Big Data Coursework - Questions

Data Processing and Machine Learning in the Cloud

This is the **INM432 Big Data coursework 2024**. This coursework contains extended elements of **theory** and **practice**, mainly around parallelisation of tasks withSpark and a bit about parallel training using TensorFlow.

Code and Report

Your tasks parallelization of tasks in PySpark, extension, evaluation, and theoretical reflection.
Please complete and submit the **coding tasks** in a copy of **this notebook**. Write your code in the **indicated cells** and **include** the **output** in the submitted notebook.
Make sure that **your code contains comments** on its **stucture** and explanations of its **purpose**.

Provide also a **report** with the **textual answers in a separate document**.
Include **screenshots** from the Google Cloud web interface (don't use the SCREENSHOT function that Google provides, but take a picture of the graphs you see for the VMs) and result tables, as well as written text about the analysis.

Submission

Download and submit **your version of this notebook** as an **.ipynb** file and also submit a **shareable link** to your notebook on Colab in your report (created with the Colab 'Share' function) (**and don't change the online version after submission**).

Further, provide your **report as a PDF document**. **State the number of words** in the document at the end. The report should **not have more than 2000 words**.

Please also submit **a PDF of your Jupyter notebook**.

Introduction and Description

This coursework focuses on parallelisation and scalability in the cloud with Spark and TesorFlow/Keras. We start with code based on **lessons 3 and 4** of the *Fast and Lean Data Science* course by Martin Gorner. The course is based on Tensorflow for data processing and MachineLearning. Tensorflow's data processing approach is somewhat similar to that of Spark, but you don't need to study Tensorflow, just make sure you understand the high-level structure. What we will do here is **parallelising pre-processing**, and **measuring** performance, and we will perform **evaluation** and **analysis** on the cloud performance, as well as **theoretical discussion**.

This coursework contains **3 sections**.

Section 0

This section just contains some necessary code for setting up the environment. It has no tasks for you (but do read the code and comments).

Section 1

Section 1 is about preprocessing a set of image files. We will work with a public dataset “Flowers” (3600 images, 5 classes). This is not a vast dataset, but it keeps the tasks more manageable for development and you can scale up later, if you like.

In **'Getting Started'** we will work through the data preprocessing code from *Fast and Lean Data Science* which uses TensorFlow's `tf.data` package. There is no task for you here, but you will need to re-use some of this code later.

In **Task 1** you will **parallelise the data preprocessing in Spark**, using Google Cloud (GC) Dataproc. This involves adapting the code from 'Getting Started' to use Spark and running it in the cloud.

Section 2

In **Section 2** we are going to **measure the speed of reading data** in the cloud. In **Task 2** we will **paralellize the measuring** of different configurations **using Spark**.

Section 3

This section is about the theoretical discussion, based on one paper, in **Task 3**. The answers should be given in the PDF report.

General points

For **all coding tasks**, take the **time of the operations** and for the cloud operations, get performance **information from the web interfaces** for your reporting and analysis.

The **tasks** are **mostly independent** of each other. The later tasks can mostly be addressed without needing the solution to the earlier ones.

Section 0: Set-up

As usual, you need to run the **imports and authentication every time you work with this notebook**. Use the **local Spark** installation for development before you send jobs to the cloud.

Read through this section once and **fill in the project ID the first time**, then you can just step straight throught this at the beginning of each session - except for the two authentication cells.

Imports

We import some **packages that will be needed throughout**. For the **code that runs in the cloud**, we will need **separate import sections** that will need to be partly different from the one below.

```
import os, sys, math
import numpy as np
import scipy as sp
import scipy.stats
```

```
import time
import datetime
import string
import random
from matplotlib import pyplot as plt
import tensorflow as tf
print("Tensorflow version " + tf.__version__)
import pickle
```

```
Tensorflow version 2.15.0
```

Cloud and Drive authentication

This is for **authenticating with with GCS Google Drive**, so that we can create and use our own buckets and access Dataproc and AI-Platform.

This section **starts with the two interactive authentications**.

First, we mount Google Drive for persistent local storage and create a directory `DB-CW` thay you can use for this work. Then we'll set up the cloud environment, including a storage bucket.

```
print('Mounting google drive...')
from google.colab import drive
drive.mount('/content/drive')
%cd "/content/drive/MyDrive"
!mkdir BD-CW
%cd "/content/drive/MyDrive/BD-CW"
```

```
Mounting google drive...
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
/content/drive/MyDrive
mkdir: cannot create directory ‘BD-CW’: File exists
/content/drive/MyDrive/BD-CW
```

Next, we authenticate with the GCS to enable access to Dataproc and AI-Platform.

```
import sys
if 'google.colab' in sys.modules:
    from google.colab import auth
    auth.authenticate_user()
```

It is useful to **create a new Google Cloud project** for this coursework. You can do this on the GC Console page by clicking on the entry at the top, right of the *Google Cloud Platform* and choosing *New Project*. **Copy** the **generated project ID** to the next cell. Also **enable billing** and the **Compute, Storage and Dataproc** APIs like we did during the labs.

We also specify the **default project and region**. The REGION should be `us-central1` as that seems to be the only one that reliably works with the free credit. This way we don't have to specify this information every time we access the cloud.

```
PROJECT = 'bigdata-421920'  ### USE YOUR GOOGLE CLOUD PROJECT ID HERE.
###
!gcloud config set project $PROJECT
REGION = 'us-central1'
CLUSTER = '{}-cluster'.format(PROJECT)
!gcloud config set compute/region $REGION
!gcloud config set dataproc/region $REGION

!gcloud config list # show some information
```

```
Updated property [core/project].
WARNING: Property validation for compute/region was skipped.
Updated property [compute/region].
Updated property [dataproc/region].
[component_manager]
disable_update_check = True
[compute]
region = us-central1
[core]
account = sahanchowdhury00@gmail.com
project = bigdata-421920
[dataproc]
region = us-central1

Your active configuration is: [default]
```

With the cell below, we **create a storage bucket** that we will use later for **global storage**. If the bucket exists you will see a "ServiceException: 409 ...", which does not cause any problems. **You must create your own bucket to have write access.**

```
BUCKET = 'gs://{}-storage'.format(PROJECT)
!gsutil mb $BUCKET
```

```
Creating gs://bigdata-421920-storage/...
```

The cell below just **defines some routines for displaying images** that will be **used later**. You can see the code by double-clicking, but you don't need to study this.

```
#@title Utility functions for image display **[RUN THIS TO ACTIVATE]**
{ display-mode: "form" }
def display_9_images_from_dataset(dataset):
  plt.figure(figsize=(13,13))
  subplot=331
  for i, (image, label) in enumerate(dataset):
    plt.subplot(subplot)
    plt.axis('off')
    plt.imshow(image.numpy().astype(np.uint8))
    plt.title(str(label.numpy()), fontsize=16)
    # plt.title(label.numpy().decode(), fontsize=16)
```

```
    subplot += 1
    if i==8:
      break
  plt.tight_layout()
  plt.subplots_adjust(wspace=0.1, hspace=0.1)
  plt.show()

def display_training_curves(training, validation, title, subplot):
  if subplot%10==1: # set up the subplots on the first call
    plt.subplots(figsize=(10,10), facecolor='#F0F0F0')
    plt.tight_layout()
  ax = plt.subplot(subplot)
  ax.set_facecolor('#F8F8F8')
  ax.plot(training)
  ax.plot(validation)
  ax.set_title('model '+ title)
  ax.set_ylabel(title)
  ax.set_xlabel('epoch')
  ax.legend(['train', 'valid.'])

def dataset_to_numpy_util(dataset, N):
    dataset = dataset.batch(N)
    for images, labels in dataset:
        numpy_images = images.numpy()
        numpy_labels = labels.numpy()
        break;
    return numpy_images, numpy_labels

def title_from_label_and_target(label, correct_label):
  correct = (label == correct_label)
  return "{} [{}{}{}]".format(CLASSES[label], str(correct), ', shoud
be ' if not correct else '',
                              CLASSES[correct_label] if not correct
else ''), correct

def display_one_flower(image, title, subplot, red=False):
    plt.subplot(subplot)
    plt.axis('off')
    plt.imshow(image)
    plt.title(title, fontsize=16, color='red' if red else 'black')
    return subplot+1

def display_9_images_with_predictions(images, predictions, labels):
  subplot=331
  plt.figure(figsize=(13,13))
  classes = np.argmax(predictions, axis=-1)
  for i, image in enumerate(images):
    title, correct = title_from_label_and_target(classes[i],
labels[i])
    subplot = display_one_flower(image, title, subplot, not correct)
```

```
    if i >= 8:
      break;

  plt.tight_layout()
  plt.subplots_adjust(wspace=0.1, hspace=0.1)
  plt.show()
```

Install Spark locally for quick testing

You can use the cell below to **install Spark locally on this Colab VM** (like in the labs), to do quicker small-scale interactive testing. Using Spark in the cloud with **Dataproc is still required for the final version**.

```
%cd
!apt-get update -qq
!apt-get install openjdk-8-jdk-headless -qq >> /dev/null # send any
output to null device
!tar -xzf "/content/drive/My Drive/Big_Data/data/spark/spark-3.5.0-
bin-hadoop3.tgz" # unpack

!pip install -q findspark
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/root/spark-3.5.0-bin-hadoop3"
import findspark
findspark.init()
import pyspark
print(pyspark.__version__)
sc = pyspark.SparkContext.getOrCreate()
print(sc)
```

```
/root
3.5.0
<SparkContext master=local[*] appName=pyspark-shell>
```

Section 1: Data pre-processing

This section is about the **pre-processing of a dataset** for deep learning. We first look at a ready-made solution using Tensorflow and then we build a implement the same process with Spark. The tasks are about **parallelisation** and **analysis** the performance of the cloud implementations.

1.1 Getting started

In this section, we get started with the data pre-processing. The code is based on lecture 3 of the 'Fast and Lean Data Science' course.

This code is using the TensorFlow `tf.data` package, which supports map functions, similar to Spark. Your **task** will be to **re-implement the same approach in Spark**.

We start by **setting some variables for the *Flowers* dataset**.

```
GCS_PATTERN = 'gs://flowers-public/*/*.jpg' # glob  pattern for input
files
PARTITIONS = 16 # no of partitions we will use later
TARGET_SIZE = [192, 192] # target resolution for the images
CLASSES = [b'daisy', b'dandelion', b'roses', b'sunflowers', b'tulips']
    # labels for the data
```

We **read the image files** from the public GCS bucket that contains the *Flowers* dataset. **TensorFlow** has **functions** to execute glob patterns that we use to calculate the the number of images in total and per partition (rounded up as we cannont deal with parts of images).

```
nb_images = len(tf.io.gfile.glob(GCS_PATTERN)) # number of images
partition_size = math.ceil(1.0 * nb_images / PARTITIONS) # images per
partition (float)
print("GCS_PATTERN matches {} images, to be divided into {} partitions
with up to {} images each.".format(nb_images, PARTITIONS,
partition_size))
```

```
GCS_PATTERN matches 3670 images, to be divided into 16 partitions with
up to 230 images each.
```

Map functions

In order to read use the images for learning, they need to be **preprocessed** (decoded, resized, cropped, and potentially recompressed). Below are **map functions** for these steps. You **don't need to study** the **internals of these functions** in detail.

```
def decode_jpeg_and_label(filepath):
    # extracts the image data and creates a class label, based on the
filepath
    bits = tf.io.read_file(filepath)
    image = tf.image.decode_jpeg(bits)
    # parse flower name from containing directory
    label = tf.strings.split(tf.expand_dims(filepath, axis=-1),
sep='/')
    label2 = label.values[-2]
    return image, label2

def resize_and_crop_image(image, label):
    # Resizes and cropd using "fill" algorithm:
    # always make sure the resulting image is cut out from the source
image
    # so that it fills the TARGET_SIZE entirely with no black bars
    # and a preserved aspect ratio.
```

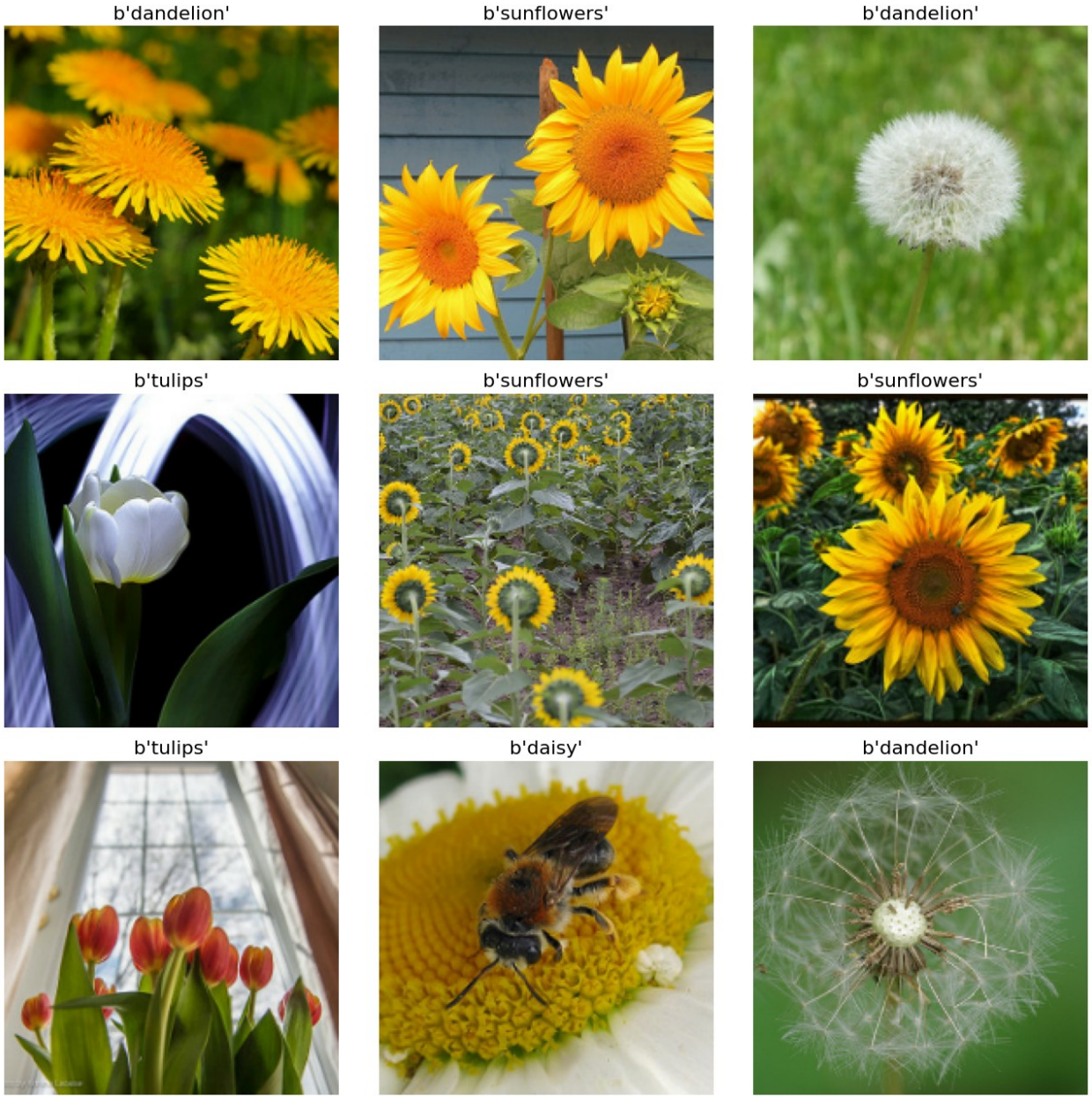
```
    w = tf.shape(image)[0]
    h = tf.shape(image)[1]
    tw = TARGET_SIZE[1]
    th = TARGET_SIZE[0]
    resize_crit = (w * th) / (h * tw)
    image = tf.cond(resize_crit < 1,
                    lambda: tf.image.resize(image, [w*tw/w, h*tw/w]),
# if true
                    lambda: tf.image.resize(image, [w*th/h, h*th/h])
# if false
                    )
    nw = tf.shape(image)[0]
    nh = tf.shape(image)[1]
    image = tf.image.crop_to_bounding_box(image, (nw - tw) // 2, (nh -
th) // 2, tw, th)
    return image, label

def recompress_image(image, label):
    # this reduces the amount of data, but takes some time
    image = tf.cast(image, tf.uint8)
    image = tf.image.encode_jpeg(image, optimize_size=True,
chroma_downsampling=False)
    return image, label
```

With `tf.data`, we can apply decoding and resizing as map functions.

```
dsetFiles = tf.data.Dataset.list_files(GCS_PATTERN) # This also
shuffles the images
dsetDecoded = dsetFiles.map(decode_jpeg_and_label)
dsetResized = dsetDecoded.map(resize_and_crop_image)
```

We can also look at some images using the image display function defined above (the one with the hidden code).

```
display_9_images_from_dataset(dsetResized)
```



Now, let's test continuous reading from the dataset. We can see that reading the first 100 files already takes some time.

```
sample_set = dsetResized.batch(10).take(10) # take 10 batches of 10
images for testing
for image, label in sample_set:
    print("Image batch shape {}, {})".format(image.numpy().shape,
        [lbl.decode('utf8') for lbl in label.numpy()]))

Image batch shape (10, 192, 192, 3), ['tulips', 'tulips', 'tulips',
'roses', 'dandelion', 'dandelion', 'tulips', 'dandelion', 'dandelion',
'sunflowers'])
```

```
Image batch shape (10, 192, 192, 3), ['roses', 'roses', 'daisy',
'daisy', 'daisy', 'roses', 'roses', 'sunflowers', 'dandelion',
'dandelion'])
Image batch shape (10, 192, 192, 3), ['daisy', 'tulips', 'tulips',
'dandelion', 'roses', 'tulips', 'sunflowers', 'sunflowers',
'dandelion', 'daisy'])
Image batch shape (10, 192, 192, 3), ['roses', 'dandelion', 'roses',
'sunflowers', 'dandelion', 'dandelion', 'dandelion', 'sunflowers',
'daisy', 'sunflowers'])
Image batch shape (10, 192, 192, 3), ['tulips', 'sunflowers', 'daisy',
'tulips', 'daisy', 'dandelion', 'roses', 'dandelion', 'daisy',
'dandelion'])
Image batch shape (10, 192, 192, 3), ['dandelion', 'dandelion',
'tulips', 'roses', 'daisy', 'dandelion', 'tulips', 'dandelion',
'dandelion', 'tulips'])
Image batch shape (10, 192, 192, 3), ['roses', 'dandelion',
'sunflowers', 'dandelion', 'roses', 'dandelion', 'sunflowers',
'sunflowers', 'sunflowers', 'dandelion'])
Image batch shape (10, 192, 192, 3), ['roses', 'tulips', 'daisy',
'roses', 'daisy', 'tulips', 'dandelion', 'dandelion', 'dandelion',
'dandelion'])
Image batch shape (10, 192, 192, 3), ['dandelion', 'sunflowers',
'dandelion', 'dandelion', 'sunflowers', 'sunflowers', 'dandelion',
'dandelion', 'daisy', 'dandelion'])
Image batch shape (10, 192, 192, 3), ['tulips', 'dandelion',
'dandelion', 'roses', 'roses', 'dandelion', 'tulips', 'tulips',
'roses', 'tulips'])
```

1.2 Improving Speed

Using individual image files didn't look very fast. The 'Lean and Fast Data Science' course introduced **two techniques to improve the speed**.

Recompress the images

By **compressing** the images in the **reduced resolution** we save on the size. This **costs some CPU time** upfront, but **saves network and disk bandwith**, especially when the data are **read multiple times**.

```
# This is a quick test to get an idea how long recompressions takes.
dataset4 = dsetResized.map(recompress_image)
test_set = dataset4.batch(10).take(10)
for image, label in test_set:
    print("Image batch shape {}, {})".format(image.numpy().shape,
[lbl.decode('utf8') for lbl in label.numpy()]))
```

```
Image batch shape (10,), ['roses', 'roses', 'dandelion', 'sunflowers',
'daisy', 'dandelion', 'daisy', 'tulips', 'roses', 'daisy'])
Image batch shape (10,), ['sunflowers', 'tulips', 'roses', 'roses',
```

```
'roses', 'dandelion', 'roses', 'dandelion', 'tulips', 'dandelion'])
Image batch shape (10,), ['tulips', 'roses', 'sunflowers',
'dandelion', 'daisy', 'dandelion', 'daisy', 'dandelion', 'tulips',
'dandelion'])
Image batch shape (10,), ['tulips', 'daisy', 'dandelion',
'sunflowers', 'daisy', 'daisy', 'daisy', 'sunflowers', 'daisy',
'roses'])
Image batch shape (10,), ['dandelion', 'daisy', 'dandelion', 'roses',
'sunflowers', 'sunflowers', 'tulips', 'daisy', 'sunflowers',
'tulips'])
Image batch shape (10,), ['tulips', 'daisy', 'tulips', 'tulips',
'roses', 'sunflowers', 'daisy', 'sunflowers', 'tulips', 'roses'])
Image batch shape (10,), ['daisy', 'daisy', 'roses', 'tulips',
'dandelion', 'dandelion', 'dandelion', 'tulips', 'roses', 'roses'])
Image batch shape (10,), ['daisy', 'sunflowers', 'dandelion', 'daisy',
'sunflowers', 'sunflowers', 'dandelion', 'roses', 'roses',
'dandelion'])
Image batch shape (10,), ['daisy', 'sunflowers', 'daisy', 'roses',
'roses', 'tulips', 'sunflowers', 'tulips', 'roses', 'daisy'])
Image batch shape (10,), ['tulips', 'tulips', 'tulips', 'roses',
'dandelion', 'daisy', 'tulips', 'dandelion', 'dandelion',
'sunflowers'])
```

Write the dataset to TFRecord files

By writing **multiple preprocessed samples into a single file**, we can make further speed gains. We distribute the data over **partitions** to facilitate **parallelisation** when the data are used. First we need to **define a location** where we want to put the file.

```
GCS_OUTPUT = BUCKET + '/tfrecords-jpeg-192x192-2/flowers'  # prefix
for output file names
```

Now we can **write the TFRecord files** to the bucket.

Running the cell takes some time and **only needs to be done once** or not at all, as you can use the publicly available data for the next few cells. For convenience I have commented out the call to `write_tfrecords` at the end of the next cell. You don't need to run it (it takes some time), but you'll need to use the code below later (but there is no need to study it in detail).

There is a **ready-made pre-processed data** versions available here: `gs://flowers-public/tfrecords-jpeg-192x192-2/`, that we can use for testing.

```
# functions for writing TFRecord entries
# Feature values are always stored as lists, a single data element
will be a list of size 1
def _bytestring_feature(list_of_bytestrings):
    return
tf.train.Feature(bytes_list=tf.train.BytesList(value=list_of_bytestrin
gs))
```

```
def _int_feature(list_of_ints): # int64
    return
tf.train.Feature(int64_list=tf.train.Int64List(value=list_of_ints))

def to_tfrecord(tfrec_filewriter, img_bytes, label): # Create tf data
records
    class_num = np.argmax(np.array(CLASSES)==label) # 'roses' => 2
(order defined in CLASSES)
    one_hot_class = np.eye(len(CLASSES))[class_num]     # [0, 0, 1, 0,
0] for class #2, roses
    feature = {
        "image": _bytestring_feature([img_bytes]), # one image in the
list
        "class": _int_feature([class_num]) #,      # one class in the
list
    }
    return
tf.train.Example(features=tf.train.Features(feature=feature))

def write_tfrecords(GCS_PATTERN,GCS_OUTPUT,partition_size): # write
the images to files.
    print("Writing TFRecords")
    tt0 = time.time()
    filenames = tf.data.Dataset.list_files(GCS_PATTERN)
    dataset1 = filenames.map(decode_jpeg_and_label)
    dataset2 = dataset1.map(resize_and_crop_image)
    dataset3 = dataset2.map(recompress_image)
    dataset4 = dataset3.batch(partition_size) # partitioning: there
will be one "batch" of images per file
    for partition, (image, label) in enumerate(dataset4):
        # batch size used as partition size here
        partition_size = image.numpy().shape[0]
        # good practice to have the number of records in the filename
        filename = GCS_OUTPUT + "{:02d}-{}.tfrec".format(partition,
partition_size)
        # You need to change GCS_OUTPUT to your own bucket to actually
create new files
        with tf.io.TFRecordWriter(filename) as out_file:
            for i in range(partition_size):
                example = to_tfrecord(out_file,
                                      image.numpy()[i], # re-compressed
image: already a byte string
                                      label.numpy()[i] #
                                      )
                out_file.write(example.SerializeToString())
        print("Wrote file {} containing {} records".format(filename,
partition_size))
    print("Total time: "+str(time.time()-tt0))
```

```
write_tfrecords(GCS_PATTERN,GCS_OUTPUT,partition_size) # uncomment to
run this cell

Writing TFRecords
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers00-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers01-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers02-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers03-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers04-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers05-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers06-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers07-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers08-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers09-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers10-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers11-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers12-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers13-
230.tfrec containing 230 records
Wrote file
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers14-
230.tfrec containing 230 records
Wrote file
```

```
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers15-
220.tfrec containing 220 records
Total time: 141.24735474586487
```

Test the TFRecord files

We can now **read from the TFRecord files**. By default, we use the files in the public bucket. Comment out the 1st line of the cell below to use the files written in the cell above.

```
#GCS_OUTPUT = 'gs://flowers-public/tfrecords-jpeg-192x192-2/'
# remove the line above to use your own files that you generated above

def read_tfrecord(example):
    features = {
        "image": tf.io.FixedLenFeature([], tf.string),  # tf.string =
bytestring (not text string)
        "class": tf.io.FixedLenFeature([], tf.int64) #,   # shape []
means scalar
    }
    # decode the TFRecord
    example = tf.io.parse_single_example(example, features)
    image = tf.image.decode_jpeg(example['image'], channels=3)
    image = tf.reshape(image, [*TARGET_SIZE, 3])
    class_num = example['class']
    return image, class_num

def load_dataset(filenames):
    # read from TFRecords. For optimal performance, read from multiple
    # TFRecord files at once and set the option
experimental_deterministic = False
    # to allow order-altering optimizations.
    option_no_order = tf.data.Options()
    option_no_order.experimental_deterministic = False

    dataset = tf.data.TFRecordDataset(filenames)
    dataset = dataset.with_options(option_no_order)
    dataset = dataset.map(read_tfrecord)
    return dataset


filenames = tf.io.gfile.glob(GCS_OUTPUT + "*.tfrec")
datasetTfrec = load_dataset(filenames)
```

Let's have a look **if reading from the TFRecord** files is **quicker**.

```
batched_dataset = datasetTfrec.batch(10)
sample_set = batched_dataset.take(10)
for image, label in sample_set:
```

```
    print("Image batch shape {}, {})".format(image.numpy().shape, \
                        [str(lbl) for lbl in label.numpy()]))
```

```
Image batch shape (10, 192, 192, 3), ['2', '4', '4', '4', '2', '3',
'3', '2', '2', '1'])
Image batch shape (10, 192, 192, 3), ['3', '3', '4', '4', '0', '2',
'4', '2', '1', '0'])
Image batch shape (10, 192, 192, 3), ['4', '2', '2', '2', '0', '1',
'0', '4', '4', '1'])
Image batch shape (10, 192, 192, 3), ['1', '1', '4', '1', '0', '4',
'2', '2', '1', '1'])
Image batch shape (10, 192, 192, 3), ['1', '3', '2', '2', '2', '3',
'1', '0', '4', '2'])
Image batch shape (10, 192, 192, 3), ['1', '4', '3', '1', '1', '1',
'4', '1', '1', '1'])
Image batch shape (10, 192, 192, 3), ['4', '3', '1', '4', '1', '1',
'3', '2', '3', '1'])
Image batch shape (10, 192, 192, 3), ['2', '0', '1', '1', '0', '1',
'0', '2', '1', '2'])
Image batch shape (10, 192, 192, 3), ['1', '2', '1', '1', '4', '4',
'2', '2', '3', '3'])
Image batch shape (10, 192, 192, 3), ['4', '2', '4', '1', '1', '1',
'1', '2', '2', '1'])
```

Wow, we have a **massive speed-up**! The repackageing is worthwhile :-)

Task 1: Write TFRecord files to the cloud with Spark (40%)

Since recompressing and repackaging is very effective, we would like to be able to do it inparallel for large datasets. This is a relatively straightforward case of **parallelisation**. We will **use Spark to implement** the same process as above, but in parallel.

1a) Create the script (14%)

Re-implement the pre-processing in Spark, using Spark mechanisms for **distributing** the workload **over multiple machines**.

You need to:

i) **Copy** over the **mapping functions** (see section 1.1) and **adapt** the resizing and recompression functions **to Spark** (only one argument). (3%)

ii) **Replace** the TensorFlow **Dataset objects with RDDs**, starting with an RDD that contains the list of image filenames. (3%)

iii) **Sample** the the RDD to a smaller number at an appropriate position in the code. Specify a sampling factor of 0.02 for short tests. (1%)

iv) Then **use the functions from above** to write the TFRecord files. (3%)

v) The code for **writing to the TFRecord files** needs to be put into a function, that can be applied to every partition with the 'RDD.mapPartitionsWithIndex' function. The return value of that function is not used here, but you should return the filename, so that you have a list of the created TFRecord files. (4%)

```
# import required libraries
import os, sys, math
import numpy as np
import scipy as sp
import scipy.stats
import time
import datetime
import string
import random
from matplotlib import pyplot as plt
import tensorflow as tf
print("Tensorflow version " + tf.__version__)
import pickle
import pyspark
from pyspark.sql import SQLContext
from pyspark.sql import Row
```

```
Tensorflow version 2.15.0
```

```
### CODING TASK ###

#Section 1A

###i) Copy over the mapping functions (see section 1.1) and adapt the
resizing and recompression functions to Spark (only one argument).
(3%)

#Function to decode JPEG image and extract label from the filepath
def decode_jpeg_and_label(filepath):
    # extracts the image data and creates a class label, based on the
filepath
    bits = tf.io.read_file(filepath)
    image = tf.image.decode_jpeg(bits)
    # parse flower name from containing directory
    label = tf.strings.split(tf.expand_dims(filepath, axis=-1),
sep='/')
    label2 = label.values[-2]
    return image, label2

#Function to resize and crop
def resize_and_crop_image(data):
    # Resizes and cropd using "fill" algorithm:
    # always make sure the resulting image is cut out from the source
image
    # so that it fills the TARGET_SIZE entirely with no black bars
```

```
    # and a preserved aspect ratio.
    #Obtain image and label from the data
    image, label = data
    #Dimension of image
    w = tf.shape(image)[0]
    h = tf.shape(image)[1]
    #Target size
    tw = TARGET_SIZE[1]
    th = TARGET_SIZE[0]
    #Resizing criteria calculated
    resize_crit = (w * th) / (h * tw)
    #Here the image gets resized according to the rezising criteria
    image = tf.cond(resize_crit < 1,
                    lambda: tf.image.resize(image, [w*tw/w, h*tw/w]),
# if true
                    lambda: tf.image.resize(image, [w*th/h, h*th/h])
# if false
                    )
    #New dimensions of image calculated
    nw = tf.shape(image)[0]
    nh = tf.shape(image)[1]
    #Image gets cropped to the target size
    image = tf.image.crop_to_bounding_box(image, (nw - tw) // 2, (nh -
th) // 2, tw, th)
    return image, label

#Function to recompress the image
def recompress_image(data):
    # this reduces the amount of data, but takes some time
    #Obtain image and label from the data
    image, label = data
    #Image is 'casted' to an 8 bit integer format
    image = tf.cast(image, tf.uint8)
    #Image gets encoded to jep format
    image = tf.image.encode_jpeg(image, optimize_size=True,
chroma_downsampling=False)
    return image, label



###ii) Replace the TensorFlow Dataset objects with RDDs, starting with
an RDD that contains the list of image filenames. (3%)


#Required glob  pattern for input files
GCS_PATTERN = 'gs://flowers-public/*/*.jpg'
#Spark context
sc = pyspark.SparkContext.getOrCreate()
#Tensorflow dataset list to contain image files
dsetFiles = tf.data.Dataset.list_files(GCS_PATTERN) # This also
```

```
shuffles the images

#RDD containing imagine filenames
#Converting from tensorflow to RDD
filenames_rdd = sc.parallelize(dsetFiles)



### iii) Sample the the RDD to a smaller number at an appropriate
position in the code. Specify a sampling factor of 0.02 for short
tests. (1%)

# Sample the RDD with a sampling factor of 0.02 (2%)

#Setting sampling factor to 0.02
sampling_factor = 0.02

#Sampling thr RDD
sampled_filenames_rdd = filenames_rdd.sample(False, sampling_factor)

#Then
# RDD decode for JPEG and label
decode_jpeg_and_label_rdd =
sampled_filenames_rdd.map(decode_jpeg_and_label)

# RDD decode for resize and crop
resize_and_crop_image_rdd =
decode_jpeg_and_label_rdd.map(resize_and_crop_image)

# Apply recompression function to each resized image
recompress_image_rdd = resize_and_crop_image_rdd.map(recompress_image)



### iv) Then use the functions from above to write the TFRecord files.
(3%)

#Output file name in google cloud
GCS_OUTPUT = BUCKET + '/tfrecords-jpeg-192x192-2/flowers'  # prefix
for output file names

# functions for writing TFRecord entries
# Feature values are always stored as lists, a single data element
will be a list of size 1


#Function for byte string feature
def _bytestring_feature(list_of_bytestrings):
  #Set to returning a tensorflow feature with list of byte strings
    return
tf.train.Feature(bytes_list=tf.train.BytesList(value=list_of_bytestrin
```

```
gs))

#Integer feature function
def _int_feature(list_of_ints): # int64
#Tensor flow feature with list of integers
    return
tf.train.Feature(int64_list=tf.train.Int64List(value=list_of_ints))

#TFRRecord entry function
def to_tfrecord(tfrec_filewriter, img_bytes, label): #Create tf data
records
    #Convert image bytes and label to TFRecord entry
    class_num = np.argmax(np.array(CLASSES)==label) # 'roses' => 2
(order defined in CLASSES)
    one_hot_class = np.eye(len(CLASSES))[class_num]     # [0, 0, 1, 0,
0] for class #2, roses
    feature = {
        "image": _bytestring_feature([img_bytes]), # one image in the
list
        "class": _int_feature([class_num]) #,        # one class in
the list
    }
    return
tf.train.Example(features=tf.train.Features(feature=feature))

#Function to write TFRecord files for each partition
def write_tfrecords(index,partition):
  #Update
  print("Writing TFRecords")
  #Setting the filename for the TFRecord file
  filename = GCS_OUTPUT + "{}.tfrec".format(index)
  with tf.io.TFRecordWriter(filename) as out_file:
    #Iterating over each element in the partition
    for element in partition:
      #Extracting image from partition
      image=element[0]
      #Extracting Label from partition
      label=element[1]
      #Image and label conerted to TFRecord entry
      example = to_tfrecord(out_file,
                            image.numpy(), # re-compressed image:
already a byte string
                            label.numpy()
                            )
      #Writing to tfRecord file
      out_file.write(example.SerializeToString())
      #Yield - generator function - iterating over a sequence
    yield [filename]
```

```
# v) The code for writing to the TFRecord files needs to be put into a
function, that can be applied to every partition with the
'RDD.mapPartitionsWithIndex' function.
#The return value of that function is not used here, but you should
return the filename, so that you have a list of the created TFRecord
files. (4%)

#Function to write TFRecord files for each partition
def write_tfrecords(index,partition):
  #Update
  print("Writing TFRecords")
  #Setting the filename for the TFRecord file
  filename = GCS_OUTPUT + "{}.tfrec".format(index)
  with tf.io.TFRecordWriter(filename) as out_file:
    #Iterating over each element in the partition
    for element in partition:
      #Extracting image
      image=element[0]
      #Extracting Label
      label=element[1]
      example = to_tfrecord(out_file,
                            image.numpy(), # re-compressed image:
already a byte string
                            label.numpy()
                            )
      out_file.write(example.SerializeToString())
      #Yield - generator function - iterating over a sequence
    yield [filename]


# Apply the write_tfrecord function to each partition of the RDD and
store it in a list
tfrecord_filenames =
recompress_image_rdd.mapPartitionsWithIndex(write_tfrecords).collect()

# Display the list of created TFRecord files
print(tfrecord_filenames)
```

```
[['gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/
flowers0.tfrec'], ['gs://bigdata-421920-storage/tfrecords-jpeg-
192x192-2/flowers1.tfrec']]
```

1b) Testing (3%)

i) Read from the TFRecord Dataset, using `load_dataset` and `display_9_images_from_dataset` to test.

```
### CODING TASK ###
```

```
#i) Read from the TFRecord Dataset, using load_dataset and
display_9_images_from_dataset to test.
# Define the read_tfrecord function to parse the TFRecord

#Function to display 9 images from the dataset
def display_9_images_from_dataset(dataset):
  plt.figure(figsize=(13,13))
  subplot=331
  #For loop to iterate over dataset and enumerate through the elements
  for i, (image, label) in enumerate(dataset):
    plt.subplot(subplot)
    plt.axis('off')
    #Converting tensor imagine to nump array
    plt.imshow(image.numpy().astype(np.uint8))
    plt.title(str(label.numpy()), fontsize=16)
    #Title of plot
    #plt.title(label.numpy().decode(), fontsize=16)
    #seeting a condition where the subplot index is incremented, until
there is 9 images displayed where the loop breaks
    subplot += 1
    if i==8:
      break
  plt.tight_layout()
  plt.subplots_adjust(wspace=0.1, hspace=0.1)
  plt.show()

#Function to parse TFRecord examples
def read_tfrecord(example):
    features = {
        "image": tf.io.FixedLenFeature([], tf.string),  # tf.string =
bytestring (not text string)
        "class": tf.io.FixedLenFeature([], tf.int64) #,   # shape []
means scalar
    }
    # decode the TFRecord
    # Parse a single TFRecord example using the specified features
    example = tf.io.parse_single_example(example, features)
    #Decoing image from TFRrecord example
    image = tf.image.decode_jpeg(example['image'], channels=3)
    #reshaping image to target size
    image = tf.reshape(image, [*TARGET_SIZE, 3])
    #Extracting class label
    class_num = example['class']
    #output will be parsed image and class label
    return image, class_num

#Function to load dataset from TFRecord files
def load_dataset(filenames):
    # read from TFRecords. For optimal performance, read from multiple
    # TFRecord files at once and set the option
```

```
experimental_deterministic = False
    # to allow order-altering optimizations.

    option_no_order = tf.data.Options()
    option_no_order.experimental_deterministic = False
    #dataset
    dataset = tf.data.TFRecordDataset(filenames)
    dataset = dataset.with_options(option_no_order)
    dataset = dataset.map(read_tfrecord)
    return dataset


#Loading dataset from Tfrecord files using defined functions
datasetTfrecRDD = load_dataset(tfrecord_filenames)
#Display images
display_9_images_from_dataset(datasetTfrecRDD)
```

ii) Write your code above into a file using the *cell magic* `%%writefile spark_write_tfrec.py` at the beginning of the file. Then, run the file locally in Spark.

```
### CODING TASK ###

#ii) Write your code above into a file using the cell magic %
%writefile spark_write_tfrec.py at the beginning of the file.
#Then, run the file locally in Spark.

%%writefile spark_write_tfrec.py
```

```
#Import the nessesary libraries
import os, sys, math
import os
os.environ['PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION'] = 'python'
import numpy as np
import time
import string
import random
import tensorflow as tf
print("Tensorflow version " + tf.__version__)
import pickle
import pyspark
from pyspark.sql import SQLContext
from pyspark.sql import Row
print(pyspark.__version__)

sc = pyspark.SparkContext.getOrCreate()
print(sc)

#variables required

PROJECT = 'bigdata-421920' #Project ID
BUCKET = 'gs://{}-storage'.format(PROJECT) # bucket storage
REGION = 'us-central1'
CLUSTER = '{}-cluster'.format(PROJECT)
GCS_PATTERN = 'gs://flowers-public/*/*.jpg' # glob  pattern for input
files
PARTITIONS = 16 # no of partitions we will use later
GCS_OUTPUT = BUCKET + '/tfrecords-jpeg-192x192-2/flowers'
CLASSES = [b'daisy', b'dandelion', b'roses', b'sunflowers', b'tulips']
TARGET_SIZE = [192, 192] # target resolution for the images

# Question 1.A.I (Part1)

#Function 1
#Function to decode JPEG image and extract label from the filepath
def decode_jpeg_and_label(filepath):
    # extracts the image data and creates a class label, based on the
filepath
    bits = tf.io.read_file(filepath)
    image = tf.image.decode_jpeg(bits)
    # parse flower name from containing directory
    label = tf.strings.split(tf.expand_dims(filepath, axis=-1),
sep='/')
    label2 = label.values[-2]
    return image, label2

#Function 2
#Function to resize and crop
def resize_and_crop_image(data):
```

```
    # Resizes and cropd using "fill" algorithm:
    # always make sure the resulting image is cut out from the source
image
    # so that it fills the TARGET_SIZE entirely with no black bars
    # and a preserved aspect ratio.
    #Obtain image and label from the data
    image, label = data
    #Dimension of image
    w = tf.shape(image)[0]
    h = tf.shape(image)[1]
    #Target size
    tw = TARGET_SIZE[1]
    th = TARGET_SIZE[0]
    #Resizing criteria calculated
    resize_crit = (w * th) / (h * tw)
    #Here the image gets resized according to the rezising criteria
    image = tf.cond(resize_crit < 1,
                    lambda: tf.image.resize(image, [w*tw/w, h*tw/w]),
# if true
                    lambda: tf.image.resize(image, [w*th/h, h*th/h])
# if false
                    )
    #New dimensions of image calculated
    nw = tf.shape(image)[0]
    nh = tf.shape(image)[1]
    #Image gets cropped to the target size
    image = tf.image.crop_to_bounding_box(image, (nw - tw) // 2, (nh -
th) // 2, tw, th)
    return image, label

#Function 3
#Function to recompress the image
def recompress_image(data):
    # this reduces the amount of data, but takes some time
    #Obtain image and label from the data
    image, label = data
    #Image is 'casted' to an 8 bit integer format
    image = tf.cast(image, tf.uint8)
    #Image gets encoded to jep format
    image = tf.image.encode_jpeg(image, optimize_size=True,
chroma_downsampling=False)
    return image, label

## Question 1.A.II (Part2)
###ii) Replace the TensorFlow Dataset objects with RDDs, starting with
an RDD that contains the list of image filenames. (3%)

#Filenames for the specified pattern
filenames = tf.io.gfile.glob(GCS_PATTERN)
```

```
#RDD for files
filenames_rdd = sc.parallelize(filenames)



## Question 1.A.III (Part3)


#RDD files are sampled to a test size
sampled_filenames_rdd = filenames_rdd.sample(False, 0.02)

#Then
# RDD decode for JPEG and label
decode_jpeg_and_label_rdd =
sampled_filenames_rdd.map(decode_jpeg_and_label)

# RDD decode for resize and crop
resize_and_crop_image_rdd =
decode_jpeg_and_label_rdd.map(resize_and_crop_image)

# Apply recompression function to each resized image
recompress_image_rdd = resize_and_crop_image_rdd.map(recompress_image)



## Question 1.A.IV (Part4)


#Function 4
#Function for byte string feature
def _bytestring_feature(list_of_bytestrings):
  #Set to returning a tensorflow feature with list of byte strings
    return
tf.train.Feature(bytes_list=tf.train.BytesList(value=list_of_bytestrin
gs))

#Function 5
#Integer feature function
def _int_feature(list_of_ints): # int64
#Tensor flow feature with list of integers
    return
tf.train.Feature(int64_list=tf.train.Int64List(value=list_of_ints))

#Function 6
#TFRRecord entry function
def to_tfrecord(tfrec_filewriter, img_bytes, label): #Create tf data
records
    #Convert image bytes and label to TFRecord entry
    class_num = np.argmax(np.array(CLASSES)==label) # 'roses' => 2
(order defined in CLASSES)
    one_hot_class = np.eye(len(CLASSES))[class_num]     # [0, 0, 1, 0,
```

```
0] for class #2, roses
    feature = {
        "image": _bytestring_feature([img_bytes]), # one image in the
list
        "class": _int_feature([class_num]) #,        # one class in
the list
    }
    return
tf.train.Example(features=tf.train.Features(feature=feature))



## Question 1.A.V (Part5)
# v) The code for writing to the TFRecord files needs to be put into a
function, that can be applied to every partition with the
'RDD.mapPartitionsWithIndex' function.
#The return value of that function is not used here, but you should
return the filename, so that you have a list of the created TFRecord
files. (4%)


#Function 7
#Function to write TFRecord files for each partition
def write_tfrecords(index,partition):
  #Update
  print("Writing TFRecords")
  #Setting the filename for the TFRecord file
  filename = GCS_OUTPUT + "{}.tfrec".format(index)
  with tf.io.TFRecordWriter(filename) as out_file:
    #Iterating over each element in the partition
    for element in partition:
      #Extracting image from partition
      image=element[0]
      #Extracting Label from partition
      label=element[1]
      #Image and label conerted to TFRecord entry
      example = to_tfrecord(out_file,
                            image.numpy(), # re-compressed image:
already a byte string
                            label.numpy()
                            )
      #Writing to tfRecord file
      out_file.write(example.SerializeToString())
      #Yield - generator function - iterating over a sequence
    yield [filename]


# Apply the write_tfrecord function to each partition of the RDD
tfrecord_filenames =
recompress_image_rdd.mapPartitionsWithIndex(write_tfrecords)
```

```
Overwriting spark_write_tfrec.py
```

```
%run spark_write_tfrec.py
#Running the spark script locally

Tensorflow version 2.15.0
3.5.0
<SparkContext master=local[*] appName=pyspark-shell>

<Figure size 640x480 with 0 Axes>
```

1c) Set up a cluster and run the script. (6%)

Following the example from the labs, set up a cluster to run PySpark jobs in the cloud. You need to set up so that TensorFlow is installed on all nodes in the cluster.

i) Single machine cluster

Set up a cluster with a single machine using the maximal SSD size (100) and 8 vCPUs.

Enable **package installation** by passing a flag `--initialization-actions` with argument `gs://goog-dataproc-initialization-actions-$REGION/python/pip-install.sh` (this is a public script that will read metadata to determine which packages to install). Then, the **packages are specified** by providing a `--metadata` flag with the argument `PIP_PACKAGES=tensorflow==2.4.0`.

Note: consider using `PIP_PACKAGES="tensorflow numpy"` or `PIP_PACKAGES=tensorflow` in case an older version of tensorflow is causing issues.

When the cluster is running, run your script to check that it works and keep the output cell output. (3%)

```
### CODING TASK ###
#CLuster 1
#Creating cluster with single machine 8vCPUs, 100gb ssd - single node

#Command to create a Dataproc cluster
!gcloud dataproc clusters create $CLUSTER \
  --image-version 1.5-ubuntu18 \
```

```
  --single-node \
  --master-machine-type n1-standard-8 \
  --master-boot-disk-type pd-ssd --master-boot-disk-size 100 \
  --initialization-actions gs://goog-dataproc-initialization-actions-
$REGION/python/pip-install.sh \
  --metadata "PIP_PACKAGES=tensorflow==2.4.0 scipy==1.4.1 pandas
numpy==1.21.6 matplotlib" \
  --zone us-central1-c



Waiting on operation
[projects/bigdata-421920/regions/us-central1/operations/17ac5e13-d25c-
37d6-a718-858b04f68712].

WARNING: Consider using Auto Zone rather than selecting a zone
manually. See
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/
auto-zone
WARNING: Don't create production clusters that reference
initialization actions located in the gs://goog-dataproc-
initialization-actions-REGION public buckets. These scripts are
provided as reference implementations, and they are synchronized with
ongoing GitHub repository changes—a new version of a initialization
action in public buckets may break your cluster creation. Instead,
copy the following initialization actions from public buckets into
your bucket :
gs://goog-dataproc-initialization-actions-us-central1/python/pip-
install.sh
WARNING: Permissions are missing for the default service account
'504478955913-compute@developer.gserviceaccount.com', missing
permissions: [storage.buckets.get, storage.objects.create,
storage.objects.delete, storage.objects.get, storage.objects.list,
storage.objects.update] on the project 'projects/bigdata-421920'. This
usually happens when a custom resource (ex: custom staging bucket) or
a user-managed VM Service account has been provided and the
default/user-managed service account hasn't been granted enough
permissions on the resource. See
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/
service-accounts#VM_service_account.
WARNING: The firewall rules for specified network or subnetwork would
allow ingress traffic from 0.0.0.0/0, which could be a security risk.
WARNING: Unable to validate the staging bucket lifecycle configuration
of the bucket 'dataproc-staging-us-central1-504478955913-tcjfqtey' due
to an internal error, Please make sure that the provided bucket
doesn't have any delete rules set.
Created
[https://dataproc.googleapis.com/v1/projects/bigdata-421920/regions/
us-central1/clusters/bigdata-421920-cluster] Cluster placed in zone
[us-central1-c].
```

```
### CODING TASK###
# get information of cluster created cluster
!gcloud dataproc clusters describe $CLUSTER

clusterName: bigdata-421920-cluster
clusterUuid: cb512dbe-e541-4b59-ae6a-3678a913c366
config:
  configBucket: dataproc-staging-us-central1-504478955913-tcjfqtey
  endpointConfig: {}
  gceClusterConfig:
    internalIpOnly: false
    metadata:
      PIP_PACKAGES: tensorflow==2.4.0 scipy==1.4.1 pandas
numpy==1.21.6 matplotlib
    networkUri:
https://www.googleapis.com/compute/v1/projects/bigdata-421920/global/
networks/default
    serviceAccountScopes:
    - https://www.googleapis.com/auth/bigquery
    - https://www.googleapis.com/auth/bigtable.admin.table
    - https://www.googleapis.com/auth/bigtable.data
    - https://www.googleapis.com/auth/cloud.useraccounts.readonly
    - https://www.googleapis.com/auth/devstorage.full_control
    - https://www.googleapis.com/auth/devstorage.read_write
    - https://www.googleapis.com/auth/logging.write
    - https://www.googleapis.com/auth/monitoring.write
    zoneUri: https://www.googleapis.com/compute/v1/projects/bigdata-
421920/zones/us-central1-c
  initializationActions:
  - executableFile: gs://goog-dataproc-initialization-actions-us-
central1/python/pip-install.sh
    executionTimeout: 600s
  masterConfig:
    diskConfig:
      bootDiskSizeGb: 100
      bootDiskType: pd-ssd
    imageUri: https://www.googleapis.com/compute/v1/projects/cloud-
dataproc/global/images/dataproc-1-5-ubu18-20230909-165100-rc01
    instanceNames:
    - bigdata-421920-cluster-m
    machineTypeUri:
https://www.googleapis.com/compute/v1/projects/bigdata-421920/zones/
us-central1-c/machineTypes/n1-standard-8
    minCpuPlatform: AUTOMATIC
    numInstances: 1
    preemptibility: NON_PREEMPTIBLE
  softwareConfig:
    imageVersion: 1.5.90-ubuntu18
    properties:
      capacity-
```

```
scheduler:yarn.scheduler.capacity.root.default.ordering-policy: fair
      core:fs.gs.block.size: '134217728'
      core:fs.gs.metadata.cache.enable: 'false'
      core:hadoop.ssl.enabled.protocols: TLSv1,TLSv1.1,TLSv1.2
      dataproc:dataproc.allow.zero.workers: 'true'
      distcp:mapreduce.map.java.opts: -Xmx768m
      distcp:mapreduce.map.memory.mb: '1024'
      distcp:mapreduce.reduce.java.opts: -Xmx768m
      distcp:mapreduce.reduce.memory.mb: '1024'
      hdfs:dfs.datanode.address: 0.0.0.0:9866
      hdfs:dfs.datanode.http.address: 0.0.0.0:9864
      hdfs:dfs.datanode.https.address: 0.0.0.0:9865
      hdfs:dfs.datanode.ipc.address: 0.0.0.0:9867
      hdfs:dfs.namenode.handler.count: '20'
      hdfs:dfs.namenode.http-address: 0.0.0.0:9870
      hdfs:dfs.namenode.https-address: 0.0.0.0:9871
      hdfs:dfs.namenode.lifeline.rpc-address: bigdata-421920-cluster-
m:8050
      hdfs:dfs.namenode.secondary.http-address: 0.0.0.0:9868
      hdfs:dfs.namenode.secondary.https-address: 0.0.0.0:9869
      hdfs:dfs.namenode.service.handler.count: '10'
      hdfs:dfs.namenode.servicerpc-address: bigdata-421920-cluster-
m:8051
      hive:hive.fetch.task.conversion: none
      mapred-env:HADOOP_JOB_HISTORYSERVER_HEAPSIZE: '4000'
      mapred:mapreduce.job.maps: '21'
      mapred:mapreduce.job.reduce.slowstart.completedmaps: '0.95'
      mapred:mapreduce.job.reduces: '7'
      mapred:mapreduce.jobhistory.recovery.store.class:
org.apache.hadoop.mapreduce.v2.hs.HistoryServerLeveldbStateStoreServic
e
      mapred:mapreduce.map.cpu.vcores: '1'
      mapred:mapreduce.map.java.opts: -Xmx2457m
      mapred:mapreduce.map.memory.mb: '3072'
      mapred:mapreduce.reduce.cpu.vcores: '1'
      mapred:mapreduce.reduce.java.opts: -Xmx2457m
      mapred:mapreduce.reduce.memory.mb: '3072'
      mapred:mapreduce.task.io.sort.mb: '256'
      mapred:yarn.app.mapreduce.am.command-opts: -Xmx2457m
      mapred:yarn.app.mapreduce.am.resource.cpu-vcores: '1'
      mapred:yarn.app.mapreduce.am.resource.mb: '3072'
      spark-env:SPARK_DAEMON_MEMORY: 4000m
      spark:spark.driver.maxResultSize: 3840m
      spark:spark.driver.memory: 7680m
      spark:spark.executor.cores: '4'
      spark:spark.executor.instances: '2'
      spark:spark.executor.memory: 11171m
      spark:spark.executorEnv.OPENBLAS_NUM_THREADS: '1'
      spark:spark.extraListeners:
```

```
com.google.cloud.spark.performance.DataprocMetricsListener
      spark:spark.scheduler.mode: FAIR
      spark:spark.sql.cbo.enabled: 'true'
      spark:spark.ui.port: '0'
      spark:spark.yarn.am.memory: 640m
      yarn-env:YARN_NODEMANAGER_HEAPSIZE: '4000'
      yarn-env:YARN_RESOURCEMANAGER_HEAPSIZE: '4000'
      yarn-env:YARN_TIMELINESERVER_HEAPSIZE: '4000'
      yarn:yarn.nodemanager.address: 0.0.0.0:8026
      yarn:yarn.nodemanager.resource.cpu-vcores: '8'
      yarn:yarn.nodemanager.resource.memory-mb: '24576'
      yarn:yarn.resourcemanager.nodemanager-graceful-decommission-
timeout-secs: '86400'
      yarn:yarn.scheduler.maximum-allocation-mb: '24576'
      yarn:yarn.scheduler.minimum-allocation-mb: '1024'
  tempBucket: dataproc-temp-us-central1-504478955913-s3uruvrj
labels:
  goog-dataproc-cluster-name: bigdata-421920-cluster
  goog-dataproc-cluster-uuid: cb512dbe-e541-4b59-ae6a-3678a913c366
  goog-dataproc-location: us-central1
projectId: bigdata-421920
status:
  state: RUNNING
  stateStartTime: '2024-05-05T06:02:58.011135Z'
statusHistory:
- state: CREATING
  stateStartTime: '2024-05-05T05:59:45.410015Z'
```

Run the script in the cloud and test the output.

```
### CODING TASK ###

#The saved python script is then run on the cluster created
!gcloud dataproc jobs submit pyspark --cluster $CLUSTER \
spark_write_tfrec.py
%time

Job [b2a8197c86cd4bf9b47d375540a74feb] submitted.
Waiting for job output...
2024-05-05 06:03:10.397160: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
not load dynamic library 'libcudart.so.11.0'; dlerror:
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 06:03:10.397203: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
Tensorflow version 2.4.0
2.4.8
```

```
24/05/05 06:03:14 INFO org.apache.spark.SparkEnv: Registering
MapOutputTracker
24/05/05 06:03:14 INFO org.apache.spark.SparkEnv: Registering
BlockManagerMaster
24/05/05 06:03:14 INFO org.apache.spark.SparkEnv: Registering
OutputCommitCoordinator
24/05/05 06:03:14 INFO org.spark_project.jetty.util.log: Logging
initialized @6068ms to org.spark_project.jetty.util.log.Slf4jLog
24/05/05 06:03:14 INFO org.spark_project.jetty.server.Server: jetty-
9.4.z-SNAPSHOT; built: unknown; git: unknown; jvm 1.8.0_382-b05
24/05/05 06:03:14 INFO org.spark_project.jetty.server.Server: Started
@6204ms
24/05/05 06:03:14 INFO
org.spark_project.jetty.server.AbstractConnector: Started
ServerConnector@55bbd3d1{HTTP/1.1, (http/1.1)}{0.0.0.0:42781}
24/05/05 06:03:15 INFO org.apache.hadoop.yarn.client.RMProxy:
Connecting to ResourceManager at
bigdata-421920-cluster-m/10.128.15.225:8032
24/05/05 06:03:16 INFO org.apache.hadoop.yarn.client.AHSProxy:
Connecting to Application History server at
bigdata-421920-cluster-m/10.128.15.225:10200
24/05/05 06:03:16 INFO org.apache.hadoop.conf.Configuration: resource-
types.xml not found
24/05/05 06:03:16 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find
'resource-types.xml'.
24/05/05 06:03:16 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = memory-mb, units = Mi, type = COUNTABLE
24/05/05 06:03:16 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = vcores, units = , type = COUNTABLE
24/05/05 06:03:19 INFO
org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted
application application_1714888875884_0001
<SparkContext master=yarn appName=spark_write_tfrec.py>
24/05/05 06:03:31 INFO
org.spark_project.jetty.server.AbstractConnector: Stopped
Spark@55bbd3d1{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
Job [b2a8197c86cd4bf9b47d375540a74feb] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/cb512dbe-e541-4b59-ae6a-
3678a913c366/jobs/b2a8197c86cd4bf9b47d375540a74feb/
driverOutputResourceUri: gs://dataproc-staging-us-central1-
504478955913-tcjfqtey/google-cloud-dataproc-metainfo/cb512dbe-e541-
4b59-ae6a-3678a913c366/jobs/b2a8197c86cd4bf9b47d375540a74feb/
driveroutput
jobUuid: 8818834d-515f-33f8-afde-45de881888b7
```

```
placement:
  clusterName: bigdata-421920-cluster
  clusterUuid: cb512dbe-e541-4b59-ae6a-3678a913c366
pysparkJob:
  mainPythonFileUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/cb512dbe-e541-4b59-ae6a-
3678a913c366/jobs/b2a8197c86cd4bf9b47d375540a74feb/staging/
spark_write_tfrec.py
reference:
  jobId: b2a8197c86cd4bf9b47d375540a74feb
  projectId: bigdata-421920
status:
  state: DONE
  stateStartTime: '2024-05-05T06:03:35.468582Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-05-05T06:03:06.500228Z'
- state: SETUP_DONE
  stateStartTime: '2024-05-05T06:03:06.532844Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-05-05T06:03:06.858970Z'
yarnApplications:
- name: spark_write_tfrec.py
  progress: 1.0
  state: FINISHED
  trackingUrl:
http://bigdata-421920-cluster-m:8088/proxy/application_1714888875884_0
001/
CPU times: user 5 µs, sys: 1e+03 ns, total: 6 µs
Wall time: 10 µs
```

In the free credit tier on Google Cloud, there are normally the following **restrictions** on compute machines:

- max 100GB of *SSD persistent disk*
- max 2000GB of *standard persistent disk*
- max 8 *vCPU*s
- no GPUs

See here for details The **disks are virtual** disks, where **I/O speed is limited in proportion to the size**, so we should allocate them evenly. This has mainly an effect on the **time the cluster needs to start**, as we are reading the data mainly from the bucket and we are not writing much to disk at all.

ii) Maximal cluster

Use the **largest possible cluster** within these constraints, i.e. **1 master and 7 worker nodes**. Each of them with 1 (virtual) CPU. The master should get the full *SSD* capacity and the 7 worker nodes should get equal shares of the *standard* disk capacity to maximise throughput.

Once the cluster is running, test your script. (3%)

```
### CODING TASK ###
#Creating cluster with 1 master node and 7 worker nodes with only one
virtual CPU
!gcloud dataproc clusters create bigdata-maxcluster \
    --image-version 1.5-ubuntu18 \
    --master-machine-type n1-standard-1 \
    --master-boot-disk-type pd-ssd \
    --master-boot-disk-size 100 \
    --num-workers 7 \
    --worker-machine-type n1-standard-1 --worker-boot-disk-size 100 \
    --metadata "PIP_PACKAGES=tensorflow==2.4.0 scipy==1.4.1 pandas
numpy==1.21.6 matplotlib"  \
    --zone us-central1-c

ERROR: (gcloud.dataproc.clusters.create) INVALID_ARGUMENT:
Insufficient 'IN_USE_ADDRESSES' quota. Requested 8.0, available 7.0.
Your resource request exceeds your available quota. See
https://cloud.google.com/compute/resource-usage. Use
https://cloud.google.com/docs/quotas/view-
manage#requesting_higher_quota to request additional quota.


#There was insufficient resources when doing 7 nodes hence we need to
reduce the number of nodes we will attempt with 3 nodes

### CODING TASK ###
#Reattempting Cluster with 1vCPUs - trying 3 worker nodes
#Cluster 2
!gcloud dataproc clusters create bigdata-maxcluster \
    --image-version 1.5-ubuntu18 \
    --master-machine-type n1-standard-1 \
    --master-boot-disk-type pd-ssd \
    --master-boot-disk-size 100GB \
    --num-workers 3 \
    --worker-machine-type n1-standard-1 \
    --worker-boot-disk-size 100GB \
    --initialization-actions gs://goog-dataproc-initialization-
actions-$REGION/python/pip-install.sh \
    --metadata "PIP_PACKAGES=tensorflow==2.4.0 scipy==1.4.1 pandas
numpy==1.21.6 matplotlib"  \
    --zone us-central1-c

Waiting on operation
[projects/bigdata-421920/regions/us-central1/operations/5feb1e67-75e7-
37bf-9d48-7423d7928fe1].

WARNING: Consider using Auto Zone rather than selecting a zone
manually. See
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/
```

```
auto-zone
WARNING: Creating clusters using the n1-standard-1 machine type is not recommended. Consider using a machine type with higher memory.
WARNING: Don't create production clusters that reference initialization actions located in the gs://goog-dataproc-initialization-actions-REGION public buckets. These scripts are provided as reference implementations, and they are synchronized with ongoing GitHub repository changes—a new version of a initialization action in public buckets may break your cluster creation. Instead, copy the following initialization actions from public buckets into your bucket : gs://goog-dataproc-initialization-actions-us-central1/python/pip-install.sh
WARNING: For PD-Standard without local SSDs, we strongly recommend provisioning 1TB or larger to ensure consistently high I/O performance. See https://cloud.google.com/compute/docs/disks/performance for information on disk I/O performance.
WARNING: Permissions are missing for the default service account '504478955913-compute@developer.gserviceaccount.com', missing permissions: [storage.objects.get, storage.objects.update] on the staging_bucket 'projects/_/buckets/dataproc-staging-us-central1-504478955913-tcjfqtey'. This usually happens when a custom resource (ex: custom staging bucket) or a user-managed VM Service account has been provided and the default/user-managed service account hasn't been granted enough permissions on the resource. See https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/service-accounts#VM_service_account.
WARNING: Permissions are missing for the default service account '504478955913-compute@developer.gserviceaccount.com', missing permissions: [storage.objects.get, storage.objects.update] on the temp_bucket 'projects/_/buckets/dataproc-temp-us-central1-504478955913-s3uruvrj'. This usually happens when a custom resource (ex: custom staging bucket) or a user-managed VM Service account has been provided and the default/user-managed service account hasn't been granted enough permissions on the resource. See https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/service-accounts#VM_service_account.
WARNING: The firewall rules for specified network or subnetwork would allow ingress traffic from 0.0.0.0/0, which could be a security risk.
WARNING: The specified custom staging bucket 'dataproc-staging-us-central1-504478955913-tcjfqtey' is not using uniform bucket level access IAM configuration. It is recommended to update bucket to enable the same. See https://cloud.google.com/storage/docs/uniform-bucket-level-access.
Created [https://dataproc.googleapis.com/v1/projects/bigdata-421920/regions/us-central1/clusters/bigdata-maxcluster] Cluster placed in zone [us-central1-c].
```

```
### CODING TASK ###
# get information of the maximal machine cluster with 3 worker nodes
!gcloud dataproc clusters describe bigdata-maxcluster

clusterName: bigdata-maxcluster
clusterUuid: 35052232-f9c1-49b1-ae63-a695f2252802
config:
  configBucket: dataproc-staging-us-central1-504478955913-tcjfqtey
  endpointConfig: {}
  gceClusterConfig:
    internalIpOnly: false
    metadata:
      PIP_PACKAGES: tensorflow==2.4.0 scipy==1.4.1 pandas
numpy==1.21.6 matplotlib
    networkUri:
https://www.googleapis.com/compute/v1/projects/bigdata-421920/global/
networks/default
    serviceAccountScopes:
    - https://www.googleapis.com/auth/bigquery
    - https://www.googleapis.com/auth/bigtable.admin.table
    - https://www.googleapis.com/auth/bigtable.data
    - https://www.googleapis.com/auth/cloud.useraccounts.readonly
    - https://www.googleapis.com/auth/devstorage.full_control
    - https://www.googleapis.com/auth/devstorage.read_write
    - https://www.googleapis.com/auth/logging.write
    - https://www.googleapis.com/auth/monitoring.write
    zoneUri: https://www.googleapis.com/compute/v1/projects/bigdata-
421920/zones/us-central1-c
  initializationActions:
  - executableFile: gs://goog-dataproc-initialization-actions-us-
central1/python/pip-install.sh
    executionTimeout: 600s
  masterConfig:
    diskConfig:
      bootDiskSizeGb: 100
      bootDiskType: pd-ssd
    imageUri: https://www.googleapis.com/compute/v1/projects/cloud-
dataproc/global/images/dataproc-1-5-ubu18-20230909-165100-rc01
    instanceNames:
    - bigdata-maxcluster-m
    machineTypeUri:
https://www.googleapis.com/compute/v1/projects/bigdata-421920/zones/
us-central1-c/machineTypes/n1-standard-1
    minCpuPlatform: AUTOMATIC
    numInstances: 1
    preemptibility: NON_PREEMPTIBLE
  softwareConfig:
    imageVersion: 1.5.90-ubuntu18
    properties:
      capacity-
```

```
scheduler:yarn.scheduler.capacity.root.default.ordering-policy: fair
      core:fs.gs.block.size: '134217728'
      core:fs.gs.metadata.cache.enable: 'false'
      core:hadoop.ssl.enabled.protocols: TLSv1,TLSv1.1,TLSv1.2
      distcp:mapreduce.map.java.opts: -Xmx576m
      distcp:mapreduce.map.memory.mb: '768'
      distcp:mapreduce.reduce.java.opts: -Xmx576m
      distcp:mapreduce.reduce.memory.mb: '768'
      hdfs:dfs.datanode.address: 0.0.0.0:9866
      hdfs:dfs.datanode.http.address: 0.0.0.0:9864
      hdfs:dfs.datanode.https.address: 0.0.0.0:9865
      hdfs:dfs.datanode.ipc.address: 0.0.0.0:9867
      hdfs:dfs.namenode.handler.count: '40'
      hdfs:dfs.namenode.http-address: 0.0.0.0:9870
      hdfs:dfs.namenode.https-address: 0.0.0.0:9871
      hdfs:dfs.namenode.lifeline.rpc-address: bigdata-maxcluster-
m:8050
      hdfs:dfs.namenode.secondary.http-address: 0.0.0.0:9868
      hdfs:dfs.namenode.secondary.https-address: 0.0.0.0:9869
      hdfs:dfs.namenode.service.handler.count: '20'
      hdfs:dfs.namenode.servicerpc-address: bigdata-maxcluster-m:8051
      hive:hive.fetch.task.conversion: none
      mapred-env:HADOOP_JOB_HISTORYSERVER_HEAPSIZE: '1000'
      mapred:mapreduce.job.maps: '24'
      mapred:mapreduce.job.reduce.slowstart.completedmaps: '0.95'
      mapred:mapreduce.job.reduces: '3'
      mapred:mapreduce.jobhistory.recovery.store.class:
org.apache.hadoop.mapreduce.v2.hs.HistoryServerLeveldbStateStoreServic
e
      mapred:mapreduce.map.cpu.vcores: '1'
      mapred:mapreduce.map.java.opts: -Xmx819m
      mapred:mapreduce.map.memory.mb: '1024'
      mapred:mapreduce.reduce.cpu.vcores: '1'
      mapred:mapreduce.reduce.java.opts: -Xmx1638m
      mapred:mapreduce.reduce.memory.mb: '2048'
      mapred:mapreduce.task.io.sort.mb: '256'
      mapred:yarn.app.mapreduce.am.command-opts: -Xmx819m
      mapred:yarn.app.mapreduce.am.resource.cpu-vcores: '1'
      mapred:yarn.app.mapreduce.am.resource.mb: '1024'
      spark-env:SPARK_DAEMON_MEMORY: 1000m
      spark:spark.driver.maxResultSize: 480m
      spark:spark.driver.memory: 960m
      spark:spark.executor.cores: '1'
      spark:spark.executor.instances: '2'
      spark:spark.executor.memory: 2688m
      spark:spark.executorEnv.OPENBLAS_NUM_THREADS: '1'
      spark:spark.extraListeners:
com.google.cloud.spark.performance.DataprocMetricsListener
      spark:spark.scheduler.mode: FAIR
```

```
      spark:spark.sql.cbo.enabled: 'true'
      spark:spark.ui.port: '0'
      spark:spark.yarn.am.memory: 640m
      yarn-env:YARN_NODEMANAGER_HEAPSIZE: '1000'
      yarn-env:YARN_RESOURCEMANAGER_HEAPSIZE: '1000'
      yarn-env:YARN_TIMELINESERVER_HEAPSIZE: '1000'
      yarn:yarn.nodemanager.address: 0.0.0.0:8026
      yarn:yarn.nodemanager.resource.cpu-vcores: '1'
      yarn:yarn.nodemanager.resource.memory-mb: '3072'
      yarn:yarn.resourcemanager.nodemanager-graceful-decommission-
timeout-secs: '86400'
      yarn:yarn.scheduler.maximum-allocation-mb: '3072'
      yarn:yarn.scheduler.minimum-allocation-mb: '256'
  tempBucket: dataproc-temp-us-central1-504478955913-s3uruvrj
  workerConfig:
    diskConfig:
      bootDiskSizeGb: 100
      bootDiskType: pd-standard
    imageUri: https://www.googleapis.com/compute/v1/projects/cloud-
dataproc/global/images/dataproc-1-5-ubu18-20230909-165100-rc01
    instanceNames:
    - bigdata-maxcluster-w-0
    - bigdata-maxcluster-w-1
    - bigdata-maxcluster-w-2
    machineTypeUri:
https://www.googleapis.com/compute/v1/projects/bigdata-421920/zones/
us-central1-c/machineTypes/n1-standard-1
    minCpuPlatform: AUTOMATIC
    numInstances: 3
    preemptibility: NON_PREEMPTIBLE
labels:
  goog-dataproc-cluster-name: bigdata-maxcluster
  goog-dataproc-cluster-uuid: 35052232-f9c1-49b1-ae63-a695f2252802
  goog-dataproc-location: us-central1
projectId: bigdata-421920
status:
  state: RUNNING
  stateStartTime: '2024-05-05T06:09:39.142613Z'
statusHistory:
- state: CREATING
  stateStartTime: '2024-05-05T06:03:47.255020Z'

###CODING TASK###
#The saved python script is then run on the max worker node cluster
created called bigdata-macxluster
!gcloud dataproc jobs submit pyspark --cluster bigdata-maxcluster \
spark_write_tfrec.py
%time
```

```
Job [057b4fcf14e14f728b2abf53157f6f99] submitted.
Waiting for job output...
2024-05-05 06:09:54.787136: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
not load dynamic library 'libcudart.so.11.0'; dlerror:
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 06:09:54.787301: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
Tensorflow version 2.4.0
2.4.8
24/05/05 06:09:59 INFO org.apache.spark.SparkEnv: Registering
MapOutputTracker
24/05/05 06:09:59 INFO org.apache.spark.SparkEnv: Registering
BlockManagerMaster
24/05/05 06:09:59 INFO org.apache.spark.SparkEnv: Registering
OutputCommitCoordinator
24/05/05 06:10:00 INFO org.spark_project.jetty.util.log: Logging
initialized @10393ms to org.spark_project.jetty.util.log.Slf4jLog
24/05/05 06:10:00 INFO org.spark_project.jetty.server.Server: jetty-
9.4.z-SNAPSHOT; built: unknown; git: unknown; jvm 1.8.0_382-b05
24/05/05 06:10:00 INFO org.spark_project.jetty.server.Server: Started
@10670ms
24/05/05 06:10:00 INFO
org.spark_project.jetty.server.AbstractConnector: Started
ServerConnector@51f17779{HTTP/1.1, (http/1.1)}{0.0.0.0:38655}
24/05/05 06:10:02 INFO org.apache.hadoop.yarn.client.RMProxy:
Connecting to ResourceManager at
bigdata-maxcluster-m/10.128.15.228:8032
24/05/05 06:10:03 INFO org.apache.hadoop.yarn.client.AHSProxy:
Connecting to Application History server at
bigdata-maxcluster-m/10.128.15.228:10200
24/05/05 06:10:03 INFO org.apache.hadoop.conf.Configuration: resource-
types.xml not found
24/05/05 06:10:03 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find
'resource-types.xml'.
24/05/05 06:10:03 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = memory-mb, units = Mi, type = COUNTABLE
24/05/05 06:10:03 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = vcores, units = , type = COUNTABLE
24/05/05 06:10:08 INFO
org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted
application application_1714889193247_0001
<SparkContext master=yarn appName=spark_write_tfrec.py>
24/05/05 06:10:29 INFO
org.spark_project.jetty.server.AbstractConnector: Stopped
```

```
Spark@51f17779{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
Job [057b4fcf14e14f728b2abf53157f6f99] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/35052232-f9c1-49b1-ae63-
a695f2252802/jobs/057b4fcf14e14f728b2abf53157f6f99/
driverOutputResourceUri: gs://dataproc-staging-us-central1-
504478955913-tcjfqtey/google-cloud-dataproc-metainfo/35052232-f9c1-
49b1-ae63-a695f2252802/jobs/057b4fcf14e14f728b2abf53157f6f99/
driveroutput
jobUuid: 73591559-b643-3a6b-af81-df1efc008943
placement:
  clusterName: bigdata-maxcluster
  clusterUuid: 35052232-f9c1-49b1-ae63-a695f2252802
pysparkJob:
  mainPythonFileUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/35052232-f9c1-49b1-ae63-
a695f2252802/jobs/057b4fcf14e14f728b2abf53157f6f99/staging/
spark_write_tfrec.py
reference:
  jobId: 057b4fcf14e14f728b2abf53157f6f99
  projectId: bigdata-421920
status:
  state: DONE
  stateStartTime: '2024-05-05T06:10:33.255418Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-05-05T06:09:46.790844Z'
- state: SETUP_DONE
  stateStartTime: '2024-05-05T06:09:46.817420Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-05-05T06:09:47.169238Z'
yarnApplications:
- name: spark_write_tfrec.py
  progress: 1.0
  state: FINISHED
  trackingUrl:
http://bigdata-maxcluster-m:8088/proxy/application_1714889193247_0001/
CPU times: user 4 µs, sys: 1 µs, total: 5 µs
Wall time: 9.06 µs
```

1d) Optimisation, experiments, and discussion (17%)

i) Improve parallelisation

If you implemented a straightfoward version, you will **probably** observe that **all the computation** is done on only **two nodes**. This can be adressed by using the **second parameter** in the initial call to **parallelize**. Make the **suitable change** in the code you have written above and mark it up in comments as `### TASK 1d ###`.

Demonstrate the difference in cluster utilisation before and after the change based on different parameter values with **screenshots from Google Cloud** and measure the **difference in the processing time**. (6%)

ii) Experiment with cluster configurations.

In addition to the experiments above (using 8 VMs),test your program with 4 machines with double the resources each (2 vCPUs, memory, disk) and 1 machine with eightfold resources. Discuss the results in terms of disk I/O and network bandwidth allocation in the cloud. (7%)

iii) Explain the difference between this use of Spark and most standard applications like e.g. in our labs in terms of where the data is stored. What kind of parallelisation approach is used here? (4%)

Write the code below and your answers in the report.

```
###Coding task ###

#i) Improve parallelisation
#If you implemented a straightfoward version, you will probably
observe that all the computation is done on only two nodes.
#This can be adressed by using the second parameter in the initial
call to parallelize.
#Make the suitable change in the code you have written above and mark
it up in comments as ### TASK 1d ###.
#Demonstrate the difference in cluster utilisation before and after
the change based on different parameter values with screenshots from
Google Cloud and measure the difference in the processing time. (6%)


%%writefile 1Dspark.py


#Import the nessesary libraries
import os, sys, math
os.environ['PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION'] = 'python'
import numpy as np
import time
import string
import random
import tensorflow as tf
print("Tensorflow version " + tf.__version__)
import pickle
import pyspark
from pyspark.sql import SQLContext
from pyspark.sql import Row
print(pyspark.__version__)

sc = pyspark.SparkContext.getOrCreate()
print(sc)
```

```
#variables required

PROJECT = 'bigdata-421920'
BUCKET = 'gs://{}-storage'.format(PROJECT)
REGION = 'us-central1'
CLUSTER = '{}-cluster'.format(PROJECT)
GCS_PATTERN = 'gs://flowers-public/*/*.jpg'
PARTITIONS = 16 # no of partitions we will use later
GCS_OUTPUT = BUCKET + '/tfrecords-jpeg-192x192-2/flowers'
CLASSES = [b'daisy', b'dandelion', b'roses', b'sunflowers', b'tulips']
TARGET_SIZE = [192, 192] # target resolution for the images

# Question 1.A.I (Part1)

#Function 1
#Function to decode JPEG image and extract label from the filepath
def decode_jpeg_and_label(filepath):
    # extracts the image data and creates a class label, based on the
filepath
    bits = tf.io.read_file(filepath)
    image = tf.image.decode_jpeg(bits)
    # parse flower name from containing directory
    label = tf.strings.split(tf.expand_dims(filepath, axis=-1),
sep='/')
    label2 = label.values[-2]
    return image, label2

#Function 2
#Function to resize and crop
def resize_and_crop_image(data):
    # Resizes and cropd using "fill" algorithm:
    # always make sure the resulting image is cut out from the source
image
    # so that it fills the TARGET_SIZE entirely with no black bars
    # and a preserved aspect ratio.
    #Obtain image and label from the data
    image, label = data
    #Dimension of image
    w = tf.shape(image)[0]
    h = tf.shape(image)[1]
    #Target size
    tw = TARGET_SIZE[1]
    th = TARGET_SIZE[0]
    #Resizing criteria calculated
    resize_crit = (w * th) / (h * tw)
    #Here the image gets resized according to the rezising criteria
    image = tf.cond(resize_crit < 1,
                    lambda: tf.image.resize(image, [w*tw/w, h*tw/w]),
# if true
                    lambda: tf.image.resize(image, [w*th/h, h*th/h])
```

```
# if false
                    )
    #New dimensions of image calculated
    nw = tf.shape(image)[0]
    nh = tf.shape(image)[1]
    #Image gets cropped to the target size
    image = tf.image.crop_to_bounding_box(image, (nw - tw) // 2, (nh -
th) // 2, tw, th)
    return image, label

#Function 3
#Function to recompress the image
def recompress_image(data):
    # this reduces the amount of data, but takes some time
    #Obtain image and label from the data
    image, label = data
    #Image is 'casted' to an 8 bit integer format
    image = tf.cast(image, tf.uint8)
    #Image gets encoded to jep format
    image = tf.image.encode_jpeg(image, optimize_size=True,
chroma_downsampling=False)
    return image, label

## Question 1.A.II (Part2)
###ii) Replace the TensorFlow Dataset objects with RDDs, starting with
an RDD that contains the list of image filenames. (3%)

#Filenames for the specified pattern
filenames = tf.io.gfile.glob(GCS_PATTERN)


### TASK 1d ###

#RDD for files
filenames_rdd = sc.parallelize(filenames,16)



## Question 1.A.III (Part3)


#RDD files are sampled to a test size
sampled_filenames_rdd = filenames_rdd.sample(False, 0.02)

#Then
# RDD decode for JPEG and label
decode_jpeg_and_label_rdd =
sampled_filenames_rdd.map(decode_jpeg_and_label)

# RDD decode for resize and crop
```

```
resize_and_crop_image_rdd =
decode_jpeg_and_label_rdd.map(resize_and_crop_image)

# Apply recompression function to each resized image
recompress_image_rdd = resize_and_crop_image_rdd.map(recompress_image)



## Question 1.A.IV (Part4)


#Function 4
#Function for byte string feature
def _bytestring_feature(list_of_bytestrings):
  #Set to returning a tensorflow feature with list of byte strings
    return
tf.train.Feature(bytes_list=tf.train.BytesList(value=list_of_bytestrin
gs))

#Function 5
#Integer feature function
def _int_feature(list_of_ints): # int64
#Tensor flow feature with list of integers
    return
tf.train.Feature(int64_list=tf.train.Int64List(value=list_of_ints))

#Function 6
#TFRRecord entry function
def to_tfrecord(tfrec_filewriter, img_bytes, label): #Create tf data
records
    #Convert image bytes and label to TFRecord entry
    class_num = np.argmax(np.array(CLASSES)==label) # 'roses' => 2
(order defined in CLASSES)
    one_hot_class = np.eye(len(CLASSES))[class_num]     # [0, 0, 1, 0,
0] for class #2, roses
    feature = {
        "image": _bytestring_feature([img_bytes]), # one image in the
list
        "class": _int_feature([class_num]) #,        # one class in
the list
    }
    return
tf.train.Example(features=tf.train.Features(feature=feature))



## Question 1.A.V (Part5)
# v) The code for writing to the TFRecord files needs to be put into a
function, that can be applied to every partition with the
'RDD.mapPartitionsWithIndex' function.
```

```
#The return value of that function is not used here, but you should
return the filename, so that you have a list of the created TFRecord
files. (4%)


#Function 7
#Function to write TFRecord files for each partition
def write_tfrecords(index,partition):
  #Update
  print("Writing TFRecords")
  #Setting the filename for the TFRecord file
  filename = GCS_OUTPUT + "{}.tfrec".format(index)
  with tf.io.TFRecordWriter(filename) as out_file:
    #Iterating over each element in the partition
    for element in partition:
      #Extracting image from partition
      image=element[0]
      #Extracting Label from partition
      label=element[1]
      #Image and label conerted to TFRecord entry
      example = to_tfrecord(out_file,
                            image.numpy(), # re-compressed image:
already a byte string
                            label.numpy()
                            )
      #Writing to tfRecord file
      out_file.write(example.SerializeToString())
      #Yield - generator function - iterating over a sequence
    yield [filename]


# Apply the write_tfrecord function to each partition of the RDD
tfrecord_filenames =
recompress_image_rdd.mapPartitionsWithIndex(write_tfrecords)
```

```
Overwriting 1Dspark.py
```

```
###Coding task ###

#submitting the improved paralleized script to the orginal cluster
!gcloud dataproc jobs submit pyspark --cluster $CLUSTER  \1Dspark.py
%time
```

```
Job [5ccb6fd5ca774910a2e3745b77fa4917] submitted.
Waiting for job output...
2024-05-05 06:10:39.670555: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
not load dynamic library 'libcudart.so.11.0'; dlerror:
```

```
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 06:10:39.670601: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
Tensorflow version 2.4.0
2.4.8
24/05/05 06:10:42 INFO org.apache.spark.SparkEnv: Registering
MapOutputTracker
24/05/05 06:10:42 INFO org.apache.spark.SparkEnv: Registering
BlockManagerMaster
24/05/05 06:10:42 INFO org.apache.spark.SparkEnv: Registering
OutputCommitCoordinator
24/05/05 06:10:42 INFO org.spark_project.jetty.util.log: Logging
initialized @4931ms to org.spark_project.jetty.util.log.Slf4jLog
24/05/05 06:10:42 INFO org.spark_project.jetty.server.Server: jetty-
9.4.z-SNAPSHOT; built: unknown; git: unknown; jvm 1.8.0_382-b05
24/05/05 06:10:42 INFO org.spark_project.jetty.server.Server: Started
@5050ms
24/05/05 06:10:42 INFO
org.spark_project.jetty.server.AbstractConnector: Started
ServerConnector@77391f2f{HTTP/1.1, (http/1.1)}{0.0.0.0:45463}
24/05/05 06:10:43 INFO org.apache.hadoop.yarn.client.RMProxy:
Connecting to ResourceManager at
bigdata-421920-cluster-m/10.128.15.225:8032
24/05/05 06:10:43 INFO org.apache.hadoop.yarn.client.AHSProxy:
Connecting to Application History server at
bigdata-421920-cluster-m/10.128.15.225:10200
24/05/05 06:10:43 INFO org.apache.hadoop.conf.Configuration: resource-
types.xml not found
24/05/05 06:10:43 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find
'resource-types.xml'.
24/05/05 06:10:43 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = memory-mb, units = Mi, type = COUNTABLE
24/05/05 06:10:43 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = vcores, units = , type = COUNTABLE
24/05/05 06:10:45 INFO
org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted
application application_1714888875884_0002
<SparkContext master=yarn appName=1Dspark.py>
24/05/05 06:10:54 INFO
org.spark_project.jetty.server.AbstractConnector: Stopped
Spark@77391f2f{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
Job [5ccb6fd5ca774910a2e3745b77fa4917] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-central1-504478955913-
```

```
tcjfqtey/google-cloud-dataproc-metainfo/cb512dbe-e541-4b59-ae6a-
3678a913c366/jobs/5ccb6fd5ca774910a2e3745b77fa4917/
driverOutputResourceUri: gs://dataproc-staging-us-central1-
504478955913-tcjfqtey/google-cloud-dataproc-metainfo/cb512dbe-e541-
4b59-ae6a-3678a913c366/jobs/5ccb6fd5ca774910a2e3745b77fa4917/
driveroutput
jobUuid: bc8cdcb4-232b-3b37-a1ec-b9e15181a49b
placement:
  clusterName: bigdata-421920-cluster
  clusterUuid: cb512dbe-e541-4b59-ae6a-3678a913c366
pysparkJob:
  mainPythonFileUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/cb512dbe-e541-4b59-ae6a-
3678a913c366/jobs/5ccb6fd5ca774910a2e3745b77fa4917/staging/1Dspark.py
reference:
  jobId: 5ccb6fd5ca774910a2e3745b77fa4917
  projectId: bigdata-421920
status:
  state: DONE
  stateStartTime: '2024-05-05T06:10:55.955851Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-05-05T06:10:36.595342Z'
- state: SETUP_DONE
  stateStartTime: '2024-05-05T06:10:36.621804Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-05-05T06:10:36.794560Z'
yarnApplications:
- name: 1Dspark.py
  progress: 1.0
  state: FINISHED
  trackingUrl:
http://bigdata-421920-cluster-m:8088/proxy/application_1714888875884_0
002/
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 9.06 µs
```

```
###Coding task ###

# submitting the improved paralleized script to the in max cluster
with 3 worker nodes, called bigdata-maxluster
!gcloud dataproc jobs submit pyspark --cluster bigdata-maxcluster  \
1Dspark.py
%time
```

```
Job [c1634459ac3a4cb7b2b9e443d7e0c5ee] submitted.
Waiting for job output...
2024-05-05 06:11:16.036782: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
```

```
not load dynamic library 'libcudart.so.11.0'; dlerror:
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 06:11:16.036958: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
Tensorflow version 2.4.0
2.4.8
24/05/05 06:11:20 INFO org.apache.spark.SparkEnv: Registering
MapOutputTracker
24/05/05 06:11:20 INFO org.apache.spark.SparkEnv: Registering
BlockManagerMaster
24/05/05 06:11:20 INFO org.apache.spark.SparkEnv: Registering
OutputCommitCoordinator
24/05/05 06:11:20 INFO org.spark_project.jetty.util.log: Logging
initialized @8417ms to org.spark_project.jetty.util.log.Slf4jLog
24/05/05 06:11:20 INFO org.spark_project.jetty.server.Server: jetty-
9.4.z-SNAPSHOT; built: unknown; git: unknown; jvm 1.8.0_382-b05
24/05/05 06:11:20 INFO org.spark_project.jetty.server.Server: Started
@8665ms
24/05/05 06:11:20 INFO
org.spark_project.jetty.server.AbstractConnector: Started
ServerConnector@51f17779{HTTP/1.1, (http/1.1)}{0.0.0.0:35465}
24/05/05 06:11:23 INFO org.apache.hadoop.yarn.client.RMProxy:
Connecting to ResourceManager at
bigdata-maxcluster-m/10.128.15.228:8032
24/05/05 06:11:23 INFO org.apache.hadoop.yarn.client.AHSProxy:
Connecting to Application History server at
bigdata-maxcluster-m/10.128.15.228:10200
24/05/05 06:11:23 INFO org.apache.hadoop.conf.Configuration: resource-
types.xml not found
24/05/05 06:11:23 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find
'resource-types.xml'.
24/05/05 06:11:23 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = memory-mb, units = Mi, type = COUNTABLE
24/05/05 06:11:23 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = vcores, units = , type = COUNTABLE
24/05/05 06:11:26 INFO
org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted
application application_1714889193247_0002
<SparkContext master=yarn appName=1Dspark.py>
24/05/05 06:11:42 INFO
org.spark_project.jetty.server.AbstractConnector: Stopped
Spark@51f17779{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
Job [c1634459ac3a4cb7b2b9e443d7e0c5ee] finished successfully.
done: true
```

```
driverControlFilesUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/35052232-f9c1-49b1-ae63-
a695f2252802/jobs/c1634459ac3a4cb7b2b9e443d7e0c5ee/
driverOutputResourceUri: gs://dataproc-staging-us-central1-
504478955913-tcjfqtey/google-cloud-dataproc-metainfo/35052232-f9c1-
49b1-ae63-a695f2252802/jobs/c1634459ac3a4cb7b2b9e443d7e0c5ee/
driveroutput
jobUuid: 3b3a4af9-7646-3a6e-b6e1-01a034995b28
placement:
  clusterName: bigdata-maxcluster
  clusterUuid: 35052232-f9c1-49b1-ae63-a695f2252802
pysparkJob:
  mainPythonFileUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/35052232-f9c1-49b1-ae63-
a695f2252802/jobs/c1634459ac3a4cb7b2b9e443d7e0c5ee/staging/1Dspark.py
reference:
  jobId: c1634459ac3a4cb7b2b9e443d7e0c5ee
  projectId: bigdata-421920
status:
  state: DONE
  stateStartTime: '2024-05-05T06:11:48.340314Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-05-05T06:11:11.108064Z'
- state: SETUP_DONE
  stateStartTime: '2024-05-05T06:11:11.136024Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-05-05T06:11:11.331021Z'
yarnApplications:
- name: 1Dspark.py
  progress: 1.0
  state: FINISHED
  trackingUrl:
http://bigdata-maxcluster-m:8088/proxy/application_1714889193247_0002/
CPU times: user 5 µs, sys: 0 ns, total: 5 µs
Wall time: 8.82 µs

### CODING TASK ###

###ii) Experiment with cluster configurations.
#In addition to the experiments above (using 8 VMs),test your program
with 4 machines with double the resources each (2 vCPUs, memory, disk)
#and 1 machine with eightfold resources. Discuss the results in terms
of disk I/O and network bandwidth allocation in the cloud. (7%)


#cluster 4 machines (1 master + 3 workers) with double resources (2
vCPUs, memory, disk)
```

```
!gcloud dataproc clusters create fourmachine-cluster \
    --image-version 1.5-ubuntu18 \
    --master-machine-type n1-standard-2 \
    --master-boot-disk-type pd-ssd --master-boot-disk-size 100 \
    --num-workers 3 --worker-machine-type n1-standard-2 --worker-boot-
disk-size 100 \
    --initialization-actions gs://goog-dataproc-initialization-
actions-$REGION/python/pip-install.sh \
    --metadata "PIP_PACKAGES=tensorflow==2.4.0 scipy==1.4.1 pandas
numpy==1.21.6 matplotlib" \
    --zone us-central1-c


Waiting on operation
[projects/bigdata-421920/regions/us-central1/operations/48da4acf-b813-
3d51-a1fe-d49f5bd45bda].

WARNING: Consider using Auto Zone rather than selecting a zone
manually. See
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/
auto-zone
WARNING: Don't create production clusters that reference
initialization actions located in the gs://goog-dataproc-
initialization-actions-REGION public buckets. These scripts are
provided as reference implementations, and they are synchronized with
ongoing GitHub repository changes—a new version of a initialization
action in public buckets may break your cluster creation. Instead,
copy the following initialization actions from public buckets into
your bucket :
gs://goog-dataproc-initialization-actions-us-central1/python/pip-
install.sh
WARNING: For PD-Standard without local SSDs, we strongly recommend
provisioning 1TB or larger to ensure consistently high I/O
performance. See
https://cloud.google.com/compute/docs/disks/performance for
information on disk I/O performance.
WARNING: Permissions are missing for the default service account
'504478955913-compute@developer.gserviceaccount.com', missing
permissions: [storage.objects.get, storage.objects.update] on the
staging_bucket 'projects/_/buckets/dataproc-staging-us-central1-
504478955913-tcjfqtey'. This usually happens when a custom resource
(ex: custom staging bucket) or a user-managed VM Service account has
been provided and the default/user-managed service account hasn't been
granted enough permissions on the resource. See
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/
service-accounts#VM_service_account.
WARNING: Permissions are missing for the default service account
'504478955913-compute@developer.gserviceaccount.com', missing
permissions: [storage.objects.get, storage.objects.update] on the
```

```
temp_bucket 'projects/_/buckets/dataproc-temp-us-central1-
504478955913-s3uruvrj'. This usually happens when a custom resource
(ex: custom staging bucket) or a user-managed VM Service account has
been provided and the default/user-managed service account hasn't been
granted enough permissions on the resource. See
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/
service-accounts#VM_service_account.
WARNING: The firewall rules for specified network or subnetwork would
allow ingress traffic from 0.0.0.0/0, which could be a security risk.
WARNING: The specified custom staging bucket 'dataproc-staging-us-
central1-504478955913-tcjfqtey' is not using uniform bucket level
access IAM configuration. It is recommended to update bucket to enable
the same. See https://cloud.google.com/storage/docs/uniform-bucket-
level-access.
Created
[https://dataproc.googleapis.com/v1/projects/bigdata-421920/regions/
us-central1/clusters/fourmachine-cluster] Cluster placed in zone [us-
central1-c].
```

```
### CODing TASK ###
# submit spark job for cluster with 4 machines with double resources
(vCPUs, memory, disk) using improved parallelization script
!gcloud dataproc jobs submit pyspark --cluster fourmachine-cluster  \
1Dspark.py
%time
```

```
Job [427da77c0f974a1b8e104f0ba49b26ff] submitted.
Waiting for job output...
2024-05-05 06:47:33.044071: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
not load dynamic library 'libcudart.so.11.0'; dlerror:
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 06:47:33.044117: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
Tensorflow version 2.4.0
2.4.8
24/05/05 06:47:37 INFO org.apache.spark.SparkEnv: Registering
MapOutputTracker
24/05/05 06:47:37 INFO org.apache.spark.SparkEnv: Registering
BlockManagerMaster
24/05/05 06:47:37 INFO org.apache.spark.SparkEnv: Registering
OutputCommitCoordinator
24/05/05 06:47:37 INFO org.spark_project.jetty.util.log: Logging
initialized @7640ms to org.spark_project.jetty.util.log.Slf4jLog
24/05/05 06:47:37 INFO org.spark_project.jetty.server.Server: jetty-
9.4.z-SNAPSHOT; built: unknown; git: unknown; jvm 1.8.0_382-b05
24/05/05 06:47:37 INFO org.spark_project.jetty.server.Server: Started
@7818ms
```

```
24/05/05 06:47:37 INFO
org.spark_project.jetty.server.AbstractConnector: Started
ServerConnector@25771daa{HTTP/1.1, (http/1.1)}{0.0.0.0:43099}
24/05/05 06:47:39 INFO org.apache.hadoop.yarn.client.RMProxy:
Connecting to ResourceManager at
fourmachine-cluster-m/10.128.15.232:8032
24/05/05 06:47:40 INFO org.apache.hadoop.yarn.client.AHSProxy:
Connecting to Application History server at
fourmachine-cluster-m/10.128.15.232:10200
24/05/05 06:47:40 INFO org.apache.hadoop.conf.Configuration: resource-
types.xml not found
24/05/05 06:47:40 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find
'resource-types.xml'.
24/05/05 06:47:40 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = memory-mb, units = Mi, type = COUNTABLE
24/05/05 06:47:40 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = vcores, units = , type = COUNTABLE
24/05/05 06:47:43 INFO
org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted
application application_1714891451528_0001
<SparkContext master=yarn appName=1Dspark.py>
24/05/05 06:48:02 INFO
org.spark_project.jetty.server.AbstractConnector: Stopped
Spark@25771daa{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
Job [427da77c0f974a1b8e104f0ba49b26ff] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/5bf1da11-83a9-45f8-b172-
68d5e4118af6/jobs/427da77c0f974a1b8e104f0ba49b26ff/
driverOutputResourceUri: gs://dataproc-staging-us-central1-
504478955913-tcjfqtey/google-cloud-dataproc-metainfo/5bf1da11-83a9-
45f8-b172-68d5e4118af6/jobs/427da77c0f974a1b8e104f0ba49b26ff/
driveroutput
jobUuid: 205a2cf5-a924-3bff-932e-e102d7edaf6e
placement:
  clusterName: fourmachine-cluster
  clusterUuid: 5bf1da11-83a9-45f8-b172-68d5e4118af6
pysparkJob:
  mainPythonFileUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/5bf1da11-83a9-45f8-b172-
68d5e4118af6/jobs/427da77c0f974a1b8e104f0ba49b26ff/staging/1Dspark.py
reference:
  jobId: 427da77c0f974a1b8e104f0ba49b26ff
  projectId: bigdata-421920
status:
  state: DONE
```

```
  stateStartTime: '2024-05-05T06:48:05.718477Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-05-05T06:47:27.671474Z'
- state: SETUP_DONE
  stateStartTime: '2024-05-05T06:47:27.702860Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-05-05T06:47:28.080328Z'
yarnApplications:
- name: 1Dspark.py
  progress: 1.0
  state: FINISHED
  trackingUrl:
http://fourmachine-cluster-m:8088/proxy/application_1714891451528_0001
/
CPU times: user 5 µs, sys: 0 ns, total: 5 µs
Wall time: 9.78 µs
```

```
#The older clusters were deleted to allow enough resources for this
cluster
```

```
### CODing task ###]
```

```
#cluster with one machine and eightfold cluster
#The older clusters were deleted to allow enough resources for this
cluster
```

```
!gcloud dataproc clusters create eightfold-cluster \
    --image-version 1.5-ubuntu18 \
    --master-machine-type n1-standard-8 \
    --master-boot-disk-type pd-ssd --master-boot-disk-size 100\
    --num-workers 0 \
    --initialization-actions gs://goog-dataproc-initialization-
actions-$REGION/python/pip-install.sh \
    --metadata "PIP_PACKAGES=tensorflow==2.4.0 scipy==1.4.1 pandas
numpy==1.21.6 matplotlib" \
    --zone us-central1-c
```

```
Waiting on operation
[projects/bigdata-421920/regions/us-central1/operations/66590797-d0e2-
3cbb-a977-9679b0f685fe].

WARNING: Consider using Auto Zone rather than selecting a zone
manually. See
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/
auto-zone
WARNING: Don't create production clusters that reference
initialization actions located in the gs://goog-dataproc-
initialization-actions-REGION public buckets. These scripts are
```

```
provided as reference implementations, and they are synchronized with
ongoing GitHub repository changes—a new version of a initialization
action in public buckets may break your cluster creation. Instead,
copy the following initialization actions from public buckets into
your bucket :
gs://goog-dataproc-initialization-actions-us-central1/python/pip-
install.sh
WARNING: Permissions are missing for the default service account
'504478955913-compute@developer.gserviceaccount.com', missing
permissions: [storage.objects.get, storage.objects.update] on the
staging_bucket 'projects/_/buckets/dataproc-staging-us-central1-
504478955913-tcjfqtey'. This usually happens when a custom resource
(ex: custom staging bucket) or a user-managed VM Service account has
been provided and the default/user-managed service account hasn't been
granted enough permissions on the resource. See
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/
service-accounts#VM_service_account.
WARNING: Permissions are missing for the default service account
'504478955913-compute@developer.gserviceaccount.com', missing
permissions: [storage.objects.get, storage.objects.update] on the
temp_bucket 'projects/_/buckets/dataproc-temp-us-central1-
504478955913-s3uruvrj'. This usually happens when a custom resource
(ex: custom staging bucket) or a user-managed VM Service account has
been provided and the default/user-managed service account hasn't been
granted enough permissions on the resource. See
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/
service-accounts#VM_service_account.
WARNING: The firewall rules for specified network or subnetwork would
allow ingress traffic from 0.0.0.0/0, which could be a security risk.
WARNING: The specified custom staging bucket 'dataproc-staging-us-
central1-504478955913-tcjfqtey' is not using uniform bucket level
access IAM configuration. It is recommended to update bucket to enable
the same. See https://cloud.google.com/storage/docs/uniform-bucket-
level-access.
Created
[https://dataproc.googleapis.com/v1/projects/bigdata-421920/regions/
us-central1/clusters/eightfold-cluster] Cluster placed in zone [us-
central1-c].
```

```
### CODing TASK ###
# submit spark job for one machine with eightfold resources cluster
!gcloud dataproc jobs submit pyspark --cluster eightfold-cluster  \
1Dspark.py
%time
```

```
Job [fd6ec7778f8d4e268b64e41d8dbc44bc] submitted.
Waiting for job output...
2024-05-05 06:15:23.860350: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
not load dynamic library 'libcudart.so.11.0'; dlerror:
```

```
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 06:15:23.860394: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
Tensorflow version 2.4.0
2.4.8
24/05/05 06:15:27 INFO org.apache.spark.SparkEnv: Registering
MapOutputTracker
24/05/05 06:15:27 INFO org.apache.spark.SparkEnv: Registering
BlockManagerMaster
24/05/05 06:15:27 INFO org.apache.spark.SparkEnv: Registering
OutputCommitCoordinator
24/05/05 06:15:27 INFO org.spark_project.jetty.util.log: Logging
initialized @5951ms to org.spark_project.jetty.util.log.Slf4jLog
24/05/05 06:15:27 INFO org.spark_project.jetty.server.Server: jetty-
9.4.z-SNAPSHOT; built: unknown; git: unknown; jvm 1.8.0_382-b05
24/05/05 06:15:27 INFO org.spark_project.jetty.server.Server: Started
@6082ms
24/05/05 06:15:27 INFO
org.spark_project.jetty.server.AbstractConnector: Started
ServerConnector@7dd3080f{HTTP/1.1, (http/1.1)}{0.0.0.0:37233}
24/05/05 06:15:29 INFO org.apache.hadoop.yarn.client.RMProxy:
Connecting to ResourceManager at
eightfold-cluster-m/10.128.15.230:8032
24/05/05 06:15:29 INFO org.apache.hadoop.yarn.client.AHSProxy:
Connecting to Application History server at
eightfold-cluster-m/10.128.15.230:10200
24/05/05 06:15:29 INFO org.apache.hadoop.conf.Configuration: resource-
types.xml not found
24/05/05 06:15:29 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Unable to find
'resource-types.xml'.
24/05/05 06:15:29 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = memory-mb, units = Mi, type = COUNTABLE
24/05/05 06:15:29 INFO
org.apache.hadoop.yarn.util.resource.ResourceUtils: Adding resource
type - name = vcores, units = , type = COUNTABLE
24/05/05 06:15:32 INFO
org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted
application application_1714889611603_0001
<SparkContext master=yarn appName=1Dspark.py>
24/05/05 06:15:44 INFO
org.spark_project.jetty.server.AbstractConnector: Stopped
Spark@7dd3080f{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
Job [fd6ec7778f8d4e268b64e41d8dbc44bc] finished successfully.
done: true
driverControlFilesUri: gs://dataproc-staging-us-central1-504478955913-
```

```
tcjfqtey/google-cloud-dataproc-metainfo/19b96cb7-e604-4336-aa84-
02d92767b865/jobs/fd6ec7778f8d4e268b64e41d8dbc44bc/
driverOutputResourceUri: gs://dataproc-staging-us-central1-
504478955913-tcjfqtey/google-cloud-dataproc-metainfo/19b96cb7-e604-
4336-aa84-02d92767b865/jobs/fd6ec7778f8d4e268b64e41d8dbc44bc/
driveroutput
jobUuid: 2297cec0-adee-3556-bd10-5e1519f415d8
placement:
  clusterName: eightfold-cluster
  clusterUuid: 19b96cb7-e604-4336-aa84-02d92767b865
pysparkJob:
  mainPythonFileUri: gs://dataproc-staging-us-central1-504478955913-
tcjfqtey/google-cloud-dataproc-metainfo/19b96cb7-e604-4336-aa84-
02d92767b865/jobs/fd6ec7778f8d4e268b64e41d8dbc44bc/staging/1Dspark.py
reference:
  jobId: fd6ec7778f8d4e268b64e41d8dbc44bc
  projectId: bigdata-421920
status:
  state: DONE
  stateStartTime: '2024-05-05T06:15:49.862733Z'
statusHistory:
- state: PENDING
  stateStartTime: '2024-05-05T06:15:19.421195Z'
- state: SETUP_DONE
  stateStartTime: '2024-05-05T06:15:19.446754Z'
- details: Agent reported job success
  state: RUNNING
  stateStartTime: '2024-05-05T06:15:19.702417Z'
yarnApplications:
- name: 1Dspark.py
  progress: 1.0
  state: FINISHED
  trackingUrl:
http://eightfold-cluster-m:8088/proxy/application_1714889611603_0001/
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 7.87 µs
```

Section 2: Speed tests

We have seen that **reading from the pre-processed TFRecord files** is **faster** than reading individual image files and decoding on the fly. This task is about **measuring this effect** and **parallelizing the tests with PySpark**.

2.1 Speed test implementation

Here is **code for time measurement** to determine the **throughput in images per second**. It doesn't render the images but extracts and prints some basic information in order to make sure

the image data are read. We write the information to the null device for longer measurements `null_file=open("/dev/null", mode='w')`. That way it will not clutter our cell output.

We use batches (`dset2 = dset1.batch(batch_size)`) and select a number of batches with (`dset3 = dset2.take(batch_number)`). Then we use the `time.time()` to take the **time measurement** and take it multiple times, reading from the same dataset to see if reading speed changes with mutiple readings.

We then **vary** the size of the batch (`batch_size`) and the number of batches (`batch_number`) and **store the results for different values**. Store also the **results for each repetition** over the same dataset (repeat 2 or 3 times).

The speed test should be combined in a **function** `time_configs()` that takes a configuration, i.e. a dataset and arrays of `batch_sizes`, `batch_numbers`, and `repetitions` (an array of integers starting from 1), as **arguments** and runs the time measurement for each combination of batch_size and batch_number for the requested number of repetitions.

```
# Here are some useful values for testing your code, use higher values
later for actually testing throughput
batch_sizes = [2,4]
batch_numbers = [3,6]
repetitions = [1]

def time_configs(dataset, batch_sizes, batch_numbers, repetitions):
    dims = [len(batch_sizes),len(batch_numbers),len(repetitions)]
    print(dims)
    results = np.zeros(dims)
    params = np.zeros(dims + [3])
    print( results.shape )
    with open("/dev/null",mode='w') as null_file: # for printing the
output without showing it
        tt = time.time() # for overall time taking
        for bsi,bs in enumerate(batch_sizes):
            for dsi, ds in enumerate(batch_numbers):
                batched_dataset = dataset.batch(bs)
                timing_set = batched_dataset.take(ds)
                for ri,rep in enumerate(repetitions):
                    print("bs: {}, ds: {}, rep: {}".format(bs,ds,rep))
                    t0 = time.time()
                    for image, label in timing_set:
                        #print("Image batch shape
{}".format(image.numpy().shape),
                        print("Image batch shape {},
{})".format(image.numpy().shape,
                                [str(lbl) for lbl in label.numpy()]),
null_file)
                    td = time.time() - t0 # duration for reading
images
                    results[bsi,dsi,ri] = ( bs * ds) / td
                    params[bsi,dsi,ri] = [ bs, ds, rep ]
```

```
    print("total time: "+str(time.time()-tt))
    return results, params
```

Let's try this function with a **small number** of configurations of batch_sizes batch_numbers and repetions, so that we get a set of parameter combinations and corresponding reading speeds. Try reading from the image files (dataset4) and the TFRecord files (datasetTfrec).

```
[res,par] = time_configs(dataset4, batch_sizes, batch_numbers,
repetitions)
print(res)
print(par)

print("=============")

[res,par] = time_configs(datasetTfrec, batch_sizes, batch_numbers,
repetitions)
print(res)
print(par)
```

```
[2, 2, 1]
(2, 2, 1)
bs: 2, ds: 3, rep: 1
Image batch shape (2,), ["b'dandelion'", "b'dandelion'"])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2,), ["b'roses'", "b'tulips'"]) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2,), ["b'daisy'", "b'roses'"]) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
bs: 2, ds: 6, rep: 1
Image batch shape (2,), ["b'tulips'", "b'roses'"]) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2,), ["b'dandelion'", "b'tulips'"])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2,), ["b'sunflowers'", "b'dandelion'"])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2,), ["b'roses'", "b'tulips'"]) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2,), ["b'tulips'", "b'tulips'"]) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2,), ["b'dandelion'", "b'dandelion'"])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
bs: 4, ds: 3, rep: 1
Image batch shape (4,), ["b'sunflowers'", "b'sunflowers'",
"b'dandelion'", "b'dandelion'"]) <_io.TextIOWrapper name='/dev/null'
mode='w' encoding='UTF-8'>
Image batch shape (4,), ["b'sunflowers'", "b'sunflowers'",
"b'tulips'", "b'tulips'"]) <_io.TextIOWrapper name='/dev/null'
mode='w' encoding='UTF-8'>
Image batch shape (4,), ["b'roses'", "b'tulips'", "b'daisy'",
```

```
"b'daisy'"]) <_io.TextIOWrapper name='/dev/null' mode='w'
encoding='UTF-8'>
bs: 4, ds: 6, rep: 1
Image batch shape (4,), ["b'tulips'", "b'dandelion'", "b'tulips'",
"b'dandelion'"]) <_io.TextIOWrapper name='/dev/null' mode='w'
encoding='UTF-8'>
Image batch shape (4,), ["b'dandelion'", "b'sunflowers'", "b'tulips'",
"b'dandelion'"]) <_io.TextIOWrapper name='/dev/null' mode='w'
encoding='UTF-8'>
Image batch shape (4,), ["b'dandelion'", "b'daisy'", "b'sunflowers'",
"b'tulips'"]) <_io.TextIOWrapper name='/dev/null' mode='w'
encoding='UTF-8'>
Image batch shape (4,), ["b'sunflowers'", "b'roses'", "b'dandelion'",
"b'sunflowers'"]) <_io.TextIOWrapper name='/dev/null' mode='w'
encoding='UTF-8'>
Image batch shape (4,), ["b'daisy'", "b'tulips'", "b'dandelion'",
"b'tulips'"]) <_io.TextIOWrapper name='/dev/null' mode='w'
encoding='UTF-8'>
Image batch shape (4,), ["b'tulips'", "b'dandelion'", "b'roses'",
"b'dandelion'"]) <_io.TextIOWrapper name='/dev/null' mode='w'
encoding='UTF-8'>
total time: 2.653258800506592
[[[12.40980147]
  [19.41044319]]

 [[22.10946661]
  [24.33795625]]]
[[[[2. 3. 1.]]

  [[2. 6. 1.]]]


 [[[4. 3. 1.]]

  [[4. 6. 1.]]]]
==============
[2, 2, 1]
(2, 2, 1)
bs: 2, ds: 3, rep: 1
Image batch shape (2, 192, 192, 3), ['2', '4']) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2, 192, 192, 3), ['4', '4']) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2, 192, 192, 3), ['2', '3']) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
bs: 2, ds: 6, rep: 1
Image batch shape (2, 192, 192, 3), ['2', '4']) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2, 192, 192, 3), ['4', '4']) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
```

```
Image batch shape (2, 192, 192, 3), ['2', '3']) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2, 192, 192, 3), ['3', '2']) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2, 192, 192, 3), ['2', '1']) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (2, 192, 192, 3), ['3', '3']) <_io.TextIOWrapper
name='/dev/null' mode='w' encoding='UTF-8'>
bs: 4, ds: 3, rep: 1
Image batch shape (4, 192, 192, 3), ['2', '4', '4', '4'])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (4, 192, 192, 3), ['2', '3', '3', '2'])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (4, 192, 192, 3), ['2', '1', '3', '3'])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
bs: 4, ds: 6, rep: 1
Image batch shape (4, 192, 192, 3), ['2', '4', '4', '4'])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (4, 192, 192, 3), ['2', '3', '3', '2'])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (4, 192, 192, 3), ['2', '1', '3', '3'])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (4, 192, 192, 3), ['4', '4', '0', '2'])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (4, 192, 192, 3), ['4', '2', '1', '0'])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
Image batch shape (4, 192, 192, 3), ['4', '2', '2', '2'])
<_io.TextIOWrapper name='/dev/null' mode='w' encoding='UTF-8'>
total time: 0.6948878765106201
[[[ 39.92395264]
  [ 67.14153955]]

 [[ 69.1834449 ]
  [140.80476253]]]
[[[[2. 3. 1.]]

  [[2. 6. 1.]]]


 [[[4. 3. 1.]]

  [[4. 6. 1.]]]]
```

Task 2: Parallelising the speed test with Spark in the cloud. (36%)

As an exercise in **Spark programming and optimisation** as well as **performance analysis**, we will now implement the **speed test** with multiple parameters in parallel with Spark. Runing

multiple tests in parallel would **not be a useful approach on a single machine, but it can be in the cloud** (you will be asked to reason about this later).

2a) Create the script (14%)

Your task is now to **port the speed test above to Spark** for running it in the cloud in Dataproc. **Adapt the speed testing** as a Spark program that performs the same actions as above, but **with Spark RDDs in a distributed way**. The distribution should be such that **each parameter combination (except repetition)** is processed in a separate Spark task.

More specifically:

- i) combine the previous cells to have the code to create a dataset and create a list of parameter combinations in an RDD (2%)
- ii) get a Spark context and create the dataset and run timing test for each combination in parallel (2%)
- iii) transform the resulting RDD to the structure (parameter_combination, images_per_second) and save these values in an array (2%)
- iv) create an RDD with all results for each parameter as (parameter_value,images_per_second) and collect the result for each parameter (2%)
- v) create an RDD with the average reading speeds for each parameter value and collect the results. Keep associativity in mind when implementing the average. (3%)
- vi) write the results to a pickle file in your bucket (2%)
- vii) Write your code it into a file using the *cell magic* `%%writefile spark_job.py` (1%)

Important: The task here is not to parallelize the pre-processing, but to run multiple speed tests in parallel using Spark.

```
###Coding task###

#Importing nessesary libraries
import pyspark
from pyspark.sql import SQLContext
from pyspark.sql import Row
from pyspark.sql.types import *
import pandas as pd
import pyspark
from pyspark.sql import SparkSession
import time

#nessesary variables
PROJECT = 'bigdata-421920'
BUCKET = 'gs://{}-storage'.format(PROJECT)
REGION = 'us-central1'
CLUSTER = '{}-cluster'.format(PROJECT)
GCS_PATTERN = 'gs://flowers-public/*/*.jpg'
PARTITIONS = 16 # no of partitions we will use later
GCS_OUTPUT = BUCKET + '/tfrecords-jpeg-192x192-2/flowers'
```

```
CLASSES = [b'daisy', b'dandelion', b'roses', b'sunflowers', b'tulips']
TARGET_SIZE = [192, 192] # target resolution for the images


#i) combine the previous cells to have the code to create a dataset
and create a list of parameter combinations in an RDD (2%)


#Function to parse TFRecord examples
def read_tfrecord(example):
    features = {
        "image": tf.io.FixedLenFeature([], tf.string),  # tf.string =
bytestring (not text string)
        "class": tf.io.FixedLenFeature([], tf.int64) #,   # shape []
means scalar
    }
    # decode the TFRecord
    # Parse a single TFRecord example using the specified features
    example = tf.io.parse_single_example(example, features)
    #Decoing image from TFRrecord example
    image = tf.image.decode_jpeg(example['image'], channels=3)
    #reshaping image to target size
    image = tf.reshape(image, [*TARGET_SIZE, 3])
    #Extracting class label
    class_num = example['class']
    #output will be parsed image and class label
    return image, class_num

#Function to load dataset from TFRecord files
def load_dataset(filenames):
    # read from TFRecords. For optimal performance, read from multiple
    # TFRecord files at once and set the option
experimental_deterministic = False
    # to allow order-altering optimizations.

    option_no_order = tf.data.Options()
    option_no_order.experimental_deterministic = False
    #dataset
    dataset = tf.data.TFRecordDataset(filenames)
    dataset = dataset.with_options(option_no_order)
    dataset = dataset.map(read_tfrecord)
    return dataset

#Function to resize and crop the image
def resize_and_crop_image(image, label):
    # Resizes and cropd using "fill" algorithm:
    # always make sure the resulting image is cut out from the source
image
    # so that it fills the TARGET_SIZE entirely with no black bars
    # and a preserved aspect ratio.
```

```
    #Dimensions of the image
    w = tf.shape(image)[0]
    h = tf.shape(image)[1]
    tw = TARGET_SIZE[1]
    th = TARGET_SIZE[0]
    #Resize image
    resize_crit = (w * th) / (h * tw)
    image = tf.cond(resize_crit < 1,
                    lambda: tf.image.resize(image, [w*tw/w, h*tw/w]),
# if true
                    lambda: tf.image.resize(image, [w*th/h, h*th/h])
# if false
                    )
    nw = tf.shape(image)[0]
    nh = tf.shape(image)[1]
    #Cropped image is resized to target size
    image = tf.image.crop_to_bounding_box(image, (nw - tw) // 2, (nh -
th) // 2, tw, th)
    #Return the new image (resized and cropped) and label
    return image, label


# Function to load dataset from TFRecord files and decode
def load_dataset_decoded():
    #Obtaining list of TFRecord files
    dataset_filename = tf.data.Dataset.list_files(GCS_PATTERN)
    datasetDecoded = dataset_filename.map(decode_jpeg_and_label)
    datasetfn = datasetDecoded.map(resize_and_crop_image)
    return datasetfn


# Define the new time_configs function
def time_configs_rdd(parameters_rdd):
    #Timer
    start = time.time()
    #Extracting parameters from RDD
    batch_size = parameters_rdd[0]
    batch_number = parameters_rdd[1]
    repetition = parameters_rdd[2]
    dataset_type = parameters_rdd[3]

    #Loading dataset based on type
    if dataset_type == 'datasetDecoded':
        filenames = tf.io.gfile.glob(GCS_OUTPUT + "*.tfrec")
        dataset = load_dataset(filenames)
    else:
        filenames_fn = tf.data.Dataset.list_files(GCS_PATTERN)
        dataset_fn = filenames_fn.map(decode_jpeg_and_label)
        dataset = dataset_fn.map(resize_and_crop_image)
```

```
    #Batch of dataset
    dataset1 = dataset.batch(batch_size)
    test_set = dataset1.take(batch_number)
    time_list = []

    #Time test
    for _ in range(repetition):
        s_time = time.time()
        for _ in test_set:
            print('string', file=open("/dev/null", mode='w'))
        e_time = time.time()
        #Reading speed is calculated
        reading_speed = e_time - s_time
        #Throughput is calculated
        throughput = float((batch_size * batch_number) / (e_time -
s_time))
        datasetsize = batch_size * batch_number
        #Times values are appended to the list
        time_list.append([batch_size, batch_number, repetition,
datasetsize, reading_speed, throughput])

    end = time.time()
    #calculating total time and toal images
    total_images = batch_size * batch_number * repetition
    total_time = total_images / (end - start)
    return total_time, time_list


# Define the parameter combinations, for which will be used
batch_sizes = [2, 4, 6, 8]
batch_numbers = [6, 9, 12, 15]
repetitions = [1, 2, 3]
dataset_types = ['datasetDecoded', 'otherDataset']

#list to store parameter combinations
parameter_list = []
for batch_size in batch_sizes:
    for batch_number in batch_numbers:
        for repetition in repetitions:
          for dataset_type in dataset_types:
              parameter_list.append([batch_size, batch_number,
repetition, dataset_type])


# Define the columns for DataFrame
columns = ["batch_size", "batch_number", "repetition", "dataset_type",
"datasetsize", "reading_speed", "throughput"]

# Create Spark session
spark =
```

```
SparkSession.builder.master("local").appName("DataProcessing").getOrCr
eate()

# Create RDD for parameter combinations
rdd_parameters = spark.sparkContext.parallelize(parameter_list)
```

```
+----------+------------+----------+-----------+-------------------
+------------------+
|batch_size|batch_number|repetition|datasetsize|      reading_speed|
throughput|
+----------+------------+----------+-----------+-------------------
+------------------+
|         2|           6|         1|         12|0.11340498924255371|
105.81545027383292|
|         2|           6|         1|         12| 0.5685160160064697|
21.107584768312737|
|         2|           6|         2|         12|0.09154415130615234|
131.084289152108|
|         2|           6|         2|         12|0.09423708915710449|
127.3383983666406|
|         2|           6|         2|         12| 0.6272392272949219|
19.13145651261654|
|         2|           6|         2|         12| 0.6502125263214111|
18.455504183978448|
|         2|           6|         3|         12| 0.1838667392730713|
65.26465878191321|
|         2|           6|         3|         12|0.16002917289733887|
74.98632769725168|
|         2|           6|         3|         12|0.15868902206420898|
75.61959765020508|
|         2|           6|         3|         12| 0.6737875938415527|
17.80976692013998|
|         2|           6|         3|         12| 0.5520656108856201|
21.736546822305552|
|         2|           6|         3|         12|  0.616633415222168|
19.460508794640166|
|         2|           9|         1|         18|0.09758877754211426|
184.44743804923812|
|         2|           9|         1|         18| 0.6584396362304688|
27.337357913398144|
|         2|           9|         2|         18|0.11090731620788574|
162.2976789579925|
|         2|           9|         2|         18|0.08464980125427246|
212.64078276960515|
|         2|           9|         2|         18| 0.7209558486938477|
24.966854811720463|
|         2|           9|         2|         18| 0.6572496891021729|
27.386852057075384|
```

```
|         2|           9|         3|         18|0.09597921371459961|
187.54060700605615|
|         2|           9|         3|         18| 0.0921475887298584|
195.33880645286484|
+----------+------------+----------+-----------+-------------------
+------------------+
only showing top 20 rows


###Coding task###

#ii) get a Spark context and create the dataset and run timing test
for each combination in parallel (2%)
#Reference: https://pypi.org/project/schema/

# Define the schema for time_list
schema = StructType([
    StructField("batch_size", IntegerType(), True),
    StructField("batch_number", IntegerType(), True),
    StructField("repetition", IntegerType(), True),
    StructField("datasetsize", IntegerType(), True),
    StructField("reading_speed", DoubleType(), True),
    StructField("throughput", DoubleType(), True),
])

# Convert RDD to DataFrame using the specified schema
df_results = spark.createDataFrame(rdd_parameters.flatMap(lambda x:
time_configs_rdd(x)[1]), schema=schema)

# Show the DataFrame
df_results.show()

+----------+------------+----------+-----------+-------------------
+------------------+
|batch_size|batch_number|repetition|datasetsize|      reading_speed|
throughput|
+----------+------------+----------+-----------+-------------------
+------------------+
|         2|           6|         1|         12|0.10601377487182617|
113.1928375770824|
|         2|           6|         1|         12| 0.4522225856781006|
26.535605208674376|
|         2|           6|         2|         12|0.09072279930114746|
132.27105085422804|
|         2|           6|         2|         12|0.09862756729125977|
121.66983663464467|
|         2|           6|         2|         12| 0.6394572257995605|
18.765915085243606|
|         2|           6|         2|         12| 0.6724894046783447|
17.844147307777533|
```

```
|         2|           6|         3|         12|0.17529773712158203|
68.45496238024514|
|         2|           6|         3|         12|0.15856575965881348|
75.67838117018734|
|         2|           6|         3|         12|0.15911555290222168|
75.41688905404575|
|         2|           6|         3|         12| 0.6878738403320312|
17.44505939375118|
|         2|           6|         3|         12| 0.5629227161407471|
21.31731347114379|
|         2|           6|         3|         12| 0.5063719749450684|
23.697993952571665|
|         2|           9|         1|         18|0.10285067558288574|
175.01100404045556|
|         2|           9|         1|         18| 0.6729984283447266|
26.745976278535903|
|         2|           9|         2|         18|0.10020780563354492|
179.6267255449653|
|         2|           9|         2|         18|0.09325480461120605|
193.01954548127392|
|         2|           9|         2|         18| 0.6931228637695312|
25.969421787801735|
|         2|           9|         2|         18| 0.6134324073791504|
29.34308618761082|
|         2|           9|         3|         18|0.09595561027526855|
187.58673878852179|
|         2|           9|         3|         18|0.09620189666748047|
187.10649814126393|
+----------+------------+----------+-----------+-------------------
+------------------+
only showing top 20 rows


###Coding Task###

# iii) Transform the resulting RDD to the structure
(parameter_combination, images_per_second) and save these values in an
array

# Transform the resulting DataFrame to include dataset size along with
throughput

parameter_dataset_throughput_array = df_results.rdd.map(lambda z:
((z['batch_size'], z['batch_number'], z['repetition']),
(z['datasetsize'], z['throughput']))).collect()



# Show the transformed RDD as an array
```

```
parameter_dataset_throughput_array


[((2, 6, 1), (12, 99.00963704212246)),
 ((2, 6, 1), (12, 12.99391270741367)),
 ((2, 6, 2), (12, 26.956735133710776)),
 ((2, 6, 2), (12, 66.3122330404079)),
 ((2, 6, 2), (12, 12.887587455024228)),
 ((2, 6, 2), (12, 14.064694357903313)),
 ((2, 6, 3), (12, 33.62414606283695)),
 ((2, 6, 3), (12, 76.19126970190555)),
 ((2, 6, 3), (12, 62.43676267363997)),
 ((2, 6, 3), (12, 19.54662784136254)),
 ((2, 6, 3), (12, 23.86318069236517)),
 ((2, 6, 3), (12, 21.14188838389354)),
 ((2, 9, 1), (18, 102.09273034851974)),
 ((2, 9, 1), (18, 18.128898624126602)),
 ((2, 9, 2), (18, 100.2251118111616)),
 ((2, 9, 2), (18, 170.40587750679953)),
 ((2, 9, 2), (18, 19.713456490422466)),
 ((2, 9, 2), (18, 23.57393877814838)),
 ((2, 9, 3), (18, 103.19219266832509)),
 ((2, 9, 3), (18, 112.7081954291191)),
 ((2, 9, 3), (18, 170.51518087636444)),
 ((2, 9, 3), (18, 23.05876057607905)),
 ((2, 9, 3), (18, 26.167471184490314)),
 ((2, 9, 3), (18, 22.98284448251708)),
 ((2, 12, 1), (24, 148.54214188217065)),
 ((2, 12, 1), (24, 16.083265962430364)),
 ((2, 12, 2), (24, 130.44119453579953)),
 ((2, 12, 2), (24, 140.57156263091747)),
 ((2, 12, 2), (24, 16.868858887382185)),
 ((2, 12, 2), (24, 17.235446190134)),
 ((2, 12, 3), (24, 119.73177914694197)),
 ((2, 12, 3), (24, 156.80251723197944)),
 ((2, 12, 3), (24, 134.34383207704298)),
 ((2, 12, 3), (24, 22.483171341126617)),
 ((2, 12, 3), (24, 25.847433751750536)),
 ((2, 12, 3), (24, 25.763729306958922)),
 ((2, 15, 1), (30, 221.5348211581935)),
 ((2, 15, 1), (30, 26.224917284535987)),
 ((2, 15, 2), (30, 179.185818830663)),
 ((2, 15, 2), (30, 188.87050654514508)),
 ((2, 15, 2), (30, 26.746857493285606)),
 ((2, 15, 2), (30, 27.681934699832716)),
 ((2, 15, 3), (30, 211.42208093340233)),
 ((2, 15, 3), (30, 242.18120508270363)),
 ((2, 15, 3), (30, 190.11296865675024)),
 ((2, 15, 3), (30, 26.777411819384955)),
 ((2, 15, 3), (30, 19.754547675311652)),
```

```
((2, 15, 3), (30, 17.721986938340912)),
((4, 6, 1), (24, 123.8336948004099)),
((4, 6, 1), (24, 13.585897689906167)),
((4, 6, 2), (24, 62.27572779976405)),
((4, 6, 2), (24, 67.87398480605708)),
((4, 6, 2), (24, 15.533567517420297)),
((4, 6, 2), (24, 16.22890355952077)),
((4, 6, 3), (24, 107.01916415942846)),
((4, 6, 3), (24, 95.5628215266577)),
((4, 6, 3), (24, 72.14382999095544)),
((4, 6, 3), (24, 14.928115159534002)),
((4, 6, 3), (24, 15.876938043642328)),
((4, 6, 3), (24, 16.08086111102236)),
((4, 9, 1), (36, 155.4579873590922)),
((4, 9, 1), (36, 16.833942645840338)),
((4, 9, 2), (36, 180.8950941045393)),
((4, 9, 2), (36, 125.45244902181201)),
((4, 9, 2), (36, 17.767597688581713)),
((4, 9, 2), (36, 18.887164053646345)),
((4, 9, 3), (36, 151.42954679931404)),
((4, 9, 3), (36, 106.77993544901406)),
((4, 9, 3), (36, 186.43698836149738)),
((4, 9, 3), (36, 16.22797652818652)),
((4, 9, 3), (36, 16.93529189625878)),
((4, 9, 3), (36, 20.686111989044488)),
((4, 12, 1), (48, 144.85198844790693)),
((4, 12, 1), (48, 24.19304000613338)),
((4, 12, 2), (48, 246.87776076008197)),
((4, 12, 2), (48, 216.6111047751437)),
((4, 12, 2), (48, 28.23918247634597)),
((4, 12, 2), (48, 29.652804890873533)),
((4, 12, 3), (48, 202.6070686740198)),
((4, 12, 3), (48, 192.3829032125517)),
((4, 12, 3), (48, 150.53479676000387)),
((4, 12, 3), (48, 17.894419802746523)),
((4, 12, 3), (48, 19.87840250492056)),
((4, 12, 3), (48, 21.02877302505791)),
((4, 15, 1), (60, 167.96016340957647)),
((4, 15, 1), (60, 22.37372417715355)),
((4, 15, 2), (60, 168.1764562007065)),
((4, 15, 2), (60, 176.7298912688531)),
((4, 15, 2), (60, 21.5076023658699)),
((4, 15, 2), (60, 21.410428430212242)),
((4, 15, 3), (60, 167.7798778743822)),
((4, 15, 3), (60, 176.32621488742888)),
((4, 15, 3), (60, 175.11557318985902)),
((4, 15, 3), (60, 30.469817948647478)),
((4, 15, 3), (60, 31.07727101469123)),
((4, 15, 3), (60, 30.05156184788664)),
((6, 6, 1), (36, 102.28311967187018)),
```

```
((6, 6, 1), (36, 18.7624196776906)),
((6, 6, 2), (36, 217.71564519524412)),
((6, 6, 2), (36, 182.83911919451222)),
((6, 6, 2), (36, 24.81269795362335)),
((6, 6, 2), (36, 27.318009648103306)),
((6, 6, 3), (36, 303.2891788421598)),
((6, 6, 3), (36, 290.5749419792857)),
((6, 6, 3), (36, 396.60782315426724)),
((6, 6, 3), (36, 28.019591360472795)),
((6, 6, 3), (36, 27.961845432483308)),
((6, 6, 3), (36, 27.19108880366728)),
((6, 9, 1), (54, 255.06735692944926)),
((6, 9, 1), (54, 27.21348449878833)),
((6, 9, 2), (54, 154.1285836095605)),
((6, 9, 2), (54, 164.39667014826682)),
((6, 9, 2), (54, 18.9839003590969)),
((6, 9, 2), (54, 19.97300300319367)),
((6, 9, 3), (54, 155.62627829109067)),
((6, 9, 3), (54, 161.86585480897045)),
((6, 9, 3), (54, 168.60544557993282)),
((6, 9, 3), (54, 26.30199721874646)),
((6, 9, 3), (54, 28.342310492461944)),
((6, 9, 3), (54, 29.674144189987963)),
((6, 12, 1), (72, 247.45987697062913)),
((6, 12, 1), (72, 30.271185577602584)),
((6, 12, 2), (72, 300.14996869191856)),
((6, 12, 2), (72, 282.5432815600654)),
((6, 12, 2), (72, 21.2170826841239)),
((6, 12, 2), (72, 19.720265072745725)),
((6, 12, 3), (72, 170.24180106072086)),
((6, 12, 3), (72, 154.14552437265266)),
((6, 12, 3), (72, 175.77254503885752)),
((6, 12, 3), (72, 20.51063325168877)),
((6, 12, 3), (72, 20.085264670465925)),
((6, 12, 3), (72, 20.96376777337547)),
((6, 15, 1), (90, 206.7365852215907)),
((6, 15, 1), (90, 20.324532542981366)),
((6, 15, 2), (90, 212.7743034437899)),
((6, 15, 2), (90, 213.379277832514)),
((6, 15, 2), (90, 19.130033242269636)),
((6, 15, 2), (90, 22.824003626326565)),
((6, 15, 3), (90, 295.0327790439211)),
((6, 15, 3), (90, 350.61060300541396)),
((6, 15, 3), (90, 313.66838947700796)),
((6, 15, 3), (90, 29.81131530110871)),
((6, 15, 3), (90, 28.451096805780733)),
((6, 15, 3), (90, 21.21999434489295)),
((8, 6, 1), (48, 133.58684187477274)),
((8, 6, 1), (48, 21.322068126288766)),
((8, 6, 2), (48, 142.56460021328755)),
```

```
 ((8, 6, 2), (48, 160.19914587655074)),
 ((8, 6, 2), (48, 20.122690419166346)),
 ((8, 6, 2), (48, 21.792721975777013)),
 ((8, 6, 3), (48, 127.7076192620832)),
 ((8, 6, 3), (48, 147.81785782725436)),
 ((8, 6, 3), (48, 139.4585814924877)),
 ((8, 6, 3), (48, 19.9691913992061)),
 ((8, 6, 3), (48, 24.30539186090605)),
 ((8, 6, 3), (48, 30.798985374020564)),
 ((8, 9, 1), (72, 264.9824403769545)),
 ((8, 9, 1), (72, 29.230390131334993)),
 ((8, 9, 2), (72, 279.6580722949584)),
 ((8, 9, 2), (72, 264.8165321201641)),
 ((8, 9, 2), (72, 36.30494287911203)),
 ((8, 9, 2), (72, 36.14530323932607)),
 ((8, 9, 3), (72, 262.82844908616187)),
 ((8, 9, 3), (72, 288.561670656022)),
 ((8, 9, 3), (72, 267.3771179500165)),
 ((8, 9, 3), (72, 28.292729377633737)),
 ((8, 9, 3), (72, 29.720196435425297)),
 ((8, 9, 3), (72, 32.09413802195504)),
 ((8, 12, 1), (96, 354.96098148822375)),
 ((8, 12, 1), (96, 31.61781451290576)),
 ((8, 12, 2), (96, 352.25562016707624)),
 ((8, 12, 2), (96, 356.88705830423925)),
 ((8, 12, 2), (96, 31.441144306488333)),
 ((8, 12, 2), (96, 38.768818735517776)),
 ((8, 12, 3), (96, 386.35888440893746)),
 ((8, 12, 3), (96, 436.62240728692257)),
 ((8, 12, 3), (96, 405.2194387053208)),
 ((8, 12, 3), (96, 36.55900653262923)),
 ((8, 12, 3), (96, 39.2710002366092)),
 ((8, 12, 3), (96, 32.32066599069262)),
 ((8, 15, 1), (120, 351.043841729258)),
 ((8, 15, 1), (120, 31.038738135610362)),
 ((8, 15, 2), (120, 392.2620065419024)),
 ((8, 15, 2), (120, 383.4441651049047)),
 ((8, 15, 2), (120, 30.465799129032327)),
 ((8, 15, 2), (120, 32.966574245820155)),
 ((8, 15, 3), (120, 472.4091137339559)),
 ((8, 15, 3), (120, 523.0033729651543)),
 ((8, 15, 3), (120, 506.37396600674873)),
 ((8, 15, 3), (120, 38.56898944816733)),
 ((8, 15, 3), (120, 40.22376242656187)),
 ((8, 15, 3), (120, 34.35089646557024))]

#### Coding Task###
#Reference:https://github.com/vighnesh32/Big-Data-Project/blob/main/
project.ipynb
#iv) create an RDD with all results for each parameter as
```

```
(parameter_value,images_per_second) and collect the result for each
parameter (2%)


#Extracting batch size and throughput
rdd_tfrecord_batch_sizes_speed = df_results.rdd.map(lambda z:
(int(z['batch_size']), float(z['throughput'])))
# Collect batch size results for TFRecord dataset
tfrecord_batch_sizes_speed = rdd_tfrecord_batch_sizes_speed.collect()

#Extracting  batch numbers and throughput
rdd_tfrecord_batch_nums_speed = df_results.rdd.map(lambda z:
(int(z['batch_number']), float(z['throughput'])))
tfrecord_batch_nums_speed = rdd_tfrecord_batch_nums_speed.collect()

# Extracting repetitions and throughput
rdd_tfrecord_repetitions_speed = df_results.rdd.map(lambda z:
(int(z['repetition']), float(z['throughput'])))
tfrecord_repetitions_speed = rdd_tfrecord_repetitions_speed.collect()

# Extracting  dataset size and throughput
rdd_tfrecord_datasetsize_speed = df_results.rdd.map(lambda z:
(int(z['datasetsize']), float(z['throughput'])))
tfrecord_datasetsize_speed = rdd_tfrecord_datasetsize_speed.collect()


###Coding Task###
#Reference:
https://github.com/vighnesh32/Big-Data-Project/blob/main/project.ipynb
#v) create an RDD with the average reading speeds for each parameter
value and collect the results.
#Keep associativity in mind when implementing the average. (3%)


#Average speed for batch size
rdd_tfrecord_batch_sizes_avg_speed = rdd_tfrecord_batch_sizes_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_batch_sizes_avg_speed =
rdd_tfrecord_batch_sizes_avg_speed.collect()

#Average speed For batch numbers
rdd_tfrecord_batch_nums_avg_speed = rdd_tfrecord_batch_nums_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_batch_nums_avg_speed =
rdd_tfrecord_batch_nums_avg_speed.collect()
```

```
#Average speed For repetitions
rdd_tfrecord_repetitions_avg_speed = rdd_tfrecord_repetitions_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_repetitions_avg_speed =
rdd_tfrecord_repetitions_avg_speed.collect()

#Average speed for dataset sizes
rdd_tfrecord_datasetsize_avg_speed = rdd_tfrecord_datasetsize_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_datasetsize_avg_speed =
rdd_tfrecord_datasetsize_avg_speed.collect()



###Coding Task####

# vi) write the results to a pickle file in your bucket (1%)

#Function to save object to file and upload it to filename which
contains directory to cloud storage bucket
def save(object,bucket,filename):
    with open(filename, mode='wb') as f:
      pickle.dump(object,f)
    print("Saving{} to {}".format(filename,bucket))
    import subprocess

proc=subprocess.run(["gsutil","cp",filename,bucket],stderr=subprocess.
PIPE)
    print("gstuil returned: " + str(proc.returncode))
    print(str(proc.stderr))

#filename
filename="2aPickle.pkl"

#dumping objects with pickle
with open(filename,mode='wb') as f:
      pickle.dump(tfrecord_batch_sizes_speed,f)
      pickle.dump(tfrecord_batch_nums_speed,f)
      pickle.dump(tfrecord_repetitions_speed,f)
      pickle.dump(tfrecord_datasetsize_speed,f)
      pickle.dump(tfrecord_batch_sizes_avg_speed,f)
      pickle.dump(tfrecord_batch_nums_avg_speed,f)
      pickle.dump(tfrecord_repetitions_avg_speed,f)
      pickle.dump(tfrecord_datasetsize_avg_speed,f)
```

```
print("Saving {} to {}".format(filename,BUCKET))
import subprocess
proc = subprocess.run(["gsutil", "cp",filename, BUCKET],
stderr=subprocess.PIPE)
print("gstuil returned: " +str(proc.returncode))
print(str(proc.stderr))
```

```
Saving 2aPickle.pkl to gs://bigdata-421920-storage
gstuil returned: 0
b'Copying file://2aPickle.pkl [Content-Type=application/octet-
stream]...\n/ [0 files][    0.0 B/ 10.2 KiB]
\r/ [1 files][ 10.2 KiB/ 10.2 KiB]
\r-\r\nOperation completed over 1 objects/10.2 KiB.
\n'
```

```
###Coding Task####

# vii) Write your code it into a file using the cell magic %%writefile
spark_job.py (1%)

%%writefile 2Aspark.py


#Required libraries
import os, sys, math
os.environ['PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION'] = 'python'
import numpy as np
import scipy as sp
import scipy.stats
import time
import datetime
import string
import random
from matplotlib import pyplot as plt
import tensorflow as tf
print("Tensorflow version " + tf.__version__)
import pickle
import pyspark
from pyspark.sql import SQLContext
from pyspark.sql import Row
from pyspark.sql import SparkSession
import pandas as pd

from pyspark.sql.types import StructType, StructField, IntegerType,
DoubleType, StringType



#Required variables
PROJECT = 'bigdata-421920'
```

```
BUCKET = 'gs://{}-storage'.format(PROJECT)
REGION = 'us-central1'
CLUSTER = '{}-cluster'.format(PROJECT)
GCS_PATTERN = 'gs://flowers-public/*/*.jpg'
PARTITIONS = 16 # no of partitions we will use later
GCS_OUTPUT = BUCKET + '/tfrecords-jpeg-192x192-2/flowers'
CLASSES = [b'daisy', b'dandelion', b'roses', b'sunflowers', b'tulips']
TARGET_SIZE = [192, 192] # target resolution for the images


#
#i) combine the previous cells to have the code to create a dataset
and create a list of parameter combinations in an RDD (2%)


#Function to parse TFRecord examples
def read_tfrecord(example):
    features = {
        "image": tf.io.FixedLenFeature([], tf.string),  # tf.string =
bytestring (not text string)
        "class": tf.io.FixedLenFeature([], tf.int64) #,   # shape []
means scalar
    }
    # decode the TFRecord
    # Parse a single TFRecord example using the specified features
    example = tf.io.parse_single_example(example, features)
    #Decoing image from TFRrecord example
    image = tf.image.decode_jpeg(example['image'], channels=3)
    #reshaping image to target size
    image = tf.reshape(image, [*TARGET_SIZE, 3])
    #Extracting class label
    class_num = example['class']
    #output will be parsed image and class label
    return image, class_num

#Function to load dataset from TFRecord files
def load_dataset(filenames):
    # read from TFRecords. For optimal performance, read from multiple
    # TFRecord files at once and set the option
experimental_deterministic = False
    # to allow order-altering optimizations.

    option_no_order = tf.data.Options()
    option_no_order.experimental_deterministic = False
    #dataset
    dataset = tf.data.TFRecordDataset(filenames)
    dataset = dataset.with_options(option_no_order)
    dataset = dataset.map(read_tfrecord)
    return dataset
```

```
def decode_jpeg_and_label(filepath):
    # extracts the image data and creates a class label, based on the
filepath
    bits = tf.io.read_file(filepath)
    image = tf.image.decode_jpeg(bits)
    # parse flower name from containing directory
    label = tf.strings.split(tf.expand_dims(filepath, axis=-1),
sep='/')
    label2 = label.values[-2]
    return image, label2

#Function to resize and crop the image
def resize_and_crop_image(image, label):
    # Resizes and cropd using "fill" algorithm:
    # always make sure the resulting image is cut out from the source
image
    # so that it fills the TARGET_SIZE entirely with no black bars
    # and a preserved aspect ratio.
    #Dimensions of the image
    w = tf.shape(image)[0]
    h = tf.shape(image)[1]
    tw = TARGET_SIZE[1]
    th = TARGET_SIZE[0]
    #Resize image
    resize_crit = (w * th) / (h * tw)
    image = tf.cond(resize_crit < 1,
                    lambda: tf.image.resize(image, [w*tw/w, h*tw/w]),
# if true
                    lambda: tf.image.resize(image, [w*th/h, h*th/h])
# if false
                    )
    nw = tf.shape(image)[0]
    nh = tf.shape(image)[1]
    #Cropped image is resized to target size
    image = tf.image.crop_to_bounding_box(image, (nw - tw) // 2, (nh -
th) // 2, tw, th)
    #Return the new image (resized and cropped) and label
    return image, label


# Function to load dataset from TFRecord files and decode
def load_dataset_decoded():
    #Obtaining list of TFRecord files
    dataset_filename = tf.data.Dataset.list_files(GCS_PATTERN)
    datasetDecoded = dataset_filename.map(decode_jpeg_and_label)
    datasetfn = datasetDecoded.map(resize_and_crop_image)
    return datasetfn


# Define the new time_configs function
```

```
def time_configs_rdd(parameters_rdd):
    #Timer
    start = time.time()
    #Extracting parameters from RDD
    batch_size = parameters_rdd[0]
    batch_number = parameters_rdd[1]
    repetition = parameters_rdd[2]
    dataset_type = parameters_rdd[3]

    #Loading dataset based on type
    if dataset_type == 'datasetDecoded':
        filenames = tf.io.gfile.glob(GCS_OUTPUT + "*.tfrec")
        dataset = load_dataset(filenames)
    else:
        filenames_fn = tf.data.Dataset.list_files(GCS_PATTERN)
        dataset_fn = filenames_fn.map(decode_jpeg_and_label)
        dataset = dataset_fn.map(resize_and_crop_image)

    #Batch of dataset
    dataset1 = dataset.batch(batch_size)
    test_set = dataset1.take(batch_number)
    time_list = []

    #Time test
    for _ in range(repetition):
        s_time = time.time()
        for _ in test_set:
            print('string', file=open("/dev/null", mode='w'))
        e_time = time.time()
        #Reading speed is calculated
        reading_speed = e_time - s_time
        #Throughput is calculated
        throughput = float((batch_size * batch_number) / (e_time -
s_time))
        datasetsize = batch_size * batch_number
        #Times values are appended to the list
        time_list.append([batch_size, batch_number, repetition,
datasetsize, reading_speed, throughput])

    end = time.time()
    #calculating total time and toal images
    total_images = batch_size * batch_number * repetition
    total_time = total_images / (end - start)
    return total_time, time_list


# Define the parameter combinations, for which will be used
batch_sizes = [2, 4, 6, 8]
batch_numbers = [6, 9, 12, 15]
repetitions = [1, 2, 3]
```

```
dataset_types = ['datasetDecoded', 'otherDataset']

#list to store parameter combinations
parameter_list = []
for batch_size in batch_sizes:
    for batch_number in batch_numbers:
        for repetition in repetitions:
          for dataset_type in dataset_types:
              parameter_list.append([batch_size, batch_number,
repetition, dataset_type])


# Define the columns for DataFrame
columns = ["batch_size", "batch_number", "repetition", "dataset_type",
"datasetsize", "reading_speed", "throughput"]

# Create Spark session
spark =
SparkSession.builder.master("local").appName("DataProcessing").getOrCr
eate()

# Create RDD for parameter combinations
rdd_parameters = spark.sparkContext.parallelize(parameter_list)




#
#ii) get a Spark context and create the dataset and run timing test
for each combination in parallel (2%)
#Reference: https://pypi.org/project/schema/

# Define the schema for time_list
schema = StructType([
    StructField("batch_size", IntegerType(), True),
    StructField("batch_number", IntegerType(), True),
    StructField("repetition", IntegerType(), True),
    StructField("datasetsize", IntegerType(), True),
    StructField("reading_speed", DoubleType(), True),
    StructField("throughput", DoubleType(), True),
])

# Convert RDD to DataFrame using the specified schema
df_results = spark.createDataFrame(rdd_parameters.flatMap(lambda x:
time_configs_rdd(x)[1]), schema=schema)


#
# iii) Transform the resulting RDD to the structure
(parameter_combination, images_per_second) and save these values in an
```

```
array

# Transform the resulting DataFrame to include dataset size along with
throughput
# Update the lambda function to correctly access DataFrame columns and
handle data types
parameter_dataset_throughput_array = df_results.rdd.map(lambda z:
((z['batch_size'], z['batch_number'], z['repetition']),
(z['datasetsize'], z['throughput']))).collect()


#
#### Coding Task###
#Reference:https://github.com/vighnesh32/Big-Data-Project/blob/main/
project.ipynb
#iv) create an RDD with all results for each parameter as
(parameter_value,images_per_second) and collect the result for each
parameter (2%)


#Extracting batch size and throughput
rdd_tfrecord_batch_sizes_speed = df_results.rdd.map(lambda z:
(int(z['batch_size']), float(z['throughput'])))
# Collect batch size results for TFRecord dataset
tfrecord_batch_sizes_speed = rdd_tfrecord_batch_sizes_speed.collect()

#Extracting  batch numbers and throughput
rdd_tfrecord_batch_nums_speed = df_results.rdd.map(lambda z:
(int(z['batch_number']), float(z['throughput'])))
tfrecord_batch_nums_speed = rdd_tfrecord_batch_nums_speed.collect()

# Extracting repetitions and throughput
rdd_tfrecord_repetitions_speed = df_results.rdd.map(lambda z:
(int(z['repetition']), float(z['throughput'])))
tfrecord_repetitions_speed = rdd_tfrecord_repetitions_speed.collect()

# Extracting  dataset size and throughput
rdd_tfrecord_datasetsize_speed = df_results.rdd.map(lambda z:
(int(z['datasetsize']), float(z['throughput'])))
tfrecord_datasetsize_speed = rdd_tfrecord_datasetsize_speed.collect()

#
###Coding Task###
#Reference:
https://github.com/vighnesh32/Big-Data-Project/blob/main/project.ipynb
#v) create an RDD with the average reading speeds for each parameter
value and collect the results.
#Keep associativity in mind when implementing the average. (3%)
```

```
#Average speed for batch size
rdd_tfrecord_batch_sizes_avg_speed = rdd_tfrecord_batch_sizes_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_batch_sizes_avg_speed =
rdd_tfrecord_batch_sizes_avg_speed.collect()

#Average speed For batch numbers
rdd_tfrecord_batch_nums_avg_speed = rdd_tfrecord_batch_nums_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_batch_nums_avg_speed =
rdd_tfrecord_batch_nums_avg_speed.collect()

#Average speed For repetitions
rdd_tfrecord_repetitions_avg_speed = rdd_tfrecord_repetitions_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_repetitions_avg_speed =
rdd_tfrecord_repetitions_avg_speed.collect()

#Average speed for dataset sizes
rdd_tfrecord_datasetsize_avg_speed = rdd_tfrecord_datasetsize_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_datasetsize_avg_speed =
rdd_tfrecord_datasetsize_avg_speed.collect()


#
# vi) write the results to a pickle file in your bucket (1%)

#Function to save object to file and upload it to filename which
contains directory to cloud storage bucket
def save(object,bucket,filename):
    with open(filename, mode='wb') as f:
      pickle.dump(object,f)
    print("Saving{} to {}".format(filename,bucket))
    import subprocess

proc=subprocess.run(["gsutil","cp",filename,bucket],stderr=subprocess.
PIPE)
    print("gstuil returned: " + str(proc.returncode))
    print(str(proc.stderr))

#filename
```

```
filename="2aPickle.pkl"

#dumping objects with pickle
with open(filename,mode='wb') as f:
      pickle.dump(tfrecord_batch_sizes_speed,f)
      pickle.dump(tfrecord_batch_nums_speed,f)
      pickle.dump(tfrecord_repetitions_speed,f)
      pickle.dump(tfrecord_datasetsize_speed,f)
      pickle.dump(tfrecord_batch_sizes_avg_speed,f)
      pickle.dump(tfrecord_batch_nums_avg_speed,f)
      pickle.dump(tfrecord_repetitions_avg_speed,f)
      pickle.dump(tfrecord_datasetsize_avg_speed,f)

print("Saving {} to {}".format(filename,BUCKET))
import subprocess
proc = subprocess.run(["gsutil", "cp",filename, BUCKET],
stderr=subprocess.PIPE)
print("gstuil returned: " +str(proc.returncode))
print(str(proc.stderr))


Overwriting 2Aspark.py
```

2b) Testing the code and collecting results (4%)

i) First, test locally with `%run`.

It is useful to create a **new filename argument**, so that old results don't get overwritten.

You can for instance use `datetime.datetime.now().strftime("%y%m%d-%H%M")` to get a string with the current date and time and use that in the file name.

```
### CODING TASK
#Running the script locally
%run 2Aspark.py

Tensorflow version 2.15.0
Saving 2aPickle.pkl to gs://bigdata-421920-storage
gstuil returned: 0
b'Copying file://2aPickle.pkl [Content-Type=application/octet-
stream]...\n/ [0 files][    0.0 B/ 10.2 KiB]
\r/ [1 files][ 10.2 KiB/ 10.2 KiB]
\r-\r\nOperation completed over 1 objects/10.2 KiB.
\n'
```

ii) Cloud

If you have a cluster running, you can run the speed test job in the cloud.

While you run this job, switch to the Dataproc web page and take **screenshots of the CPU and network load** over time. They are displayed with some delay, so you may need to wait a little.

These images will be useful in the next task. Again, don't use the SCREENSHOT function that Google provides, but just take a picture of the graphs you see for the VMs.

```
### CODING TASK ###

#Running ths script on a cluster
!gcloud dataproc jobs submit pyspark --cluster eightfold-cluster \
2Aspark.py

Job [79c44c8ef32a4c85a55c86356d090fbf] submitted.
Waiting for job output...
2024-05-05 08:30:19.692296: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
not load dynamic library 'libcudart.so.11.0'; dlerror:
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 08:30:19.692334: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
Tensorflow version 2.4.0
24/05/05 08:30:22 INFO org.apache.spark.SparkEnv: Registering
MapOutputTracker
24/05/05 08:30:22 INFO org.apache.spark.SparkEnv: Registering
BlockManagerMaster
24/05/05 08:30:22 INFO org.apache.spark.SparkEnv: Registering
OutputCommitCoordinator
24/05/05 08:30:22 INFO org.spark_project.jetty.util.log: Logging
initialized @6047ms to org.spark_project.jetty.util.log.Slf4jLog
24/05/05 08:30:22 INFO org.spark_project.jetty.server.Server: jetty-
9.4.z-SNAPSHOT; built: unknown; git: unknown; jvm 1.8.0_382-b05
24/05/05 08:30:22 INFO org.spark_project.jetty.server.Server: Started
@6184ms
24/05/05 08:30:22 INFO
org.spark_project.jetty.server.AbstractConnector: Started
ServerConnector@3e6b8642{HTTP/1.1, (http/1.1)}{0.0.0.0:46695}
2024-05-05 08:30:29.000480: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
not load dynamic library 'libcudart.so.11.0'; dlerror:
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 08:30:29.000604: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
24/05/05 08:30:29 ERROR org.apache.spark.api.python.PythonRunner:
Python worker exited unexpectedly (crashed)
org.apache.spark.api.python.PythonException: Traceback (most recent
call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
362, in main
    eval_type = read_int(infile)
```

```
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 724, in read_int
    raise EOFError
EOFError

      at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePyth
onException(PythonRunner.scala:457)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:592)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:575)
      at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(Py
thonRunner.scala:410)
      at
org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.s
cala:37)
      at scala.collection.Iterator.foreach(Iterator.scala:941)
      at scala.collection.Iterator.foreach$(Iterator.scala:941)
      at
org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.s
cala:28)
      at scala.collection.generic.Growable.
$plus$plus$eq(Growable.scala:62)
      at scala.collection.generic.Growable.$plus$plus$eq$
(Growable.scala:53)
      at scala.collection.mutable.ArrayBuffer.
$plus$plus$eq(ArrayBuffer.scala:105)
      at scala.collection.mutable.ArrayBuffer.
$plus$plus$eq(ArrayBuffer.scala:49)
      at scala.collection.TraversableOnce.to(TraversableOnce.scala:315)
      at scala.collection.TraversableOnce.to$
(TraversableOnce.scala:313)
      at
org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:
28)
      at
scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:307)
      at scala.collection.TraversableOnce.toBuffer$
(TraversableOnce.scala:307)
      at
org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.
scala:28)
      at
scala.collection.TraversableOnce.toArray(TraversableOnce.scala:294)
      at scala.collection.TraversableOnce.toArray$
(TraversableOnce.scala:288)
      at
```

```
org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.s
cala:28)
     at org.apache.spark.rdd.RDD.$anonfun$collect$2(RDD.scala:990)
     at org.apache.spark.SparkContext.
$anonfun$runJob$5(SparkContext.scala:2116)
     at
org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
     at org.apache.spark.scheduler.Task.run(Task.scala:123)
     at org.apache.spark.executor.Executor$TaskRunner.
$anonfun$run$3(Executor.scala:414)
     at
org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360)
     at
org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:417)
     at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.j
ava:1149)
     at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.
java:624)
     at java.lang.Thread.run(Thread.java:750)
Caused by: org.apache.spark.api.python.PythonException: Traceback
(most recent call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
```

```
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File
"/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descri
ptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of
date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible
workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will
use pure-Python parsing and will be much slower).

More information:
https://developers.google.com/protocol-buffers/docs/news/2022-05-
06#python-updates

     at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePyth
onException(PythonRunner.scala:457)
     at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:592)
     at org.apache.spark.api.python.PythonRunner$
```

```
$anon$3.read(PythonRunner.scala:575)
     at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(Py
thonRunner.scala:410)
     at
org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.s
cala:37)
     at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
     at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.hasNext(SerDe
Util.scala:153)
     at scala.collection.Iterator.foreach(Iterator.scala:941)
     at scala.collection.Iterator.foreach$(Iterator.scala:941)
     at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.foreach(SerDe
Util.scala:148)
     at
org.apache.spark.api.python.PythonRDD$.writeIteratorToStream(PythonRDD
.scala:224)
     at org.apache.spark.api.python.PythonRunner$
$anon$2.writeIteratorToStream(PythonRunner.scala:561)
     at org.apache.spark.api.python.BasePythonRunner$WriterThread.
$anonfun$run$1(PythonRunner.scala:346)
     at
org.apache.spark.util.Utils$.logUncaughtExceptions(Utils.scala:1945)
     at
org.apache.spark.api.python.BasePythonRunner$WriterThread.run(PythonRu
nner.scala:196)
24/05/05 08:30:29 ERROR org.apache.spark.api.python.PythonRunner: This
may have been caused by a prior exception:
org.apache.spark.api.python.PythonException: Traceback (most recent
call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
```

```
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File
"/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descri
ptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of
date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible
workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
```

```
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will
use pure-Python parsing and will be much slower).

More information:
https://developers.google.com/protocol-buffers/docs/news/2022-05-
06#python-updates

      at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePyth
onException(PythonRunner.scala:457)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:592)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:575)
      at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(Py
thonRunner.scala:410)
      at
org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.s
cala:37)
      at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.hasNext(SerDe
Util.scala:153)
      at scala.collection.Iterator.foreach(Iterator.scala:941)
      at scala.collection.Iterator.foreach$(Iterator.scala:941)
      at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.foreach(SerDe
Util.scala:148)
      at
org.apache.spark.api.python.PythonRDD$.writeIteratorToStream(PythonRDD
.scala:224)
      at org.apache.spark.api.python.PythonRunner$
$anon$2.writeIteratorToStream(PythonRunner.scala:561)
      at org.apache.spark.api.python.BasePythonRunner$WriterThread.
$anonfun$run$1(PythonRunner.scala:346)
      at
org.apache.spark.util.Utils$.logUncaughtExceptions(Utils.scala:1945)
      at
org.apache.spark.api.python.BasePythonRunner$WriterThread.run(PythonRu
nner.scala:196)
24/05/05 08:30:29 ERROR org.apache.spark.executor.Executor: Exception
in task 0.0 in stage 0.0 (TID 0)
org.apache.spark.api.python.PythonException: Traceback (most recent
call last):
```

```
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File
```

```
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framework/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File "/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descriptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will use pure-Python parsing and will be much slower).

More information: https://developers.google.com/protocol-buffers/docs/news/2022-05-06#python-updates

      at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:457)
      at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:592)
      at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:575)
      at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:410)
      at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
      at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.hasNext(SerDeUtil.scala:153)
      at scala.collection.Iterator.foreach(Iterator.scala:941)
      at scala.collection.Iterator.foreach$(Iterator.scala:941)
      at org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.foreach(SerDeUtil.scala:148)
      at org.apache.spark.api.python.PythonRDD$.writeIteratorToStream(PythonRDD.scala:224)
```

```
     at org.apache.spark.api.python.PythonRunner$
$anon$2.writeIteratorToStream(PythonRunner.scala:561)
     at org.apache.spark.api.python.BasePythonRunner$WriterThread.
$anonfun$run$1(PythonRunner.scala:346)
     at
org.apache.spark.util.Utils$.logUncaughtExceptions(Utils.scala:1945)
     at
org.apache.spark.api.python.BasePythonRunner$WriterThread.run(PythonRu
nner.scala:196)
24/05/05 08:30:29 WARN org.apache.spark.scheduler.TaskSetManager: Lost
task 0.0 in stage 0.0 (TID 0, localhost, executor driver):
org.apache.spark.api.python.PythonException: Traceback (most recent
call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
```

```
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File
"/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descri
ptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of
date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible
workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will
use pure-Python parsing and will be much slower).

More information:
https://developers.google.com/protocol-buffers/docs/news/2022-05-
06#python-updates

     at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePyth
onException(PythonRunner.scala:457)
     at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:592)
     at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:575)
     at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(Py
thonRunner.scala:410)
     at
org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.s
cala:37)
     at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
```

```
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.hasNext(SerDe
Util.scala:153)
	at scala.collection.Iterator.foreach(Iterator.scala:941)
	at scala.collection.Iterator.foreach$(Iterator.scala:941)
	at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.foreach(SerDe
Util.scala:148)
	at
org.apache.spark.api.python.PythonRDD$.writeIteratorToStream(PythonRDD
.scala:224)
	at org.apache.spark.api.python.PythonRunner$
$anon$2.writeIteratorToStream(PythonRunner.scala:561)
	at org.apache.spark.api.python.BasePythonRunner$WriterThread.
$anonfun$run$1(PythonRunner.scala:346)
	at
org.apache.spark.util.Utils$.logUncaughtExceptions(Utils.scala:1945)
	at
org.apache.spark.api.python.BasePythonRunner$WriterThread.run(PythonRu
nner.scala:196)

24/05/05 08:30:29 ERROR org.apache.spark.scheduler.TaskSetManager:
Task 0 in stage 0.0 failed 1 times; aborting job
Traceback (most recent call last):
  File "/tmp/79c44c8ef32a4c85a55c86356d090fbf/2Aspark.py", line 214,
in <module>
    parameter_dataset_throughput_array = df_results.rdd.map(lambda z:
((z['batch_size'], z['batch_number'], z['repetition']),
(z['datasetsize'], z['throughput']))).collect()
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/rdd.py", line
816, in collect
  File
"/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py",
line 1257, in __call__
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/utils.py",
line 63, in deco
  File
"/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/protocol.py", line
328, in get_return_value
py4j.protocol.Py4JJavaError: An error occurred while calling
z:org.apache.spark.api.python.PythonRDD.collectAndServe.
: org.apache.spark.SparkException: Job aborted due to stage failure:
Task 0 in stage 0.0 failed 1 times, most recent failure: Lost task 0.0
in stage 0.0 (TID 0, localhost, executor driver):
org.apache.spark.api.python.PythonException: Traceback (most recent
call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
```

```
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_shape_pb2.py", line 42, in <module>
```

```
    serialized_options=None, file=DESCRIPTOR),
  File
"/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descri
ptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of
date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible
workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will
use pure-Python parsing and will be much slower).

More information:
https://developers.google.com/protocol-buffers/docs/news/2022-05-
06#python-updates

      at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePyth
onException(PythonRunner.scala:457)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:592)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:575)
      at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(Py
thonRunner.scala:410)
      at
org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.s
cala:37)
      at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.hasNext(SerDe
Util.scala:153)
      at scala.collection.Iterator.foreach(Iterator.scala:941)
      at scala.collection.Iterator.foreach$(Iterator.scala:941)
      at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.foreach(SerDe
Util.scala:148)
      at
org.apache.spark.api.python.PythonRDD$.writeIteratorToStream(PythonRDD
.scala:224)
      at org.apache.spark.api.python.PythonRunner$
$anon$2.writeIteratorToStream(PythonRunner.scala:561)
```

```
	at org.apache.spark.api.python.BasePythonRunner$WriterThread.
$anonfun$run$1(PythonRunner.scala:346)
	at
org.apache.spark.util.Utils$.logUncaughtExceptions(Utils.scala:1945)
	at
org.apache.spark.api.python.BasePythonRunner$WriterThread.run(PythonRu
nner.scala:196)

Driver stacktrace:
	at
org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DA
GScheduler.scala:1926)
	at org.apache.spark.scheduler.DAGScheduler.
$anonfun$abortStage$2(DAGScheduler.scala:1914)
	at org.apache.spark.scheduler.DAGScheduler.
$anonfun$abortStage$2$adapted(DAGScheduler.scala:1913)
	at
scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:6
2)
	at scala.collection.mutable.ResizableArray.foreach$
(ResizableArray.scala:55)
	at
scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)
	at
org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:
1913)
	at org.apache.spark.scheduler.DAGScheduler.
$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:948)
	at org.apache.spark.scheduler.DAGScheduler.
$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:948)
	at scala.Option.foreach(Option.scala:407)
	at
org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGSchedul
er.scala:948)
	at
org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DA
GScheduler.scala:2147)
	at
org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGS
cheduler.scala:2096)
	at
org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGS
cheduler.scala:2085)
	at org.apache.spark.util.EventLoop$
$anon$1.run(EventLoop.scala:49)
	at
org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:759)
	at org.apache.spark.SparkContext.runJob(SparkContext.scala:2076)
	at org.apache.spark.SparkContext.runJob(SparkContext.scala:2097)
```

```
	at org.apache.spark.SparkContext.runJob(SparkContext.scala:2116)
	at org.apache.spark.SparkContext.runJob(SparkContext.scala:2141)
	at org.apache.spark.rdd.RDD.$anonfun$collect$1(RDD.scala:990)
	at
org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.sc
ala:151)
	at
org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.sc
ala:112)
	at org.apache.spark.rdd.RDD.withScope(RDD.scala:385)
	at org.apache.spark.rdd.RDD.collect(RDD.scala:989)
	at
org.apache.spark.api.python.PythonRDD$.collectAndServe(PythonRDD.scala
:166)
	at
org.apache.spark.api.python.PythonRDD.collectAndServe(PythonRDD.scala)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62)
	at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:498)
	at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
	at
py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
	at py4j.Gateway.invoke(Gateway.java:282)
	at
py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
	at py4j.commands.CallCommand.execute(CallCommand.java:79)
	at py4j.GatewayConnection.run(GatewayConnection.java:238)
	at java.lang.Thread.run(Thread.java:750)
Caused by: org.apache.spark.api.python.PythonException: Traceback
(most recent call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
```

```
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File
"/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descri
ptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of
date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible
workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will
```

```
use pure-Python parsing and will be much slower).

More information:
https://developers.google.com/protocol-buffers/docs/news/2022-05-
06#python-updates

      at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePyth
onException(PythonRunner.scala:457)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:592)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:575)
      at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(Py
thonRunner.scala:410)
      at
org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.s
cala:37)
      at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
      at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.hasNext(SerDe
Util.scala:153)
      at scala.collection.Iterator.foreach(Iterator.scala:941)
      at scala.collection.Iterator.foreach$(Iterator.scala:941)
      at
org.apache.spark.api.python.SerDeUtil$AutoBatchedPickler.foreach(SerDe
Util.scala:148)
      at
org.apache.spark.api.python.PythonRDD$.writeIteratorToStream(PythonRDD
.scala:224)
      at org.apache.spark.api.python.PythonRunner$
$anon$2.writeIteratorToStream(PythonRunner.scala:561)
      at org.apache.spark.api.python.BasePythonRunner$WriterThread.
$anonfun$run$1(PythonRunner.scala:346)
      at
org.apache.spark.util.Utils$.logUncaughtExceptions(Utils.scala:1945)
      at
org.apache.spark.api.python.BasePythonRunner$WriterThread.run(PythonRu
nner.scala:196)

24/05/05 08:30:29 INFO
org.spark_project.jetty.server.AbstractConnector: Stopped
Spark@3e6b8642{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
ERROR: (gcloud.dataproc.jobs.submit.pyspark) Job
[79c44c8ef32a4c85a55c86356d090fbf] failed with error:
```

```
Google Cloud Dataproc Agent reports job failure. If logs are
available, they can be found at:
https://console.cloud.google.com/dataproc/jobs/79c44c8ef32a4c85a55c863
56d090fbf?project=bigdata-421920&region=us-central1
gcloud dataproc jobs wait '79c44c8ef32a4c85a55c86356d090fbf' --region
'us-central1' --project 'bigdata-421920'
https://console.cloud.google.com/storage/browser/dataproc-staging-us-
central1-504478955913-tcjfqtey/google-cloud-dataproc-metainfo/
19b96cb7-e604-4336-aa84-02d92767b865/jobs/
79c44c8ef32a4c85a55c86356d090fbf/
gs://dataproc-staging-us-central1-504478955913-tcjfqtey/google-cloud-
dataproc-metainfo/19b96cb7-e604-4336-aa84-02d92767b865/jobs/
79c44c8ef32a4c85a55c86356d090fbf/driveroutput
```

2c) Improve efficiency (6%)

If you implemented a straightfoward version of 2a), you will **probably have an inefficiency** in your code.

Because we are reading multiple times from an RDD to read the values for the different parameters and their averages, caching existing results is important. Explain **where in the process caching can help**, and **add a call to `RDD.cache()`** to your code, if you haven't yet. Measure the the effect of using caching or not using it.

Make the **suitable change** in the code you have written above and mark them up in comments as `### TASK 2c ###`.

Explain in your report what the **reasons for this change** are and **demonstrate and interpret its effect**

```
###Coding Task###



#Script with cache

%%writefile 2Cspark.py


#Required libraries
import os, sys, math
os.environ['PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION'] = 'python'
import numpy as np
import scipy as sp
import scipy.stats
import time
import datetime
import string
import random
from matplotlib import pyplot as plt
```

```
import tensorflow as tf
print("Tensorflow version " + tf.__version__)
import pickle
import pyspark
from pyspark.sql import SQLContext
from pyspark.sql import Row
from pyspark.sql import SparkSession
import pandas as pd

from pyspark.sql.types import StructType, StructField, IntegerType,
DoubleType, StringType



#Required variables
PROJECT = 'bigdata-421920'
BUCKET = 'gs://{}-storage'.format(PROJECT)
REGION = 'us-central1'
CLUSTER = '{}-cluster'.format(PROJECT)
GCS_PATTERN = 'gs://flowers-public/*/*.jpg'
PARTITIONS = 16 # no of partitions we will use later
GCS_OUTPUT = BUCKET + '/tfrecords-jpeg-192x192-2/flowers'
CLASSES = [b'daisy', b'dandelion', b'roses', b'sunflowers', b'tulips']
TARGET_SIZE = [192, 192] # target resolution for the images


#
#i) combine the previous cells to have the code to create a dataset
and create a list of parameter combinations in an RDD (2%)


#Function to parse TFRecord examples
def read_tfrecord(example):
    features = {
        "image": tf.io.FixedLenFeature([], tf.string),  # tf.string =
bytestring (not text string)
        "class": tf.io.FixedLenFeature([], tf.int64) #,   # shape []
means scalar
    }
    # decode the TFRecord
    # Parse a single TFRecord example using the specified features
    example = tf.io.parse_single_example(example, features)
    #Decoing image from TFRrecord example
    image = tf.image.decode_jpeg(example['image'], channels=3)
    #reshaping image to target size
    image = tf.reshape(image, [*TARGET_SIZE, 3])
    #Extracting class label
    class_num = example['class']
    #output will be parsed image and class label
```

```
    return image, class_num

#Function to load dataset from TFRecord files
def load_dataset(filenames):
    # read from TFRecords. For optimal performance, read from multiple
    # TFRecord files at once and set the option
experimental_deterministic = False
    # to allow order-altering optimizations.

    option_no_order = tf.data.Options()
    option_no_order.experimental_deterministic = False
    #dataset
    dataset = tf.data.TFRecordDataset(filenames)
    dataset = dataset.with_options(option_no_order)
    dataset = dataset.map(read_tfrecord)
    return dataset

def decode_jpeg_and_label(filepath):
    # extracts the image data and creates a class label, based on the
filepath
    bits = tf.io.read_file(filepath)
    image = tf.image.decode_jpeg(bits)
    # parse flower name from containing directory
    label = tf.strings.split(tf.expand_dims(filepath, axis=-1),
sep='/')
    label2 = label.values[-2]
    return image, label2

#Function to resize and crop the image
def resize_and_crop_image(image, label):
    # Resizes and cropd using "fill" algorithm:
    # always make sure the resulting image is cut out from the source
image
    # so that it fills the TARGET_SIZE entirely with no black bars
    # and a preserved aspect ratio.
    #Dimensions of the image
    w = tf.shape(image)[0]
    h = tf.shape(image)[1]
    tw = TARGET_SIZE[1]
    th = TARGET_SIZE[0]
    #Resize image
    resize_crit = (w * th) / (h * tw)
    image = tf.cond(resize_crit < 1,
                    lambda: tf.image.resize(image, [w*tw/w, h*tw/w]),
# if true
                    lambda: tf.image.resize(image, [w*th/h, h*th/h])
# if false
                    )
    nw = tf.shape(image)[0]
    nh = tf.shape(image)[1]
```

```
    #Cropped image is resized to target size
    image = tf.image.crop_to_bounding_box(image, (nw - tw) // 2, (nh -
th) // 2, tw, th)
    #Return the new image (resized and cropped) and label
    return image, label


# Function to load dataset from TFRecord files and decode
def load_dataset_decoded():
    #Obtaining list of TFRecord files
    dataset_filename = tf.data.Dataset.list_files(GCS_PATTERN)
    datasetDecoded = dataset_filename.map(decode_jpeg_and_label)
    datasetfn = datasetDecoded.map(resize_and_crop_image)
    return datasetfn


# Define the new time_configs function
def time_configs_rdd(parameters_rdd):
    #Timer
    start = time.time()
    #Extracting parameters from RDD
    batch_size = parameters_rdd[0]
    batch_number = parameters_rdd[1]
    repetition = parameters_rdd[2]
    dataset_type = parameters_rdd[3]

    #Loading dataset based on type
    if dataset_type == 'datasetDecoded':
        filenames = tf.io.gfile.glob(GCS_OUTPUT + "*.tfrec")
        dataset = load_dataset(filenames)
    else:
        filenames_fn = tf.data.Dataset.list_files(GCS_PATTERN)
        dataset_fn = filenames_fn.map(decode_jpeg_and_label)
        dataset = dataset_fn.map(resize_and_crop_image)

    #Batch of dataset
    dataset1 = dataset.batch(batch_size)
    test_set = dataset1.take(batch_number)
    time_list = []

    #Time test
    for _ in range(repetition):
        s_time = time.time()
        for _ in test_set:
            print('string', file=open("/dev/null", mode='w'))
        e_time = time.time()
        #Reading speed is calculated
        reading_speed = e_time - s_time
        #Throughput is calculated
        throughput = float((batch_size * batch_number) / (e_time -
```

```
s_time))
        datasetsize = batch_size * batch_number
        #Times values are appended to the list
        time_list.append([batch_size, batch_number, repetition,
datasetsize, reading_speed, throughput])

    end = time.time()
    #calculating total time and toal images
    total_images = batch_size * batch_number * repetition
    total_time = total_images / (end - start)
    return total_time, time_list


# Define the parameter combinations, for which will be used
batch_sizes = [2, 4, 6, 8]
batch_numbers = [6, 9, 12, 15]
repetitions = [1, 2, 3]
dataset_types = ['datasetDecoded', 'otherDataset']

#list to store parameter combinations
parameter_list = []
for batch_size in batch_sizes:
    for batch_number in batch_numbers:
        for repetition in repetitions:
          for dataset_type in dataset_types:
              parameter_list.append([batch_size, batch_number,
repetition, dataset_type])


# Define the columns for DataFrame
columns = ["batch_size", "batch_number", "repetition", "dataset_type",
"datasetsize", "reading_speed", "throughput"]

# Create Spark session
spark =
SparkSession.builder.master("local").appName("DataProcessing").getOrCr
eate()

# Create RDD for parameter combinations
### TASK 2c ###
rdd_parameters =
spark.sparkContext.parallelize(parameter_list).cache()




#
#ii) get a Spark context and create the dataset and run timing test
for each combination in parallel (2%)
#Reference: https://pypi.org/project/schema/
```

```
# Define the schema for time_list
schema = StructType([
    StructField("batch_size", IntegerType(), True),
    StructField("batch_number", IntegerType(), True),
    StructField("repetition", IntegerType(), True),
    StructField("datasetsize", IntegerType(), True),
    StructField("reading_speed", DoubleType(), True),
    StructField("throughput", DoubleType(), True),
])

# Convert RDD to DataFrame using the specified schema
df_results = spark.createDataFrame(rdd_parameters.flatMap(lambda x:
time_configs_rdd(x)[1]), schema=schema)


#
# iii) Transform the resulting RDD to the structure
(parameter_combination, images_per_second) and save these values in an
array

# Transform the resulting DataFrame to include dataset size along with
throughput
# Update the lambda function to correctly access DataFrame columns and
handle data types
parameter_dataset_throughput_array = df_results.rdd.map(lambda z:
((z['batch_size'], z['batch_number'], z['repetition']),
(z['datasetsize'], z['throughput']))).collect()


#
#### Coding Task###
#Reference:https://github.com/vighnesh32/Big-Data-Project/blob/main/
project.ipynb
#iv) create an RDD with all results for each parameter as
(parameter_value,images_per_second) and collect the result for each
parameter (2%)


#Extracting batch size and throughput
rdd_tfrecord_batch_sizes_speed = df_results.rdd.map(lambda z:
(int(z['batch_size']), float(z['throughput'])))
# Collect batch size results for TFRecord dataset
tfrecord_batch_sizes_speed = rdd_tfrecord_batch_sizes_speed.collect()

#Extracting  batch numbers and throughput
rdd_tfrecord_batch_nums_speed = df_results.rdd.map(lambda z:
(int(z['batch_number']), float(z['throughput'])))
tfrecord_batch_nums_speed = rdd_tfrecord_batch_nums_speed.collect()
```

```
# Extracting repetitions and throughput
rdd_tfrecord_repetitions_speed = df_results.rdd.map(lambda z:
(int(z['repetition']), float(z['throughput'])))
tfrecord_repetitions_speed = rdd_tfrecord_repetitions_speed.collect()

# Extracting  dataset size and throughput
rdd_tfrecord_datasetsize_speed = df_results.rdd.map(lambda z:
(int(z['datasetsize']), float(z['throughput'])))
tfrecord_datasetsize_speed = rdd_tfrecord_datasetsize_speed.collect()

#
###Coding Task###
#Reference:
https://github.com/vighnesh32/Big-Data-Project/blob/main/project.ipynb
#v) create an RDD with the average reading speeds for each parameter
value and collect the results.
#Keep associativity in mind when implementing the average. (3%)


#Average speed for batch size
rdd_tfrecord_batch_sizes_avg_speed = rdd_tfrecord_batch_sizes_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_batch_sizes_avg_speed =
rdd_tfrecord_batch_sizes_avg_speed.collect()

#Average speed For batch numbers
rdd_tfrecord_batch_nums_avg_speed = rdd_tfrecord_batch_nums_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_batch_nums_avg_speed =
rdd_tfrecord_batch_nums_avg_speed.collect()

#Average speed For repetitions
rdd_tfrecord_repetitions_avg_speed = rdd_tfrecord_repetitions_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_repetitions_avg_speed =
rdd_tfrecord_repetitions_avg_speed.collect()

#Average speed for dataset sizes
rdd_tfrecord_datasetsize_avg_speed = rdd_tfrecord_datasetsize_speed \
    .mapValues(lambda z: (z, 1)) \
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) \
    .mapValues(lambda z: z[0] / z[1])
tfrecord_datasetsize_avg_speed =
rdd_tfrecord_datasetsize_avg_speed.collect()
```

```
#
# vi) write the results to a pickle file in your bucket (1%)

#Function to save object to file and upload it to filename which
contains directory to cloud storage bucket
def save(object,bucket,filename):
    with open(filename, mode='wb') as f:
      pickle.dump(object,f)
    print("Saving{} to {}".format(filename,bucket))
    import subprocess

proc=subprocess.run(["gsutil","cp",filename,bucket],stderr=subprocess.
PIPE)
    print("gstuil returned: " + str(proc.returncode))
    print(str(proc.stderr))

#filename
filename="2cPickle.pkl"

#dumping objects with pickle
with open(filename,mode='wb') as f:
      pickle.dump(tfrecord_batch_sizes_speed,f)
      pickle.dump(tfrecord_batch_nums_speed,f)
      pickle.dump(tfrecord_repetitions_speed,f)
      pickle.dump(tfrecord_datasetsize_speed,f)
      pickle.dump(tfrecord_batch_sizes_avg_speed,f)
      pickle.dump(tfrecord_batch_nums_avg_speed,f)
      pickle.dump(tfrecord_repetitions_avg_speed,f)
      pickle.dump(tfrecord_datasetsize_avg_speed,f)

print("Saving {} to {}".format(filename,BUCKET))
import subprocess
proc = subprocess.run(["gsutil", "cp",filename, BUCKET],
stderr=subprocess.PIPE)
print("gstuil returned: " +str(proc.returncode))
print(str(proc.stderr))
```

```
Overwriting 2Cspark.py
```

```
###CODING TASK###
#Running the script with cache on a cluster
!gcloud dataproc jobs submit pyspark --cluster eightfold-cluster \
2Cspark.py
```

```
Job [cbe1921790484d57a153fc141972df31] submitted.
Waiting for job output...
2024-05-05 08:30:53.650635: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
not load dynamic library 'libcudart.so.11.0'; dlerror:
```

```
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 08:30:53.650679: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
Tensorflow version 2.4.0
24/05/05 08:30:56 INFO org.apache.spark.SparkEnv: Registering
MapOutputTracker
24/05/05 08:30:56 INFO org.apache.spark.SparkEnv: Registering
BlockManagerMaster
24/05/05 08:30:56 INFO org.apache.spark.SparkEnv: Registering
OutputCommitCoordinator
24/05/05 08:30:56 INFO org.spark_project.jetty.util.log: Logging
initialized @5995ms to org.spark_project.jetty.util.log.Slf4jLog
24/05/05 08:30:56 INFO org.spark_project.jetty.server.Server: jetty-
9.4.z-SNAPSHOT; built: unknown; git: unknown; jvm 1.8.0_382-b05
24/05/05 08:30:56 INFO org.spark_project.jetty.server.Server: Started
@6102ms
24/05/05 08:30:56 INFO
org.spark_project.jetty.server.AbstractConnector: Started
ServerConnector@706fad44{HTTP/1.1, (http/1.1)}{0.0.0.0:44317}
2024-05-05 08:31:02.878202: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could
not load dynamic library 'libcudart.so.11.0'; dlerror:
libcudart.so.11.0: cannot open shared object file: No such file or
directory; LD_LIBRARY_PATH: :/usr/lib/hadoop/lib/native
2024-05-05 08:31:02.878273: I
tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart
dlerror if you do not have a GPU set up on your machine.
24/05/05 08:31:03 WARN org.apache.spark.storage.BlockManager: Putting
block rdd_6_0 failed due to exception
org.apache.spark.api.python.PythonException: Traceback (most recent
call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
```

```
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File
"/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descri
ptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of
date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible
workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will
use pure-Python parsing and will be much slower).
```

```
More information:
https://developers.google.com/protocol-buffers/docs/news/2022-05-
06#python-updates
.
24/05/05 08:31:03 WARN org.apache.spark.storage.BlockManager: Block
rdd_6_0 could not be removed as it was not found on disk or in memory
24/05/05 08:31:03 ERROR org.apache.spark.executor.Executor: Exception
in task 0.0 in stage 0.0 (TID 0)
org.apache.spark.api.python.PythonException: Traceback (most recent
call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
```

```
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framework/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File "/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framework/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File "/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framework/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File "/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descriptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will use pure-Python parsing and will be much slower).

More information: https://developers.google.com/protocol-buffers/docs/news/2022-05-06#python-updates

     at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:457)
     at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:592)
     at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:575)
     at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:410)
     at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
     at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
     at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
     at org.apache.spark.sql.execution.columnar.CachedRDDBuilder$$anon$1.hasNext(InMemoryRelation.scala:125)
```

```
	at
org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.sc
ala:221)
	at
org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(Memory
Store.scala:299)
	at org.apache.spark.storage.BlockManager.
$anonfun$doPutIterator$1(BlockManager.scala:1165)
	at
org.apache.spark.storage.BlockManager.doPut(BlockManager.scala:1091)
	at
org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala
:1156)
	at
org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.sca
la:882)
	at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:357)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:308)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at org.apache.spark.sql.execution.SQLExecutionRDD.
$anonfun$compute$1(SQLExecutionRDD.scala:52)
	at
org.apache.spark.sql.internal.SQLConf$.withExistingConf(SQLConf.scala:
92)
	at
org.apache.spark.sql.execution.SQLExecutionRDD.compute(SQLExecutionRDD
.scala:52)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
```

```
2)
      at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
      at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
      at
org.apache.spark.api.python.PythonRDD.compute(PythonRDD.scala:65)
      at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
      at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
      at
org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
      at org.apache.spark.scheduler.Task.run(Task.scala:123)
      at org.apache.spark.executor.Executor$TaskRunner.
$anonfun$run$3(Executor.scala:414)
      at
org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360)
      at
org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:417)
      at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.j
ava:1149)
      at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.
java:624)
      at java.lang.Thread.run(Thread.java:750)
24/05/05 08:31:03 WARN org.apache.spark.scheduler.TaskSetManager: Lost
task 0.0 in stage 0.0 (TID 0, localhost, executor driver):
org.apache.spark.api.python.PythonException: Traceback (most recent
call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
```

```
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File
"/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descri
ptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of
date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible
workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will
use pure-Python parsing and will be much slower).

More information:
https://developers.google.com/protocol-buffers/docs/news/2022-05-
06#python-updates
```

```
	at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePyth
onException(PythonRunner.scala:457)
	at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:592)
	at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:575)
	at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(Py
thonRunner.scala:410)
	at
org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.s
cala:37)
	at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at org.apache.spark.sql.execution.columnar.CachedRDDBuilder$
$anon$1.hasNext(InMemoryRelation.scala:125)
	at
org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.sc
ala:221)
	at
org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(Memory
Store.scala:299)
	at org.apache.spark.storage.BlockManager.
$anonfun$doPutIterator$1(BlockManager.scala:1165)
	at
org.apache.spark.storage.BlockManager.doPut(BlockManager.scala:1091)
	at
org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala
:1156)
	at
org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.sca
la:882)
	at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:357)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:308)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
```

```
	at org.apache.spark.sql.execution.SQLExecutionRDD.
$anonfun$compute$1(SQLExecutionRDD.scala:52)
	at
org.apache.spark.sql.internal.SQLConf$.withExistingConf(SQLConf.scala:
92)
	at
org.apache.spark.sql.execution.SQLExecutionRDD.compute(SQLExecutionRDD
.scala:52)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.api.python.PythonRDD.compute(PythonRDD.scala:65)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
	at org.apache.spark.scheduler.Task.run(Task.scala:123)
	at org.apache.spark.executor.Executor$TaskRunner.
$anonfun$run$3(Executor.scala:414)
	at
org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360)
	at
org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:417)
	at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.j
ava:1149)
	at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.
java:624)
	at java.lang.Thread.run(Thread.java:750)

24/05/05 08:31:03 ERROR org.apache.spark.scheduler.TaskSetManager:
Task 0 in stage 0.0 failed 1 times; aborting job
Traceback (most recent call last):
  File "/tmp/cbe1921790484d57a153fc141972df31/2Cspark.py", line 215,
```

```
in <module>
    parameter_dataset_throughput_array = df_results.rdd.map(lambda z:
((z['batch_size'], z['batch_number'], z['repetition'],
z['dataset_type']), (z['datasetsize'], z['reading_speed'],
z['throughput']))).collect()
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/rdd.py", line
816, in collect
  File
"/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py",
line 1257, in __call__
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/utils.py",
line 63, in deco
  File
"/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/protocol.py", line
328, in get_return_value
py4j.protocol.Py4JJavaError: An error occurred while calling
z:org.apache.spark.api.python.PythonRDD.collectAndServe.
: org.apache.spark.SparkException: Job aborted due to stage failure:
Task 0 in stage 0.0 failed 1 times, most recent failure: Lost task 0.0
in stage 0.0 (TID 0, localhost, executor driver):
org.apache.spark.api.python.PythonException: Traceback (most recent
call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
```

```
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as
tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File
"/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descri
ptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of
date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible
workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will
use pure-Python parsing and will be much slower).

More information:
https://developers.google.com/protocol-buffers/docs/news/2022-05-
06#python-updates

      at
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePyth
onException(PythonRunner.scala:457)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:592)
      at org.apache.spark.api.python.PythonRunner$
$anon$3.read(PythonRunner.scala:575)
      at
```

```
org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:410)
	at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
	at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at org.apache.spark.sql.execution.columnar.CachedRDDBuilder$$anon$1.hasNext(InMemoryRelation.scala:125)
	at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:221)
	at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:299)
	at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1165)
	at org.apache.spark.storage.BlockManager.doPut(BlockManager.scala:1091)
	at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1156)
	at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:882)
	at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:357)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:308)
	at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
	at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
	at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at org.apache.spark.sql.execution.SQLExecutionRDD.$anonfun$compute$1(SQLExecutionRDD.scala:52)
	at org.apache.spark.sql.internal.SQLConf$.withExistingConf(SQLConf.scala:92)
	at org.apache.spark.sql.execution.SQLExecutionRDD.compute(SQLExecutionRDD.scala:52)
```

```
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.api.python.PythonRDD.compute(PythonRDD.scala:65)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
	at org.apache.spark.scheduler.Task.run(Task.scala:123)
	at org.apache.spark.executor.Executor$TaskRunner.
$anonfun$run$3(Executor.scala:414)
	at
org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360)
	at
org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:417)
	at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.j
ava:1149)
	at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.
java:624)
	at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:
	at
org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DA
GScheduler.scala:1926)
	at org.apache.spark.scheduler.DAGScheduler.
$anonfun$abortStage$2(DAGScheduler.scala:1914)
	at org.apache.spark.scheduler.DAGScheduler.
$anonfun$abortStage$2$adapted(DAGScheduler.scala:1913)
	at
scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:6
2)
	at scala.collection.mutable.ResizableArray.foreach$
```

```
(ResizableArray.scala:55)
      at
scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)
      at
org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:
1913)
      at org.apache.spark.scheduler.DAGScheduler.
$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:948)
      at org.apache.spark.scheduler.DAGScheduler.
$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:948)
      at scala.Option.foreach(Option.scala:407)
      at
org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGSchedul
er.scala:948)
      at
org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DA
GScheduler.scala:2147)
      at
org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGS
cheduler.scala:2096)
      at
org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGS
cheduler.scala:2085)
      at org.apache.spark.util.EventLoop$
$anon$1.run(EventLoop.scala:49)
      at
org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:759)
      at org.apache.spark.SparkContext.runJob(SparkContext.scala:2076)
      at org.apache.spark.SparkContext.runJob(SparkContext.scala:2097)
      at org.apache.spark.SparkContext.runJob(SparkContext.scala:2116)
      at org.apache.spark.SparkContext.runJob(SparkContext.scala:2141)
      at org.apache.spark.rdd.RDD.$anonfun$collect$1(RDD.scala:990)
      at
org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.sc
ala:151)
      at
org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.sc
ala:112)
      at org.apache.spark.rdd.RDD.withScope(RDD.scala:385)
      at org.apache.spark.rdd.RDD.collect(RDD.scala:989)
      at
org.apache.spark.api.python.PythonRDD$.collectAndServe(PythonRDD.scala
:166)
      at
org.apache.spark.api.python.PythonRDD.collectAndServe(PythonRDD.scala)
      at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
      at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.j
ava:62)
```

```
	at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccess
orImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:498)
	at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
	at
py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
	at py4j.Gateway.invoke(Gateway.java:282)
	at
py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
	at py4j.commands.CallCommand.execute(CallCommand.java:79)
	at py4j.GatewayConnection.run(GatewayConnection.java:238)
	at java.lang.Thread.run(Thread.java:750)
Caused by: org.apache.spark.api.python.PythonException: Traceback
(most recent call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
364, in main
    func, profiler, deserializer, serializer = read_command(pickleSer,
infile)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line
69, in read_command
    command = serializer._read_with_length(file)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 173, in _read_with_length
    return self.loads(obj)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py",
line 587, in loads
    return pickle.loads(obj, encoding=encoding)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/cloudpickle.py",
line 875, in subimport
    __import__(name)
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/__init__.py
", line 41, in <module>
    from tensorflow.python.tools import module_util as _module_util
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
__init__.py", line 41, in <module>
    from tensorflow.python.eager import context
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/python/
eager/context.py", line 32, in <module>
    from tensorflow.core.framework import function_pb2
  File
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framew
ork/function_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File
```

```
"/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framework/attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as tensorflow_dot_core_dot_framework_dot_tensor__pb2
  File "/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framework/tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File "/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framework/resource_handle_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_shape_pb2 as tensorflow_dot_core_dot_framework_dot_tensor__shape__pb2
  File "/opt/conda/default/lib/python3.7/site-packages/tensorflow/core/framework/tensor_shape_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File "/opt/conda/default/lib/python3.7/site-packages/google/protobuf/descriptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of date and must be regenerated with protoc >= 3.19.0.
If you cannot immediately regenerate your protos, some other possible workarounds are:
 1. Downgrade the protobuf package to 3.20.x or lower.
 2. Set PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python (but this will use pure-Python parsing and will be much slower).

More information: https://developers.google.com/protocol-buffers/docs/news/2022-05-06#python-updates

     at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:457)
     at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:592)
     at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:575)
     at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:410)
     at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
     at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:489)
```

```
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:458)
	at org.apache.spark.sql.execution.columnar.CachedRDDBuilder$
$anon$1.hasNext(InMemoryRelation.scala:125)
	at
org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.sc
ala:221)
	at
org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(Memory
Store.scala:299)
	at org.apache.spark.storage.BlockManager.
$anonfun$doPutIterator$1(BlockManager.scala:1165)
	at
org.apache.spark.storage.BlockManager.doPut(BlockManager.scala:1091)
	at
org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala
:1156)
	at
org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.sca
la:882)
	at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:357)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:308)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at org.apache.spark.sql.execution.SQLExecutionRDD.
$anonfun$compute$1(SQLExecutionRDD.scala:52)
	at
org.apache.spark.sql.internal.SQLConf$.withExistingConf(SQLConf.scala:
92)
	at
org.apache.spark.sql.execution.SQLExecutionRDD.compute(SQLExecutionRDD
.scala:52)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
```

```
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:5
2)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.api.python.PythonRDD.compute(PythonRDD.scala:65)
	at
org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:346)
	at org.apache.spark.rdd.RDD.iterator(RDD.scala:310)
	at
org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
	at org.apache.spark.scheduler.Task.run(Task.scala:123)
	at org.apache.spark.executor.Executor$TaskRunner.
$anonfun$run$3(Executor.scala:414)
	at
org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360)
	at
org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:417)
	at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.j
ava:1149)
	at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.
java:624)
	... 1 more

24/05/05 08:31:03 INFO
org.spark_project.jetty.server.AbstractConnector: Stopped
Spark@706fad44{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
ERROR: (gcloud.dataproc.jobs.submit.pyspark) Job
[cbe1921790484d57a153fc141972df31] failed with error:
Google Cloud Dataproc Agent reports job failure. If logs are
available, they can be found at:
https://console.cloud.google.com/dataproc/jobs/cbe1921790484d57a153fc1
41972df31?project=bigdata-421920&region=us-central1
gcloud dataproc jobs wait 'cbe1921790484d57a153fc141972df31' --region
'us-central1' --project 'bigdata-421920'
https://console.cloud.google.com/storage/browser/dataproc-staging-us-
central1-504478955913-tcjfqtey/google-cloud-dataproc-metainfo/
19b96cb7-e604-4336-aa84-02d92767b865/jobs/
cbe1921790484d57a153fc141972df31/
gs://dataproc-staging-us-central1-504478955913-tcjfqtey/google-cloud-
dataproc-metainfo/19b96cb7-e604-4336-aa84-02d92767b865/jobs/
cbe1921790484d57a153fc141972df31/driveroutput
```

2d) Retrieve, analyse and discuss the output (12%)

Run the tests over a wide range of different paramters and list the results in a table.

Perform a **linear regression** (e.g. using scikit-learn) over **the values for each parameter** and for the **two cases** (reading from image files/reading TFRecord files). List a **table** with the output and interpret the results in terms of the effects of overall.
Also, **plot** the output values, the averages per parameter value and the regression lines for each parameter and for the product of batch_size and batch_number

Discuss the **implications** of this result for **applications** like large-scale machine learning. Keep in mind that cloud data may be stored in distant physical locations. Use the numbers provided in the PDF latency-numbers document available on Moodle or here for your arguments.

How is the **observed** behaviour **similar or different** from what you'd expect from a **single machine**? Why would cloud providers tie throughput to capacity of disk resources?

By **parallelising** the speed test we are making **assumptions** about the limits of the bucket reading speeds. See here for more information. Discuss, **what we need to consider** in **speed tests** in parallel on the cloud, which bottlenecks we might be identifying, and how this relates to your results.

Discuss to what extent **linear modelling** reflects the **effects** we are observing. Discuss what could be expected from a theoretical perspective and what can be useful in practice.

Write your **code below** and **include the output** in your submitted `ipynb` file. Provide the answer **text in your report**.

```
### CODING TASK ###
###Loading pickle files
!gsutil cp $BUCKET/2cPickle.pkl .
with open("2cPickle.pkl",mode = 'rb') as f:
          tfrecord_batch_sizes_speed = pickle.load(f)
          tfrecord_batch_nums_speed = pickle.load(f)
          tfrecord_repetitions_speed = pickle.load(f)
          tfrecord_datasetsize_speed = pickle.load(f)
          tfrecord_batch_sizes_avg_speed = pickle.load(f)
          tfrecord_batch_nums_avg_speed = pickle.load(f)
          tfrecord_repetitions_avg_speed = pickle.load(f)
          tfrecord_datasetsize_avg_speed = pickle.load(f)
```

```
CommandException: No URLs matched:
gs://bigdata-421920-storage/2cPickle.pkl
```

```
---------------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
<ipython-input-176-56adcf57f59f> in <cell line: 4>()
      2 ### CODING TASK ###
      3 get_ipython().system('gsutil cp $BUCKET/2cPickle.pkl .')
```

```
----> 4 with open("2cPickle.pkl",mode = 'rb') as f:
      5           tfrecord_batch_sizes_speed = pickle.load(f)
      6           tfrecord_batch_nums_speed = pickle.load(f)

FileNotFoundError: [Errno 2] No such file or directory: '2cPickle.pkl'
```

Section 3. Theoretical discussion

Task 3: Discussion in context. (24%)

In this task we refer an idea that is introduced in this paper:

- Alipourfard, O., Liu, H. H., Chen, J., Venkataraman, S., Yu, M., & Zhang, M. (2017). Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics.. In USENIX NSDI 17 (pp. 469-482).

Alipourfard et al (2017) introduce the prediction an optimal or near-optimal cloud configuration for a given compute task.

3a) Contextualise

Relate the previous tasks and the results to this concept. (It is not necessary to work through the full details of the paper, focus just on the main ideas). To what extent and under what conditions do the concepts and techniques in the paper apply to the task in this coursework? (12%)

3b) Strategise

Define - as far as possible - concrete strategies for different application scenarios (batch, stream) and discuss the general relationship with the concepts above. (12%)

Provide the answers to these questions in your report.

Final cleanup

Once you have finshed the work, you can delete the buckets, to stop incurring cost that depletes your credit.

```
!gsutil -m rm -r $BUCKET/* # Empty your bucket
!gsutil rb $BUCKET # delete the bucket
```

```
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers0.tfrec#17
14888780129877...
Removing gs://bigdata-421920-storage/2aPickle.pkl#1714897069091998...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers00-
230.tfrec#1714888638348878...
```

```
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers01-
230.tfrec#1714888649651751...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers02-
230.tfrec#1714888657748792...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers03-
230.tfrec#1714888665939480...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers04-
230.tfrec#1714888673304767...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers05-
230.tfrec#1714888682419825...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers06-
230.tfrec#1714888691404620...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers07-
230.tfrec#1714888699967782...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers08-
230.tfrec#1714888709112834...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers09-
230.tfrec#1714888716249723...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers1.tfrec#17
14888779931363...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers10-
230.tfrec#1714888724300987...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers11-
230.tfrec#1714888732385930...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers12-
230.tfrec#1714888739427999...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers13-
230.tfrec#1714888748340587...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers14-
230.tfrec#1714888756556931...
Removing
gs://bigdata-421920-storage/tfrecords-jpeg-192x192-2/flowers15-
220.tfrec#1714888763266666...
/ [19/19 objects] 100% Done
```

```
Operation completed over 19 objects.
Removing gs://bigdata-421920-storage/...
```