk buffer overflow and halt any further searches until the server is restarted.
14. In “Step 2: Field to predict”, select the field “brokenevent”
15. In “Step 3: Fields to use for predicting”, select the following fields
    a. source
    b. timestartpos
    c. timeendpos
    d. <Optional: One or two other fields of your choice>
16. Click the “Fit Model” button. This search will take about 5 – 10 minutes.
17. The accuracy of the regression will be shown at the bottom of the page.

18. Now go to “Showcase” -> “Predict Categorical Fields.” This is the normal implementation of logistic regression in Spark.
19. Repeat steps 13 – 16 here. Is it faster?

Source Code

The custom code that I wrote for this project is in Jackson_Splunk_ML_Toolkit app folder. Go to “/splunkdata/splunk/etc/apps/Jackson_Splunk_ML_Toolkit” after installing the app on your Splunk installation.

The majority of new code written is in bin/replay.py and bin/algos/LogisticRegressionWithSpark.py