Efficient Processing and recouping of data using combiners in Map Reduce framework

V HEMANTH KUMAR
Dept of Computer Science and Engineering, JNTU-K, Raghu Institute of Technology, Visakhapatnam, Andhra Pradesh 530017, India hemanth12.vitam@gmail.com

M PURNA CHANDRA RAO
Dept of Computer Science and Engineering, JNTU-K, Raghu Institute of Technology, Visakhapatnam, Andhra Pradesh 530017, India
Purnachandrarao.m@gmail.com

CH NARAYANARAO
Dept of Computer Science and Engineering, JNTU-K, Raghu Institute of Technology, Visakhapatnam, Andhra Pradesh 530017, India
narayanarao.chokkapu@gmail.com

P HARIBABU
Dept of Computer Science and Engineering, JNTU-K, Raghu Institute of Technology, Visakhapatnam, Andhra Pradesh 530017, India
haribabupogiri@gmail.com

Abstract Consider any data structure, an Array for instance and declare the size of an Array either using static approach or dynamic approach. This cannot be a generic solution for large text files as this involves in huge memory allocations for the data structure. Even this can be a difficult procedure as the data size increases, processing the data will be time consuming process. Existing solutions such as lists and even heap will process the data effectively for large text files even to a certain boundary level (depends on the ram constraint). Addressing these huge volumes of data, the solution will not work in a single node and it has to spread across the cluster (storing data on the disk) .Hadoop will address all these big data problems using map reduce technique, as processing will be done in parallel manner. Map reduce is a functional programming model which has two functions map and reduce and will perform distributed parallel processing. In order to make the retrieval much faster, introducing the concept of implementing combiners between mapper and reducer. Implement a combiner function after the mapper function as the mapper generates output. The combined data that is performed by the combiners will be sent to the shuffle and sort functionality. And then from there it sends to the reduce function for obtaining the final output. The time taken to retrieve the data after processing by map reduce without using combiners will be more when compared with the map reduce processing using combiners. We generally make use of computation time and data transfer time constraints to support the above statement. This paper presents an effective approach for processing big data using combiners which will be also considered as map side reducers or mini reducers.

***Keywords*:** Map Reduce; Cluster; HDFS; Yarn; Combiners; Hadoop

1. Introduction

Over the past five years, Google have implemented hundreds of special-purpose computations for processing massive amounts of raw data, such as documents, request logs, etc., and such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across thousands of machines in order to finish in a reasonable period of time. The main issues to be focused on how to perform massive amount of computation and distribution of the data in simplified manner. So, a new abstraction has been designed that allows to express the simple computations but hiding the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. The new abstraction is being inspired by the map and reduces primitives present in Lisp. Most of computations involved are applying a map operation to each logical "record" in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key, in order to combine the data appropriately[1][5][7].

Storing and processing huge volumes of data according to the requirement became one of the major challenges in the real world. Hadoop is one of the solutions to the big data problems as the data is completely stored in files and the implementation of these Hadoop concepts will be done efficiently by the combination of HDFS, Map Reduce, Yarn where HDFS supports distributed parallel processing using map reduce and Yarn supports resource management [2]. And the major problem is if we want to process these huge volumes of data in a single node , it is a difficult procedure as we have to distribute the data across many nodes and should be processed in parallel way with the help of map reduce[4].

The next section of paper deals with map reduces architecture, processing of massive amount of data without using combiners. The next section of the paper deals with usage of combiners and processing of huge amount of data using combiners.

2. Architecture of Mapreduce Framework

Map-reduce is a programming model used for processing massive amount of data across different machines in cluster. In the concepts of Map reduce, first the input is given (can be any large text file) will be divided into blocks and that will be processed through the Input Format (IF) and generates (key, value) <k1, v1> combinations [2][6][8]. The (key, value) <k1, v1> combinations that are generated will be sent to the mapper and then the respective mapper (M) generates the list (key, value) <k2, v2> combinations. Further the list of (key, value) <k2,v2> combinations will be sent to the shuffle and sort(S&S) functionality and the sorting can be done based on any sorting technique assume the best one i .e Quick sort or key based sorting and then generates (key, list(values)) <k2,list(v2)> .Then the output of shuffle and sort will be sent to the reducer side(R) and generates the output in terms of (key, value) <k3, v3> combinations. This will be the final phase of output and that will be processed through the output format (OF).The following figure illustrates various phases for processing the data from input to output [4][7].

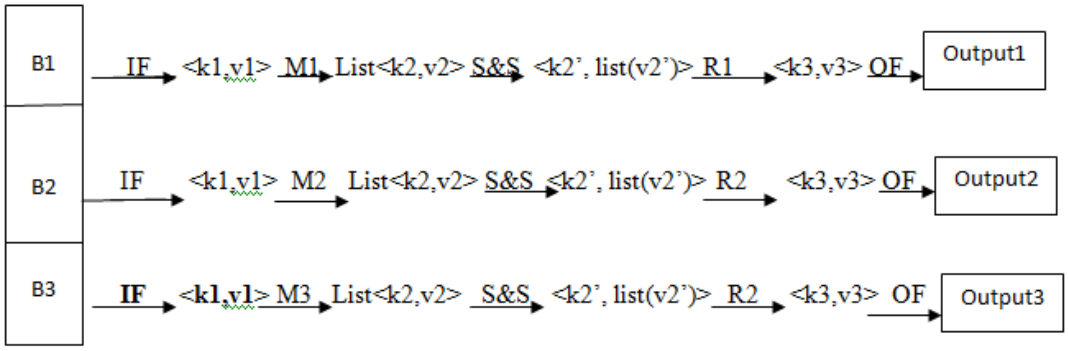

Fig.1. Architecture of Map Reduce

2.1. *Map Reduce Processing Without Using Combiner Function*

In this phase, the processing of the data is performed without using combiner between mapper function and reducer function. As a result the time that takes to process and retrieve the huge volumes of data will be more. And this will be illustrated below with the following use case taking computation time and data transfer time over the network as constraints[2][10].

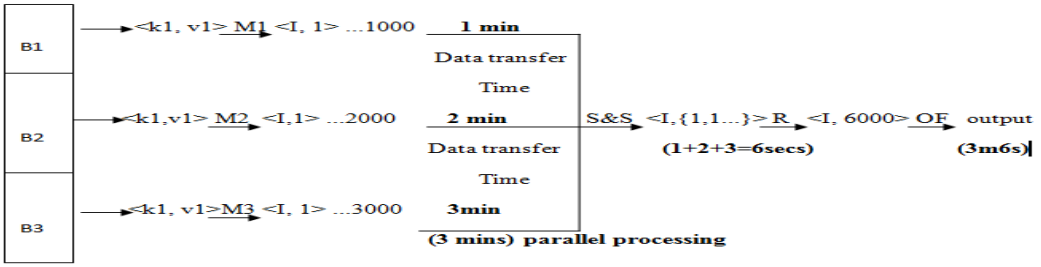

Fig.2. Without using combiner function between map (M) and reducer(R)

As illustrated in the figure we have an example, number of times a word is repeated in every block and we are calculating the computing time and data transfer time without using combiners.

Computation Time:

Assume the time taken to compute 1000 records was 1 second

i . e. 1000 records = 1 second

1000 records = 1000 milliseconds

1 record = 1 millisecond

Data Transfer Time:

Assume the time taken to transfer the data over the network for 1000 records was 1 minute

i . e 1000 records = 1 minute

1000 records = 60 seconds

1 record = 60 milliseconds

The number of blocks that are used in the above use case was three, therefore the number of map functions we can use were three. The data is read by the input format function (IF) and generates <k1, v1> for each map function which will pass through map function and generates the list of values the number of times each word is repeated say <I, 1> <I, 1> <I, 1> <I, 1> <I, 1> where I is the word that is repeated for many times within a single block [6][8].

Let us assume the number of times that a word is repeated in Block 1 was 1000 times, Block 2 was 2000 times and block 3 was 3000 times. And the data transfer time that is required to transfer over the network from map function to shuffle and sort function was **(1+2+3) minutes**. Since the data transfer time for 1000 records was 1 minute. And the important point to consider is as the mapper perform parallel processing the time that is required to process all the three blocks was **3 minutes** [7][9].

Then it reaches to the shuffle and sort functionality and then the list if values will be like <I , {1,1,1,1....1 (1000 times) ,1,1,1,1,1.....1 (2000 times),1,1,1,1,1.....1(3000 times) }.And then the values will be computed by the reducer which was sent from the shuffle and sort functionality. The time taken to compute these records was **(1+2+3) seconds**. Since the computation time for 1000 records to compute was 1 second and finally the output will be sent to the output using output format (OF) functionality from the reducer [5].

The overall time both the data transfer and computing time required to process the data without using combiners was **3 minutes and 6 seconds.**

3. Combiners

Combiners are executed on each machine that performs a map task. Typically the same code is used to implement both the combiner and the reduce functions. The only difference between a reduce function and a combiner function is handling the output of the function by libraries. The output of a combiner function is written to an intermediate file that will be sent to a reduce task and the output of a reduce function is written to the final output file and also these combiners speed up computation process on the data sent over the network [1][6][9]

3.1. *Map Reduce Processing Using Combiner Function:*

In this proposed paper, we are going to place the combiner function between the map function and reducer and before the shuffle and sort functionality. The data that is produced as an output from the map function in the format <I,1> <I,1> for 1000 times w.r.t block 1 <I,1> <I,1> for 2000 times w.r.t block 2 and <I,1> <I,1>... for 3000 times w.r.t block 3 from all the three blocks will be sent to the combiner function. Here the combiner function will combine all the 1000 records of block1 to 1 record and 2000 records of block2 to 1 record and 3000 records of block3 to 1 record. Now the computation time that is required to combine the 1000 records of block1 to 1 record was 1 second. Similarly for block2 was 2seconds (2000 records) and block3 was 3seconds (3000 records). The overall time required for combiner function to combine the records from all the three blocks was **3 seconds** as they were working in a parallel process.

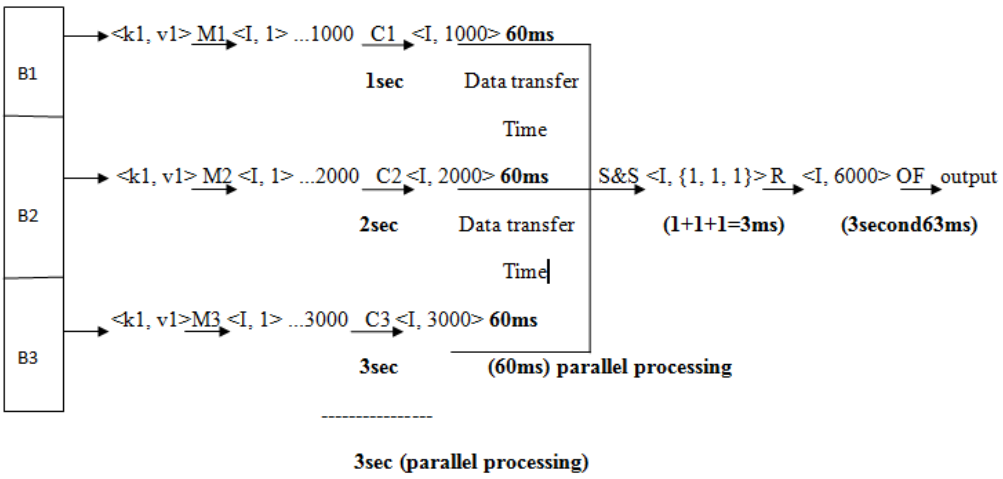

Fig.3. Using combiner functions between map (M) and reduce(R) functions

The output that is obtained from the combiner function(C1,C2,C3) will be sent to the shuffle and sort functionality and the data transfer time that requires is 60ms for block1,60ms for block2 and 60ms for block3. Since the output of the combiner for each block is one record each. The time that is required to transfer 1 record is 60ms. The overall data transfer time that requires is **60ms** as all the process will perform in parallel manner. From shuffle and sort functionality the list of values <I, 1000, 2000, 3000> i.e. <I,1,1,1> considered as

individual record. And the time that is required to compute 3 records in the reducer phase was **3ms** since each record takes 1ms. From reducer phase the output will be sent to the output through output format (OF) functionality [3][7][9].

The overall time both the data transfer and computing time required to process the data using combiners was **3 seconds and 63 milliseconds.**

The different steps for processing of map-reduce job is summarized below [3]

- **C**lient/user submits map-reduce jobs, it goes to Job tracker. Client program contains all information like the map, combine and reduce function, input and output path of the data.
- The Job Tracker puts the job in a queue of pending jobs and then executes them on a FCFS (first come first serve) basis.
- The Job Tracker first determine the number of splits from the input path and assign different map and reduce tasks to each Task Tracker in the cluster. There will be one map task for each split.
- Job tracker talks to the Name Node to determine the location of the data i.e. to determine the data node which contains the data
- The task tracker is pre-configured with a number of slots which indicates that how many tasks (in number) Task Tracker can accept. For example, a Task Tracker may be able to run two map tasks and two reduce tasks simultaneously.
- When the job tracker tries to schedule a task, it looks for an empty slot in the Task Tracker running on the same server which hosts the data node where the data for that task resides. If not found, it looks for the machine in the same rack. There is no consideration of system load during this allocation.
- Now when the Task is assigned to Task Tracker, Task tracker creates local environment to run the Task.
- Task Tracker needs the resources to run the job. Hence it copies any files needed from the distributed cache by the application to the local disk
- Task Tracker can also spawn multiple JVMs to handle many map or reduce tasks in parallel. Task Tracker actually initiates the Map or Reduce tasks and reports progress back to the Job Tracker.
- When all the map tasks are done by different task tracker they will notify the Job Tracker. Job Tracker then ask the selected Task Trackers to do the Reduce Phase
- The Task Tracker nodes are monitored. A heartbeat is sent from the Task Tracker to the Job Tracker every few minutes to check its status.
- If Task Tracker does not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different Task Tracker.
- When the work is completed, the Job Tracker updates its status.
- Client applications can poll the Job Tracker for information.

The whole process is depicted in figure below

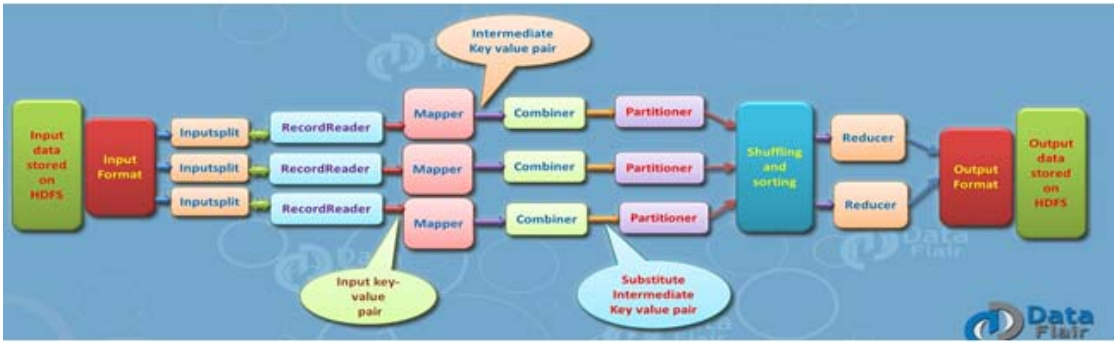

Fig.4. Mapreduce Framework Application Flow

Conclusion

By considering the above mentioned use case and comparing the existing system with proposed system we can say that the retrieval and processing of huge volumes of data using combiners will take less time when compared with the process without using combiners. The usage of combiner functions by the map reduce frame work depends on the run time environment, if enough memory was available then the combiner functions will call ,otherwise the mapper directly sends the map function output to shuffle and sort functionality and then to reducer function and finally output phase.

This paper deals with architecture of map reduce framework and its processing explained without combiners and how the processing speed is being improved with use of combiners.

References

[1] https://en.wikipedia.org/wiki/MapReduce
[2] feng li, beng chin ooi, m. tamer özsu, sai wu, Distributed Data Management Using MapReduce,ACM,2009
[3] Seema Maitrey,C.K.Jha, Map Reduce: Simplified Data Analysis of Big Data,Elseiver,2015
[4] http://ieeexplore.ieee.org/document/4125858/
[5] Mohd RehanGhazi, DurgaprasadGangodkar, Hadoop, MapReduce and HDFS: A Developers Perspective, Elseiver, 2015
[6] http://data-flair.training/blogs/shuffling-and-sorting-in-hadoop
[7] Katarina Grolinger , Michael Hayes, Wilson A. Higashino,Alexandra L'Heureux,David S. Allison,Miriam A.M. Capretz, Challenges for MapReduce in Big Data,IEEE,2014
[8] http://hadooptutorial.info/combiner-in-mapreduce
[9] Tzu-Chi-Huang, Kuo-Chih-Chu,Wei-Tsong Lee,Yu-Sheng Ho, Adaptive Combiner for MapReduce on cloud computing, ACM, 2014
[10] https://hadooptutorial.wikispaces.com/MapReduce