Laboratory 7: Spark Programming

Instructor: Young H. Cho

Apache Spark is a powerful distributed computing framework for efficiently processing very large datasets. You will explore its libraries using the PySpark API in both a standalone and distributed environment. (start on this early as you have to run your program on Hadoop, MPI and spark with varying number of worker nodes)

1. PySpark

We will first practice using Spark in local mode on a single-node machine. The following instructions only cover installation on an EC2 instance. If you are already familiar with using Docker containers, you may do this part on your own laptop or computer.

You may refer: https://aws.amazon.com/emr/pricing/ and select an instance with reduced cost. One you may use is c5.large instance type.

Create an EC2 instance using the **Amazon Linux 2 AMI** and select **c5.large** for your instance type. Make sure to modify your instance's security group inbound rules so that **port 8888** is accessible by **your IP address (tcp)**. You would be needing only one network interface and accordingly select auto public ip with the steps mentioned in lab8. Specify hdd space as 12 GB. (your username in Amazon Linux AMI is **ec2-user**) (you will need to perform this from AWS free tier account or from student account).(you should modify security group and network ACL rule in your VPC to allow public internet access while downloading the package).

Run the following commands to install Docker onto your instance and logout of your SSH session.

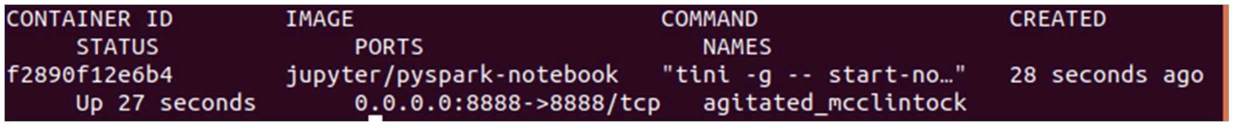
```
sudo yum update -y
sudo yum install -y docker
sudo service docker start
sudo usermod -a -G docker ec2-user
exit
```

SSH tunnel into your instance with the -L flag as shown below and run the subsequent commands to install the Docker image containing PySpark and Jupyter Notebook.

```
## Connect localhost port 8888 to instance port 8888
ssh -i myKey.pem ec2-user@XXXX.compute.amazonaws.com -L 8888:127.0.0.1:8888

## Install Docker image with PySpark Notebook
docker run -v ~/work:/home/jovyan/work -d -p 8888:8888 jupyter/pyspark-notebook

## Allow preserving Jupyter notebook
sudo chown 1000 ~/work

## Install tree to see our working directory next
sudo yum install -y tree
```

Run the following command to get the name of your container.

```
docker ps
```

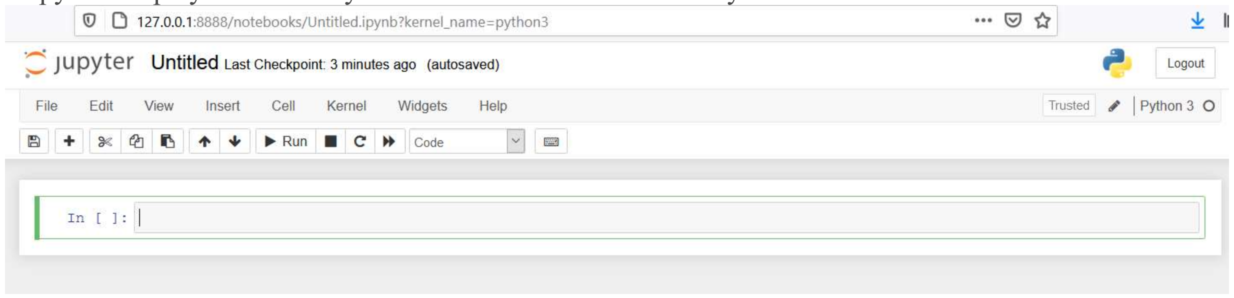


```
CONTAINER ID        IMAGE                       COMMAND                  CREATED
     STATUS              PORTS                    NAMES
f2890f12e6b4        jupyter/pyspark-notebook    "tini -g -- start-no…"   28 seconds ago
     Up 27 seconds       0.0.0.0:8888->8888/tcp   agitated_mcclintock
```

In this case, my container's name is "agitated_mcclintock". Use the Docker logs to get the link to access your Jupyter Notebook running PySpark by running the following command.

```
docker logs "name of container with no quotes"
```

Copy the displayed link into your browser and create a new Python 3 notebook.



127.0.0.1:8888/notebooks/Untitled.ipynb?kernel_name=python3
jupyter
Untitled Last Checkpoint: 3 minutes ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code
In [ ]:

Follow the tutorial linked below starting from section "Spark Context" to get familiar with the PySpark API. **The dataset that is used has been altered since the tutorial's writing, changing several features and breaking certain functions. Additionally, there are several typos that you will need to fix**. When you're finished, download your notebook and make sure to **<u>TERMINATE</u>** your instance to avoid additional costs. (You might want to google for how to use jupyter notebook instance and how to run the program step by step and re-execution)

https://www.guru99.com/pyspark-tutorial.html

For a better fundamental understanding of how Spark and its API work, it is recommended that you create an edX account and watch videos on machine learning with Spark.

https://www.edx.org/search?q=machine+learning+spark+

After completing the tutorial(s), download the "Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set" from the UCI Machine Learning Repository. This dataset contains accelerometer and gyroscope data from smartphones of people doing various activities. You can read more about the dataset and download it from the below link.

https://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions

Complete the following tasks on the dataset:

- Create a **machine learning model** using any algorithm not used in the tutorial from Spark's MLLib.
- Report the training accuracy, test accuracy, and test F1-score for your model.

2. AWS Elastic MapReduce

When you run Spark in local mode, it will allocate tasks to local threads instead of worker nodes. Spark is a distributed computing framework and was not primarily designed to be used on a single-node machine. Here, we will practice running a Spark job on a real cluster using AWS Elastic MapReduce.

First, upload the smartphone data from the previous part into an **S3** bucket.

You may refer: https://aws.amazon.com/emr/pricing/ and select an instance with reduced cost. You may use c5.xlarge instance type.

Go to EMR in the AWS console and click on the "Notebook" tab. Create a **two-**node cluster of instance type c5.xlarge (or m5.xlarge ->if you face issues). EMR will create several EC2 instances of your selected type (as specified in the instance count) and automatically configure your cluster manager. (you may try these out from student account)

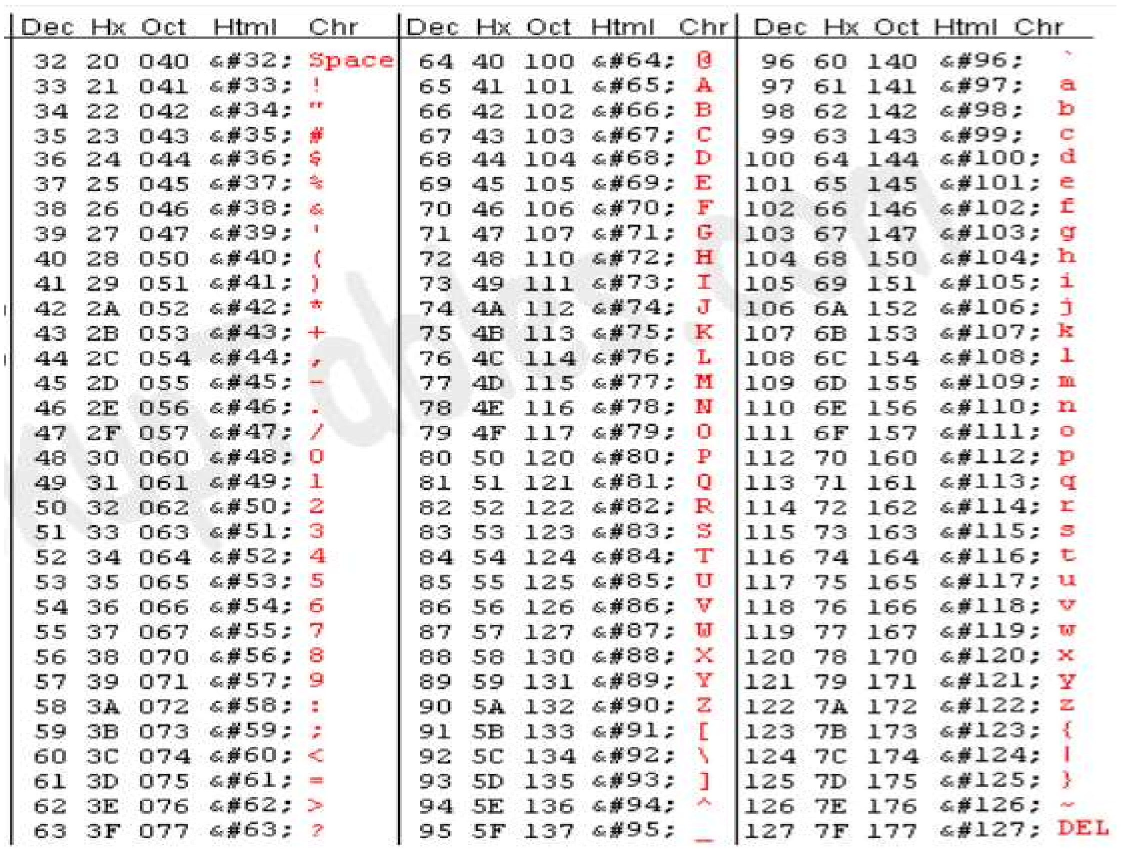
Wait for your cluster to start and your notebook to attach to the cluster. This can take anywhere from 5-10 minutes. Running a cluster in EMR is very expensive, so do not leave it idle and start using your notebook immediately when it becomes ready.

Open your notebook in JupyterLab, select your created notebook, and select PySpark for your kernel. Run the code from model you made in the previous part (i.e on Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set). In EMR, your **SparkContext object is already created** so you don't have to declare it. You can reference it using "sc" as usual. You will also have to **reference files from S3** instead of a local file directory.

When you're finished, **TERMINATE** your cluster in the "Clusters" tab to avoid extra charges. You will certainly **accumulate huge charges** if this is not done properly. Your notebook is saved in an S3 bucket (which you specified while creating), which you can download and delete later if you choose. (you might check EC2 instance tab and terminate instances created by EMR manually)

3. Custom Spark Program

Replicate functionally equivalent system as the codes you implemented using Hadoop and MPI on the previous Laboratory assignment with Spark. (i.e word count and frequency count of character with all the min and max word, character appended to an output file which gets stored in S3 bucket) (first try it on local spark and then on EMR at AWS -> to reduce cost of usage)
(you might want to first start testing with 5 MB Gutenberg file and after making everything work test it on 1 GB Gutenberg file ->
`http://www.i3s.unice.fr/~jplozi/hadooplab_lsds_2015/datasets/gutenberg-1G.txt.gz`
Also ensure that you **don't count space** as character for frequency count program and only take ascii characters **within ascii values of range [33 – 122]** and print it as character and not as integer . -> applicable for your previous Hadoop and MPI program also)



Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Compare **Spark system against Hadoop and MPI**. Make a diagram showing **scalability performances for different number of nodes**. (so, you might want to try out with different number of worker nodes and execute your program on MPI, Hadoop and Spark on your created cluster and plot a graph by varying number of worker nodes). (**your program should be written in a generic way to work with varying number of worker nodes**). (you may try these out from student account)
Make sure to take precautions for "AWS data transfer out charges" with security group, network ACL as well as iptables as mentioned during lab8.

Sources

https://www.guru99.com/pyspark-tutorial.html
https://courses.edx.org/courses/BerkeleyX/CS190.1x/1T2015/course/
https://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions
https://spark.apache.org/docs/latest/rdd-programming-guide.html