MAPREDUCE AND AWS(PYTHON + SPARK)

INTRODUCTION

MAPREDUCE

MapReduce is a framework for processing parallelizable problems across huge datasets using a large number of computers.

- Map step: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. The worker node processes the smaller problem and passes the answer back to its master node.
- Reduce step: The master node then collects the answers to all the sub-problems and combines them in some way to form the output the answer to the problem it was originally trying to solve.

MapReduce allows for distributed processing of the map and reduction operations. Provided that each mapping operation is independent of the others, all maps can be performed in parallel. Similarly, a set of reducers can perform the reduction phase, provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time, or that the reduction function is associative. Another way to look at MapReduce is as a 3-step parallel and distributed computation:

< K1,V 1 > −map− < K2,V 2 > −shuffle− < K2,a_list_of V 2 > −reduce− < K3,V 3 >

1. Prepare the Map() input the MapReduce system designates Map processors, assigns the K1 input key value each processor would work on, and provides that processor with all the input data associated with that key value. Run the user-provided Map() code Map() is run exactly once for each K1 key value, generating output organized by key values K2.
2. Shuffle the Map output to the Reduce processors the MapReduce system designates Reduce processors, assigns the K2 key value each processor would work on, and provides that processor with all the Map-generated data associated with that key value.
3. Run the user provided Reduce() code Reduce() is run exactly once for each K2 key value produced by the Map step. Produce the final output the MapReduce system collects all the Reduce output and sorts it by K2 to produce the final outcome.

A SIMPLE WORDCOUNT EXAMPLE

Go through document and video in lab 4-1 for setup and the word count example. Complete the setup process before you continue.

Important Notes:

1. **Don’t forget to stop/terminate your EC2 Instance after using it, otherwise it will cost your extra credits/money!**

   Also Please note that different regions are independent, so make sure you stop/terminate all running instances in different regions.

   In case your charges exceed the promotional credit, you can email AWS support request waiver for once only -- however this is only from our experience last year and may subject to change.

2. **You data on the EC2 instance will be lost if you terminate it. You can save your output results on S3, for Ipython Notebooks you create, you can download them from the browser via File -> Download As Ipython notebook.**

SHORT INTRO TO SPARK

Apache Spark is a fast and general-purpose cluster computing system, the main concept it introduces is RDD (resilient distributed dataset). You can think of RDD as a distributed dataset spread out in different servers, and offers convenient API for map and reduce operations, as well as other frequently used basic operations that can run in parallel.

Continue with the word count Example in the setup guide, and go over the programming guide (python), skip the following parts:

Linking with Spark, Initializing Spark, External Datasets

SPECIFICALLY FOCUS ON RDD OPERATIONS:

Make sure you understand the following concepts and know how to use the transformations and actions below, and experiment with it using your test text file, and the rdd given in the programming guide.

1. Closure
2. Transformations, the following operations are required.
   a. Map
   b. Flatmap
   c. reduceByKey
   d. groupByKey
   e. Filter
2. Actions
   a. Reduce
   b. Collect
   c. Count

d. countByKey

Question:

Can you just use map/flatmap and reduce/reduceByKey to implement the functionality of filter, count?

EXTRACT VALID WIKILINKS AND GENERATE ADJACENCY GRAPH

In this lab we will use the WikiLinks data set downloadable here (~30MB):

https://ufl.instructure.com/files/74909149/download?download_frd=1 or

https://cap4770-introds-east-2.s3.us-east-2.amazonaws.com/enwiki-latest-pages-articles1-trunc.xml

We recommend you download a very small example xml file to debug first:

https://ufl.instructure.com/files/7909004/download?download_frd=1 or

https://cap4770-introds-east-2.s3.us-east-2.amazonaws.com/example.xml

Instead of processing the large XML file directly.

Our main goal is to generate an adjacency graph from the dataset provided.

1. Extracts wikilinks and also remove all the red links. The dataset contains, for each article, a list of links going out of that article, as well as the articles title. The input for this job should be the Wikipedia dataset above, and the job should look at the XML text for each article, extract title and wikilinks. A single page would be represented as follows (with some fields and attributes omitted):
   <page> <title>Title</title> ...
   <text ...>
   Body of page
   [[link text]]
   </text>
   </page>
   Your task is to extract the Title and the wikilinks from these articles. Within the text of the page, a wikilink is indicated by [[link text]]. In the simplest kind of link, the link text is the name of another page (with any spaces replaced with underscores). A more complicated kind gives different text to appear as well as the name of the page: [[page name|text to appear]]. How to differentiate wikilinks to other types of links? Please refer to the Help:Link.

   There two formats of wikilinks as described in the above Wikipedia page. And repeating here:

   A. [[abc]] is seen as “abc” in text and links to page “abc”. We need to extract “abc” out.

B. [[a|b]] is labelled “b” on this page but links to page “a”. We need to extract “a” out. But “abc” in (A) and “a” in (B) may be a red link which basically means there is no page with title “abc” and “a” in the wiki dataset provided. You need to exclude all red links in this job.

   Note: we need to replace all the empty space ‘ ’ in the Title and links(title of other pages) with an underline ‘_’. No other processing is needed in this project.

2. Generate the Adjacency graph, i.e a graph with the format:

   page_title1   *link1*   *link2*
   page_titile2   *link2*   *link3*
   …

   With each line representing a page, and each item separated by **tab (\t)**.

3. Calculate the sparsity of the adjacency graph generated.

Important Note: you should only use the operations on RDD as mentioned above in Part 3 (including those not listed in this document, but in the programming guide). Directly processing the dataset using Python will not be scalable and won't handle large dataset.

HOW TO HANDLE XML FILE IN SPARK.

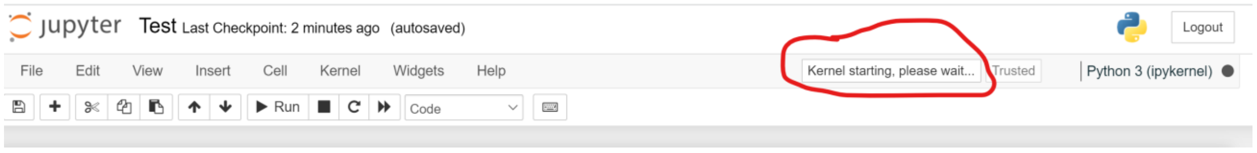
To handle xml with Spark, run pyspark with extra dependency like in lab 4-1:

```
$ pyspark --packages org.apache.spark:spark-hadoop-cloud_2.12:3.2.0,com.databricks:spark-
xml_2.12:0.16.0

Or

$ pyspark --packages com.amazonaws:aws-java-sdk:1.12.403,org.apache.hadoop:hadoop-
aws:3.3.1,com.databricks:spark-xml_2.12:0.16.0
```

wait for the kernel to be ready, then run your code blocks



In Ipython Notebook, use the following:

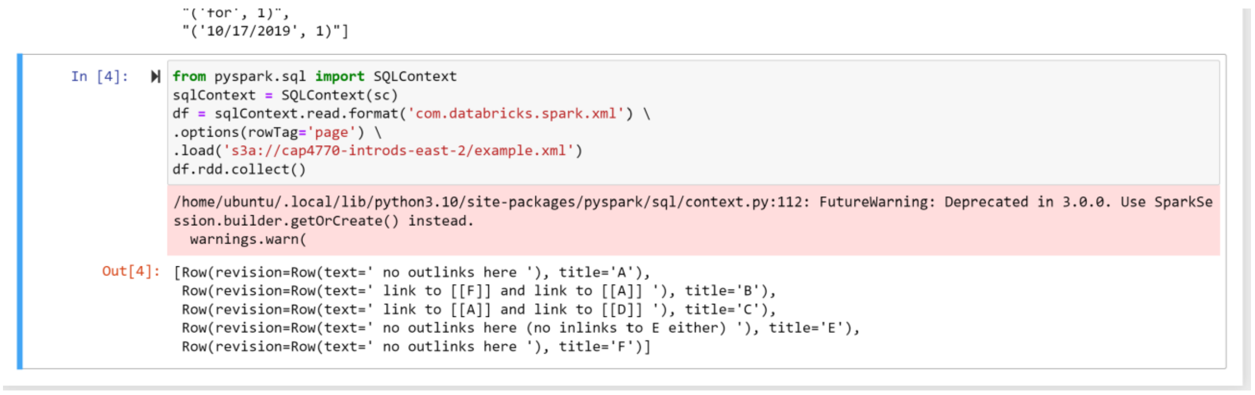
```
from pyspark.sql import
SQLContext

sqlContext = SQLContext(sc)

df = sqlContext.read.format('com.databricks.spark.xml') \
    .options(rowTag='page') \

    .load('s3a://path_to_your_xml_file/example.xml')

df.rdd.collect()
```



```
 "('for', 1)",
 "('10/17/2019', 1)"]
```

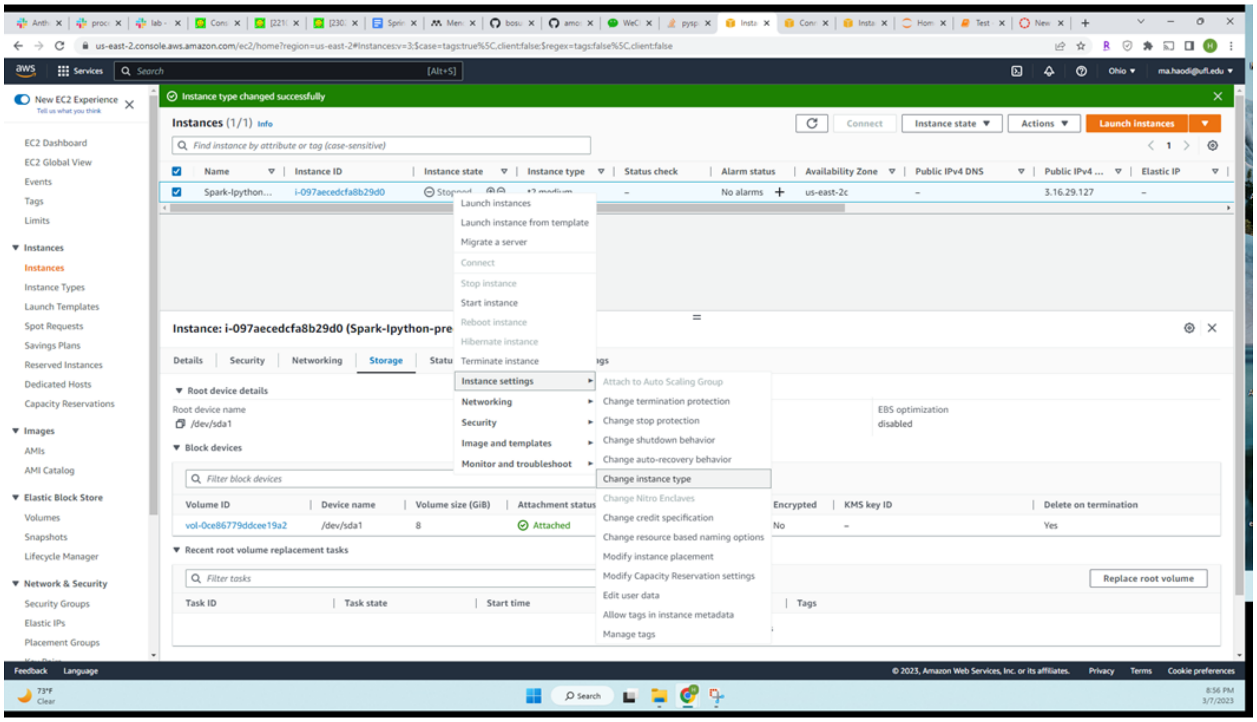
```
In [4]:   from pyspark.sql import SQLContext
          sqlContext = SQLContext(sc)
          df = sqlContext.read.format('com.databricks.spark.xml') \
          .options(rowTag='page') \
          .load('s3a://cap4770-introds-east-2/example.xml')
          df.rdd.collect()

          /home/ubuntu/.local/lib/python3.10/site-packages/pyspark/sql/context.py:112: FutureWarning: Deprecated in 3.0.0. Use SparkSe
          ssion.builder.getOrCreate() instead.
            warnings.warn(

Out[4]:   [Row(revision=Row(text=' no outlinks here '), title='A'),
           Row(revision=Row(text=' link to [[F]] and link to [[A]] '), title='B'),
           Row(revision=Row(text=' link to [[A]] and link to [[D]] '), title='C'),
           Row(revision=Row(text=' no outlinks here (no inlinks to E either) '), title='E'),
           Row(revision=Row(text=' no outlinks here '), title='F')]
```

If you get error message related to time-out or memory issues, please try the following steps:

1. change your instance type to t2.medium or t2.large. (You need to stop your instance to change this)



us-east-2.console.aws.amazon.com/ec2/home?region=us-east-2#Instances:v=3;$case=tags:true%5C,client:false;$regex=tags:false%5C,client:false

Instance type changed successfully

Instances (1/1)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Spark-Ipython...	i-097aecedcfa8b29d0	Stopped	t2.medium	–	No alarms	us-east-2c	–	3.16.29.127	–

Instance: i-097aecedcfa8b29d0 (Spark-Ipython-pre

Root device details

Root device name: /dev/sda1

EBS optimization: disabled

Block devices

Volume ID	Device name	Volume size (GiB)	Attachment status	Encrypted	KMS key ID	Delete on termination
vol-0ce86779ddcee19a2	/dev/sda1	8	Attached	No	–	Yes

Recent root volume replacement tasks

Task ID	Task state	Start time	Tags

2. run Pyspark with pyspark --executor-memory 1g --executor-cores 1 --driver-memory 1g --packages org.apache.spark:spark-hadoop-cloud_2.12:3.2.0,com.databricks:spark-xml_2.12:0.16.0

3. You can change the parameters above (--executor, --driver-memory) to what fits the data size.

HOW TO EXTRACT WIKILINKS IN TEXT

You’ll need to extract links using regular expressions, to avoid the excruciating details on how to construct the regular expression and extract wikilinks, we provide you the code to do it:

```
import re

def get_links(text):

    pattern = re.compile(r"\[\[(.*?)\]\]")

    res = pattern.findall(text)

    return [links.split('|')[0] for links in res] if res
!= None else []
```

This function “get_links(text)” will extract the links we need in text. Note the input parameter “text” is a python **string** which contains the “[[...]]”.

The code above will not remove red links. You have to check and remove them by yourselves.

EXTRA TUTORIALS

AWS Training and Certification

MapReduce

Amazon EMR

Wikilinks