Implementation of Artificial Neural Networks in MapReduce Optimization

Changlong Li[1], Xuehai Zhou[1], Kun Lu[1], Chao Wang[1], Dong Dai[2]
[1] *University of Science and Technology of China,*
[2] *Texas Tech University, USA*
Email: {liclong, xhzhou, local, saint}@mail.ustc.edu.cn, dong.dai@ttu.edu

***Abstract*—The ability to handle large datasets has become a critical consideration for the success and ability of industrial organizations such as Microsoft, Amazon, Yahoo! and Facebook. As an important cloud computing framework for data processing, MapReduce is widely used by these organizations. However, its performance has been seriously limited by its stiff configuration strategy. In practice, even for a single simple job in a MapReduce framework, a large number of tuning parameters have to be set by end users, who often run into performance problems since they do not know how to configure them. Besides, once set, most parameters will never be changed again. This may easily lead to performance loss due to some misconfigurations. In this paper, we present a soft computing technique: Artificial Neural Network(ANN) to achieve the automatic configuration of parameters for MapReduce. Given a cluster and MapReduce job, frameworks can adapt the hardware and software configurations to the system dynamically and drive the system to an optimal configuration in acceptable time with the help of ANN. Experimental results show that ANN has a great contribution to optimize system performance and let the system at the speedup of 9x.**

***Keywords*-Neural Network; MapReduce; Automatic Configuration; Optimization; Big Data;**

I. Introduction

Since its inception, MapReduce [1] has frequently been associated with cloud computing and large-scale datasets. Widely deployment and application at industry organizations have thrust this programming framework to the forefront of cloud computing and data processing application domain. There are growing interests in deploying such a framework in the Cloud to harness the unlimited availability of virtualized resources of cloud computing. For example, Amazon's Elastic MapReduce provides data processing services by using Hadoop on top of their compute cloud EC2. However, the performance of MapReduce need to be further improved: a MapReduce program can run 2-50x slower than a similar relational query run on an RDBMS with identical hardware [2], Anderson also showed that Hadoop which is an implementation of a MapReduce framework performed bulk data processing at a rate of less than 5 megabytes per node per second [6]. Current technologies mainly achieve MapReduce optimization through the way of data locality, cluster heterogeneity and scheduling strategy. However, pushing the responsibility for performance optimization into underlying code will make the code complex and hard to maintain. Besides, these technologies are always limited by their scalability and operability.

Existing programming environments for running MapReduce jobs in a cloud platform aim to remove the burden of hardware and software setup from end users. However, they expect end users to provide appropriate parameters for running a job. In the absence of automatic configuration scheme, users are forced to make job provisioning decisions manually using best practices. As a result, customers may suffer from a lack of performance guarantee. The difficulty of setting up those parameters contains two folders. First, empirical evidence suggests that the performance of submitted job is some complex function of the configuration parameter settings. Second, the features of cluster hardware (memory capacity, network bandwidth) as well as the characteristics of MapReduce program (data-intensive, compute-intensive) also have significant impact on the performance.

As the performance of MapReduce is seriously limited by its configuration, therefore many techniques that evaluate and configure system parameters for traditional cluster have been proposed [7]. However, the effectiveness of these approaches are often inter-dependent, some of them focus on a single element only and hence are not able to address the complex, high-dimensional configuration problem. Others such as Minerva [3] are extremely complex for non-expert users, requiring expertise with advanced tools and a large number of experiments. Kambatla et al. [5] found this problem and proposed to select optimal configuration parameters using a given set of resources, but there is no guidance on deciding the appropriate number and type of resources to be allocated. So it is critical and necessary to propose an auto-configuration scheme that integrates various aspects of factors as well as parameters to provide optimal configuration in acceptable time for MapReduce.

There are some complex functions between configuration parameters, hardware features, program characteristics, and MapReduce performance. In this paper, we propose ANN, an artificial neural network, to optimize system performance through learning the internal relationship between impact factors. Given an application to run on a given platform, ANN automatically searches for optimized system configurations from candidate settings. We analyze the relationship between influential configurations and system performance with the implementation of the network and then dynam-

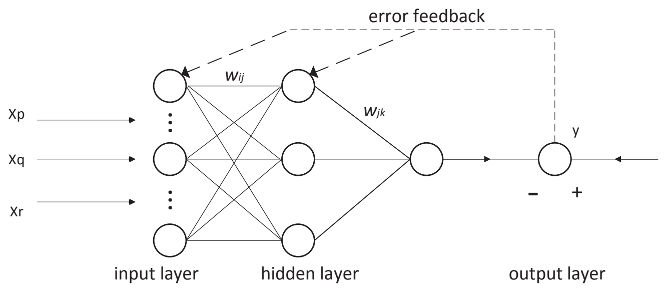


Figure 1. ANN: x_p represents the setting of configuration parameters, x_q represents program characteristics and x_r represents hardware features.

ically adjust the parameters to achieve performance optimization. After trained on the target platform, ANN learns the internal relationship and makes predictions accordingly. Based on the prediction, the network could recommend an optimized configuration and adjust them dynamically. New collected data will be learned by ANN to improve the accuracy of recommendation. Experimental results show that MapReduce framework is able to adapt the configurations to the system dynamically and drive the system to an optimal configuration in acceptable time.

This paper is organized as follows. In Section II, we describe the design of ANN. Section III shows evaluation based on real-world deployment. We introduce the related work in Section IV, and finally, Section V presents our conclusion and the future work.

II. Design

The empirical evidence from Section III suggests that the performance of a MapReduce job J is some complex function of the job configuration parameter settings. In additional, the features of the hardware as well as program characteristics will impact its performance. There exists some function F_J such that:

$$y = F_J(\vec{p} \in \vec{P}, \vec{q} \in \vec{Q}, \vec{r} \in \vec{R}) \quad (1)$$

Here, y represents a metric of system performance(e.g., CPU usage), $\vec{P} = \{x_1, ..., x_p\}$ represents the setting of configuration parameters which have significant impact on performance (parameters have little impact on performance are ignored), $\vec{Q} = \{x_{p+1}, ..., x_{p+q}\}$ represents the characteristics of program and $\vec{R} = \{x_{p+q+1}, ..., x_{p+q+r}\}$ represents the features of hardware. Since these parameters display strong performance interactions with other parameters, the function relationship is complex and unpredictable. Thus we design an artificial neural network to learn this function.

ANN, the artificial neural network, consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. Each neuron is shown as a circle in the diagram, and the lines connecting them are known as weights. As illustrated in Figure 1, the $\vec{P}$, $\vec{Q}$ and $\vec{R}$ is the input and performance metric y is the output. Input variables are applied to the input units at the left of the diagram. The input layer send data via synapses to the hidden layer, and then via more synapses to the output layer. If the output $\hat{y}$ were inconsistent with y, the target value of performance metric (the actual output in samples), error messages will be back propagated and values of weights will be adaptively modified during the process of network training. Hence, the network provides a direct mapping from system parameters onto weight values associated with the best fit function. Although the training of neural networks is computationally intensive, the trained networks can process new data very rapidly. The more comprehensive the training set, the more representative the trained neural network becomes.

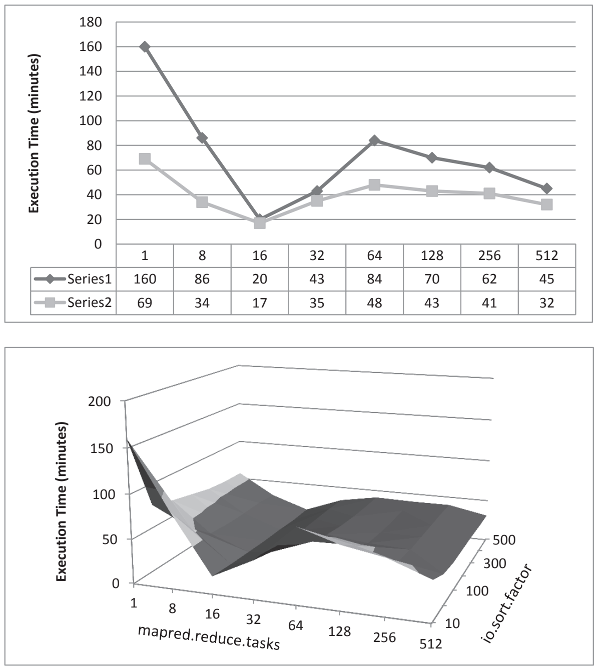
Given a cluster and MapReduce job, we can think of $\vec{P}$ which represents the setting of configuration parameters as the only factors that affect system performance. We provide a table to log all these parameters and their possible values. Once training complete and the relationship learned, our recommendation system will traverse the table and let each parametric combination as input for prediction, then recommends an optimized configuration and adjust them dynamically. Since the table size is not very large, we can quickly lock the optimal solution and adjust them dynamically. On the other hand, new configurations and their corresponding performance will be collected as ANN's feedback information to help improving its accuracy and efficiency through self-learning.

The problem of optimizing the parameters of a given functional form to fit experimental data points is frequently encountered in data analysis. In this Section we have shown that the artificial neural network can provide a direct mapping from the measured data onto the parameter values associated with the best fit function.

III. Evaluation

A. Experimental Setup

In this experiment, we choose Hadoop as our platform: as an open source implementation of a MapReduce framework, Hadoop is widely used in production deployments for applications such as log file analysis, scientific simulation, Web indexing, report generation and genome sequencing [8]. All our experiments were performed using local cluster running on 9 nodes, with 1 master and 8 worker nodes: all machines have Xeon dual-core 2.53GHz processor with 6GB memory. We choose the IOR [4] synthetic benchmark as it is generic, open-source, and highly configurable. ANN carries out the initial training by running the benchmarks on Hadoop. For each training run, it collects the performance metric with the candidate configurations. We choose dstat tool on each node to measure CPU usage (in terms of User, system, Idle and Wait percentages), disk usage (number of blocks in and out) and network usage (bytes/second into and out of network card) every second.



	1	8	16	32	64	128	256	512
Series1	160	86	20	43	84	70	62	45
Series2	69	34	17	35	48	43	41	32

Figure 2. Execution time of TeraSort benchmark with 100 GB datasets. There are some complex relationship between performance and parameters *mapred.reduce.tasks* + *io.sort.factor*. Series1 means *io.sort.factor = 10*, and Series2 means *io.sort.factor = 100*.

B. Parameter Analyze

There are 200+ parameters are specified to control the behavior of MapReduce programs in Hadoop-2.2.0 and more than 30 of these parameters have significant impact on job performance. Here we divide the parameters into three categories. (i) Most parameters are configured for normal operation and have no impact on performance. For example, *dfs.datanode.dns.nameserver* determines IP address and *dfs.datanode.address* configures namenode's port information. (ii) Some parameters can be set when job is submitted. To run the program in Hadoop, the system will create a job configuration object based on some given parameters from users. This job configuration object usually is highly relevant with the performance. (iii) Apart from the job configuration parameters whose values are specified explicitly by users, there are a large number of parameters whose values are specified implicitly by the system.

All these parameters except the first type control various aspects of job behavior during execution such as memory allocation, I/O optimization and network bandwidth usage. For example, *mapred.reduce.tasks* determines the number of reducer tasks and *dfs.block.size* denotes HDFS block size. Although different parameters affect performance in different ways, they are not independent. On the contrary, they have strong performance interactions with one or more other parameters. Here we present some empirical evidence to demonstrate differences in job running times between good and bad parameter settings in Hadoop. Figure 2 shows the execution time of TeraSort on cluster for 100GB datasets. The two parameters, *mapred.reduce.tasks* and *io.sort.factor*, are varied in these figures while all other job configuration parameters are kept constant. In Figure 2-a, the function relationship between execution time and *mapred.reduce.tasks* is changed when the value of *io.sort.factor* is different. From the comparison we know that: (i) Configuration parameters have significant impact on system performance. (ii) A number of instances of inter-parameter interactions were seen in our experiments. (iii) There are some complex and high-dimensional function relationship between parameters and system performance.

C. Performance Comparison

Statistics show that there are more than 200 parameters are specified to control the behavior of submitted job in Hadoop-2.2.0, and more than 30 of these parameters have significant impact on job performance. Figure 3 (a), (b), (c) and (d) compare the impact of using the default Hadoop configuration with ANN's auto-tuned configuration on the job execution time. It compares the Hadoop configurable parameter values due to ANN's auto-configuration with the default Hadoop values. The network provides different configuration scenarios for running TeraSort, WordCount and PiEstimator benchmark with input data of 1GB, 10GB, 50GB and 100GB. As shown in Figure 3, with the help of ANN, Hadoop is able to adapt the hardware and software configurations based on the system dynamically and drive the system to an optimal configuration in an acceptable time. We know from the comparison that the running time of benchmarks are about 9 times faster than default configuration. Moreover, the effect of ANN is more obvious when datasets scale increased. In particular, parameters *io.sort.factor* and *mapred.reduce.tasks* have significant different values. Hadoop TeraSort benchmark sets the default value of *mapred.reduce.tasks* to about 0.9 times the total number of reduce slots in the cluster.

D. Fault Tolerance

Other than performance, we also measured the fault tolerance of the network. As we know, the success of neural network's machine learning depends to a great extent on the sample. So if the sample size is insufficient or the quality is not high, ANN cannot learn the relationship accurately. Here we consider an extreme case, since the hardware environment changed (or any other reasons), the sample will be out of date as a result. In this case, the configuration scenarios putted by ANN may be even worse than the default because the real relationship may have changed. Our feedback mechanism solves this challenge. In the process of self-learning, the proportion of sample is gradually replaced

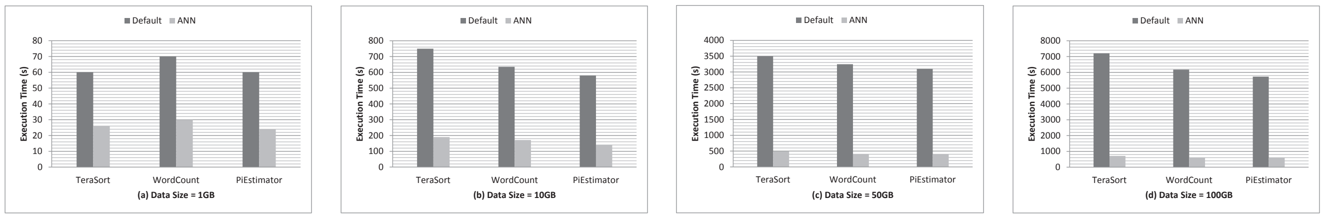


Figure 3. Execution time of TeraSort, WordCount and PiEstimator benchmark with 1GB, 10GB, 50GB and 100GB datasets. The performance of MapReduce is greatly improved with the help of ANN. Besides, the effect of optimization is more obvious when datasets scale increased.

by subsequent data. In our evaluation, the error or deficiency of sample can be covered by the measurement data.

IV. Related Work

Recently MapReduce infrastructures and its open source implementation Hadoop have gained much attention, and different approaches have been developed to automatically and efficiently optimize configurations. W. Zheng [10] presented an approach to achieve automated configuration. Although this heuristics approach efficient in time consuming, sometimes it cannot reach global optimality. The method of eliminate database tuning knobs through code rewrites have been used in Hadoop, but setting the parameter through code modifying can make the code complex and hard to maintain. There are also many other schemes proposed to achieve automatic configuration, Gideon et al. study the impact of different data sharing options for scientific workflows on Amazon EC2, Elastisizer selects the proper cluster size and instance types for MapReduce workloads running in the cloud. Most of these existing efforts assume certain knowledge on the application/middleware internals, but they did not solve the essential problem: finding the internal relationship. We solve the problems above by ANN. It achieves system-wide parameters configuration automatically through training and self-learning. Also, ANN offers the expandability and flexibility that allow it to work across cloud platforms and across hardware updates.

V. Conclusion and Future Work

Optimization through the way of automatic configuration is becoming an increasingly important concern for MapReduce frameworks. In this paper, the technology of Artificial Neural Network is implemented to learn the internal and high-dimensional function relationship. It accurately fits the relationship between performance, parameters and other factors through training and further learning. We also present a recommend scheme to MapReduce with the help of ANN, allowing for parameters to be automatically adjusted between tasks without the need for the framework to be shutdown or restarted. The experiment results demonstrate the performance of our approach.

In the near future, we plan to keep on investigating the factors that affect system performance, and consider to improve the speed and accuracy of the neural network.

Acknowledgment

We deeply appreciate the anonymous reviewers for their insightful comments and suggestions. This work was supported by the National Science Foundation of China under grants No. 61379040, No. 61272131 and No. 61202053, Jiangsu Provincial Natural Science Foundation grant No. SBK201240198.

References

[1] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proc. of ISDI*, 2004.

[2] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. Dewitt, S. Madden, and M. Stonebraker. A comparison of approaches to large-scale data analysis. In *SIGMOD Conference*, pages 165-178, 2009.

[3] G. Alvarez, E. Borowsky, and S. e. a. Go. Minerva: An Automated Resource Provisioning Tool for Large-scale Storage Systems. *ACM Transactions on Computer Systems (TOCS)*, 19(4):483-518, 2001.

[4] H. Shan, K. Antypas, and J. Shalf. Characterizing and Predicting the I/O Performance of HPC Applications Using a Parameterized Synthetic Benchmark. In *SC. IEEE*, 2008.

[5] K. Kambatla, A. Pathak, and H. Pucha. Towards optimizing hadoop proisioning in the cloud. In *HotCloud Workshop in conjunction with USENIX Annual Technical Conference*, 2009.

[6] E. Anderson and J. Tucek. Efficiency matters. In *SIGOPS Workshop*, pages 40-45, 2010.

[7] A. Verma, L. Cherkasova, and R. Campbell. ARIA: automatic resource inference and allocation for MapReduce environments. In *Proc. IEEE/ACM Int'l Conference on Autonomic Computing(ICAC)*, 2011.

[8] Chao Wang, Xi Li, Xuehai Zhou, Jim Martin, Ray C. C. Cheung: Genome sequencing using mapreduce on FPGA with multiple hardware accelerators, 2013.

[9] Z. Liu, H. Li, and G. Miao. MapReduce-based backpropagation neural network over large scale mobile data, in *ICNC10*, 2010.

[10] W. Zheng, R. Bianchini, and T. D. Nguyen. Automatic configuration of internet services. In *Proc. of ACM European Conference on Computer Sytems (EuroSys)*, 2007.