European Journal of Academic Essays 1(5): 1-4, 2014
ISSN: 2183-1904
www.euroessays.org

Survey on Methodology & Examples of Mapper / Reducer /Combiner for Big data

Vasudha rani Vaddadi
IT Department
GMRIT
Rajam, Andhra Pradesh, India
Vasudharani.v@gmrit.org

Abstract: Map-Reduce is one of the Big data technologies talks about distributed computations on massive amounts of data and an execution environment for high scale data processing on clusters. Principle behind Mapper/ Reducer is the general Divide and Conquer algorithm in many different domains. Map/Reduce is giving advantages over parallel databases include storage-system independence and fine-grain fault tolerance for larger jobs. Map /Reduce parallelism is shown as combined paradigm of Shared memory and data, message passing with pipelined implementation at a higher level. Hadoop is an Open source. It is the Java-based implementation of MapReduce which is detailed discussed here. In this paper, methodology behind Mapper/Reducer/Combiner required for big data with an example Web Search Engine is clearly demonstrated. Web Search Engine working is compared having combiner and not having combiner .

Keywords:Mapper, Reducer ,Combiner , hadoop , HDFS. Message passing ,pipelining ,Shared Data

1. Introduction

Map Reduce being a higher level parallel programming paradigm includes the capabilities of Message passing ,Data Parallel ,pipeline work .Map/Reduce motivates to redesign and convert the existing sequential algorithms to Map/Reduce algorithms for big data so that the paper presents method of map reduce for data analysis in an efficient manner reducing the Time complexity. Map Reduce is a Programming model used by Google and A combination of the Map and Reduce models with an associated implementation ,used for processing and generating large data sets. Map Reduce is highly scalable and can be used across many computers. Many small machines can be used to process jobs that normally could not be processed by a large machine.

Map Reduce has the implementation of Shared Memory where in DATA is shared among the multiple processors and the work is divided as work units w1,w2,w3 …..wn. In sequence with message passing capability it has updated to have both Partitioning the data as D1,D2,D3….Dn including Partitioning of work with intermediate Key / Value pairs, ends with finalized Key / Value pairs ,Starting pairs are sorted by key,Iterator supplies the values for a given key to the Reduce function.

Typically a function that: Starts with a large number of key/value pairs , One key/value for each word in all files being greped (including multiple entries for the same word)

– Ends with very few key/value pairs

One key/value for each unique word across all the files with the number of instances summed into this entry ,Broken up so a given worker works with input of the same key.
Users specify the computation in terms of a *map* and a *reduce* function, Underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, and

Underlying system also handles machine failures, efficient communications, and performance issues. Hadoop is a framework written in Java for running applications on large clusters of commodity hardware and incorporates features similar to those of the Google File System (GFS) and of the MapReduce computing paradigm. Hadoop's HDFS is a highly fault-tolerant distributed file system and, like Hadoop in general, designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications that have large data sets.

One of the attractive qualities about the MapReduce programming model is its simplicity: an MR program consists only of two functions, called Map and Reduce, that are written by a user to process key/value data pairs. The input data set is stored in a collection of partitions in a distributed

Corresponding Author: Vasudha rani Vaddadi, GMRIT,Rajam, vasudharani.v@gmrit.org,+91 9177377469

file system deployed on each node in the cluster. The program is then injected into a distributed processing framework and executed in a manner to be described. The Map function reads a set of "records" from an input file, does any desired filtering and/or transformations, and then outputs a set of intermediate records in the form of new key/value pairs. As the Map function produces these output records, a "split" function partitions the records into R disjoint buckets by applying a function to the key of each output record.

This split function is typically a hash function, though any deterministic function will suffice. Each map bucket is written to the processing node's local disk. The Map function terminates having produced R output files, one for each bucket. In general, there are multiple instances of the Map function running on different nodes of a compute cluster.

We use the term instance to mean a unique running invocation of either the Map or Reduce function. Each Map instance is assigned a distinct portion of the input file by the MR scheduler to process. If there are M such distinct portions of the input file, then there are R files on disk storage for each of the M Map tasks, for a total of M × R files; Fij , 1 _ i _ M, 1 _ j _ R. The key observation is that all Map instances use the same hash function; thus, all output records with the same hash value are stored in the same output file. The second phase of a MR program executes R instances of the Reduce program, where R is typically the number of nodes. The input for each Reduce instance Rj consists of the files Fij , 1 _i _ M. These files are transferred over the network from the Map nodes' local disks. Note that again all output records from the Map phase with the same hash value are consumed by the same Reduce instance, regardless of which Map instance produced the data. Each
Reduce processes or combines the records assigned to it in some way, and then writes records to an output file (in the distributed file system), which forms part of the computation's final output.The input data set exists as a collection of one or more partitions in the distributed file system. It is the job of the MR scheduler to decide how many Map instances to run and how to allocate them to available nodes. Likewise, the scheduler must also decide on the number and location of nodes running Reduce instances. The MR central controller is responsible for coordinating the system activities on each node. A MR program finishes execution once the final result is written as new files in the distributed file system.

2. Hadoop

Hadoop is an Open source. It is the Java-based implementation of MapReduce. Use HDFS as underlying file system. Hadoop implements a computational paradigm named Map/Reduce, where the application is divided into many small fragments of work, each of which may be executed or reexecuted on any node in the cluster. In addition, it provides a distributed file system (HDFS) that stores data on the compute nodes, providing very high aggregate bandwidth across the cluster. Both Map/Reduce and the distributed file system are designed so that node failures are automatically handled by the framework.

Hadoop's Distributed File System is designed to reliably store very large files across machines in a large cluster. It is inspired by the Google File System. Hadoop DFS stores each file as a sequence of blocks, all blocks in a file except the last block are the same size. Blocks belonging to a file are replicated for fault tolerance. The block size and replication factor are configurable per file. Files in HDFS are "write once" and have strictly one writer at any time.

Hadoop Distributed File System – Goals:

- Store large data sets
- Cope with hardware failure
- Emphasize streaming data access

The Hadoop Map/Reduce framework harnesses a cluster of machines and executes user defined Map/Reduce jobs across the nodes in the cluster. A Map/Reduce computation has two phases, a map phase and a reduce phase. The input to the computation is a data set of key/value pairs.Tasks in each phase are executed in a fault-tolerant manner, if node(s) fail in the middle of a computation the tasks assigned to them are re-distributed among the remaining nodes. Having many map and reduce tasks enables good load balancing and allows failed tasks to be re-run with small runtime overhead.

Hadoop Map/Reduce – Goals:

• Process large data sets

• Cope with hardware failure

• High throughput

3. Mapper /Reducer Methodology

Map Reducer consists Shared memory –Partitioning the work means Different parts of the work will be done on same data by different processors.

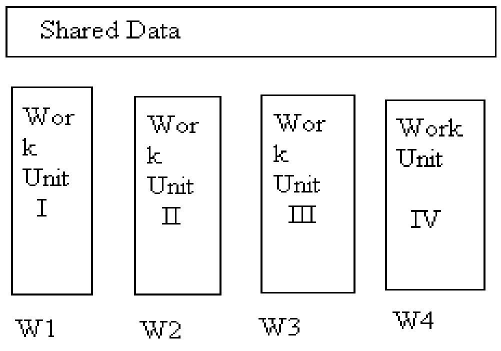

Figure 1: Shared Memory

Map reduce Supports Message Passing means Different Parts of the work will be done Different data by different Processors.

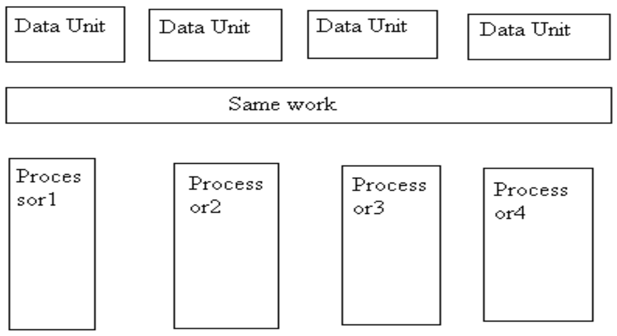

Figure 2: Message Passing

In the Implementation of Mapper Reducer Technology, Data is shared among the processors and even work is also shared among all the processors in pipelined manner. And the Architectural diagram is given below.

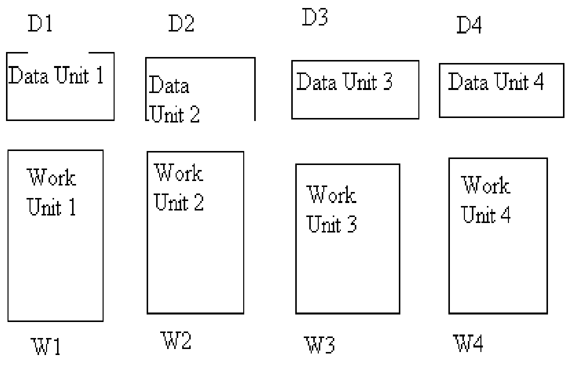

Figure 3: Map reduce = Shared Memory + Message Passing

4. Map Reduce Architecture

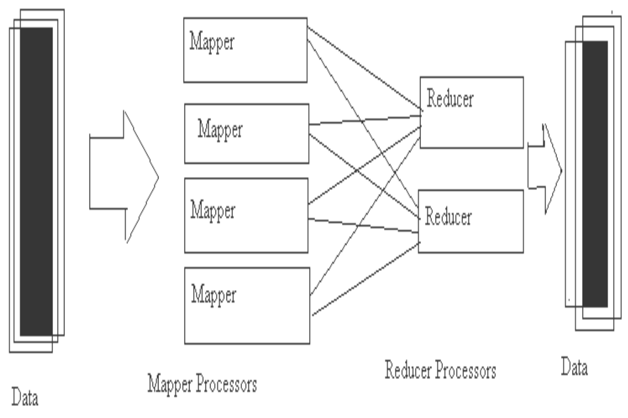

Figure 4: Map Reduce Architecture

For the Mapper Reducer Data should be given , upon which Mapper is assigned with certain work and reducer is assigned with certain work. All the mappers will be working at the same time for solving any problem ,So no of processors required is the no of mappers maintained. And for the task of reducer the same processors can be used and applied on the output data from Mappers.

5. Mapper/Reducer Example

Example for Map Reduce: Web Mining:

Web searcher for data obviously go with a web search engine like Google. To understand its process clearly we can start with an example of counting words in group of documents.

Need to calculate computing frequency of words using Map Reduce.

If the task given searching data which performs word count on web data (web Intelligence) using Map Reduce Architecture. Mapper is given with set of Documents ,In finding the word count , for a word search needs to happen in all the documents and is maintained as (Did , Conid) -→ (word , count) which gives the word count information in respective documents. Similarly all the mappers will do the concerned work for concerned documents in the format (Did , Conid) information to the Reducer. Work of Reducer is to combine the information for the same word from different mappers and results the words and counts.

Did: Document id

Conid: Contents

Wd: Word

Countid : Count occurrence

Of the word.

Mapper:

(Did , Conid) -> Map(Wd ,Countid)

Reducer:

Wd ,FUN(countid1,…..CountidN…….)

<-- Reduce (Wd , [….Countid ….]

Map –Reduce Platform responsible for routing pairs to reducers → all pairs the same key values end up at the same reducers. Map reduce essentially sorts the values to finally end up in the correct reducers. Map reduce reads data from a file System or data base and writes fresh data it’s a batch process always processes more data very different from querying data from Data Base.

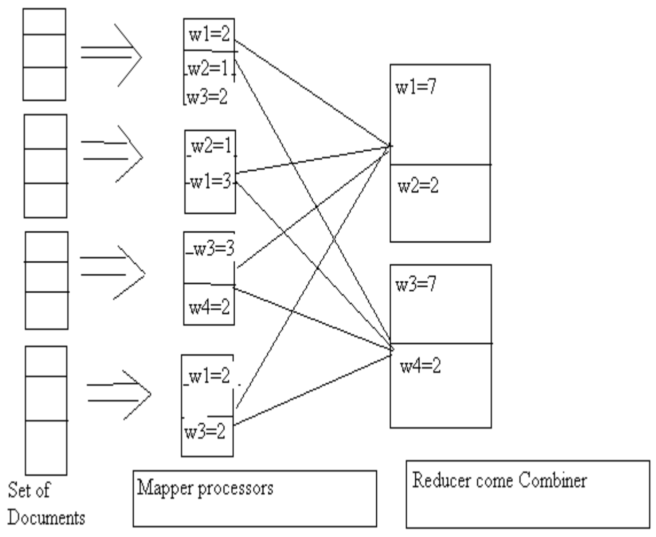

Figure 5: Working Example

Mapper reducer without combiner and with Combiner Phase can be implemented. We can have Calculations over how much data is created at each phase . After mapper phase the same sized data is generated because each word is entered every time it is seen. If word count is summed up before moving onto the reduce phase. Process of reduce phase operation partially as a part of mapper process is called Combine before emitting for sorting by reducer.

6. Conclusion & Future work

My Conclusion is that map reduce is the better method for Parallel processing of big data in the clustered environment takes less processing time and Fault Tolerant.

In order to go for big data analytics , detailed knowledge of methodology with examples of map reduce is a key area to learn.

It has been proved in this paper for the application of web mining with clear explanation of its methodology. Showing the results graphically will be the extension of my work.

This paper has discussed the methodology of Map Reduce with a good example of web mining .In continuation to this learning Hadoop famework implementation leads us to work in the area of Big data. And more analysis can be done in terms of time complexity and Graphs can be drawn. Map Reduce in combination with Hadoop having lot of its applications in the ares of marketing ,Banking Data ,Cricket information , Sales information etc…

References

[1]. Jongwook Woo, Yuhang Xu, "Market Basket Analysis Algorithm with Map/Reduce of Cloud Computing," .April 2011.

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

[3]. "Data-Intensive Text Processing with MapReduce", Jimmy Lin and Chris Dyer, Tutorial at the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2010), June 2010, Los Angeles, California.

[4]. Ping ZHOU †, Jingsheng LEI, Wenjun YE , "Large-Scale Data Sets Clustering Based on MapReduce and Hadoop", Journal of Computational Information Systems 7: 16 (2011) 5956-5963.