International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

Hadoop Ecosystem: An Introduction

Sneha Mehta[1], Viral Mehta[2]

[1]International Institute of Information Technology, Department of Information Technology, Pune, India

[2]MasterCard Technology Pvt. Ltd., Pune, India

Abstract: ***Hadoop a de facto industry standard has become kernel of the distributed operating system for Big data. Hadoop has gained its popularity due to its ability of storing, analyzing and accessing large amount of data, quickly and cost effectively through clusters of commodity hardware. But, No one uses kernel alone. "Hadoop" is taken to be a combination of HDFS and MapReduce. To complement the Hadoop modules there are also a variety of other projects that provide specialized services and are broadly used to make Hadoop laymen accessible and more usable, collectively known as Hadoop Ecosystem. All the components of the Hadoop ecosystem, as explicit entities are evident to address particular needs. Recent Hadoop ecosystem consists of different level layers, each layer performing different kind of tasks like storing your data, processing stored data, resource allocating and supporting different programming languages to develop various applications in Hadoop ecosystem.***

Keywords: HDFS, HBase, MapReduce, YARN, Hive, Pig, Mahout, Avro, Sqoop, Oozie, Chukwa, Flume, Zookeeper

1. Introduction

Hadoop Ecosystem is a framework of various types of complex and evolving tools and components which have proficient advantage in solving problems. Some of the elements may be very dissimilar from each other in terms of their architecture; however, what keeps them all together under a single roof is that they all derive their functionalities from the scalability and power of Hadoop. Hadoop Ecosystem is alienated in four different layers: data storage, data processing, data access, data management. Figure 1 depicts how the diverse elements of hadoop [1] involve at various layers of processing data.

All the components of the Hadoop ecosystem, as explicit entities are evident. The holistic view of Hadoop architecture gives prominence to Hadoop common, Hadoop YARN, Hadoop Distributed File Systems (HDFS) and Hadoop MapReduce of the Hadoop Ecosystem. Hadoop common provides all Java libraries, utilities, OS level abstraction, necessary Java files and script to run Hadoop, while Hadoop YARN is a framework for job scheduling and cluster resource management. HDFS in Hadoop architecture provides high throughput access to application data and Hadoop MapReduce provides YARN based parallel processing of large data sets.

The Hadoop ecosystem [15] [18] [19] includes other tools to address particular needs. Hive is a SQL dialect and Pig is a dataflow language for that hide the tedium of creating MapReduce jobs behind higher-level abstractions more appropriate for user goals. HBase is a column-oriented database management system that runs on top of HDFS. Avro provides data serialization and data exchange services for Hadoop. Sqoop is a combination of SQL and hadoop. Zookeeper is used for federating services and Oozie is a scheduling system. In the absence of an ecosystem [11] [12], developers have to implement separate sets of technologies to create Big Data [20] solutions.

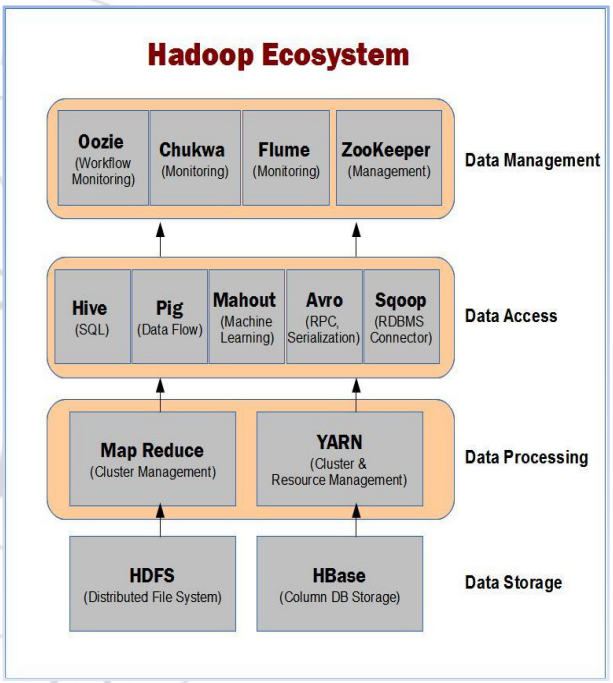

Figure 1: Hadoop Ecosystem

2. Data Storage Layer

Data Storage is the layer where the data is stored in a distributed file system; consist of **HDFS** and **HBase** ColumnDB Storage. HBase is scalable, distributed database that supports structured data storage for large tables.

2.1 HDFS

HDFS, the storage layer of Hadoop, is a distributed, scalable, Java-based file system adept at storing large volumes of data with high-throughput access to application data on the community machines, providing very high aggregate bandwidth across the cluster. When data is pushed to HDFS, it automatically splits up into multiple blocks and stores/replicates the data thus ensuring high availability and fault tolerance.

HDFS comprises of 2 important components (Fig. 2) – NameNode and DataNode. HDFS operates on a Master-Slave architecture model where the NameNode acts as the master node for keeping a track of the storage cluster and the DataNode acts as a slave node summing up to the various systems within a Hadoop cluster.

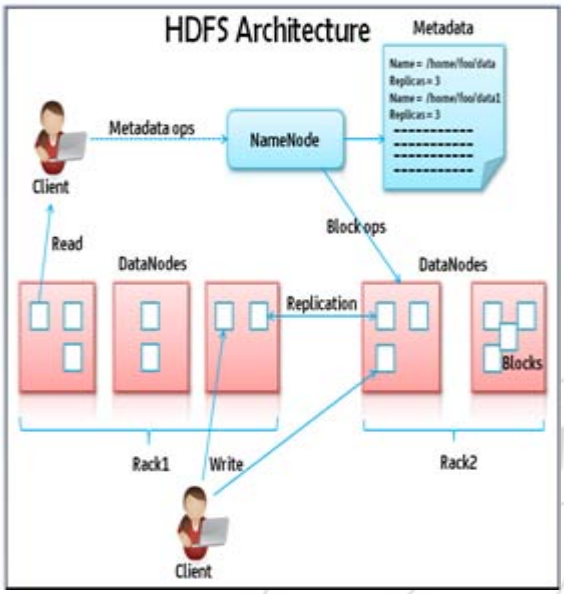

Figure 2: HDFS Architecture

HDFS works on the write once-read many times approach and this makes it capable of handling such huge volumes of data with least possibilities of errors caused by replication of data. This replication of data across the cluster provides fault tolerance and resilience against server failure. Data Replication, Data Resilience, and Data Integrity are the three key features of HDFS.

- **NameNode:** It acts as the master of the system. It maintains the name system i.e., directories and files and manages the blocks which are present on the DataNodes.
- **DataNodes:** They are the slaves which are deployed on each machine and provide the actual storage and are responsible for serving read and write requests for the clients. Additionally, DataNodes communicate with each other to co-operate and co-ordinate in the file system operations.

2.2 HBase

HBase is a scalable, distributed database that supports structured data storage for large tables. Apache HBase provides Bigtable - like capabilities on top of Hadoop and HDFS. HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. HBase facilitates reading/writing of Big Data randomly and efficiently in real time. It stores data into tables with rows and columns as in RDBMs. HBase tables have one key feature called "versioning" which helps in keeping a track of the changes made in a cell and allows the retrieval of the previous version of the cell contents, if required.

HBase [2] also endow with a variety of functional data processing features, such as consistency, sharding, high availability, client API and support for IT operations.

3. Data Processing Layer

Scheduling, resource management and cluster management is premeditated here. YARN job scheduling and cluster resource management with Map Reduce are located in this layer.

3.1 MapReduce

MapReduce [8] is a software framework for distributed processing of large data sets that serves as the compute layer of Hadoop which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A MapReduce *job* usually splits the input data-set into independent chunks which are processed by the *map tasks* in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the *reduce tasks*. The "Reduce" function aggregates the results of the "Map" function to determine the "answer" to the query. On average both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executing any failed tasks.

The compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuration allows the framework to effectively schedule task on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

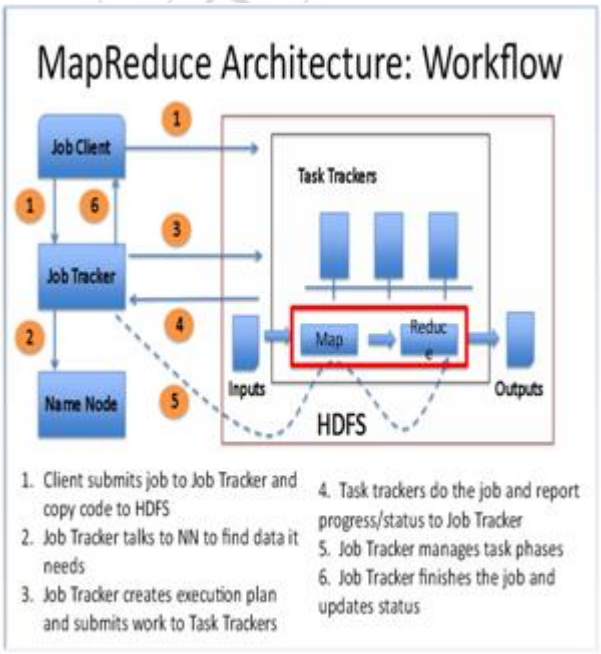

Figure 3: MapReduce Architecture: Workflow

The MapReduce framework (Fig. 3), [7] consists of a single master Job Tracker and one slave Task Tracker per cluster-node. The master is responsible for scheduling the jobs'

component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

Although the Hadoop framework is implemented in Java, MapReduce [25] applications need not be written in Java. Hadoop Streaming is a utility which allows users to create and run jobs with any executables (e.g. shell utilities) as the mapper and/or the reducer. Hadoop Pipes is a SWIG-compatible C++ API sto implement MapReduce applications.

3.2 YARN

YARN (Yet Another Resource Negotiator) forms an integral part of Hadoop 2.0.YARN is great enabler for dynamic resource utilization on Hadoop framework as users can run various Hadoop applications without having to bother about increasing workloads. The inclusion of YARN in hadoop 2 also means scalability provided to the data processing applications.

YARN [19] [17] is a core hadoop service that supports two major services:

--Global resource management (ResourceManager)
--Per-application management (ApplicationMaster)

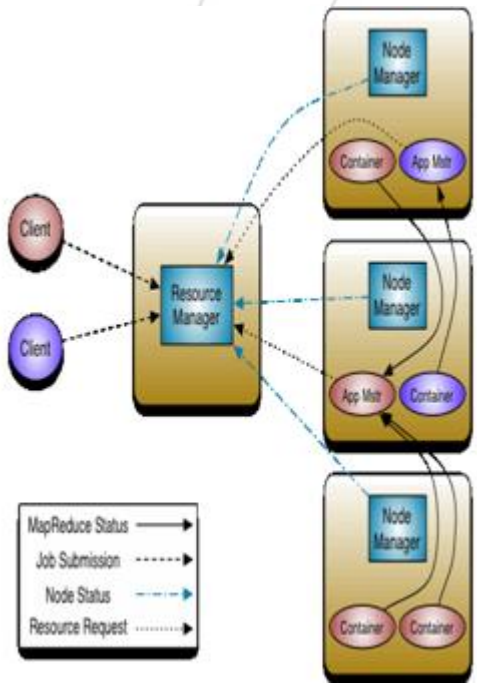

Figure 4: YARN Architecture

Resource Manager: Resource Manager, in YARN architecture (Figure 4), is supreme authority that controls all the decisions related to resource management and allocation. It has a Scheduler Application Programming Interface (API) that negotiates and schedules resources. However, The Scheduler API doesn't monitor or track the status of applications.

The main purpose of introducing Resource Manager in YARN is to optimize the utilization of resources all the time by managing all the restrictions, which involve capacity guarantees, fairness in allocation of resources etc. Thus, YARN Resource Manager is responsible for almost all the tasks. Resource Manager performs all its tasks in integration with NodeManager and Application Manager.

Application Manager: Every instance of an application running within YARN is managed by an Application Manager, which is responsible for the negotiation of resources with the Resource Manager. Application Manager also keeps track of availability and consumption of container resources, and provides fault tolerance for resources. Accordingly, it is responsible for negotiating for appropriate resource containers from the Scheduler, monitoring of their status, and checking the progress.

Node Manager: NodeManager is the per-machine slave, which is responsible for launching the applications' containers, monitoring their resource usage, and reporting the status of the resource usage to the Resource Manager. NodeManager manages each node within YARN cluster. It provides per-node services within the cluster. These services range from managing a container to monitoring resources and tracking the health of its node.

YARN benefits include efficient resource utilization, highly scalability, beyond Java, novel programming models and services and agility.

4. Data Access Layer

The layer, where the request from Management layer is sent to Data Processing Layer. Some projects have been setup for this layer, Some of them are: **Hive,** A data warehouse infrastructure that provides data summarization and ad hoc querying; **Pig**, A high-level data-flow language and execution framework for parallel computation; **Mahout,** A Scalable machine learning and data mining library; **Avro**, data serialization system.

4.1 Hive

Hive [5] is a Hadoop-based data warehousing-like framework originally developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy. It allows users to write queries in a SQL-like language called HiveQL which are then converted to MapReduce. This allows SQL programmers with no MapReduce experience to use the warehouse and makes it easier to integrate with business intelligence and visualization tools.

Features of Hive:

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It provides SQL type language for querying called HiveQL or HQL.
- It is familiar, fast, scalable, and extensible.

4.2 Apache Pig

Apache Pig [6] is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. The salient property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

At the present time, Pig's infrastructure layer consists of a compiler that produces sequences of Map-Reduce programs, for which large-scale parallel implementations already exist (e.g., the Hadoop subproject). Pig's language layer currently consists of a textual language called Pig Latin, which has the following key properties:

- **Ease of programming:** It is trivial to achieve parallel execution of simple, "embarrassingly parallel" data analysis tasks. Complex tasks comprised of multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
- **Optimization Opportunities:** The way in which tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.
- **Extensibility:** Users can create their own functions to do special-purpose processing.

4.3 Apache Mahout

Apache Mahout [9] [14] is a project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focused primarily in the areas of collaborative filtering, clustering and classification and implements them using the Map Reduce model.

The primitive features of Apache Mahout include:

- The algorithms of Mahout are written on top of Hadoop, so it works well in distributed environment.
- Mahout uses the Apache Hadoop library to scale effectively in the cloud.
- Mahout offers the coder a ready-to-use framework for doing data mining tasks on large volumes of data.
- Mahout lets applications to analyze large sets of data effectively and in quick time.
- Includes several MapReduce enabled clustering implementations such as k-means, fuzzy k-means, Canopy etc.
- Supports Distributed Naive Bayes and Complementary Naive Bayes classification implementations.
- Comes with distributed fitness function capabilities for evolutionary programming.
- Includes matrix and vector libraries.

4.4 Avro

Avro [10] is a data serialization system that allows for encoding the schema of Hadoop files. It is adept at parsing data and performing removed procedure calls.

It was developed by Doug Cutting, the father of Hadoop. Since Hadoop writable classes lack language portability, Avro has become quite helpful, as it deals with data formats that can be processed by multiple languages. Avro is a preferred tool to serialize data in Hadoop.

Avro has a schema-based system. A language-independent schema is associated with its read and writes operations. Avro serializes the data which has a built-in schema. Avro serializes the data into a compact binary format, which can be deserialized by any application.

Avro uses JSON format to declare the data structures. Presently, it supports languages such as Java, C, C++, C#, Python, and Ruby. Below are some of the prominent features of Avro:

- Avro is a language-neutral data serialization system. It can be processed by many languages (currently C, C++, C#, Java, Python, and Ruby).
- Avro creates binary structured format that is both compressible and splittable. Hence it can be efficiently used as the input
- to Hadoop MapReduce jobs.
- Avro provides rich data structures. For example, you can create a record that contains an array, an enumerated type, and a sub record. These data types can be created in any language can be processed in Hadoop, and the results can be fed to a third language.
- Avro schemas defined in JSON facilitate implementation in the languages that already have JSON libraries.
- Avro creates a self-describing file named Avro Data File, in which it stores data along with its schema in the metadata section.

4.5 Apache Sqoop

"SQL to Hadoop and Hadoop to SQL"

Sqoop [11] is a connectivity tool for moving data from non-Hadoop data stores – such as relational databases and data warehouses into Hadoop. It allows users to specify the target location inside of Hadoop and instruct Sqoop to move data from Oracle, Teradata or other relational databases to the target.

Sqoop is a tool designed to transfer data between Hadoop and relational database servers. It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases. It is provided by the Apache Software Foundation.

–Sqoop Import

The import tool imports individual tables from RDBMS to HDFS. Each row in a table is treated as a record in HDFS. All records are stored as text data in text files or as binary data in Avro and Sequence files.

–Sqoop Export

The export tool exports a set of files from HDFS back to an RDBMS. The files given as input to Sqoop contain records,

which are called as rows in table. Those are read and parsed into a set of records and delimited with user-specified delimiter.

5. Data Management Layer

A layer that meets, user. User access the system through this layer which has the components like: Chukwa, A data collection system for managing large distributed system and Zookeeper, high-performance coordination service for distributed applications.

5.1 Oozie

Oozie is a workflow processing system that lets users define a series of jobs written in multiple languages – such as Map Reduce, Pig and Hive -- then intelligently link them to one another. Oozie allows users to specify, for example, that a particular query is only to be initiated after specified previous jobs on which it relies for data are completed. Oozie is a scalable, reliable and extensible system.

Oozie workflow is a collection of actions (i.e. Hadoop Map/Reduce jobs, Pig jobs) arranged in a control dependency DAG (Direct Acyclic Graph), specifying a sequence of actions execution. This graph is specified in hPDL (a XML Process Definition Language).

Benefits of Oozie are as follows:

- **Complex workflow action dependencies** – The Oozie workflow contains actions and dependencies among these actions. At runtime, Oozie manages dependencies and executes actions when dependencies identified in DAG are satisfied.
- **Frequency Execution** – Oozie workflow specification supports both data and time triggers.
- **Native Hadoop stack integration** – Oozie supports all types of hadoop jobs and is integrated with Hadoop stack.
- **Reduces Time-To-Market (TTM)** – The DAG specification enables users to specify the workflow, which saves time to build and maintain custom solutions for dependency and workflow management.
- **Mechanism to manage a variety of complex data Analysis –** Oozie is integrated with the yahoo! Distribution of Hadoop with security and is a primary mechanism to manage a variety of complex data analysis workloads across Yahoo!

5.2 Apache Chukwa

Chukwa [24] aims to provide a flexible and powerful platform for distributed data collection and rapid data processing. It is an open source data collection system for monitoring large distributed system and is built on top of the Hadoop Distributed File System (HDFS) and Map/Reduce framework that inherits Hadoop's scalability and robustness.

Chukwa also includes a flexible and powerful toolkit for displaying, monitoring and analyzing results to make the best use of the collected data. In order to maintain this flexibility, Chukwa is structured as a pipeline of collection and processing stages, with clean and narrow interfaces between stages.

Chukwa has four primary components:

- Agents that run on each machine and emit data and Collectors that receive data from the agent and write it to stable storage.
- MapReduce jobs for parsing and archiving the data.
- HICC, the Hadoop Infrastructure Care Center; a web-portal style interface for displaying data.

5.3 Apache Flume

Flume [23] is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

Features of Flume

- Flume ingests log data from multiple web servers into a centralized store (HDFS, HBase) efficiently.
- Using Flume, we can get the data from multiple servers immediately into Hadoop.
- Along with the log files, Flume is also used to import huge volumes of event data produced by social networking sites.
- Flume supports a large set of sources and destinations types and can be scaled horizontally.
- Flume supports multi-hop flows, fan-in and fan out flows, contextual routing, etc.

5.4 Apache Zookeeper

Apache Zookeeper [22] is a coordination service for distributed application that enables synchronization across a cluster. Zookeeper in Hadoop can be viewed as centralized repository where distributed applications can put data and get data out of it. It is used to keep the distributed system functioning together as a single unit, using its synchronization, serialization and coordination goals. For simplicity's sake Zookeeper can be thought of as a file system where we have znodes that store data instead of files or directories storing data. Zookeeper is a Hadoop Admin tool used for managing the jobs in the cluster.

Features of Zookeeper includes managing configuration across nodes, implementing reliable messaging, implementing redundant services and to synchronize process execution.

Benefits of Zookeeper

- Simple **distributed coordination process.**
- **Synchronization, Reliability and Ordered Messages** − Mutual exclusion and co-operation between server processes. This process helps in Apache HBase for configuration management.
- **Serialization** − Encode the data according to specific rules. Ensure your application runs consistently.

- **Atomicity** − Data transfer either succeed or fail completely, but no transaction is partial.

6. Conclusion

With a rapid pace in evolution of Big Data, its processing frameworks also seem to be evolving in a full swing mode. The huge data giants on the web has adopted Apache Hadoop had to depend on the partnership of Hadoop HDFS with the resource management environment and MapReduce programming. Hadoop ecosystem has introduced a new processing model that lends itself to common big data use cases including interactive SQL over big data, machine learning at scale, and the ability to analyze big data scale graphs. Apache Hadoop is not actually single product but instead a collection of several components. When all these components are merged, it makes the Hadoop very user friendly. The Hadoop ecosystem and its commercial distributions continue to evolve, with new or improved technologies and tools emerging all the time.

7. Acknowledgment

We express our gratitude to our parents, family members, Prof. (Dr). Vijay Patil (principal) and Prof. Vilas Mankar (HOD IT Dept.) I^{2}IT, Pune, for his constant encouragement and motivation throughout this work.

References

[1] Hadoop - Apache Software Foundation project home page. http://hadoop.apache.org.
[2] HBase - Apache Software Foundation project home page http://hbase.apache.org.
[3] Kiran kumara Reddi & Dnvsl Indira "Different Technique to Transfer Big Data: survey" IEEE Transactions on 52(8) (Aug.2013) 2348 {2355}.
[4] Konstantin Shvachko, et al., "The Hadoop Distributed File System, "Mass Storage Systems and Technologies (MSST), IEEE 26th Symposium on IEEE, 2010,http://storageconference.org/2010/Papers/MSST/Shvachko.pdf.
[5] Hive - Apache Software Foundation project home page http://hive.apache.org.
[6] Pig - Apache Software Foundation project home page. http://pig.apache.org.
[7] Jens Dittrich, Jorge-Arnulfo Quiane-Ruiz "Efficient Big Data Processing in Hadoop MapReduce", 2014.
[8] Jeffrey Dean, Sanjay Ghemawat, "MapReduce: simplified data processing on large clusters", Communications of the ACM, v.51 n.1, January 2008 [doi>10.1145/1327452.1327492].
[9] Mahout - Apache Software Foundation project home page http://mahout.apache.org
[10] Avro - Apache Software Foundation project home page http://avro.apache.org
[11] J. Yates Monteith, John D. McGregor, and John E. Ingram, "Hadoop and its evolving ecosystem", IWSECO@ ICSOB, Citeseer, 2013.
[12] Monteith, J.Y., McGregor, J.D.: "A three viewpoint model for software ecosystems", In: Proceedings of Software Engineering and Applications', 2012.
[13] Ivanilton Polato, Reginaldo Ré, Alfredo Goldman, Fabio Kon, " A comprehensive view of hadoop research – A systematic literature review", Journal of network and computer applications, volume 46, november 2014.
[14] Arantxa Duque Barrachina, Aisling O'Driscoll, "A big data methodology for categorising technical support requests using Hadoop and Mahout", Journal of Big Data, February 2014 [DOI: 10.1186/2196-1115-1-1].
[15] Mr.NileshVishwasrao Patil, Mr.Tanvir Patel**, "**Apache Hadoop: Resourceful Big Data Management**",** IJRSET, Volume 3, Special Issue 4, April 2014.
[16] Shilpa Manjit Kaur," BIG Data and Methodology- A review" ,International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 10, October 2013.
[17] Varsha B.Bobade, "Survey Paper on Big Data and Hadoop", IRJET, Volume: 03 Issue: 01, Jan-2016.
[18] A Katal, M. Wazid, R.H. Goudar, "Big Data: Issues, Challenges, tools and Good practices," Aug. 2013.
[19] Deepika P, Anantha Raman G R," A Study of Hadoop-Related Tools and Techniques", IJARCSSE, Volume 5, Issue 9, September 2015.
[20] Anand Loganathan, Ankur Sinha, Muthuramakrishnan V., and Srikanth Natarajan, "A Systematic Approach to Big Data Exploration of the Hadoop Framework", International Journal of Information & Computation Technology, Volume 4, 2014.
[21] Andrew Pavlo, "A comparison of approaches to large scale data Analysis", SIGMOD, 2009.
[22] Zookeeper - Apache Software Foundation project home page https://zookeeper.apache.org.
[23] Flume - Apache Software Foundation project home page https://flume.apache.org.
[24] Chukwa - Apache Software Foundation project home page https://chukwa.apache.org.
[25] J. Dean and S. Ghemawat, "MapReduce: A Flexible Data Processing Tool". CACM, 53(1):72–77, 2010.