A Review on Apache Hadoop Performance Enhancement by using Network Levitated Merge

Prashant B. Kanhere
M.E student, Dept. of C.E
Dr. D.Y. Patil SOET, Pune

Sathishkumar Penchala
Assistant Professor
Dr. D.Y. Patil SOET,Pune

ABSTRACT

Hadoop is popular large scale open source software framework which is written in JAVA programming for securely distributes storage and it is the master implementation of Map-Reduce programming used for cloud computation [1]. Now a days, hadoop faces a lot of problems to obtain the best outcomes from underlying system. The issue includes a serialization needs to gain quality performance which setback the aspect. Disk access and repetitive merges causes to current speedy interconnections that increases the volume of data sets. To stay with increasing volume of data sets, Hadoop also requires I/O ability from the underlying system nodes to process and examine data. So, for this 'HADOOP-A' [12] architecture is formed. Hadoop-A is an enhancement of framework that minimizes hadoop with peripherals for speedily data movement and bounding the existing limits to keep updating the architecture. A novel network algorithm for merging the data is explained in this paper. In supplementary, a full pipeline which is designed to overlay the shuffle, minimize phases and merge. The experimental results which shows that HADOOP-A is intensely speeds up data processing in Map – Reduce and extends the hadoop's throughput as double. HADOOP-A is significantly helps to optimize disk accesses which are caused by intermediate data.

General Terms

Hadoop, Map-Reduce, Hadoop acceleration, cloud computing, Network Levitated.

Keywords

Serialization, Repetitive Merges, Disk Access, Network Portability, Network-Levitated Merge, Pipelined Shuffle, Merge, and Reduce.

1. INTRODUCTION

MapReduce [7],[10] has appeared as a favored and simple-to-use programming structure for many firms to process massive size of data, perform massive computation, and accurate censorious knowledge for business analysis purposes in Business intelligence (BI). Hadoop is a free and open source software (FOSS) exertion of Map-Reduce, presently hold by the Apache Software Foundation (ASF), and supported by leading IT firms such as Yahoo! and Facebook [6]. Hadoop expands Map-Reduce framework with two heading component such as: Job-Tracker and bulk of Task-Trackers. The Job-Tracker orders Task-Trackers (as knownas slaves) to process data in parallel over two main operations: map and reduce. In this progression, the Job-Tracker is in privilege of scheduling map tasks (MapTasks) and minimize tasks (ReduceTasks) to Task Trackers. It also keeps track to their progress, mobilize run-time execution stats, and stem possible defects and errors through task re-execution. A reduce tasks requires to fetch a chunk element of the common or transitional output from all completed MapTasks. Globally, this tends to shuffling of transitional data (in segments) from each and every MapTasks to same ReduceTasks. For many data-demanding MapReduce [7], [10] programs, data shuffling can accelerated an important number of disk operations and argue for the finite I/O bandwidth. A number of experiments [3], [4], [5] have been carried out to enhance the performance of hadoop MapReduce framework [9], [10]. Condie et al. [5] has offered explicit network channels between MapTasks and ReduceTasks that significantly improve the delivery of data from MapTasks and ReduceTasks. It remains as a complex problem to debug the relationship of hadoop MapReduce's [7] three phases of data processing unit i.e. shuffle, merge, reduce and their conclusion or connotation to the productivity of hadoop. With an extensive diagnosis of hadoop MapReduce framework, especially it's ReduceTask, developer confess that the original architecture faces a lot of challenging problems to accomplish the remarkable performance from existing system. No ReduceTask can start cutting down the data as far as data have been merged closely together that to ensure the accuracy of MapReduce [10]. This results in serialization faults that constantly delays the reduce operation of ReduceTask. Generally, the present merge algorithm in hadoop merges the intermediate data chunks (a.k.a segments) from MapTasks. Whenever the numerous availability of chunks (including already merged segments) goes over the origin. Also whenever the entire size is hefty than the available memory that time these chunks are lavished to local disk storage. The data segments are merged repetitively due to the design of this

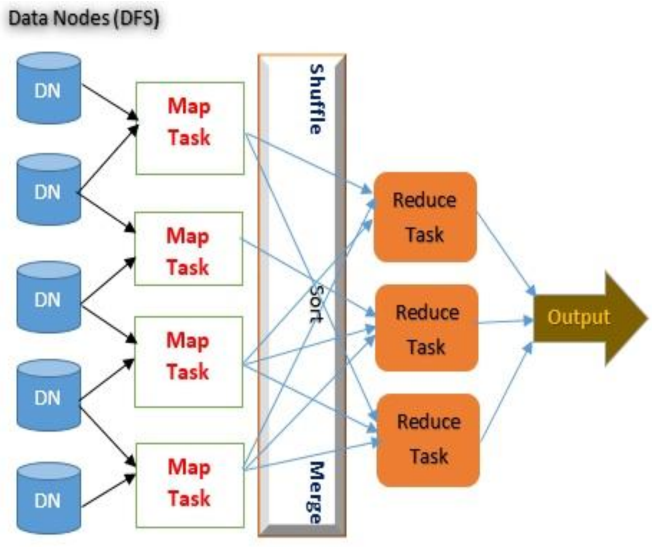

Fig. 1 –Hadoop MapReduce Framework for Data Processing.

Algorithm and therefore, this same data is accessed by multiple rounds of disk. To resolve these critical problems for hadoop MapReduce framework [7], authors have designed 'HADOOP-A' [12], a portable enhanced framework which can take the advantage of peripheral (a.k.a plug-in) components for performance acceleration and protocol minimization. A numerous improvement are introduced: 1) to perform the data merging without repetitive merge and additional disk accesses by ReduceTask that are enabled by Noval algorithm. 2) To overrun the shuffle, merge and reduce stages for ReduceTask, a full pipeline is designed and 3) A compact 'Hadoop-A' is implemented to remote direct memory access (RDMA) as well as for TCP/IP [2].After all by staying the ReduceTask above the local disk to enable the data merge operation, Researcher introduced the new algorithm i.e. Network Levitated Merge (NLM) [13]. Which have been implemented on the basis of large set of experiments on 'HADOOP-A' for enhancing the performance ability. These enhancement demonstration that the NLM algorithm is capable to remove the serializationbarrier and gracefully overrun the operations like merge and reduce for Hadoop ReduceTasks. Normally, to produce double throughput of hadoop data processing unit, 'Hadoop-A' is capable to do that. The rest part of this paper classified as follows. Section 2 is an overview of data processing in Hadoop MapReduce Framework [7]. The currently existing issue in hadoop framework and acceleration structure of hadoop is discussed in section 3 with that software architecture of 'Hadoop-A' [12] and accelerated data shuffling for RDMA are also included. Section 4 derives a Network levitated merge algorithm. Finally this paper is concluded in section 5.

2. OVERVIEW OF DATA PROCESSING IN HADOOP MAPREDUCE FRAMEWORK

Pipelined data process is one of the great fundamental feature of Hadoop MapReduce framework [7], [10]. As displayed in Figure no.1, Framework has three important execution phases as -> Map, Shuffle/Merge, and Reduce. User's input (jobs) datasets are splits in many data chunks (a.k.a segments), this is performed by job tracker when user's job is assigned to it. In that divided chunks, user's data is formulated as many records of {'Key',' Val'} combination. To execute the map function, job tracker prefers a numerous TaskTracker and lineup them for process, this is done in first phase. Every TaskTracker casts several Map-Tasks as per chunk of split. The mapping function alters original records into transitional outputs, which are in the above format {'key', 'Val'} combination. Files which are generated from splits are named as Map Output Files (MOFs). This Map Output File is arranged into data segments which are relevant as one per ReduceTask. Every segment contains a set of records. If MapTask completes one data split then it needs to reschedule to process the next split. Whenever the MOFs are accessible that time job Tracker fetches some TaskTracker to process ReduceTask. TaskTracker can generate certain simultaneous ReduceTasks. Every ReduceTask startedup by obtaining a segment which is expected for it from Map Output Files (also named as partition).

ReduceTask always requires to fetch the chunk portions from all Output files (MOFs). Throughout the ReduceTask whatever the data segments are obtained that will going to **shuffle** in next phase. While in the process of shuffling, the data chunks are merged on the basis of keys order which is already defined in data records. Now the ReduceTask are spilled so, need to store some data chunks in order to ease memory which has loads in local disks. Shuffling, Sorting and merging of data segments are usually known as '**Copy Phase**' in Hadoop framework. Generally, this stage is introduced as **'Shuffle/Merge'** stage. Now, the ending stage is Reduce phase. By using the reduction function, every ReduceTask start to process the merged chunks. When the final result outs then it is stored in HDFS (Hadoop Distributed File System) [11]. This is the extension of output segment which is derived after successfully execution of all processes.

3. ACCELERATION STRUCTURE OF HADOOP

First of all there are several issues in existing Hadoop framework like Serialization in data processing, Repetitive merges and disk accesses by ReduceTasks, unable to use RDMA interconnects [8]i.e. unable to support high capacitive working in high speed networks. So, these issues are the important factors which plays vital role in performance enhancement of existing hadoop [9] who removes all above issues and work with great functionality. For this purpose it needs to accelerate the functionality of existing Hadoop. So, for that 'Hadoop-A' **is** formed.

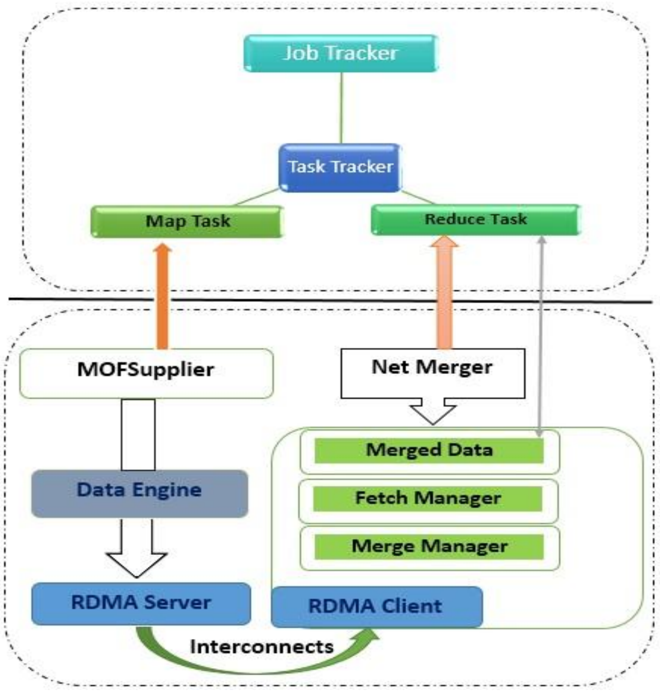

Fig. 2 –Software Architecture of Hadoop- A

3.1 Software Architecture Of Hadoop-A

Simple architecture of 'Hadoop-A' [12] is as shown in figure 2. Two new user configurable peripherals are introduced i.e. *MOFSupplier* and *Net-Merger* which are capable to leverage RDMA interconnects [8] and enables merge algorithms do to alternative data merging. MOFSupplier and Net-Merger are strapped C implementation. To prevent the data processing by overhead of Java Virtual Machine (JVM) and grant adaptable choice of new network system such as RDMA, which is still unavailable in JAVA. A basic demand of Hadoop-A is to preserve the same setup and maintain interface for end users. The TaskTracker launches the MOFSupplier and Net-Merger peripherals (plug-ins) which are designed and programed in 'C'- language. The configuration file contains acceleration parameters which gives the control choice to user for enabling and disabling the acceleration. Whenever the 'Hadoop-A' peripheral is activated, at that time hadoop programs are executed without any modification.

4. NETWORK - LEVITATED SHUFFLE, MERGE AND REDUCE PIPELINE

As mentioned in section (3), One major issue occurs in hadoop framework as – The barrier of shuffle/merge and reduce in serialization phase, which is resolved by using new Network -Levitated merge algorithm [13] as follows;

4.1 Data merging without serialization Barrier

Due to some certain limitations of memory as compared to the data size, hadoop resorts to merge repetition. For every completed Map Output File, it conjures a request i.e. *HTTP GET* which firstly makes an inquiry for length of partition and then pull whole data which will be going to store in memory or on disks. This provokes many memory related operations loads/stores, disk I/O etc… Unnecessarily the data partitions are pulled locally before merging due to the RDMA interconnects [8] which are comes to close to memory unwisely. The new algorithm having a key feature i.e. all the data partitions are merged exactly once and at that time it remains *levitated* over local disks. For clear understanding, just take an overlook on figure no.3, which shows the new Network – Levitated algorithms working [13]. New algorithm is derived from hadoop's priority queue based merge sort. The main idea behind this is to keep data laying on remote disk until the time to merge the expected data segments. The heading pair<key, Val> will be the starting point of merge operation for separate segments known as merge point, which is shown in figure (3). S1, S2 and S3 are the segments which will fetched and merged. Rather fetching all of them in local disk, it is better to create small header individually and fetch that header only. So, new algorithm uses this kind of ideology. Each header contain offset, partition length and the first combination of record <key, Val>. This <key, Val> combination is enough to compose a priority queue (PQ) to formulate these segments. Algorithm merges available data records <key, Val> combination in the same manner which is done in hadoop also. Algorithm contains four phase to get successes. First phase contain Header fetching, Priority Queue setup is derived in second phase, Fetch and merge operation are concurrently goes on, Last fourth phase is towards the completion.

5. DISCUSSIONS AND CONCLUSION

Cloud computing is the backbone of currently leading I.T firms and one of the vast issue in cloud computing is big data process which will going to rule on technology keyword at least for 10 future years. Hadoop is the master piece design of technology, which process a large data sets just in time. So, on the basis of brief study of architecture, functionality and working nature of hadoop, this review paper is designed.Wheneverbig data processing comes in consideration program need this approach and should work efficiently. Above Described Hadoop –A architecture has wide future scope as compare other processing frameworks who has low efficiency power to process. Also Hadoop –A will grow and take the advantage of working in heterogeneous cloud and distributed environment which leads that will deploy the improved MapReduce framework in heterogonous over homogenous environment with adaptive task tuning.This paper also describe and helps to understand Hadoop MapReduce [10] framework from basic level and up

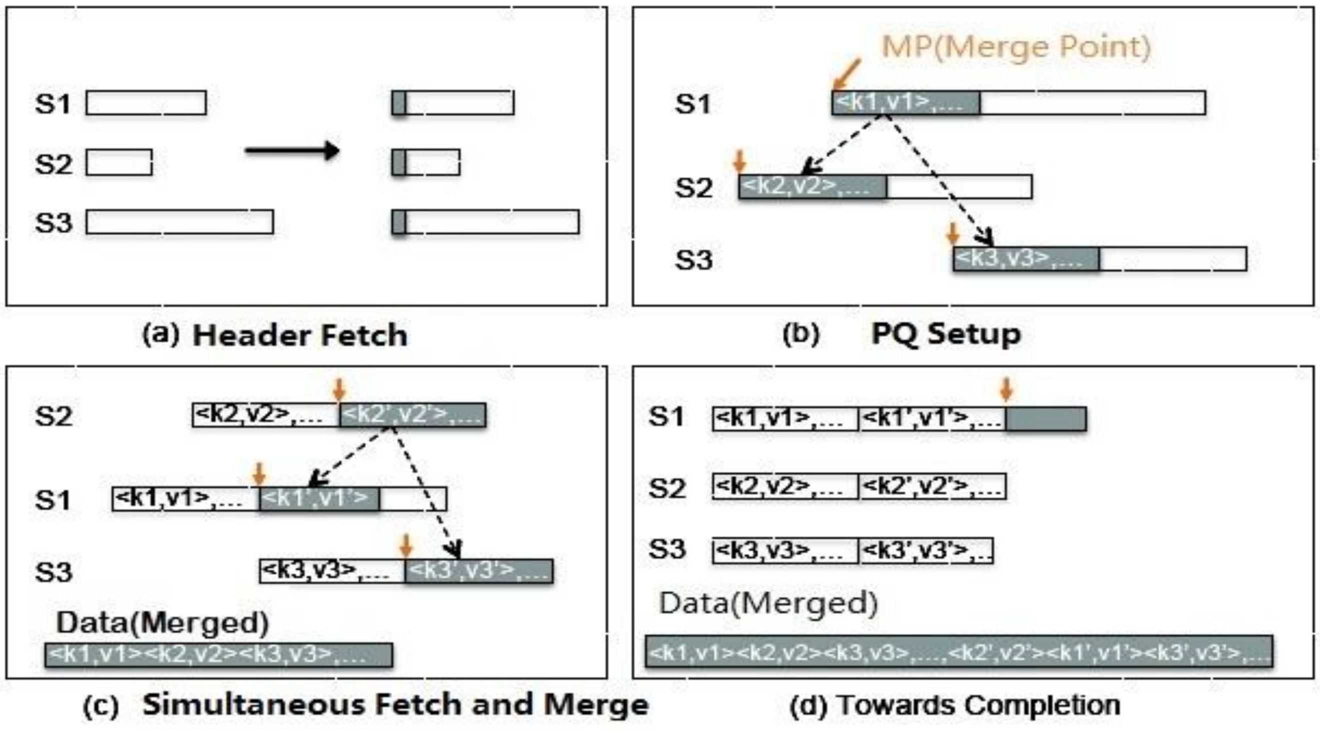

Fig.3 Algorithm for Network –Levitated Merge

To its working functionality level. This review paper contains the appropriate figures which gives a blunt ideology of hadoop framework and other working features.

6. ACKNOWLEDGMENTS

Any accomplishment requires work or any effort of many people. This paper is noDifferent. Working on this paper was a source of immense knowledge to us. Authors would like to express sincere gratitude to Assistant. Prof. Roshani Ade for her valuable support thought out this paper design. Authors acknowledge with a deep sense of gratitude and inspiration earned from faculty members and colleagues. Last but not the least heartily thanks to Kedarnath Dixit and Pritish Mehtre for encouraging to carry forward this work.

7. REFERENCES

[1] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Proc. Sixth Symp. Operating.

[2] Test-TCP. http://www.pcausa.com/Utilities/pcattcp.htm.

[3] D. Jiang, B.C. Ooi, L. Shi, and S. Wu, “The Performance of MapReduce: An In-Depth Study,” Proc. VLDB Endowment,

[4] System Design and Implementation (OSDI ’04), pp. 137-150, Dec. 2004.vol. 3, no. 1, pp. 472-483, 2010.M. Zaharia, A. Konwinski, A.D. Joseph, R.H. Katz, and I. Stoica, “Improving MapReduce Performance in Heterogeneous Environments,” Proc. Eighth USENIX Symp. Operating Systems Design and Implementation (OSDI ’08), Dec. 2008.

[5] T. Condie, N. Conway, P. Alvaro, J.M. Hellerstein, K. Elmeleegy, and R. Sears, “MapReduce Online,” Proc. Seventh USENIX Symp. Networked Systems Design and Implementation (NSDI), pp. 312-328, Apr. 2010.

[6] Apache Hadoop Project, http://hadoop.apache.org/, 2013.

[7] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. Sixth Symp. On Operating System Design and Implementation (OSDI), pages 137–150, December 2004.

[8] InfinibandTradeAssociation.http://www.infinibandta.org.

[9] Dawei Jiang, Beng Chin Ooi, Lei Shi, and Sai Wu. The performance of MapReduce: An in-depth study. In Proceedings of the 36th International Conference on Very Large Data Bases (VLDB), volume 3, pages 472–483, 2010.

[10] Yandong Mao, Robert Morris, and Frans Kaashoek. Optimizing MapReduce for multicore architectures. Technical Report MIT-CSAIL-TR-2010-020, MIT, May 2010.

[11] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[12] Yandong Wang, Xinyu Que, Weikuan Yu. Hadoop Acceleration through Network Levitated Merge, pages 3 12,http://mmc.geofisica.unam.mx/acl/edp/SC11/src/pdf/papers/tp50.

[13] Weikuan Yu, Member, IEEE, Yandong Wang, and Xinyu Que. Design and Evaluation of Network-Levitated Merge for Hadoop Acceleration: IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 3, MARCH 2014.