International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Big Data Hadoop: Aggregation Techniques

Vidya Pol

Department of Computer Engineering, KJ College of Engineering Management & Research, Savitribai Phule University, Pune, India

Abstract: ***The term 'Big Data', refers to data sets whose size (volume), complexity (variability), and rate of growth (velocity) make them difficult to capture, manage, process or analyzed. To analyze this enormous amount of data Hadoop can be used. However, processing is often time-consuming. One way to decrease response time is to executing the job partially, where an approximate, early result becomes available to the user, before completion of job. The implementation of the technique will be on top of Hadoop which will help to sample HDFS blocks uniformly. We will evaluate this technique using real-world datasets and applications and we will try to demonstrate the system's performance in terms of accuracy and time. The objective of the proposed technique is to significantly improve the performance of Hadoop MapReduce for efficient Big Data processing.***

Keywords: privacy preservation, security, e-healthcare systems, data mining, image feature extraction.

1. Introduction

Big data is a term that refers to data sets or combinations of data sets whose size (volume), complexity (variability), and rate of growth (velocity) make them difficult to be captured, managed, processed or analyzed by conventional technologies and tools, such as relational databases. Hadoop MapReduce programming model is being used for processing Big Data, which consists of data processing functions: Map and Reduce. Parallel Map tasks are run on input data which is partitioned into fixed sized blocks and produce intermediate output as a collection of <key, value> pairs. These pairs are shuffled across different reduce tasks based on <key, value> pairs. Each Reduce task accepts only one key at a time and process data for that key and outputs the results as <key, value> pairs. The Hadoop MapReduce architecture consists of one JobTracker (Master) and many TaskTrackers (Workers). The MapReduce Online is a modified version of Hadoop MapReduce which supports Online Aggregation and reduces response time. Traditional Map Reduce implementations materialize the intermediate results of mapper and do not allow pipelining between the map and the reduce phases. This approach has the advantage of simple recovery in the case of failures, however, reducers cannot start executing tasks before all mapper have finished. This limitation lowers resource utilization and leads to inefficient execution for many applications. The main motivation of Map Reduce Online is to overcome these problems, by allowing pipelining between operators, while preserving

Fault tolerance guarantees. Redis is an open-source, networked, in-memory, key-value data store with optional durability. It is written in ANSI C.

The name Redis means REmote DIctionary Server. In its outer layer, the Redis data model is a dictionary which maps keys to values. One of the main differences between Redis and other structured storage systems is that Redis supports not only strings, but also abstract data types like lists of strings, sets of strings (collections of non-repeating unsorted elements), sorted sets of strings (collections of non-repeating elements ordered by a floating-point number called score), hashes where keys and values are strings. The type of a value determines what operations (called commands) are available for the value itself. Redis supports high-level, atomic, server-side operations like intersection, union, and difference between sets and sorting of lists, sets and sorted sets. The main goal of the project work is to implement Online MapReduce and Redis on the top of the Hadoop, which will improve the performance of Hadoop for efficient Big Data processing.

2. Related Work

Most existing work focuses on MapReduce performance improvement by optimizing its data transmission. Blancaetal have investigated the question of whether optimizing network usage can lead to better systemperformance and found that high network utilization and low network congestion should be achieved simultaneously for a job with good performance. Palanisamyetal have presented Purlieus, a MapReduce resourceallocation system, to enhance the performance of MapReduce jobs in the cloud by locating intermediatedata to the local machines or close-by physical machines.This locality-awareness reduces network trafficin the shuffle phase generated in the cloud data center.However, little work has studied to optimize networkperformance of the shuffle process that generates largeamounts of data traffic in MapReduce jobs. A criticalfactor to the network performance in the shuffle phaseis the intermediate data partition. The default schemeadopted by Hadoop is hash-based partition that would yield unbalanced loads among reduce tasks due to itsunawareness of the data size associated with each key.To overcome this shortcoming, Ibrahietal havedeveloped a fairness-aware key partition approach that keeps track of the distribution of intermediate keys'frequencies, and guarantees a fair distribution among reduce tasks. Meanwhile, Liya etal have designedan algorithm to schedule operations based on the key distribution of intermediate key/value pairs to improvethe load balance. Larsetal have proposed and evaluated two effective load balancing approaches to dataskew handling for MapReduce-based entity resolution.Unfortunately, all above work focuses on load balanceat reduce tasks, ignoring the network traffic during the shuffle phase.In addition to data partition, many efforts have been made on local aggregation, in-mapper combining and in-network aggregation to reduce network traffic withinMapReduce jobs. Condieetal have introduced a combiner function that reduces the amount of data tobe shuffled and merged to reduce tasks. Lin and Dyer have proposed an in-mapper combining scheme byexploiting the fact that mappers can preserve state across the processing of multiple input key/value pairs and defer emission of intermediate data until all input records have been processed. Both proposals are constrainedto a single map task, ignoring the data aggregation opportunities from multiple map tasks. Costaetal have proposed a MapReduce-like system to decrease the traffic by pushing aggregation from the edge into thenetwork. However, it can be only applied to the network topology with servers directly linked to other servers,which is of limited practical use. Different from existing work, we investigate network traffic reduction within MapReduce jobs by jointly exploiting traffic-aware intermediate data partition and data aggregation among multiple map tasks

3. Proposed Work

The Map Reduce Online is a modified version of Hadoop Map Reduce, a popular open-source implementation of the Map Reduce programming model. It supportsOnline Aggregation and stream processing, while also improving utilization and reducing response time. Traditional Map Reduce implementations materialize theintermediate results of mappers and do not allow pipelining between the map and thereduce phases. This approach has the advantage of simple recovery in the case of failures, however, reducers cannot start executing tasks before all mappers have finished.This limitation lowers resource utilization and leads to inefficient execution for manyapplications. The main motivation of Map Reduce Online is to overcome theseproblems, by allowing pipelining between operators, while preserving fault-toleranceguarantees. Although MapReduce was originally designed as a batch oriented system, it is often used for interactive data analysis: a user submits a job to extractinformation from a data set, and then waits to view the results before proceedingwith the next step in the data analysis process. This trend has accelerated with thedevelopment of high-level query languages that are executed as MapReduce jobs, suchas Hive, Pig. Traditional MapReduce implementations provide a poor interface forinteractive data analysis, because they do not emit any output until the job has beenexecuted to completion In many cases, an interactive user would prefer a quickand dirtyapproximationover acorrectanswer that takes much longer to compute. In the database literature, online aggregation has been proposed to address this problem,but the batch-oriented nature of traditional MapReduceimplementations makes these techniques difficult to apply.

4. Simulation Results

We first evaluate the performance gap between ourproposed distributed algorithm and the optimal solutionobtained by solving the MILP formulation. Due to thehigh computational complexity of the MILP formulation,we consider small-scale problem instances with 10 keysin this set of simulations. Each key associated with randomdata size within [1-50]. There are 20 mappers, and2 reducers on a cluster of 20 machines. The parameter is set to 0.5. The distance between any two machines israndomly chosen within [1-60].As shown in Fig.1, the performance of our distributedalgorithm is very close to the optimal solution. Althoughnetwork traffic cost increases as the number of keysgrows for all algorithms, the performance enhancementof our proposed algorithms to the other two schemesbecomes larger. When the number of keys is set to10, the default algorithm HNA has a cost of 5.0 × 10^4while optimal solution is only 2.7×10^4, with 46% trafficreduction.We then consider large-scale problem instances, andcompare the performance of our distributed algorithmwith the other two schemes. We first describe a defaultsimulation setting with a number of parameters, andthen study the performance by changing one parameter while fixing others.

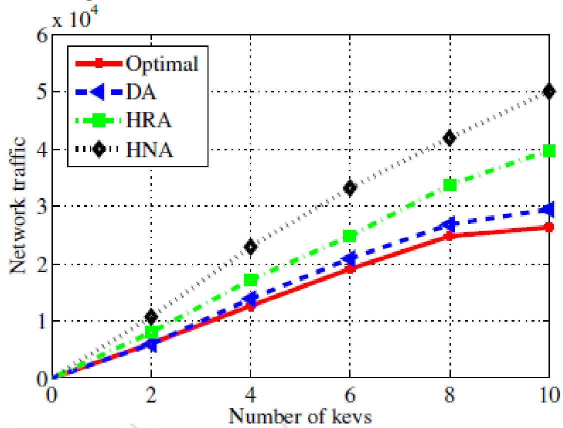

Figure 1: Network traffic cost versus number of keys from 1 to 10

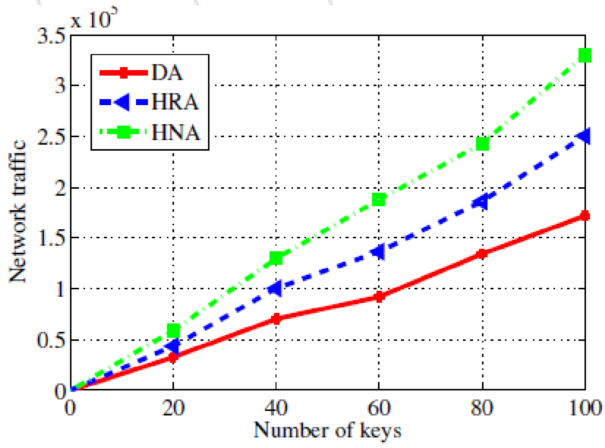

Figure 2: Network traffic cost versus different number of keys

We consider a MapReduce job with100 keys and other parameters are the same above.As shown in Fig. 3, the network traffic cost shows asan increasing function of number of keys from 1 to 100under all algorithms. In particular, when the numberof keys is set to 100, the network traffic of the HNAalgorithm is about 3.4×10^5, while the traffic cost of ouralgorithm is only 1.7 $\times10^5$, with a reduction of 50%. Incontrast to HRA and HNA, the curve of DA increasesslowly because most map outputs are aggregated andtraffic-aware partition chooses closer reduce tasks foreach key/value pair, which are beneficial to networktraffic reduction in the shuffle phase.We then study the performance of three algorithmsunder different values of α in Fig. 4 by changing itsvalue from 0.2 to 1.0. A small value of α indicates alower aggregation efficiency for the intermediate data.We observe that network traffic increases as the growthof under both DA and HRA. In particular, when α is 0.2, DA achieves the lowest traffic cost of 1.1 × 10^5.On the other hand, network traffic of HNA keeps stablebecause it does not conduct data aggregation.The affect of available aggregator number on networktraffic is investigated in Fig. 5. We change aggregatornumber from 0 to 6, and observe that DA alwaysoutperforms other two algorithms, and network traffics

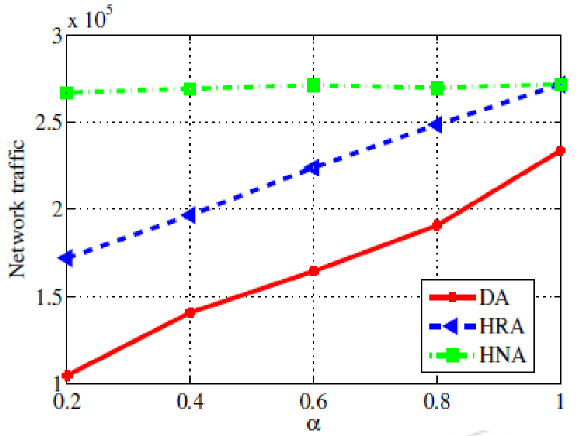

Figure 3: Network traffic cost versus data reduction ratio α

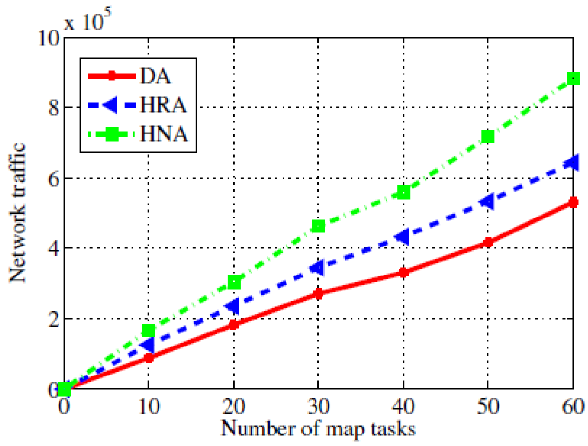

Figure 6: Network traffic cost versus number of map tasks

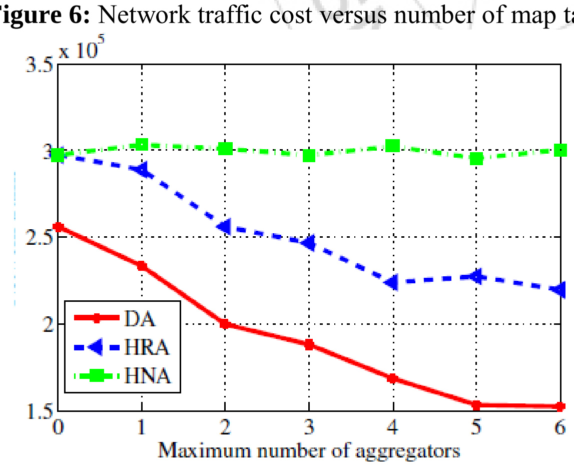

Figure 4: Network traffic cost versus number of aggregators.

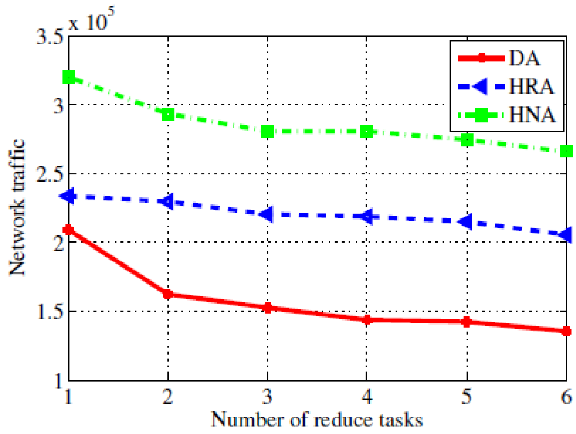

Figure 5: Network traffic cost versus number of reduce tasks

Decrease under both HRA and DA. Especially, when thenumber of aggregator is 6, network traffic of the HRAalgorithm is 2.2×10^5, while of DA's cost is only 1.5×10^5,with 26.7% improvement. That is because aggregatorsare beneficial to intermediate data reduction in the shuffleprocess. Similar with Fig. 4, the performance of HNAshows as a horizontal line because it is not affected byavailable aggregator number.We study the influence of different number of maptasks by increasing the mapper number from 0 to 60. Asshown in Fig. 5, we observe that DA always achievesthe lowest traffic cost as we expected because it jointlyoptimizes data partition and aggregation. Moreover, asthe mapper number increases, network traffic of allalgorithms increases.We shows the network traffic cost under differentnumber of reduce tasks in Fig. 6. The number of reducersis changed from 1 to 6. We observe that the highestnetwork traffic is achieved when there is only one reducetask under all algorithms. That is because all key/valuepairs may be delivered to the only reducer that locatesfar away, leading to a large amount of network trafficdue to the many-to-one communication pattern. As thenumber of reduce tasks increases, the network trafficdecreases because more reduce tasks share the loadof intermediate data. Especially, DA assigns key/valuepairs to the closest reduce task, leading to least network traffic.

5. Conclusion

The proposed system is based on implementation of Online Aggregation ofMapReduce in Hadoop for ancient big data processing. Traditional Map Reduceimplementations materialize the intermediate results of mappers and do not allowpipelining between the map and the reduce phases. This approach has the advantageof simple recovery in the case of failures, however, reducers cannot start executing tasks before all mappers have finished. As the Map Reduce Online is a modeled version of Hadoop Map Reduce, it supports Online Aggregation and stream processing,while also improving utilization and reducing response time. The limitation of traditional mapreduces lowers resource utilization and leads to incident execution formany applications. The main motivation of Map Reduce Online is to overcome theseproblems, by allowing pipelining between operators.

References

[1] S. Vikram Phaneendra & E. Madhusudhan Reddy Big Data- solutions for RDBMSproblems- A survey In 12th IEEE/IFIP Network Operations & Management Symposium (NOMS 2010) (Osaka, Japan, Apr 1923 2013)
[2] Kiran kumara Reddi & Dnvsl Indira Di_erent Technique to Transfer Big Data:survey IEEE Transactions on 52(8) (Aug.2013) 2348 2355
[3] Jimmy Lin MapReduce Is Good Enough? The control project. IEEE Computer32 (2013).
[4] Hongfei Li, Usage analysis for smart meter management in Proc of 2011 IEEEConference.
[5] Daswin De Silva, XinghuoYu,DammindaAlahakoon, and Grahame Holmes, A DataMining Framework for Electricity Consumption Analysis From Meter Data IEEETrans.on Ind. Informatics, vol. 7, no. 3.
[6] Yang Wang,, Qing Xia, Chongqing Kang, Secondary Forecasting Based on Deviation Analysis for Short-Term Load Forecasting IEEE Trans..on Power Systems, vol.26, no.2.
[7] XindongWu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, HiroshiMotoda , Geo_rey J.McLachlan, Angus Ng, Bing Liu, Philip S. Yu, ZhiHua Zhou,Michael Steinbach, David J. Hand, Dan Steinberg, "Top 10 algorithms in data min-ing", KnowlInfSyst, 2008 14, pp. 1-37.
[8] Jiawei Han and MichelineKamber, Classification and Prediction inData Mining:Concepts and Techniques 2nd ed., San Francisco, CA The Morgan Kaufmann, 2006.
[9] http://www.nyiso.com/public/markets operations/market data/load data/index.jsp
[10] Report from Pike research, http://www.pikeresearch.com/research/smartgrid-dataanalytics.
[11] National Climate Data Center [Online]. Available:http://www.ncdc.noaa.gov/oa/ncdc.html

Author Profile

Vidya Vasant Polis pursuing her Masters of Engineering in the Computer Networks, Computer Department, KJ Collegeof Engineering Management & Research, and Savitribai Phule University. She received Bachelor of Engineering degree inInformationTechnology from University Of Pune, Pune, India.