A Novel Method for Tuning Configuration Parameters of Spark Based on Machine Learning

Guolu Wang, Jungang Xu, Ben He
School of Computer and Control Engineering, University of Chinese Academy of Sciences
State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences

Contribution

--proposed a novel method for tuning configuration parameter of spark based on machine learning

--established a model based on binary classification and multi classification

--several common machine learning algorithms for configuration parameter tuning are explored based on the model

--Decision Tree (C5.0) has good accuracy and computational performance across diverse workloads for binary classification and multi-classification

--An average of 36% performance improvement can be got with the proposed method. Moreover, with the increase of input data, the effect of performance improvement is more obvious

The Framework of Performance Modeling

--use the execution time of Spark application to represent the performance

--goal: compare the performance improvement under a given set of values of configuration parameters with the default value of these parameters

--divide the problem solving process into two steps

The Framework of Performance Modeling

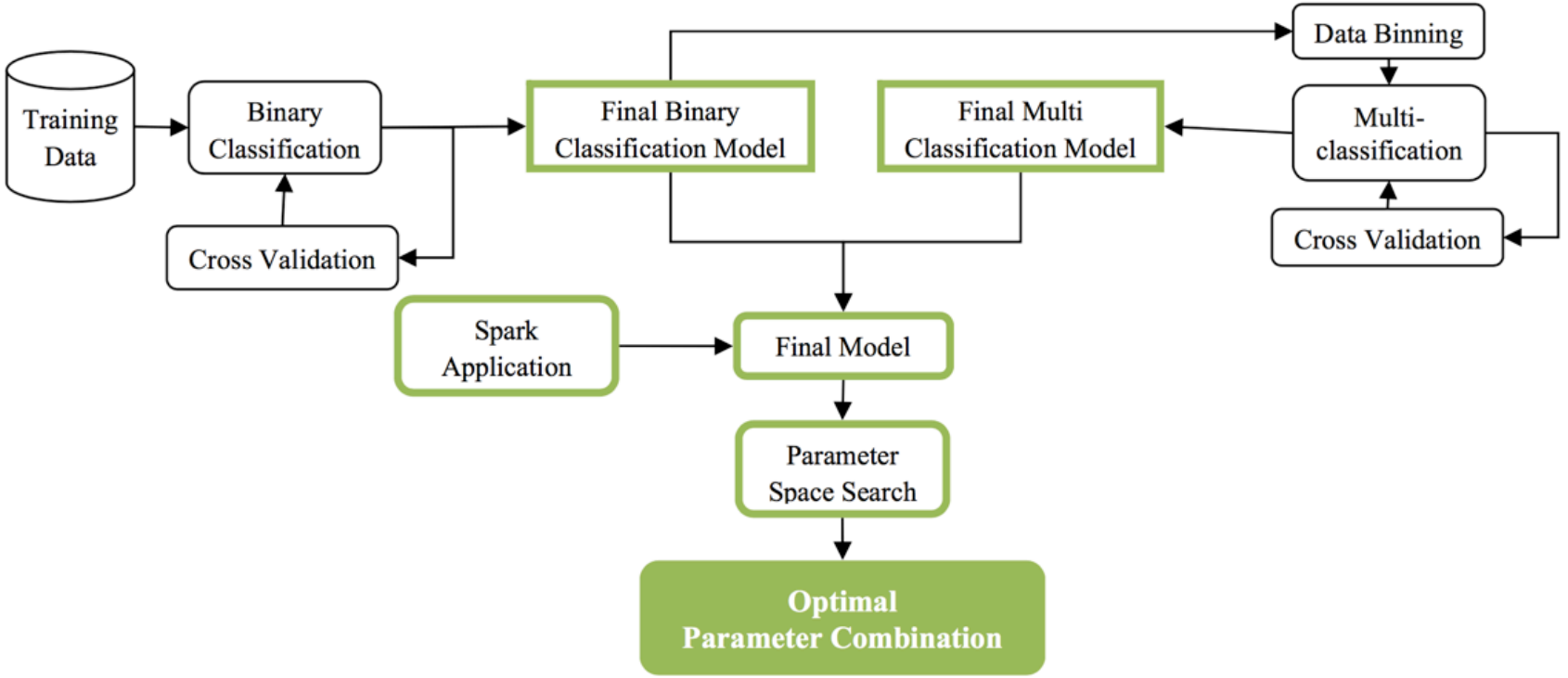


Figure 2. The framework of performance model and optimization method for Spark

--Firstly, a binary classification model is built, and the execution time of Spark application under a given set of parameters is predicted, if it is shorter than the the execution time under the default parameters, then the label of the category is set to 1, else 0

The Framework of Performance Modeling

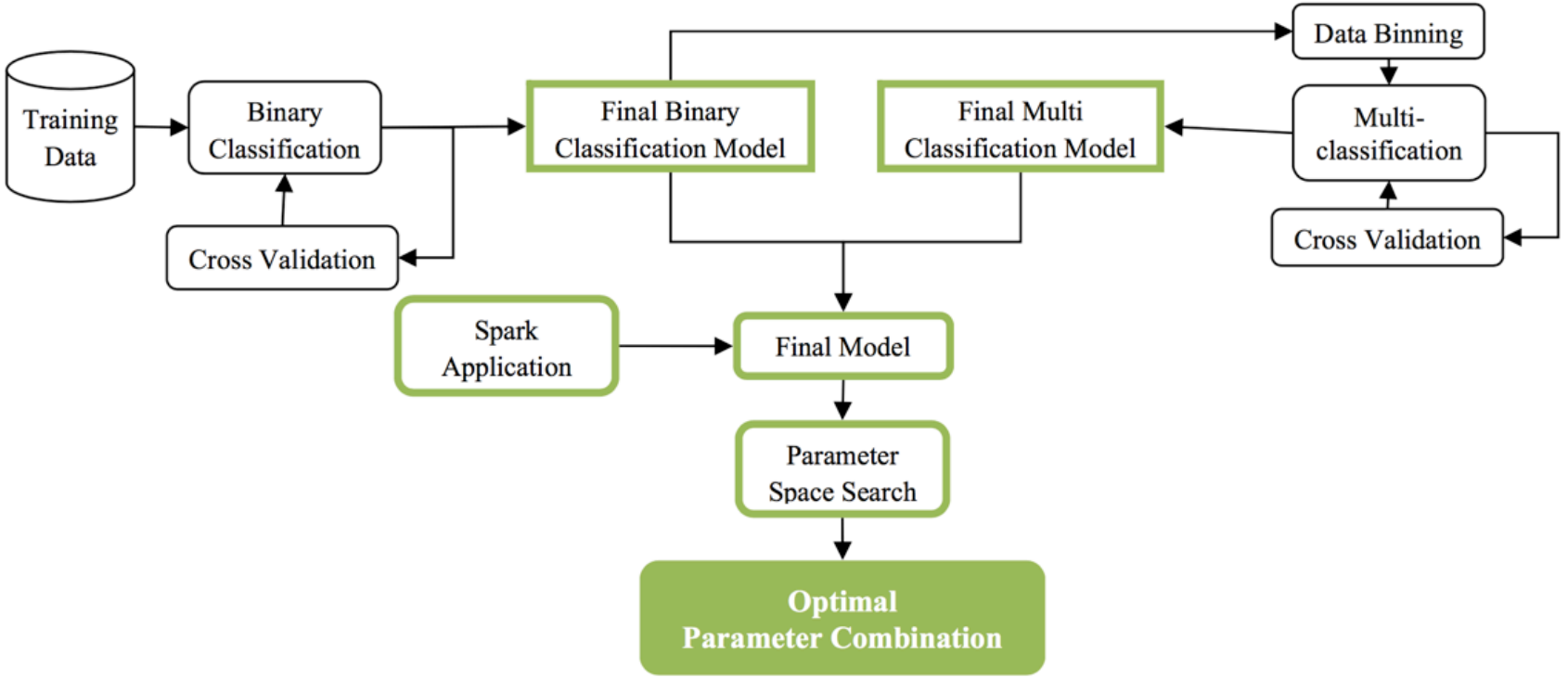


Figure 2. The framework of performance model and optimization method for Spark

--Secondly, a multi- classification model is built. The data labeled 1 is split into some bins according to the percentage of execution time reduction relative to the default parameters. For example, the data is split into 5 bins, and their execution times decrease by 5%, 10%, 15%, 20%, and 25% relative to the default parameters, which means that the data is divided into 5 categories, and then a multi-classification model can be trained according to the 5 categories

Parameter Selection

TABLE I. PARAMETERS THAT WE HAVE USED IN OUR PERFORMANCE MODELS

Parameter	Description	Default	Rules of Thumb
spark.driver.cores	Number of cores to use for the driver process	1	1--8
spark.driver.memory	Amount of memory to use for the driver process	1g	1g—4g
spark.executor.cores	Number of cores to use for the executor process	1	10-40
spark.executor.memory	Amount of memory to use per executor process	1g	2g--8g
spark.reducer.maxSizeInFlight	Maximum size of map outputs to fetch simultaneously from each reduce task	48m	24m--96m
spark.shuffle.manager	Implementation to use for shuffling data	sort	sort/hash
spark.shuffle.compress	Whether to compress map output files	true	true/false
spark.shuffle.spill.compress	Whether to compress data spilled during shuffles	true	true/false
spark.broadcast.compress	Whether to compress broadcast variables before sending them	true	true/false
spark.io.compression.codec	The codec used to compress internal data	Snappy	lz4/lzf/snappy
spark.broadcast.blockSize	Size of each piece of a block for TorrentBroadcastFactory	0.2	0.1-0.3
spark.default.parallelism	Default number of partitions in RDDs returned by transformations like join, reduceByKey and parallelize when not set by user.	0.6	0.4-0.8
spark.speculation	Whether to perform speculative execution of tasks.	4m	2-8m
InputDataSize	The size of the input dataset	NULL	NULL

-- The first thirteen parameters are configuration parameters of Spark and the last one is the size of input data.
-- the ‘default’ column lists the default value of each parameter
-- the ‘rules of thumb’ column lists the values of each parameter that the industry recommends

Data Collection

-- Four kinds of typical workloads from BigDataBench are selected to build performance model, including Sort, Wordcount, Grep and NavieBayes

-- which parameter values should explore while collecting the training data
  -- exhaustive search over different small subsets of the parameter space (exploring a specific range of parameter values)
  -- random exploration that picks parameter values uniformly random from different range of values are used while collecting data
  -- combination of these two approaches
  -- A parameter list of 500 records for each workload is generated

-- the workloads are executed three times for every parameter configuration, which results in about 1500 training points for each workload

Model Selection

--Decision Tree

--SVM

-- Logistic regression

-- Artificial neuron network

Parameter Space Search

-- After getting the final model, we can use it to predict the execution time of Spark application in a given set of parameters

-- we cannot penetrate the whole parameter space

-- an optimized search algorithm to reduce the search space should be used –Recursive Random Search

-- RRS algorithm can provide probabilistic guarantees on how close the searching it finds is to the optimal parameter combination

Evaluation Metrics

-- computational performance
  -- the time for building the model and the time for making a single prediction using the model are evaluated

-- prediction accuracy

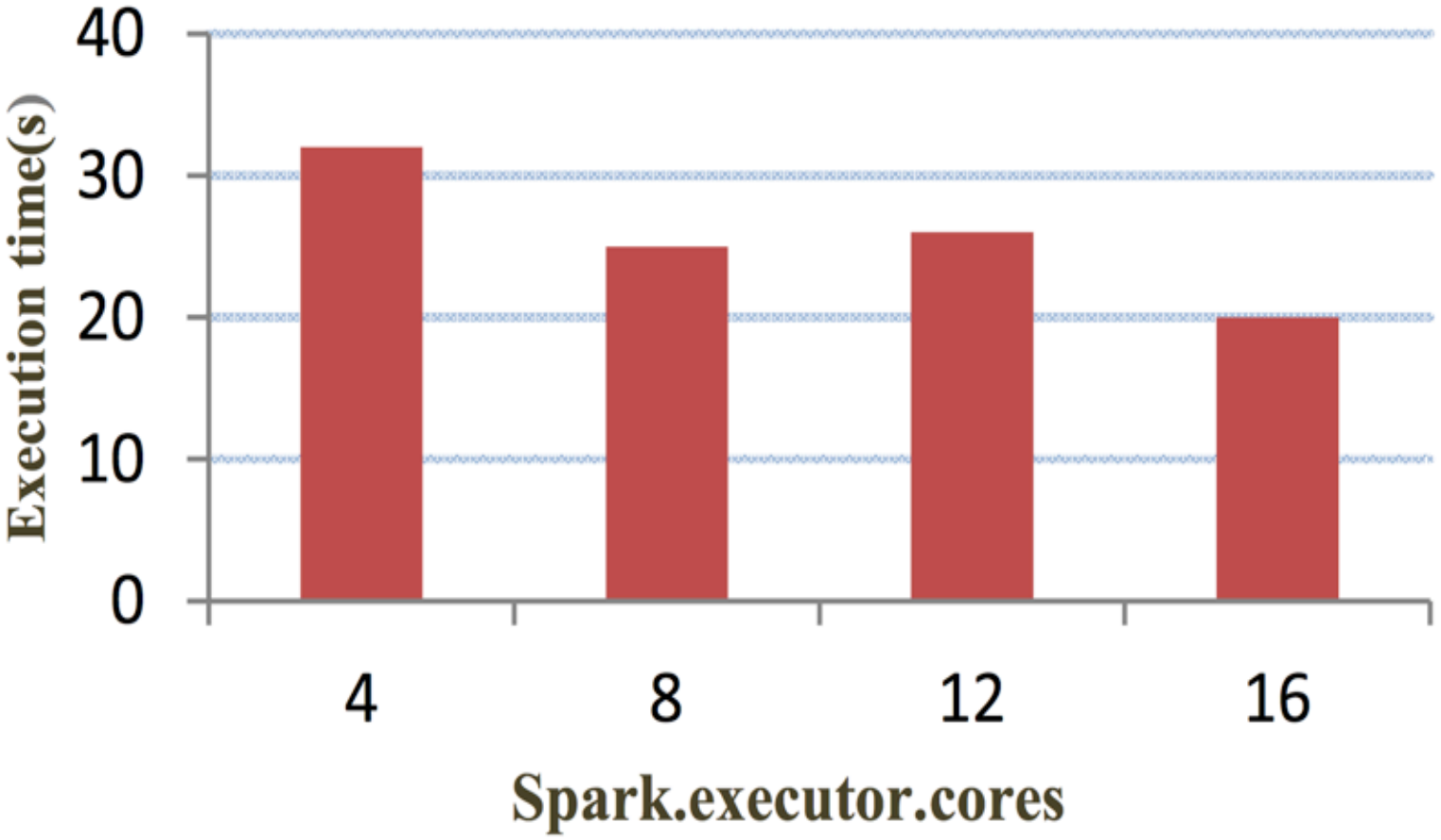


Figure 3. The execution time with different spark.executor.cores

-- workload is the WordCount in BigDataBench with 5GB input data on the 4-node IBM cluster

-- the execution times change a lot under different values of spark.executor.cores

-- the maximum change rate of execution time is 37%

Experiments--Necessity and Effectivity of Tuning Configuration Parameter of Spark

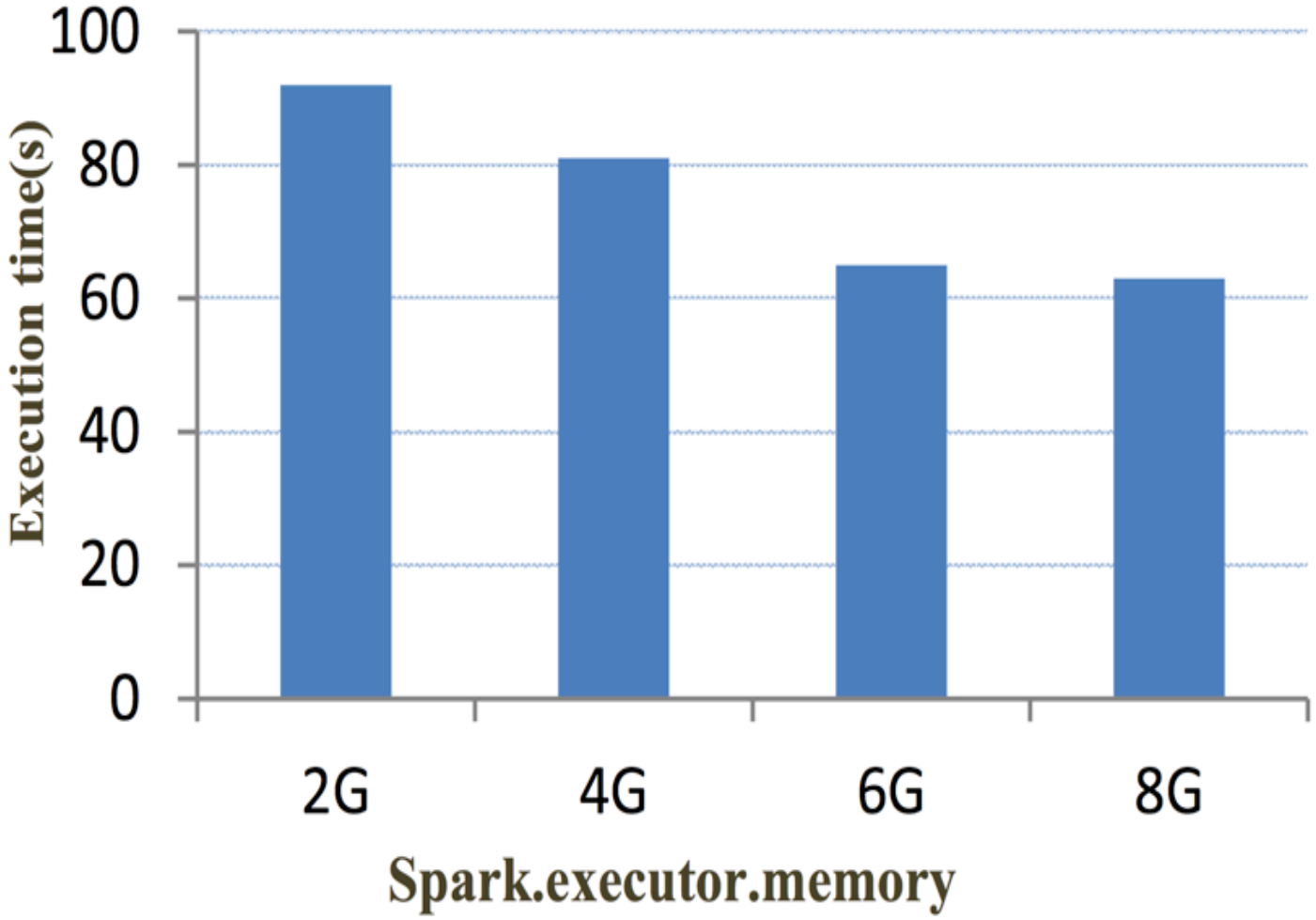


Figure 4. The execution time with different spark.executor..memory

-- The workload is the Grep in BigDataBench with 10GB input data on the 4-node IBM cluster

-- the execution times change a lot under different values of spark.executor.memory

-- the maximum change rate of execution time is 29%

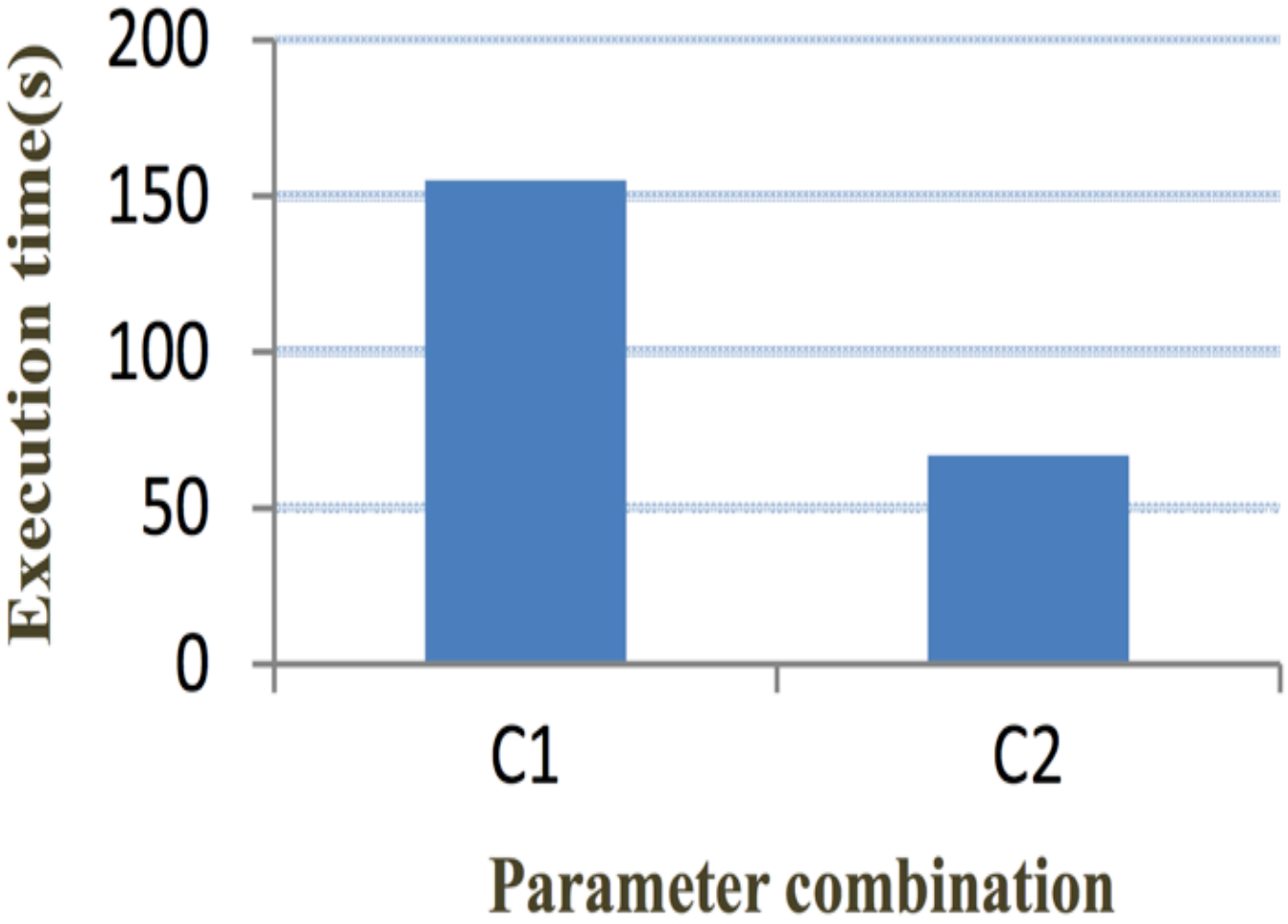


Figure 5. The execution time with two sets of parameters

-- First 13 parameters in Table I are selected and combined as parameter set C

-- *C1*=[*4,1g,8,4g,96m,sort,false,false,false,lz4,8m,100,true*]
-- *C2*=[*2,1g,16,2g,24m,hash,true,false,false,lzf,8m,110,true*]

-- workload is the Sort in BigDataBench with 1GB input data on the 4-node IBM cluster

-- tuning only a small number of parameters can reduce the execution time by half

Experiments--Evaluation of accuracy and computational performance of the models

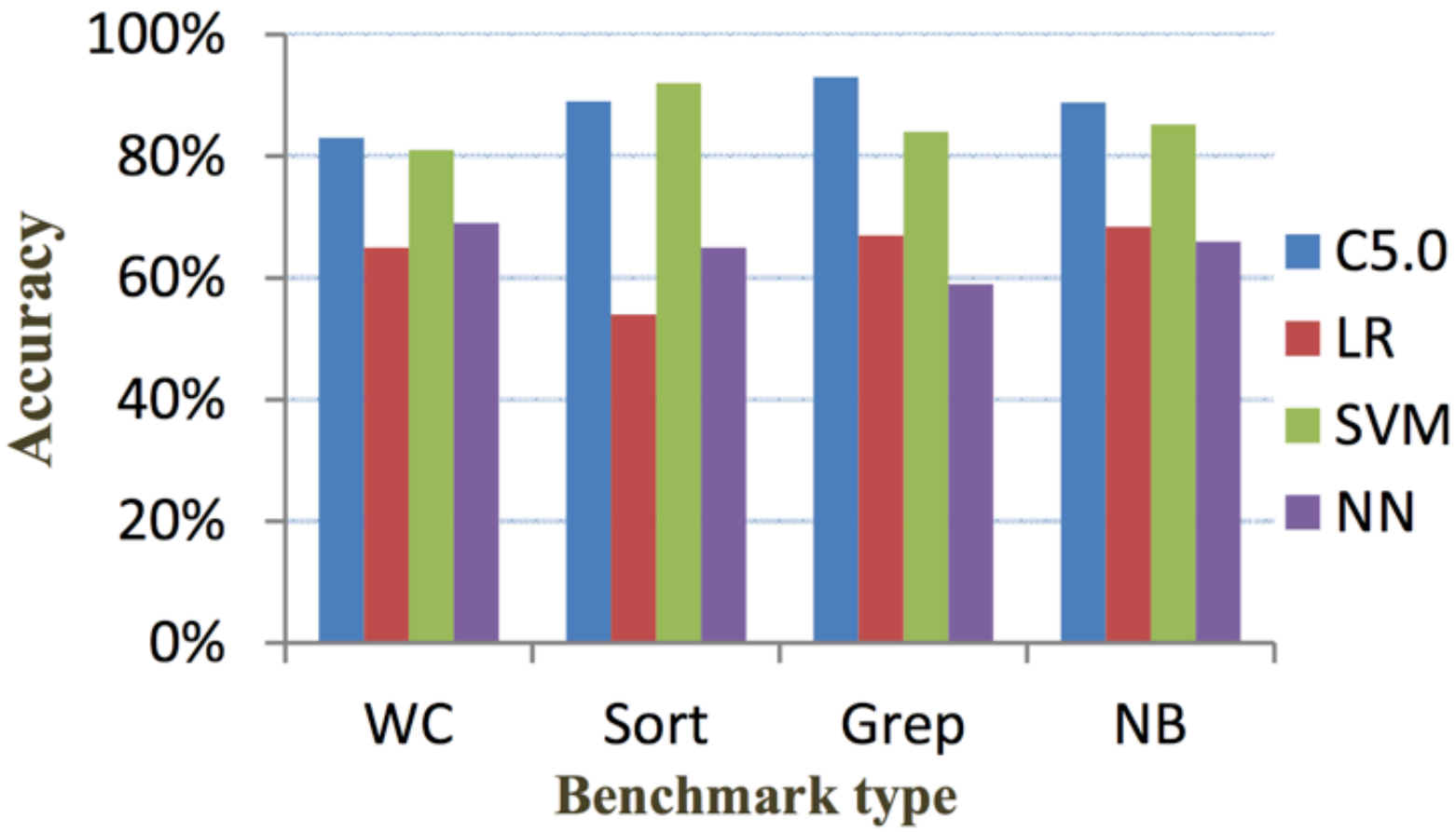
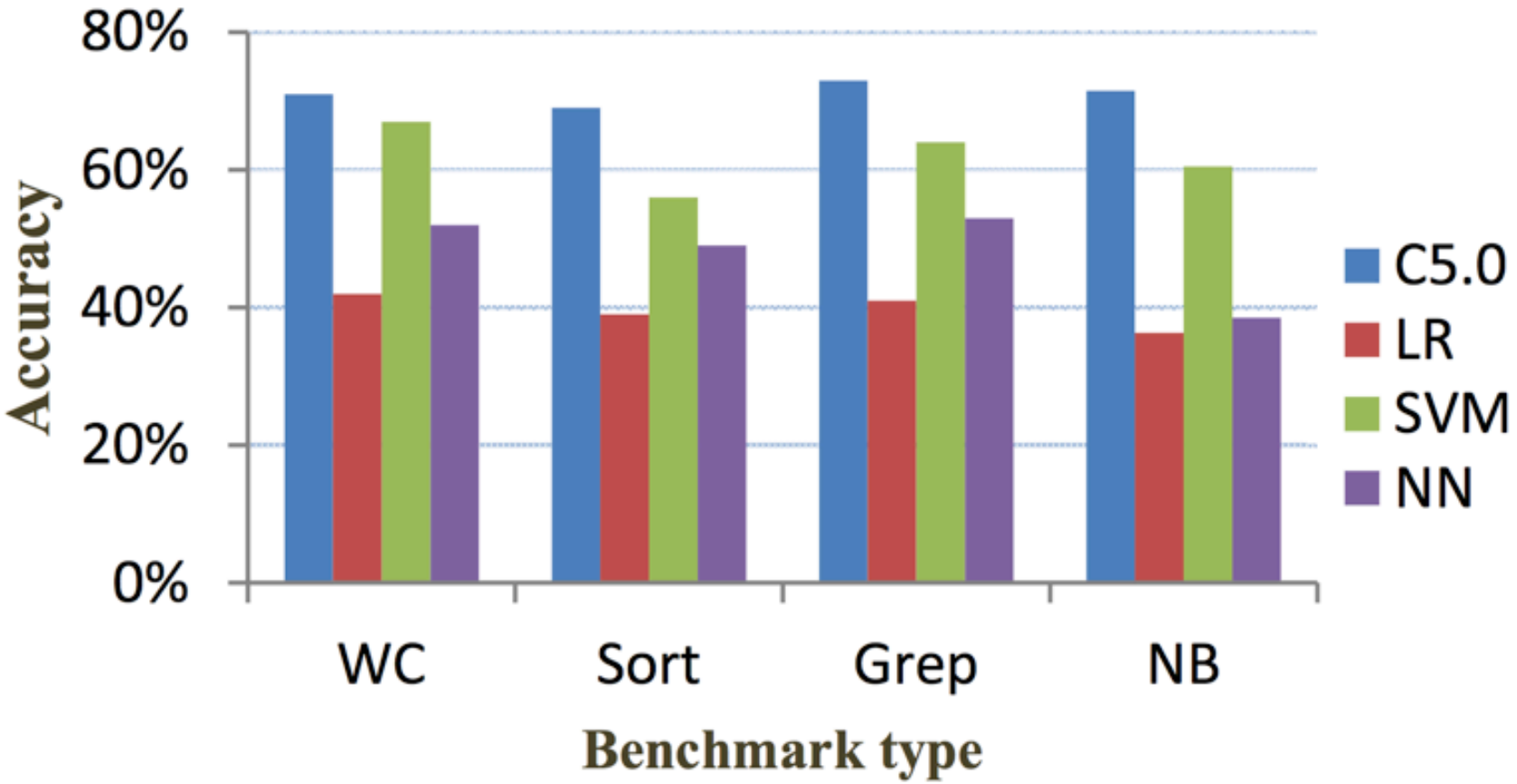


Figure 6. The accuracy of the models for Binary-classification

-- C5.0 and SVM have better accuracy on 4 kinds of benchmarks than other models

Experiments--Evaluation of accuracy and computational performance of the models



--C5.0 and SVM have better accuracy on 4 kinds of benchmarks than other models

Figure 7. The accuracy of the models for Multi-classification

Experiments--Evaluation of accuracy and computational performance of the models

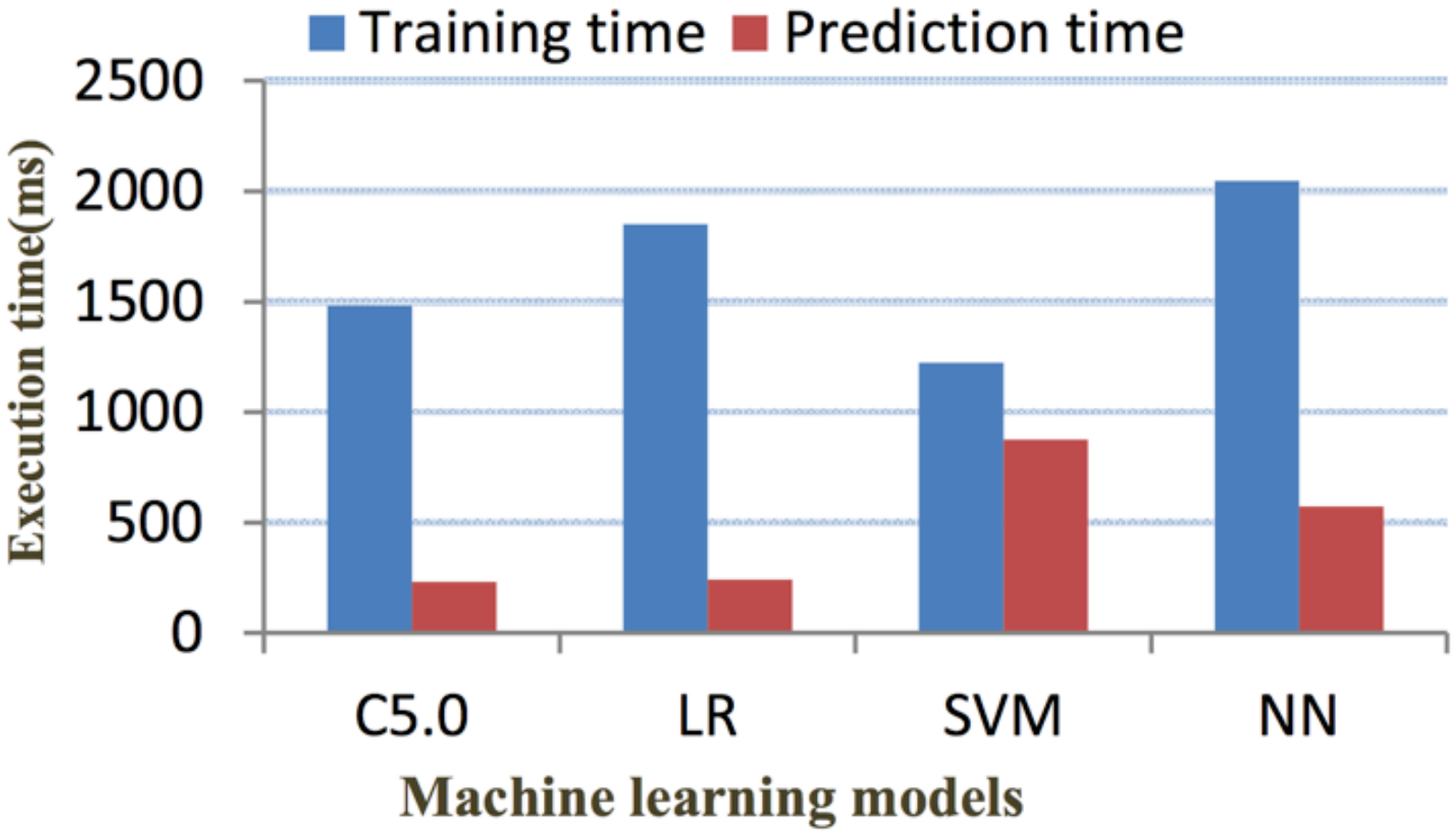


Figure 8. The computational performance of models

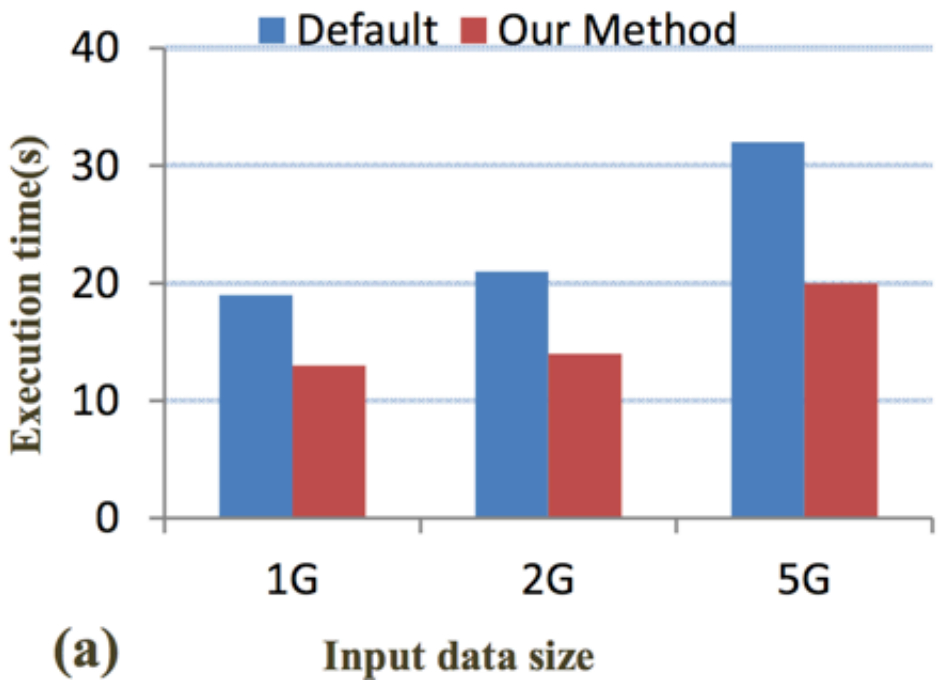
--training/ prediction time in the figure represents the time required by the model to train/prediction 1000 samples

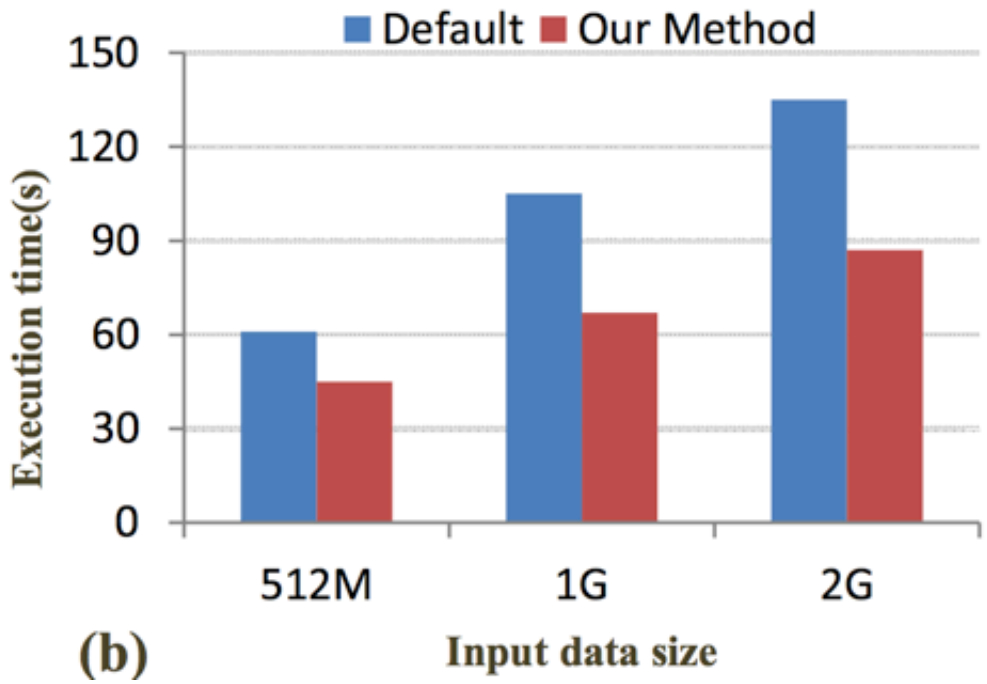
--SVM has the shortest training time, and C5.0 also has a shorter training time

--C5.0 and LR has shorter predict time than other models

--C5.0 has a greater advantage relative to other models in computational performance

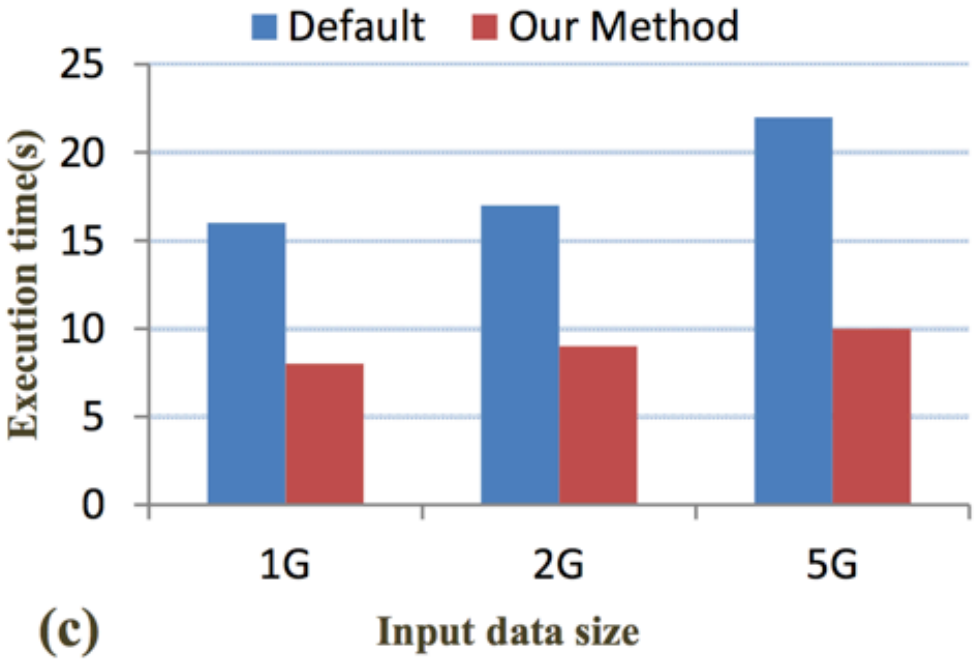
--Conclusion:C5.0 has a better performance both in accuracy and computational performance for binary-classification and multi-classification

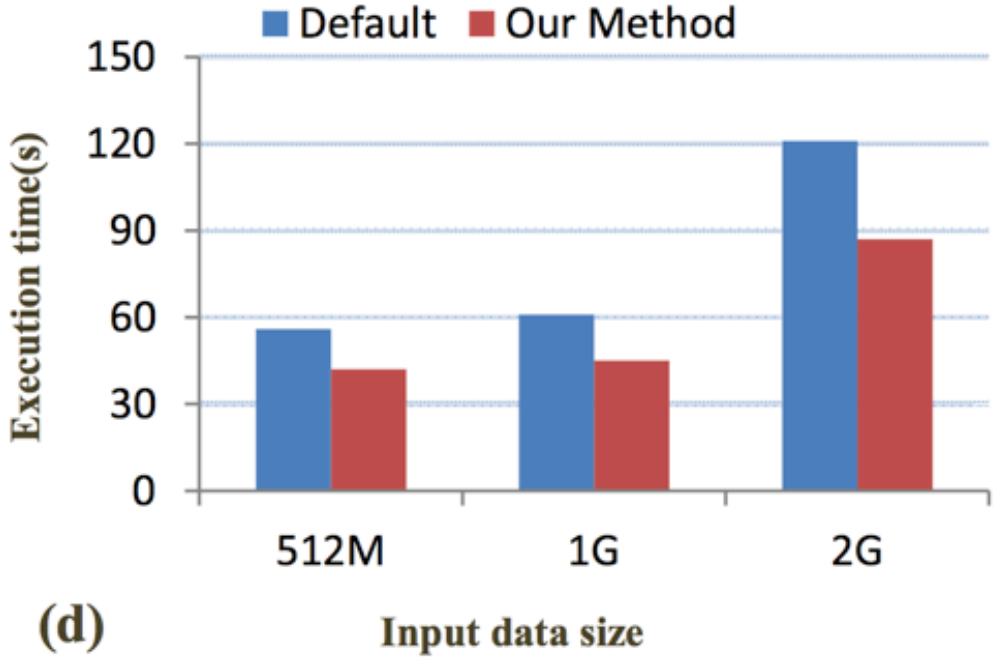




--(a) shows optimization effect of the WordCount benchmark
-- the proposed method has 31% performance gain when the input data is 1G and 38% performance gain when the input data is 5G. On average, the proposed method has a performance gain of 34%

--(b) shows optimization effect of Sort benchmark
-- method has maximum performance improvement of 36% and average performance improvement of 33%





--(c) shows optimization effect of Grep benchmark
-- method has maximum performance improvement of 55% and average performance improvement of 51%

--(d) shows optimization effect of NavieBayes benchmark
--method has maximum performance improvement of 28% and average performance improvement of 26%

--Conclusion: model has a significant performance improvement over the default parameters with the average improvement of nearly 36%. With the increase of input data, the effect of performance improvement is more obvious

Future Work

--They will conduct further research on parameter selection in order to select more suitable parameters

--They will also consider more aspects affecting on the performance, such as the resource competition between the jobs, theme fluctuation of network bandwidth and so on