ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

Minimization of Network Traffic Using Partition and Aggregation for Data Stored in Cloud

P.Priyanga[1],Tushar.R.Shetty[2],Utsav.K.P[2],V.Rahul[2],Darshan.C[2].

Assistant professor,CSE department, K.S.I.T, Bengaluru, India. [1.]

B.E. Students, Department of CSE, K.S.I.T.Bengaluru, India[2]

ABSTRACT: The MapReduce programming model simplifies large-scale data processing on commodity cluster by exploiting parallel map tasks and reduce tasks. Although many efforts have been made to improve the performance of MapReduce jobs, ignore the network traffic generated in the shuffle phase, which plays a critical role in performance enhancement. Traditionally, a hash function is used to partition intermediate data among reduce tasks, which, however, is not traffic-efficient because network topology and data size associated with each key are not taken into consideration. In this paper, studying about how to reduce network traffic cost for a MapReduce job by designing a novel intermediate data partition scheme. Furthermore, consider the aggregator placement problem, where each aggregator can reduce merged traffic from multiple map tasks. A decomposition-based distributed algorithm is proposed to deal with the large-scale optimization problem for big data application and an online algorithm is also designed to adjust data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that proposals can significantly reduce network traffic cost under both offline and online cases.

KEYWORDS: MapReduce, partition, aggregation, big data, lagrangian decomposition.

I. INTRODUCTION

The term big data is regarded as an umbrella which includes everything from digital data to health data. Big data has evolved from various steps starting from primitive and structured data to complex relational data and now very complex and unstructured data[1][2]. The big data is defined as datasets whose size is beyond the ability of typical database tools to store, capture, manage and analyze. MapReduce has emerged as the most popular computing framework for big data processing due to its complete programming model and automatic management of parallel execution. MapReduce[3] divides a computation into two important phases, namely map and reduce, which in turn are carried out by several maps[4] tasks and reduce tasks, correspondingly. In the map phase, map tasks are launched in parallel to convert the original input splits into in-between data in a form of Key/value pairs[5]. These Key/value pairs are stored on local machine and prearranged into multiple data partition. In the reduce phase, each reduces task fetches its own data partition from all maps to generate the final result[7]. There is a shuffle step between map and decrease phase. In this step, the data produced by the map phase are ordered, divided and transferred to the appropriate machines executing the reduce phase. The resulting network traffic pattern from all map tasks to all reduce tasks can cause a great volume of network traffic, imposing a serious constraint on the efficiency of data analytic applications. For example, with tens of thousands of machines, data shuffling [8] accounts for 58.6% of the cross-pod traffic [9] and amounts to over 200 petabytes in total in the analysis of SCOPES jobs. For shuffle-heavy [10] MapReduce tasks, the high traffic could incur considerable recital overhead up to 30-40%[11] .

II. RELATED WORK

In the big data analytics community has favored MapReduce as a programming model for processing massive data on distributed systems such as a Hadoop cluster. MapReduce has been developing to improve its performance. We

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

identified skewed workload among workers in the MapReduce ecosystem. The issue of skewed workload is of serious concern for massive data processing. We tackled the workload balancing problem by introducing a hierarchical MapReduce, or h-MapReduce for short. h-MapReduce identifies a heavy task by a correctly defined cost function. The heavy task is divided into child tasks that are distributed among accessible workers as a new job in MapReduce framework. The invocation of new jobs from a task poses several challenges that are addressed by h-MapReduce. Their experiments on h-MapReduce proved the performance gain over standard MapReduce for data-intensive algorithms. More exclusively, the increase of the performance gain is exponential in terms of the size of the networks. Along with the exponential performance gains, our investigations also found a negative effect of deploying h-MapReduce due to an incorrect definition of heavy tasks, which provides us a guideline for an effective application of h-MapReduce.

This research unfolds an advanced two-phase MapReduce solution that is able to efficiently address skyline queries on large datasets. Unlike existing parallel skyline avenues, our scheme considers data partitioning, filtering, and parallel skyline evaluation as a holistic query process. In finical, we apply filtering techniques and angle-based partitioning in the first phase, in which unqualified objects are rejected and the processed objects are divided by their angles to the origin. In the second phase, local skyline objects in each partition are calculated in parallel, and global skyline objects are output after a combining skyline process. To improve the parallel local skyline calculation, we propose two partition-aware filtering techniques that keep skyline candidates in a balanced manner. The aggressive partition-aware filtering aggressively excludes objects in the partition with the greatest population of candidate objects, whereas the proportional partition-aware filtering delays down the growth of partition population proportionally. Recognizing the absence of studies that incorporate the MapReduce framework into parallel skyline processing, we propose a partial report grid-based partition skyline algorithm that is capable to significantly improve the merging skyline computation on large datasets. The pre-sort process can be finished in the shuffle phase with little overhead. Our experimental results show the competence and effectiveness of the proposed parallel skyline solution utilizing MapReduce on large-scale datasets.

Data clustering is a significant data mining technology that plays a crucial role in numerous scientific applications. However, it is interesting to cluster big data as the size of datasets has been growing rapidly to extra-large scale in the real world. Meanwhile, R. Dhanalakshmi in his paper he says MapReduce is a desirable parallel programming platform that is widely applied in data processing fields. Also, it generates more accurate clusters than both K-Means MapReduce algorithm and DBSCAN MapReduce-Algorithm. A hybrid approach based on parallel K-Means and parallel DBSCAN is proposed to overcome the disadvantages of both these algorithms. This approach allows combining the benefits of both the clustering techniques.

III. PROPOSED ALGORITHM

Design Considerations: Most existing work concentrates on MapReduce performance improvement by optimizing its data transmission. Narayan et al. Have explored the use of Open Flow to provide betterlink bandwidth for shuffle traffic. Palanisamy et al.have presented Purlieus, a MapReduce resource allocation system, to improve the performance of MapReduce jobs in the cloud by finding intermediate data to the local machines otherwise close-by physical machines. This locality awareness reduces network traffic in the shuffle phase produced in the cloud data center. However, little work has studied to optimize network performance of the shuffle process that generates large amounts of data traffic in MapReduce jobs. A critical factor to the network performance in the shuffle phase is the intermediate data partition. The default scheme adopted by Hadoop is hash-based the partition that would yield unbalanced loads among reducing tasks due to its unawareness of the data size associated with each key. To prevail this shortcoming, Ibrahim et al.have developed a fairness-aware key partition approach that keeps track of the spreading of intermediate keys ‘frequencies, and guarantees a fair distribution among reduce tasks. Meanwhile, Yan et al. have introduced ask etch-based data structure for capturing MapReduce key group size statistics and existed an optimal packing algorithm which allocates the key groups to the reducers in a load balancing manner. Hsueh et al. have proposed and assessed two effective load balancing approaches to data skew managing for MapReduce-based entity resolution. Unfortunately, all above work efforts on load balance at reducing tasks, ignoring the network traffic through the shuffle phase.

In addition to data partition, many efforts have been made on local aggregation, in-mapper uniting and in network aggregation to reduce network traffic within Map-Reduce jobs. Condie et al. have introduced a combiner function that

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

decreases the amount of data to be shuffled and merged to reduce tasks. Lin and Dyer have proposed an in-mapper uniting scheme by exploiting the fact that mappers can preserve state across the processing of multiple input key/value pairs and delay emission of intermediate data until all input records have been processed. Both proposals are forced to a single map task, ignoring the data aggregation opportunities from multiple map tasks. Costa et al. have proposed a MapReduc Fig 1.like system to decrease the traffic by pushing aggregation from the edge into the network. However, it can be only functional to the network topology with servers directly linked to other servers, which is of restricted practical use. Different from existing work, we investigate network traffic reduction within MapReduce jobs by jointly misusing traffic-aware intermediate data partition and data aggregation among multiple map tasks.

we verify that our distributed algorithm can be applied in practice using real trace in a cluster consisting of five virtual machines with 1 GB memory and 2 GHzCPU. Our network topology is based on three-tier architectures.

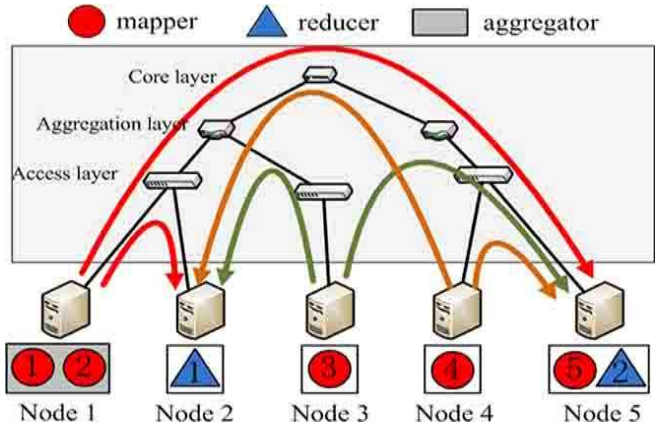

Fig 1. MapReduce technique

IV. PSEUDO CODE

STEP 1: GET THE MESSAGE

STEP 2: CONVERT MESSAGE IN TO THE BITS.

STEP 3: APPEND PADDING BITS (MAKE THE MESSAGE BIT LENGTH SHOULD BE THE EXACT MULTIPLE OF 512 BITS AS WELL AS 16 WORD BLOCKS).

STEP 4: DIVIDE TOTAL BITS IN TO 128 BITS BLOCKS EACH

STEP 5: INITIALIZE MD BUFFER.

A FOUR WORD BUFFER (A,B,C,D) IS USED TO COMPUTE THE MESSAGE DIGEST , TOTAL 128 BITS.

STEP 6: DO AND,XOR,OR,NOT OPERATIONS ON A,B,C,D BY GIVING THREE INPUTS AND GET ONE OUTPUT.

STEP 7: DO THE STEP 6 UNTIL GET THE 128 BITS HASH(16 BYTES).

STEP 8: STOP

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

V. SIMULATION RESULTS

The simulation studies involve the deterministic small network topology first evaluate the performance gap between our proposed distributed algorithm and the optimal solution obtained by solving the MILP formulation. Due to the high computational complexity of the MILP formulation, we consider small-scale problem instances with 10 keys in this set of simulations. Each key associated with random data size within [1-50]. There are 20 mappers, and 2 reducers on a cluster of 20 machines. The parameter α is set to 0.5. The distance between any two machines is randomly chosen within [1-60]. As shown in Fig. 2, the performance of our distributed algorithm is very close to the optimal solution. Although network traffic cost increases as the number of keys grows for all algorithms, the performance enhancement of our proposed algorithms to the other two schemes becomes larger. When the number of keys is set to 10, the default algorithm HNA has a cost of 5.0 × 104 while optimal solution is only 2.7×104 , with 46% traffic reduction.

As shown in Fig. 3, the network traffic cost shows as an increasing function of number of keys from 1 to 100 under all algorithms. In particular, when the number of keys is set to 100, the network traffic of the HNA algorithm is about 3.4 × 105 , while the traffic cost of our algorithm is only 1.7 × 105 , with a reduction of 50%. In contrast to HRA and HNA, the curve of DA increases slowly because most map outputs are aggregated and traffic-aware partition chooses closer reduce tasks for each key/value pair, which are beneficial to network traffic reduction in the shuffle phase.

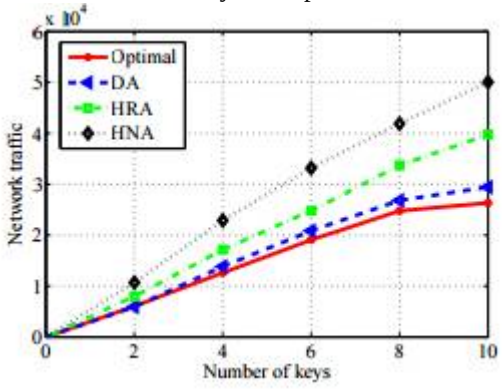

Fig. 3.Network traffic cost versus number of keys from 1to 10

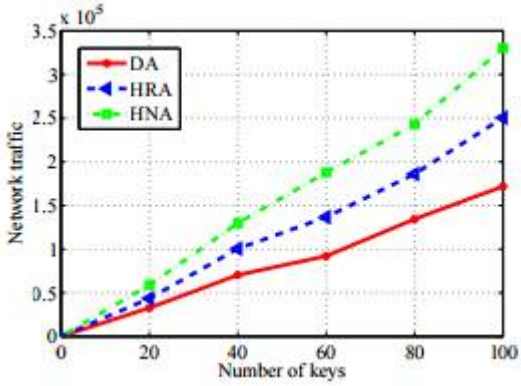

Fig 4. Network traffic cost versus different number of keys from 1to100

VI. CONCLUSION AND FUTURE WORK

In this paper, studying the joint optimization of intermediate data partition and aggregation in MapReduce to minimize network traffic cost for big data applications. We propose a three-layer model for this problem and formulate it as a mixed-integer nonlinear problem, which is then transferred into a linear form that can be solved by mathematical tools. To deal with the large-scale formulation due to big data, we design a distributed algorithm to solve the problem on multiple machines. Furthermore, we extend our algorithm to handle the MapReduce job in an online manner when some system parameters are not given. Finally, conduct extensive simulations to evaluate our proposed algorithm under both offline cases and online cases. The simulation results demonstrate that our proposals can effectively reduce network traffic cost under various network settings.

REFERENCES

1. Huan Ke , Peng Li and Minyi Guo "OnTraffic-Aware PartitionandAggregationin MapReduce for Big Data Applications" IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 27, NO. 3, MARCH 201
2. Venkata Swamy Martha; Weizhong Zhao; Xiaowei Xu Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference
3. Ji Zhang; Xunfei Jiang; Wei-Shinn Ku; Xiao QinIEEE Transactions on Parallel and Distributed SystemsYearEfficient Parallel Skyline Evaluation Using MapReduc: 2016, Volume: 27, Issue: 7
4. R.Dhanalakshmi,S.Mohamed Jakkariya, S. MangaiarkarasiInternational Journal of Innovative Research in Computer and Communication EngineeringVol. 4, Issue 2, February 2016

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

5. Y. Dinesh Reddy and A. Pio Sajin Indian Journal of Science and Technology, Vol 9(10), DOI: 10.17485/ijst/2016/v9i10/88981, March 2016
6. S.Deepa, K.Dhanusha, K.RevathiOn Traffic-Aware Partition and Aggregation in MapReduce for Big Data Applications IJMTES International Journal of Modern Trends in Engineering and Science
7. D. Arul Selve, Dr. K. KavithaInternational Journal of Advanced Research in Computer Engineering & Technology(IJARCET)Volume 5, Issue 4, April2016
8. LiyaFan, Bo Gao,XiSun ,FaZhang ,Zhiyong Liu 1 IBM China Research Laboratory 2 Institute of Computing Technology,Chinese Academy of Sciences Beijing China
9. R Bahirathy et al, International Journal of Computer Science & Communication Networks,Vol 6(2),54-60IJCSCN April-May 2016 ISSN:2249-5789
10. .Priya Gawande and Nuzhaft Shaikh International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT) - 2016978-1-4673-9939-5/16/$31.00 ©2016 IEEEImproving Network Traffic in MapReduce f or Big Data Applications
11. Gomathipriya R,Janani E,Rajesh P International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)Vol. 2, Special Issue 10, March 2016
12. J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters,"Commun. ACM, vol. 51, no. 1, pp. 107–113,2008.
13. W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "Map task scheduling in mapreduce with data locality: Throughput and heavy- traffic optimality," in Proc. IEEE INFOCOM , 2013, pp. 1609–1617.
14. F. Chen, M. Kodialam, and T. Lakshman, "Joint scheduling of processing and shuffle phases in mapreduce systems," in Proc.IEEE INFOCOM, 2012, pp. 1143–1151.
15. Y. Wang, W. Wang, C. Ma, and D. Meng, "Zput: A speedy datauploading approach for the hadoop distributed file system," inProc. IEEE Int. Conf. Cluster Comput., 2013, pp. 1–5.