Robert Kaszubski

CSC 555

Assignment 4

1)

a) With a replication factor of 3, each output block as to be written 3 times.

89 \* 3 = 267 blocks

Each takes 1 minutes so 267 minutes

267 / 5 nodes = 53.4, we must round up so **54 minutes**.

b)

Replication: 1

89 blocks \* 1 minute = 89 minutes

89 / 10 nodes = 8.9, round up so **9 minutes**

c)

Replication: 3

89 blocks \* 3 = 267

267 / 10 = 26.7 so **27 minutes**

d)

Replication: 1

89 blocks \* 1 minute = 89 minutes

89 / 100 = .89 so **1 minute**

e)

Replication: 3

89 \* 3 = 267

267 / 100 = 2.67 so **3 minutes**

2)

a)

Two small primes: 13 (p) and 23 (q)

N = 13 \* 23 = 299

Ø(n) = (13-1)(23-1) = 264

The encryption key e has to be > 1 and < 264, and the gcd of (e,264) = 1

e can be 17

The decryption key d is then

(17)(d) mod 264 = 1

17(233) = 3961

3961 mod 264 = 1

So d is 233

Public key: KU={17,264}

Private key: KR={233,264}

b)

Let’s say the message M is 101

Encryption using public key {17,264}:

C = 10117 mod 264 = 29

c)

Decryption using private key {233, 264}

M = 29233 mod 264 = 101

Not the easiest numbers to calculate but it works.

d)

The encrypted message can’t be larger than the value of n because of the way the modulo operator works. In our case here, n is equal to 264. If our message is 1, then 1 mod 264 is 1, but if we go over 264 and our message is 265 then 265 mod 264 is also 1. Basically, 265 would just be truncated to 1. The wrong message would be sent.

I attempted the calculations and:

C = 26517 mod 264 = 1

C = 117 mod 264 = 1

So sending 265 is just like sending only 1 as the message. You wouldn’t want that as it’ll get confused for the wrong message. This is true for any number over n.

3)

a)

Find mod 3 of any key will return a remainder of either a 0,1, or 2. This output can then be used to assign keys to reducers as shown below:

Mod 3 = 0 (Reducer 0): 12, 15, 51, 87, 93

Mod 3 = 1 (Reducer 1): 1, 4, 25, 28

Mod 3 = 2 (Reducer 2): 5, 8, 11, 14, 17, 26, 50, 59, 89, 98

I labeled the reducers based on the remainder corresponding to each.

b)

We can see that in part a, the keys weren’t evenly distributed which will hinder performance, as Reducer 2 has by far the most keys. We want to partition the data evenly into 3 sets. We have 19 keys. 19/3 = 6.33. So we’ll have to split 6, 6, 7. One reducer will have an extra key because we can’t evenly split.

Custom Partitioner Code:

If key <= 12:

Return 0

Elif key <= 28:

Return 1

Else:

Return 2

Reducer 0: 1, 4, 5, 8, 11, 12

Reducer 1: 14, 15, 17, 25, 26, 28

Reducer 2: 50, 51, 59, 87, 89, 93, 98

c)

The custom partitioners main downside is that you must manually decide the partitions. This can take a long time if it is a large dataset and can lead to uneven splits. For instance, if we just base our partitions off equal intervals, certain intervals can have a higher frequency of keys than others. To calculate the partitions in my custom partitioner, I had to first find the total amount of keys, determine how many keys had to belong in each partition, then look at the data and find the cut off points. It’s more time consuming and can have worse performance due to increased complexity than a simple mod operator.

4)

Command: (ran after cd $HADOOP\_HOME)

hadoop jar hadoop-streaming-2.6.4.jar -input /user/ec2-user/lineorder.tbl -output /data/outputtest4 -mapper myMapper.py -reducer myReducer.py -file ../myReducer.py -file ../myMapper.py

Code (also attached separately to submission)

myMapper.py

A screenshot of a computer

Description automatically generated with medium confidence

myReducer.py

Text

Description automatically generated

Text

Description automatically generated  
And it created a file in the output directory:

Text

Description automatically generated

These are the same answers as when I ran the code locally using pipes.

5)

Text

Description automatically generated

Adding new columns and families:

Columns added to private: salary, dob, phone

Column added to public: position

New family: topsecret

Columns in topsecret: clearance, score

Graphical user interface, text

Description automatically generated