Two Stages Query Processing Optimization Based on ELM in the Cloud

Linlin Ding[1], Yu Liu[1], Baoyan Song[1], and Junchang Xin[2]

[1] School of Information, Liaoning University, Shenyang Liaoning, China, 110036
{dinglinlin,bysong}@lnu.edu.cn
[2] College of Information Science & Engineering, Northeastern University, Shenyang Liaoning, China, 110819

Abstract. As one variant of MapReduce framework, ComMapReduce adds the lightweight communication mechanisms to improve the performance of query processing programs. Although the existing research work has already solved the problem of how to identify the communication strategy of ComMapReduce, there are still some drawbacks, such as relative simple model and too much user participation. Therefore, in this paper, we propose a two stages query processing optimization model based on ELM, named ELM to ELM (*E2E*) model. Then, we develop efficient sample training strategy, predicting and execution algorithm to construct the *E2E* model. Finally, extensive experiments are conducted to verify the effectiveness and efficiency of the *E2E* model.

Keywords: ELM, MapReduce, ComMapReduce, Prediction Model.

1 Introduction

Nowadays, MapReduce [1] has emerged as a famous programming framework for big data analysis. MapReduce and its variants are used for big data applications, such as Web indexing, data mining, machine learning, financial analysis [2–5]. As one of a successful improvements of MapReduce, ComMapRedcue [2, 3] adds simple lightweight communication mechanisms to generate the certain *shared information* and executes the query processing applications with large scale datasets in the Cloud. In ComMapReduce framework, three basic and two optimization communication strategies are proposed to illustrate how to communicate and obtain the *shared information* of different applications. During further analyzing ComMapReduce execution course and the abundant experiments, we find out that different communication strategies of ComMapReduce can substantially affect the performance of query processing applications.

The existing work [6] proposes a query processing model named *ELM_CMR* based on ELM [7] which has the classification performance at an excellent performance. *ELM_CMR* can identify the communication strategy of ComMapReduce according to the characteristics of query processing programs. However, *ELM_CMR* still has the following drawbacks. First, to the MapReduce or

© Springer International Publishing Switzerland 2015

J. Cao et al. (eds.), *Proceedings of ELM-2014 Volume 1*,
Proceedings in Adaptation, Learning and Optimization 3, DOI: 10.1007/978-3-319-14063-6_9

ComMapReduce, a program only consists of black_box Map and Reduce functions, without knowing the distributed details about the framework. But the configuration parameters of the framework can fully influence the performance of query processing. Finding the most suitable configuration parameters setting itself is difficult. Second, *ELM_CMR* only adopts simple training method to gain the classification model.

Therefore, in this paper, for solving the above drawbacks, we propose the two stages query processing optimization model based on ELM, named ELM to ELM (*E2E*) model. The first stage gains the *feature parameters* according to the program of users by using ELM algorithm. After that, according to the results of the first stage, the second stage can identify the final classification result by ELM too. Furthermore, an efficient training strategy, predicting and execution algorithm are presented. The contributions of this paper can be summarized as follows.

- We propose an efficient two stages query processing optimization model based on ELM, *E2E* model, which can realize the most optimal executions of query processing programs in MapReduce or ComMapReduce framework.
- We develop a sample training strategy, predicting and execution algorithm to construct the *E2E* model.
- The experimental studies using synthetic data show the effectiveness and efficiency of the *E2E* model.

The remainder of this paper is organized as follows. Section 2 briefly introduces the background, containing the ELM and *ELM_CMR*. Our *E2E* model is proposed in Section 3. The experimental results to show the performance of *E2E* model are reported in Section 5. Finally, we conclude this paper in Section 6.

2 Background

2.1 Review of ELM

Nowadays, Extreme Learning Machine (ELM) [7] and its variants [8–18] have the characteristics of excellent generalization performance, rapid training speed and little human intervene, which have attracted increasing attention from more and more researchers. ELM is originally designed for single hidden-layer feedforward neural networks (SLFNs [19]) and is then extended to the "generalized" SLFNs. ELM algorithm first randomly allocates the input weights and hidden layer biases and then analytically computes the output weights of SLFNs. Contrary to the other conventional learning algorithms, ELM reaches the optimal generalization performance with a very fast learning speed. ELM is less sensitive to the user defined parameters, so it can be deployed fast and convenient.

For N arbitrary distinct samples $(\mathbf{x}_j, \mathbf{t}_j)$, where $\mathbf{x}_j = [x_{j1}, x_{j2}, \ldots, x_{jn}]^T \in \mathbb{R}^n$ and $\mathbf{t}_j = [t_{j1}, t_{j2}, \ldots, t_{jm}]^T \in \mathbb{R}^m$, standard SLFNs with hidden nodes L and activation function $g(x)$ are mathematically modeled as

$$\sum_{i=1}^{L} \beta_i g_i(\mathbf{x}_j) = \sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{o}_j \qquad (j = 1, 2, \ldots, N) \tag{1}$$

where L is the number of hidden layer nodes, $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T$ is the weight vector between the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the ith hidden node and the output nodes, b_i is the threshold of the ith hidden node, and $\mathbf{o}_j = [o_{j1}, o_{j2}, \ldots, o_{jm}]^T$ is the jth output vector of the SLFNs.

The standard SLFNs can approximate these N samples with zero error. The error of ELM is $\sum_{j=1}^{L} ||\mathbf{o}_j - \mathbf{t}_j|| = 0$ and there exist β_i, $\mathbf{w}_i$ and b_i such that

$$\sum_{i=1}^{L} \beta_i g(\mathbf{w}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j \qquad (j = 1, 2, \ldots, N) \tag{2}$$

Equation (2) can be expressed compactly as follows:

$$\mathbf{H}\beta = \mathbf{T} \tag{3}$$

where $\mathbf{H}(\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_L, b_1, b_2, \ldots, b_L, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L)$

$$= \begin{bmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & g(\mathbf{w}_2 \cdot \mathbf{x}_1 + b_2) & \ldots & g(\mathbf{w}_L \cdot \mathbf{x}_1 + b_L) \\ g(\mathbf{w}_1 \cdot \mathbf{x}_2 + b_1) & g(\mathbf{w}_2 \cdot \mathbf{x}_2 + b_2) & \ldots & g(\mathbf{w}_L \cdot \mathbf{x}_2 + b_L) \\ \vdots & \vdots & \vdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & g(\mathbf{w}_2 \cdot \mathbf{x}_N + b_2) & \ldots & g(\mathbf{w}_L \cdot \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L} \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \beta_2^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} t_1^T \\ t_2^T \\ \vdots \\ t_N^T \end{bmatrix}_{N \times m} \tag{5}$$

$\mathbf{H}$ is set as the hidden layer output matrix of the neural network. The ith column of $\mathbf{H}$ is called the ith hidden node output with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. The smallest norm least-squares solution of the above multiple regression system is shown as follows:

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{6}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$. Then the output function of ELM can be modeled as follows.

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\beta = \mathbf{h}(\mathbf{x})\mathbf{H}^{\dagger}\mathbf{T} \tag{7}$$

2.2 *ELM_CMR* Model

ComMapReduce [2, 3] is an improved MapReduce framework with lightweight communication mechanisms. A new node, named the Coordinator node, is added to store and generate the certain *shared information* of different applications. In ComMapReduce, three basic communication strategies, LCS, ECS and HCS, and two optimization communication strategies, PreOS and PostOS are proposed

to identify how to receive and generate the *shared information*. In short, without affecting the existing characteristics of the original MapReduce framework, ComMapReduce is a successful parallel programming framework with global *shared information* to filter the unpromising data of query processing programs.

Figure 1 shows the architecture of *ELM_CMR* model. The four components of *ELM_CMR* are respectively the *Feature Selector*, the *ELM Classifier*, the *Query Optimizer* and the *Execution Fabric*.

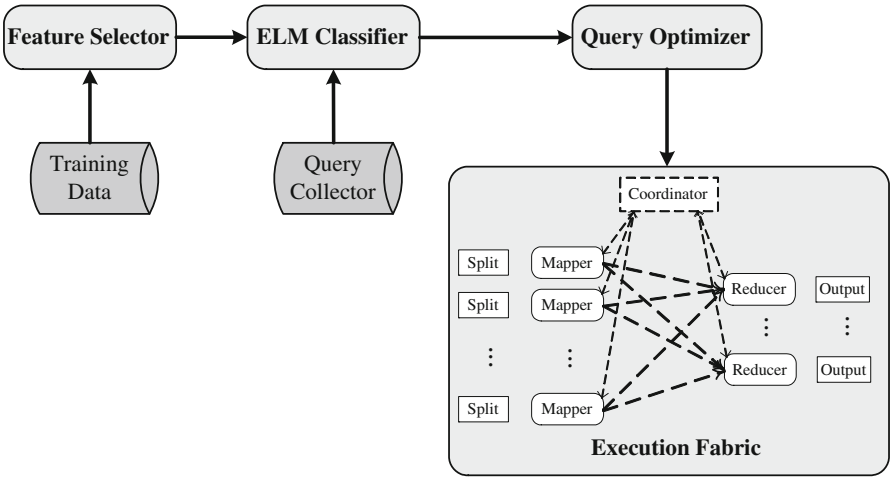

Fig. 1. Architecture of ELM_CMR Model

The *Feature Selector* mainly examines the training query processing programs and selects the configuration parameters that can wholly affect the query performance by the job profiles. Naturally, the parameters of program p can be divided into three types, parameters that predominantly affect Map task execution; parameters that predominantly affect Reduce task execution and the cluster parameters. Then, in each cluster, we adopt the minimum-redundancy-maximum-relevance (mRMR) [20] feature selection to find the optimal parameters sharply affecting the performance. And then, we generate the globally optimal configuration parameter settings by combining the results of each subspace. Therefore, the near-optimal configuration parameter setting can be generated.

After selecting the features of training data, the *Feature Selector* sends the extracted training data to the *ELM Classifier*. It uses the training data to construct the ELM model by the traditional ELM algorithm. After that, when there are one or multiple queries to be processed, the *ELM Classifier* can rapidly obtain the classification results of the queries, and then sends them to the *Query Optimizer*.

The *Query Optimizer* applies the classification results of the *ELM Classifier* and combines the implementation patterns to choose an optimized *execution*

order. After gaining the *execution order*, the query is sent to the *Execution Fabric*.

The *Execution Fabric* implements the program in ComMapReduce framework. When there is one query to be processed, the *Execution Fabric* implements the query according to the classification result of the *Query Optimizer* in *ELM_CMR*. When there are multiple queries to be processed, the multiple queries can be classified by *ELM Classifier* and gain the best communication strategy of each program. Then, a Task Scheduler Simulator is used to simulate the execution time of queries. According to the execution time and the classification results of the queries, the *Query Optimizer* designs an *execution order* following the common principle of Shortest Job First (*SJF*) to implement multiple queries.

3 *E2E* Model

3.1 Overview of *E2E* Model

Our *E2E* model can identify the optimal communication strategies of query processing programs in MapReduce or ComMapReduce, which contains three main phases, respectively the *training phase*, the *prediction phase* and the *execution phase*. Figure 2 shows the whole workflow of query processing in the *E2E* model. The main workflow is as follows.

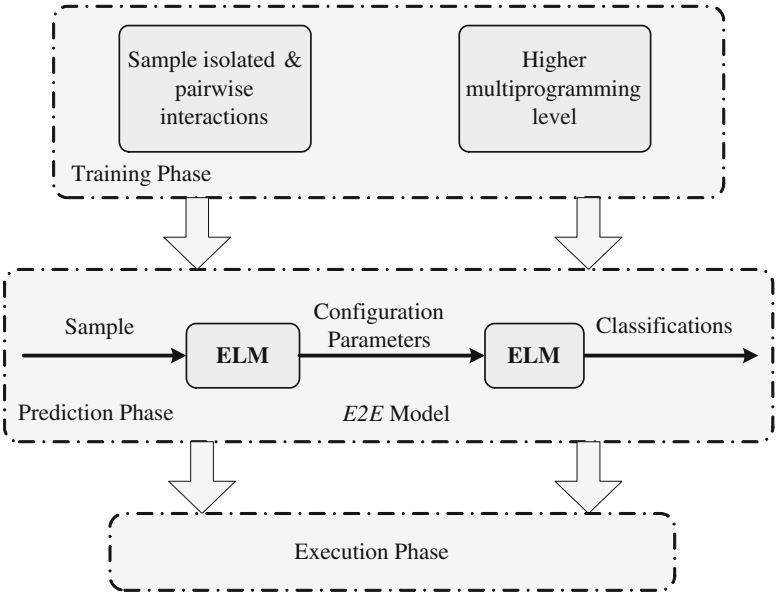

Fig. 2. Workflow of *E2E* Model

First, the *training phase* is responsible for extracting the training query processing programs that have a large affect on the *E2E* model. In the *training*

phase, we use sample-based training strategies to run our workload, in isolation, pairwise and at several higher multi-programming levels. After gaining the training samples, they can be used to generate the *E2E* model in the *prediction phase*.

Second, in the *prediction phase*, by using the training samples from the *training phase*, the two stages *E2E* model can be generated based on the traditional ELM algorithm. According to the user's programs, the first stage can obtain the most optimal *feature parameters* of the programs by ELM. And then, using the *feature parameters* of the first stage, the second stage can identify the classification results of the query processing programs by ELM.

Third, after gaining the *E2E* model, for the query processing program submitted to MapReduce or ComMapReduce, we can predict the optimal communication strategy in the *execution phase*.

3.2 Training *E2E* Model

To obtain the prediction model more accurately, we need to train the *E2E* model. Different from the simple training course of *ELM_CMR* model, the *training phase* of our *E2E* model consists of running the queries in isolation, pairwise as well as at several higher multi-programming levels.

The *E2E* model realizes the *training phase* by sampling approach, containing isolation, pairwise and higher degree of concurrency, which allows us to approximate the running of queries in MapReduce or ComMapReduce. The course of the *training phase* is displayed in Figure 2. First, we sample the workload in isolation to gain how each query behaves, which can be seen as the baseline of training. Second, according to the query types in our experiments, we build a matrix of interaction by running all unique pairwise combinations. Pairwise sample can help us to simply estimate the degree of concurrency. Here, Latin hypercube sampling approach (LHS) can uniformly distribute our samples throughout our prediction space, which can be realized by creating a hypercube with the same dimension as the multi-programming level. We adopt LHS to sample at pairwise or several higher multi-programming levels, and then select the samples that every value on every plane gets inter selected exactly one. A simple example of two dimensional LHS is shown in Table 1.

Table 1. An example of LHS

Query	1	2	3	4
1		√		
2			√	
3	√			
4				√

3.3 Predicting *E2E* Model

Then, we introduce the details of the *prediction phase*. A MapReduce job j can be expressed by a MapReduce program p running on input data d and cluster r, which can be expressed as $j=< p, d, r, c >$ in short. We call the d, r, c the *feature parameters* of p. The users only submit their jobs to MapReduce or ComMapReduce without knowing the internal configuration details of the system.

However, a number of choices can be made in order to fully specify how the job should be executed. These choices, represented by c in $< q, d, r, c >$, stand for a high dimensional space of configuration parameter settings, such as the number of Map and Reduce task, the block size, the amount of memory, and so on. The performance changes a lot in different configuration parameters. For any parameter, its value is not specified explicitly during job submission, either the default values shipped with the system or specified by the system administrator. However, the normal users don't understand the running details of MapReduce. That is to say, finding good configuration settings for MapReduce job is time consuming and requires extensive knowledge of system internals. Because the users have little information of the parallelization details of MapReduce, it is necessary to gain the suitable configuration parameters. The first stage of our *E2E* model is to generate the model for identifying the suitable configuration parameters of query processing programs by ELM. The main goal of the first stage is to construct a black_box *feature parameters* of queries, containing $< d, r, c >$. The configuration parameters wholly affecting the performance adopted by *E2E* are the same as our *ELM_CMR* approach. The corresponding information of the job can be obtained from sampling a few tasks of the job. After the first stage, the *feature parameters* setting can be obtained by ELM algorithm. After gaining the configuration parameters, the second stage is to use them to predict the communication strategy of ComMapReduce or MapReduce by ELM algorithm too, and then generates the classification results.

4 Execution of *E2E* Model

When there is one new query being submitted, the *E2E* model generates its classification result as soon as possible. According to the characteristics of the coming query, the first stage of *E2E* model can generate the *feature parameters* of this query based on ELM. After that, the user can make a decision that whether adopting ComMapReduce or adopting which communication strategy by the second stage of *E2E* model, and then implement the query processing application.

The course of execution is shown in Algorithm 1. First, the most optimal *feature parameters* of query processing job j are extracted in the first stage of *E2E* model (Line 1). Second, after obtaining the *feature parameters* of job j, the *E2E* generates the classification result of j (Line 2). Third, according to the classification of j, the *E2E* ensures how to implement the query and

Algorithm 1. Execution of *E2E* Model

1: Generate the *feature parameters* of j by the first stage of *E2E* model;
2: Generate the execution plan of j by the second stage of *E2E* model;
3: Execute j with its communication strategy;

sends it to MapReduce or ComMapReduce framework. The framework uses the optimization result to execute the query program (Line 3).

For example, for a submitted skyline query, the first stage of *E2E* can obtain the most optimal *feature parameters* of this skyline query. After abstracting its *feature parameters*, the *E2E* can generate its classification and then identifies the communication strategy of this skyline query, such as PostOS. After that, the skyline query will be implemented in ComMapReduce with PostOS.

5 Performance Evaluation

5.1 Experimental Setup

The experimental setup is the same as *ELM_CMR* model as follows. The experimental setup is a Hadoop cluster running on 9 nodes in a high speed Gigabit network, with one node as the Master node and the Coordinator node, the others as the Slave nodes. Each node has an Intel Quad Core 2.66GHZ CPU, 4GB memory and CentOS Linux 5.6. We use Hadoop 0.20.2 and compile the source codes under JDK 1.6. The ELM algorithm is implemented in MATLABR2009a.

The data in our experiments are synthetic data. The classification results of *E2E* model contains 7 types, respectively ECS, HCS-0.5, HCS-1, HCS-2, PreOS, PostOS and MapReduce (MR). HCS-0.5 means the preassigned time interval of HCS is 0.5s. We evaluate the performance of *E2E* model by comparing with *ELM_CMR*. Four typical query processing applications are adopted to evaluate the peformance, respectively top-k, kNN, skyline and join.

5.2 Experimental Results

Figure 3 shows the performance of top-k queries (k=1000), with the data size is 2G, 4G, 6G and 8G in uniform distribution. We can see that the performance of top-k queries in the classification results of *E2E* model is better than *ELM_CMR*. The reason is that *E2E* model can effectively evaluate the optimal configuration parameters of the queries than *ELM_CMR* model. When k is much smaller than the original data, the global *shared information* of ComMapReduce can reach the most optimal one quickly, so the Mappers can retrieve the *shared information* in the initial phase to filter the unpromising data.

Figure 4 shows the performance of kNN queries with different data size of uniform distribution. The performance of *E2E* is also optimal to *ELM_CMR* model, where the reason is that *E2E* model can gain the suitable classification results.

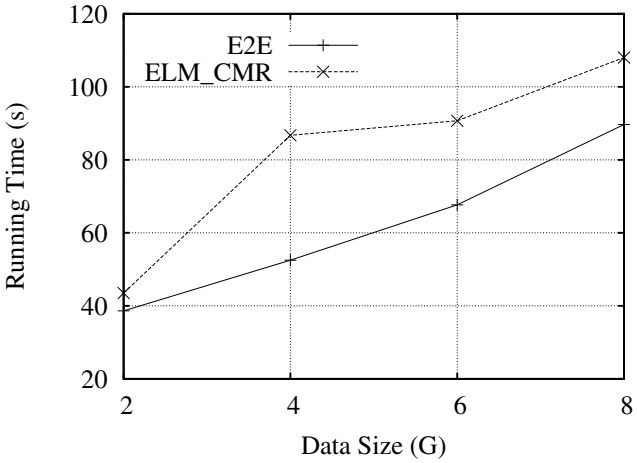

Fig. 3. Performance of top-k Query

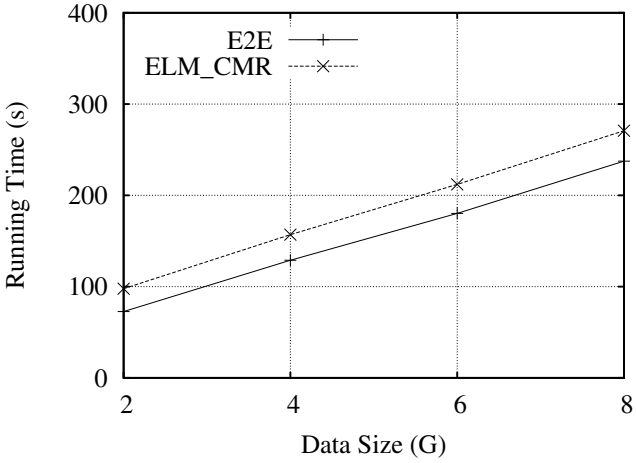

Fig. 4. Performance of kNN Query

Figure 5 shows the performance of skyline queries in anti-correlated distribution with different data size. We can see that the performance of different execution plans is not obviously different, but PostOS is a little better. The reason is that the original data are skewed to the final results in anti-correlated distribution. The percentage of filtering is low, so the performance difference is not obvious. In this situation, although *E2E* and *ELM_CMR* can obtain the classification, it can also choose the other communication strategies.

Figure 6 shows the performance of join queries in different data size of small-big tables, with the same data size of the small table 2G, and the different data

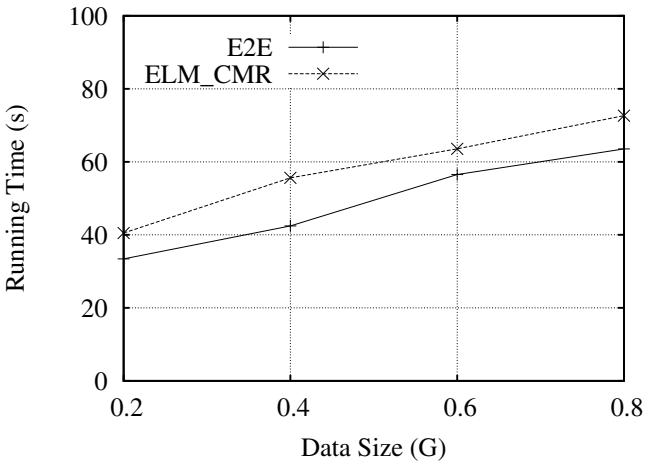

Fig. 5. Performance of Skyline Query

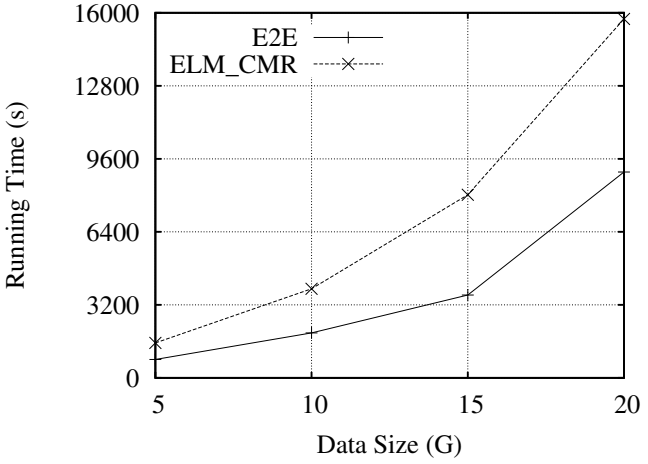

Fig. 6. Performance of join Query

sizes of big table are shown in Figure 6. The performance of *E2E* is much better than *ELM_CMR*. In ComMapReduce, the join attributes of the small table can be set as the *shared information* to filter the unpromising intermediate results.

6 Conclusions

In this paper, we propose an efficient query processing optimization model based on ELM, *E2E* model. Our *E2E* can effectively analyze the query processing

applications, and then generates the most optimized executions of query processing applications. After analyzing the problems of the former *ELM_CMR* model, we use two stages model to classify the query processing applications in ComMapReduce framework. Then, we propose an efficient training approach to train our model. We also give the predicting and query executions. The experiments demonstrate that the *E2E* model is efficient and the query processing applications can reach an optimal performance.

Acknowledgement. This research is supported by National Natural Science Foundation of China (NO. 60873068, 61003003), Talent Projects of the Educational Department of Liaoning Province (NO. LR201017), the National Natural Science Foundation of China under Grant Nos. 61100022.

References

1. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM 51(1), 107–113 (2008)
2. Ding, L., Xin, J., Wang, G., Huang, S.: ComMapReduce: An improvement of mapReduce with lightweight communication mechanisms. In: Lee, S.-g., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part II. LNCS, vol. 7239, pp. 150–168. Springer, Heidelberg (2012)
3. Ding, L., Wang, G., Xin, J., Wang, X., Huang, S., Zhang, R.: Commapreduce: an improvement of mapreduce with lightweight communication mechanisms. Data Knowledge Engineering 88, 224–247 (2013)
4. Deng, D., Li, G., Hao, S., Wang, J., Feng, J.: Massjoin: A mapreduce-based method for scalable string similarity joins. In: ICDE, pp. 340–351. IEEE (2014)
5. Qin, L., Yu, J.X., Chang, L., Cheng, H., Zhang, C., Lin, X.: Scalable big graph processing in mapreduce. In: SIGMOD Conference, pp. 827–838. ACM (2014)
6. Ding, L., Xin, J., Wang, G.: An efficient query processing optimization based on elm in the cloud. Neural Computing and Applications (2014)
7. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of the 2004 IEEE International Joint Conference on Neural Networks, vol. 2, pp. 985–990. IEEE (2004)
8. Lendasse, A., He, Q., Miche, Y., Huang, G.-B.: Advances in extreme learning machines (elm2012). Neurocomputing 128, 1–3 (2014)
9. Zong, W., Huang, G.-B.: Learning to rank with extreme learning machine. Neural Processing Letters 39(2), 155–166 (2014)
10. Basu, A., Shuo, S., Zhou, H., Lim, M.-H., Huang, G.-B.: Silicon spiking neurons for hardware implementation of extreme learning machines. Neurocomputing 102, 125–134 (2013)
11. Xi-Zhao, W., Qing-Yan, S., Qing, M., Jun-Hai, Z.: Architecture selection for networks trained with extreme learning machine using localized generalization error model. Neurocomputing 102, 3–9 (2013)
12. Zong, W., Huang, G.-B., Chen, Y.: Weighted extreme learning machine for imbalance learning. Neurocomputing 101, 229–242 (2013)
13. He, Q., Shang, T., Zhuang, F., Shi, Z.: Parallel extreme learning machine for regression based on mapreduce. Neurocomputing 102, 52–58 (2013)

14. Zhang, R., Lan, Y., Huang, G.-B., Xu, Z.-B., Soh, Y.C.: Dynamic extreme learning machine and its approximation capability. IEEE T. Cybernetics 43(6), 2054–2065 (2013)
15. Zhai, J.-H., Xu, H.-Y., Wang, X.-Z.: Dynamic ensemble extreme learning machine based on sample entropy. Soft Computing 16(9), 1493–1502 (2012)
16. Sun, Y., Yuan, Y., Wang, G.: An os-elm based distributed ensemble classification framework in p2p networks. Neurocomputing 74(16), 2438–2443 (2011)
17. Zhao, X.-g., Wang, G., Bi, X., Gong, P., Zhao, Y.: Xml document classification based on elm. Neurocomputing 74(16), 2444–2451 (2011)
18. Jun, W., Shitong, W., Chung, F.-l.: Positive and negative fuzzy rule system, extreme learning machine and image classification. International Journal of Machine Learning and Cybernetics 2(4), 261–271 (2011)
19. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing 70(1), 489–501 (2006)
20. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(8), 1226–1238 (2005)