DESIGN AND DEVELOPMENT OF OPTIMIZATION ON MAPREDUCE METHOD FOR DATA MINING

Dr. Jitendra Sheetlani [1], Dr. J. P. Patra [2] Bhramara Bar Biswal [3]

[1]*Research Guide, Dept. of* Computer Science

Sri Satya Sai University of Technology and Medical Sciences,

Sehore Bhopal-Indore Road, Madhya Pradesh, India.

[2]*Research Co-Guide, Director. Dept. of* Computer Science *& Engineering*

Dept. of CSE, CSVTU University, Chhattisgarh

[3]*Research Scholar, Dept. of* Computer Science

Sri Satya Sai University of Technology and Medical Sciences,

Sehore Bhopal-Indore Road, Madhya Pradesh, India

Abstract

MapReduce is a developer-friendly framework that encapsulates the underlying complexities of distributed computing. It is increasingly being used across enterprises for advanced data analytics, business intelligence, and data mining tasks. But there are two questions bothering Hadoop users: how to improve the performance of MapReduce workloads, and how to estimate the time needed to run a MapReduce job. In this paper, we provide some performance optimization techniques on the premise of workload characterization. After the cluster achieving the best performance, we further propose a modeling method to help Hadoop users estimate the execution time of MapReduce jobs. For evaluation, typical benchmarks are utilized to evaluate the accuracy of our techniques.

Keywords: MapReduce, Data Mining , Hadoop.

INTRODUCTION

Map Reduce is a software framework that Hadoop uses to distribute work across hundreds or thousands of servers in a cluster. It is now widely used among companies for efficient large scale data processing. From our experience, most Hadoop developers are eager to know how to estimate the execution time of MapReduce jobs, which can help to find more efficient job scheduling policies or help to formulate resource allocation strategy more reasonably. However, the prediction results are useless unless the cloud computing environment has been tuned to supply its best service. However, there is no standard method to optimize the performance of all kinds of Map Reduce workloads. It depends on the resource consumption pattern of the job. The tasks Hadoop optimization and execution time estimation are extremely difficult, even for experienced Hadoop developers. In this case, there are two challenging problems remain unsolved: (1) how to get the best performance of different kinds of MapReduce jobs, and (2) how to estimate the execution time of MapReduce jobs. This paper is our first attempt to explore some optimization techniques and model the execution time of MapReduce jobs. Briefly, the main contributions of this paper are:

• Characterize MapReduce workloads by analyzing the resource consumption patterns.

• Provide some techniques that help to resolve bottlenecks and improve the performance of MapReduce jobs.

Propose an execution time model which can help to estimate the execution time of MapReduce jobs.

Related work

Hadoop [1] is a popular open source project. Due to the distribute features and complex mechanisms of the Map Reduce framework, optimization techniques are urgently needed by engineers, researchers and companies to improve its overall performance. Some basic configuration parameters which could be tuned for better performance were listed in [2]. Microsoft IT SES Enterprise Data Architect Team provided some suggestions for general optimization of Hadoop jobs in [3]. These suggestions help us to reach better performance of the Hadoop cluster. Early works on modeling Map Reduce framework started almost since 2008. Herodotos proposed a detailed set of mathematical performance models in [4], which described dataflow and cost information at the fine granularity of phases. A framework that offered a new resource sizing and provisioning service in Map Reduce environments was proposed by Abhishek in [5]. They focused on estimating the

resource required for processing a targeted dataset specially. In [6], an approach to model the correlation between two major MapReduce configuration parameters (number of Mappers and Reducers) and total execution time of MapReduce applications was proposed. Above approaches were all based on resource provisioning on Hadoop service providers. They assumed that the resources they can lease from service providers are unlimited. They ignored the performance bottlenecks or resource competitions which can obviously reduce the performance of Hadoop cluster. In this paper, firstly we focus on balancing the system resource utilization for different types of Map Reduce jobs to achieve the best performance. Then the execution time model is proposed on the condition that the Hadoop cluster achieves its best performance. The experimental results show that our model works well. The proposed method provides some valuable guidance for Hadoop developers and can be applied to actual cloud computing environments.

Qu, J. et al.[1] Extracting and mining social networks information from massive Web data is of both theoretical and practical significance. However, one of definite features of this task was a large scale data processing, which remained to be a great challenge that would be addressed. MapReduce is a kind of distributed programming model. Just through the implementation of map and reduce those two functions, the distributed tasks can work well. Nevertheless, this model does not directly support heterogeneous datasets processing, while heterogeneous datasets are common in Web. This article proposes a new framework which improves original MapReduce framework into a new one called Map-Reduce-Merge. It adds merge phase that can efficiently solve the problems of heterogeneous data processing. At the same time, some works of optimization and improvement are done based on the features of Web data.

Pole G, et al[2] Traditional technologies and data processing applications are inadequate for big data processing. Big Data concern very large-volume, complex formats, growing data sets with multiple, heterogeneous sources, and formats. With the reckless expansion in networking, communication, storage, and data collection capability, the big data science is rapidly growing in every engineering and science domain. Challenges in front of data scientists include different tasks, such as data capture, classification, storage, sharing, transfer, analysis, search, visualization, and decision making. This paper is aimed to discuss the need of big data analytics, journey of raw data to meaningful decision, and the different tools and technologies emerged to process the big data at different levels, to derive meaningful decisions out of it.

Rajendran, S., et al[3] The proposed model is executed in the Hadoop MapReduce environment to manage big data. Initially, the CPIO algorithm is applied to select a useful subset of features. In addition, the Harris hawks optimization (HHO)-based DBN model is derived as a classifier to allocate

appropriate class labels. The design of the HHO algorithm to tune the hyperparameters of the DBN model assists in boosting the classification performance. To examine the superiority of the presented technique, a series of simulations were performed, and the results were inspected under various dimensions. The resultant values highlighted the supremacy of the presented technique over the recent techniques.

Banchhor, C., et al[4] The CNB is developed by extending the standard naïve Bayes classifier with applied correlation among the attributes to become a dependent hypothesis. The cuckoo search and grey wolf optimization algorithms are integrated with the CNB classifier, and significant performance improvement is achieved. The resulting classifier is called a cuckoo grey wolf correlative naïve Bayes classifier (CGCNB).

Proposed methodology

The study successfully developed and implemented a MapReduce tool for data mining and data optimization using cheaply and easily available tools and procedures. The MapReduce tool was developed particularly for multi-core shared memory systems, taking advantage of current trends in parallelism. The efficiency of the tool highly depended on the granularity of the tasks executed by the worker threads and the usability of the API by maintaining a simple and restricted interface. Even if the framework were able to attain a speedup of 30%, it would still be beneficial to the researcher as it provides simple and efficient auto-parallelism. It is evident that the advantages provided by the tool under study, would clearly outweigh the disadvantages associated with it. Achievement of scalability in the clustering environment was a major milestone. This was attested to be easily achieved and sustainable since additional nodes could easily be added on the cluster without negatively affecting the performance of other nodes on the cluster hence the cluster could easily grow and increase resource utilization. The security of data is confident than usual operations since data is replicated across on all the nodes equally and in the event of destruction of one node, data can be recovered from other working nodes. The evaluation results show that the prototype exhibits satisfactory performance in the parameters such as speedup, good scaleup, and reasonable sizeup. A key observation made in the evaluation was that the input data size is an important factor in achieving reasonable speedup. This is due to the fact that for optimal parallelism, one has to ensure the effects of overheads (such as inter-machines communications) in the parallelism is minimized by the computation time; and in most cases, large input size maximizes the computation time. The unexpected speedup performance leads to a detailed analysis of the results and it suggests that, although intermachines communications and input size are factors in limiting ideal parallelism, the non-parallelizable portion in the individual computation task also plays a paramount role in

limiting speedup. The evaluation also shows that Hadoop MapReduce exhibits good scalability with growth in both data size and number of computing nodes. There is still need for continuous research in this field to develop more enhanced tools, systems and frameworks of MapReduce techniques implemented on Hadoop distributed file system in distributed systems. Further work on improving this methodology should be considered by making use of latest tools, procedures and technologies such as in the Cloud Computing to solve issues of data optimization, reliability, scalability and data security, among other issues fault tolerance in cluster computing.

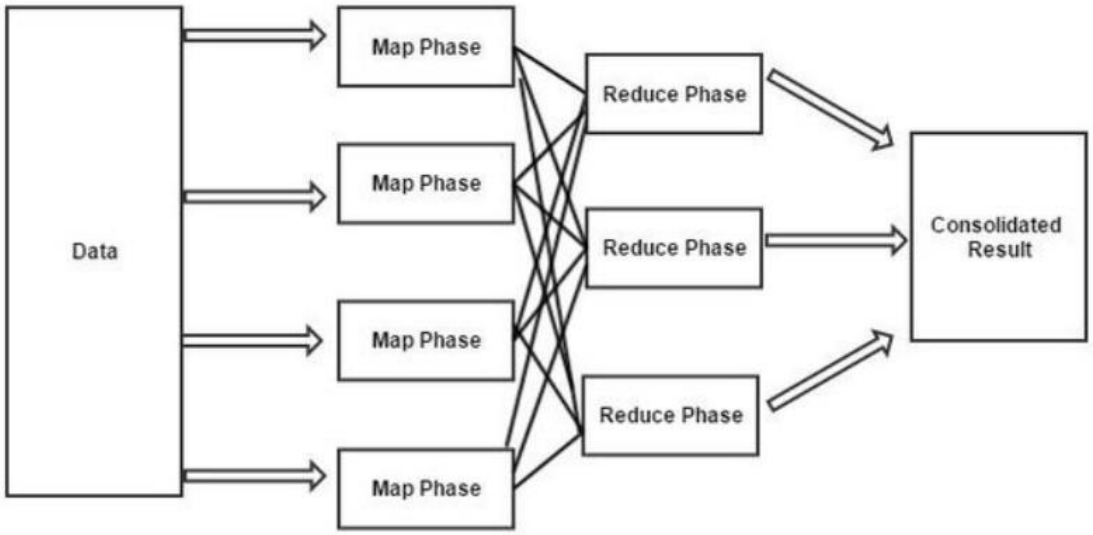

Figure 1: Optimization On Mapreduce Method

Apriori Algorithm Apriori is an iterative algorithm which alternates between the two important tasks, the first one is generation of candidates from frequent itemsets of previous iteration and the second one is scanning of database for support counting of candidates against each transaction. In kth iteration ($k \geq 2$), candidate k-itemsets Ck is generated from frequent (k-1)-itemsets Lk-1 and then k-itemset subsets of each transaction is checked against candidates in Ck for support counting. Candidate itemsets Ck is obtained by conditionally joining Lk-1 with itself and then pruning those itemsets that does not satisfy Apriori property. According to this property all the itemsets of Ck can be removed from Ck if anyone of their (k-1)-subsets does not present in Lk-1 [1].

Naïve Bayes classifier (NB)

The different classifiers based on the Bayesian theorem are known as Bayesian classifiers; for example, NB is one of the bayesian classifiers. The Bayesian classification is based on the posterior probability calculation by assumed prior probabilities and the probability of different data object under given assumptions.

The NB classifier is based on the fact that each attribute of the object to be classified is independent of each other. The NB classifier is based on the approach where the probability of each category is calculated, and the object belongs to the category with the largest probability associated.

Correlative Naive Bayes classifier (CNB)

One of the highly utilized classifiers is NB classifier, and the typical classifier is adopted with the Map-Reduce framework and used for big data classification [59]. At the initial phase of the training process, the input data are arranged in different groups based on the number of classes.

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

CNB.QXm={μQ×m,σQ×m,RQ×1}CNB.QXm={μQ×m,σQ×m,RQ×1}
(1)

where, μQ×mμQ×m is the mean to be calculated, σQ×mσQ×m is specified as variance, RQ×1RQ×1 denotes correlation function, and it is illustrated in vector form. Testing data result is represented using the following equation:

C=argMaxq=i…Q[P(Cq)×P(X |Cq)]×Rq C=argq=i…QMax[P(Cq)×P(X |Cq)]×Rq
(2)

The Eq. (2) indicates that the highest posterior value is only selected as a consequential class.

Figure 2: Correlative Naive Bayes classifier (CNB)

Hadoop Distributed File System and MapReduce

Hadoop integrates the computational power (MapReduce) with distributed storage (Hadoop Distributed File System) to reduce the communication cost by providing local access to data and local computation on data. Hadoop Distributed File System (HDFS) is designed based on GFS (Google File System) to be deployed on low-cost hardware. It breaks files in fixed size blocks (default block size is 64 MB) and replicates (default replication factor is 3) the blocks across multiple machines in cluster to provide high availability and fault tolerance [8]. MapReduce [9] is an efficient, scalable and simplified programming model for large scale distributed data processing on a large cluster of commodity computers. It process large volumes of data in parallel by breaking the job into independent tasks across a large number of machines. MapReduce program generally consists of Mapper, Combiner and Reducer tasks, which runs on all machines in a Hadoop cluster. The input and output of these functions must be in the form of (key, value) pairs [8]. Followings are the specific details of MapReduce paradigm followed in our implementations. An InputFormat class selects input file residing in HDFS, defines the InputSplit that splits the file and provides RecordReaders that reads each split for individual mapper. A single map task is a unit of work in a MapReduce program which corresponds to a single InputSplit. RecordReader reads the splitted data and converts it into (key,

value) pairs for the mapper. The Mapper takes the input (k1, v1) pairs, performs the user defined computation and produces a list of intermediate (k2, v2) pairs. These pairs are partitioned per reducer by the Partitioner class. The Reducer takes (k2, list (v2)) pairs as input, make sum of the values in list (v2) and produce new pairs (k2, v3). Each reducer writes its output in a separate file in HDFS which is directed by OutputFormat class. An optional Combiner class is used to reduce the communication cost of transferring intermediate output of mappers to reducers. It is known as mini reducer since it works locally on all the (key, value) pair emitted by the map tasks on a given node only [8].

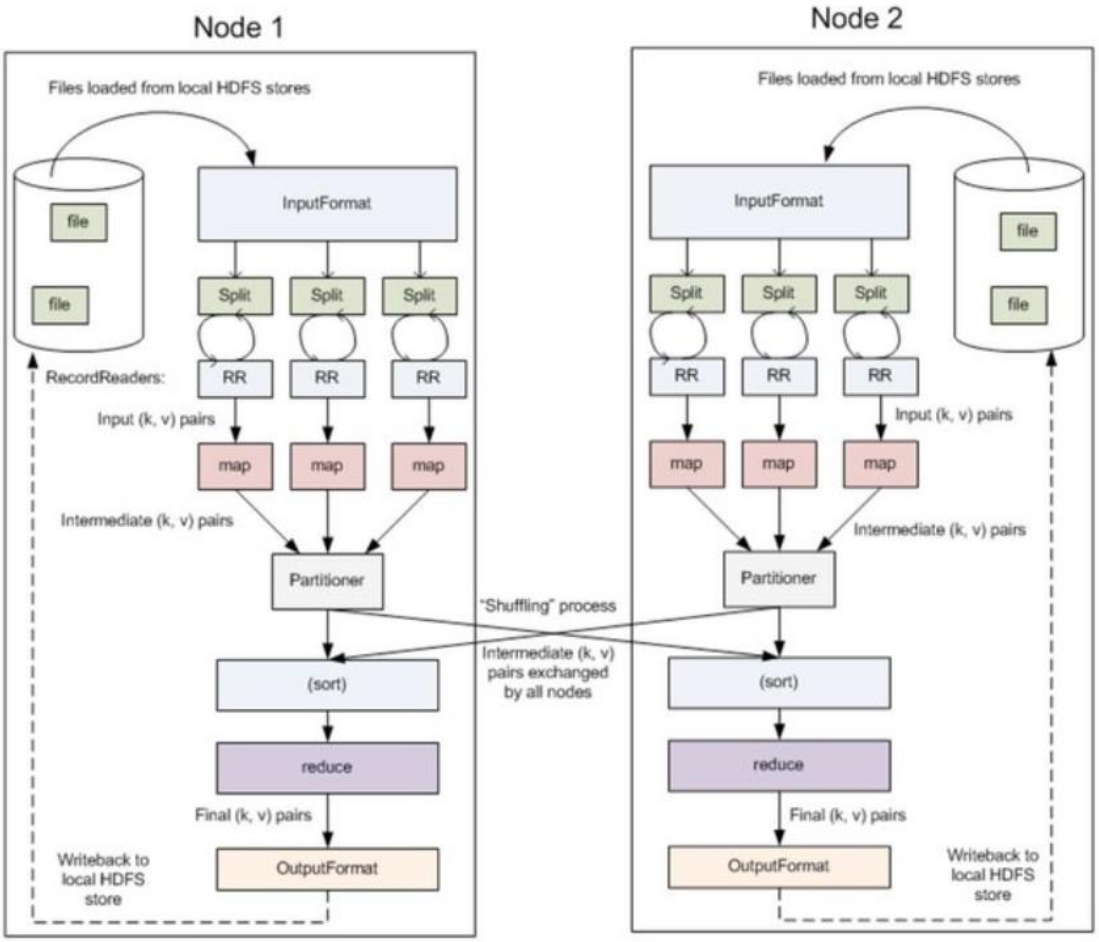

Figure 3: optimization on mapreduce method for data mining, H. Kocakulak[17]

Results Analysis

Evaluation We perform our experiments with WordCount and TeraSort. We gradually increase the input datasets as training process and analyze the log files to extract job details. The experiment results. We can see that Mavg and SelectivitYM of each workload almost keep constant. But Ravg

grows linearly with the increase of data size processed by one reduce task,. We use polyjit function in Matlab to get the scaling factors, and compute invariants of the job profile by values listed. They can be used for predicting the execution time of the same applications processing larger input datasets (e.g. 36GB, 42GB, 48GB, 54GB and 60GB). Firstly, by using the extracted job profiles and applying the and we can predict the execution time of these jobs. Then we run the jobs and collect the execution time of every job. Both predicted and measured execution time of these jobs were. We can see that the relative error between measured execution time and the predicted value is less than 10, which means that our modeling method could accurately estimate the execution time of Map Reduce workloads.

Conclusion

In this paper, we first evaluated and characterized several MapReduce workloads in terms of resource utilizations. Then we introduced some techniques that could be used to improve the performance of MapReduce workloads. Based on this condition, we proposed a novel framework that helps to estimate the execution time of MapReduce jobs. Finally, we validated the accuracy of this model by conducting several experiments. Evaluation results of two standard applications on a 4- node MapReduce cluster shows that our modeling technique can effectively predict the total execution time of these applications with the average prediction error of less than 10%. In this paper, we validated our model with WordCount and TeraSort selected from the Hadoop example package. We want to apply this model to more complex applications such as Pig queries and some others in the fields of machine learning and web searching. Additionally, the model we proposed in this paper was only based on job profiling, and we didn't dig into the details of Map Reduce framework. In the future, we will study the mechanism of MapReduce framework to find more detailed modeling techniques that could predict the execution time of a MapReduce workload more accurately.

Reference

[1]. Qu, J. , Yin, C. and Song, S. (2015) The Optimization and Improvement of MapReduce in Web Data Mining. *Journal of Software Engineering and Applications*, **8**, 395-406. doi: 10.4236/jsea.2015.88039.

[2]. Pole, G., Gera, P. (2016). A Recent Study of Emerging Tools and Technologies Boosting Big Data Analytics. In: Saini, H., Sayal, R., Rawat, S. (eds) Innovations in Computer Science and Engineering. Advances in Intelligent Systems and Computing, vol 413. Springer, Singapore. https://doi.org/10.1007/978-981-10-0419-3_4.

[3]. Rajendran, S., Khalaf, O.I., Alotaibi, Y. *et al.* MapReduce-based big data classification model using feature subset selection and hyperparameter tuned deep belief network. *Sci Rep* **11,** 24138 (2021). https://doi.org/10.1038/s41598-021-03019-y.

[4]. Banchhor, C., Srinivasu, N. Analysis of Bayesian optimization algorithms for big data classification based on Map Reduce framework. *J Big Data* **8,** 81 (2021). https://doi.org/10.1186/s40537-021-00464-4

[5]. M. A. Bhuiyan and M. Al Hasan, "An Iterative MapReduce Based Frequent Subgraph Mining Algorithm," in IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 3, pp. 608-620, 1 March 2015, doi: 10.1109/TKDE.2014.2345408.

[6]. Z. Tianrui, W. Mingqi and L. Bin, "An Efficient Parallel Mining Algorithm Representative Pattern Set of Large-Scale Itemsets in IoT," in IEEE Access, vol. 6, pp. 79162-79173, 2018, doi: 10.1109/ACCESS.2018.2884888.

[7]. M. A. Bhuiyan and M. A. Hasan, "FSM-H: Frequent Subgraph Mining Algorithm in Hadoop," 2014 IEEE International Congress on Big Data, 2014, pp. 9-16, doi: 10.1109/BigData.Congress.2014.12.

[8]. K. Chavan, P. Kulkarni, P. Ghodekar and S. N. Patil, "Frequent itemset mining for Big data," 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), 2015, pp. 1365-1368, doi: 10.1109/ICGCIoT.2015.7380679.

[9]. B. Jena, M. K. Gourisaria, S. S. Rautaray and M. Pandey, "Improvising name node performance by aggregator aided HADOOP framework," 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2016, pp. 382-388, doi: 10.1109/ICCICCT.2016.7987978.

[10]. S. Yang, B. Wang, H. Zhao and B. Wu, "Efficient Dense Structure Mining Using MapReduce," 2009 IEEE International Conference on Data Mining Workshops, 2009, pp. 332-337, doi: 10.1109/ICDMW.2009.48.

[11]. H. Chen, Q. Luo, Z. Chen and Y. Chen, "Distributed pruning optimization oriented FP-Growth method based on PSO algorithm," 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2017, pp. 1244-1248, doi: 10.1109/ITNEC.2017.8284975.

[12]. L. Kolhe, V. Khaimar and A. Kumar Jetawat, "Performance Measures Analysis Using Parallel Clustered Optimization Techniques for I-Commerce," 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2018, pp. 244-249, doi: 10.1109/CONFLUENCE.2018.8442978.

[13]. Q. Li, K. Dai, W. Wang, P. Wang, R. He and M. Dong, "Exploring Computation Locality of Graph Mining Algorithms on MapReduce," 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2013, pp. 45-50, doi: 10.1109/WI-IAT.2013.7.

[14]. A. Khoshkbarchi, A. Kamali, M. Amjadi and M. Amir Haeri, "A modified hybrid Fuzzy clustering method for big data," 2016 8th International Symposium on Telecommunications (IST), 2016, pp. 196-201, doi: 10.1109/ISTEL.2016.7881809.

[15]. Jinsong Yin and Yuanyuan Qiao, "Performance modeling and optimization of MapReduce programs," 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems, 2014, pp. 180-186, doi: 10.1109/CCIS.2014.7175726.

[16]. X. Wang, X. Liu and S. Matwin, "A distributed instance-weighted SVM algorithm on large-scale imbalanced datasets," 2014 IEEE International Conference on Big Data (Big Data), 2014, pp. 45-51, doi: 10.1109/BigData.2014.7004467.

[17]. H. Kocakulak and T. T. Temizel, "A Hadoop solution for ballistic image analysis and recognition," 2011 International Conference on High Performance Computing & Simulation, 2011, pp. 836-842, doi: 10.1109/HPCSim.2011.5999917.

[18]. A. S. Balkir, I. Foster and A. Rzhetsky, "A distributed look-up architecture for text mining applications using MapReduce," SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, 2011, pp. 1-11.

[19]. U. Kang, C. E. Tsourakakis and C. Faloutsos, "PEGASUS: A Peta-Scale Graph Mining System Implementation and Observations," 2009 Ninth IEEE International Conference on Data Mining, 2009, pp. 229-238, doi: 10.1109/ICDM.2009.14.