DSC 102: Systems for Scalable Analytics

Programming Assignment 1

1 Introduction

In this assignment, you will be using Dask library to explore task parallelism on multiple cores across different machines. You will be performing feature explorations, data consistency checks, and computing several descriptive statistics about the data to build intuitions for feature engineering for the next assignment.

2 Dataset Description

You are provided with the Amazon Reviews dataset with the *reviews* and *products* tables as CSV files. The schemas are provided in Table 1. The goal for the final assignment is to predict user's star rating. Thus, "overall" is our label.

(A) Column name	**Column description**	**Example**
reviewerID	ID of the reviewer	A32DT10X9WS4D0
asin	ID of the product	B003VX9DJM
reviewerName	name of the reviewer	Slade
helpful	helpfulness rating of the review	[0, 0]
reviewText	text of the review	this was a gift for my friend who loves touch lamps.
overall	rating of the product	1
summary	summary of the review	broken piece
unixReviewTime	summary of the review	1397174400
reviewTime	time of the review (raw)	04 11, 2014

(B) Column name	**Column description**	**Example**
asin	ID of the product	143561
salesRank	sales rank information	{'Movies & TV': 376041}
imUrl	url of the product image	http://g-ecx.images-amazon.com /31mC.jpg
categories	list of categories the product	[['Movies & TV', 'Movies']]
title	name of the product	Everyday Italian (with Giada de Laurentiis)
description	description of the product	3Pack DVD set - Italian Classics
price	price in US dollars	12.99
related	related products (also bought, also viewed, bought together, buy after viewing)	{'also_viewed': ['B0036FO6SI', '000014357X'],'buy_after_viewing ': ['B0036FO6SI', 'B000KL8ODE']}
brand	brand name	1

Table 1: (A) *Reviews* table and (B) *Products* table

3 Tasks

You will compute several descriptive statistics for both *reviews* and *products* table as follows:

Q1. Get percentage of missing values for all columns in the *reviews* table and the *products* table

Q2. Find pearson correlation value between the price and ratings

Q3. Get mean, standard deviation, median, min, and max for the price column in the *products* table

Q4. Get number of products for each super-category (the first entry in the "categories" column in the products table).

Q5. Check (Return 1 or 0) if there are any dangling references to the product ids from the *reviews* table to *products* table. Return 1 if there are dangling references.

Q6. Check (1 or 0) if there is any dangling reference between product ids in the related column and "asin" of the *product* table. Return 1 if there are dangling references.

A code stub with function signature for this task has been provided to you. The input to the function is the *reviews* CSV file and the *products* CSV file. The output is a json file (saved as `results_PA1.json`). The schema has been shared with you (`OutputSchema_PA1.json`). You will write your answers to each sub-task as values. **Do not modify any keys of the json file**. We will time the execution of the function `PA1`.

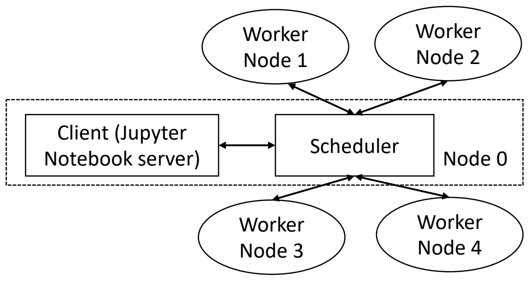
We have shared with you the "development" dataset and our accuracy results. Our code's runtime on 4 nodes and 1 node are roughly 170s and 720s respectively. You can use this to validate your results and debug your code. The final evaluation will happen on separate held-out test sets. The runtime and the speedup numbers will be different for the held-out test set.

4 Deliverables

Submit your source code as `<YOUR-TEAM-ID>.py` on Canvas. Your source code must confirm to the function signatures provided to you. Make sure that your code is writing results to `results_PA1.json`.

5 Setup

1. You will be launching 5 EC2 *Spot* instances, where you will create one instance for setting up the client (jupyter notebook server) and scheduler, and four instances for running workers.



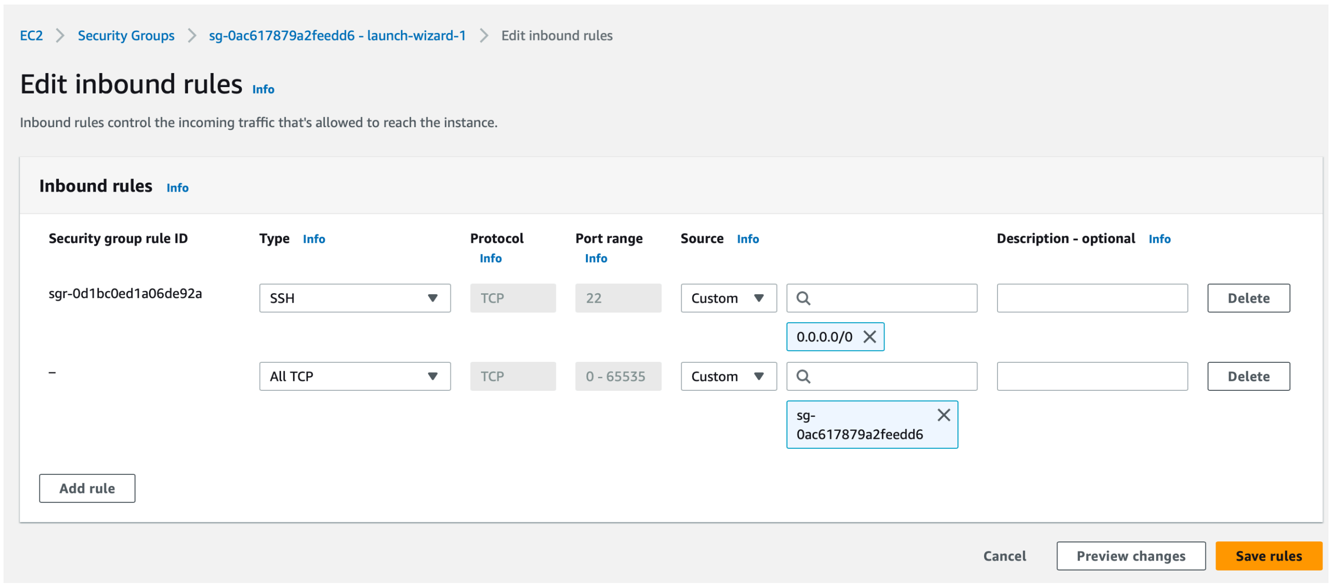
2. Follow the getting started guide given in PA0 for launching EC2 instances. When you are launching, as before, in the Network Settings tab leave "allow SSH traffic from" checked and set to 'Anywhere 0.0.0.0/0'. This will allow you to ssh into each of the machines. In the top right of the "Launch an instance" page, you can select your desired number of instances. Note that all fo these instances will share the same name but have distinct instance IDs. You will have to decide which one is the master/head node and which ones are worker nodes. Keep track of which instance ID corresponds to which.

3. Once the EC2 instances are launched, go to the security group of the instances (under "Security" in the bottom panel) and click the link for your security group. It will start with 'sg.' Click "edit inbound rules" then "add rule." Add one rule (under "Inbound Rules") as follows: A rule with type "All TCP", source as "Custom" with the security group id listed in the search pane (it will look something like "sg-________". This rule will allow workers and scheduler to communicate with each other. See Figure 1.

4. In this step, you will start the jupyter notebook server on the client, 1 scheduler process, and 4 worker processes.

a. Change permission of the ssh keyfile to make sure your private key file isn't publicly viewable: `chmod 400 <keyfilename>.pem`. Linux and Mac users in particular will need the chmod.

b. SSH into one of the nodes using command: `ssh -i ‘‘dask-key.pem" ubuntu@<ip-address-of-EC2-instance>`. Activate the dask environment with command: `source dask_env/bin/activate`. Start jupyter notebook server on one terminal with: `jupyter notebook --port=8888` and start the scheduler on another terminal with command: `dask-scheduler --host 0.0.0.0` This will run the scheduler on port 8786 and dashboard on port 8787.



EC2 > Security Groups > sg-0ac617879a2feedd6 - launch-wizard-1 > Edit inbound rules

Edit inbound rules Info

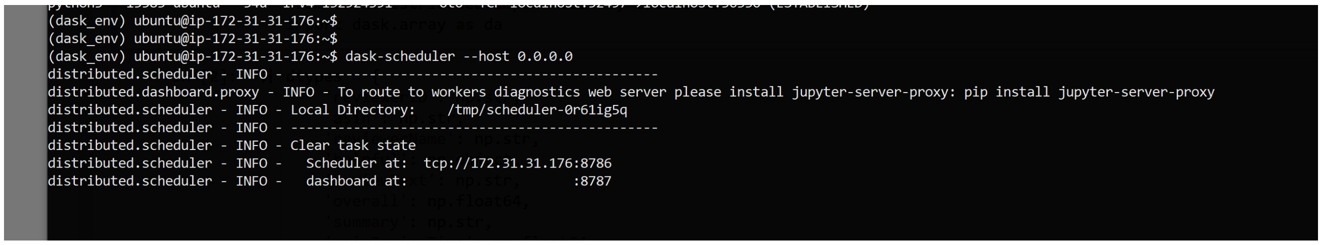
Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-0d1bc0ed1a06de92a	SSH	TCP	22	Custom	0.0.0.0/0		Delete
–	All TCP	TCP	0 - 65535	Custom	sg-0ac617879a2feedd6		Delete

Figure 1

<address-of-scheduler> is given at `tcp://172.31.31.176:8786` in the Figure 2 below.



```
(dask_env) ubuntu@ip-172-31-31-176:~$
(dask_env) ubuntu@ip-172-31-31-176:~$
(dask_env) ubuntu@ip-172-31-31-176:~$ dask-scheduler --host 0.0.0.0
distributed.scheduler - INFO - -----------------------------------------------
distributed.dashboard.proxy - INFO - To route to workers diagnostics web server please install jupyter-server-proxy: pip install jupyter-server-proxy
distributed.scheduler - INFO - Local Directory:    /tmp/scheduler-0r61ig5q
distributed.scheduler - INFO - -----------------------------------------------
distributed.scheduler - INFO - Clear task state
distributed.scheduler - INFO -   Scheduler at:  tcp://172.31.31.176:8786
distributed.scheduler - INFO -   dashboard at:                     :8787
```

Figure 2

c. Open a new terminal and SSH to jupyter notebook using: `ssh -i ‘‘dask-key.pem" ubuntu@<ip-address-of-EC2-instance> -L 8000:localhost:8888`. ‘-L’ will port forward any connection to port 8000 on the local machine to port 8888 on <ip-address-of-EC2-instance>. Type in `jupyter notebook list` to get the token/password for the jupyter notebook. Open your browser and go to localhost:8000 and paste the token. You can write your code here using jupyter notebook. To see dashboard on localhost port 8001 use command: `ssh -i ‘‘dask-key.pem" ubuntu@<ip-address-of-EC2-instance> -L 8001:localhost:8787`.

d. SSH into other 4 nodes and activate the dask environment. Start workers with command: `dask-worker <address-of-scheduler>:8786 --nworkers 4`. <address-of-scheduler> was displayed with command `dask-scheduler --host 0.0.0.0`. `nworkers` option will make sure that all the cores of the machine are used. In the scheduler terminal screen, you will be able to see the connected workers.

5. The data and files are available from the s3 bucket (s3://dsc102-public). This contains the function signature (PA1.py), datasets (user_reviews.csv and products.csv), schema of expected output (OutputSchema_PA1.json), and the expected result on the development dataset (results_PA1.json). Make sure that data is available in the same path where scheduler and workers are running. Refer to step 5 given in the getting started guide of PA0 for more details.

6. Open the dashboard and click on "Workers" to double check if all workers are connected and you are now ready to code up.

7. Terminate EC2 instances once you are done.

6 Workflow Guidance

Once you are comfortable launching your cluster and are confident that you are able to take advantage of all the workers on each node, you do not have to do all of your prototyping on the cluster. You can go back to a single instance, prototype your code, then scale back up to a cluster to yield runtime performance benefits.. The decision of how to proceed here is based on pareto efficiency, where the trade-off is between convenience, performance, and cost.