International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Challenges for HDFS to Read and Write Using Different Technologies

Sunil Kumar S [1], Sanjeev G Kanabargi [2]

[1, 2] Manglore Institute of Technology and Engineering, Moodbidari, Karnataka

Abstract: ***Advancement in information and technology gives us ability to store a huge amount of data. In other terms we have huge bandwidth to store and analyze huge data which can be also termed as Big-Data. There are many tools and framework that can be used to manage and analyze the Big-data and Hadoop is one the most used frame work to analyze and manage Big-data. Hadoop uses HDFS to store and manage the Big-data and we can use different tools to read and write into HDFS. This paper compares different technologies that can be used to read and write into HDFS.***

Keywords: HDFS, Big-Data, YARN, **.**

1. Introduction

Google had some challenges in analyze the Big-data from the database, It had to analyze Big-Data, do parallel computation in hundreds or thousands of machines with large complex codes and generate results in reasonable amount of time. It also had issues of how to do parallel computing with data distributed over all machines and handles failures of machine and data residing in machine on failures.

To deal with these issues they designed a new abstraction that allows to express complex codes for parallelization, fault-tolerance, data distribution and load balancing in a simple way by hiding them into libraries. They come up with the map and reduce primitives which made easy to handle the issues mentioned above.

Hadoop is an open source frame work which is developed by a group of engineers from yahoo. They referred white papers published by Google Inc. on map-reduce and using same framework they extended it to make hadoop framework. This framework is very much useful while analyzing unstructured Big-Data provide features like reliability and data motion. Hadoop uses multiple machines to process and store Big-Data, to perform these operations, it uses map-reduce programing paradigm, where the applications are divided into small sub applications and they are sent to different connected machines for parallel execution of sub applications. In addition, It provides Hadoop Distributed File System[1] (HDFS) , where data is stored. Both map-reduce and HDFS framework is designed in manner which can handles node failures automatically.

Map-Reduce are two functions namely mapper function and reducer function, Mapper function get input data and it partially processes it by mapping, sorting, shuffling and making it ready for reducer function, this partially processed intermediate data is not stored into system and remove from system after the data is given to reducer job. In this paper we exploring different ways or technologies which can be used by mapper and reducer program to read and write to HDFS.

2. Literature Survey

2.1. Big Data

Big Data encompasses everything from click stream data from the web to genomic and proteomic data from biological research and medicines. Big Data is a heterogeneous mix of data both structured (traditional datasets –in rows and columns like DBMS tables, CSV's and XLS's) and unstructured data like e-mail attachments, manuals, images, PDF documents, medical records such as x-rays, ECG and MRI images, forms, rich media like graphics, video and audio, contacts, forms and documents. Businesses are primarily concerned with managing unstructured data, because over 80 percent of enterprise data is unstructured [26] and require significant storage space and effort to manage.“Big data” refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyse [3].

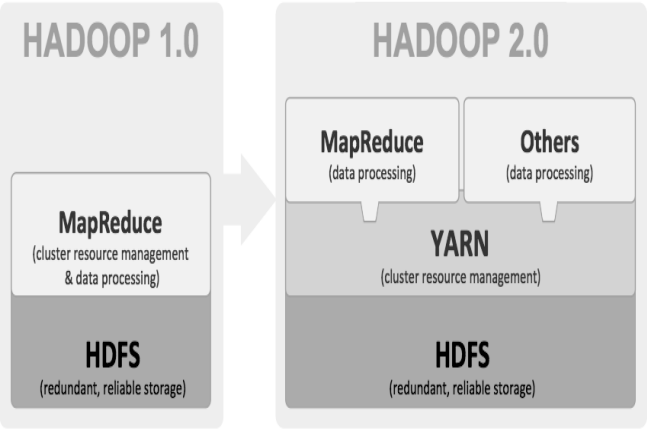
Big Data encompasses everything from click stream data from the web to genomic and proteomic data from biological research and medicines. Big Data is a heterogeneous mix of data both structured (traditional datasets –in rows and columns like DBMS tables, CSV's and XLS's) and unstructured data like e-mail attachments, manuals, images, PDF documents, medical records such as x-rays, ECG and MRI images, forms, rich media like graphics, video and audio, contacts, forms and documents. Businesses are primarily concerned with managing unstructured data, because over 80 percent of enterprise data is unstructured [26] and require significant storage space and effort to manage.“Big data” refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyse [3].

Discovery analytics against big data can be enabled by different types of analytic tools, including those based on SQL queries, data mining, statistical analysis, fact clustering, data visualization, natural language processing, text analytics, artificial intelligence etc [4-6]. A unique challenge for researchers system and academicians is that the large datasets needs special processing systems [5]. Map Reduce over

HDFS gives Data Scientists [1-2] the techniques through which analysis of Big Data can be done. HDFS is a distributed file system architecture which encompasses the original Google File System [13].Map Reduce jobs use efficient data processingtechniques which can be applied in each of the phases of MapReduce; namely Mapping, Combining, Shuffling,Indexing, Grouping and Reducing [7].

Discovery analytics against big data can be enabled by different types of analytic tools, including those based on SQL queries, data mining, statistical analysis, fact clustering, data visualization, natural language processing, text analytics, artificial intelligence etc [4-6]. A unique challenge for researchers system and academicians is that the large datasets needs special processing systems [5]. Map Reduce over HDFS gives Data Scientists [1-2] the techniques through which analysis of Big Data can be done. HDFS is a distributed file system architecture which encompasses the original Google File System[13].Map Reduce jobs use efficient data processing techniques which can be applied in each of the phases of MapReduce; namely Mapping, Combining, Shuffling,Indexing, Grouping and Reducing[7].

Driven by very similar requirements, software developers at Yahoo!, Facebook, and other large Web companies followed suit. Taking Google‟s GFS and Map Reduce papers as rough technical specifications, open-source equivalents were developed, and the Apache Hadoop Map Reduce platform and its underlying file system (HDFS, the Hadoop Distributed File System) were born [1] [12]. The Hadoop system has quickly gained traction, and it is now widely used for use cases including Web indexing, clickstream and log analysis, and certain large-scale information extraction and machine learning tasks. Soon tired of the low-level nature of the Map Reduce programming model, the Hadoop community developed a set of higher-level declarative languages for writing queries and data analysis pipelines that are compiled into Map Reduce jobs and then executed on the Hadoop Map Reduce platform. Popular languages include Pig from Yahoo! [18], Jaql from IBM [28], and Hive from Facebook [18]. Pig is relational-algebra-like in nature, and is reportedly used for over 60% of Yahoo!‟s



MapReduce use cases; Hive is SQL-inspired and reported to be used for over 90% of the Facebook Map Reduce use cases. Microsoft‟s technologies include a parallel runtime system called Dryad and two higher-level programming models, Dryad LINQ and the SQLlike SCOPE language [27], which utilizes Dryad under the covers. Interestingly, Microsoft has also recently announced that its future “Big Data” strategy includes support for Hadoop[24].

2.2. Apache's Hadoop 1.x and Hadoop 2.x with YARN

Hadoop[1] is an open source project of apache which implements Google's map-reduce[2] programing model. Hadoop have 3 ways to install on hardware, they are: stand alone, single node and Multi node cluster[3]. Hadoop architecture is evolved and developed by apache[3] group, from hadoop 1.x to hadoop 2.x[3] had been into many changes like introduction of map-reduce v2 or YARN[6].

Hadoop runs on JVM[4] (Java Virtual Machine). With respect to Single node and Multi node hadoop1.x JVM creates different instances of JVM for different nodes and trackers. The different nodes are Name Node[5], Data Node, Secondary Name Node and trackers are Job Tracker, Task Tracker, these works together and provide us Hadoop frame work. The Name Node will be holding the meta-data for whole system which can help us to locate the data in the system. The Data Node primary task is to store data, analyze it locally and generate results. Secondary Name node will be capturing the snapshot of the Name Node so it can take place of Name node at the time of failure. The Job Tracker is the service with in Hadoop that farms out Map-Reduce tasks to specific nodes in the cluster, the data nodes have the data and Data Node JVM will be running on them properly, the data nodes which aren't working properly are deemed to have failed and their work is scheduled on a different Task Tracker. The Task Tracker report back to the Job Tracker about the status of its processing, if everything worked well then, it will reply with result of else it will reply a failure message to Job Tracker, then job tracker will decide what to do, it may assign same task to another task tracker with another data node. When whole work is completed the Job Tracker updates it status to Name Node. Client applications can use the Job Tracker for the work information which includes time to process, ID etc.

2.2.1. Hadoop 1

In Hadoop 1, a single Namenode managed the entire namespace for a Hadoop cluster. With HDFS federation, multiple Namenode servers manage namespaces and this allows for horizontal scaling, performance improvements, and multiple namespaces. The implementation of HDFS federation allows existing Namenode configurations to run without changes. For Hadoop administrators, moving to HDFS federation requires formatting Namenodes, updating to use the latest Hadoop cluster software, and adding additional Namenodes to the cluster.

2.2.2. Hadoop 2: YARN

HDFS federation brings important measures of scalability and reliability to Hadoop. YARN, the other major advance in Hadoop 2, brings significant performance improvements for some applications, supports additional processing models, and implements a more flexible execution engine.

YARN is a resource manager that was created by separating the processing engine and resource management capabilities of MapReduce as it was implemented in Hadoop 1. YARN is often called the operating system of Hadoop because it is responsible for managing and monitoring workloads, maintaining a multi-tenant environment, implementing security controls, and managing high availability features of Hadoop.

YARN supports multiple processing models in addition to MapReduce. One of the most significant benefits of this is that we are no longer limited to working the often I/O intensive, high latency MapReduce framework. This advance means Hadoop users should be familiar with the pros and cons of the new processing models and understand when to apply them to particular use cases.

3. Avro

Apache Avro™ is a data serialization system.

Avro provides:

- Rich data structures.
- A compact, fast, binary data format.
- A container file, to store persistent data.
- Remote procedure call (RPC).
- Simple integration with dynamic languages

3.1. Schemas

Avro relies on *schemas*. When Avro data is read, the schema used when writing it is always present. This permits each datum to be written with no per-value overheads, making serialization both fast and small. This also facilitates use with dynamic, scripting languages, since data, together with its schema, is fully self-describing.

When Avro data is stored in a file, its schema is stored with it, so that files may be processed later by any program. If the program reading the data expects a different schema this can be easily resolved, since both schemas are present. When Avro is used in RPC, the client and server exchange schemas in the connection handshake. (This can be optimized so that, for most calls, no schemas are actually transmitted.) Since both client and server both have the other's full schema, correspondence between same named fields, missing fields, extra fields, etc. can all be easily resolved. Avro schemas are defined with JSON . This facilitates implementation in languages that already have JSON libraries.

3.2. Comparison with other systems

Avro provides functionality similar to systems such as Thrift, Protocol Buffers, etc. Avro differs from these systems in the following fundamental aspects.

Dynamic typing: Avro does not require that code be generated. Data is always accompanied by a schema that permits full processing of that data without code generation, static datatypes, etc. This facilitates construction of generic data-processing systems and languages.

Untagged data: Since the schema is present when data is read, considerably less type information need be encoded with data, resulting in smaller serialization size.

No manually-assigned field IDs: When a schema changes, both the old and new schema are always present when processing data, so differences may be resolved symbolically, using field names.

3.3. Sequiential file

Sequence Files are flat files consisting of binary key/value pairs. SequenceFile provides SequenceFile.Writer, SequenceFile.Reader and SequenceFile.Sorter classes for writing, reading and sorting respectively. There are three SequenceFile Writers based on the SequenceFile.CompressionType used to compress key/value pairs:

Writer: Uncompressed records.

Record Compress Writer: Record-compressed files, only compress values.

Block Compress Writer: Block-compressed files, both keys & values are collected in 'blocks' separately and compressed. The size of the 'block' is configurable.

The actual compression algorithm used to compress key and/or values can be specified by using the appropriate Compression Codec. The recommended way is to use the static create Writer methods provided by the Sequence File to chose the preferred format. The Sequence File.Reader acts as the bridge and can read any of the above Sequence File formats. Sequence File Formats Essentially there are 3 different formats for Sequence Files depending on the Compression Type specified. All of them share a common header described below.

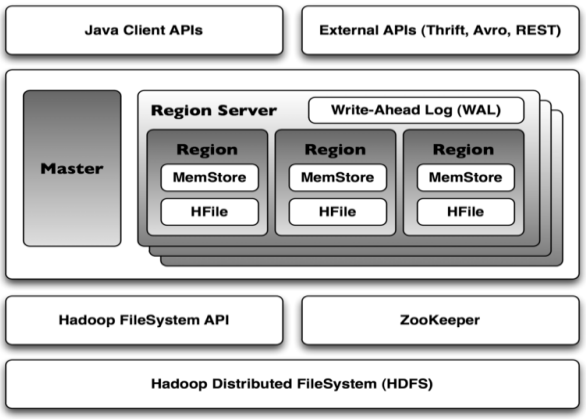
Sequence File Header

Version - 3 bytes of magic header **SEQ**, followed by 1 byte of actual version number (e.g. SEQ4 or SEQ6)
KeyClassName -key class
ValueClassName - value class
Compression - A boolean which specifies if compression is turned on for keys/values in this file.
BlockCompression - A boolean which specifies if block-compression is turned on for keys/values in this file.
Compression codec - CompressionCodec class which is used for compression of keys and/or values (if compression is enabled).
Metadata - SequenceFile.Metadata for this file.
Sync - A sync marker to denote end of the header.

3.4. HBase

HBase is a distributed database, meaning it is designed to run on a cluster of dozens to possibly thousands or more servers. As a result it is more complicated to install than a single RDBMS running on a single server. And all the typical problems of distributed computing begin to come into play

such as coordination and management of remote processes, locking, data distribution, network latency and number of round trips between servers. Fortunately HBase makes use of several other mature technologies, such as Apache Hadoop and Apache ZooKeeper, to solve many of these issues. The figure below shows the major architectural components in HBase.



In the above figure you can see there is a single HBase master node and multiple region servers. (Note that it is possible to run HBase in a multiple master setup, in which there is a single active master.) HBase tables are partitioned into multiple regions with each region storing a range of the table's rows, and multiple regions are assigned by the master to a region server.

HBase is a column-oriented data store, meaning it stores data by columns rather than by rows. This makes certain data access patterns much less expensive than with traditional row-oriented relational database systems. For example, in HBase if there is no data for a given column family, it simply does not store anything at all; contrast this with a relational database which must store null values explicitly. In addition, when retrieving data in HBase, you should only ask for the specific column families you need; because there can literally be millions of columns in a given row, you need to make sure you ask only for the data you actually need.

The HDFS component is the Hadoop Distributed Filesystem, a distributed, fault-tolerant and scalable filesystem which guards against data loss by dividing files into blocks and spreading them across the cluster; it is where HBase actually stores data. Strictly speaking the persistent storage can be anything that implements the Hadoop FileSystem API, but usually HBase is deployed onto Hadoop clusters running HDFS. In fact, when you first download and install HBase on a single machine, it uses the local filesystem until you change the configuration!

3.5. Hadoop Java Map-reduce Libraries

MapReduce has been emerging as a popular programming paradigm for data intensive computing in clustered environments such as enterprise data-centers and clouds. There has been an extensive use of the MapReduce as a framework for solving embarassingly parallel problems, using a large number of computers (nodes), collectively referred to as a cluster. These frameworks support ease computation of petabytes of data mostly through the use of a distributed file system. For example, the Google File System - used bythe proprietary 'Google Map-Reduce', or the 'Hadoop Distributed File System' used by Hadoop, an open source product from Apache.

In the "Map", the master node takes the input, divides it up into smaller sub-problems, and distributes those to work nodes. The worker node processes the smaller problem, and passes the answer back to its master node. In the "Reduce", the master node then takes the answers to all the sub-problems and combines them in a way to get the final output after reduces. The advantage of MapReduce is that it allows for distributed processing of the map and reduction operations. Provided each mapping operation is independent of the other, all maps can be performed in parallel and so is true for reduce.

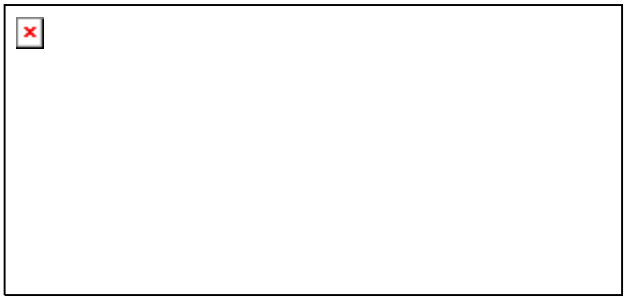


Figure above: Mapping creates a new output list by applying a function to individual elements of an input list.

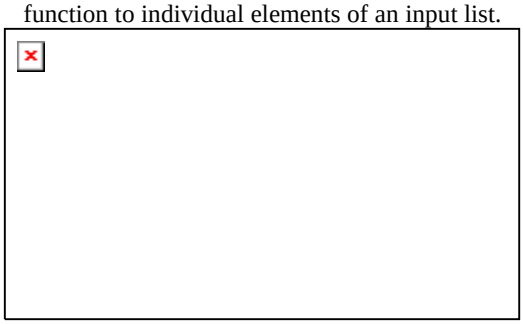


Figure above: Reducing a list iterates over the input values to produce an aggregate value as output.

3.6. MapReduce abstraction on top of CometCloud

We found that the file writes and reads to the distributed file system, and have an overhead especially for smaller data sizes in the order of few tens of GB's. Our solution provides the MapReduce programming framework built over the Comet framework which used TCP sockets for communication and coordination, and uses in-memory operations for data whenever possible. Our objectives were:

Understand the behaviors and limitations of MapReduce in the case of small to moderate data sets (understand what the cross over points are)

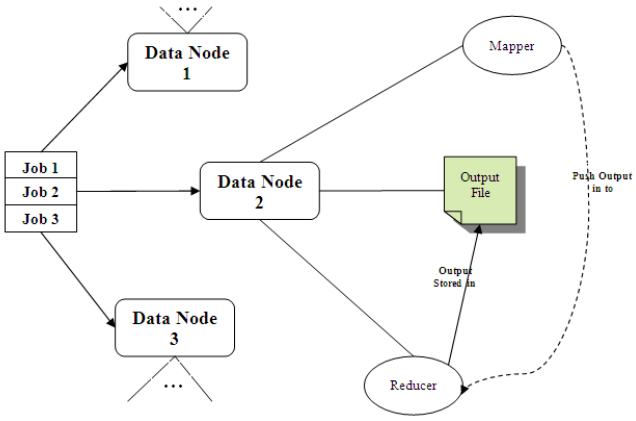
Develop coordination and interaction framework to complement MapReduce-Hadoop to address these shortcomings
Demonstrate and evaluate using a real world application

We use the Comet and its services to build a MapReduce infrastructure that addresses the above requirements - specifically enable pull based scheduling of Map tasks as well as stream based coordination and data exchange. The framework is based on the master-worker concept already supported by Comet. CometG is a decentralized (peer-to-peer) computational infrastructure that extends Desktop Grid environments to support applications that have high computational requirement. It provides a decentralized and scalable tuple-space, efficient communication and coordination support, and application-asynchronous iterative algorithms using the master-worker/BOT paradigm.

Our System's interfaces are similar to the Hadoop MapReduce framework, to make applications built on Hadoop easily portable to Comet-based framework. The details of the implementation and evaluation of an actual pharmaceutical problem, with its results have been described. We found out that solution can be used to accelerate the computations of medium sized data by delaying or avoiding the use of distributed file reads and writes.

4. Methodology

The experiment bed is setup with one name node and 3 data nodes. Name node have 8GB of RAM, 500 GB of HDD. Data nodes have 4 GB of RAM, 500 GB of HDD. All machine is powered with intel i5 1.7 Ghz speed.



The experiment ran for different size of big data sample ranging from 1GB to 10 GB data set and output is tabulated.

5. Result and Discussion

5.1. Sequential file

a) When you use sequential file as Source, at the time of Compilation it will convert to native format from ASCII whereas, when you go for using datasets conversion is not required. Also, by default sequential files we be Processed in sequence only. Sequential files can accommodate up to 2GB only. Sequential files does not support NULL values. All the above can me overcome using dataset Stage but selection is depends on the Requirement suppose if you want to capture rejected data in that case you need to use sequential file or file set stage.

b) Sequential file is used to Extract the data from flat files and load the data into flat files and limit is 2GB.Dataset is a intermediate stage and it has parallelism when load data into dataset and it improve the performance.

c) Data set mainly consists of two files.
- Descriptor file which consists of Metada,data location but not actual data itself
- Data file contains the data in multiple files and one file file per partition.
- Orchadmin command is used to delete the datasets where as rm unix command is used to remove the flat files.

5.2. Observation regarding HBase and HDFS

HDFS is a distributed file system and has the following properties:

- It is optimized for streaming access of large files. You would typically store files that are in the 100s of MB upwards on HDFS and access them through MapReduce to process them in batch mode.
- HDFS files are write once files. You can append to files in some of the recent versions but that is not a feature that is very commonly used. Consider HDFS files as write-once and read-many files. There is no concept of random writes.
- HDFS doesn't do random reads very well.

HBase on the other hand is a database that stores it's data in a distributed filesystem. The filesystem of choice typically is HDFS owing to the tight integration between HBase and HDFS. Having said that, it doesn't mean that HBase can't work on any other filesystem. It's just not proven in production and at scale to work with anything except HDFS. HBase provides you with the following:

- Low latency access to small amounts of data from within a large data set. You can access single rows quickly from a billion row table.
- Flexible data model to work with and data is indexed by the row key.
- Fast scans across tables.
- Scale in terms of writes as well as total volume of data.

6. Conclusion

HDFS reading and writing methods or technologies can selected depending upon the type of data or scenario we facing. We found that sequential file is slower than normal file system because of extra data overhead with sequential file and it can be used for data storage in memory but not HDFS. Similarly HBase kind of system is good for data with structure and where we need to extract data in columnar manner rather than row by row.

References

[1] Jefry Dean and Sanjay Ghemwat, MapReduce:A Flexible Data Processing Tool, Communications of the ACM, Volume 53, Issuse.1,January 2010, pp 72-77.
[2] Jefry Dean and Sanjay Ghemwat,.MapReduce: Simplified data processing on large clusters, Communications of the ACM, Volume 51 pp. 107–113, 2008

[3] Brad Brown, Michael Chui, and James Manyika, Are you ready for the era of „big data"?,McKinseyQuaterly,Mckinsey Global Institute, October 2011.
[4] DunrenChe, MejdlSafran, and ZhiyongPeng, From Big Data to Big Data Mining: Challenges, Issues, and Opportunities, DASFAA Workshops 2013, LNCS 7827, pp. 1–15, 2013.
[5] MarcinJedyk, MAKING BIG DATA, SMALL, Using distributed systems for processing, analysing and managing large huge data sets, Software Professional"s Network, Cheshire Data systems Ltd.
[6] OnurSavas, YalinSagduyu, Julia Deng, and Jason Li,Tactical Big Data Analytics: Challenges, Use Cases and Solutions, Big Data Analytics Workshop in conjunction with ACM Sigmetrics 2013,June 21, 2013.
[7] Kyuseok Shim, MapReduce Algorithms for Big Data Analysis, DNIS 2013, LNCS 7813, pp. 44–48, 2013.
[8] Raja.Appuswamy,ChristosGkantsidis,DushyanthNarayanan,OrionHodson,AntonyRowstron, Nobody ever got fired for buying a cluster, Microsoft Research, Cambridge, UK, Technical Report,MSR-TR-2013-2
[9] Carlos Ordonez, Algorithms and Optimizations for Big Data Analytics: Cubes, Tech Talks,University of Houston, USA.
[10] Spyros Blanas, Jignesh M. Patel,VukErcegovac, Jun Rao,Eugene J. Shekita, YuanyuanTian, A Comparison of Join Algorithms for Log Processing in MapReduce, SIGMOD"10, June 6–11, 2010, Indianapolis, Indiana, USA.
[11] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein,JohnGerth, Justin Talbot,KhaledElmeleegy, Russell Sears, Online Aggregation and Continuous Query support.
[12] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in USENIXSymposium on Operating Systems Design and Implementation, San Francisco, CA, Dec. 2004, pp. 137–150.
[13] S. Ghemawat, H. Gobioff, and S. Leung, "The Google File System." in ACM Symposium on Operating Systems Principles, Lake George, NY, Oct 2003, pp. 29 – 43.
[14] HADOOP-3759: Provide ability to run memory intensive jobs without affecting other running tasks on the nodes. https://issues.apache.org/jira/browse/HADOOP-3759
[15] VinayakBorkar, Michael J. Carey, Chen Li, Inside "Big Data Management":Ogres, Onions, or Parfaits?, EDBT/ICDT 2012 Joint Conference Berlin, Germany,2012 ACM 2012, pp 3-14.
[16] GrzegorzMalewicz, Matthew H. Austern, Aart J. C. Bik, James C.Dehnert, Ilan Horn, NatyLeiser, and GrzegorzCzajkowski,Pregel: A System for Large-Scale Graph Processing, SIGMOD"10, June 6–11, 2010, pp 135-145.
[17] Hadoop,"PoweredbyHadoop,"http://wiki.apache.org/hadoop/PoweredBy.
[18] PIGTutorial,YahooInc.,http://developer.yahoo.com/ hadoop/ tutorial/pigtutorial.html
[19] Apache: Apache Hadoop, http://hadoop.apache.org
[20] Apache Hive, http://hive.apache.org/
[21] Apache Giraph Project, http://giraph.apache.org/
[22] Mahout, http://lucene.apache.org/mahout/
[23] Amazon Simple Storage Service (Amazon S3). http://aws.amazon.com/s3/
[24] Windows.Azure. Storage .http://www.microsoft.com /windowsazure/ features /storage/
[25] The Age of Big Data. Steve Lohr. New York Times, Feb 11, 2012. http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html
[26] Information System & Management, ISM Book, 1st Edition 2010, EMC2, Wiley Publishing
[27] Dryad - Microsoft Research, http://research. microsoft.com /en-us/projects/dryad/
[28] IBM-What.is.Jaql, www .ibm.com/ software /data /infosphere /hadoop/jaql/

Author Profile

Sunil Kumar S. is Asst Proffessor, MITE , Moodbidri. He is Publised many papers and research journals on Wireless sonsor networking and technologies. Author has completed Diploma, BE and MTech in Software Engg. From MIT university . Currently doing research in Big-Data, cloud computing, Hadoop and grid computing.

Sanjeev G Kanabargi is MTech Student, MITE Moodbidri. Author has completed his Diploma (CSE), BE (ISE) and perusing his Mtech in MITE, moodbidri. He has worked in "Persistent Systems" as software developer for 2 yrs. Currently working with research work in Big-Data and Hadoop.