Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.663 – 674

RESEARCH ARTICLE

Mapreduce Algorithms Optimizes the Potential of Big Data

Lalit Malik
Student, M.Tech, SBMNEC, Rohtak
Malik.kabir22@gmail.com

Sunita Sangwan
Prof, CSE Dept, SBMNEC, Rohtak

ABSTRACT: ***Today, we"re surrounded by data like the air. The exponential growth of data first presented challenges to cutting-edge businesses such as whatsapp, Google, Yahoo, Amazon, Microsoft, Facebook, Twitter etc. Data volumes to be processed by cloud applications are growing much faster than computing power. This growth demands new strategies for processing and analyzing information. Hadoop- MapReduce has become a powerful Computation Model addresses to these problems. Hadoop HDFS became more popular amongst all the Big Data tools as it is open source with flexible scalability, less total cost of ownership & allows data stores of any form without the need to have data types or schemas defined. Hadoop MapReduce is a programming model and software framework for writing applications that rapidly process vast amounts of data in parallel on large clusters of compute nodes. In this paper I have provided an overview, architecture and components of Hadoop, HCFS (Hadoop Cluster File System) and MapReduce programming model, its various applications and implementations in Cloud Environments.***
Keywords: ***Mapreduce, HDFS, DFS, Hive, Pig, Hadoop***

INTRODUCTION

"90% of the world's data was generated in the last few years."

Due to the advent of new technologies, devices, and communication means like social networking sites, the amount of data produced by mankind is growing rapidly every year. The amount of data produced by us from the beginning of time till 2003 was 5 billion gigabytes. If you pile up the data in the form of disks it may fill an entire football field. The same amount was created in every two days in 2011, and in every ten minutes in 2013. This rate is still growing enormously. Though all this information produced is meaningful and can be useful when processed, it is being neglected.

Cloud computing has been driven fundamentally by the need to process an exploding quantity of data in terms of exabytes as we are approaching the Zetta Byte Era. One critical trend shines through the cloud is Big Data. Indeed, it"s the core driver in cloud computing and will define the future of IT. When a company needed to store and access more data they had one of two choices. One option would be to buy a bigger machine with more CPU, RAM, disk space, etc. This is known as scaling vertically. Of course, there is a limit to how big of a machine you can actually buy and this does not work when you start

talking about internet scale. The other option would be to scale horizontally. This usually meant contacting some database vendor to buy a bigger solution. These solutions do not come cheap and therefore required a significant investment. Today, the source of data generated not only by the users and applications but also "machine-generated," and such data is exponentially leading the change in the Big Data space. Dealing with big datasets in the order of terabytes or even petabytes is a challenging. In Cloud computing environment a popular data processing engine for big data is Hadoop-MapReduce due to ease-of-use, scalability, and failover properties. Hadoop-MapReduce programming model consists of data processing functions: Map and Reduce. Parallel Map tasks are run on input data which is partitioned into fixed sized blocks and produce intermediate output as a collection of pairs. These pairs are shuffled across different reduce tasks based on pairs. Each Reduce task accepts only one key at a time and process data for that key and outputs the results as pairs. The Hadoop-MapReduce architecture consists of one Job Tracker (also called as Master) and many Task Trackers (also called as Workers). The Job Tracker receives job submitted from user, breaks it down into map and reduce tasks, assigns the tasks to Task Trackers, monitors the progress of the Task Trackers, and finally when all the tasks are complete, reports the user about the job completion. Each Task Tracker has a fixed number of map and reduce task slots that determine how many map and reduce tasks it can run at a time. HDFS supports reliability and fault tolerance of MapReduce computation by storing and replicating the inputs and outputs of a Hadoop job.

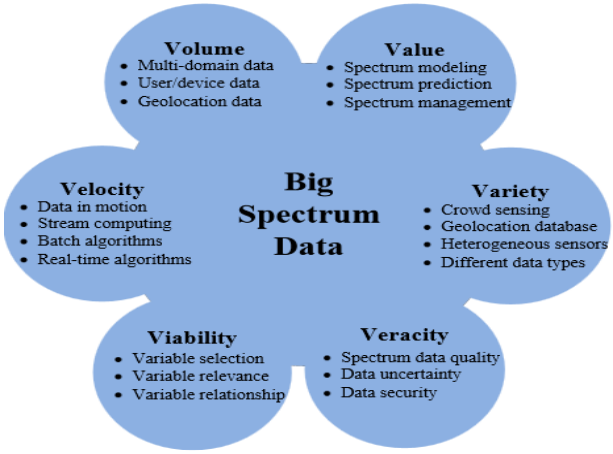


Figure 1 : Big data block diagram

BIG DATA

Big data is a collection of data sets so large and complex which is also exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn‟t fit the structures of our current database architectures. Big Data is typically large volume of un-structured (or semi structured) and structured data that gets created from various organized and unorganized applications, activities and channels such as emails, tweeter, web logs, Facebook, etc. The main difficulties with Big Data include capture, storage, search, sharing, analysis, and visualization. The core of Big Data is Hadoop which is a platform for distributing computing problems across a number of servers. It is first developed and released as open source by Yahoo, it implements the MapReduce approach pioneered by Google in compiling its search indexes. Hadoop‟s MapReduce involves distributing a dataset among multiple servers and operating on the data: the "map" stage. The partial results are then recombined: the "reduce" stage. To store data, Hadoop utilizes its own distributed filesystem, HDFS, which makes data available to multiple computing nodes.

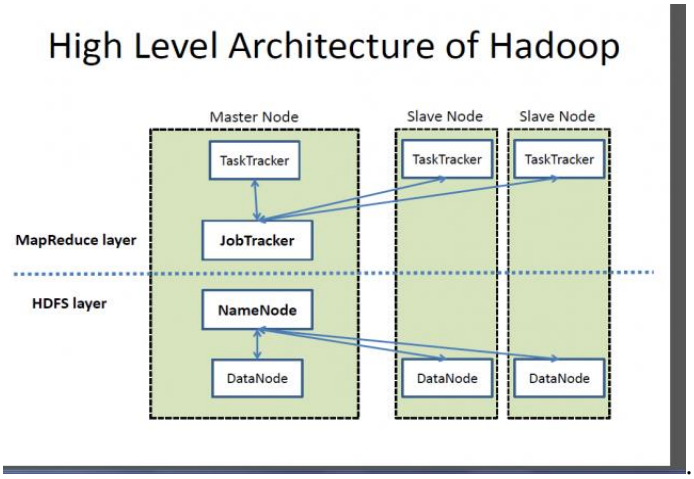
Big data explosion, a result not only of increasing Internet usage by people around the world, but also the connection of billions of devices to the Internet. Eight years ago, for example, there were only around 5 exabytes of data online. Just two years ago, that amount of data passed over the Internet over the course of a single month. And recent estimates put monthly Internet data flow at around 21 exabytes of data. This explosion of data - in both its size and form - causes a multitude of challenges for both people and machines.

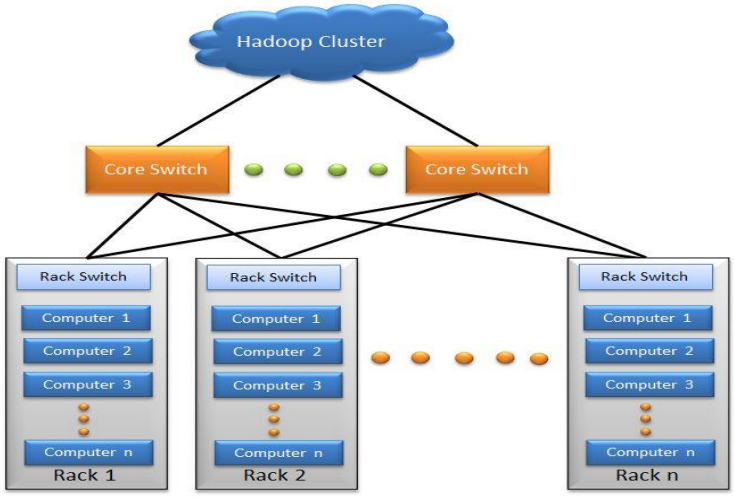
HADOOP

Hadoop is a batch processing system for a cluster of nodes that provides the underpinnings of most BigData analytic activities because it bundle two sets of functionality most needed to deal with large unstructured datasets nsmely, Distributed file system and MapReduce processing. it is a project from the Apache Software Foundation written in Java to support data intensive distributed applications. Hadoop enables applications to work with thousands of nodes and petabytes of data. The inspiration comes from Google‟s MapReduce and Google File System papers. Hadoop‟s biggest contributor has been the search giant Yahoo, where Hadoop is extensively used across the business platform. Hadoop is an umbrella of subprojects around distributed computing and although is best known for being a runtime environment for MapReduce programs and its distributed filesystem HDFS, the other sub-projects provide complementary services and higher level abstractions. Some of the current sub-projects are:

Core

The Hadoop core consist of a set of components and interfaces which provides access to the distributed filesystems and general I/O (Serialization, Java RPC, Persistent data structures). The core components also provide "Rack Awareness", an optimization which takes into account the geographic clustering of servers, minimizing network traffic between servers in different geographic clusters.





MapReduce

Hadoop MapReduce is a programming model and software framework for writing applications that rapidly process vast amounts of data in parallel on large clusters of computer nodes. MapReduce uses the HDFS to access file segments and to store reduced results.

HDFS

Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop applications. HDFS is, as its name implies, a distributed file system that provides high throughput access to application data creating multiple replicas of data blocks and distributing them on compute nodes throughout a cluster to enable reliable and rapid computations.

HBase: HBase is a distributed, column-oriented database. HBase uses HDFS for its underlying storage. It maps HDFS data into a database like structure and provides Java API access to this DB. It supports batch style computations using MapReduce and point queries (random reads). HBase is used in Hadoop when random, realtime read/write access is needed. Its goal is the hosting of very large tables running on top of clusters of commodity hardware.

Pig:

It is a dataflow processing (scripting) language Apache Pig is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs. The main characteristic of Pig programs is that their structure can be substantially parallelized enabling them to handle very large data sets, simple syntax and advanced built-in functionality provide an abstraction that makes development of Hadoop jobs quicker and easier to write than traditional Java MapReduce jobs.

ZooKeeper:

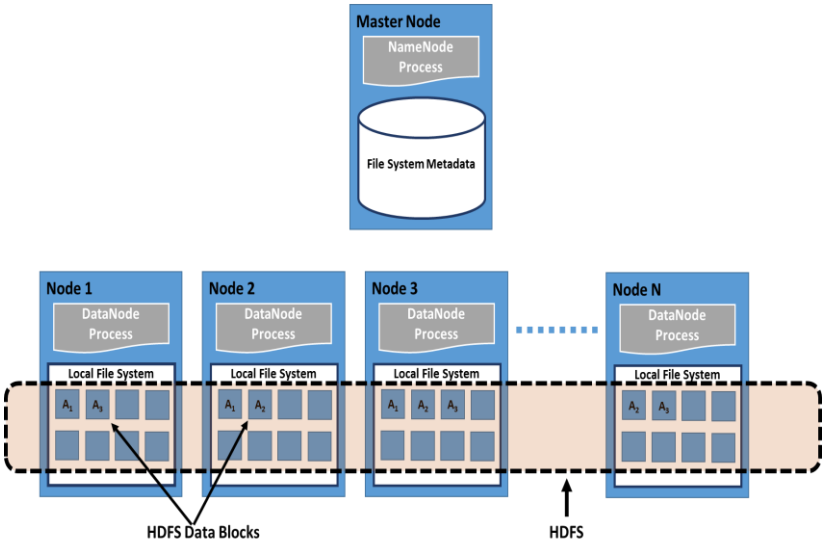
It is a cluster configuration tool and distributed serialization manager useful to build large clusters of Hadoop nodes, high performance coordination service for distributed applications. ZooKeeper centralizes the services for maintaining the configuration information, naming, providing distributed synchronization, and providing group services.

Hive

Hive is a data warehouse infrastructure built on top of Hadoop. Hive provides tools to enable easy data summarization, ad-hoc querying and analysis of large datasets stored in Hadoop files. It provides a mechanism to put structure on this data and it also provides a simple query language called Hive QL, based on SQL, enabling users familiar with SQL to query this data.

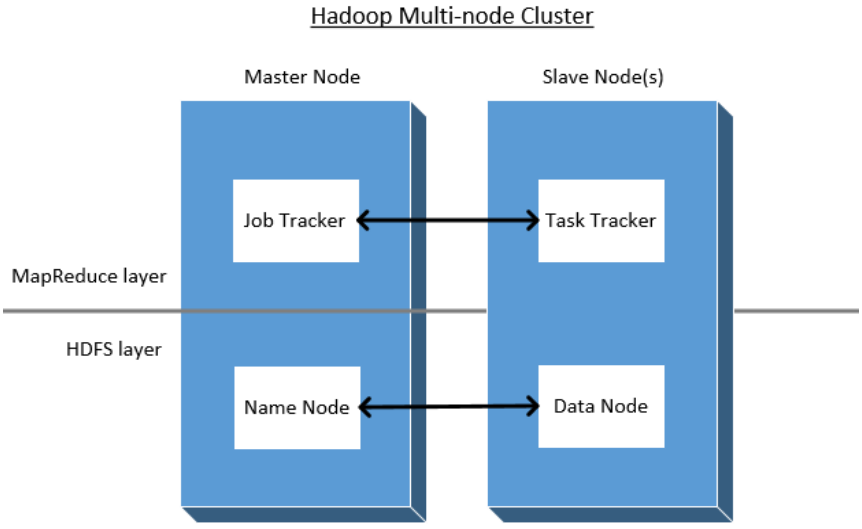
Chukwa

It is used for monitoring large distributed clusters of servers. It is a data collection system for monitoring large distributed systems. Chukwa includes a flexible and powerful toolkit for displaying, monitoring and analyzing results to make the best use of the collected data.



HCatalog

It is a storage management layer for Hadoop that enables users with different data processing tools. HCatalog‟s table abstraction presents users with a relational view of data in the Hadoop distributed file **A)**



Architecture of Hadoop

Hadoop is a Map/Reduce framework that works on HDFS or on HBase. The main idea is to decompose a job into several and identical tasks that can be executed closer to the data (on the DataNode). In addition, each task is parallelized : the Map phase. Then all these intermediate results are merged into one result : the Reduce phase. In Hadoop, The JobTracker (a java process) is responsible for monitoring the job, managing the Map/Reduce phase, managing the retries in case of errors. The

TaskTrackers (Java process) are running on the different DataNodes. Each TaskTracker executes the tasks of the job on the locally stored data.

The core of the Hadoop Cluster Architecture is given below.

HDFS

(Hadoop Distributed File System): HDFS is the basic file storage, capable of storing a large number of large files.

MapRedure:

MapReduce is the programming model by which data is analyzed using the processing resources within the cluster. Each node in a Hadoop cluster is either a master or a slave. Slave nodes are always both a Data Node and a Task Tracker. While it is possible for the same node to be both a Name Node and a JobTracker

Name Node: Manages file system metadata and access control. There is exactly one Name Node in each cluster.

Secondary Name Node
Downloads periodic checkpoints from the name Node for fault-tolerance. There is exactly one Secondary Name Node in each cluster.

Job Tracker
Hands out tasks to the slave nodes. There is exactly one Job Tracker in each cluster.

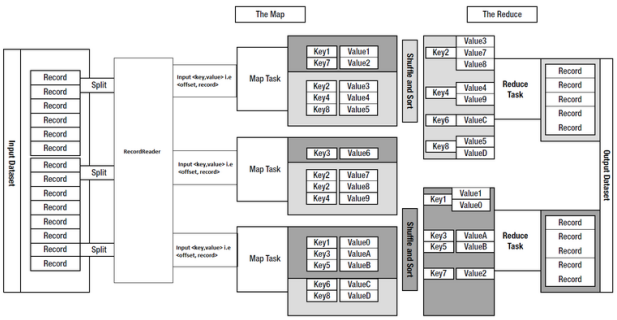
Data Node
Holds file system data. Each data node manages its own locally-attached storage and stores a copy of some or all blocks in the file system. There are one or more Data Nodes in each cluster.

Task Tracker
Slaves that carry out map and reduce tasks. There are one or more Task Trackers in each cluster.

Uses of Hadoop
♣ Building search index at Google, Amazon, Yahoo ♣ Analyzing user logs, data warehousing and analytics ♣ Used for large scale machine learning and data mining applications ♣ Legacy data processing where it requires massive computational.

MAP-REDUCE

MapReduce is a data processing or parallel programming model introduced by Google. In this model, a user specifies the computation by two functions, Map and Reduce. In the mapping phase, MapReduce takes the input data and feeds each data element to the mapper. In the reducing phase, the reducer processes all the outputs from the mapper and arrives at a final result. In simple terms, the mapper is meant to filter and transform the input into something that the reducer can aggregate over. The underlying MapReduce library automatically parallelizes the computation, and handles complicated issues like data distribution, load balancing and fault tolerance. Massive input, spread across many machines, need to parallelize. Moves the data, and provides scheduling, fault tolerance. The original MapReduce implementation by Google, as well as its open-source counterpart, Hadoop, is aimed for parallelizing computing in large clusters of commodity machines. Map Reduce has gained a great popularity as it gracefully and automatically achieves fault tolerance. It automatically handles the gathering of results across the multiple nodes and returns a single result or set. MapReduce model advantage is the easy scaling of data processing over multiple computing nodes



MapReduce Architecture & Implementation Programming model

Fault tolerance

MapReduce is designed to be fault tolerant because failures are common phenomena in large scale distributed computing and it includes worker failure and master failure.

Worker failure

The master pings every mapper and reducer periodically. If no response is received for a certain amount of time, the machine is marked as failed. The ongoing task and any tasks completed by this mapper will be re-assigned to another mapper and executed from the very beginning. Completed reduce tasks do not need to be re-executed because their output is stored in the global file system.

Master failure

Since the master is a single machine, the probability of master failure is very small. MapReduce will re-start the entire job if the master fails. There are currently three popular implementations of the MapReduce programming model namely Google MapReduce, Apache Hadoop, Stanford Phoenix

Execution Process in MapReduce Programming Model

In MapReduce programming model and a MapReduce job consists of a map function, a reduce function, and When a function called the below steps of actions take place.

♣ MapReduce will first divide the data into N partitions with size varies from 16MB to 64MB

♣ Then it will start many programs on a cluster of different machines. One of program is the master program; the others are workers, which can execute their work assigned by master. Master can distribute a map task or a reduce task to an idle worker.

♣ If a worker is assigned a Map task, it will parse the input data partition and output the key/value pairs, then pass the pair to a user defined Map function. The map function will buffer the temporary key/value pairs in memory. The pairs will periodically be written to local disk and partitioned into P pieces. After that, the local machine will inform the master of the location of these pairs.

♣ If a worker is assigned a Reduce task and is informed about the location of these pairs, the Reducer will read the entire buffer by using remote procedure calls. After that, it will sort the temporary data based on the key.

♣ Then, the reducer will deal with all of the records. For each key and according set of values, the reducer passes key/value pairs to a user defined Reduce function. The output is the final output of this partition.

♣ After all of the mappers and reducers have finished their work, the master will return the result to users' programs. The output is stored in F individual files

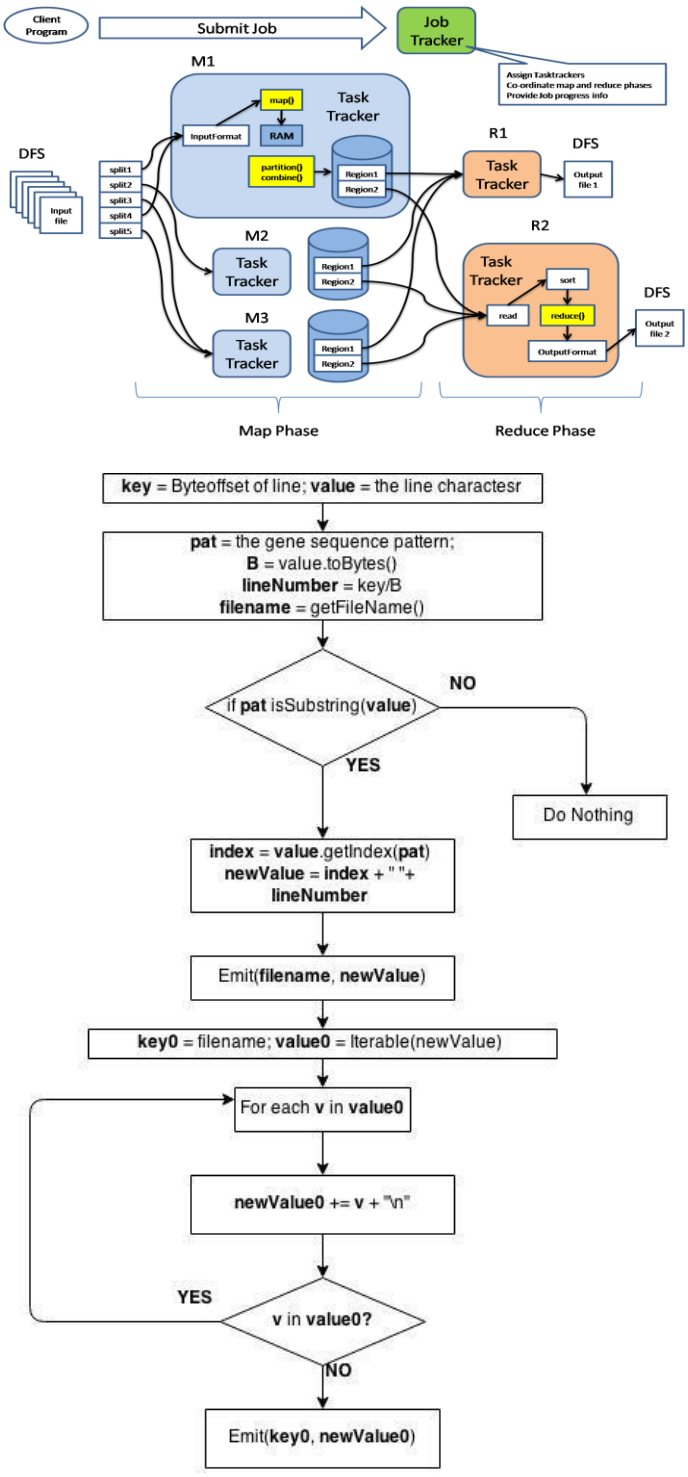
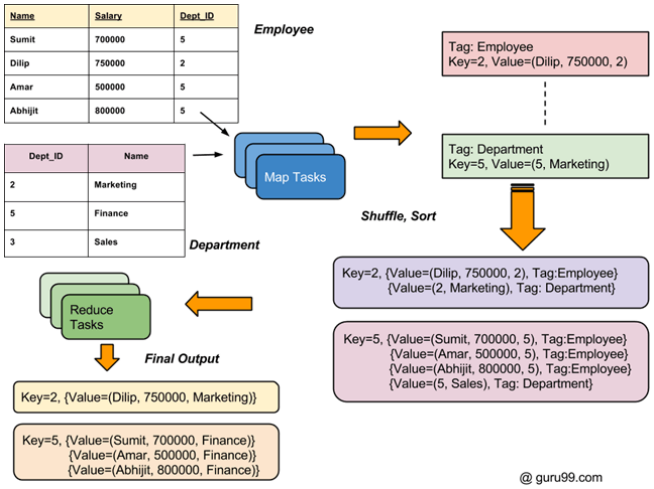


Fig: Flow Chart For Mapreduce



A MapReduce Progamming Model Example:

In essence MapReduce is just a way to take a big task and split it into discrete task that can be done in parallel. A simple problem that is often used to explain how MapReduce works in practice consists in counting the occurrences of single words within a text. This kind of problem can be easily solved by launching a single MapReduce job as given in the below figure.

♣ Input data

♣ Input data are partitioned into smaller chunks of data

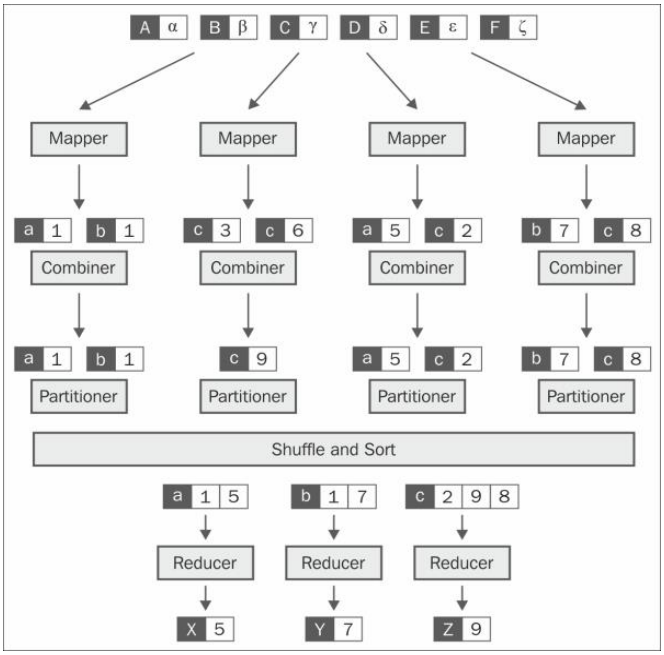
♣ For each chunk of input data, a "map task" runs which applies the map function resulting output of each map task is a collection of key-value pairs.

♣ The output of all map tasks is shuffled for each distinct key in the map output; a collection is created containing all corresponding values from the map output.

♣ For each key-collection resulting from the shuffle phase, a "reduce task" runs which applies the reduce function to the collection of values.

♣ The resulting output is a single key-value pair.

♣ The collection of all key-value pairs resulting from the reduce step is the output of the MapReduce job.



Advantages of the MapReduce

Using MapReduce, a programmer defines his job with only Map and Reduce functions, without having to specify physical distribution of his job across nodes

Flexible

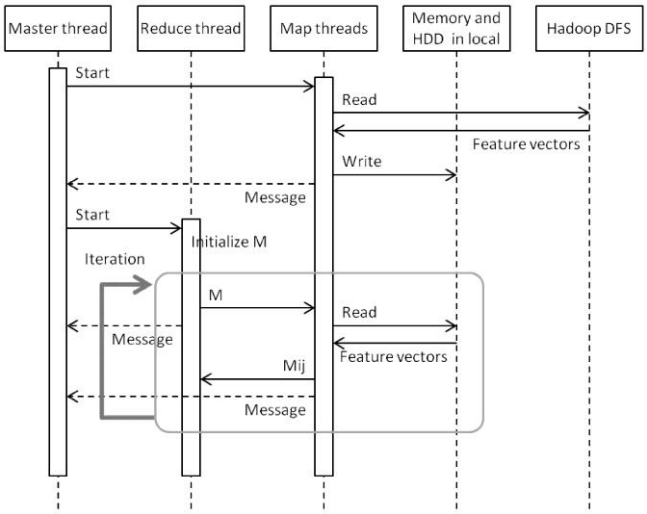
MapReduce does not have any dependency on data model and schema.

Fault tolerance

MapReduce is highly fault-tolerant.

High scalability

The best advantage of using MapReduce is high scalability.



CONCLUSION

Hadoop-MapReduce programming paradigm have a substantial base in the Big Data community due to the cost-effectiveness on commodity Linux clusters, and in the cloud via data upload to cloud vendors who have implemented Hadoop/HBase. The effectiveness and ease-of-use of the MapReduce method in parallelization involves many data analysis algorithms. HDFS, the Distributed File System, is a distributed file system designed to hold very large amounts of data (terabytes or even petabytes), and provide high-throughput access to these information

Hadoop MapReduce is a large scale, open source software framework dedicated to scalable, distributed, data-intensive computing

• The framework breaks up large data into smaller parallelizable chunks and handles scheduling
- ▫ Maps each piece to an intermediate value
- ▫ Reduces intermediate values to a solution
- ▫ User-specified partition and combiner options

• Fault tolerant, reliable, and supports thousands of nodes and petabytes of data

• If you can rewrite algorithms into Maps and Reduces, and your problem can be broken up into small pieces solvable in parallel, then Hadoop's MapReduce is the way to go for a distributed problem solving approach to large datasets

• Tried and tested in production

• Many implementation options

REFERENCES

[1]JJ. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. Commun. ACM, 51(1):107– 113, 2008.

[2] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In EuroSys ‟10: IJ-CLOSER ISSN: 2089-3337 ρ Big Data Processing with Hadoo-Map Reduce in Cloud Systems (Rabi Prasad Padhy) 27 Proceedings of the 5th European conference on Computer systems, pages 265–278, New York, NY, USA, 2010. ACM.

[3] R. Taylor. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics BMC bioinformatics,11(Suppl 12):S1, 2010.
[4] A. Pavlo et al . A comparison of approaches to large-scale data analysis. In Proceedings of the ACM SIGMOD, pages 165–178, 2009.
[5] C. Ordonez et al . Relational versus Non-Relational Database Systems for Data Warehousing. In Proceedings of the ACM DOLAP, pages 67–68, 2010.
[6] F. Chang et al . Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2):1–26, 2008.
[7] S. Melnik et al . Dremel: interactive analysis of web-scale datasets. Proceedings of the VLDB Endowment,3(1- 2):330–339, 2010.
[8] Cassandra Query Language Documentation. http://caqel.deadcafe.org/cql-doc. [online; accessed 25-July-2012].
[9] HBase: Bigtable-like structured storage for Hadoop HDFS. http://wiki.apache.org/hadoop/hbase. [online; accessed 26-July-2012.
[10] S. Ghemawat et al . The google file system. ACM SIGOPS Operating Systems Review, 37(5):29–43, 2003.
[11] HDFS (hadoop distributed file system) architecture. http://hadoop.apache.org/common/docs/current/hdfs
[12] TCP/IP Implementation of Hadoop Acceleration by Cong Xu A thesis submitted to the Graduate Faculty of Auburn University in partial fulfillment of the requirements for the Degree of Master of Science Auburn, Alabama Aug 4, 2012
[13] R. Lammel. Google‟s mapreduce programming model – revisited. Science of Computer Programming, 70(1):1 – 30, 2008.
[14] A Workload Model for MapReduce by Thomas A. de Ruiter, Master‟s Thesis in Computer Science, Parallel and Distributed Systems Group Faculty of Electrical Engineering, Mathematics, and Computer Science. Delft University of Technology, 2nd June 2012..
[15] H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. Map-reduce-merge: simplified relational data processing on large clusters. In SIGMOD „07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pages 1029–1040, New York, NY, USA, 2007. ACM.
[16] T. Condie et al. MapReduce online. In Proceedings of the 7th USENIX conference on Networked systems design and implementation, pages 21–21, 2010.
[17] W. Jiang et al . A Map-Reduce System with an Alternate API for Multi-core Environments. In Proceedings of the 10th IEEE/ACM CCGrid, pages 84–93, 2010.
[18] H. Yang et al. Map-reduce-merge: simplified relational data processing on large clusters. In Proceedings of the 2007 ACM SIGMOD, pages 1029–1040, 2007. “Map-Reduce Online”; Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein UC Berkeley, Khaled Elmeleegy, Russell Sears Yahoo! Research
[19] M. Grant, S. Sehrish, J. Bent, and J. Wang. “Introducing Map-reduce to High End Computing”. 3rd Petascale Data Storage Workshop, Nov 2008.
[20] Y. Hung-Chih, D. Ali, H. Ruey-Lung, and D.S. Parker, “Map-Reduce-Merg e: Simplified Relation al Data Processing On Large Clusters”, in Proceedings of the 2007 ACM Sigmod International Conference on Management of Data Beijing”, China: acm, 2007
[21] G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. Wang, “Introducing Map-reduce to High End Computing”, in Petascale Data Storage Workshop”, 2008. pdsw „08. 3rd, 2008, pp. 1-6.
[22] Amdahl, G. (1967). Validity of the single processor approach to achieving large-scale computing capabilities (pp. 483–485).
[23] Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 30(1–7), 107–117.
[24] Cacheda, F., Plachouras, V., & Ounis, I. (2005). A case study of distributed information retrieval architectures to index one terabyte of text. Information Processing & Management, 41(5), 1141–1161.
[25] I. Adams, D.D.E. Long, E.L. Miller, S. Pasupathy, M.W. Storer, Maximizing efficiency by trading storage for computation, in: Workshop on Hot Topics in Cloud Computing, HotCloud„09, San Diego, CA, 2009, pp. 1–5.
[26] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (2009) 599– 616.
[27] Dhruba Borthaku, The Hadoop Distributed File System: Architecture and Design. Retrieved from, 2010, http://hadoop.apache.org/common/
[28] McInnes C., Virtual screening strategies in drug discovery. Current Opinion in Chemical Biology, 2007, 11:494-502. [CrossRef]
[29] Drug discovery of anti-H5N1 and other influenza virus entry inhibitors, http://otm.uic.edu/technologies/drug-discovery-anti-h5n1-and-other-influenza- virus/.

[30] Keyin Ruan, Ruisheng Zhang, Fan Ding, Lian Li, A User-Friendly Task Editor Environment for Large-scale Virtual Screening Application.In Proceedings of 9th International Conference Grid and Cooperative Computing (GCC). 2010. Abstract | Full Text: PDF (438KB) | Full Text: HTML
[31] Yunxia Pei, Yue Zhang, The map-reduce parallelism framework for task scheduling in grid computing.Advanced Materials Research, v 216, p 111-115, 2011, Optical, Electronic Materials and Applications. [CrossRef]
[32] DEAN, J., AND GHEMAWAT, S. MapReduce: Simplied data processing on large clusters. In Proc. of the 6th OSDI (Dec. 2004), pp. 137.150.
[33] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, et al., Bigtable: A Distributed Storage System for Structured Data. OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November, 2006.
[34] Apache Hadoop, http://hadoop.apache.org/.
[35] Dhruba Borthakur, HDFS Architecture.The Apache Software Foundation, 2008.
[36]Ankur Khetrapal, Vinay Ganesh, HBase and Hypertable for large scale distributed storage systems. Computer Science, 2006.
[37] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, The Google File System. SOSP'03, October 19-22, Bolton Landing, New York, USA, 2003. [CrossRef]
[38]Addis, B. and Schoen, F. (2003), A randomized global optimization method for protein-protein docking. Technical Report DSI 4-2003.
[39] Maxim Totrov, Ruben Abagyan, Flexible ligand docking to multiple receptor conformations: a practical alternative. Current Opinion in Structural Biology, Volume 18, Issue 2, April 2008, Pages 178-184. [CrossRef]
[40] Vieth, M., Hirst, J.D., Kolinski, A. & Brooks, C.L.I.Assessing energy functions for flexible docking.J Comp Chem 19, 1612-1622 (1998). [CrossRef]
[41] Molecular docking, http://en.wikipedia.org/wiki/Docking(molecular)/.
[42] DOCK:http://dock.compbio.ucsf.edu/.
[43] Buzko OV, Bishop AC, Shokat KM, Modified AutoDock for accurate docking of protein kinase inhibitors.J Comput Aided Mol Des 2002, 16:113-127.
[44] Kramer B., Rarey M, Lengauer T: Evaluation of the FLEXX incremental construction algorithm for protein-ligand docking. Proteins 1999, 37:228-241. [CrossRef]
[45] Sally R. Ellingson, Jerome Baudry, High-Throughput Virtual Molecular Docking:Hadoop Implementation of AutoDock4 on a Private Cloud.ECMLS'11, June 8, 2011, San Jose, California, USA. [CrossRef]
[46] VARIA J. Cloud architectures-Amazon Web Service [EB/OL]. [2009- 03- 01].http://acmbangalore.org/events/monthly2talk/ may2200822cloud2architectures222amazon2web2services. html/.
[47] Yuan Luo, Zhenhua Guo, Yiming Sun, Beth Plale, Judy Qiu, Wilfred W. Li, A Hierarchical Framework for Cross-Domain MapReduce Execution. [CrossRef]
[48] P. Therese Lang, Scott R. Brozell, DOCK 6: Combining techniques to model RNA-small molecule complexes.2009.
[49] Yuki Miyamoto, Junji Yamauchi, Atsushi Sanbe, Akito Tanoue, Dock6, a Dock-C subfamily guanine nucleotide exchanger, has the dual specificity for Rac1 and Cdc42 and regulates neurite outgrowth. Experimental Cell Research, Volume 313, Issue 4, 15 February 2007, Pages 791-804. [CrossRef]
[50] ZINC, A free database for virtural screening, http://zinc.docking.org/.

