Course: INFO7250 Big Data Systems Engineering
Name: Pramod Nagare
NUID: 001858910
Email: nagare.p@husky.ney.edu
Project: Streaming data pipeline for real-time analytics and Sentiment Analysis
GitHub: https://github.com/pramodnagare/INFO7250BigDataSystems

Highlights:

- GCP Streaming and Batch Processing data analytical pipeline
- Bloom filter
- Hadoop cluster setup
- Google data explorer interactive dashboard
- Optimization (Combiner, distributed caches, cluster, compact data type)
- Multithread/multiprocessor scripts for data preparation
- Map-Reduce in Python and Java

Dataset:

- https://registry.opendata.aws/amazon-reviews/
- https://github.com/awslabs/open-data-registry/blob/master/datasets/amazon-reviews.yaml
- https://s3.amazonaws.com/amazon-reviews-pds/readme.html
- 34+ GB Dataset .gzip format
- 130+ millions amazon customer reviews

Task 1: Data download and extraction

Implementation:

1. Downloading AWS Customer review data from source (AWS S3 bucket)
    - Data Size: 130+ millions of rows, ~80GB of data over 51 files
    - More details can be found on index.txt file
    - Developed parallel processing python script
2. Extracting downloaded .gzip file into tsv
    - Developed parallel processing python script

Screenshots:

Downloading dataset script: File is available **here**

In [1]:
```
#This jupyter notebook is for downloading data parallelly from s3 bucket to local file system:
```

In [2]:
```
import boto3
import botocore
import time
import os
import pandas as pd
from multiprocessing.pool import ThreadPool
```

In [3]:
```
data_link = "amazon-reviews-pds"
output_dir = "../Data2/"

s3 = boto3.client('s3')
response = s3.list_objects_v2(Bucket=data_link)
```

In [12]:
```
def downloadFile(fileItem):
    head, file = os.path.split(fileItem)
    filename = output_dir + file
    s = time.time()
    s3.download_file(Bucket=data_link, Key=fileItem, Filename=filename)
    e = time.time()
    print("Downloaded file {} in {} secs!".format(fileItem, e-s))
    return None
```

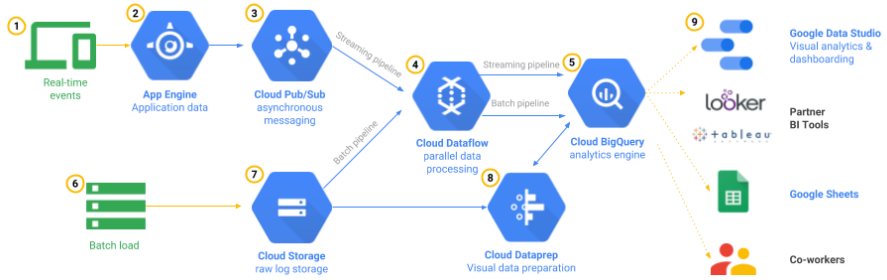
```
In [5]: data = []
        for fileItem in response['Contents']:
            if str(fileItem['Key']).startswith('tsv'):
                head, file = os.path.split(fileItem['Key'])
                if file != "":
                    data.append(fileItem)

In [6]: df = pd.DataFrame(data)
        df = df.sort_values('Size').reset_index()

In [47]: batches = {}
         batch_num = 0
         data = []
         size_sum = 0
         for i in range(df.shape[0]):
             size_sum += int(df['Size'][i])
             data.append(df['Key'][i])
             if (size_sum/1024/1024 > 200) or (len(data) > 8):
                 if len(data) > 1:
                     data.pop()
                     batches[batch_num] = data
                     size_sum = int(df['Size'][i])
                     data = [df['Key'][i]]
                 else:
                     batches[batch_num] = data
                     size_sum = 0
                     data = []
                 batch_num += 1
         batches[batch_num] = data

In [13]: start = time.time()
         for batch in batches.values():
             results = ThreadPool(len(batch)).imap_unordered(downloadFile, batch)
             for result in results:
                 while result != None:
                     continue
         end = time.time()
         print("Total Time for Downloading files is {} secs!".format(end-start))
```

Extracting dataset script: File is available **here**

```
In [1]: import concurrent.futures
        import os
        import glob
        import time
        import shutil
        import gzip

In [4]: input_folder = r"H:/INFO7250_BigData/Project/Data/"
        output_folder = r"../Main_Data/"
        file_type = ".tsv.gz"

        NUM_WORKERS = int(os.environ['NUMBER_OF_PROCESSORS'])

In [5]: files = glob.glob(input_folder+"*.tsv.gz")
        print("Total Number of file to extract: {}".format(len(files)))

        Total Number of file to extract: 51

In [6]: def extractFile(filename):
            path, file = os.path.split(filename)
            file = file.replace(".gz", "")
            with gzip.open(filename) as file_input:
                with open(output_folder+file, 'wb') as file_output:
                    shutil.copyfileobj(file_input, file_output)
            return None

In [7]: start = time.time()

        with concurrent.futures.ThreadPoolExecutor(max_workers=NUM_WORKERS) as executor:
            futures = {executor.submit(extractFile, file) for file in files}
            concurrent.futures.wait(futures)

        end = time.time()
        print("Total Time for extraction of files is {} secs!".format(end-start))

        Total Time for extraction of files is 1601.342586517334 secs!
```

Task 2: Building streaming data analytics pipeline

Implementation:



1. Architecture implementation:
    - Created streaming data analytical pipeline using GCP pub/sub and python
    - Developed a multi-process python script for data upload to GCP Big Query
    - This script can take .tsv.gz as an input, we are saving time of extracting multiple files
2. Data querying using Big Query:
    - Implemented simple queried to explore Big Query

Screenshots: File is available **here**

```
# !pip install google-cloud
# !pip install google-cloud-pubsub
# !pip install google.cloud_bigquery
# !pip install google.cloud_storage
```

```
from google.oauth2 import service_account #For GCP Account connection
from google.cloud import pubsub_v1 # For PubSub Client
from google.cloud import bigquery # For BigQuery Client
from google.cloud import storage # For Cloud Storage Client
import json # For Message syntax
import os
```

```
#Setting up the credential file
cred = service_account.Credentials.from_service_account_file('../Configuration/INFO7250-b2d76e3086d3.jso
n')
```

```
#Setting up the Configuration Variables:
project_id = "info7250"
project_name = "info7250"
topic_name = "info7250-topic"
bucket_name = "aws-review-data"
subscription_name = "info7374-subscription"
dataset_name = "AWS_Product_Review"
table_name = "Reviews"
job_name = "INFO7250_DataFlow"
```

```
#Creating a Cloud Storag Bucket:
storage_client = storage.Client(project=project_id,credentials=cred)
bucket = storage_client.create_bucket(bucket_name)
print('Bucket {} created.'.format(bucket.name))
```

```
Bucket aws-review-data created.
```

```
#Creating PubSub Topic:
publisher = pubsub_v1.PublisherClient(credentials=cred)
topic_path = publisher.topic_path(project_id, topic_name)
topic = publisher.create_topic(topic_path)
print('Topic created: {}'.format(topic))
```

```
Topic created: name: "projects/info7250/topics/info7250-topic"
```

```
#Creating PubSub Subscription:
subscriber = pubsub_v1.SubscriberClient(credentials = cred)
topic_path = subscriber.topic_path(project_id, topic_name)
subscription_path = subscriber.subscription_path(project_id, subscription_name)
subscription = subscriber.create_subscription(subscription_path, topic_path)
print('Subscription created: {}'.format(subscription))
```

```
Subscription created: name: "projects/info7250/subscriptions/info7374-subscription"
topic: "projects/info7250/topics/info7250-topic"
push_config {
}
ack_deadline_seconds: 10
message_retention_duration {
  seconds: 604800
}
expiration_policy {
  ttl {
    seconds: 2678400
  }
}
```

```
#Creating a BigQuery Dataset:
client = bigquery.Client(project=project_id, credentials=cred)
dataset_id = client.project+"."+dataset_name
dataset = bigquery.Dataset(dataset_id)
dataset.location = "US"
dataset = client.create_dataset(dataset)
print("Created dataset {}.{}".format(client.project, dataset.dataset_id))
```

```
Created dataset info7250.AWS_Product_Review
```

Dataset schema file available here

```
#Import Table schema from a JSON file:
schema = json.loads(open('../Configuration/Reviews').read())
table_schema = []
for line in schema:
    col = bigquery.SchemaField(name=line['name'], field_type=line['type'], mode=line['mode'], description=
line['description'])
    table_schema.append(col)
```

```
#Creating a BigQuery Table under the Datset created:
table_id = project_id+"."+dataset_name+"."+table_name
table = bigquery.Table(table_id, schema=table_schema)
table = client.create_table(table)
print("Created table {}.{}.{}".format(table.project, table.dataset_id, table.table_id))
```

```
Created table info7250.AWS_Product_Review.Reviews
```

gcloud dataflow jobs run JOB_NAME \ --gcs-location gs://dataflow-templates/latest/PubSub_Subscription_to_BigQuery \ --parameters \ inputSubscription=projects/YOUR_PROJECT_ID/subscriptions/YOUR_SUBSCRIPTION_NAME,\ outputTableSpec=YOUR_PROJECT_ID:YOUR_DATASET.YOUR_TABLE_NAME

```
#Creating a Dataflow for PubSub to Big Query using template:
command = "gcloud dataflow jobs run {} --gcs-location gs://dataflow-templates/latest/PubSub_Subscription_t
o_BigQuery ".format(job_name)
command += "--parameters "
command += "outputTableSpec={}:{}.{},".format(project_id, dataset_name, table_name)
command += "inputSubscription=projects/{}/subscriptions/{}".format(project_name, subscription_name)


print(command)

os.system(command)
```

```
gcloud dataflow jobs run INFO7250_DataFlow --gcs-location gs://dataflow-templates/latest/PubSub_Subscripti
on_to_BigQuery --parameters outputTableSpec=info7250:AWS_Product_Review.Reviews,inputSubscription=project
s/info7250/subscriptions/info7374-subscription
0
```

Data real-time streaming script

```
#Considering you have already created Streaming Dataflow manually using GCP Console for PubSub Subscriptio
n to BiqQuery Table.
#Publish a real-time event/message on PubSub Topic.

message_data = {
                "marketplace":"Sunil",
                "customer_id":18778586,
                "review_id":"RDIJS7QYB6XNR",
                "product_id":"B00EDBY7X8",
                "product_parent":122952789,
                "product_title":"Pramod",
                "product_category": "Toys",
                "star_rating":5,
                "helpful_votes":0,
                "total_votes":0,
                "vine": False,
                "verified_purchase": True,
                "review_headline": "Five Stars",
                "review_body": "Excellent!!!",
                "review_date": "2015-08-31"
                }
```

```
#Formatting message data before publishing:
message_data = json.dumps(message_data)
message_data = message_data.encode('utf-8')

#Publishing a message on the PubSub Topic Created:
response = publisher.publish(topic_path, message_data , origin='python-sample')
```

Batch Processing script: File available here

```
import os
import glob
import time
import concurrent.futures
```

```
input_folder = r"D:\Semester4\INFO7250-BigData\Project/Data/"
file_type = ".tsv.gz"

NUM_WORKERS = int(os.environ['NUMBER_OF_PROCESSORS'])
print("There are {} number of processors".format(NUM_WORKERS))
```

```
There are 8 number of processors
```

```
files = glob.glob(input_folder+"*.tsv.gz")
print("Total Number of file to upload: {}".format(len(files)))
```

```
Total Number of file to upload: 51
```

```
def uploadFile(filename):
    path, file = os.path.split(filename)
    print("Uploading file {} on big query".format(file))
    command = "bq --location=US load --null_marker=NULL --quote \"\" "
    command += "-E UTF-8 --autodetect --source_format=CSV --field_delimiter \\t AWS_Product_Review.Reviews
" + filename
    os.system(command)
    return None
```

```
for file in files[:1]:
    file = r"D:\Semester4\INFO7250-BigData\Project\Data\amazon_reviews_us_Personal_Care_Appliances_v1_00.t
sv.gz"
    uploadFile(file)
```

```
Uploading file amazon_reviews_us_Personal_Care_Appliances_v1_00.tsv.gz on big query
```

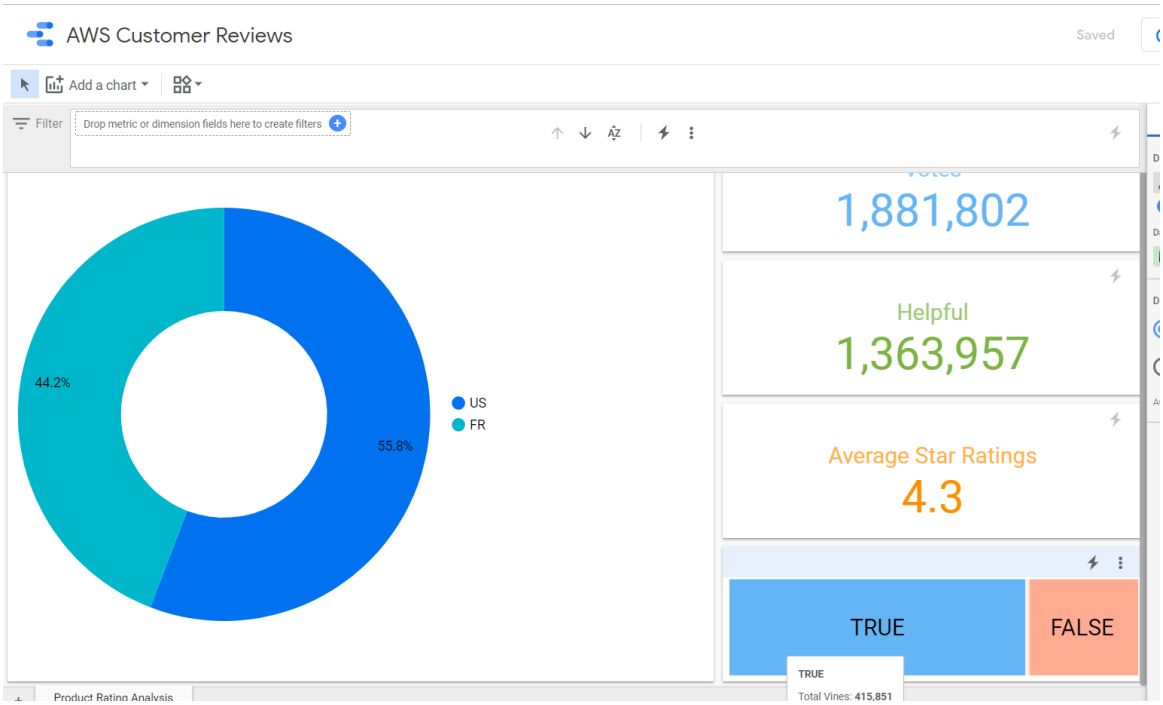
```
start = time.time()

with concurrent.futures.ThreadPoolExecutor(max_workers=NUM_WORKERS) as executor:
    futures = {executor.submit(uploadFile, file) for file in files}
    concurrent.futures.wait(futures)

end = time.time()
print("Total Time for uploading of files is {} secs!".format(end-start))
```

Task 3: Real Time data visualization

Implementation:

1. Developed interactive dashboard using **Google Data explorer**
    - Created custom fields for detail data analysis
    - Data visualization using interactive graphs
    - Export and shared the dashboard

Screenshot:



Task 4: Bloom filter

Implementation:

1. Designed and developed bloom filter for stop word, positive and negative word analysis
2. Tried importing and exporting bloom filter using pickle file in python and data output stream in java

Screenshots: File is available [here]

JAVA Implementation:

```java
import com.google.common.hash.BloomFilter;
import com.google.common.hash.Funnels;

import java.io.*;
import java.util.ArrayList;

import static com.google.common.base.Charsets.UTF_8;

public class MyBloomFilter {

    public static ArrayList<String> getList(String filename) throws IOException {

        ArrayList<String> words = new ArrayList<String>();

        File wordFile = new File(filename);

        BufferedReader br = new BufferedReader(new FileReader(wordFile));

        String st;
        while ((st = br.readLine()) != null)
            words.add(st);

        return words;
    }
```

```
    public static void main( String [] args) throws IOException {

        ArrayList<String> positiveWords = getList("files/positive_words.txt");//getList(args[0]);
        ArrayList<String> negativeWords = getList("files/negative_words.txt");//getList(args[1]);

        float fpr = 0.0001f; //Float.parseFloat(args[2]);

        BloomFilter<String> positiveWordsFilter = BloomFilter.create(Funnels.stringFunnel(UTF_8), positiveWords.size(), fpr);

        for (String word : positiveWords)
            positiveWordsFilter.put(word);

        BloomFilter<String> negativeWordsFilter = BloomFilter.create(Funnels.stringFunnel(UTF_8), positiveWords.size(), fpr);

        for (String word : negativeWords)
            negativeWordsFilter.put(word);


        System.out.println(positiveWordsFilter.mightContain("winner"));
        System.out.println(positiveWordsFilter.mightContain("pramod"));
        OutputStream os = new FileOutputStream("PositiveBloomFilter.txt");
        positiveWordsFilter.writeTo(os);

        InputStream bf = new FileInputStream("PositiveBloomFilter.txt");
        BloomFilter loadBloomFilter = BloomFilter.readFrom(bf,Funnels.stringFunnel(UTF_8));

        System.out.println(loadBloomFilter.mightContain("winner"));
        System.out.println(loadBloomFilter.mightContain("Pramod"));



         System.out.println(negativeWordsFilter.mightContain("absurd"));
         System.out.println(negativeWordsFilter.mightContain("pramod"));
         OutputStream os1 = new FileOutputStream("NegativeBloomFilter.txt");
         negativeWordsFilter.writeTo(os1);

         InputStream bf1 = new FileInputStream("NegativeBloomFilter.txt");
         BloomFilter loadBloomFilter1 = BloomFilter.readFrom(bf1,Funnels.stringFunnel(UTF_8));

         System.out.println(loadBloomFilter1.mightContain("absurd"));
         System.out.println(loadBloomFilter1.mightContain("Pramod"));


    }
}
```

Python Implementation: File is available [here]

```
import math
import mmh3
from bitarray import bitarray

class BloomFilter(object):

        '''
        Class for Bloom filter, using murmur3 hash function
        '''

        def __init__(self, items_count,fp_prob):
                '''
                items_count : int
                        Number of items expected to be stored in bloom filter
                fp_prob : float
                        False Positive probability in decimal
                '''
                # False posible probability in decimal
                self.fp_prob = fp_prob

                # Size of bit array to use
                self.size = self.get_size(items_count,fp_prob)

                # number of hash functions to use
                self.hash_count = self.get_hash_count(self.size,items_count)

                # Bit array of given size
                self.bit_array = bitarray(self.size)

                # initialize all bits as 0
                self.bit_array.setall(0)

```

```
        def add(self, item):
                '''
                Add an item in the filter
                '''
                digests = []
                for i in range(self.hash_count):

                        # create digest for given item.
                        # i work as seed to mmh3.hash() function
                        # With different seed, digest created is different
                        digest = mmh3.hash(item,i) % self.size
                        digests.append(digest)

                        # set the bit True in bit_array
                        self.bit_array[digest] = True

        def check(self, item):
                '''
                Check for existence of an item in filter
                '''
                for i in range(self.hash_count):
                        digest = mmh3.hash(item,i) % self.size
                        if self.bit_array[digest] == False:

                                # if any of bit is False then,its not present
                                # in filter
                                # else there is probability that it exist
                                return False
                return True


        @classmethod
        def get_size(self,n,p):
                '''
                Return the size of bit array(m) to used using
                following formula
                m = -(n * lg(p)) / (lg(2)^2)
                n : int
                        number of items expected to be stored in filter
                p : float
                        False Positive probability in decimal
                '''
                m = -(n * math.log(p))/(math.log(2)**2)
                return int(m)

        @classmethod
        def get_hash_count(self, m, n):
                '''
                Return the hash function(k) to be used using
                following formula
                k = (m/n) * lg(2)

                m : int
                        size of bit array
                n : int
                        number of items expected to be stored in filter
                '''
                k = (m/n) * math.log(2)
                return int(k)
```

Analysis:

Bloom Filter files:

NegativeBloomFilter.txt Size: 4.7 KB

PositiveBloomFilter.txt Size: 4.7 KB

Word list files:

negative_words.txt Size: 43.7 KB

positive_words.txt Size: 16.8 KB

Task 5: Map Reduce for Sentiment Analysis

Implementation:

1. Created map-reduce jobs in python on Hadoop using Hadoop streaming jar
2. Tested map-reduce on local machine

3. Implemented both MR jobs with bloom filter and without bloom filter
4. Optimized the MR jobs with Hadoop best practices
    - Using compact data type for the job
    - Using combiner
    - Running MR jobs on Hadoop cluster on GCP and AWS
    - Implemented distributed caches

JAVA Implementation of MR job available here:

INFO7250_Project

Mapper

```java
package Project;

import com.google.common.hash.BloomFilter;
import com.google.common.hash.Funnels;
import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import static com.google.common.base.Charsets.UTF_8;

public class ProjectMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    BloomFilter postitiveBloomFilter, negativeBloomFilter;
    Text word = new Text();
    IntWritable one = new IntWritable(1);

    public ProjectMapper() throws IOException {
    }



    @Override
    protected void setup(Context context) throws IOException, InterruptedException {
        try{
            Path[] Files = DistributedCache.getLocalCacheFiles(context.getConfiguration());
            InputStream bf = new FileInputStream(Files[0].getName());
            postitiveBloomFilter = BloomFilter.readFrom(bf,Funnels.stringFunnel(UTF_8));
            bf.close();

            InputStream bf1 = new FileInputStream(Files[1].getName());
            negativeBloomFilter = BloomFilter.readFrom(bf1,Funnels.stringFunnel(UTF_8));
            bf1.close();

        } catch(IOException ex) {
            System.err.println("Exception in mapper setup: " + ex.getMessage());
        }
    }


    public void map(LongWritable key, Text value, ProjectMapper.Context context) throws IOException, InterruptedException{
        String line = value.toString();
        String[] tokens = line.split("\t");
        if (tokens.length >= 13){
            String[] words = tokens[13].trim().toLowerCase().split(" ");
            for (String token : words){
                if(!token.matches(".*\\d.*")){
                    if(postitiveBloomFilter.test(token)) {
                        word.set("Positive: "+ token.replaceAll("\\W", " "));
                        context.write(word, one);
                    }
                    else if(negativeBloomFilter.test(token)) {
                        word.set("Negative: "+ token.replaceAll("\\W", " "));
                        context.write(word, one);
                    }
                }
            }
        }
    }
}
```

Reducer:

```java
package Project;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ProjectReducer extends Reducer<Text, IntWritable, Text, IntWritable> {


    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IO

        int count = 0;

        for(IntWritable val: values){
            count += val.get();
        }

        IntWritable total = new IntWritable(count);
        context.write(key, total);

    }
}
```

Driver:

```java
package Project;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

public class Driver extends Mapper<Text, Text, Text, IntWritable> {

    public static void main (String[] args) throws IOException, ClassNotFoundException, InterruptedException {

        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "INFO7250 Final Project");

        job.setJarByClass(Driver.class);
        job.setMapperClass(ProjectMapper.class);
//        job.setMapperClass(NormalMapper.class);
        job.setReducerClass(ProjectReducer.class);

        job.setCombinerClass(ProjectReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);


        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        DistributedCache.addCacheFile(new Path(args[2]).toUri(), job.getConfiguration());
        DistributedCache.addCacheFile(new Path(args[3]).toUri(), job.getConfiguration());

        System.exit(job.waitForCompletion(true) ? 0:1);
    }
}
```

Python Implementation of MR job is available here:

Python

Mapper

```python
#!/usr/bin/env python
import sys
import re
from BloomFilter import BloomFilter

stop_words = []
sw_file = open('stop_words.txt', 'r')
for line in sw_file:
    words = line.replace(' ','').replace('\'', '').replace('\n', '').split(',')
    stop_words.extend(words)

n = len(stop_words) #no of items to add
p = 0.0001 #false positive probability

bloomf = BloomFilter(n,p)

for item in stop_words:
    bloomf.add(item)

RE_D = re.compile('\d')

for line in sys.stdin:
    #--- remove leading and trailing whitespace---
    line = line.strip().split("\t")

    if len(line) >= 13:

        #--- split the line into words ---
        review = re.findall(r'\w+', line[13])
        for word in review:
            if not bloomf.check(word) and len(word) > 1 and not RE_D.search(word): #not bloomf.check(word) and
                print('%s\t%s' % (word.lower(), "1"))
```

Reducer

```python
#!/usr/bin/env python
# coding: utf-8

import sys
# maps words to their counts
word2count = {}

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        continue

    try:
        word2count[word] = word2count[word]+count
    except:
        word2count[word] = count

# write the tuples to stdout
# Note: they are unsorted
for word in word2count.keys():
    print('%s\t%s'% ( word, word2count[word] ))
```

Task 6: Hadoop Cluster setup on GCP

Implementation:

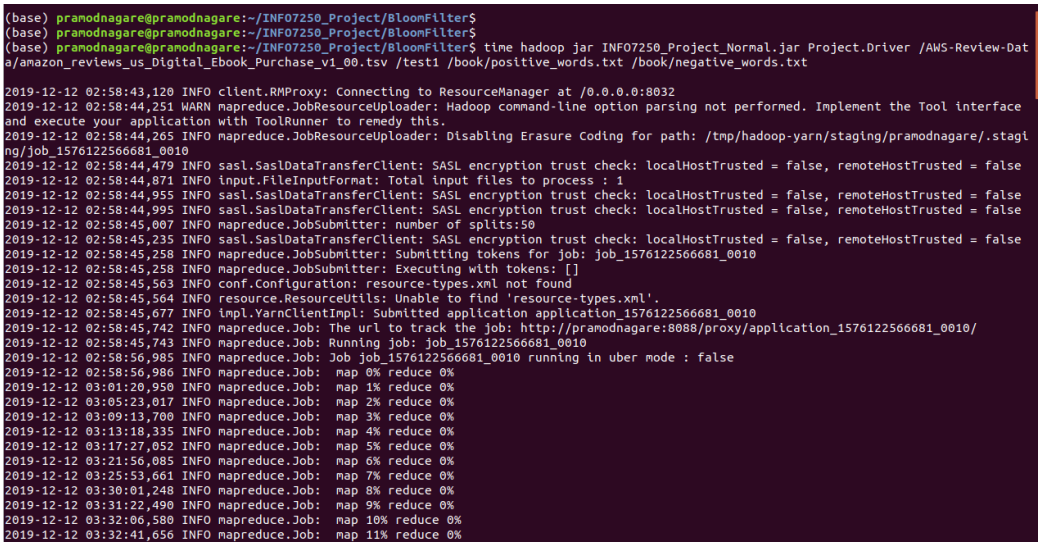
1. Created a GCP Hadoop cluster with 2 worker nodes and 1 master
2. Submitted MR jobs on cluster using GCP utility
3. Implemented persistent storage for MR jobs
4. Ran some MR jobs on cluster with SSH to master node
5. Accessed the Hadoop dashboard for the job monitoring

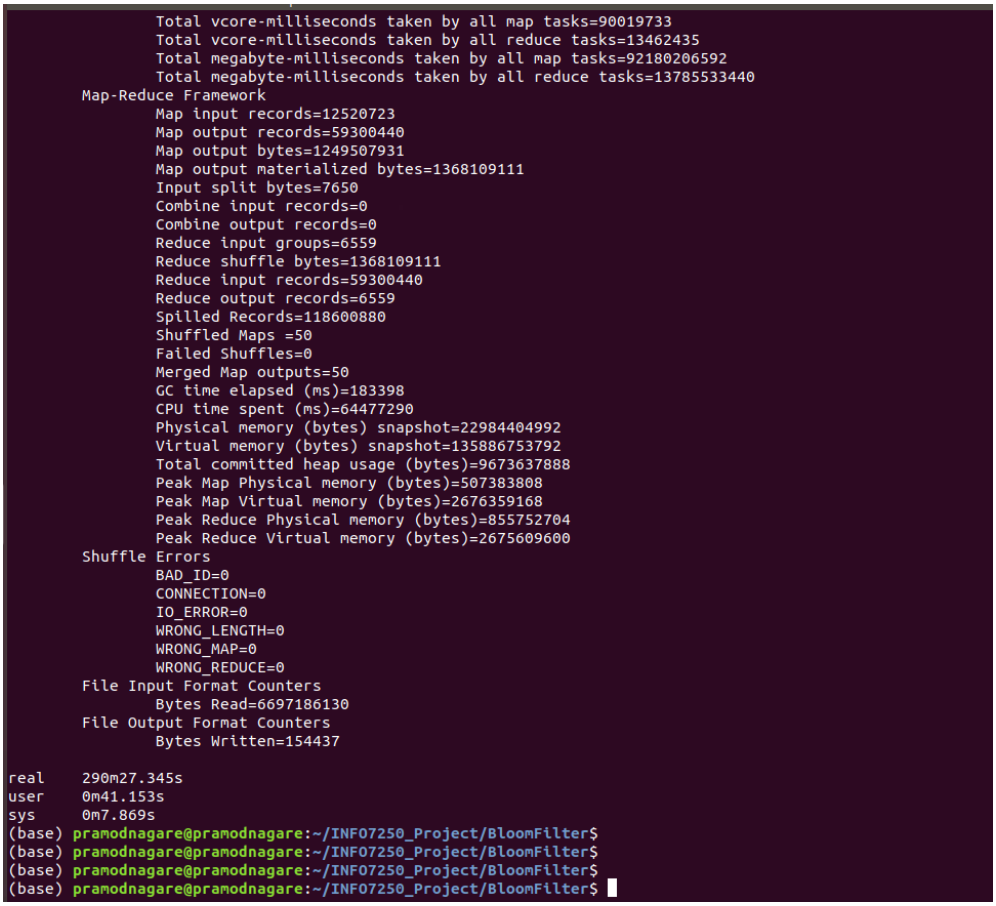
MR Job performance Time:

1. On Local machine with 1 node: **5 hrs**
2. On GCP Hadoop cluster with 2 worker node and 1 master: **1 hr. 42 min**

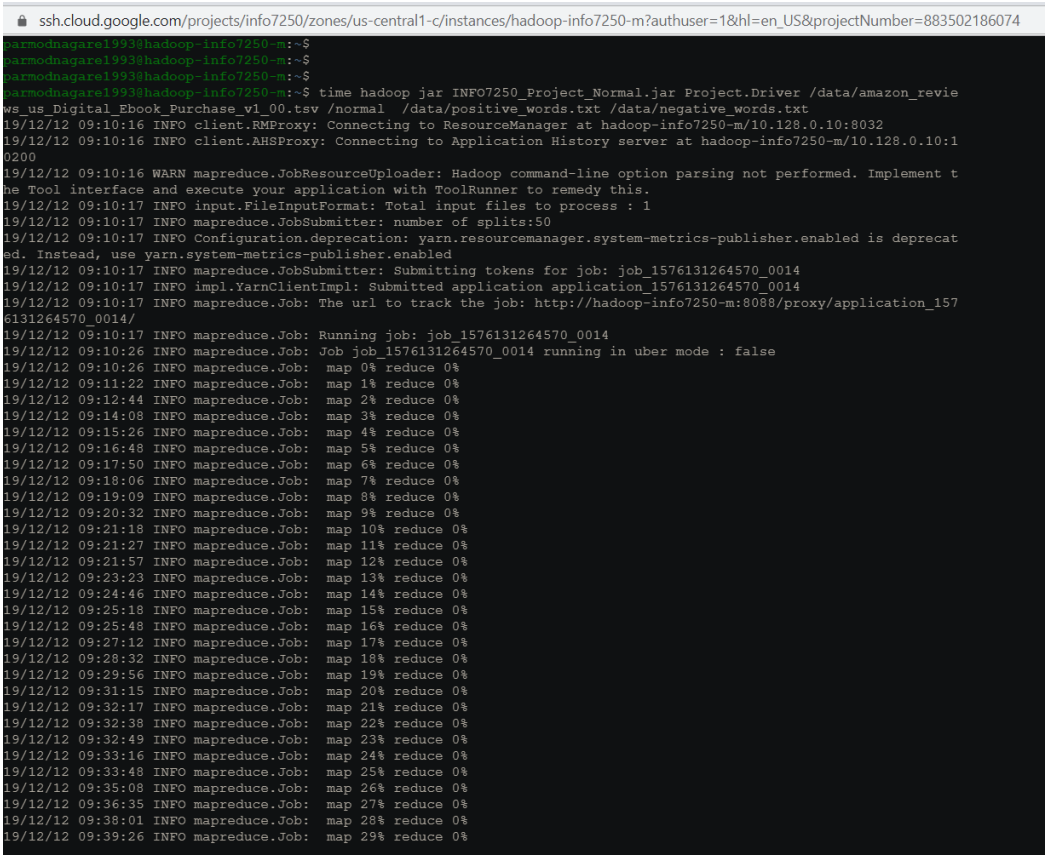
Screenshots:

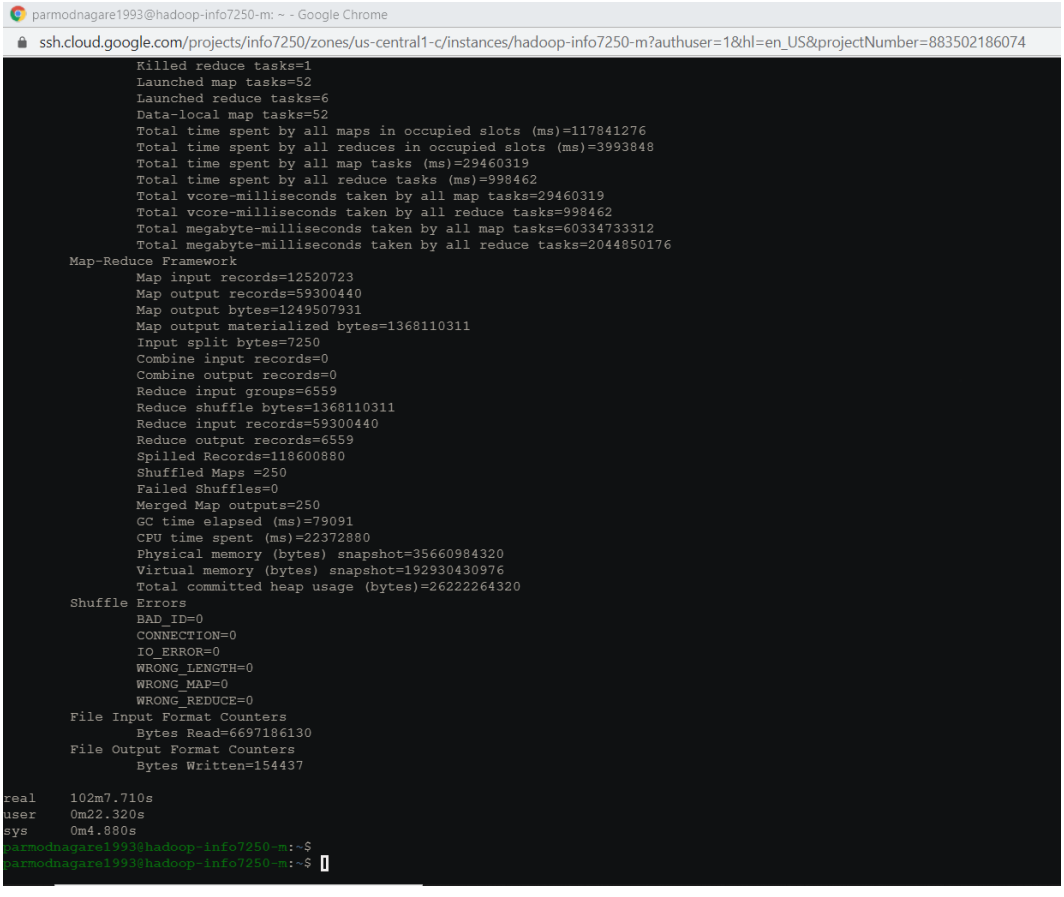
On local machine:





On GCP Hadoop cluster:





Bloom Filter Performance:

Bloom Filter files:

NegativeBloomFilter.txt Size: 4.7 KB

PositiveBloomFilter.txt Size: 4.7 KB

Word list files:

negative_words.txt Size: 43.7 KB

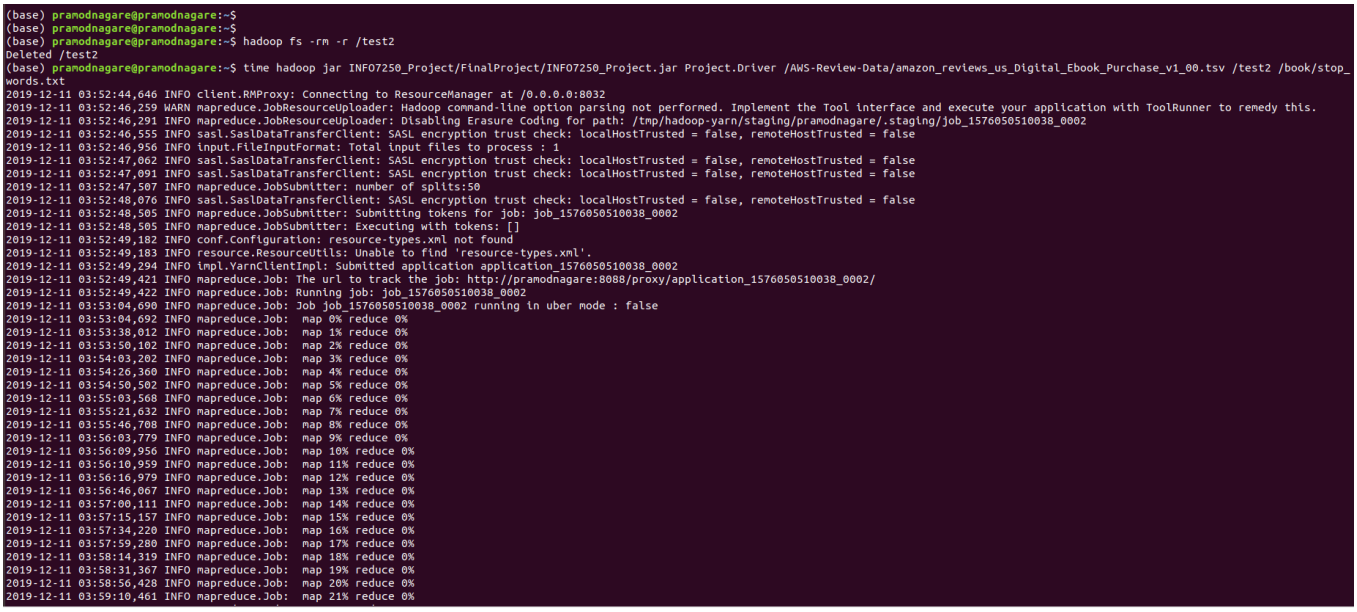
positive_words.txt Size: 16.8 KB

MR Job Performance

1. **Without bloom filter: 41 min 9 sec**
2. **With Bloom Filer: 33 min 13 sec**
3. **With optimization: 20 min 37 sec**

Screenshots:

Without Bloom Filter



```
                Total time spent by all reduces in occupied slots (ms)=2057388
                Total time spent by all map tasks (ms)=9251526
                Total time spent by all reduce tasks (ms)=2057388
                Total vcore-milliseconds taken by all map tasks=9251526
                Total vcore-milliseconds taken by all reduce tasks=2057388
                Total megabyte-milliseconds taken by all map tasks=9473562624
                Total megabyte-milliseconds taken by all reduce tasks=2106765312
        Map-Reduce Framework
                Map input records=12520723
                Map output records=407547175
                Map output bytes=4326086575
                Map output materialized bytes=5141181406
                Input split bytes=7650
                Combine input records=0
                Combine output records=0
                Reduce input groups=2468937
                Reduce shuffle bytes=5141181406
                Reduce input records=407547175
                Reduce output records=2468937
                Spilled Records=1316874350
                Shuffled Maps =50
                Failed Shuffles=0
                Merged Map outputs=50
                GC time elapsed (ms)=50803
                CPU time spent (ms)=6791920
                Physical memory (bytes) snapshot=26216099840
                Virtual memory (bytes) snapshot=135762329600
                Total committed heap usage (bytes)=22725263360
                Peak Map Physical memory (bytes)=519913472
                Peak Map Virtual memory (bytes)=2676498432
                Peak Reduce Physical memory (bytes)=957042688
                Peak Reduce Virtual memory (bytes)=2676445184
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=6697186130
        File Output Format Counters
                Bytes Written=32863155

real    41m9.809s
user    0m24.454s
sys     0m3.406s
(base) pramodnagare@pramodnagare:~$
```

With Bloom Filter

```
(base) pramodnagare@pramodnagare:~$ hadoop fs -rm -r /test1
Deleted /test1
(base) pramodnagare@pramodnagare:~$ time hadoop jar INF07250_Project/FinalProject/INFO7250_Project_BF.jar Project.Driver /AWS-Review-Data/amazon_reviews_us_Digital_Ebook_Purchase_v1_00.tsv /test1 /book/Ja
vaBloomFilter.txt
2019-12-11 03:09:11,628 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2019-12-11 03:09:13,093 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2019-12-11 03:09:13,114 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/pramodnagare/.staging/job_1576050510038_0001
2019-12-11 03:09:13,782 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-12-11 03:09:15,247 INFO input.FileInputFormat: Total input files to process : 1
2019-12-11 03:09:15,400 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-12-11 03:09:15,447 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-12-11 03:09:15,465 INFO mapreduce.JobSubmitter: number of splits:50
2019-12-11 03:09:15,974 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-12-11 03:09:16,004 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1576050510038_0001
2019-12-11 03:09:16,004 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-12-11 03:09:16,523 INFO conf.Configuration: resource-types.xml not found
2019-12-11 03:09:16,524 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2019-12-11 03:09:17,847 INFO impl.YarnClientImpl: Submitted application application_1576050510038_0001
2019-12-11 03:09:17,982 INFO mapreduce.Job: The url to track the job: http://pramodnagare:8088/proxy/application_1576050510038_0001/
2019-12-11 03:09:17,983 INFO mapreduce.Job: Running job: job_1576050510038_0001
2019-12-11 03:09:39,477 INFO mapreduce.Job: Job job_1576050510038_0001 running in uber mode : false
2019-12-11 03:09:39,479 INFO mapreduce.Job:  map 0% reduce 0%
2019-12-11 03:10:18,126 INFO mapreduce.Job:  map 1% reduce 0%
2019-12-11 03:10:24,169 INFO mapreduce.Job:  map 2% reduce 0%
2019-12-11 03:10:30,209 INFO mapreduce.Job:  map 3% reduce 0%
2019-12-11 03:10:43,308 INFO mapreduce.Job:  map 4% reduce 0%
2019-12-11 03:11:01,434 INFO mapreduce.Job:  map 5% reduce 0%
2019-12-11 03:11:12,507 INFO mapreduce.Job:  map 6% reduce 0%
2019-12-11 03:11:19,556 INFO mapreduce.Job:  map 7% reduce 0%
2019-12-11 03:11:37,649 INFO mapreduce.Job:  map 8% reduce 0%
2019-12-11 03:11:49,729 INFO mapreduce.Job:  map 9% reduce 0%
2019-12-11 03:11:55,752 INFO mapreduce.Job:  map 10% reduce 0%
2019-12-11 03:12:01,830 INFO mapreduce.Job:  map 11% reduce 0%
2019-12-11 03:12:06,855 INFO mapreduce.Job:  map 12% reduce 0%
2019-12-11 03:12:30,989 INFO mapreduce.Job:  map 13% reduce 0%
```

```
                Total time spent by all reduce tasks (ms)=1666474
                Total vcore-milliseconds taken by all map tasks=6829749
                Total vcore-milliseconds taken by all reduce tasks=1666474
                Total megabyte-milliseconds taken by all map tasks=6993662976
                Total megabyte-milliseconds taken by all reduce tasks=1706469376
        Map-Reduce Framework
                Map input records=12520723
                Map output records=407433584
                Map output bytes=4325052456
                Map output materialized bytes=5139920105
                Input split bytes=7650
                Combine input records=0
                Combine output records=0
                Reduce input groups=2468614
                Reduce shuffle bytes=5139920105
                Reduce input records=407433584
                Reduce output records=2468614
                Spilled Records=1316747462
                Shuffled Maps =50
                Failed Shuffles=0
                Merged Map outputs=50
                GC time elapsed (ms)=52332
                CPU time spent (ms)=4942800
                Physical memory (bytes) snapshot=26140684288
                Virtual memory (bytes) snapshot=135700455424
                Total committed heap usage (bytes)=22860005376
                Peak Map Physical memory (bytes)=611602432
                Peak Map Virtual memory (bytes)=2668908544
                Peak Reduce Physical memory (bytes)=962953216
                Peak Reduce Virtual memory (bytes)=2679267328
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=6697186130
        File Output Format Counters
                Bytes Written=32858684

real    33m13.193s
user    0m28.185s
sys     0m4.764s
(base) pramodnagare@pramodnagare:~$
(base) pramodnagare@pramodnagare:~$
(base) pramodnagare@pramodnagare:~$
```

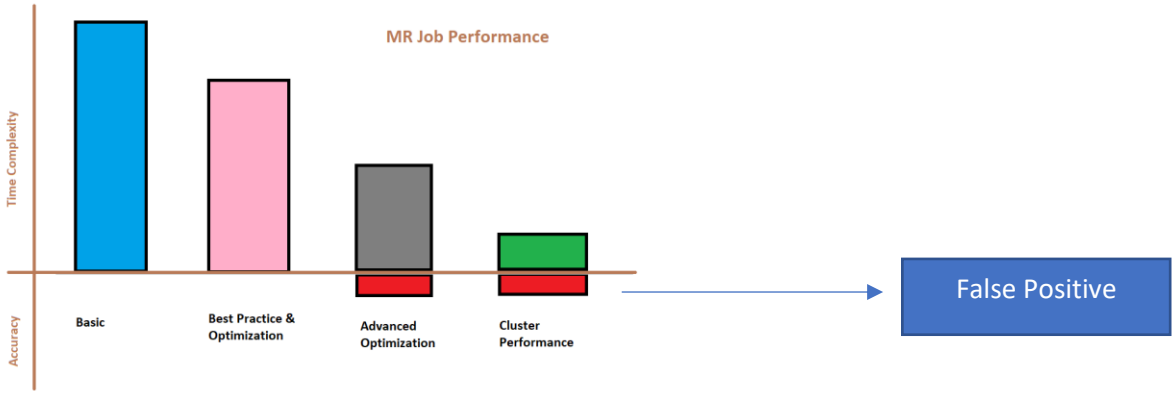
With Bloom Filter and Optimization:

```
(base) pramodnagare@pramodnagare:~$
(base) pramodnagare@pramodnagare:~$ hadoop fs -ls /
Found 3 items
drwxr-xr-x   - pramodnagare supergroup          0 2019-12-03 00:34 /AWS-Review-Data
drwxr-xr-x   - pramodnagare supergroup          0 2019-12-10 21:56 /book
drwxr-xr-x   - pramodnagare supergroup          0 2019-11-22 14:50 /hive
(base) pramodnagare@pramodnagare:~$
(base) pramodnagare@pramodnagare:~$ time hadoop jar INFO7250_Project/FinalProject/Optimized/INFO7250_Project_BF.jar Project.Driver /AWS-Review-Data/amazon_reviews_us_Digital_Ebook_Purchase_v1_00.tsv /test
1 /book/JavaBloomFilter.txt
2019-12-11 16:50:50,393 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
2019-12-11 16:50:51,604 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2019-12-11 16:50:51,649 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/pramodnagare/.staging/job_1576099692672_0002
2019-12-11 16:50:51,931 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-12-11 16:50:52,370 INFO input.FileInputFormat: Total input files to process : 1
2019-12-11 16:50:52,467 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-12-11 16:50:52,922 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-12-11 16:50:52,931 INFO mapreduce.JobSubmitter: number of splits:50
2019-12-11 16:50:53,160 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2019-12-11 16:50:53,190 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1576099692672_0002
2019-12-11 16:50:53,190 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-12-11 16:50:53,572 INFO conf.Configuration: resource-types.xml not found
2019-12-11 16:50:53,572 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2019-12-11 16:50:53,696 INFO impl.YarnClientImpl: Submitted application application_1576099692672_0002
2019-12-11 16:50:53,772 INFO mapreduce.Job: The url to track the job: http://pramodnagare:8088/proxy/application_1576099692672_0002/
2019-12-11 16:50:53,773 INFO mapreduce.Job: Running job: job_1576099692672_0002
2019-12-11 16:51:06,086 INFO mapreduce.Job: Job job_1576099692672_0002 running in uber mode : false
2019-12-11 16:51:06,087 INFO mapreduce.Job:  map 0% reduce 0%
2019-12-11 16:51:35,424 INFO mapreduce.Job:  map 1% reduce 0%
2019-12-11 16:51:52,692 INFO mapreduce.Job:  map 2% reduce 0%
2019-12-11 16:51:59,748 INFO mapreduce.Job:  map 3% reduce 0%
2019-12-11 16:52:11,835 INFO mapreduce.Job:  map 4% reduce 0%
2019-12-11 16:52:36,005 INFO mapreduce.Job:  map 5% reduce 0%
2019-12-11 16:52:43,045 INFO mapreduce.Job:  map 6% reduce 0%
2019-12-11 16:52:55,131 INFO mapreduce.Job:  map 7% reduce 0%
2019-12-11 16:53:18,283 INFO mapreduce.Job:  map 8% reduce 0%
2019-12-11 16:53:23,361 INFO mapreduce.Job:  map 9% reduce 0%
2019-12-11 16:53:26,372 INFO mapreduce.Job:  map 10% reduce 0%
2019-12-11 16:53:28,380 INFO mapreduce.Job:  map 11% reduce 0%
2019-12-11 16:53:30,387 INFO mapreduce.Job:  map 12% reduce 0%
2019-12-11 16:53:55,500 INFO mapreduce.Job:  map 13% reduce 0%
2019-12-11 16:54:03,536 INFO mapreduce.Job:  map 14% reduce 0%
2019-12-11 16:54:11,575 INFO mapreduce.Job:  map 15% reduce 0%
2019-12-11 16:54:23,633 INFO mapreduce.Job:  map 16% reduce 0%
2019-12-11 16:54:45,729 INFO mapreduce.Job:  map 17% reduce 0%
2019-12-11 16:54:55,770 INFO mapreduce.Job:  map 18% reduce 0%
2019-12-11 16:55:08,820 INFO mapreduce.Job:  map 19% reduce 0%
2019-12-11 16:55:26,903 INFO mapreduce.Job:  map 20% reduce 0%
```

```
                Total time spent by all reduces in occupied slots (ms)=938572
                Total time spent by all map tasks (ms)=6109892
                Total time spent by all reduce tasks (ms)=938572
                Total vcore-milliseconds taken by all map tasks=6109892
                Total vcore-milliseconds taken by all reduce tasks=938572
                Total megabyte-milliseconds taken by all map tasks=6256529408
                Total megabyte-milliseconds taken by all reduce tasks=961097728
        Map-Reduce Framework
                Map input records=12520723
                Map output records=407433584
                Map output bytes=4325052456
                Map output materialized bytes=142078448
                Input split bytes=7650
                Combine input records=421693592
                Combine output records=23360977
                Reduce input groups=2468614
                Reduce shuffle bytes=142078448
                Reduce input records=9100969
                Reduce output records=2468614
                Spilled Records=32461946
                Shuffled Maps =50
                Failed Shuffles=0
                Merged Map outputs=50
                GC time elapsed (ms)=47124
                CPU time spent (ms)=3777470
                Physical memory (bytes) snapshot=26111266816
                Virtual memory (bytes) snapshot=135858139136
                Total committed heap usage (bytes)=22232956928
                Peak Map Physical memory (bytes)=523751424
                Peak Map Virtual memory (bytes)=2668056576
                Peak Reduce Physical memory (bytes)=519512064
                Peak Reduce Virtual memory (bytes)=2674622464
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=6697186130
        File Output Format Counters
                Bytes Written=32858684

real    20m37.524s
user    0m21.181s
sys     0m2.800s
(base) pramodnagare@pramodnagare:~$
```

Task 7: Conclusion

- **Parallel processes for file download are limited by internet speed**
- **Parallel processes for file extraction are limited by disk I/O**
- **MR Job Performance observation**



Best Practices:

1. Use compact / optimize data type in MR jobs
2. Use of combiners
3. Use file compressions format (ex. LZO)
4. Distributed caching
5. Multiple mapper and reducer

Advanced Optimization:

1. Bloom filter