Assignment #1

1 Multiple Choice:

1. What are the daemons that are required to start the HDFS?

   (a) Resource Manager, Name Node

   (b) Name Node, Secondary Name Node, Data Node

   (c) Data Node, Node Manager

   (d) Data Node, Node Manager, Secondary Name Node

   Answer: B

2. You need to move a file titled weblogs into HDFS. When you try to copy the file, you can't. You know you have ample space on your DataNodes. Which action should you take to relieve this situation and store more files in HDFS?

   (a) Increase the block size on all current files in HDFS

   (b) Increase the block size on your remaining files

   (c) Decrease the block size on your remaining files

   (d) Increase the amount of memory for the NameNode

   Answer: C

3. How will the namenode decide that which datanode the data has to be written? Assume the replication factor is 3.

   (a) It chooses randomly

   (b) It chooses the datanodes which are near by in that cluster

   (c) It depends on the load on the datanodes

   (d) Both (b) and (c)

   Answer: D

4. Which command does Hadoop offer to discover missing or corrupt HDFS data?

   (a) fsck

   (b) du

   (c) dskchk

   (d) Hadoop does not provide any tools to discover missing or corrupt data; there is no need because three replicas are kept for each data block

   Answer: A

5. Which describes how a client reads a file from HDFS?

   (a) The client queries the NameNode for the block location(s). The NameNode returns the block location(s) to the client. The client reads the data directory off the DataNode(s).

   (b) The client queries all DataNodes in parallel. The DataNode that contains the requested data responds directly to the client. The client reads the data directly off the DataNode.

   (c) The client contacts the NameNode for the block location(s). The NameNode then queries the DataNodes for block locations. The DataNodes respond to the NameNode and the NameNode redirects the client to the DataNode that holds the requested data block(s).

   (d) The client contacts the NameNode for the block location(s). The NameNode contacts the DataNode that holds the requested data block. Data is transferred from the DataNode to the NameNode and then from the NameNode to the client.

   Answer: A

6. Which of the following are not true?

   (a) In standalone no daemons will be running

   (b) In Pseudo distributed mode no daemons will be running

   (c) In fully distributed mode all the daemons will be running on the same machine

   (d) (b) and (c)

   Answer: D

7. What action occurs automatically on a cluster when a DataNode is marked as dead?

   (a) The NameNode forces re-replication of all the blocks which were stored on the dead DataNode.

   (b) The next time a client submits job that requires blocks from the dead DataNode, the Resource Manager receives no heart beats from the DataNode. The Resource Manager tells the NameNode that the DataNode is dead, which triggers block re-replication on the cluster.

   (c) The replication factor of the files which had blocks stored on the dead DataNode is temporarily reduced, until the dead DataNode is recovered and returned to the cluster.

   (d) The NameNode informs the client which writes the blocks that are no longer available; the client then re-writes the blocks to a different DataNode.

   Answer: A

8. During the MapReduce Job processing, splitting of an input file happens

   (a) Randomly and decided by name node

   (b) Randomly and decided by job tracker

   (c) Line by Line and decided by Input Splitter

   (d) None of the above and need to be specified by the mapper method explicitly

   Answer: C

9. At what stage in MapReduce Job execution, the reduce function of the Job starts?

   (a) at least one mapper is ready with its output

   (b) map() and reduce() starts simultaneously

   (c) after processing for all the map tasks is completed

   (d) All of above options are possible depending on each case

   Answer: C

10. MapReduce programming model provides a way for reducers to communicate with each other?

    (a) Yes, reducers running on the same machine can communicate with each other through shared memory

    (b) No, each reducer runs independently and in isolation

    Answer: B

2 Short Answers:

You have 100 TB of data to store and process with Hadoop. The configuration of each available DataNode is as follows:

- 8 GB RAM
- 10 TB HDD
- 100 MB/s read-write speed

1. Assuming the in memory execution time is negligible, how long would it take to process the 100 TB of data using only 1 DataNode?
   Answer :
   $$\frac{100 \text{ TB } \times 1024 \text{ GB / TB} \times 1024 \text{ MB / GB}}{100 \text{ MB/s}} = 1048576 \text{ s}$$

2. Your company has a Hadoop Cluster with replication factor = 3 and block size = 64 MB. With this configuration, how many DataNodes are required to store the 100 TB data in HDFS?
   Answer:
   $$\frac{100 \times 3 \text{ TB}}{10 \text{ TB / Data Node}} = 30 \text{ Data Nodes}$$

3. How long would it take a MapReduce program to finish the same task?
   Answer:
   $$\frac{1048576 \text{ s}}{30} = 34952.53 \text{ s}$$

4. If you want to be able to finish processing the 100 TB of data in 5 minutes, how many DataNodes would you need?
   Answer:
   $$\frac{\frac{100 \text{ TB} \times 1024 \text{ GB / TB} \times 1024 \text{ MB / GB}}{100 \text{ MB / s}}}{5 \text{ mins } \times 60 \text{ s / min}} = 3495.2533 \text{ Data Nodes}$$

3 Programming:

1. Apply your MapReduce programming knowledge and write a MapReduce program to calculate the size of each word and count the number of words of that size in a given text file.

   This is the mapper, mapQ3.py.

```
#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial count of length of word is 1
        print('%s\t%s' % (len(word), 1))
```

This is the reducer, reduceQ3.py.

```
#!/usr/bin/env python

from operator import itemgetter
import sys

current_length = None
current_count = 0
length = None

print("Word Size    Word Count")

# 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
    length, count = line.split('\t', 1)

    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:  # count was not a number
        continue

    # this IF-switch only works because Hadoop sorts map output
    # by key (here: word) before it is passed to the reducer
    if current_length == length:
        current_count += count
    else:
        if current_length:
            # write result to STDOUT
            print('%s\t%s' % (current_length, current_count))
        current_count = count
        current_length = length

# do not forget to output the last length if needed!
if current_length == length:
    print('%s\t%s' % (current_length, current_count))
```

2. Apply your MapReduce programming knowledge and write a MapReduce program to process a dataset with temperature records in the weatherData.text file. Your task is to find out the dates with maximum temperature greater than 40 (a Hot Day) and minimum temperature lower than 10 (a Cold Day).

   This is the mapper, mapQ4.py.

```
#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:

        date = line[6:14]
        hotTemp = float(line[39:45])
        coldTemp = float(line[47:53])

        if hotTemp > 40:
                print('%s\t%s' % ('Hot Day', date))
        if coldTemp < 10:
                print('%s\t%s' % ('Cold Day', date))
```

   This is the reducer, reduceQ4.py.

```
#!/usr/bin/env python

from operator import itemgetter
import sys

# input comes from STDIN
for line in sys.stdin:

        line = line.strip()

        day, date = line.split('\t', 1)

        print('%s\t%s' % (day, date))
```