Northeastern University
CS6240 – Parallel Data Processing
Fall 2017, Rohan Chitnis and Saurabh Singh

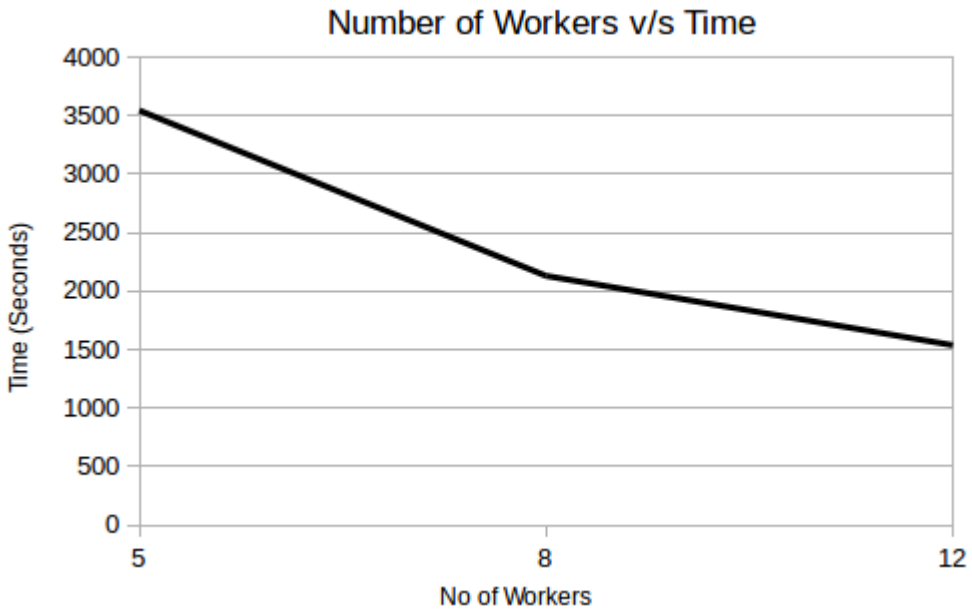
Project Report

Q1. Type of model used and parameters explored for it

We have used *Random forest classification* model of Spark-MLLib library to classify an image pixel as foreground and background.

Since Random Forest Classifier model is an ensemble of decision trees, we varied the number of trees, the maximum depth of the tree of our model. Below is some of our interesting findings :

Graph of number of machines v/s running time:



Q2. Pseudo-code with discussion

Following is the Psuedo-code of the program:

(i) Read all 4 + 1 Images into memory (as Spark RDD)

(ii) Transform the Image values from String to Double (from RDD[String] to RDD[Array(Double)])

(iii) Sample the data : Use 4 Images as the training data, 1 Image as validation data and the other as the testing data

(iv) Build the model using *Random Forest Algorithm* on training data using certain parameter values of tree depth and number of trees. Here is the short scala program for the same :

$$val\ model = RandomForest.trainClassifier(trainingData, new\ Strategy(algorithm, impurity, maximumDepth),\\ treeCount, featureSubsetStrategy, seed)$$

(v) Test the model on validation data for its accuracy. Below is scala code used for the same :

$$val\ labeledPredictions = testData.map\{labeledPoint =>\\ valpredictions = model.predict(labeledPoint.features)\\ (1, predictions)\}$$

Repeat (iv) to (v) by varying the RF model parameters if necessary.

(vi) Run the model on the testing data and report the predicted output

Detailed discussion about training and prediction.

1) Training :

- How many tasks are created during each stage of the model training process? It is equal to the number of decision trees times depth of tree, since Random forests train a set of decision trees separately, so the training can be done in parallel.
- Is data being shuffled ? Yes, Random forests train a set of decision trees separately, so the training can be done in parallel. The algorithm injects randomness into the training process so that each decision tree is a bit different using *bootstrap sampling*.
- How many iterations are executed during model training (for methods that have multiple iterations)? 1. Random Forest model runs only for one iteration.
- How did changes of parameters controlling partitioning affect the running time? Random Forest model takes care of the number of partitions based on tree count and max tree depth. For exact results, see table in $Q4$.

2) Prediction :

- How many tasks are created during each stage of the model prediction process? The use of *coalesce*(1) function results in 1 task othewise we observed 97 tasks were created for prediction (97 part files generated after prediction).
- Is data being shuffled ? No. While making prediction on a new instance, a random forest aggregates the predictions from its set of decision trees built during training proces, therefore no shuffling.
- How many iterations are executed during model prediction (for methods that have multiple iterations)? 1. Random Forest model runs only for one iteration.
- How did changes of parameters controlling partitioning affect the running time? Changes in Random Forest model parameter does not affect the running time of Prediction. However the use of *coalesce*(1) function increases the execution time because of no parallelism.

Q3. Summarizing pre-processing step

Major pre-processing steps that were followed are:

- Converted the format of Image values from string to double to be used in Random Forest algorithm.
- Since this an image recongition problem, and every row of data discribes an unique pixel of the brain scan image, we took all the features to train our model.
- Created a *LabeledPoint* containing entire neighbourhood vector as $Feature$ and foregroung/background binary value as *Label*
- The inclusion of *Gini Impurity* parameter of Random Forest model takes care of the un-homogeneity of the labels (The frequency of 0s and 1s) while training the data.

Q4. Accuracy numbers for different parameter settings

Accuracy acheived by varying the parameters (No of machines = 6) :

Max Depth	No of Trees	Accuracy	Job Time(Seconds)
3	150	0.99447	2304
3	200	0.99455	2352
5	100	0.99513	2590
5	150	0.99513	2436
5	200	0.99514	2630
6	100	0.99538	2958

Q5. Running time and speedup results for the model training and the prediction phase

Below are the timings for our runs on different number of machines ($For\ Tree\ Depth = 5$ and $No\ of\ trees = 200$):

No. of Workers	Job Time(Seconds)	Configuration
5	3544	coalesce
8	2130	coalesce
12	1538	coalesce
12	1090	no-coalesce

The *coalesce*(1) function of Spark-MLLib was used to get the entire output in a single file. The two rows for 12 machines show the effect of using the coalesce function. With coalesce(1), Spark will use only 1 partition and hence higher execution time.

Below are the seperate timings for training and prediction :

No. of Workers	Steps	Job Time(Seconds)
5	Model Training Prediction	2941 603
8	Model Training Prediction	1568 524
12	Model Training Prediction	989 510