Final Project Report

CAP 5768

MapReduce Project

Brenda Izquierdo PID# 3886796

Walter Izquierdo PID# 4959309

Susana Palacios PID# 2136587

School of Computing and Information Sciences

Florida International University

Project Division

We contributed equally in this project by writing the conventional kNN and the kNN MapReduce programs in Java using Eclipse as platform. The Hadoop setup was performed individually. Brenda installed Hadoop in Windows, Walter in Ubuntu, and Susana in MacOS. We ran the MapReduce code in Hadoop and drew comparisons. The results presented in this document pertain to the Hadoop installation in Ubuntu.

Project Specification

As industries become heavily reliant on Big Data to make decisions, Apache Hadoop provides the software framework to process such large data sets using the MapReduce programming model. In this particular project, a task of Optical Character Recognition is asked due to the already vast volume of data that was not created using a computer word processor, such is the case of handwritten numeric values. The objective of the project is to evaluate the performance of the kNN algorithm in the MapReduce paradigm of Apache Hadoop and compare the same against the conventional implementation of kNN.

Apache Hadoop - MapReduce Setup

We initially started with the setup and configuration of a single-node in Hadoop, following the single-node architecture instructions given in class and instructions from Hadoop.apache.org for each of our different systems.

We installed Oracle JDK and set JAVA_HOME in our environment. We then proceeded to download the latest and most stable version of Hadoop. We continued with the setup of the Environment path and dependencies until running Hadoop. We used a multiple-node cluster installation to run the experiments. (Virtual Nodes)

kNN Classifier Algorithm

We parallelized the kNN Classifier into the MapReduce paradigm. MapReduce consists of two main parts: the map function and the reduce function.

The map function will receive as input a testing set and the partition of the training set that Hadoop provides. Then, it will find the k nearest neighbours in such partition for each data point in the test set. Finally, this function will output a sorted by distance list of pairs <distance, label> of the k nearest neighbours for each sample point in the testing set.

The reduce (only one reducer in our case) function will receive as input a list of pairs from each mapper (the node that runs the map function) for each sample point in the testing set. This function will combine all the <distance, label> pairs from each mapper into a single list of pairs and will sort the same. Of course this is done for each sample in the testing set. This new list will contain k*m pairs (m is the number of mappers), but will output the class label based as the most frequent label in the first k neighbours of this new <distance, label> list.

Questions

A) *On a single machine, using only the first 1000 training points, list your classifier's error (fraction of mistakes) over the test set for all odd values of k from 1 to 25 in table. For what value(s) of k is the best performance attained? (10 points)*

k	Error Rate
1	0.10164425
3	0.11111111
5	0.11659193
7	0.13203787
9	0.14798206
11	0.15346288
13	0.15894370
15	0.16691579
17	0.16841056
19	0.17140010
21	0.17588440
23	0.17987045
25	0.18036871

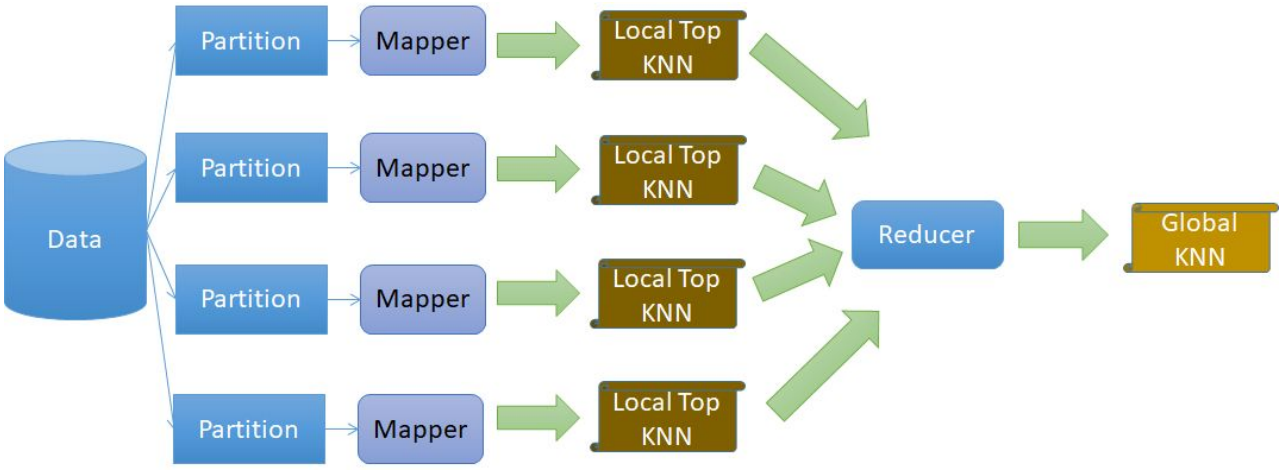
The best performance is attained when the value of k equals 1.

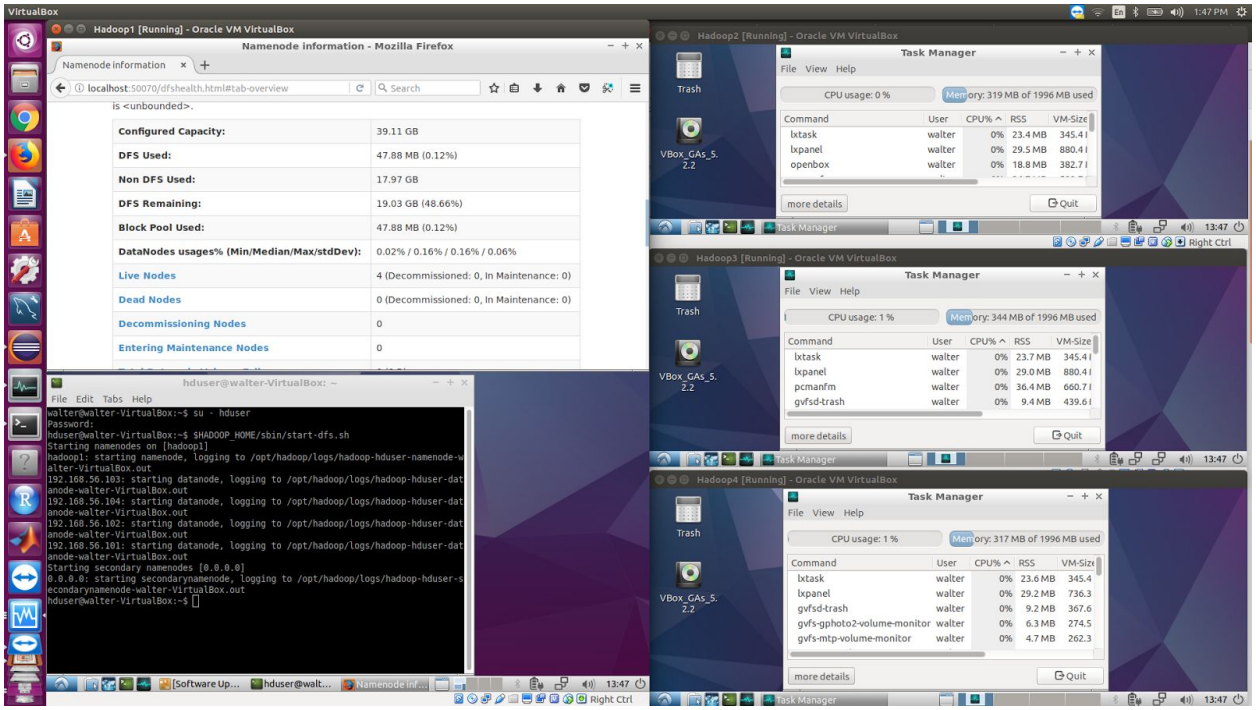
B) On a single machine, but now using the entire training set, list your classifier's error (fraction of mistakes) over the test set for all odd values of k from 1 to 25 in table. What value of k is the optimal? (10 points)

k	Error Rate
1	0.05630294
3	0.05480817
5	0.05530643
7	0.05680120
9	0.06028899
11	0.06527155
13	0.06975585
15	0.07125062
17	0.07424016
19	0.07673144
21	0.08021923
23	0.07972098
25	0.08121574

The best performance is attained when the value of k equals 3.

C) Design your own MapReduce including the configuration information (e.g. how many mappers nodes and reducers nodes). Draw the workflow of your MapReduce. (10 points)





D) The pseudocodes of the mapper and reducer function. (10 points)

Mapper:

- Create an ArrayList of floats to store the test_set
- Create an ArrayList of TreeMaps to store (distance, label) from each test_set to a partition of train_set
    - Setup(Context cxt)
        - Store the testing points into the test_set ArrayList
        - For each test_set record, initialize a new TreeMap
        - Get context configuration
    - Cleanup(Context cxt)
        - Output all elements in TreeMap (key is a constant, value is list of pairs)
    - Map(LongWritable key, VectorWritable vec, Context cxt)
        - Store the training sample into a float array
        - for(each record in test_set)
            - pull the TreeMap containing the (distance, label) tuples from the Array of TreeMaps
            - Obtain distance to the new training example
            - Store the (distance, label) tuple in the aforementioned TreeMap
            - If the TreeMap has more than k tuples, remove the last one
            - Store back the TreeMap into the ArrayList of TreeMaps

Reducer:

- Combine the TreeMaps for every for every test sample coming from different mappers into a single reduced_TreeMap (m maps * k tuple per map = k*m tuples per test sample). (already sorted because TreeMap structure)
- Find the class (label) prediction for each test sample by iterating over the first k tuples of the reduced_TreeMaps: (equivalent to keep only the first k and then iterate over the entire TreeMap)
    - Classifier(int k, TreeMap reduced_TreeMap)
        - Create a TreeMap voting to store the tuple <label, label_count>
        - for (the first k labels stored in the reduced_TreeMap)
            - increment the count of such label in voting
            - recover the maximum value of label count
            - Find the label that corresponds to such max label count

E) *On your cluster, using the entire training set, list your classifier's error (fraction of mistakes) over the test set for all odd values of k from 1 to 25 in table. What value of k is the optimal? (10 points)*

k	Error Rate
1	0.05630294
3	0.05480817
5	0.05530643
7	0.05680120
9	0.06028899
11	0.06527155
13	0.06975585
15	0.07125062
17	0.07424016
19	0.07673144
21	0.08021923
23	0.07972098
25	0.08121574

The value of k is optimal when it is 3.

F) ***Performance Comparison. (50 points)***

1) *Compare the performance (e.g. time cost) between traditional kNN algorithm and your Map-Reduced kNN algorithm. Using the first 1000, 3000, 5000, and 7000 training points respectively, list your classifier's error (fraction of mistakes) and time cost over the test set for all odd values of k from 1 to 25 of both traditional kNN algorithm and Map-Reduced kNN algorithm in table and highlight the optimal k for each training set. (30 points)*

Optimal values of k are highlighted in blue. The error rate is the same in the kNN conventional code as in the kNN MapReduce code for all the experiments (n=1000, 3000, 5000, 7000).

Training_samples = 1000			
k =	Error Rate	kNN Conventional Running Time	kNN MapReduce Running Time
1	0.10164425	0.635	2.786
3	0.11111111	0.618	2.767
5	0.11659193	0.6	2.678
7	0.13203787	0.591	2.656
9	0.14798206	0.607	2.709
11	0.15346288	0.595	2.682
13	0.1589437	0.604	2.655
15	0.16691579	0.61	2.706
17	0.16841056	0.618	2.684
19	0.1714001	0.61	2.788
21	0.1758844	0.623	2.641
23	0.17987045	0.628	2.684
25	0.18036871	0.623	2.773

Training_samples = 3000			
k =	Error Rate	kNN Conventional Running Time	kNN MapReduce Running Time
1	0.06776283	2.001	3.692
3	0.0692576	1.962	3.758

5	0.0772297	1.823	3.711
7	0.07972098	1.8	3.663
9	0.08719482	1.819	3.712
11	0.09267564	1.852	3.695
13	0.09466866	1.885	3.668
15	0.09715994	1.848	3.673
17	0.1021425	1.846	4.636
19	0.10612855	1.867	4.715
21	0.11210762	1.931	4.668
23	0.11659193	1.854	4.751
25	0.12007972	1.969	4.708

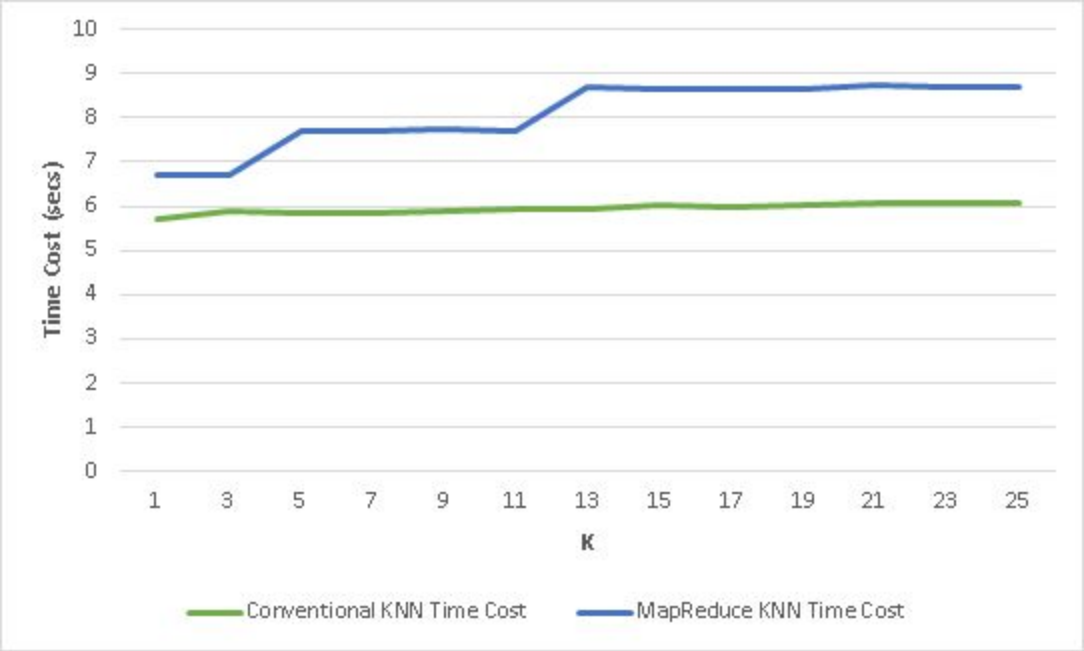
Training_samples = 5000			
k =	Error Rate	kNN Conventional Running Time	kNN MapReduce Running Time
1	0.05929248	3.642	4.661
3	0.06278027	3.626	5.74
5	0.06527155	3.684	5.744
7	0.06875934	3.728	5.699
9	0.07324365	3.792	5.695
11	0.07822621	3.754	6.688
13	0.08420528	3.833	6.675
15	0.08619831	3.954	5.689

17	0.08619831	3.894	6.653
19	0.09018435	3.915	6.73
21	0.09217738	3.931	6.683
23	0.09267564	3.923	6.694
25	0.09217738	3.993	6.712

Training_samples = 7000			
k =	Error Rate	kNN Conventional Running Time	kNN MapReduce Running Time
1	0.05729945	5.2	6.747
3	0.05381166	5.484	6.729
5	0.05829596	5.888	7.697
7	0.05779771	5.581	7.726
9	0.06228201	5.537	7.705
11	0.06676632	5.825	7.745
13	0.07125062	5.77	7.711
15	0.07274539	5.733	7.662
17	0.07523667	5.683	7.691
19	0.07922272	5.748	8.702
21	0.08071749	5.645	8.689
23	0.08021923	5.723	8.674
25	0.08121574	5.618	8.728

2) *Using the entire training set, for all odd values of k from 1 to 25, calculate the time cost of two algorithms respectively over the test set and draw the line chart. The x axis represents the k which is the odd value from 1 to 25, while y axis indicates the time cost (secs). (10 points)*

k	Conventional kNN Time Cost	MapReduce kNN Time Cost
1	5.714	6.694
3	5.903	6.721
5	5.829	7.691
7	5.866	7.697
9	5.894	7.763
11	5.922	7.703
13	5.955	8.718
15	6.005	8.666
17	5.999	8.672
19	6.012	8.671
21	6.051	8.741
23	6.061	8.703
25	6.078	8.678
Total Execution Time	77.289	105.118



3) *Compared with the conventional kNN algorithm, list the advantages and disadvantages of kNN algorithm using MapReduce model. (10 points)*

Typically, when using large datasets, the use of the MapReduce model provides a substantial reduction of the computation time proportional to the increase of number of mappers. However, in our particular experiment, since our datasets are not large enough, the conventional kNN implementation outperformed the MapReduce one.

When dealing with large-scale problems, the implementation of the kNN algorithm using MapReduce is superior in performance and represents an advantageous option compared to the sequentially run algorithm.

The main disadvantage is that MapReduce is more difficult to implement and has dependencies such as Hadoop that the conventional kNN does not have.