AENSI Journals

Australian Journal of Basic and Applied Sciences

ISSN:1991-8178

Journal home page: www.ajbasweb.com

Partition by Percentage using MapReduce in Hadoop

[1]Gothai Ekambaram and [2]Dr. Balasubramanie Palanisamy

[1]*Associate Professor, Department of CSE, Kongu Engineering College, Perundurai-638052, Tamilnadu, India*
[2]*Professor, Department of CSE, Kongu Engineering College, Perundurai-638052, Tamilnadu, India*

ARTICLE INFO

Article history:
Received 19 August 2014
Received in revised form
19 September 2014
Accepted 29 September 2014
Available online 8 November 2014

Keywords:
Hadoop, MapReduce, Percentage Partitioning, Big-Data

ABSTRACT

The Hadoop Distributed File System (HDFS) is designed to store very large data sets unfailingly and to stream those datasets at extraordinary bandwidth to end user applications. A significant characteristic of Hadoop is the partitioning of data and computation through many thousands of hosts and the implementation of application computations in parallel close to their data. MapReduce is a programming model and software structure first established by Google with an idea to enable and streamline the processing of huge amounts of data in parallel. Since Hadoop is using key partitioning as a only partitioning, the authors proposed an better partitioning algorithm using Percentage rather than Key which improves load balancing and memory consumption. This is completed through an enhanced sampling algorithm and partitioner. To estimate the proposed algorithm, its performance was compared against some existing partitioning mechanisms. Experiments show that the proposed algorithm is more memory efficient, faster and more correct than the current implementation in some parameters.

© 2014 AENSI Publisher All rights reserved.

To Cite This Article: Gothai Ekambaram and Dr. Balasubramanie Palanisamy, Partition by Percentage using MapReduce in Hadoop. ***Aust. J. Basic & Appl. Sci.,*** *8(17): 131-136, 2014*

INTRODUCTION

Hadoop provides a distributed file system and a structure for the analysis and conversion of very large datasets using the MapReduce paradigm by Dean and Ghemawat (2004). Correctness to standards was forfeited in favor of better performance for the applications at hand, while the interface to HDFS is patterned after the Unix file system. A significant characteristic of Hadoop is the partitioning of data and computation across many thousands of hosts and the execution of application computations in parallel close to their data.

The peopleat Google implemented hundreds of special purpose computations whichprocess large amounts of raw data such as crawled documents and Webrequest logs etc. to calculate various classes of derived data such asnumerous representations of the graph structure of Web documents, inverted indices, summaries of the number of pages crawled per host andthe set of most frequent queries in a given day, Over the past five years. Most such computations are abstractly straightforward. However the input data is habitually large and the computations have to be dispersed across hundredsor thousands of machines in order to complete in a reasonable amount of time. The concerns of how the computation parallelized, data distributed and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

MATERIALS AND METHODS

Bigtable is a distributed storage system for managing organized data that is designed to scale to a very large size. Many projects at Google store data in Bigtable such as Google Earth, Google Finance and web indexing. These applications place very different demands on Bigtable both in terms of data size and latency requirements. In spite of these different hassles, Bigtable has effectively provided a flexible and high-performance solution for all of these Google products. Chang *et al* (2006) termed the simple data model provided by Bigtable which gives clients dynamic control over data layout and format. Also they described the design and implementation of Bigtable.

MapReduce is a programming paradigm for parallel processing that is increasingly being used for data intensive applications in cloud computing surroundings. An indulgent of the characteristics of workloads

Corresponding Author: Gothai Ekambaran, Associate Professor, Department of CSE, Kongu Engineering College, Perundurai – 638108, Tamilnadu, India.
Tel: +91 98427 26627, E-mail: kothaie@yahoo.co.in

Australian Journal of Basic and Applied Sciences, 8(17) November 2014, Pages: 131-136

running in MapReduce environments benefits both the service providers in the cloud and users: the service provider can use this knowledge to make better scheduling decisions while the user can learn what aspects of their jobs impact performance. Kavulya *et al* (2010) analyzed 10-months of MapReduce logs from the M45 supercomputing cluster which Yahoo! made freely available to select universities for systems research. They characterized sources of failures, resource utilization patterns and job patterns. They used an instance based learning method that exploits chronological locality to forecast job completion times from historical data and identify potential performance problems in our dataset.

Kenn *et al* (2013) presented extended partitioning techniques XTrie and ETrie to improve load balancing for distributed applications. MapReduce programs can become more efficient at handling tasks by reducing the overall computation time spent processing data on each node by improving load balancing. The TeraSort developed by O'Malley Yahoo was designed based on randomly generated input data on a very large cluster. In that particular computing environment and for that data configuration each partition generated by MapReduce appeared on only one or two nodes. In contrast their study looks at small- to medium-sized clusters. This study modifies their design and optimizes it for a smaller environment. A series of experiments have shown that the ETrie architecture was able to allocate more computing resources on average, conserve more memory and do so with less time complexity for given skewed data sample.

Hsu and Chen (2012) introduced a research model and two methods to derive new lists of processor logical id according to the characteristics of heterogeneous network. Both methods provided choices of more low-cost communication schedules in grid. The simulations showed both proposed methods yield outstanding performance in grid.

Fadika and Govindaraju (2011) introduced the concept of dynamic elasticity to MapReduce. They presented the design decisions and implementation tradeoffs for DELMA an acronym for Dynamically Elastic MapReduce which is a framework that follows the MapReduce paradigm just like Hadoop MapReduce but that is capable of growing and shrinking its cluster size as jobs are underway. They tested DELMA in diverse performance scenarios ranging from diverse node additions to node additions at various points in the application run-time with various dataset sizes in their study. They concluded on the benefits of providing MapReduce with the capability of dynamically growing and shrinking its cluster configuration by adding and removing nodes during jobs and explain the possibilities presented by this model.

Jiang and Agarwal (2011) described a system Extended MATE or Ex-MATE which supports alternate API with reduction objects of arbitrary sizes. They develop support for managing disk-resident reduction objects and updating them efficiently. They evaluated their system using three graph mining applications and compare its performance to that of PEGASUS which is a graph mining system implemented based on the original map-reduce API and its Hadoop implementation. Their results on a cluster with 128 cores how that for all three applications their system outperforms PEGASUS by factors ranging between 9 and 35.

Liu and Orban (2011,described Cloud MapReduceie.CMR which implements the MapReduce programming model on top of the Amazon cloud Operating System. Cloud MapReduce is a demonstration that it is possible to overcome the cloud limitations and simplify system design and implementation by building on top of a cloud OS. They described how they overcome the limitations presented by horizontal scaling and the weaker consistency guarantee. Their experimental results show that CMR runs faster than Hadoop another implementation of MapReduce and that CMR is a practical system. They believed that the techniques they used are general enough that they can be used to build other systems on top of a cloud OS.

Kenn *et al* (2013) mentioned that the time required to process a MapReduce job is dependent on whichever node is last to complete a task. This problem is impaired by assorted environments. So they proposed a method to improve MapReduce execution in heterogeneous environments. This was done by dynamically partitioning data during the Map phase and by using virtual machine mapping in the Reduce phase in order to maximize resource utilization.

In Hash Partitioning suggested by Gothai and Balasubramanie (2014), the partitioning technique that is used when the keys are diverse and large data skew can exist when the key is present in large volume and it is apt for parallel data processing. The Round Robin partition technique developed by Gothai and Balasubramanie (2014) uniformly distributes the data on every destination data partitions and when number of records is divisible by number of partitions and then the skew is most probably zero. For example a pack of 52 cards is distributed among 4 players in a round-robin fashion. Also the Partition by Expression done by Gothai and Balasubramanie (2014) distributes data records to its output flow partitions according to a specified expression. In the function parameter we need to mention the required expression. The proposed idea Partition by Percentage distributes all the records to the different partitions based on the percentage specified as parameter.

Partition by percentage works as per our requirement when the records are more than 100 or 200. For example, if we have 10 records and 3 partitions and if we specify the percentage as 3 5, the first 3 records go to the first partition, the next 5 records go to the second partition and the remaining to the third partition. If we have 200 records and if we specify the percentage as 9 9, here we get what we want. The first 18 records go to first partition, next 18 to the second partition and the remaining records will go to the third partition.

Australian Journal of Basic and Applied Sciences, 8(17) November 2014, Pages: 131-136

Partition by percentage does not calculate the number of records to be sent to each partition according to the percentages specified; rather it takes chunks of 100 records at a time and sends exactly the number of records as mentioned in percentage parameter. For example suppose we have 200 records with some key running from 1 through 200 and the percentages mentioned are 10, 40, 20 and there are 4 partitions in the out port. Then partition by percentage will take first 100 records, send 1-10 to partition 0, 11-50 to partition 1, 51 - 70 to partition 2 and rest, i.e. 71 - 100 to partition 3. Then it will again take the next 100 records, i.e. 101 - 200, and send 101 - 110 to partition 0, 111 - 150 to partition 1, 151 - 170 to partition 2 and rest, i.e. 171 - 200 to partition 3. But if we have less than 100 records, say 8 records, then also it will try to take a chunk of 100 records, and send the first 10 in part 0. Since we don't have that many records, all the 8 records will go to first partitions.

In order to evaluate the performance, percentage partitioning is implemented in Hadoop even though this partitioning is not available in Hadoop. This section describes the percentage partitioning as an alternative of hash partitioning, Round Robin partitioning and Expression partitioning which will be incorporated in Hadoop. Also this section discusses how memory can be saved by means of a ReMap technique.

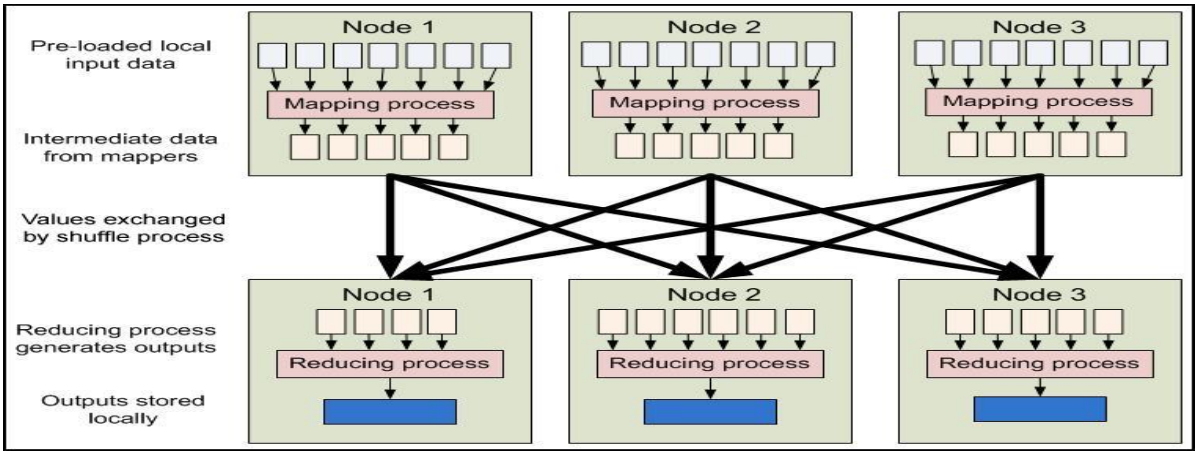

Fig. 1: Map Reduce Dataflow.

As designated in Figure 1, the data splits are applied to the Mapper and the outcome is sorted splits. Further these splits are facsimiled to the splits of Reducer for merging. During facsimileing the proposed percentage partitioning is incorporated. The partitioning is done as designated in Figure 2. After that the reducer does its work and engenders the final partitions.

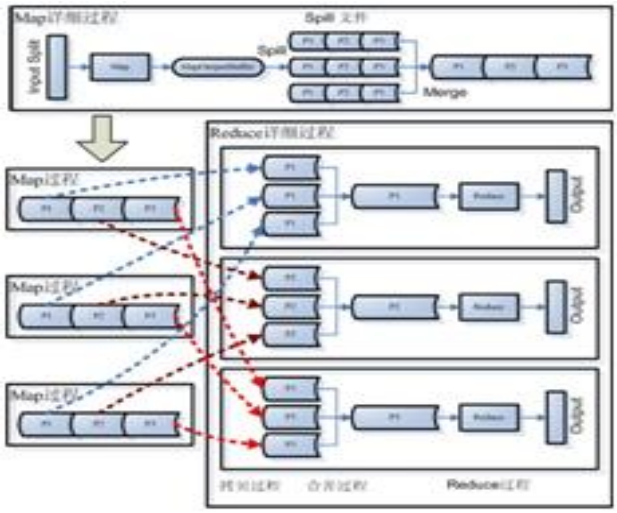

Fig. 2: Percentage Partitioning Framework.

Results:

This work examines how fine the algorithms dispense the workload and looks at how fine the memory is used, to estimate the performance of the proposed method. Tests performed in this paper were completed using LastFm Dataset with each record containing the user profile with fields like age, date, country and gender. The

Australian Journal of Basic and Applied Sciences, 8(17) November 2014, Pages: 131-136

authors simulated a network using VMware for Hadoop file system using these records as input. The tests are carried out with a range of size of dataset such as 1 Lakh, 3 Lakhs, 5 Lakhs, 10 Lakhs, 50 Lakhs and 1 Crore records. During the first experiment, an input file containing 1 lakh records is considered. The input set is divided into various splits and forwarded to Map Phase as mentioned in the MapReduce Framework. Here for this input file only one mapper is considered since the number of mappers is depends on the size of the input file. After mapping, partition algorithm is used to reduce the number of output records by grouping records based on the given percentage. After grouping, four partitions are created using the procedure Gender-Group-by-Country. All the corresponding log files and counters are analyzed to view the performance. In the other 5 experiments, input files with 3 Lakhs, 5 Lakhs, 10 Lakhs, 50 Lakhs and 1 Crore records are considered. As per the above said method all the input files are partitioned into four partitions.

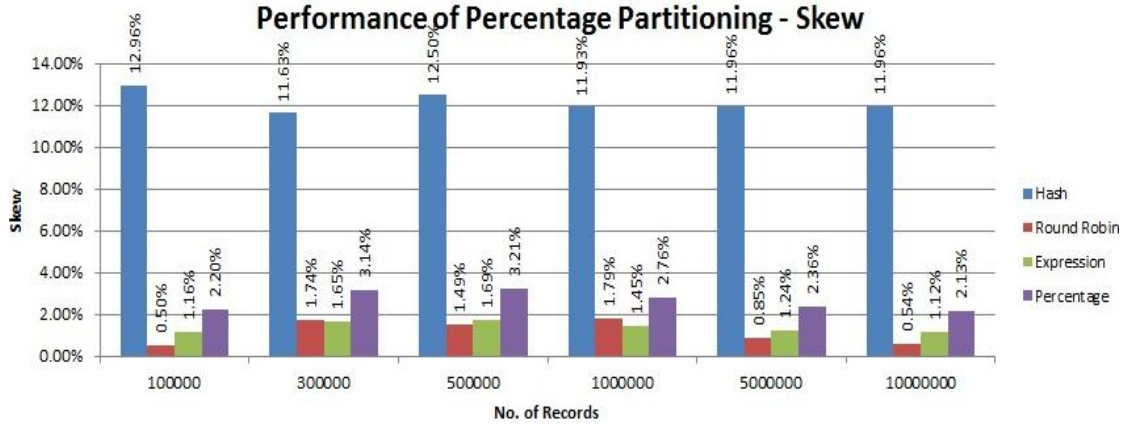

Fig. 3: Comparison Chart of Skew.

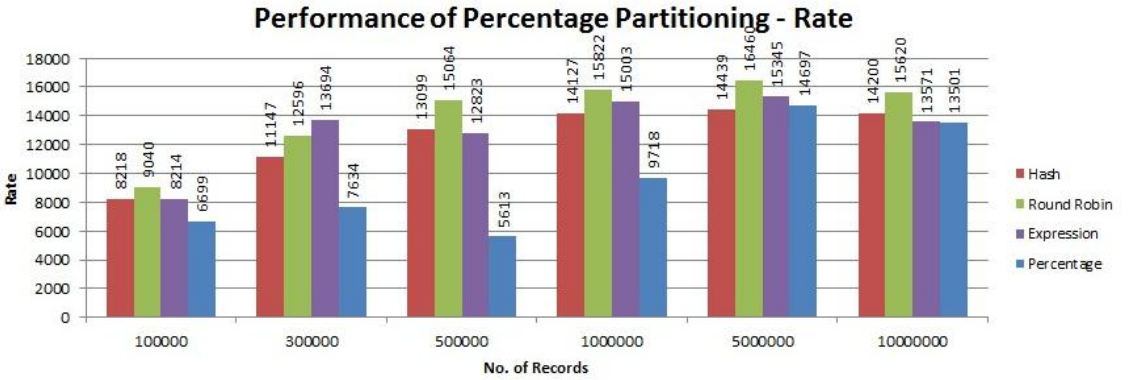

Fig. 4: Comparison Chart of Rate.

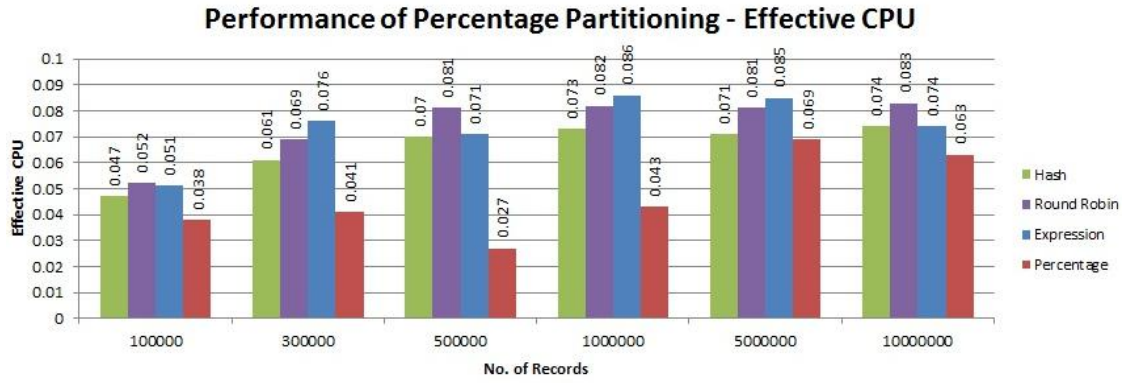

Fig. 5: Comparison Chart of Effective CPU.

Australian Journal of Basic and Applied Sciences, 8(17) November 2014, Pages: 131-136

Discussion:

In order to compare the different methodologies presented in this paper, this study uses only three metrics among various metrics such as Effective CPU, Rate and Skew, since only these three parameters shows the significant difference in outcomes. Rate displays the number of bytes from the Bytes column divided by the number of seconds elapsed since the previous report, rounded to the nearest kilobyte. Effective CPU displays the CPU-seconds consumed by the job between reports, divided by the number of seconds elapsed since the previous report. The skew of a data or flow partition is the amount by which its size deviates from the average partition size expressed as a percentage of the largest partition.

The tables 1, 2 and 3 shows the results when using various sized input files for the comparison of the performance of existing Hash partitioning, Round Robin Partitioning, Expression partitioning and proposed Percentage partitioning with the parameters Skew, Effective CPU and Rate respectively. Similarly, the figures 3, 4 and 5 shows comparison chart of the results of the above. From the tables and figures for results, it is shown that the proposed method is performing better than Hash Partitioning, Round Robin Partitioning and Expression partitioning based on the parameters Rate and Effective CPU but no better performance in the parameter skew.

Table 1: Performance Comparison of Skew.

No. of records	Hash	Round Robin	Expression	Percentage
100000	12.96%	0.50%	1.16%	2.20%
300000	11.63%	1.74%	1.65%	3.14%
500000	12.50%	1.49%	1.69%	3.21%
1000000	11.93%	1.79%	1.45%	2.76%
5000000	11.96%	0.85%	1.24%	2.36%
10000000	11.96%	0.54%	1.12%	2.13%

Table 2: Performance Comparison of Rate.

No. of records	Hash (in kb)	Round Robin (in kb)	Expression (in kb)	Percentage (in kb)
100000	8218	9040	8214	6699
300000	11147	12596	13694	7634
500000	13099	15064	12823	5613
1000000	14127	15822	15003	9718
5000000	14439	16460	15345	14697
10000000	14200	15620	13571	13501

Table 3: Performance Comparison of Effective CPU

No. of records	Hash (in sec)	Round Robin (in sec)	Expression (in sec)	Percentage (in sec)
100000	0.047	0.052	0.051	0.038
300000	0.061	0.069	0.076	0.041
500000	0.070	0.081	0.071	0.027
1000000	0.073	0.082	0.086	0.043
5000000	0.071	0.081	0.085	0.069
10000000	0.074	0.083	0.074	0.063

Conclusion:

This paper presented Percentage partitioning which is a comprehensive partitioning technique to improve load balancing for distributed applications. By means of improving load balancing MapReduce programs can turn out to be more proficient at managing tasks by reducing the overall computation time spent processing data on each node. Our work concentrates at small-sized to medium-sized clusters rather than large clusters. This study changes existing model of hash partitioning, round robin partitioning and expression partitioning and boosts it for a smaller environment with percentage partitioning. A sequence of experimentations have exposed that given a data sample, the Percentage partitioning architecture was capable to reduce Rate and effective CPU and by distributing records on average when compared with existing Hash Partitioning, Round Robin partitioning and Expression Partitioning. After this additional research can be made to introduce few other partitioning mechanisms so that it can be incorporated with Hadoop for applications using different input samples since Hadoop File System is not having any partitioning mechanism except hash key partitioning.

REFERENCES

Chang, F., J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrws, T. Chandra, A. Fikes and R.E. Gruber, 2006. Bigtable: a distributed storage system for structured data. In the proceedings of 7th UENIX symposium on operating systems design and implementation, pp: 205-218.

Fadika, Z. and M. Govindaraju, 2011. DELMA: Dynamic elastic mapreduce framework for CPU intensive applications. In the proceedings of the IEEE/ACM International. Symposium on Cluster, Cloud and Grid Computing, pp: 453-464.

Gothai, E. and P. Balasubramanie, 2014. A modified Key Partitioning for BIGDATA using MapReduce in Hadoop. Journal of Computer Science. (Accepted for April 2014)

Gothai, E. and P. Balasubramani, 2014. A novel approach for partitioning in Hadoop using Round Robin technique. Journal of Theoretical and Applied Information Technology. (Accepted for May 2014)

Gothai, E. and P. Balasubramanie, 2014. Partition by Expression using MapReducein Hadoop. International Journal of Engineering and Technology. (Under review)

Hsu, C.H. and S.C. Chen, 2012. Efficient selection strategies towards processor reordering techniques for improving data locality in heterogeneous clusters. Journal of Supercomputing, 60(3): 284-300.

Jeffrey Dean and Sanjay Ghemawat, 2004. MapReduce: Simplified Data Processing on Large Clusters. In the proceedings of *Sixth* Symposium on Operating System Design and Implementation, pp: 107-113.

Jiang, W. and G. Agrawal, 2011. Ex-MATE data intensive computing with large reduction objects and its application to graph mining. In the proceedings of the IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing, pp: 475-484.

Kavulya, S., J. Tany, R. Gandhi and P. Narasimhan, 2010. An analysis of traces from a production MapReduce cluster. In the proceedings of IEEE/ACM international conference on cluster, cloud and grid computing, pp: 94-95.

Kenn Slagter, Ching-Hsien Hsu, Yeh-Ching Chung and Daqiang Zhang, 2013. An improved partitioning mechanism for optimizing massive data analysis using MapReduce.Journal of Supercomputing, 66(1): 539-555.

Kenn Slagter, Ching-Hsien Hsu and Yeh-Ching Chung, 2013. Dynamic Data Partitioning and Virtual Machine Mapping: Efficient Data Intensive Computation with MapReduce.In the proceedings of IEEE 5th International Conference on Cloud Computing Technology and Science, pp: 220-223.

Liu, H. and D. Orban, 2011. Cloud mapreduce: A mapreduce implementation on top of a cloud operating system. In the proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp: 464-474.