International Journal of Grid and Distributed Computing
Vol. 9, No. 8, (2016), pp.197-208
http://dx.doi.org/10.14257/ijgdc.2016.9.8.17

An Optimization Scheme in MapReduce for Reduce Stage

Qi Liu[1], Weidong Cai[1], Baowei Wang[1], Zhangjie Fu[1], Nigel Linge[2]

[1]*Nanjing University of Information Science and Technology, 219 Ningliu Road, Nanjing, Jiangsu, 210044, China*
[2]*The University of Salford, Salford, Greater Manchester, M5 4WT, UK*
qrankl@163.com

Abstract

As a widely used programming model for the purposes of processing large data sets, MapReduce (MR) becomes inevitable in data clusters or grids, e.g. a Hadoop environment. Load balancing as a key factor affecting the performance of map resource distribution, has recently gained high concerns to optimize. Current MR processes in the realization of distributed tasks to clusters use hashing with random modulo operations, which can lead to uneven data distribution and inclined loads, thereby obstruct the performance of the entire distribution system. In this paper, a virtual partition consistent hashing (VPCH) algorithm is proposed for the reduce stage of MR processes, in order to achieve such a trade-off on job allocation. Besides, experienced programmers are needed to decide the number of reducers used during the reduce phase of the MR, which makes the quality of MR scripts differ. So, an extreme learning method is employed to recommend potential number of reducer a mapped task needs. Execution time is also predicted for user to better arrange their tasks. According to the results, VPCH can lead to load balancing and our prediction model can provide fast prediction than SVM with similar accuracy maintained.

Keywords*: MapReduce; Load Balancing; Consistent Hashing; Extreme Learning; Fast Prediction;*

1. Introduction

In recent years, with the explosive growth of data and its processes in the Internet, cloud computing has been widely studied in both academia and industry, in order to provide users such a distributed system with on-demand services, computing abilities and storage resources. Proposed by Google in 2004, MapReduce (MR) [1] has become the most popular distributed computing model used in a cloud environment, where large-scale datasets can be handled/processed using map and reduce procedures in the cloud infrastructure transparently.

Besides map and reduce, other internal processes integrated into MR have also been analyzed and optimized [2]. Taking the partition procedure as an instance, Hadoop uses a hash function with modulo operations to calculate partition keys, such that same number of packets maintained in each group can be guaranteed. However, such a method can cause tilted allocation of tasks to different reducers. In this paper, a new scheme called VPCH is proposed to implement virtual partitioning. Such a scheme can ensure the load of each reducer during the reduce phase is relatively balanced. The total execution time is actually reduced due to even distribution of task to each reducer. Moreover, in this paper, a novel prediction model based on the ELM algorithm is proposed to facilitate the execution of reduce operations in a cloud environment.

A practical Hadoop environment has been implemented in our laboratory, retaining both original and refined partitioning schemes so users can select and compare either of them for their actual tasks and/or performance evaluation. According to our results, the

ISSN: 2005-4262 IJGDC

VPCH algorithm has depicted shorten execution time on both reduce phases and entire task completion. ELM for prediction also has a good performance in the environment.

The rest sections of the paper are organized as followed. Related work is given in Section 2, followed by Section 3, where our load balancing approach is detailed. In Section IV, testing environment and corresponding scenarios are design for the verification and evaluation of the method. Our fast prediction method is described in Section 4. Finally, conclusion and future work on load balancing strategies in a cloud platform are discussed in Section 6.

2. Related Work

There are various approaches to obtain work load in the datanodes of a MR system. Authors in [3] tried to repartition tasks from slow workers to faster ones by monitoring real-time Map and Reduce jobs to ensure that all available nodes can finish the jobs at the same time. This method can handle all kinds of load deflection, but it changes Hadoop greatly with complicated modification/configuration, as well as extra network cost on the redistribution of tasks.

Some studies address the issues of mapping sets of tasks onto sets of processors according to context information (e.g. execution history [4], sampled data skew [5], *etc*.), such that overall execution time can be minimized. However, these methods failed to have their models verified via a practical cloud platform, which consequently ignored corresponding configuration in a cloud computing environment, e.g. number of reducers.

Neural Network (NN) algorithms have been employed in a cloud. An adaptively partition algorithm called HAP was proposed in [6], where reduce jobs can be distributed based on estimated work threshold using SVM in a heterogeneous environment. Extra cost on execution time, on the other hand is consumed when splitting and merging input K-V chains at the reduce phase. In addition, training time of these models needed to be taken into account as well.

Offline or online profiling has been proposed by previous work to predict application resource requirements by using benchmarks or real application workloads. Timothy Wood *et al.* [7] designed a general approach to estimate the resource requirements of applications running in a virtualized environment. They profiled different types of virtualization overhead and built a regression-based model to map the native system profile into the virtualized system. Their model focused on relating the resource requirements of real hardware platform to the virtual one. Sadeka Islam *et al.* [8] studied the changing workload demands by starting new VM instances, and proposed a prediction model for adaptively resource provisioning in a cloud. Complex machine learning techniques were proposed in [9] to create accurate performance models of applications. They estimated the resource usage for an application by an approach called PQR2. Jing *et al.* [10] presented a model that can predict the computing resource consumption of MapReduce applications based on a Classified and Regression Tree.

3. Virtual Partition Consistent Hashing

Consistent hashing has been well used for P2P communication based on DHT [11], chord [12], KAD [13], *etc*. In terms of nodes allocation in a cloud environment, especially when the load-balancing strategy is taken into account, consistent hashing shows its advantage on hitting probability with uniform distribution. Such arrangement also makes the platform a (nearly) complete decentralized system.

In addition, virtual nodes are defined in order to fulfill the gap between data splits and to-be-assigned physical nodes. A Toeplitz matrix has been designed and implemented to ensure…, such that each virtual node is responsible for a respective range of items, based on their IDs. These virtual nodes can then be assigned to corresponding physical ones.

Virtual Partition Consistent Hashing provides such an even allocation of mapped data splits for reduce purposes.

3.1. Generation of VPCH Hash Circle

Based on consistent hashing mechanism, the VPCH iterates all reducers in a cloud platform and initialize them into a hash circle. The generation of such a VPCH hash circle is shown in Algorithm 1.

Algorithm 1. Generation of VPCH Hash Circle

Input: All reducers of the system

Output: VPCH Hash Circle of the nodes

Steps:

```
1.  Design a hash function Toeplitz(x)
2.  For each reducer of the system
3.      For each virtual node of the system
4.          Use the Hash function   to calculate the hash value
5.      End For
6.  End For
7.  Sort hash values of the nodes
8.  Make all nodes into a circle hc
9.  Return hc
```

In Algorithm 1, the values of all nodes in the cloud platform are collected as input parameters. The hash circle is generated as the output.

3.2. Allocation of Mapped Data Splits

Through Algorithm 1, hash values of splits can be found so virtual nodes can then be mapped to reducer. The virtual nodes in the *hc* can be regarded as a linker with the shortest path to the reducer. The implementation details are shown in Algorithm 2.

Algorithm 2. Allocation of Mapped Data Splits

Input: Mapped Data splits list, *l_MDsplits*

Output: Keys of reducers, *reducer_Id*

Steps:

```
1.  Get hash values of  l_MDsplits
2.      For each reducer of the system
3.          If contains the value
4.              Return reducer_Id
5.          Else
6.              For each virtual node
7.                  If contains the value
8.                      Hit the reducer associated to the virtual node
9.                  Else
10.                     Hit the virtual with the shortest distance
Hit the associated reducer
11.                 End If
12.                     Return reducer_id
13.             End For
14.         End If
15.  End For
```

In Algorithm 2, mapped splits are employed as input sources. Keys of reducers according to the hash value are then returned.

4. A Prediction Model based on NO-ELM

Author names and affiliations are to be centered beneath the title and printed in Times New Roman 12-point, non-boldface type. Multiple authors may be shown in a two or three-column format, with their affiliations below their respective names. Affiliations are centered below each author name, italicized, not bold. Include e-mail addresses if possible. Follow the author information by two blank lines before main text.

Recently, artificial neural networks (ANNs) have been widely applied in applications involving classification or function approximation [14]. However, the training speed of ANNs is much slower than what a practical application needs. In order to overcome this drawback, the approximation capability of feed is employed to advance neural networks, especially in a limited training set. One of the most important achievement of this work is putting forward a novel learning algorithm in single hidden layer feed forward neural network (SLFNs)[15], i.e. ELM [15]– [21].

4.1. Number of Hidden Neurons Optimized ELM (NO-ELM)

In the basic ELM algorithm, the number of hidden neurons L is usually generated by iterating. To find the min $RMSE$ or R^2 that is close to 1, L needs to be trained into the best value. An optimized algorithm is therefore introduced to achieve the process, as shown in Algorithm 3.

Algorithm 3. Generate the number of hidden neurons

Input:

TS: size of the training set

IT: times of iteration

RT: times of running

Output: *L*: the number of hidden neurons

Steps:

```
While times of running is smaller than RT
    While times of iteration is smaller than IT
        If the accuracy Acc get this time is smaller the RMSE
            RMSE = Acc
            LRT equals the number of iteration
    End While
    If there is no LRT in the Collection <K, V>
        Add <LRT, 1> to the result Collection <K, V>
    else
        Get the number of element value in Collection <K, V>
        Add < LRT, value+1> to the result Collection <K, V>
    End If
End While
For each LRT in Collection <K, V>
    Find LRT with the max value
    Record LRT as L
End For
Return L
```


In Algorithm 3, the size of training set, as well as the time of iteration and execution is collected as input parameters. The number of hidden neurons is generated as the output.

4.2. The Process to Build the Prediction Model based on NO-ELM

The building progresses of the prediction model for the number of reducers and the execution time are as follows:

Step 1: **Data preprocessing**. First, samples that may contain great network congestion need to be removed. Then, the refined datasets will be split into training samples and test samples. The training samples are used for training prediction model and test samples are used to check if the prediction model has been well trained.

Step 2: **Model training**. To build the prediction model, training parameters of the model are obtained by using the training samples generated in Step1. The Specific processes include:

a) Randomly generate the weights between input layer and hidden layer, where hidden layer neurons w and the threshold b are set;

b) Calculate the hidden layer output matrix H;

c) Work out output layer weights.

Step 3: **DataValidation**. Use the data generated in Step 1 to validate the NO-ELM prediction model. According to the parameters trained in step 2 to get the predictive value of test set, and compare with the actual value to verify prediction performance of the model.

For the model to predict the number of reducers, the data format is set as *{reducer_no, execution_time, input_data_vol}*. Under the default circumstance, the prediction model recommends the number of reducers that can complete the task as soon as possible. The input format can then be simplified as *{reducer_no, input_data_vol}*. If the complete time of a task needs to be specified, the prediction model will recommend corresponding number of reducers. For doing that, the input format is as *{execution_time, input_data_vol}*.

5. Experiment and Analysis

In order to test the performance and benefits of the VPCH scheme, a practical Hadoop environment was built consisting of personal computers and a server. Each personal computer has 12GB of memory, a single 500GB disk and dual-core processors. The server is equipped with 288 GB of memory, a 10 TB SATA driver. Eight virtual instances are therefore created in the server with same specification as personal computers, i.e. the same amount of memory and storage space, as well as the same number of processors. In terms of role configuration, the server suns as the name node, whilst the virtual machines and personal computers run as the data nodes.

A shared open dataset [22] was manipulated as the input workload containing 26GB of text files. The dataset was further separated into 5 groups for testing purposes. A K-means (KM) clustering algorithm provided by Purdue MR Benchmarks Suite was used for partitioning operation in the cloud platform.

5.1. Evaluation of VPCH

In order to implement the VPCH scheme in the practical Hadoop system, the original partitioner class has been modified with Algorithm 1 and 2 integrated. An MR log analysis tool has also been implemented in the Hadoop system to record usage status and

individual load of reducers. Both original hashing strategy and VPCH were executed 20 times for each group of datasets in order to rule out potential contingency.

Figure 1 shows average execution time on 5 datasets by each reducer. It can be seen that reducers shared fairly uniform execution time using the VPCH hashing scheme, though in both the original and VPCH hashing schemes, reducer 1 and 2 spent more time than other four. The execution time of each reducer spent on each group of datasets using the VPCH scheme has been therefore analyzed, as depicted in Figure 2. The differences between reducer 1, 2 and other four are clearly illustrated. The reason that caused such an uneven situation is because by default, the Hadoop system enables reduce tasks right after 5% of map tasks have been finished. Such limited amount of mapped splits can be handled by reducer 1 and 2, which leaves the rest four suspended. Refined results can be seen in Figure 3 after the corresponding parameter, "mapreduce.job.reduce.slowstart.completedmaps" (MJR) was modified.

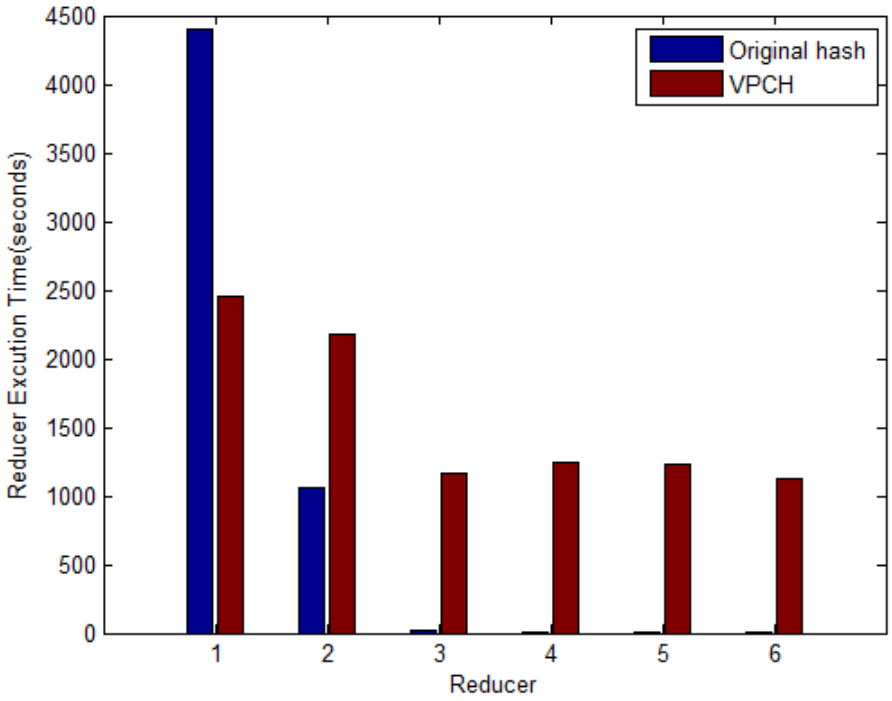

Figure 1. Execution Time of Each Reducer Using Original and VPCH Hashing Scheme

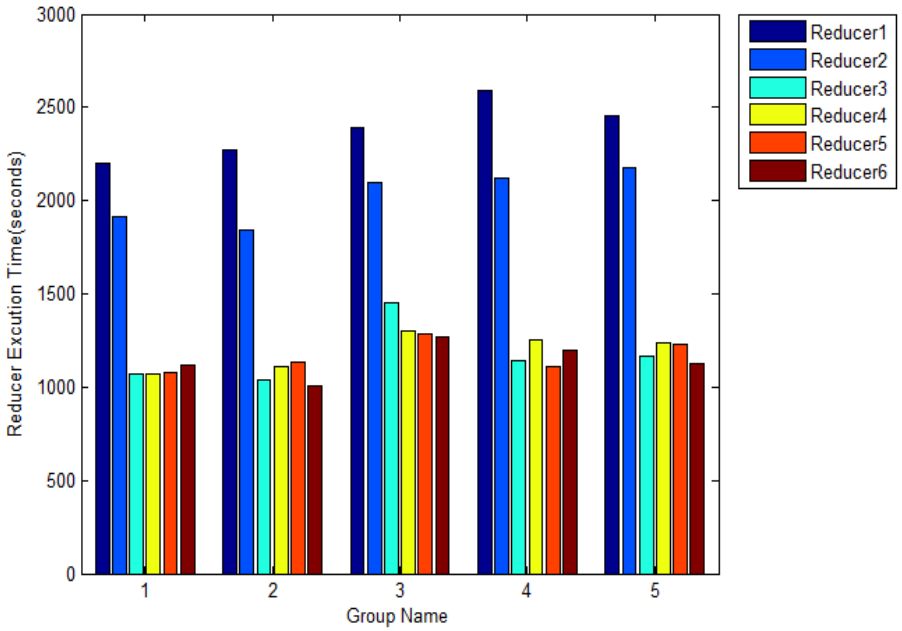

Figure 2. Workload of Each Reducer Using VPCH with Original MJR Settings

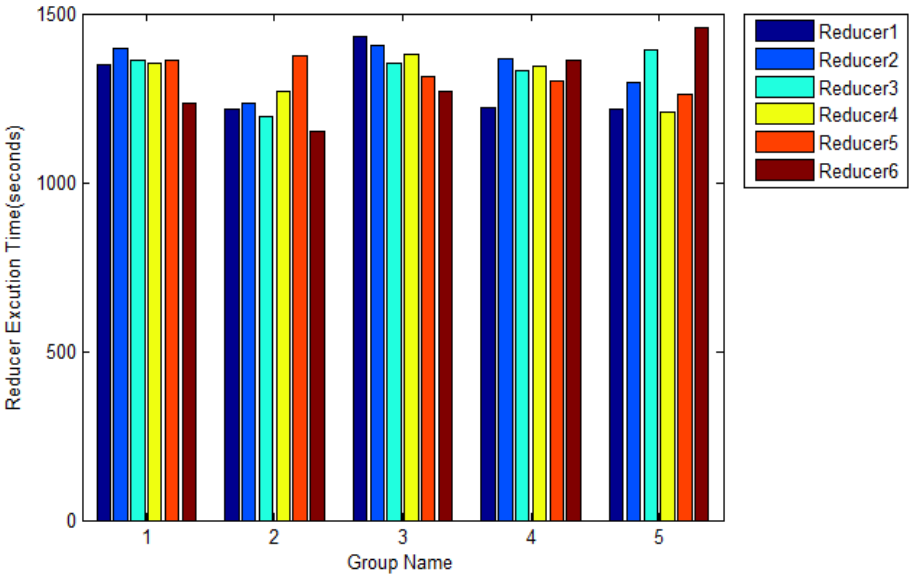

Figure 3. Workload of Each Reducer Using VPCH with Refined MJR Settings

Having MJR set modified facilitates improvement of VPCH. This can be seen in Table 1 and 2. Considering the execution duration of entire map-reduce procedures, comprehensive execution time of a task has also been measured, which can be seen in Table 3.

Table 1. Elapsed Time of Each Reducer Using VPCH with Refined MJR

Reducers	Elapsed Time in each phase			Total Elapsed Time
	Shuffle	Merge	Reduce	
1	7m18s	17s	15m18s	22m54s
2	7m3s	11s	15m49s	22m56s
3	6m54s	8s	14m38s	21m41s
4	6m45s	9s	15m53s	22m48s
5	6m42s	4s	14m33s	21m20s
6	6m37s	5s	15m41s	22m24s

Table 2. Workload of Reducers Using VPCH and Original Hashing

Hashing Scheme	Workload (GB) of Each Reducer					
	1	2	3	4	5	6
VPCH	4.5	4.3	4.5	4.6	4.1	4.2
Original Hashing	26.3	0	0	0	0	0

Table 3. Reduce and Total Execution Time of Both Hashing Schemes with or without MJR Modified

MJR	Reduce Phase		Total Execution Time	
	VPCH	Original Hashing	VPCH	Original Hashing
5%	39m38s	1h13m23s	50m12s	1h21m25s
100%	22m43s	1h3m47s	47m22s	1h27m43s

5.2. Evaluation of NO-ELM

Before training the NO-ELM prediction model, the samples are prepared following the equation below in order to meet the requirement of the model:

$$h_t = (s_t - \bar{s}) / (\bar{s}) \quad (1)$$

where $\bar{s}$ is the mean value of sample series, s_t is the value of one sample. Here, we remove s_t from the samples if h_t is greater than 5% and s_t is greater than $\bar{s}$ considering the cases where these samples may be affected by the network congestion.
The sample data are then normalized following the equation below:

$$s_t = (s_t - s_{\min}) / (s_{\max} - s_{\min}) \quad (2)$$

where $s_{\min}$ is the minimum value of sample series, $s_{\max}$ is the maximum value of samples. After normalization, the variation range of sample data is [0, 1].

In order to keep the generality, experiments in all performance evaluation parameters were run 50 times to get the average value. All the experiments bellow were operated under the circumstances that reduce tasks started when map tasks had finished using "Sigmoidal Function" as activation function. To verify the performance of the NO-ELM prediction model, we compare the predicted values of the NO-ELM prediction model with the test set samples (real values) and the SVM model.

5.2.1. NO-ELM for Predicting the Number of Reducers

The input data size varies from 1GB to 23.5GB, while the number of reducers is selected from 4 to 8. As seen in Figure 1, the predicted values generated by NO-ELM show a better trend following the real results than the SVM. In Table 4, 12 groups of the training time are depicted running the application with NO-ELM and SVM consumed, where the NO-ELM consumes less time than SVM in the train stage. Figure 4 shows the simulation results when predicting the number of reducers.

Table 4. Comparison between NO-ELM and SVM in Training Time

	1	2	3	4	5	6	7	8	9	10	11	12
NO-ELM (ms)	32	47	36	45	54	32	33	43	47	40	41	39
SVM (ms)	384	446	415	394	363	347	407	396	341	387	471	468

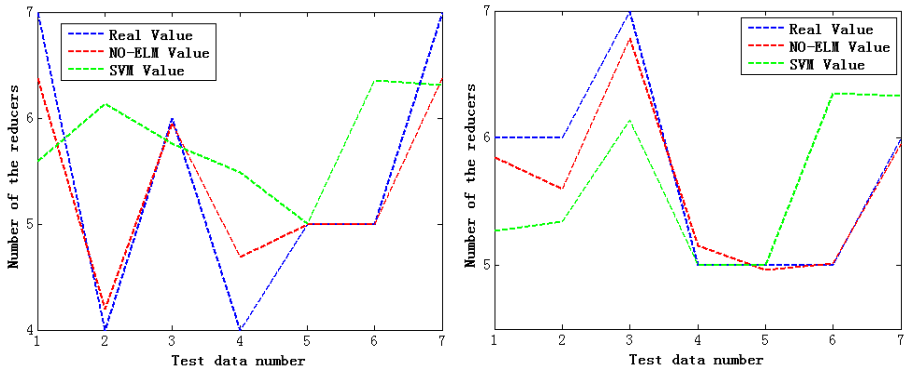

Figure 4. Experiment Comparison for Prediction Model on the Number of Reducers

5.2.2 NO-ELM for Predicting Execution Time

In this section, two samples are prepared in each group for training and testing purposes, as shown in Table 5. The simulation results are depicted in Figure 5.

Table 5. Comparison between NO-ELM and SVM in Training Ttime

Group No.	Input Data Size	Number of Split Datasets	Number of Training Set	Number of Test Set
1	1 GB	68	66	2
2	2 GB	5	3	2
3	5 GB	63	61	2
4	8.5 GB	56	54	2
5	10 GB	15	13	2
6	12.5 GB	57	55	2
7	16 GB	53	51	2
8	17 GB	16	14	2
9	19.5 GB	54	52	2
10	23.5 GB	62	60	2
11	25 GB	8	6	2
12	26.5 GB	67	65	2

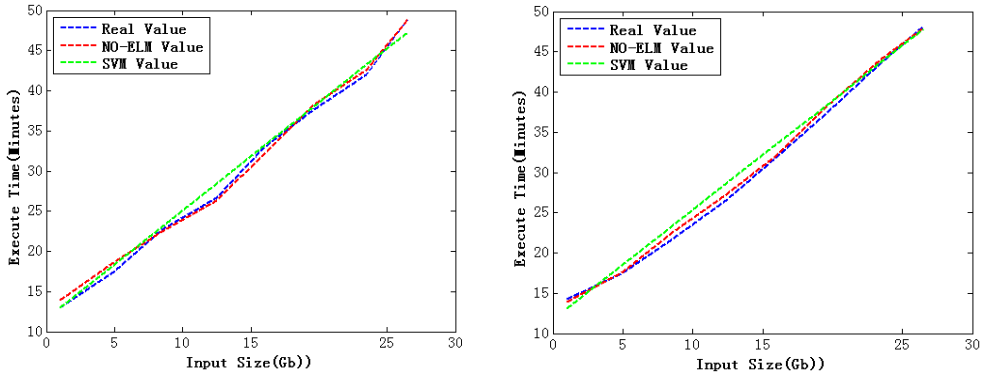

Figure 5. Experiment Comparison for Prediction Model of Execution Time

6. Conclusion

In this paper, a novel consistent hashing algorithm called virtual partition consistent hashing (VPCH) has been designed, through which the workload of each reducer is relatively balanced. The VPCH has been implemented in a real Hadoop environment, where the original hashing scheme in the Hadoop platform remains for comparison purposes. Through the results, the VPCH has depicted better performance in execution time and achieved fairly uniform distribution of mapped jobs. An extreme learning machine with the number of hidden neurons optimized (NO-ELM) has been introduced to analyze and predict the data. The NO-ELM method has been implemented in a real Hadoop environment, where the SVM algorithm has also been replicated for comparison purposes. Through the results, the NO-ELM has depicted better performance in the prediction of execution time and the number of reducers to be used.

Acknowledgments

This work is supported by the NSFC (61300238, 61300237, 61232016, U1405254, 61373133), Basic Research Programs (Natural Science Foundation) of Jiangsu Province (BK20131004), Scientific Support Program of Jiangsu Province (BE2012473) and Suzhou City (SYG201315), and the PAPD fund.

References

[1] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters", Communications of the ACM, vol. 5, no. 1, **(2008)**, pp. 107-113.
[2] Z. Fu, X. Sun, Q. Liu, L. Zhou and J. Shu, "Achieving Efficient Cloud Search Services: Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing", IEICE Transactions on Communications, vol. 98, no. 1, **(2015)**. pp. 190-200.
[3] Y. C. Kwon, M. Balazinska, B. Howe and J. Rolia, "SkewTune: Mitigating Skew in MapReduce Applications," ACM SIGMOD, Scottsdale, USA, **(2012)** May 20-24.
[4] J. Fu and Z. Du, "Load Balancing Strategy on Periodical MapReduce Job", Computer Science, vol. 40, no. 30, **(2010)**, pp. 38-40 (In Chinese).
[5] B. Gufler, N. Augsten,A. Reiser and A. Kemper, "Handling data skew in MapReduce", Proceedings of International Conference on Cloud Computing and Services Science, Noordwijkerhout, Netherlands, **(2011)** May 7-9.
[6] Y. Fan, W. Wu, Y. Xu and H. Chen, "Improving MapReduce Performance by Balancing Skewed Loads", China Communications, vol. 11, no. 8, **(2014)**, pp. 85-108.
[7] Wood, Timothy, Ludmila Cherkasova, Kivanc Ozonat, and Prashant Shenoy. "Profiling and modeling resource usage of virtualized applications." Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, New York, USA, **(2008)** December 1-4.
[8] Islam, Sadeka, Jacky Keung, Kevin Lee, and Anna Liu, "Empirical prediction models for adaptive resource provisioning in the cloud", Future Generation Computer Systems, vol. 28, no. 1, **(2012)**, pp. 155-162.

[9] A. Matsunaga and. A. Fortes, "On the use of machine learning to predict the time and resources consumed by applications", Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, **(2010)** May 17-20.
[10] T. J. Piao and J. Yan, "Computing resource prediction for mapreduce applications using decision tree", Proceedings of Web Technologies and Applications, Kunming, China, **(2012)** April 11-13.
[11] H. C. Hsiao and C. W. Chang, "A Symmetric Load Balancing Algorithm with Performance Guarantees for Distributed Hash Tables", IEEE Transactions on Computers, vol. 62, no. 4, **(2013)**, pp. 662-675.
[12] I. Woungang, F-H. Tseng, Y.-H. Lin, L.-D. Chou, H.-C. Chao and M. S. Obaidat, "MR-Chord: Improved Chord Lookup Performance in Structured Mobile P2P Networks", IEEE Systems Journal, vol. 9, no. 3, **(2014)**, pp. 1-9.
[13] R. Akavipat, M. N. Al-Ameen, A.Kapadia, Z. Rahman, R. Schlegel and M. Wright, "ReDS: A Framework for Reputation-Enhanced DHTs", IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 2, **(2014)**, pp. 321-331.
[14] T. H. OOng and N. A. M. Isa, "Adaptive Evolutionary Artificial Neural Networks for pattern classification", IEEE transactions on Neural Networks, vol. 22, no. 11, **(2011)**, pp. 1823-1836.
[15] G. B. Huang, Q. Y. Zhu and C. K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks", Proceedings of International Joint Conference on Neural Networks, Budapest, Hungary, **(2004)** July 25-29.
[16] G. B. Huang, Q. Y. Zhu and C. K. Siew, "Extreme learning machine: Theory and applications", Neurocomputing, vol. 70, no. 1, **(2006)**, pp. 489-501.
[17] A. Samat, P. Du, S. Liu, J. Li and L. Cheng, "E2LMs: Ensemble extreme learning machines for hyperspectral image classification", IEEE Journal of Selected Topics in Applied Earth Observations & Remote Sensing, vol. 7, no. 4, **(2014)**, pp. 1060-1069.
[18] M. Bianchini, M. and F. Scarselli, "On the Complexity of Neural Network Classifiers: A Comparison between Shallow and Deep Architectures", IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 5, **(2013)**, pp. 1553-1565.
[19] N. Wang, M. J. Er and M. Han, "Generalized Single-Hidden Layer Feedforward Networks for Regression Problems", IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 6, **(2015)**, pp. 1161-1176.
[20] C. Giusti and V. Itskov, "A no-go theorem for one-layer feedforward networks", Neural Computation, vol. 26, no. 11, **(2014)**, pp. 2527-2540.
[21] G. Huang, H. Zhou, X. Ding and R. Zhang, "Extreme learning machine for regression and multiclass classification", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 42, no. 2, **(2012)**, pp. 513-529.
[22] F. Ahmad, S. Chakradhar, A. Raghunathan and T. Vijaykumar, "Tarazu: optimizing MapReduce on heterogeneous clusters," ACM SIGARCH Computer Architecture News, vol. 4, no.1, **(2012)**, pp. 61-74.

Authors

Qi Liu, (M'11) received his BSc degree in Computer Science and Technology from Zhuzhou Institute of Technology, China in 2003, and his MSc and PhD in Data Telecommunications and Networks from the University of Salford, UK in 2006 and 2010. His research interests include context awareness, data communication in MANET and WSN, smart grid and cloud computing. His recent research work focuses on intelligent agriculture, meteorological observation systems based on WSN and Cloud Computing.