JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 24 No. 11, pp. 51-59, November 2019
https://doi.org/10.9708/jksci.2019.24.11.051

Web access prediction based on parallel deep learning

Gantur Togtokh*, Kyung-Chang Kim**

* Assistant Professor, Dept. of Information Technology, Ulaan Baatar University, Mongolia
**Professor, Dept. of Computer Engineering, Hongik University, Seoul, Korea

[Abstract]

Due to the exponential growth of access information on the web, the need for predicting web users' next access has increased. Various models such as markov models, deep neural networks, support vector machines, and fuzzy inference models were proposed to handle web access prediction. For deep learning based on neural network models, training time on large-scale web usage data is very huge. To address this problem, deep neural network models are trained on cluster of computers in parallel. In this paper, we investigated impact of several important spark parameters related to data partitions, shuffling, compression, and locality (basic spark parameters) for training Multi-Layer Perceptron model on Spark standalone cluster. Then based on the investigation, we tuned basic spark parameters for training Multi-Layer Perceptron model and used it for tuning Spark when training Multi-Layer Perceptron model for web access prediction. Through experiments, we showed the accuracy of web access prediction based on our proposed web access prediction model. In addition, we also showed performance improvement in training time based on our spark basic parameters tuning for training Multi-Layer Perceptron model over default spark parameters configuration.

▸ **Key words**: Apache Spark, Neural network, Parallel deep learning, Parameter tuning, Web access prediction

[요 약]

웹에서 정보 접근에 대한 폭발적인 주문으로 웹 사용자의 다음 접근 페이지를 예측하는 필요성이 대두되었다. 웹 접근 예측을 위해 마코브(markov) 모델, 딥 신경망, 벡터 머신, 퍼지 추론 모델 등 많은 모델이 제안되었다. 신경망 모델에 기반한 딥러닝 기법에서 대규모 웹 사용 데이터에 대한 학습 시간이 엄청 길어진다. 이 문제를 해결하기 위하여 딥 신경망 모델에서는 학습을 여러 컴퓨터에 동시에, 즉 병렬로 학습시킨다. 본 논문에서는 먼저 스파크 클러스터에서 다층 Perceptron 모델을 학습 시킬 때 중요한 데이터 분할, shuffling, 압축, locality와 관련된 기본 파라미터들이 얼마만큼 영향을 미치는지 살펴보았다. 그 다음 웹 접근 예측을 위해 다층 Perceptron 모델을 학습 시킬 때 성능을 높이기 위하여 이들 스파크 파라미터들을 튜닝 하였다. 실험을 통하여 논문에서 제안한 스파크 파라미터 튜닝을 통한 웹 접근 예측 모델이 파라미터 튜닝을 하지 않았을 경우와 비교하여 웹 접근 예측에 대한 정확성과 성능 향상의 효과를 보였다.

▸ **주제어**: 아파치 스파크, 신경망, 병렬 딥러닝, 파라미터 튜닝, 웹 접근 예측

• First Author: Gantur Togtokh, Corresponding Author: Kyung-Chang Kim
*Gantur Togtokh (gantur.t@gmail.com), Dept. of Information Technology, Ulaan Baatar University
**Kyung-Chang Kim (kckim@hongik.ac.kr), Dept. of Computer Engineering, Hongik University
• Received: 2019. 10. 31, Revised: 2019. 11. 27, Accepted: 2019. 11. 27.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

Due to information shift from local site to the web, access latency and server overloading to the web server have been increasing. It results in increased delay to access web server and to download web pages to client. One of the main solutions to solve the delay is web prefetching. Web prefetching reduces access latency for web user by fetching user's next web page to client cache before user request it. Web prefetching is based on the result of web access prediction. Web access prediction is predicting a web page which user may visit next, while the user is visiting current page. Web prefetching is not the only area of utilizing web access prediction. There are many other areas that utilizes web access prediction such as web personalization, web recommendation, etc. One of the models that is used for web access prediction is the deep neural networks. In the web access prediction model based on deep neural network as shown Figure 1, web logs are analyzed by data preprocessing techniques in web usage mining, then web users' next access is predicted on the output of data preprocessing (users' sessions) using deep neural networks since web access prediction is the classification problem that try to predict user's next access based on the history of the user's previously visited web pages [1]. Web usage mining (WUM) is one of the categories of web mining which is used to mine web logs to discover useful patterns and knowledge [2]. Web usage mining has three phases: data preprocessing, pattern discovery and pattern analysis. Data cleaning, user identification and session identification are basic steps of data preprocessing in WUM [3]. For deep neural network models, training time in training deep neural network models on large-scale web usage data is very much. Because recent web usage data is very large. To speedup training time in training deep neural network models, the deep neural network models is run on the cluster of computers in parallel (distributed deep neural network). In this paper, we chose the Multi-Layer Perceptron model as deep neural network model for web access prediction. In order to train Multi-Layer Perceptron rapidly on large-scale users' sessions for web access prediction, we used the Spark that is the current state of the art distributed computing framework for processing big data. Spark has component (Spark MLlib) which have distributed implementation of various machine learning algorithms. Spark uses Resilient Distributed Dataset (RDD) [4] abstraction which is a fault-tolerant distributed collection of data over nodes of cluster. Each partition of RDD can be computed on different nodes of the cluster in parallel. RDDs support two types of operations: transformations, actions. Transformations create new RDDs from an existing one and actions return any result of computation on RDDs.

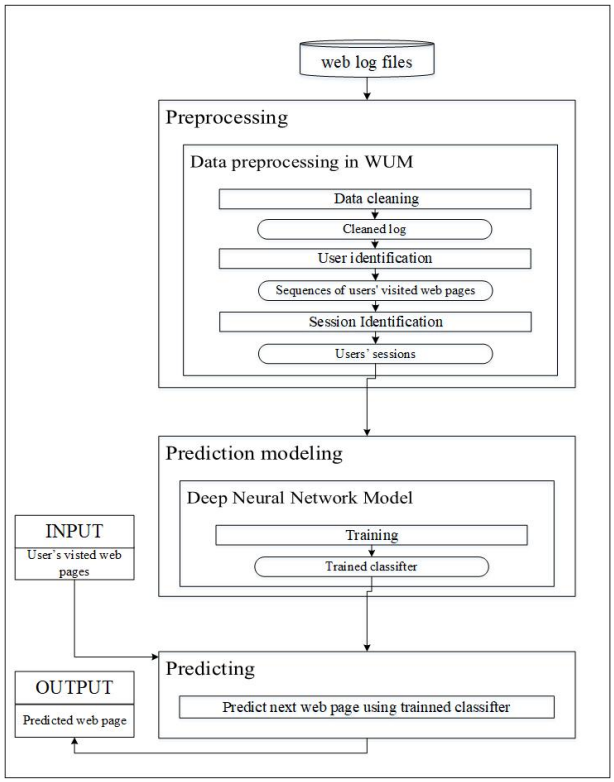

Fig. 1. A typical architecture of web access prediction model based on deep neural network.

Using Spark, application performance depends on parameters tuning (parameters optimization).

Tuning spark parameters is a complex and challenging task [5]. Because spark has 150 configurable parameters and default configuration is enough for run a spark program, and impact of parameters may vary from application to application and also from cluster to cluster [6]. Therefore, in the some case, tuning spark parameters is needed in application-independent manner. The related works regarding spark parameters tuning in chapter II does not consider deep neural network models as investigation benchmarks for investigating their spark parameter tuning. However, Multi-Layer Perceptron training is computation-intensive application like K-means and we investigate the impact of several important Spark parameters related to data partitions, shuffling, compression, and locality for training Multi-Layer Perceptron on Spark standalone cluster because the performance of applications on Spark may vary from application to application [4]. Then, based on experiment result of impact of spark parameters on MLP, we obtain spark basic parameters tuning algorithm on MLP training. We then use it for tuning spark parameters when training Multi-Layer Perceptron model for web access prediction on Spark.

The rest of this paper is organized as follows. Section II introduce related works for web access prediction and spark parameter tuning. Section III considers spark-based web access prediction and spark parameters tuning for training MLP model. Section IV shows the experiment results. Section V concludes the work.

II. Related works

1. Parameter tuning in Spark

In the Spark, as we mentioned in introduction, application performance is depend on parameters tuning. Tuning spark parameters is time consuming and challenging task [5]. Because Spark has 150 configurable parameters, default configuration is enough for run a spark program, and impact of parameters on performance may vary from application to application and also from cluster to cluster [6]. Therefore, for some application, Spark is needed application-independent parameter tuning. In this section, we will discuss several works related to spark parameter tuning and consider important parameters to be used for tuning spark when training Multi-Layer Perceptron model.

Spark guidelines documentation [7] summarized followings as tuning guidelines: 1) Data serialization plays an important role in the performance of any distributed application. Spark provides two serialization libraries: Java and Kryo. Java is default serialization and based on Java's ObjectOutputStream framework. It is flexible but quit slow. Kryo serialization is based on Kryo library, and faster and compact than java serialization. [7] recommended to try Kryo serialization in any network intensive application. 2) Memory is used for two purposes in spark: execution and storage. Execution memory is used for computation and storage memory is used for caching and distributing internal data across the cluster. Execution and storage share a unified region in spark. There are two relevant configurations related to Execution and storage memory. Default configuration is adapted for the most of application. In the most case user does not need to change the default configuration. 3) Level of parallelism must be adapted to utilize cluster resources fully. They recommend 2-3 tasks per core in the cluster. 4) Data locality gives biggest impact on the performance of spark application. If data and code are together then computation is performed fast. Spark tries to schedule all tasks to place to closest to its data. However it is not always possible. In situation there is no unprocessed data on any idle executor but there is unprocessed data another busy executors, Spark waits until busy executors finish its tasks and process its unprocessed data, or immediately start new task on idle executors to process unprocessed data where

placed in busy executors in another nodes. In the second case it requires data movement between nodes. In spark, we can configure the waiting time for waiting to start new task on idle executors. [7] recommended increasing the waiting time when our tasks are long and with poor locality. In contrast, we should decrease waiting time when our tasks are too short and with poor locality.

[8] was first attempt to examine important spark parameters on real world cluster with a high-performance computing (HPC) setup. It mainly focused on parallelism configuration and network interfaces (Ethernet or InfiniBand). It used two applications: sorting and k-means as workload for testing. Main results of this work are summarized as follows 1) in the case of k-means which is computation-intensive application, configuring level of parallelism (number of data partitions) by allocating single data partition per core obtains better performance. For sort-by-key which is shuffling intensive application, allocating two data partition per core obtains better performance than single data partitions per core. 2) Number of executors per worker node or number of cores per executer is configurable parameter. For the two applications, experiment shows that executers with large number of cores obtain better performance. 3) For network interface test, InfiniBand interface is better on sort-by-key application than Ethernet interface.

Alpine Data [9] propose tips for tuning spark to system administrators. It is for spark Yarn cluster. Such as data partitions, optimization, garbage collection, and some checks are covered in the tips. It is available online.

[5] and [6] investigates impact of important 12 tunable spark parameters with regards to shuffling, compression and serialization on the short-by-key, and shuffling, and k-means applications using Marenostrum III computing infrastructure of the Barcelona Supercomputing Center. Then based on the results of impact, a trail-and-error methodology, which requires little number of experimental runs is derived and propose a systematic methodology for generating candidate configurations that can adapt to any spark platform.

[10] investigates impact of data volume on the performance of applications such as word count, Grep, Sort, Nave Bayes and K-Means.

2. Web access prediction based on neural networks

In recent years, the needs of predicting web user's next access have been increasing, and several model are used on web access prediction area such as markov models, deep neural networks, support vector machines, and fuzzy interference models. in this paper, we chose the Multi-Layer Perceptron model as deep neural network for web access prediction. Multi-Layer Perceptron (MLP) is considered as a type of deep learning models since there are multiple hidden layers of nodes between input and output layers. In MLP, nodes of any layer in the network connect to nodes of next layer in a graph which directs forward. Connections between nodes in the neural network represent weights of nodes. Figure 2 shows an example of Multi-Layer Perceptron with three hidden layers.

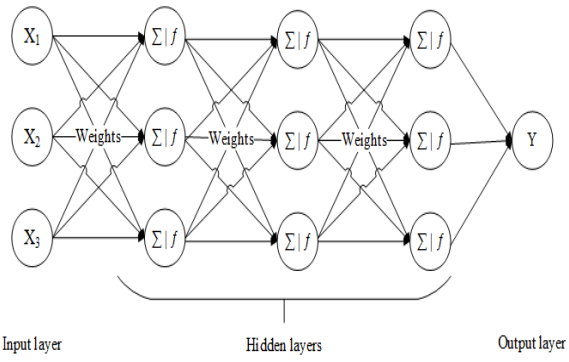

Fig. 2. Sample architecture of Multi-Layer Perceptron model with three hidden layers

Here we discuss web access prediction based on Artificial Neural Network (ANN).

Om et al. [12] are proposed a prediction model uses feed forward artificial neural network and k-mean clustering. In this work, author used ANN for finding closest user sessions cluster with

current user's session. K-mean clustering is used for clustering user sessions by its similarity. Then feed forward ANN model is trained on these session clusters and used to find highest probability cluster with current user's visited web pages sequence.

Pruthvi [13] proposed a next page prediction model which uses ANN. In this model, ANN implemented in map-reduce programming model in Hadoop framework for training large amount of dataset rapidly. This paper is aimed to reduce the time of the training process.

Vidushi et al. [14] proposed a model uses self-organizing map (SOM) and ANN. This model has three layers: clustering, self-organizing map and neural network. In clustering, dataset is filtered using clustering approach. Cluster that represents the high usage pages will selected for further processing. Next, SOM is applied to analysis web pages usage and prediction to assign weightage to various web pages. Finally, ANN is trained to perform next page prediction.

III. Spark-based MLP for web access prediction

In order to train multi-layer perceptron model for web access prediction on large-scale input rapidly, we used Spark for distributed training.

1. Spark basic parameters for tuning on MLP training

In order to tune spark parameters for training MLP model for web access prediction, we investigate the impact of spark several important parameters. In this paper, we consider the following 8 parameters related to data partitions, shuffling, compression, and locality as spark basic parameters for tuning on MLP training.

1. **spark.serializer:** This parameter determines class to be used for serializing objects [11]. Default is java serialization. In the spark documentation, it is defined that Kyro serialization is faster than java serialization. And [9] recommended Kyro serialization. However, [5] is also considered this parameter impact on its test benchmarks, we consider its impact on MLP because that parameters impact may vary from application to application, cluster to cluster [6].

2. **spark.defualt.parallelism:** This parameter determines level of parallelism (data partitions). For distributed shuffle operations like reduceByKey and join, the largest number of partitions in a parent RDD is defined default [11]. For operations like with no parent RDDs, default is depends on the cluster manager e.g. number of cores on the local machine, total number of cores on all executor nodes or ext. [11] In our experiment, we set default as 1 task per core. In the spark documentation [7], it is recommended 2-3 task per core. In [8], for computation-intensive application like K-means, it results in better performance on 1 task per core. However, MLP is computation-intensive like K-means. We also consider its impact on MLP training.

3. **spark.executor.instances:** This parameter defines number of executors per worker node. [8] shows that increasing number of executors per worker node degrades performance on k-means.

4. **spark.executor.cores:** This parameter defines number of core per executer. [8] shows that increasing number of cores per executor by decreasing number of executors increases performance on k-means.

5. **spark.shuffle.manager:** This parameter defines implementation to use for shuffling data. Default is sort, available implementations on our spark version for experiment is sort and tungsten-sort. [6] choose between tungsten-sort and hash. we choose between sort and tungsten-sort.

6. **spark.shuffle.compress:** This parameter defines whether data partitions are compressed before transferred over the network or not. Default is true. It's clear that if the amount of data that transfer over network is a lot, it's better to compressed than uncompressed. However, [6] investigate this parameter on its benchmark, we investigated it on MLP.

7. **spark.shuffle.ReduceLocality:** This parameter defines whether scheduling all reducers to the same executor or any executor. This parameter not mentioned on spark configuration document [11] explicitly. Default is true in our spark version. We investigated impact this parameter on different number of data partitions.

8. **spark.locality.wait:** This parameter defines how long wait between locality levels. There are four locality levels [11]: process, node, rack and then any. Default value is 3sec. In Spark documentation [7], it is recommended to increase these setting if tasks are long and with poor locality. [5] and [6] did not consider impact of locality.

2. Web page prediction using MLP

We obtain spark basic parameters tuning algorithm on MLP as shown in Figure 3 and use it when we train MLP model on Spark for web access prediction. In the algorithm, parameters that have more than 3% impact are used. spark.executor.cores, spark.executor.instances parameters were well enough configured for MLP model training by default. Two main trials of the algorithm is based on observation from investigation that if number of data partitions is equal to number of cores for application, spark schedules to distribute number of data partitions to all executors equally in the case of scheduling all reducers to any executors (Spark.shuffle.ReduceLocality = false). Using the algorithm, we can tune spark important parameters for training MLP model with three experimental trials.

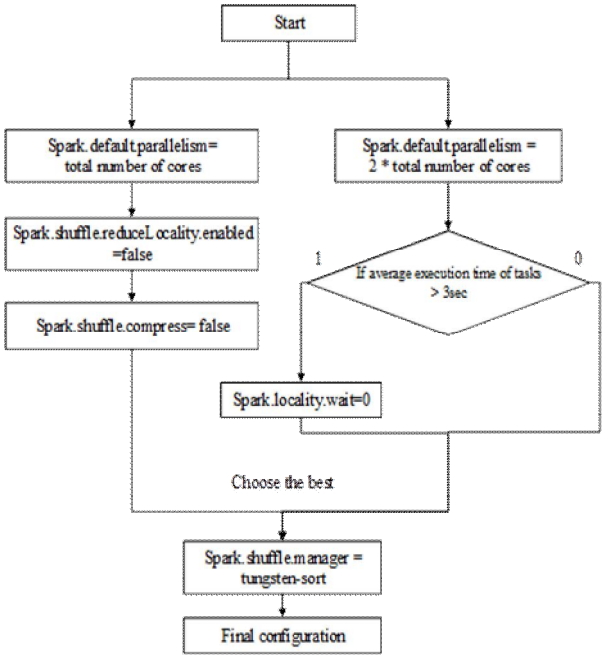

Fig. 3. A spark basic parameter tuning on MLP as block diagram

Spark basic parameters tuning algorithm is based on this observation and experiment result of parameters impact in explained later.

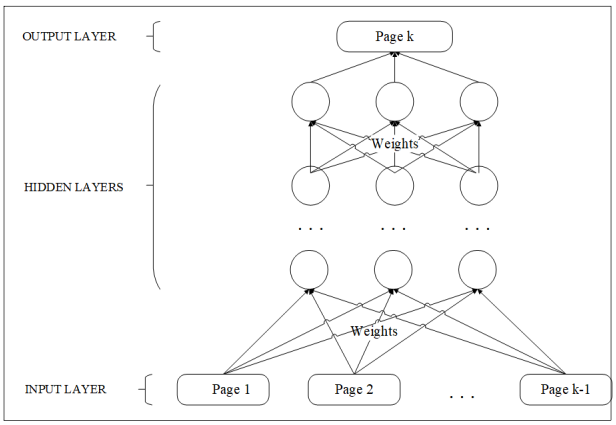

Fig. 4. Modeling users' sessions for web access prediction on MLP model

In our MLP model for web access prediction as shown in Figure 4, last web page in a user session is passed as output and remaining web pages in the user session are passed as input. If length of users' sessions is k, pages from 1 to k-1 in a user session are passed to input nodes and the k-th page in the user session is passed to output node.

IV. Experiments

1. Experiment specification

Specification of the experiment environment is as follows. The computers used in our experiment have all Intel® core i3-1430, CPU is 3.4GHz, 4GB RAM. The operating system is windows 10, and Spark-2.3.2 is used.

2. Datasets

We used two well-known server access logs: NASA [15] and Clarknet [16] that are available for free. The NASA dataset contains seven days' worth of all HTTP request to the NASA Kennedy Space Center server in Florida. Clarknet dataset contains two weeks' worth of all HTTP request to the ClarkNet server. The log format of the two datasets are command log format.

3. Preprocessing

Before modeling users' sessions for web access prediction, large-scale raw web log must be preprocessed by data preprocessing steps in WUM: data cleaning, user identification, and session identification, and formed as users' sessions. In data cleaning, all irrelevant log records, that are graphic file records, error status records, are removed. Then too rear and frequent pages are also removed. In user identification, individual users are identified by the same IP address. In the session identification, we used uniform 15 minutes page stay-time as threshold value, when we create new sessions.

4. Prediction results

We tested the accuracy of web access prediction based on our model on both the NASA and ClarkNet datasets.

We tuned MLP model for web access prediction by hyper parameter, which has [500, 700, and 1000] hidden layer structure, 0.005 learning rate and 32 batch size, with sufficient number of training epochs. In addition, we divided the datasets into 70% training and 30% testing. We randomly oversampled minority classes by 25% because users' sessions based on these datasets were totally unbalanced.

Figure 5 shows accuracy of prediction based on our proposed web access prediction model on test datasets.

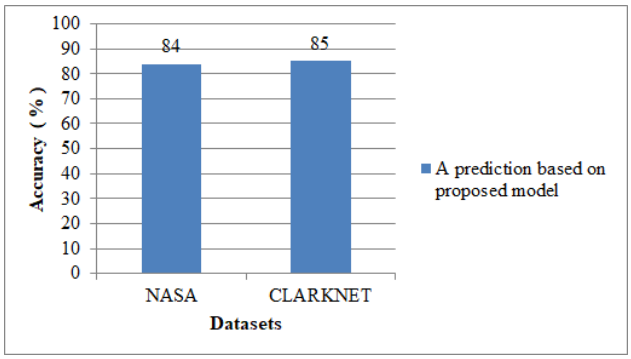

Fig. 5. Prediction accuracy

5. Impact of spark basic parameters on MLP

In order to tune spark parameters on MLP for prediction, we investigated the impact of spark basic parameters on spark standalone cluster with 1 master node and 4 worker nodes.

For experiment, we measured parameters impact in terms of training time by comparing with default parameters configuration on dataset with 10000 records of 10 dimensions after 10 training epoch. Figure 6 shows experiment result of impact of some basic spark parameters for training MLP. We show the parameters that have impact with more than 2%. Figure 6 shows that parameters related to locality have major impact on the performance, while parameters related to locality are not configured enough on MLP training by spark default configuration.

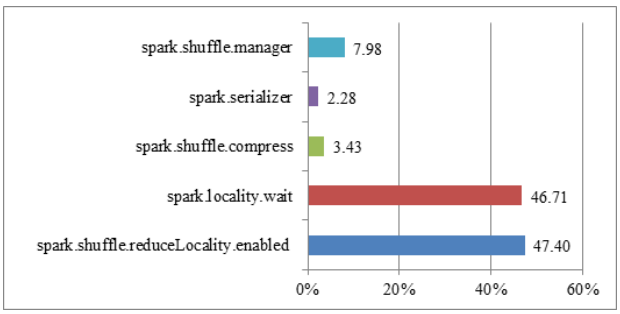

Fig. 6. Impact of basic spark parameters on MLP training

In addition, we investigated impact of number of data partitions for training MLP. Figure 7 shows experiment result of impact of number of data partitions for training MLP on two different lengths of users' sessions.

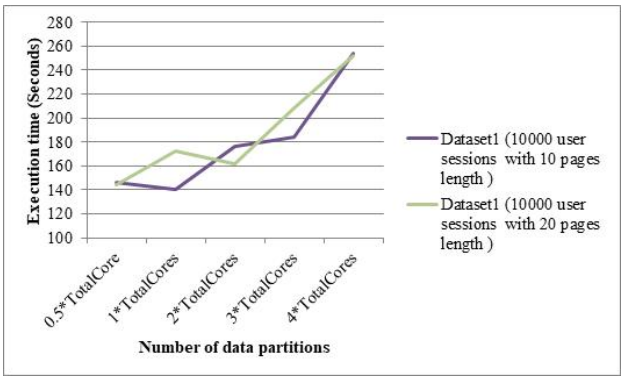

Fig. 7. Impact of the number of data partitions on MLP training

6. Performance result of models for training MLP

In order to investigate the performance of distributed deep nerual network on Spark, we conducted the experiment for training MLP. In order to investigate performance of distrusted deep learning on spark, we performed various experiment for training MLP. The experiments for training the MLP model include: (1) MLP training without distributed training, (2) MLP training on Spark with spark default parameter configuration and (3) MLP training on Spark with configuration using spark basic parameter tuning algorithm on MLP. The result is shown in Table 1.

Table 1. Comparison of models for training MLP

Models for training MLP	Training time in seconds after 10 epoch
MLP training without distributed training	330
MLP training on Spark with spark default parameter configuration /4 worker nodes/	280
MLP training on Spark with configuration based on spark basic parameter tuning algorithm /4 worker nodes /	147

The experiment result in Table 1 for training MLP shows that the training time using our Spark basic parameters tuning algorithm is a lot less than the training time to train Multi-Layer Perceptron on Spark using spark default parameter configuration. In either case, the training time is less than without distributed training.

V. Conclusion & future work

We used deep neural network model for web access prediction. To reduce training time. deep neural network models are trained on cluster of computers in distributed manner. Apache spark is distributed computing framework leveraged on main memory for processing big data. In the Spark, depending on parameter tuning, application performance is different. Tuning spark parameters is a complex and challenging task. In this paper, we investigated the impact of some important spark parameters on Multi-Layer Perceptron training. Then based on these observation and the results of experiment on the impact of spark basic parameters on MLP, we tuned spark basic parameters for training Multi-Layer Perceptron, and used it for web access prediction on Spark. The experiment for distributed deep neural network on Spark shows that, in terms of training time, the parameter configuration based on tuning the spark basic parameters on Multi-Layer Perceptron performs better (i.e. reduces training time) than the training time using spark default parameter configuration. In the future, we will consider recurrent neural network (RNN) for web access prediction on Spark.

REFERENCES

[1] Mamoun A.Awad, Issa Khalil, "Prediction of User's web-browsing behavior: Application of Markov Model", IEEE Transactions on Systems, Man, And Cybernetics – Part B: Cybernetic, vol. 42, no. 4, pp. 1131-1142, August 2012.

[2] Wang, Yan "Web Mining and Knowledge Discovery of Usage

Patterns", http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.6743& rep=rep1&type=pdf,2000.

[3] Giovanna Castellano, Anna M. Fanelli, and Maria A. Torsello, "Web Usage Mining: Discovering Usage Patterns for Web Applications", Advanced Techniques in Web Intelligence-2, SCI 452, pp. 75-104, 2013.

[4] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M.J.Franklin, S. Shenker, I. Stoica "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing" 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI) (2012), pp. 15-28

[5] AnastasiosGounaris, Jordi Torres, "A Methodlogy for Spark Parameter Tuning", Big Data Research, Volume 11, pages 22-32, 2018

[6] P. Petridis, A. Gounaris, J. Torres "Spark parameter tuning via trial-and-error," Advances in Big Data - Proceedings of the 2nd INNS Conference on Big Data (2016), pp. 226-237

[7] Spark guidelines documentatin for tunning https://spark.apache.org/docs/latest/tuning.html

[8] R. Tous, A. Gounaris, C. Tripiana, J. Torres, S. Girona, E. Ayguadé, J.Labarta, Y. Becerra, D. Carrera, M. Valero "Spark deployment and performance evaluation on the marenostrum supercomputer" IEEE International Conference on Big Data (Big Data) (2015), pp. 299-306

[9] Alpine Data tuning tip http://techsuppdiva.github.io/spark1.6.htm

[10] A.J. Awan, M. Brorsson, V. Vlassov, E. Ayguade "How data volume affects spark data data analysitcs on a scale-up server" (2015)

[11] Spark parameters configuration http://spark.apache.org/docs/latest/configuration.htm

[12] Om Prakash Mandal, Hiteshware Kumar Azad "Web Access Prediction Model using Clustering and Artificial Neural Network", IJERT, Vol.3 Issue 9, 2014.

[13] Pruthvi, "Web-Users' Browsing behavior Prediction by Implementing Neural Network in MapReduce", IJAFRC, Vol.1 Issue 5, 2014

[14] Vidushi, Yashpal Singh, "SOM Improved Neural Network Approach for Next Page Prediction" International Journal of Computer Science and Mobile Computing, Volume 4, Issue 5, pg. 175-181, May 2015

[15] NASA: web access log dataset: http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

[16] ClarkNet: web access log dataset: http://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html

Authors

Gantur Togtokh received the B.S. in Computer Engineering from Ulaan Baatar University, M.S. in Computer Engineering from Soongsil University in 2010 and Ph.D in Computer Engineering from Hongik University in 2019. Currently, Dr. Togtokh is assistant professor in the Department of Computer Engineering at Ulaan Baatar University, Mongolia. He is interested in image processing, bid data analysis, sensor databases, web databases, and deep learning.

Kyung-Chang Kim received the B.S. in Computer Science from Hongik University in 1978, M.S. in Computer Science from KAIST in 1980 and Ph.D in Computer Science from University of Texas at Austin in 1990. Dr. Kim joined the faculty of the Department of Computer Engineering at Hongik University, Seoul, Korea, in 1991. He is currently a Professor in the Department of Computer Engineering, Hongik University. He is interested in main memory databases, sensor databases, web databases, Internet of Things and big data processing.