**Indian Journal of Science and Technology,** *Vol 9(1), DOI:10.17485/ijst/2016/v9i1/78964, January 2016*
*ISSN (Print) : 0974-6846*
*ISSN (Online) : 0974-5645*

# MapReduce: A Technical Review

**T. Y. J. Naga Malleswari[1]* and G. Vadivu[2]**

[1]Department of Computer Science and Engineering, SRM University, Chennai - 603203, Tamil Nadu, India; nagamalleswari.t@ktr.srmuniv.ac.in
[2]Department of Information Technology, SRM University, Chennai - 603203, Tamil Nadu, India; vadivu.g@ktr.srmuniv.ac.in

## Abstract

MapReduce, a programming model, allows parallel processing of large amount of data sets where various data mining techniques are not quite useful. It's Map and Reduce functions can be customized by the developers according to their application. This paper gives an idea of MapReduce, its advantages and disadvantages. This paper also focuses on how MapReduce is used, how map and reduce computations are customized, implemented in several scenarios such as in medical field to generate medical reports by processing large medical data sets, stream processing and workflow scheduling in multi core processors, in distributed environment, for processing distributed data sets by using pilot abstractions. We also represent how MapReduce used for deduplication of files to save disk space in data centers. MapReduce based Pre-Post (MRPre-Post) a parallel data mining algorithm is adapted in Hadoop platform to achieve scalability. MapReduce is implemented in vHadoop (Virtual Hadoop), a scalable hadoop virtual cluster to process machine learning algorithms. The scenarios discussed in this paper help developers and researchers how to customize and use MapReduce in their applications.

**Keywords:** Big Data, Data Management, Distribute Environment, Iterative Computation, Large Scale Data Processing, MapReduce

## 1. Introduction

A new technology, by utilizing high speed network multiple users share the computing resources like storage, network servers, services and applications through cloud computing[1]. This model is based on utility computing (pay per use). Hence, data centers of several companies are transformed to the cloud to become burden free of managing and configuring their infrastructure.

Many industries, Government, academia are transferring the data to the cloud with high speed. This Big data, an unstructured data is in the size of terabytes and peta bytes. It has three properties: Variety, velocity and volume. Big data processing[2] is one of the challenges as the data is very large in volume. To address this challenge many parallel computing platforms[3] has been developed which use thousands of commodity hardware MapReduce is an efficient programming model to process large volumes of data. This model was introduced by Google. Hadoop, a framework with HDFS, an open source implementation is developed by Apache. Fault tolerance, automatic parallelization, data locality-based optimizations, and scalability[4] are the features of this model.

## 2. MapReduce

Previously large scientific calculations are processed in parallel by using parallel computers. Later all the nodes are connected by a network in a rack. There exists several numbers of racks in a single cluster. Complex calculations are processed in matter of minutes rather than hours. Distributed File System (DFS) is used for better results.

*Author for correspondence

Google File System (GFS), Hadoop Distributed File System (HDFS) are several distributed file systems. In order to handle node failure and rack failure three chunks of each file of size 64 MB are replicate in different nodes and racks. MapReduce is a parallel data processing approach[5] and is implemented in cloud environment on a computer cluster. Many calculations are performed on large scale of data on computing cluster in an efficient manner and are tolerant to hardware failures. For processing big data[6], a parallel computing framework MapReduce is used. Google used MapReduce for computation of large vector multiplications that are used in finding page rank[7]. Relational Algebra operations widely use MapReduce to retrieve the results from a large database. MapReduce is implemented on HDFS from Apache.

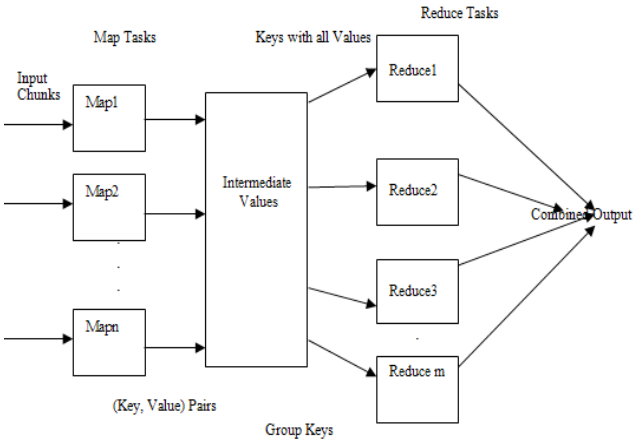


**Figure 1.** MapReduce workflow.

Map tasks and Reduce tasks are to be developed by the user. Figure 1 explains work flow of Map Reduce. Map tasks work on chunks of files and (key, value) pairs are produced based on the input function and the work given to the Map task. In the cluster one node is designated as name node (master) and others as data nodes. Map tasks are carried by data nodes. Name nodes are selected based on network state, data locality and name node statistics. This is done by scheduler program which plays a vital role in MapReduce performance. All (key, value) pairs are gathered by master node and are shuffled according to the key. All the (key, value) pairs are combined in the way specified by the user in Reduce tasks. The number of Map tasks and Reduce tasks can be configured by the user. The master nodes assigns map tasks and reduce tasks to data nodes. There exists a secondary name node which keeps on reading from name node RAM and sent to hard disk periodically. This is used to back up the data when a failure in master node. Any failure in data node is detected by name as it continuously pings the data nodes. If any failure is detected the map task is rescheduled to other data node that is available and the map task been redone even if it had completed because the output of the map task is on the failed node and is not available for reduce tasks.

**Table 1.** **Map and Reduce tasks**

| | |
|---|---|
| Map | (Key.value) pairs-àIntermediate (Key, Value) pairs |
| Reduce | Intermediate (Key, Value) pairs---à list of values |

## 3. Pros and Cons

Many applications like big data analytics, data mining, scientific computation MapReduce, which is a distributed data processing framework is used because of the following benefits:

- Scalable: The requirements of computation are dynamically changed as increasing data volumes and system performance scale up and down. Petabytes of data handled by MapReduce as it are deployed on thousands of nodes by supporting parallel execution.
- Efficient: MapReduce is efficient for many applications as no need of loading data into a database which involves very high cost.
- Flexible: The user can specify the exact processing code in the map and reduce functions and peta bytes of data are processed.
- Fault Tolerance: High availability is ensured by maintaining multiple replicas of file chunk in a distributed file system. Job consists of many tasks and is carried by many nodes. If anyone failed that is compensated by another node capable of handling the load.
- Low-cost: MapReduce run on very less expensive commodity hardware configured in the form of clusters and are managed by an open source operating system which in turn is cost effective.
- Parallelism: MapReduce is a framework of parallel processing of many tasks. The user can concentrate on the problem rather than implementation.

Drawbacks as follows:

- Single point failure of the name node: A secondary name node creates checkpoints and backup the metadat of name node but real redundancy is not provided.

- MapReduce is not suitable for:
  - Computation of those problems which depends on previous computed values.
  - Algorithms for those problems that depend on shared global state.
  - Real time processing.
  - Online Transactions Processing.
  - When data are widely distributed[9].

# 4. MapReduce Implementations

Some of the scenarios where MapReduce is implemented

## 4.1 Medical Applications

MapReduce is used in medical field where thousands of medical records to be analyzed to diagnose the disease of a patient. As an example MapReduce is used to reduce the time of analyzing the very large data sets of medical records of Mild Hypertrophic Cardiomyopathy (HCM). HCM[10] leads to Sudden Cardiac Death (SCD) of young athletes while doing some physical activity[11]. It involves unclear parameter dependencies[12], high dimensional data and data intensive computations. Scoring function is designed with the help of 35 medial parameters obtained from athletes who had SCD risk factors. These 35 parameters are SCD high risk factors[13]. Some of the parameters are personal history, family history, wall thickness of left ventricular etc… For high risk SCD with HCM, overall score ≥ 23 and high risk score ≥ 1, for high risk SCD without HCM, overall score < 23 and high risk score ≥ 1, with mild HCM overall score ≥ 23 and high risk score = 0.

MapReduce is used to classify large data sets into pathologic and physiologic cases in parallel. Scoring function is used in map tasks and produces classification, assigned an integer. The outcomes of all map tasks are observed and the outputs with the same key are combined by the combiner module. Based on the four tuple of all classifications computation of component wise sum is performed by the reduce tasks. Machine learning[14] optimization technique rule based approach[15] is used to find related combinations of medical parameters by which the dependencies between some parameters are satisfied, classification of data set into pathologic and physiologic cases is done successfully, for good results. This algorithm is implemented in parallel by MapReduce and in iterations on Beowulf Cluster, Hadoop enabled. Based on previous MapReduce outcomes and some set of rules parameter set is further improved.

## 4.2 C-MR

MapReduce[16] is a programming model that supports scheduling, parallelization and communication and distributed batch processing. C-MR[17] is continuous execution of MapReduce for difficult workflow scheduling by maintaining stream orders with window management on multi core processors in parallel. Temporal subdivisions of stream with size description is defined a window. The interval between windows is called as slide. The architecture of C-MR consists of computing nodes which executes any map or reduce tasks by getting data from shared memory buffers. Prior to assignment of additional tasks to these nodes, they wait for the other nodes until completion of their tasks. A C-MR host process is launched by the computer which C-MR instance is running. As specified by the programmer the Host creates workflow operators whose purpose is to know code and data to be processed from the nodes. It attaches input streams to the workflow of Map and Reduce operations.

The intermediate workflow data is stored in Workflow Buffer which is used by Reduce tasks. Based on the window's slide value boundaries are inserted in the stream by introduced punctuations at ordered stream location. The Scheduler uses either operator scheduling or progressive scheduling technique. When data is requested by a node it acts as an interface. Which operator is to be executed is defined by operator scheduling which uses Oldest Data First (ODF) or Best Memory Tradeoff (MEM) scheduling policies. Progressive scheduling[18] uses multiple policies of scheduling and it is advantages when system resources are scarce. Phoenix++[19], a single host, multi-core MapReduce framework is used in the implementation.

## 4.3 Pilot-MapReduce

PMR[20] is an independent runtime environment on flexible infrastructure for MapReduce. Running one MapReduce application is good on Hadoop, but pipelining of multiple iterative MapReduce tasks is not supported. Data Management between Map and Reduce phases is a critical aspect. Source data is coupled to Map phase computation by Pilot Jobs. Intermediate data are transferred to reduce phase in parallel with Pilot Data. The task of managing scheduling and compute-data placement is addressed by pilot API[21] and it is implemented by BigData (BD) by extending framework Big Job (BJ)[22]. For Data Unit (DU) entities storage space is reserved by

Pilot Data (PD). Metadata management, replications, data movement scheduling is monitored by the BD - Manager. Decentralized intelligence is existed in BD-Agents (physical hosts) in which Master Worker (MW) scheme is used. Compute and data dependencies are called as affinities.

The procedure is as follows:

- Allocation of compute (pilot-Jobs) and data (pilot-data) on different resources is done by MR-Manager.
- Based on the chunk size defined and stored in DU the input data is chunked by Compute Unit (CU) (Chunking).
- A Map CU is assigned to each chunk by the MR - Manager. The appropriate DU is concerned with its CU and it is ensured by BD and BJ (Mapping).
- Sorting and partitioning of data is done. Reduce task process each partition. Each CU is assigned to DU, which is created for each partition. The number of data transfers is reduced by affinity-aware scheduler.
- Intermediate data which is represented by DU are executed by reduce tasks. Based on affinities data management is done by BJ/BD.
- Termination of Pilots.

## 4.4 Duplication Detector

In this age of Internet of Things large volume of data is continuing to outgrow in data centers. Storage, retrieve, capture, computational requirements become challenging tasks. Deduplication[23] is required to address the above challenges. Duplication Detector[24] uses Client side Deduplication[25] where duplicate copies of files are detected at client before sending on to the network, thus reducing the bandwidth requirements. To make file system suitable for Hadoop Distributed File System (HDFS) data files are converted to large (sequence) files in the form of binary key-value pairs where key is file name and its value is contents of the file. By using MapReduce sequence files are splitted. Key/value pair is fetched by Map tasks in the split. Sync Marker indicates the end of the record. MD5 is used to check the duplicates in sequence files and location, message digest key/value pair is stored. Common MD key/value pairs are reduced and combined to location and another list of fingerprints which are same for duplicate copies. If the data is redundant it is not stored else the data is stored and new hash is inserted in hash table. Large file takes more time to generate message digest is the drawback.

## 4.5 MRPrePost

An N-list data structure, vertical database modification where all information required storing association rule mining is use by PrePost algorithm. N-list FIM1[26] is generated by the use of PPC-tree[27] which is constructed by scanning database twice. There are five elements in PPC-Tree (PrePost Code). They are: Item-name, count, children-list, preorder, and post order. Based on N-list frequent item sets are found from the PPC-Tree and deleted, thus to reduce memory overhead. Parallelization of PrePost algorithm is done by using MapReduce, three times. In the first MapReduce[28] phase Map, Reduce and Combine functions are performed by each node on statistic 1-frequent item sets. Infrequent items are cropped based on frequent threshold. PPC-Tree is constructed and N-List frequent one set is generated by traversing in the second MapReduce phase. In the last MapReduce phase frequent item sets are generated by prefix pattern with the use of Round Robin[29]. Load balancing is ensured by distributing N-List to the clusters.

Steps involved in MRPrePost Algorithm are:

- Based on FIM1 which are formed based on frequent threshold by merging the output of map stage where number of items in each data block is counted and F-list is generated.
- According to F-list data block is filtered and frequent data items are sorted by Map function. N-list of 1-frequent item sets are generated by preorder and post order traversals of FP Tree which is constructed by reduce function.
- Shared Cache contains 1-frequent itemsets N-list which are used by all maps. Load balancing of cluster is ensured by group (pi) = i mod m where m represent number of groups of 1-frequent itemsets N-list using Round Robin.
- Frequent itemsets in the group are generated by traversing the PPC Tree which are performed by Map function with the use of current prefix pattern. The output is combined by Reduce function.

## 4.6 vHadoop

vHadoop is a scalable Hadoop Virtual Cluster for processing large scale of data in parallel by MapReduce. The execution flow of Hadoop Virtual Cluster (vHadoop)[30] is as follows:

- A Hadoop virtual cluster request is sent when Machine Learning Algorithm Library which includes clustering, classification etc., is triggered.

**Table 2.** **MapReduce implementations in different scenarios**

| Name of the Approach | Implementation | Framework/ Platform used | Purpose | Additional Algorithms / Methods used | Results |
|---|---|---|---|---|---|
| HCM | MapReduce in iterations | Hadoop enabled Beowulf Cluster | Classification of Large Data Sets | Machine Learning Algorithms -Rule Based Approach | Medical Reports Generation in less time |
| C-MR | Continuous execution of MapReduce tasks | Phoenix++ | Workflow scheduling, Stream ordering | Operator Scheduling, Progressive Scheduling | Latency Reduction |
| PMR | Extensible and Flexible MapReduce | Pilot MapReduce | Data Management and Scheduling | Affinity aware scheduler, Pilot Abstractions | Compute (pilot-Jobs) and data (Pilot-Data) placement, transfers are optimized |
| Duplicate Detector | MapReduce | Hadoop | Deduplication | MD5 Non Cryptographic function | optimization of Storage, computation |
| VHadoop | MapReduce | VHadoop cluster | Parallel Machine learning algorithms | Clustering Algorithms | Efficient Static and Dynamic performance of VHadoop Cluster platform |
| MRPrePost | MapReduce three times | Hadoop | Generation of Frequent itemsets | MRPrePost | Mining association rules on large data sets fastly |

- Hadoop virtual cluster is called and started by virtualization module which supports resource virtualization and live migration[31], energy saving, load balancing, online maintenance is achieved.
- The parameters like names of master and worker nodes, number of map tasks and reduce tasks, number of replicas in DFS, block size is configured by Hadoop Module.
- Uploading of input data to HDFS is performed.
- Map and Reduce functions are performed by assigning to worker nodes.
- Master and worker nodes utilization of CPU, memory, disk status, performance is observed by nmon[32] Monitor. Output data is collected and analyzed.
- Based on the performance monitored by nmon Hadoop virtual cluster parameters are reconfigured by MapReduce Tuner.

## 5. Findings

## 6. Conclusion

The process of duplicate detection is done in parallel using MapReduce to find the duplicates faster thus achieving optimization of storage. MapReduce is modified to maintain stream ordering, scheduling of workflows to reduce the latency in Continuous MapReduce. Pilot MapReduce is a flexible run time environment which is independent of infrastructure used for data management by taking the advantages of pilot abstractions. MRPrePost is used for mining large data's association rules by using parallel algorithm MapReduce on Hadoop platform by adding prefix pattern to PrePost technique thus improving its performance by generating frequent itemsets faster. In MapReduce programming model computations (code) and input data are moved to the cluster for processing the data. Reliability, fault tolerance, replication of data is managed by the distributed file system which supports the environment. We aim to use MapReduce in deduplicating virtual machine files to accelerate the process of live migration.

## 7. References

1. Malleswari TYJN, Malathi D, Vadivu G. A survey of cloud computing, architecture and services provided by various cloud service providers. Proceedings of the 2nd International Conference on Demand Computing (ICODC'12); Bangalore. 2012. p. 201.
2. Kim BS, Kim TG, Song HS. Parallel and distributed framework for standalone Monte Carlo simulation using MapReduce. Indian Journal of Science and Technology. 2015 Oct; 8(25).

3. Yefei T, Lin W, Xiao X. Minimal MapReduce algorithms. Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD'13; 2013 Jun. p. 529–40.
4. Nurain N, Sarwar H, Sajjad MP, Mostakim M. An in-depth study of MapReduce in cloud environment. Proceedings of the Conference on Advanced Computer Science Applications and Technologies; Kuala Lumpur. 2012. p. 263–8.
5. Li F, Ooi BC. Distributed data management using MapReduce. ACM ComputingSurveys. 2014; 46(3):31.
6. Kyoo-Sung N, Doo-Sik L. Bigdata platform design and implementation model. Indian Journal of Science and Technology. 2015 Aug; 8(18).
7. Leskovec J, Rajaraman A, Ullman JD. Mining of massive datasets. UK: Cambridge University Press; 2014.
8. NameNode; 2015. Available from: http://wiki.apache.org/hadoop/NameNode
9. Cardosa M, Wang C, Nangia A, Chandra A, Weissman J. Exploring MapReduce efficiency with highly-distributed data. Proceedings of the 2nd International Workshops on MapReduce and its Applications; 2013. p. 27–34.
10. Dligiannis P, Loidl H-W, Kouidi E. Improving the diagnosis of mild hypertrophic cardiomyopathy with mapreduce. ACM Proceedings of the 3rd International Workshop on MapReduce and its Application, MapReduce'12; 2012 Jun 18-19. p. 41–8.
11. Maron BJ. Distinguishing hypertrophic cardiomyopathy from athlete's heart: A clinical problem of increasing magnitude and significance. Heart. 2005; 91(11):1380–2.
12. Savage N. Better medicine through machine learning. Commun ACM. 2012; 55(1):17–9.
13. Epstein AE, DiMarco JP, Ellenbogen KA, et al. ACC/AHA/HR. Guidelines fordevice based therapy of cardiac rhythm abnormalities. J Am Coll Cardiol. 2008; 51(21):2085–105.
14. Chu C, Kim SK, Lin YA, et al. MapReduce for machine learning on multicore. NIPS. 2007; 281–8.
15. Mitchell TM. Machine Learning. Netherlands: Mc GrawHill; 1997.
16. Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. OSDI, Communications of the ACM 50th Anniversary Issue: 1958-2008; 2004. p. 107–13.
17. Tucker PA, Maier D, Sheard T, Fegaras L. Exploiting punctuation semantics in continuous data streams. IEEE Transactions on Knowledge and Data Engineering. 2003; 15(3):555–68.
18. Backman N, Pattabiraman K, Fonseca R, Centintemel U. C-MR: Continuously executing mapreduce workflows on multi-core processors. Proceedings of 3rd International Workshops on MapReduce and its Applications Date; 2012. p. 1–8.
19. Talbot J, Yoo RM, Kozyraki C. Phoenix++: Modular MapReduce for shared-memory Systems. Proceedings of the 2nd Iternational Workshop on MapReduce and its Applications, MapReduce'11; 2011. p. 9–16.
20. Mantha PK, Lucknow A, Jha S. Pilot-MapReduce: An extensible and flexible MapReduce implementation for distributed data. Proceedings of 3rd International Workshops on MapReduce and its Applications Date; 2012. p. 17–24.
21. Lucknow A, Santcross M, Weider O, Merzky A, Maddineni S, Jha S. Towards a common model for pilot-jobs. Proceedings of the International ACM Symposium on High-Performance Parallel and Distributed Computing; 2012. p. 123–4.
22. SAGA BigJob; 2012. Available from: https://github.com/saga-project/BigJob/wiki
23. Geer D. Reducing the storage burden via data deduplication. Computer, the Flagship Publication of the IEEE Computer Society. 2008; 41(12):15–7.
24. Sethi P, Kumar P. Leveraging hadoop framework to develop duplicationdetector and analysis using MapReduce, Hive and Pig. 2014 7th International Conference on Contemporary Comnputing (IC3); 2014. p. 454–60.
25. Malleswari TYJN, Malathi D, Vadivu G. Deduplication techniques: A technical survey. International Journal for Innovative Research in Science Technology. 2014; 1(7):318–25.
26. Liao J, Zhao Y, Long S. MRPrePost- A parallel algorithm for miningbig data. IEEE Workshops on Electronics, Computer and Applications; Ottawa, ON. 2014. p. 564–8.
27. Assuncao J, Fernandes P, Lopes L, et al. Distributed stochastic aware random forests-efficient data mining for big data. International Congress on Big Data; Santa Clara, CA. 2013. p. 425–6.
28. Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Communications of the ACM 50th Anniversary Issue 1958-2008. 2008; 51(1):107–13.
29. Moens S, Aksehirli E, Goethals B. Frequent itemset mining for big data. IEEE International Conference on Big Data; Silicon Valley, CA. 2013 Oct 6-9. p. 111–8.
30. Ye K, Jiang X, He Y, Li X, Yan H, Huang P. vHadoop: A scalable H platform for MapReduce-based parallel machine learning with perfomance consideration. IEEE International Conference on Cluster Computing Workshops; 2012. p. 152–60.
31. Malleswari TYJN, Malathi D, Vadivu G. Live virtual machine migrationtechniques-A technical survey. Advances in Intelligent Systems and Computing. 2014; 308:303–19.
32. nmon for Linux; 2015. Available from: http://nmon.sourceforge.net/