International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 9 (2014), pp. 869-878
© International Research Publications House
http://www. irphouse.com

A Systematic Approach to Big Data
Exploration of the Hadoop Framework

Anand Loganathan, Ankur Sinha, Muthuramakrishnan V., and Srikanth Natarajan

*Dept of Computer Science & Engineering, SRM University
Chennai, India*

Abstract

With the growing developments and advancements in the fields of computing, it is necessary for institutions and organizations to handle large masses of data at faster speeds. Not only are the sizes of data increasing, so are the varied file types. Due to the inadequacy of traditional file management systems to handle this kind of large data, a need for a more appropriate system arose. This need led to the introduction and development of Big Data Technology. Big Data Technology includes different modules capable of moving beyond exabytes of data. In this paper, we provide a comparison between relational and non-relational database systems, their uses, implementations, advantages and disadvantages. Apart from this, we also provide an in-depth overview of the modules related to Hadoop, a Big Data management framework.
Keywords: Big Data, Apache Hadoop, NoSQL, Databases

1. Introduction
The exponential growth of data in today’s world has necessitated a paradigm shift in the way we manage and process data. Various fields such as banking, business informatics, meteorology, sports and medicine have felt the need for expanding the horizons of data management and mining. The traditional structure of relational database systems is not accoutered to handle this kind of data. Hence, the advancement into what is known as Big Data Technology today was not only unavoidable but also inexorable.

The need to look beyond traditional and relational databases led to the introduction of the non-relational databases such as the NoSQL (Not Only SQL). The NoSQL architecture was designed to address the problems of massive horizontal scalability and agility. In this paper, we draw comparisons between the relational and non-relational databases, their advantages, disadvantages and uses. But when analyzing the capabilities of non-relational databases, we discover that it is not a convincing solution to handle the accretion of Big Data. To undertake the problem of the same, Hadoop, an open source data computing framework has led the way, providing the solution with a number of modules to tackle all aspects of Big Data and its implementations. The modules we look at are HDFS, MapReduce, Pig, Hive, JAQL, HBase, Flume, Sqoop, Oozie, Zookeeper, YARN, Mahout, Ambari, Spark, Whirr, Hue, Lucene, Chukwa, Hama, Cassandra, and Impala. Each module serves its own purpose in the large Hadoop ecosystem, right from administration of large clusters of datasets to query management. By studying each module and attaining knowledge on it, we can effectively implement solutions to Big Data.

2. Comparison of various databases

Traditional database is any structured collection of data which stores information in an organized way such that the desired information can be quickly selected from the database. In traditional databases, the files included a lot of duplicated data causing redundancy problems. This was solved by using a relational database model where each piece of data had varied relationships with other pieces of data. There is no predefined hierarchical order for the data or for its representation. In a relational DBMS, the information is stored in set of tables where each one has its own unique identifier or primary key. The tables are then related to one another using foreign keys. Data storage in a RDBMS is better because undesirable data redundancy can be circumvented making the data management comparatively easier. Failure to normalize the data might pose as a challenge to users, and with increasing complexity and size of data, non-relational database was introduced.

NoSQL[1] is one such type of non-relational database architecture. NoSQL does not impose any limits to the variety and size of data thereby helping it attain horizontal scalability by following the highly optimized key-value store format. NoSQL databases also have a structure to store data, but these structures are less strict as relational schema, so it became a better choice for some applications. It is important to say that NoSQL databases are a complement in database field, and not a replacement for RDBMS. If the data is huge and complex, the traditional DBMS tools can fail to handle it.

Big Data is data which goes beyond terabytes and is often unstructured. Big Data addresses the problem of 4 V's: (1) Volume: Data at rest; from terabytes to zettabytes of existing data to process; (2) Velocity: Data in motion; streaming data and processing them between a time frame of milliseconds and seconds; (3) Variety: Data in many forms; the various files types of data and the magnitude of structured-ness; (4) Veracity: Data in doubt; inconsistency, incompleteness, latency, ambiguity in data. To handle big data, Apache software foundation introduced a tool called Apache Hadoop.

Apache Hadoop[2] is an open source software framework for storage and large scale processing of datasets on clusters of commodity hardware. It is a framework comprising of several modules and in this paper, we discuss twenty two such modules related to Hadoop. All of these modules are designed under an assumption that it is common to have a hardware problem, provide automatic fault tolerance and recovery, give better throughput, and is highly efficient.

3. Hadoop components

A. Hadoop Distributed File System (HDFS)

HDFS, which stands for Hadoop Distributed File System[3], is a distributed file system which has been designed to handle large data sets to run on low cost hardware and provide high fault tolerance. HDFS enables streaming access to file system data on cost of relaxing few POSIX requirements. There are many components associated with Hadoop and each component gives a non-trivial probability of detection of faults, and quick, automatic recovery. In case any hardware failures, it can be easily recovered because a HDFS instance may consist of hundreds or thousands of server machines, each storing part of the file system’s data. Applications running on HDFS need streaming access to their data sets. HDFS is designed for more of batch processing than user interactive process. The emphasis is on high throughput of data access rather than low latency of the same.

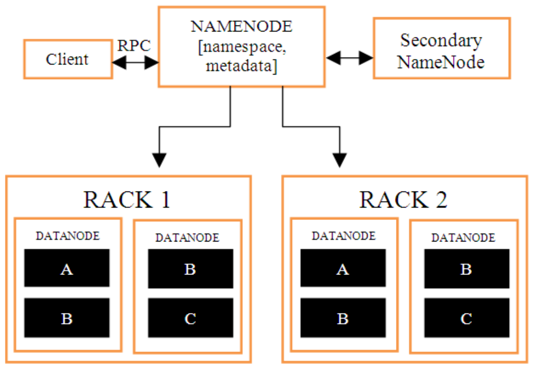

Fig. 1 Replication of Data in HDFS

POSIX semantics in a few key areas have been relaxed to gain an augmentation in data throughput rates. A typical file in HDFS can range between gigabytes and terabytes in size. Thus, the HDFS is tuned to support large files. It provides high aggregate data bandwidth and scales to hundreds of nodes on a single cluster and also supports tens of millions of files on a single instance. Applications running on HDFS follow a write-once, read-many access models for all files. Once a file is created, it need not be changed except for the appends. This assumption simplifies the data coherency issues and enables high throughput data access. A MapReduce application or a web crawler application fits perfectly with this model. A HFDS cluster primarily consists of a NameNode that manages file system metadata and a DataNode that stores the actual data. It also creates duplicates of data in case a particular data becomes inaccessible or deleted.

B. MapReduce

To process a large set of data, MapReduce model is used. Typically, MapReduce[4] model has two distinct tasks. The first is the Map task which takes data from a large pool, where individual elements are broken down into key-value pairs. The Reduce task takes the output from the executed Map task as input and combines those data tuples into a smaller set of key-value pairs. The Reduce job is always performed after the Map job. MapReduce achieves reliability by parceling out a number of operations on the set of data to each node in the network. Each node is expected to report back periodically with completed work and status updates. The MapReduce engine consists of a JobTracker and a TaskTracker. MapReduce Jobs are submitted to the JobTracker by the client. The JobTracker passes the job to the TaskTracker node which tries to keep the work close to the data. Since HDFS is a rack aware file system, the JobTracker knows which node contains the data, and which other machines are nearby. If the work cannot be hosted on the actual node where the data resides, priority is given to nodes on the same rack. This reduces network traffic on the main backbone network. If a TaskTracker fails or times out, that part of the job is rescheduled.

C. Yet Another Resource Navigator (YARN)
The initial release of Hadoop faced problems where cluster was tightly coupled with Hadoop and there were several cascading failures. This led to the development of a framework called YARN[5] (Yet Another Resource Navigator). Unlike the previous version, the addition of YARN has provided with better scalability, cluster utilization and, user agility. The incorporation of MapReduce as a YARN framework has provided full backward compatibility with existing MapReduce tasks and applications. It promotes effective utilization of resources while providing distributed environment for the execution of an application. The advent of YARN has opened the possibilities of building new applications to be built on top of Hadoop.

D. Pig
The Apache Pig, includes a Pig Latin programming language for expressing data flows, is a high-level dataflow language which is used to reduce the complexities of MapReduce by converting its operators into MapReduce code. It uses SQL-like operations to be performed on large distributed datasets[6]. Pig makes use of both, the Hadoop Distributed File System as well as the MapReduce. Pig can run on two types of environments: the local environment in a single JVM or the distributed environment on a Hadoop cluster. The flow of data can be simple as well as complex, that is, nested. The script of Pig Latin is just like a Directed Acyclic Graph, where edges are the data flow and the nodes are operators that process the data[7]. One of the highlights of Pig is that unlike SQL, it does not require to have a schema, hence being apt for processing unstructured or semi-structured data. Pig has variety of scalar data types and standard data processing options. Pig supports Map data; a map being a set of key-value pairs. Most Pig operators take a relation as an input and give a relation as the output (LOAD & STORE respectively). It allows normal arithmetic operations and relational operations too. Apart from keywords and operators, everything else in Pig is case sensitive[8].

E. Hive
With traditional RDBMS not being accoutred enough to handle, store or process large data in efficient way and make it scalable, Apache Hive came into play. Apache Hive is a data warehouse built on top of Hadoop. The query language is known is HiveQL since it is also a SQL dialect; it is a high-level declarative language. Hive has its own DDL and DML commands and queries. Unlike most SQL having schema-on-write feature, Hive has schema-on-read and supports multiple schemas, which defers the application of a schema until you try to read the data. Though the benefit here is that it loads faster, the drawback is that the queries are relatively slower[9]. Hive supports variety of storage formats: TEXTFILE for plaintext, SEQUENCEFILE for binary key-value pairs, RCFILE stores columns of a table in a record columnar format. Hive lacks a few things compared to RDBMS though, for example, it is best suited for batch jobs not real-time application processing. Hive lacks full SQL support and does not provide row-level inserts, updates or delete[10]. This is where HBase, another Hadoop module is worth investing.

F. HBase
HBase is a non-relational distributed database model. HBase comes as a handy alternative to Hive which lacks full SQL support as mentioned earlier. HBase not only provides row-level queries but is also used for real-time application processing unlike Hive. Though HBase is not an exact substitute for traditional RDBMS, it offers both, linear and modular scalability and is strictly maintains consistency of read and write which in return helps in automatic failover support. A distributed system can only guarantee two of the three CAP (Consistency, Availability, and Partition Tolerance) properties. HBase convincingly implements the consistency and partition tolerance feature. HBase is a non-ACID (Atomicity, Consistency, Isolation, and Durability) compliant[11]. All these features make HBase different and unique and also the go-to tool for horizontal scaling of large datasets.

G. JAQL
JAQL is a JSON based query language, which is high-level just like Pig Latin and MapReduce. To exploit massive parallelism, JAQL converts high-level queries into low-level queries. Like Pig, JAQL also does not enforce the obligation of having a schema. JAQL supports a number of in-built functions and core operators. Input and Output operations on JAQL are performed using I/O adapters, which is responsible for processing, storing and translating and returning the data as JSON format[12].

H. Flume
Flume is a tool built by Cloudera that acts around the Hadoop cluster. It is used for efficiently collecting, aggregating and managing large sets of log data and streaming data into the HDFS. It is also known as a pseudo-distributed model for its ability to run several processes on a single machine. It was designed to address four problems: reliability, scalability, manageability and extensibility. It lets users to stream data from multiple sources and high volume logs for real-time analysis, scale them horizontally and ensure data delivery[13]. All these features of Flume make it robust

and agile. Flume is one of the rare tools with wide variety of fault tolerant and reliability mechanisms.

I. Oozie

With so many Hadoop jobs running on different clusters, there was a need for a scheduler when Oozie came into the scene. The highlight of Oozie is that it combines multiple sequential jobs into one logical unit of work. There are two basic types of Oozie jobs: Oozie Workflow Jobs which is more like a Directed Acyclic Graph, which specifies a sequence of jobs to be executed, and the other is Oozie Coordinator Jobs which are recurrent Workflow Jobs that are triggered by the date and time availability. Oozie Bundle helps packaging both the jobs and maintaining a proper lifecycle. The outputs of a workflow in Oozie become the input of the next workflow and this process is known as data application pipeline[14].

J. Sqoop

Sqoop is a tool which provides a platform for exchange of data between Hadoop and any relational databases, data warehouses and NoSQL datastore. The transformation of the imported data is done using MapReduce or any other high-level language like Pig, Hive or JAQL. It allows easy integration with HBase, Hive and Oozie. It provides parallel operation to be performed and also supports fault tolerance[15].

K. Mahout

Mahout[16] is a scalable machine learning library built on top of Hadoop concentrating on collaborative filtering, clustering and classification. With data growing at faster rate every day, Mahout solved the need for remembering yesterday’s methods to process tomorrow’s data. It supports a gamut of machine learning algorithms to go about with its task. Apart from the machine learning algorithms, it also supports a number of clustering algorithms like k-means, dirichlet, mean-shift, and canopy.

L. Zookeeper

Zookeeper is a high performance coordination service for distributed applications where distributed processes coordinate with each other through a shared hierarchical name space of data registers. Zookeeper is associated with certain aspects that are required while designing and developing some coordination services. The configuration management service helps storing configuration data and sharing the data across all nodes in the distributed setup. The naming service allows one node to find a specific machine in a cluster of thousands of servers. The synchronization service provides the building blocks for Locks, Barriers and Queues. The locking service allows serialized access to a shared resource in the distributed system. The Leader Election service helps to recover the system from automatic failure[17].

M. Avro

Avro is a RPC (Remote Procedure Call) data serialization system focussed on dynamic access, platform independence and evolution of a schema. The RPC in Avro supports cross-language access and permits different versions of services to interoperate. It supports rich data structures with schema describe via JSON. The data in Avro is compressible and dividable and consists of arbitrary metadata. It can also include data with different schemas in the same file and detect them dynamically[18]. In near future, Hadoop's RPC mechanism is likely to be replaced completely by Avro.

N. Ambari

Ambari is a tool for provisioning, managing, and monitoring Hadoop clusters. The huge collection of operator tools and APIs hide the complexity of Hadoop thereby simplifying the operation of and on clusters. Irrespective of the size of the cluster, Ambari simplifies the deployment and maintenance of the host. It pre-configures alters for watching the Hadoop services and visualizes and displays the cluster operations in a simple web interface. The job diagnostic tools help to visualize job interdependencies and view timelines for historic job performance execution and troubleshooting for the same[19]. The latest version contains HBase multi-master, controls for host and simplified local repository setup.

O. Lucene

Lucene is an open source project for open source full text search. The core of Lucene[20] provides supports spell checking, hit highlighting and advanced tokenization capabilities. It consists of a high performance search server built using Lucene Core known as Solr (pronounced as Solar) providing XML/HTTP and JSON APIs. It supports faceting, highlighting and caching beside replication and sharding capabilities.

P. Spark

Spark is a fast and general engine for large scale data processing. It is an alternative to MapReduce for some cases. It is a low latency cluster computing system[21]. It can run programs upto 100 times faster than MapReduce in memory or 10 times faster on disk. It offers over 80 high-level operators that make it very easy to build parallel and scalable apps. It combines SQL, streaming and complex analytics.

Q. Cassandra

Cassandra[22] was developed to address the problem of traditional databases. It follows NoSQL structure and thereby produces linear scalability and provides fault-tolerance by automatically replicated to multi nodes on commodity hardware or any other cloud infrastructure services. It boasts of lower latency and inhibits regional outages. It is decentralized, elastic and has highly available asynchronous operations which are optimized with various features.

R. Hama

Hama[6] is a parallel matrix computation framework for massive scientific calculations, distributed computing, large-scale numerical analysis and data mining. It uses Bulk Synchronous Parallel (BSP) computing methodology. It uses Hadoop RPC for communication. The architecture consists of two main components: (1) BSPMaster which maintains GroomServer statuses, scheduling jobs, control faults, provide cluster control to users and maintaining job progress information, (2) GroomServer to perform BSP tasks and report status to master.

S. Chukwa
Chukwa[23] is a data collection system for monitoring large distributed systems. It is built on top of the HDFS and MapReduce framework and inherits Hadoop's scalability and robustness. It transfers data to collectors and saves data to HDFS. It contains data sins which stores raw unsorted data. A functionality called Demux is used to add structures to create Chukwa records which eventually go to the database for analysis. It includes a flexible toolkit for displaying, monitoring and analyzing results to make a better use of the collected data.

T. Whirr
Whirr[24] is a set of libraries for running cloud services. It provides high-level interaction between Hadoop and the cloud. It is based on JClouds. Clusters can be created as per the need using Whirr. The advantage of using Whirr is that it gives independence from Cloud vendor and makes it easier to move from them when required. Cluster can be used and expanded as per demand and can be reduced when not needed. It can compress data to reduce costs.

U. Impala
Impala is an open source query language for massive parallel processing developed by Cloudera that runs natively on Hadoop. The key benefits of using Impala is that it can perform interactive analytics in real-time, reduce data movement and duplicate storage thereby reducing costs and providing integration with leading Business Intelligence tools.

V. Hue
Hue stands for Hadoop User Experience[25]. It is an open source GUI for Hadoop, developed by Cloudera. Its goal is to let user free from worries about the underlying and backend complexity of Hadoop. It has a HDFS file browser, YARN & MapReduce Job Browser, HBase and Zookeeper browser, Sqoop and Spark editor, a query editor for Hive and Pig, app for Ozzie workflows, access to shell and app for Solr searches.

4. Conclusion
Despite the fact that NoSQL and Hadoop may be better suited for large amounts of data, it is not the recommended solution or the replacement for all problems. Only in the case of data sets exceeding exabytes demanding large scalability and complexity is Hadoop a suitable option. Apart from clarifying on the capabilities of each system, this paper provides an insight on the functionalities of the various modules in the Hadoop ecosystem.

With data growing every day, it is evident that Big Data and its implementations are the technological solution of the future. Soon, almost all industries and organizations around the world will adopt Big Data technology for data management.

5. References

[1] Shashank Tiwari, “Professional NoSQL”, Wrox Publications, 2011 Edition

[2] Tom White, “Hadoop: The Definitive Guide”, O’Reilly Media, 2012 Edition

[3] Hadoop Distributed File System Architecture Guide, Online: http://hadoop.apache.org/docs/stable1/hdfs_design.html

[4] Donald Miner, Adam Shook, “MapReduce Design Patterns”, O’Reilly Media, 2012 Edition

[5] Hadoop Yet Another Resource Navigator – Hortonworks, Online: http:// hortonworks.com/hadoop/yarn/

[6] Jason Venner, “Pro Hadoop, Apress, 2009 Edition

[7] Alan Gates, “Programming Pig”, O’Reilly Media, 2011 Edition

[8] Warren Pettit, “Introduction to Pig”, Big Data University, Online: http://bigdatauniversity.com/bdu-wp/bdu-course/introduction-to-pig/

[9] Edward Capriolo, Dean Wampler, Jason Rutherglen, “Programming Hive”, O’Reilly Media, 2012 Edition

[10] Aaron Ritchie, “Using Hive for Data Warehousing”, Big Data University, Online: http://bigdatauniversity.com/bdu-wp/bdu-course/using-hive-for-data-warehousing/

[11] Henry L. Quach, “Using HBase for Real-time Access to your Big Data”, Big Data University, Online: http://bigdatauniversity.com/bdu-wp/bdu-course/using-hbase-for-real-time-access-to-your-big-data/

[12] Glenn Mules, Warren Pettit, “Introduction to JAQL”, Big Data University, Online: http://bigdatauniversity.com/bdu-wp/bdu-course/414 /

[13] Apache Flume – Hortonworks, Online: http://hortonworks.com/hadoop/flume/

[14] Apache Oozie – Hortonworks, Online: http://hortonworks.com/hadoop/oozie/

[15] Kathleen Ting, Jarek Jarcec Cecho, “Apache Sqoop Cookbook”, O’Reilly Media, 2013 Edition

[16] Sean Owen, Robin Anil, Ted Dunning, Ellen Friedman, “Mahout in Action, Manning, 2011 Edition

[17] Aaron Ritchie, Henry Quach, “Developing Distributed Applications Using Zookeeper”, Big Data University, Online: http://bigdatauniversity.com/bdu-wp/bdu-course/developin-distributed-applications-using-zookeeper /

[18] Jim Scott, “Avro – More Than Just A Serialization Framework”, Chicago Hadoop Users Group, April 2012

[19] Apache Ambari – Hortonworks, Online: http://hortonworks.com/hadoop/ambari/

[20] Apache Lucene Core, Online: http://lucene.apache.org/core/

[21] Spark – Lighting Fast Cluster Computing, Online: http://spark.incubator. apache.org/

[22] Ebin Hewitt, “Cassandra: The Definitive Guide”, O’Reilly Media, 2010 Edition

[23] Chukwa Processes and Data Flow, Online: http://wiki.apache.org/hadoop /Chukwa_Processes_and_Data_Flow/
[24] Apache Whirr, "Apache Whirr Quick Start Guide", Online : http://archive. cloudera.com/cdh/3/whirr/quick-start-guide.html
[25] Apache Hue, Online: http://gethue.tumblr.com/