International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161

Available at http://inpressco.com/category/ijcet

General Article

A Survey of Big Data Processing in Perspective of Hadoop and Mapreduce

D.Usha[Á*] and Aslin Jenil A.P.S[Á]

[A]Hindustan University, Chennai

Accepted 05 March 2014, Available online 01 April 2014, **Vol.4, No.2 (April 2014)**

Abstract

Big Data is a data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it. Hadoop is the core platform for structuring Big Data, and solves the problem of making it useful for analytics purposes. Hadoop is an open source software project that enables the distributed processing of large data sets across clusters of commodity servers. It is designed to scale up from a single server to thousands of machines, with a very high degree of fault tolerance. Hadoop MapReduce is an implementation of the algorithm developed and maintained by the Apache Hadoop project. Map Reduce is a programming model for processing large data sets with parallel distributed algorithm on cluster. This paper presents the survey of bigdata processing in perspective of hadoop and mapreduce.

Keywords: *Big data, Hadoop, Mapreduce, HDFS.*

1. Introduction

Big Data is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it. Traditional databases analytics says what happened and what is happening, however gives the predictive analysis of what is likely to happen in future. Infrastructure requirements of big are data acquisition, data organization and data analysis. Hadoop is the open source software founded by Apache. It is a software framework for processing large datasets. Hadoop Distributed File System (HDFS) for storage and MapReduce for processing are the two components of Hadoop.

MapReduce is a programming for processing large datasets. MapReduce works with 2 functions: Map and Reduce function. The Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce function. The Reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges these values to form a possibly smaller set of values.

This paper is organized as follows: Section II describes what big data is, how the big data differ from the traditional data and the infrastructure management of big data. Section III explains about Hadoop and its architecture. Section IV briefs the MapReduce and its programming model.

2. Big Data

Big data refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze. The Potential sources of big data are:

Traditional enterprise data includes customer information from CRM systems, Transactional ERP data, Web store transactions, and General ledger data. Machine-generated /sensor data includes Call Detail Records (CDR), weblogs, smart meters, manufacturing sensors, equipment logs (often referred to as digital exhaust), and trading systems data. Social data includes customer feedback streams, micro-blogging sites like Twitter, social media platforms like Facebook. Other sources of data are Health care, Public sector, Retail and Manufacturing.

Big Data requires scalable technologies to efficiently process large quantities of data within tolerable elapsed times. Big Data scalable technologies include MPP databases, the Apache Hadoop Framework, the Internet, and archival storage systems.

2.1 3 V's of Big Data

Volume of data

Volume refers to amount of data. Volume of data stored in enterprise repositories have grown from megabytes and gigabytes to petabytes.

Variety of data

Different types of data and sources of data. Data variety exploded from structured and legacy data stored in

*Corresponding author: **D.Usha**

enterprise repositories to unstructured, semi structured, audio, video, XML etc.

Velocity of data

Velocity refers to the speed of data processing. For time-sensitive processes such as catching fraud, big data must be used as it streams into your enterprise in order to maximize its value.

2.2 Traditional DBMS vs Big Data

MapReduce is complementary to DBMS, not a competing technology.

- Parallel DBMS are for efficient querying of large data sets.
- MR-style systems are for complex analytics and ETL tasks.
- Parallel DBMS require data to fit into the relational paradigm of rows and columns.
- In contrast, the MR model does not require that data files adhere to a schema defined using the relational data model. That is, the MR programmer is free to structure their data in any manner or even to have no structure at all.

2.3 Pillars of Big Data

1. Big Table – Relational, Tabular format – rows & columns.
2. Big Text – All kinds of unstructured data, natural language, grammatical data, semantic data.
3. Big Metadata – Data about data, taxonomies, glossaries, facets, concepts, entity.
4. Big Graphs – object connections, semantic discovery, degree of separation, linguistic analytic, subject predicate object.

2.4 Infrastructure Requirements Of Big Data

Data acquisition in Big Data

Even though the data will be in distributed environment, infrastructure must support to carry out very high transaction volumes and also support flexible data structures. To collect and store data, NoSQL are often used in Big data. NoSQL will not have any fixed schema since it supports high variety of data by capturing all types of data. Keys are used to identify the data point without designing schema with relationship between entities.

Data organization in Big Data

In the classical term of data warehousing, organizing data is called as data integration. Big data requires good infrastructure, so that processing and manipulating data in the original storage location can be done easily. It must also supports very high throughput to deal with processing steps of large data and handles large variety of data formats like structured format, unstructured format etc. Hadoop is a new technology that allows large data volumes to be organized and processed while keeping the data on the original data storage cluster. For example Hadoop Distributed File System (HDFS) is the long - term storage system for web logs. These web logs are turned into browsing behavior (sessions) by running MapReduce programs on the cluster and generating aggregated results on the same cluster. These aggregated results are then loaded into a Relational DBMS system.

Data analysis in Big Data

Since data is not always moved during the organization phase, the analysis may also be done in a distributed environment, where some data will stay where it was originally stored and be transparently accessed from a data warehouse. The infrastructure required for analyzing big data must be able to support deeper analytics such as statistical analysis and data mining, on a wider variety of data types stored in diverse systems; scale to extreme data volumes; deliver faster response times driven by changes in behavior; and automate decisions based on analytical models. Most importantly, the infrastructure must be able to integrate analysis on the combination of big data and traditional enterprise data. New insight comes not just from analyzing new data, but from analyzing it within the context of the old to provide new perspectives on old problems. For example, analyzing inventory data from a smart vending machine in combination with the events calendar for the venue in which the vending machine is located, will dictate the optimal product mix and replenishment schedule for the vending machine.

4. What is hadoop?

Hadoop was founded by Apache. It is a Open source software project written in Java. It is used to optimize massive volume of data. It is a software framework for distributed processing of large dataset across large clusters of commodity servers. It is implemented for Google MapReduce as a open source. Hadoop is based on simple programming model called MapReduce. It provides reliability through replication. Hadoop is complement to OLTP & OLAP.

Characteristics of Hadoop

Scalable– As required new nodes can be added without changing the data formats, the way which data is loaded, way the jobs or application are written.

Cost effective– Hadoop brings massively parallel computing to commodity servers. The output is a sizeable decrease in cost which in turn makes it affordable to model all the data.

Flexible– Hadoop is not schema oriented so it can absorb any type of data, structured or not, from number of sources. Data from multiple sources can be joined and aggregated in arbitrary ways enabling deeper analysis than any other system can provide.

Fault tolerant– When a node was lost, the system redirects work to another location of the data and continues processing without missing a beat.

Hadoop consists of 2 components:

- HDFS (Hadoop Distributed File System) for

storage.

- Map Reduce for Processing

3.1 Hadoop Architecture

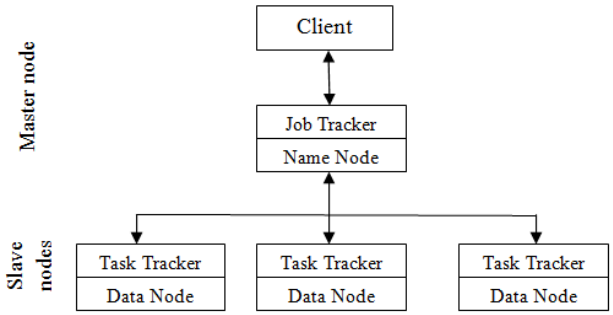

Fig.3.1 Hadoop Master/Slave Architecture

HDFS is one of the primary component of Hadoop cluster and it is designed like a Master-slave architecture. The Master (NameNode) manages the file system namespace operations like opening, closing, renaming files and directories and also determines the mapping of blocks to DataNodes along with regulating access to files by clients. Slaves (DataNodes) are responsible for serving read and write request from the clients along with performing block creation, deletion, and replication upon instruction from the Master (NameNode).

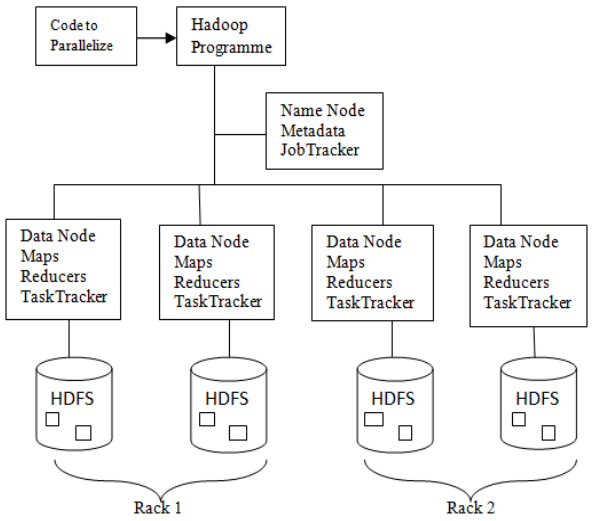

Fig. 3.2 HDFS Architecture

When a client makes a request of a Hadoop cluster, this request is managed by the JobTracker. The JobTracker, working with the NameNode, distributes work as closely as possible to the data on which it will work. The NameNode is the master of the file system, providing metadata services for data distribution and replication. The JobTracker schedules map and reduce tasks into available slots at one or more TaskTrackers. The TaskTracker working with the DataNode (the slave portions of the distributed file system) to execute map and reduce tasks on data from the DataNode. When the map and reduce tasks are complete, the TaskTracer notifies the JobTracker, which identifies when all tasks are complete and eventually notifies the client of job completion.

Master {Jobtracker} is the point of interaction between users and the map/reduce framework. When a map/reduce job is submitted, Jobtracker puts it in a queue of pending jobs and executes them on a first-come/first-served basis and then manages the assignment of map and reduce tasks to the tasktrackers. Slaves {tasktracker} execute tasks upon instruction from the Master {Jobtracker} and also handle data motion between the map and reduce phases.

Key benefits of Hadoop

1. Designed to large files
2. Designed to be parallelism
3. Flexible development platform
4. HDFS runs to existing file system

4. MapReduce

MapReduce is a programming model for processing and generating large dataset. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system. Feature of MapReduce Model is its simplicity. Because Only Map() and Reduce() to be written by users. Input data are stored as a collection of partitions in a distributed file system. Programs are injected into a distributed-processing framework.

4.1 MapReduce Programming Model

The user of the MapReduce library expresses the computation as two functions: Map and Reduce. The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce function.

The Reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges these values to form a possibly smaller set of values. Typically just zero or one output value is produced per Reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory. The following are the features of current environment:

(1) Machines are typically dual-processor x86 processors running Linux, with 2-4 GB of memory per machine.

(2) Commodity networking hardware is used . typically either 100 megabits/second or 1 gigabit/second at the machine level.
(3) A cluster consists of hundreds or thousands of machines, and therefore machine failures are common.
(4) Storage is provided by inexpensive IDE disks attached directly to individual machines.
(5) Users submit jobs to a scheduling system. Each job consists of a set of tasks, and is mapped by the scheduler to a set of available machines within a cluster.

4.2 Map Reduce Engine Execution overview

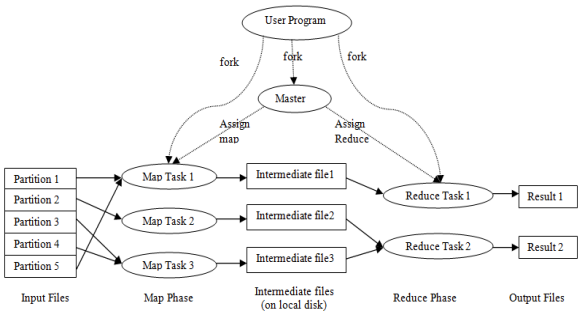

Fig. 4.1 MapReduce Execution Flow

MapReduce is a technology from Google. Map Reduce job is to broken the work into task that run in parallel. It it is maintained by Name Node , Master Node and heck point Node – it consist of reliable data. The work of Job Tracker (Master node) is to receive the user job and TaskTracker (slave mode) receives job from job tracker.

Map Reduce Engine contains map phase and it produce key value pair. It contains reduce phase to aggregate and gives output. Mappers and Reduces are user's code. It consist of H-base which is nothing but column oriented database. Mappers consume key value pair.

MapReduce Data flow

An Input Reader
A map function
A partition function
A compare function
A Reduce function
An Output Writer.

Input reader

The input reader divides the input into appropriate size 'splits' (in practice typically 16 MB to 128 MB) and the framework assigns one split to each Map function. The input reader reads data from stable storage (typically a distributed file system) and generates key/value pairs.

Map function

A series of key value pairs are taken by the Map function, processes it and generates zero or more output key value pairs. Input and output types of the map can be different.

Partition function

Each Map function output is allocated to a particular reducer by the application's partition function for sharing purposes. The partition function is given the key and the number of reducers so that it returns the index of the desired reduce.

A typical default is to hash the key and use the hash value to modulo the number of reducers. It is important to pick a partition function that gives an approximately uniform distribution of data per share for load-balancing purposes, otherwise the MapReduce operation can be held up waiting for slow reducers (reducers assigned more than their share of data) to finish. Between the map and reduce stages, the data is shuffled (parallel-sorted / exchanged between nodes) in order to move the data from the map node that produced it to the shard in which it will be reduced. The shuffle can sometimes take longer than the computation time depending on network bandwidth, CPU speeds, data produced and time taken by map and reduce computations.

Comparison function

The input for each Reduce is pulled from the machine where the Map ran and sorted using the application's comparison function.

Reduce function

The framework calls the application's Reduce function once for each unique key in the sorted order. The Reduce can iterate through the values that are associated with that key and produce zero or more outputs. In the word count example, the Reduce function takes the input values, sums them and generates a single output of the word and the final sum.

Output writer

The Output Writer writes the output of the Reduce to the stable storage, usually a distributed file system.

4.3 YARN or MapReduce 2.0 architecture

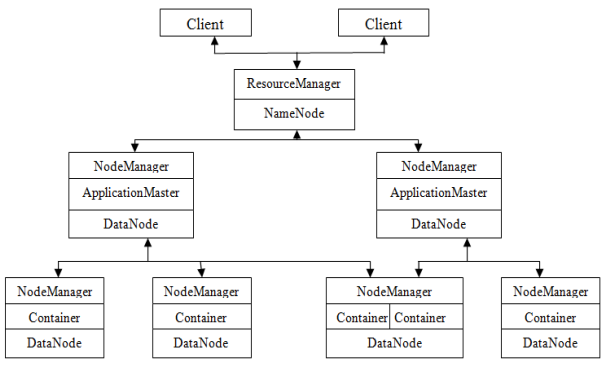

Fig. 4.2: YARN Architecture

The ResourceManager governs the entire cluster and manages the assignment of applications to underlying

computation of resources. An ApplicationMaster manages each instance of an application that runs within YARN. The ApplicationMaster is responsible for negotiating resources from the ResourceManager and through the NodeManager, monitoring the execution and resource consumption of containers.

The ApplicationMaster has taken some of the role of the prior TaskTracker, and the ResourceManager has taken the role of the JobTracker.

Hadoop Eco-system tools

NoSQL: DatabasesMongoDB, CouchDB, Cassandra, Redis, BigTable, Hbase, Hypertable, Voldemort, Riak, ZooKeeper.
MapReduce: Hadoop, Hive, Pig, Cascading, Cascalog, mrjob, Caffeine, S4, MapR, Acunu, Flume, Kafka, Azkaban, Oozie, Greenplum .
Storage: S3, Hadoop Distributed File System .
Servers: EC2, Google App Engine, Elastic, Beanstalk, Heroku
Processing: R, Yahoo! Pipes, Mechanical Turk, Solr/Lucene, ElasticSearch, Datameer, BigSheets, Tinkerpop .

Conclusion

This paper analyzes the concept of Big data and how it differs from traditional database. It also clearly specifies the Hadoop environment, its architecture and how it can be implemented using MapReduce along with various functions. So, it is sure that this paper helps the researches to understand the basic concepts of Big data, Hadoop and MapReduce to move further.

References

J. Dean and S. Ghemawat. (2004),MapReduce: Simplified Data Processing on Large Clusters, p.10.
Andrew Pavlo, et al. (2009) A Comparison of Approaches to Large-Scale Data Analysis, SIGMOD'09.
Jean-Pierre Dijcks (2013), Oracle: Big Data for the Enterprise.
Stonebraker, M., et al. (2010), MapReduce and parallel DBMS: friends or foes? ACM.
DeWitt, D (2008); Stonebraker, M., MapReduce: A major step backwards.
Dean, J. and Ghemawat, S. (2010), MapReduce: a flexible data processing tool, ACM
Fay Chang, Jeffrey Dean, Sanjay Ghemawat,Bigtable:A Distributed Storage System for Structured Data, OSDI'06.
Xindong Wu, Xingquan Zhu, Gong-Qing Wu, Wei Ding (2013), Data Mining with Big Data.
Ralf Lamme, Google's MapReduce Programming Model Revisited
M. Tim Jones, Micah Nelson (2013), Moving ahead with Hadoop YARN: An introduction to Yet Another Resource Negotiator.