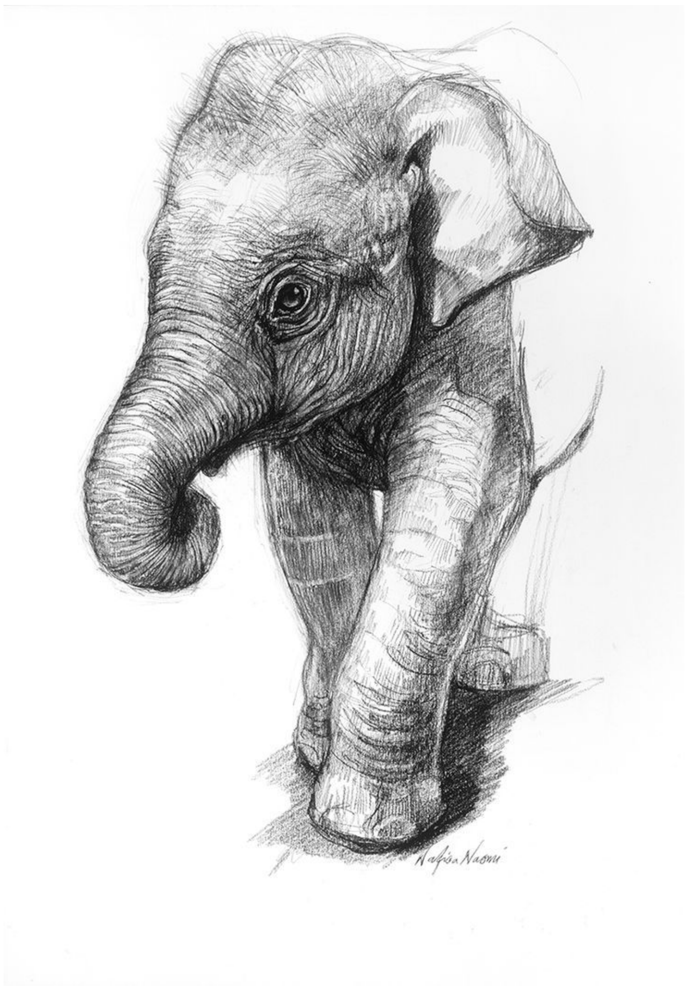


Apache Hadoop

A Brief Introduction to the Hadoop Ecosystem and the Hadoop Distributed File System (HDFS)

London South Bank University, Januar 6, 2019
School of Engineering, Data Science MSc

Module: Statistical Analysis and Modelling
Lecturer: Dr Christos Chrysoulas
Author: Patric Oliver Weber
Version: 1.0

Contents

List of Figures

1 Introduction

Humanity has been collecting data for millenia, and in fact, recording data or keeping records defines our history. With our fast developing environment, 90 per cent of all the data ever recorded has been captured over just the last two years. With the new era of Big Data, we get powerful analytics, algorithms and Artificial Intelligence (AI) as well as cheap storage. Big Data affects us in almost every industry and job. However, one major consequence of ubiquitous and cheap storage is that data deletion costs more, hence it is easier to keep all our data. This is no longer possible to process with traditional tools. Big data has become too complicated to handle and has demanded new tools, as well as a new understanding of its nature. Hadoop is a large-scale operating system and can handle this data. Apache Hadoop, an open-source software appears in the suite of nearly every organisation, with one goal - to handle the massive amounts of data. This paper aims to introduce the reader to Apache Hadoop and its unique file system. Furthermore, a big data lab together with the Cloudera Virtual machine will show what the environment of Hadoop looks like.

Figure 1: Apache Hadoop logo (source: `hadoop.apache.org/`)

2 State of the Art

When collecting vast amounts of data, one might question what it can be used for. Big data affects every job, at work or at home, and produces massive amounts of data that multiple organisations collect, analyse and act on for their profit. New products connected to the internet are creating the Internet of Things (IoT); this shift enables every organisation, regardless of its size, to embrace big data and analytics as part of its business model. [Marr, 2016]

With these emerging trends, there are many new practices for corporations to learn. It is essential that companies invest in a digital strategy [Van Rijmenam, 2014, Marr, 2016]. However, with a new strategy, there are new core tasks to solve [Van Rijmenam, 2014]. Currently, there are five major tasks for the IT workforce to overcome, namely:

1. Handle big data ideally in real time, and from many multiple sources.
2. Visualise data.
3. Allow predictions and modelling.
4. Profile customers and users.
5. Recommend further action based on the predictions.

Most companies are placing their trust in a variety of open source and proprietary tools for their data science operations. There is one tool, named Hadoop, which appears in the suite of all corporations. [Marr, 2016]
Hadoop is an open source system that can be considered a "large-scale operating system" which has been revolutionary in the way it handles significant amounts of data. Hadoop is easily scalable due to its implementation in Java which runs on all platforms around the world. MapReduce, a particular algorithm, is integrated into Hadoop so that the system can work with both structured and unstructured data, and divides data into useful units for processing [Van Rijmenam, 2014].

Apache Hadoop is a collection of many open-source software tools which simplifies the storage of big data. Hadoop is designed to be used in up to thousands of machines, where each offers its own local computation and storage. The library is designed to detect and handle failures at the application layer and thus it delivers excellent high availability in a cluster of computers where each might be prone to failure. It was specifically made for commodity hardware, however it has also found use in high-end server farms [Woodie, 2014, Hemsoth, 2014].

This is the open source project of the Apache Software Foundation(ASF), but there are many commercial sellers of this big data software. There are four big vendors of this big data platform: Amazon Web Service, Cloudera, Hortonworks and MapR Technologies. Additionally, there are some other big players such as Google and Microsoft that have built upon these platforms and sell these separately as cloud-based managed services [Rouse, 2014].

Now, with better handling of Big Data, big enterprises such as Amazon, Facebook, eBay and LinkedIn are using Hadoop to mine and interpret their data stashes. Thanks to the MapReduce algorithm with its unique shuffle and reduce principles, it has become easy to compare files against each other. With that, a new marketplace has arisen, and "reputation" is the new combat zone. Every collectable footprint can form an analyzable dataset. These datasets are then used by algorithm firms like Delphi Analytics or SAS to generate "reputation scores" [Fertik and Thompson, 2015]. These far advanced algorithms do not know the individual or any preferences. They use the numerical trail for comparison and ranking: reputation, health and scores which define if the individual is trustworthy, like credit scores. This can generate very valuable data, not only for leading corporations and insurers but also for bosses and partners [Fertik and Thompson, 2015].

3 Apache Hadoop

The Hadoop Framework has been described by Hortonworks as “an open source software platform for distributed storage and distributed processing of very large datasets on computer clusters built from commodity hardware”.

Hadoop splits files into big chunks and will then distribute these data across many computers(nodes) in a cluster. It also transfers packaged code into the nodes to process the data in parallel. With the parallel approach there is the significant advantage of the data locality. The nodes manipulate the revived data with de MapReduce algorithm. Within this algorithm, it will map, shuffle and reduce the data. This will ultimately allow the dataset to be processed much faster and more efficiently. Hadoop works much faster than other supercomputer architectures [Wang et al., 2014]. All the processed output, as well as the shuffled and reduced data, will then be stored on the File System (HDFS).

The Framework was created by two computer scientists Doug Cutting and Mike Cafarella. Both developers were initially assisting on another open source search engine project named Nutch. At that time, Doug was working for Yahoo! when Google massively released white papers of its new Google File Systems (GFS) and its unique algorithm MapReduce Framework from 2003 and 2004, which had only been used for lab experiments before. At that time, the two developers took a keen interest in this and modified early versions by developing a Java-based MapReduce application together with the GFS. The project was named after the first word spoken by Doug’s two-year-old son; that word was the name of a yellow stuffed elephant, “Hadoop”. [Bappalige, 2014, Rouse, 2014]

Hadoop helps to overcome many issues. Firstly, Hadoop offers a new way of data storage with no bottleneck of scalability. Adding a new cluster works in a linear way instead of horizontally [Frank Kane, 2018]. This means by adding one new cluster you can get more performance and more data.
Secondly, it can be used as an additional Input for the Enterprise Data Warehouse for crawling new big data, namely real-time data. Thirdly, Hadoop is a new platform for business intelligence and analytics [Hanson, 2011].

In short, it can be said that Hadoop was initially created to solve problems with Relational Database Management System (RDBMS) and Data-Warehouses (DWH). [Torr, 2014]. This can also been seen in Figure 2.

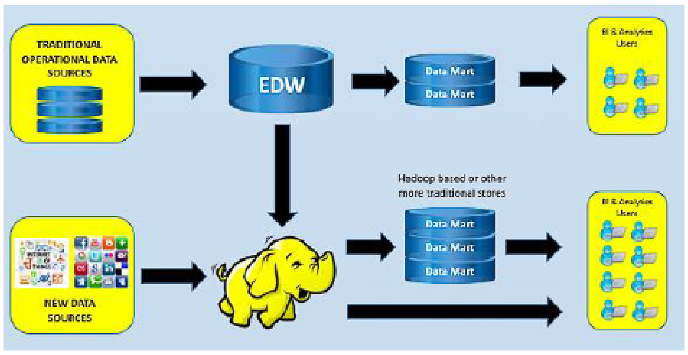


Figure 2: Ways to use Hadoop (source:Torr [2014])

In summary, we can conclude that Hadoop is vital for:

- Storing vast amounts of real-time data in a new way, quickly
- Computing power: with each node you get storage and processing power
- Hardware redundancy: when a node goes down, jobs are automatically redirected
- Flexibility: as it can be used as Data Lake, there is no need to preprocess data and go through the ETL (Extract, Transform, Load) procedure
- Low cost: As it comes as an open-source framework and made for low-cost hardware
- Scalability: with its linear scalability, nodes can offer more processing power as well as data storage

3.1 Components of Hadoop

The framework is made up of the following packages:

- Hadoop Common: This is a container for libraries and tools which are being used by other modules.
- Hadoop Distributed File System (HDFS): This is the filesystem in the cluster that works on commodity software. This system offers a very high aggregation within the cluster.
- Hadoop Yet Another Resource Negotiator (YARN); a 2012 implemented module that handles and schedules the computing resources within a cluster.

- Hadoop MapReduce: this is the algorithm for large-scale data handling.

These modules are just the main pillars in the Hadoop ecosystem. As this is a framework, there are many sub-modules and additional packages available that can be implemented together with Hadoop. Figure 3 shows what the Ecosystem of Hadoop looks likes. When people say "Hadoop", they are mostly referring to the two main components: HDFS and the MapReduce Algorithm.

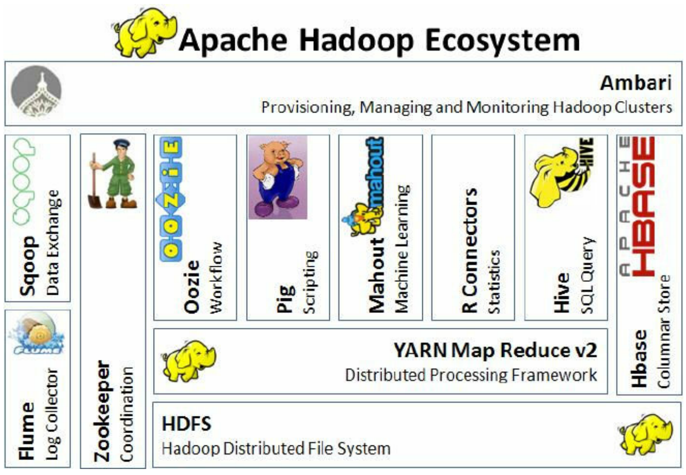


Figure 3: Overview of the Hadoop Ecosystem (source: Pereira and Capitão [2014a])

There are many good videos available on the internet that explain the essentials of MapReduce which can be watched on YouTube. "Hadoop, the Story"[1] makes an analogy with villagers that shows in a very abstract way how Hadoop works and Geoffrey Challen shows how the algorithm works[2].

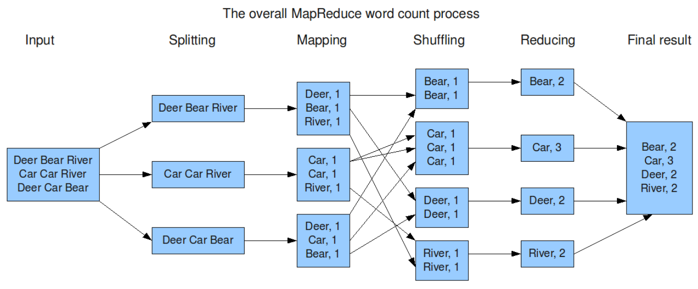


Figure 4: Hadoop MapReduce algorithm (source: Pereira and Capitão [2014b])

[1]`https://www.youtube.com/watch?v=h2LzEvPU4jY`
[2]`https://www.youtube.com/watch?v=43fqzaSH0CQ`

4 Hadoop Distributed File System (HDFS)

In this section, the Hadoop Distributed File System will be explained in terms of more detailed architecture.

HDFS is a file system or the storage layer of Hadoop. HDFS takes care of where the data will be saved and can storage with massive amounts of files, even on a petabyte scale [Kerzner and Maniyam, 2008]. HDFS provides the opportunity to connect commodity computer (nodes) in its system and include them into a cluster. When accessing the storage, it appears as one seamless file system. The File System handles this with a streaming method so that the commands or applications are directly executed in MapReduce. Additionally, the file system is fault tolerant and, with the high-throughput approach, very efficient at accessing the data [Wang et al., 2014, Malak, 2014].

When comparing the HDFS against another file system, there are many similarities. Nevertheless, there are some differences in several respects. There are three noticeable changes. [White, 2012, p. 41-42]

Streaming data access

The motivation behind streaming is that the write-once and read many times pattern is more efficient.

Commodity hardware

The system is made for low-end hardware for which the chance of failure in a cluster is high. However, HDFS is designed to load-balance a continuous workstream without any noticeable interruption for the end-user. Hadoop refers to the commodity servers as including a regular server which can be ordered from various vendors such as HP or Dell.

Huge files

Files can be as big as hundreds of gigabytes or even go up to the petabyte level.

By maintaining the primary focus on big data and saving huge files, HDFS does not work as well with low-latency data access, as it is designed for high throughput of data and that is at the expense of its latency. Secondly, lots of small files can also burst the memory storage of the file directory in the memory of the Name Node. Further, when doing BI, the system does lots of arbitrary file modification. HDFS is designed to be a single writer and does only write once, thus it clearly shows a bottleneck to support these fast modifications [White, 2012, p. 42].

4.1 HDFS Architecture

The cluster of an HDFS system comprises different nodes where files and directories are saved. A single node, known as Name Node manages the system and namespace, thus regulates the client access to files. In other terms, this is also considered the master worker. A Data Node will then store the data as blocks within the filesystem and is the worker[Bappalige, 2014]. This represents traditional hierarchical data management [Hanson, 2011].
The big difference between Master and Slaves (worker) computers is that the Master contains two additional components. These are the Job Tracker and the Name Node. Besides this, the master manages all requests sent by the client. The Master which manages the jobs (JobTracker) is designed to delegate services to the slaves. The Slave services can also be seen as TaskTrackers and are distributed to each node. This means that the JobTraker manages the TaskTrackers and assigns Hadoop MapReduce tasks to them.

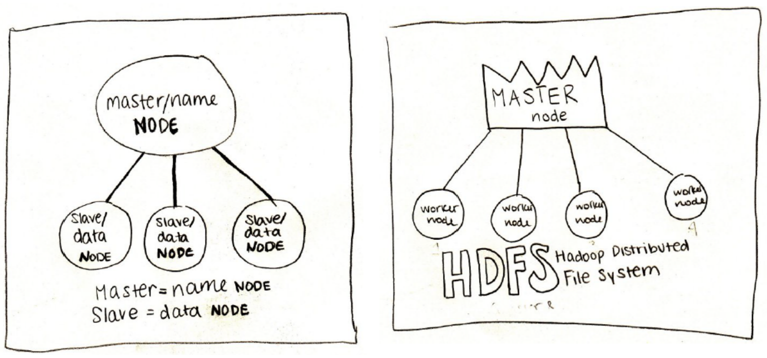


Figure 5: Architecture of HDFS (source: Kerzner and Maniyam [2008])

4.2 Name Node and Data Nodes

The Name Node coordinates the FS namespace tasks and metadata; this includes accessing, opening, closing and renaming files and whole directories. Moreover, the name nodes create, erase and replicate data on behalf of the governing name node. All communication in the HDFS cluster is built on the TCP/IP protocol. Thus, it can be concluded that the Name Node oversees all operations and that without a Name Node in a cluster, the ecosystem would be inoperable. This is the doorkeeper and is called the Single Point of Failure. Since the Name node is so important, it is essential to make the Name Node resilient to failure. If the Name Node goes down and cannot be restored, there is no way of knowing how to reconstruct the cluster. Hadoop offers two ways of addressing this issue, by back up the persistent status of the metadata to other filesystems or to implement a secondary Name Node. Despite the fact that switching to the other Name Node causes delays, data loss is practically inevitable. In the case of a total failure, a state-of-the-art approach is to combine both methods - once the Name Node fails, upload the backup to the secondary name node and assign it to the new primary. [Hanson, 2011, White, 2012, p. 44]

4.3 Data Storage and Replication in HDFS

Conventional file systems, for instance, Linux EXT or NTFS from Windows will store files by varying their size; this could be a few bytes or up several gigabytes. Unlike HDFS, it can only save to a few gigabytes. This is because HDFS was built for mechanical hard disk drives (HDD). In accordance with Moore's law, capacities are going up each year, yet the search times have not improved overall. HDFS addresses this matter by minimising disk search times by saving smaller chunks. Furthermore, HDFS writes files only once. This means that they cannot be updated. However, appending to the files is supported, which is included in the suite of HBase, a module of the Hadoop Ecosystem. [Kerzner and Maniyam, 2008, p. 28]

With regard to to the saving method of HDFS, a typical hard disk has a predefined block size; this block represents a container with a minimum amount of data to read and write. Many File Systems are built upon this and already begin to slice at a few kilobytes, whereas HDD's blocks usually start at 512 bytes. Most of the time that it matches the original file length. HDFS has implemented the same strategy with chunks, but the blocks are much bigger units. By default, HDFS sets 64 MB. All data saved on the Hadoop ecosystem will be sliced into 64 MB blocks and will then be stored as independent files within the cluster. When the file is smaller as a single block, HDFS does not occupy a whole block. HDFS implemented this strategy due to the better performance in seek times; this approach drastically decreases seek times to make blocks large enough, so that the head of an HDD does need to take much time to transfer to another start of a block. [White, 2012, p, 45]

With this new implementation comes much simplicity; namely, as blocks are individual, they can be saved anywhere across the cluster and do not need to be stored on the same disk. Secondly, making everything in 64 MB chunks it gets easier with calculations, and furthermore, the block sizes are in favour of data replication [White, 2012, p. 43]. In general, Hadoop replicates each block three times where each will be saved on physically separate machines. Once a block becomes corrupt, or the node fails, it will then replicate the block by the coordination of the Name Node to one of the other locations [Borthakur, 2018].

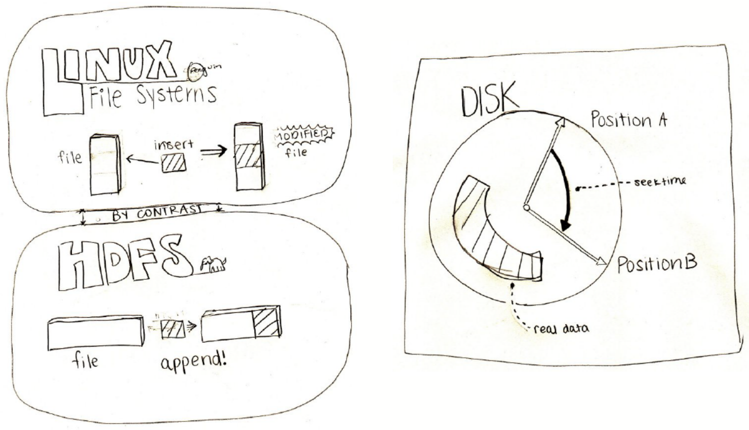


Figure 6: HDFS file append and disk seek time (source: Kerzner and Maniyam [2008])

4.3.1 Example

Imagine a file with the name data.txt, with the size of 150MB, will be saved on the cluster. HDFS will split these into three chunks as it can be seen in figure 7. These files are getting a unique name, for instance, blk_1, blk_2, blk_3. In total, this makes up three files, where two have 64 MB, and the leftover has 22 MB. All the blocks get stored three times in the cluster. To know where are the blocks saved and how the blocks make up the original file, all the metadata will be transmitted to the daemon called Name Node.

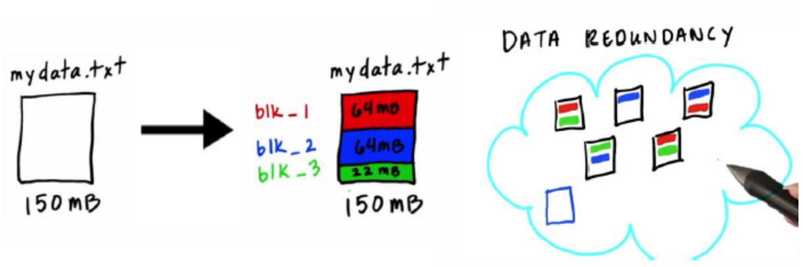


Figure 7: Example of saving a file in HDFS (source: [Sproehnle et al., 2015])

Let's see this in more detail, accompanied by Figure 8:

1. The HDFS client sends a Create request to the Name Node.
2. The Name Node checks if the file exists.
3. The Name Node creates a metadata file of mydata.txt (name, size).
4. The Name Node sends the HDFS client a message. This contains the numbers of the Slaves on which the blocks will be stored.

5. The first block (blk_1) is converted into a Data Node of a slave.
6. The block (blk_1) is written twice into a different Data Node duplicated in the cluster (default value of Hadoop = 2 replications).
7. Each Data Node will now send an acknowledged packet (ACK) to the previous node so that the Name Node knows where the blocks are.
8. The first node sends an ACK to the Name Node.
9. The Name Node sends an ACK packet to the client. Now the client knows that it can process the remaining blocks.
10. Once all blocks have been stored in the Data Node, the Name Node reports Complete status to the client.
11. The different Data Nodes now send a block report to the Name Node. This contains the block names and the storage locations, which are written into the metadata file.

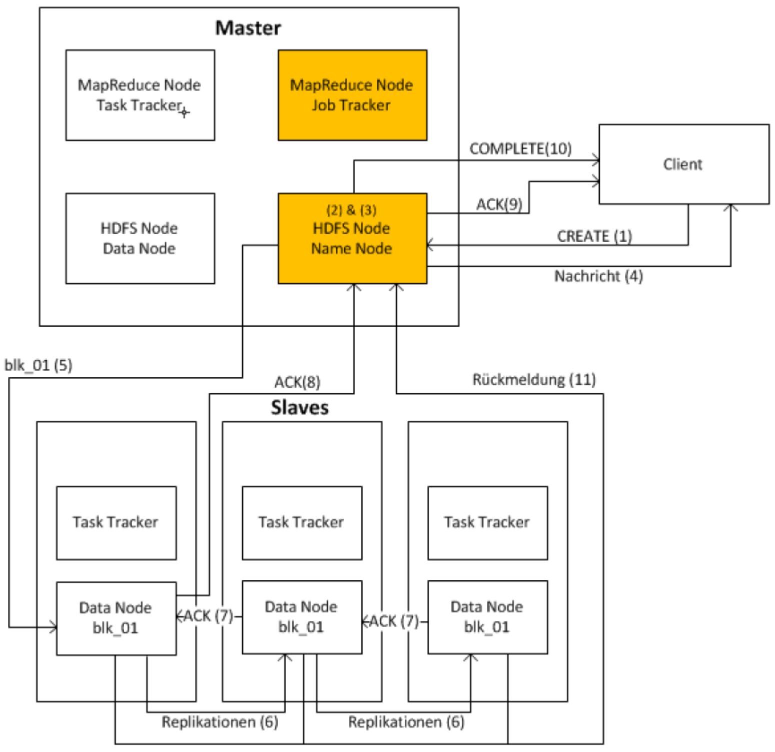


Figure 8: Process of a File creation in HDFS (source: hslu.ch)

4.4 Application Interfaces into HDFS

HDFS can be accessed in different ways. With the implementation of the native Java application (API) programming and the C-language wrapper classes, there are many implementations possible to create a graphical user interface (GUI). There is also a web interface to browse the stored files.
The table 1 shows the different applications of the HDFS file system.

Table 1: Table of different Interfaces for HDFS

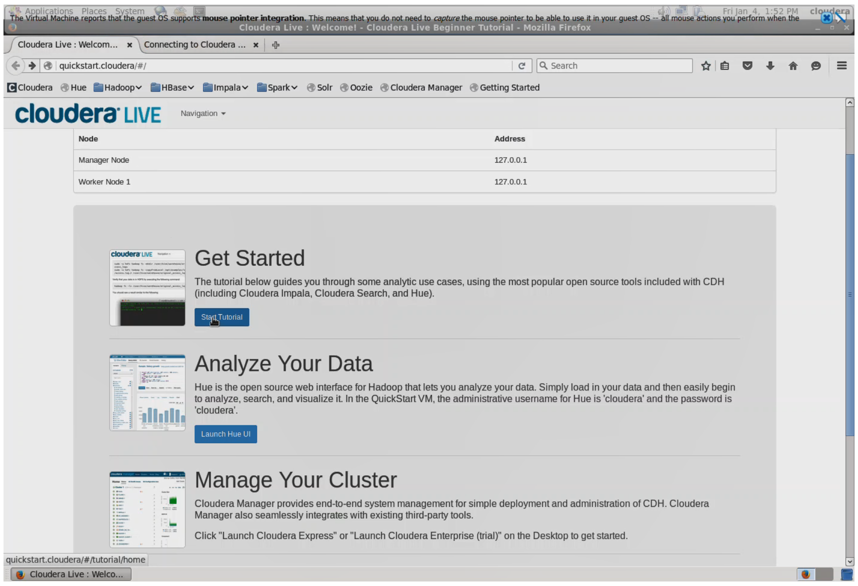
Application	Description
File System (FS) shell	This is a command line interface (CLI) which is similar to UNIX that allows to fire off commands in HDFS.
DFSAdmin	This is an administrative tool to control a Cluster.
Fsck (FS maintenance)	An important subcommand for the Hadoop environment. The statement allows it to check for inconsistencies in files, for instance for missing blocks. However, blocks can be repaired with this statement.
Data nodes and name node	These have built-in web servers that allow an administrator to check and verify the status of the cluster.

5 Big Data Lab Cloudera

To get an insight of what Hadoop is capable of, and understand how the process works, it is suggested to work with a predefined virtual machine. Cloudera offers a QuickStart VM[3] which representsa single-node cluster which is very well thought-out and intuitive. With the predefined tutorials and sample data, Cloudera makes it easy to get the right overview of what big data is capable of.

With this implementation, the cutting-edge technology is included directly from the Chief Architect Doug Cutting of Cloudera, together with humorous anecdotes. Cutting works for Cloudera and has also been Chairman of the Apache Software Foundation (ASF) since 2010.

To get started with the VM, the author recommends working through the tutorials and then, optionally, starting to set up a multi-node cluster, if the main focus is, as in this paper, the File System. In short, shown below are interfaces of the environment.



The Virtual Machine reports that the guest OS supports **mouse pointer integration.** This means that you do not need to *capture* the mouse pointer to be able to use it in your guest OS -- all mouse actions you perform when the

Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox

Node	Address
Manager Node	127.0.0.1
Worker Node 1	127.0.0.1

Get Started

The tutorial below guides you through some analytic use cases, using the most popular open source tools included with CDH (including Cloudera Impala, Cloudera Search, and Hue).

Analyze Your Data

Hue is the open source web interface for Hadoop that lets you analyze your data. Simply load in your data and then easily begin to analyze, search, and visualize it. In the QuickStart VM, the administrative username for Hue is 'cloudera' and the password is 'cloudera'.

Manage Your Cluster

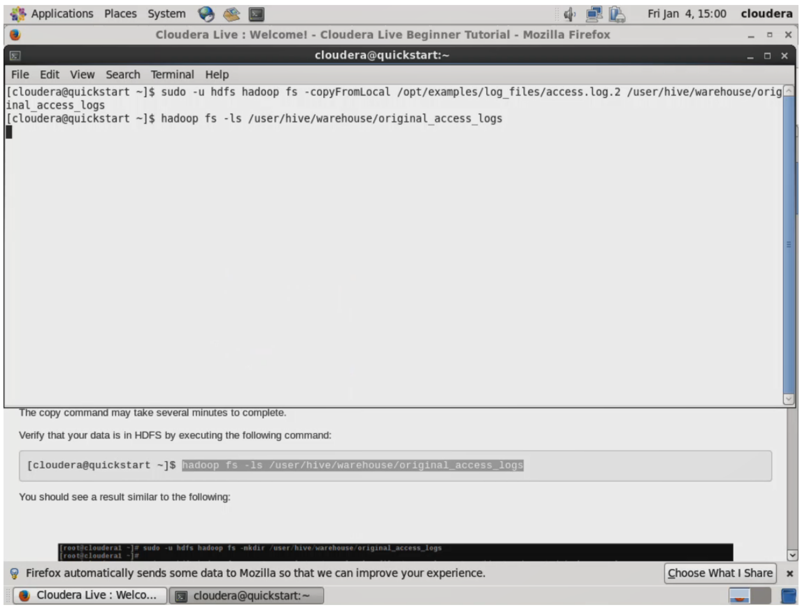
Cloudera Manager provides end-to-end system management for simple deployment and administration of CDH. Cloudera Manager also seamlessly integrates with existing third-party tools.

Click "Launch Cloudera Express" or "Launch Cloudera Enterprise (trial)" on the Desktop to get started.

Figure 9: Start Screen of Cloudera VM

[3]`https://www.cloudera.com/downloads/quickstart_vms/5-13.html`

Figure 9 shows the startup of the VM; with the first button, *"Getting Stated"*, a four-step intuitive tutorial shows the whole environment. Furthermore, it guides you through the most crucial Web Interfaces on the Cloudera Platform. Figure 10 demonstrates the command line interface along with HDFS commands from their tutorial. In Figure 11 you can see with three simple oneliners - how to create a text file and how to upload the data into the cluster.


Applications Places System
Fri Jan 4, 15:00 cloudera
Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox
cloudera@quickstart:~
File Edit View Search Terminal Help

```
[cloudera@quickstart ~]$ sudo -u hdfs hadoop fs -copyFromLocal /opt/examples/log_files/access.log.2 /user/hive/warehouse/orig
inal_access_logs
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/original_access_logs
```

The copy command may take several minutes to complete.

Verify that your data is in HDFS by executing the following command:

```
[cloudera@quickstart ~]$ hadoop fs -ls /user/hive/warehouse/original_access_logs
```

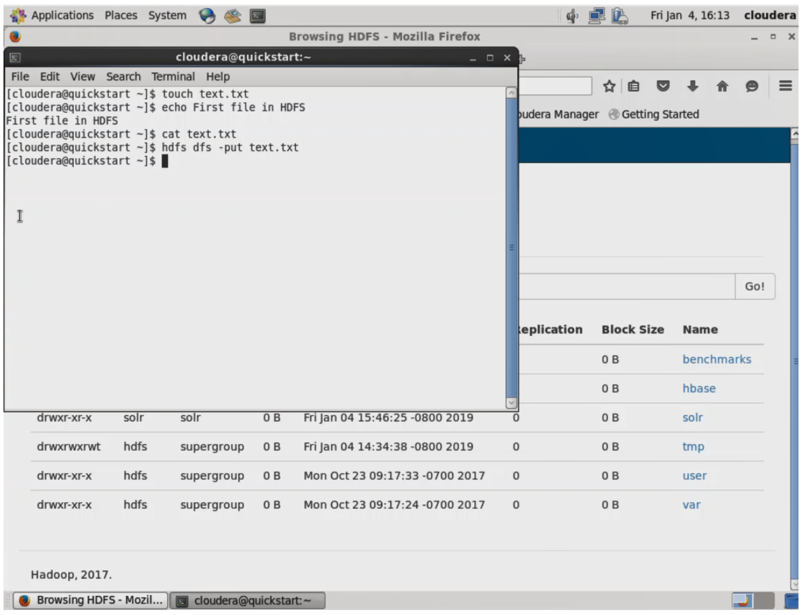
You should see a result similar to the following:

```
[root@cloudera1 ~]# sudo -u hdfs hadoop fs -mkdir /user/hive/warehouse/original_access_logs
[root@cloudera1 ~]#
```

Firefox automatically sends some data to Mozilla so that we can improve your experience.

Figure 10: Command-line Interface Hadoop

To see where the data is stored, the HDFS Web Service can be accessed with the right port (here: 8020). The Interfaces provides you with the status of the current single node. With the Register Utilities, the File Explorer opens and in Figure 13 you can see the stored file.

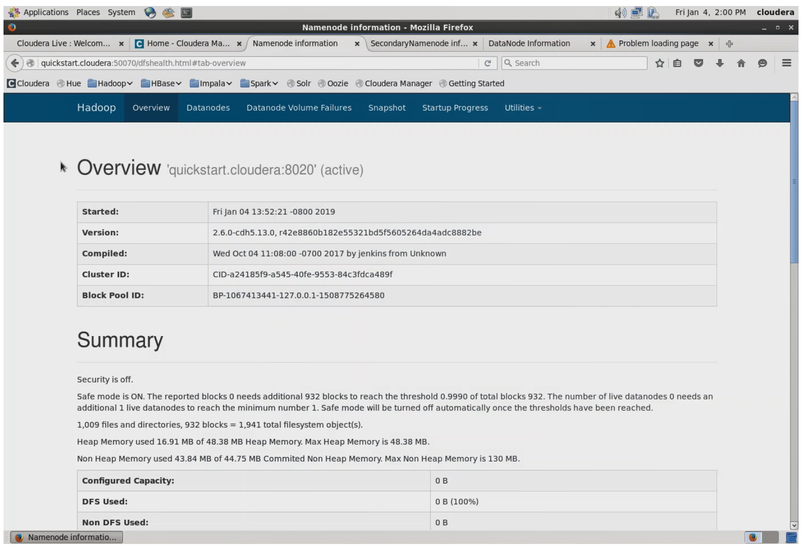


```
[cloudera@quickstart ~]$ touch text.txt
[cloudera@quickstart ~]$ echo First file in HDFS
First file in HDFS
[cloudera@quickstart ~]$ cat text.txt
[cloudera@quickstart ~]$ hdfs dfs -put text.txt
[cloudera@quickstart ~]$
```

```
                                                                   eplication   Block Size   Name
                                                                                0 B          benchmarks
                                                                                0 B          hbase
drwxr-xr-x   solr   solr         0 B   Fri Jan 04 15:46:25 -0800 2019   0       0 B          solr
drwxrwxrwt   hdfs   supergroup   0 B   Fri Jan 04 14:34:38 -0800 2019   0       0 B          tmp
drwxr-xr-x   hdfs   supergroup   0 B   Mon Oct 23 09:17:33 -0700 2017   0       0 B          user
drwxr-xr-x   hdfs   supergroup   0 B   Mon Oct 23 09:17:24 -0700 2017   0       0 B          var

Hadoop, 2017.
```

Figure 11: Commands to create and store a text file on Hadoop



Namenode information - Mozilla Firefox
quickstart.cloudera:50070/dfshealth.html#tab-overview
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities

Overview 'quickstart.cloudera:8020' (active)

Started:	Fri Jan 04 13:52:21 -0800 2019
Version:	2.6.0-cdh5.13.0, r42e8860b182e55321bd5f5605264da4adc8882be
Compiled:	Wed Oct 04 11:08:00 -0700 2017 by jenkins from Unknown
Cluster ID:	CID-a24185f9-a545-40fe-9553-84c3fdca489f
Block Pool ID:	BP-1067413441-127.0.0.1-1508775264580

Summary

Security is off.

Safe mode is ON. The reported blocks 0 needs additional 932 blocks to reach the threshold 0.9990 of total blocks 932. The number of live datanodes 0 needs an additional 1 live datanodes to reach the minimum number 1. Safe mode will be turned off automatically once the thresholds have been reached.

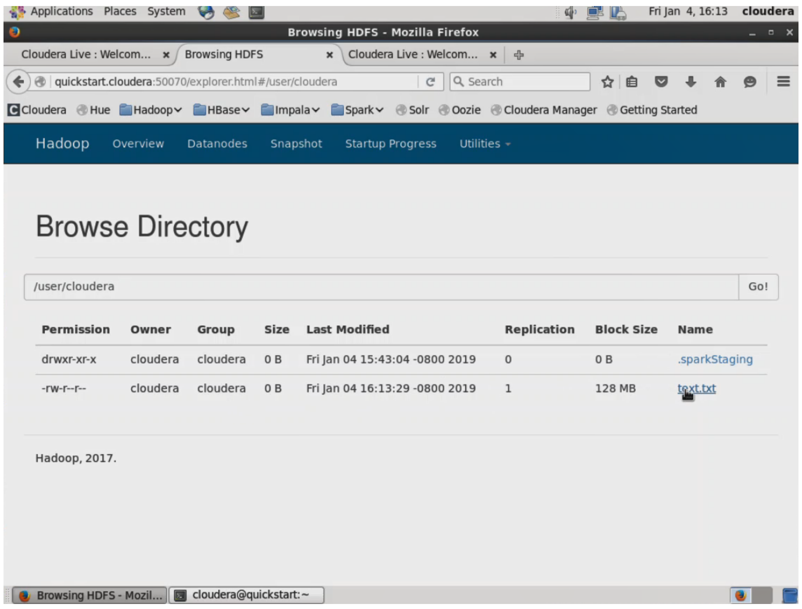
1,009 files and directories, 932 blocks = 1,941 total filesystem object(s).

Heap Memory used 16.91 MB of 48.38 MB Heap Memory. Max Heap Memory is 48.38 MB.

Non Heap Memory used 43.84 MB of 44.75 MB Commited Non Heap Memory. Max Non Heap Memory is 130 MB.

Configured Capacity:	0 B
DFS Used:	0 B (100%)
Non DFS Used:	0 B

Figure 12: HDFS Web-interface


Applications Places System
Fri Jan 4, 16:13 cloudera
Browsing HDFS - Mozilla Firefox
Cloudera Live : Welcom...
Browsing HDFS
Cloudera Live : Welcom...
quickstart.cloudera:50070/explorer.html#/user/cloudera
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Hadoop Overview Datanodes Snapshot Startup Progress Utilities
Go!
Browsing HDFS - Mozil...
cloudera@quickstart:~

Browse Directory

/user/cloudera

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	cloudera	cloudera	0 B	Fri Jan 04 15:43:04 -0800 2019	0	0 B	.sparkStaging
-rw-r--r--	cloudera	cloudera	0 B	Fri Jan 04 16:13:29 -0800 2019	1	128 MB	text.txt

Hadoop, 2017.

Figure 13: Commands to create and store a text file on Hadoop

6 Conclusion

With the new features, there are also some new ways in which data must be handled and, with this, there are some challenges.

Managing Big Data can be very complex as this is a whole new “large-scale operating system” and works inside its own wheel. That means it is difficult to find programmers who are able to reproduce or amend the MapReduce for the needs of the business. This is almost on the border between art and data science; that is why there are many enterprise-built SQL on top of Hadoop.
MapReduce is a very powerful algorithm for simple information, but when it comes to doing iterative and interactive analytics MapReduce shows a lack of performance as it is very file-intensive. This is because the nodes are not communicating with each other except through the algorithm with sort and shuffle functions. While doing interactive analytics, which literally means pulling and storing data within microseconds, it creates multiple files that would require some sort of an iterative algorithm to compute faster. With the status quo implementation, MapReduce creates multiple files, and this makes it inefficient for advanced Business Intelligence.
Hadoop is not yet a full-fledged data management ecosystem and governance. In the era of Data Privacy, it is essential to be able to conduct proper data cleansing, governance, and to store and maintain metadata. Hadoop still lacks data quality tools and standardization.

Bibliography

Mark Torr. 3 ways to use Hadoop without throwing out the DWH, October 2014. URL http://www.marktorr.com/3-ways-to-use-hadoop-without-throwing-out-the-dwh/.

Óscar Pereira and Micael Capitão. Figure 2.1: Hadoop ecosystem overview. 1, December 2014a. URL https://www.researchgate.net/figure/Hadoop-ecosystem-overview-1_fig2_270448794.

Óscar Pereira and Micael Capitão. Figure 2.5: Word count program flow executed with MapReduce. 5, December 2014b. URL https://www.researchgate.net/figure/Word-count-program-flow-executed-with-MapReduce-5_fig6_270448794.

Mark Kerzner and Sujee Maniyam. *Hadoop Illuminated*. 2008.

Sarah Sproehnle, Ian Wrigley, and Gundega Dekena. Intro to Hadoop and MapReduce | Udacity, February 2015. URL https://eu.udacity.com/course/intro-to-hadoop-and-mapreduce--ud617.

Bernard Marr. *Big data in practice: how 45 successful companies used big data analytics to deliver extraordinary results*. Wiley, Chichester, West Sussex, 2016. ISBN 978-1-119-23138-7.

Mark Van Rijmenam. *Think bigger: developing a successful big data strategy for your business*. AMACOM, American Management Association, New York, 2014. ISBN 978-0-8144-3415-4.

Alex Woodie. Why Hadoop on IBM Power, May 2014. URL https://www.datanami.com/2014/05/12/hadoop-ibm-power/.

Nicole Hemsoth. Cray Launches Hadoop into HPC Airspace, October 2014. URL https://www.hpcwire.com/2014/10/15/cray-launches-hadoop-hpc-airspace/.

Margaret Rouse. What is Hadoop? - Definition from WhatIs.com, 2014. URL https://searchdatamanagement.techtarget.com/definition/Hadoop.

Michael Fertik and David C. Thompson. *The reputation economy: how to optimize your digital footprint in a world where your reputation is your most valuable asset*. Crown Business, New York, first edition edition, 2015. ISBN 978-0-385-34759-4.

Y. Wang, R. Goldstone, W. Yu, and T. Wang. Characterization and Optimization of Memory-Resident MapReduce on HPC Systems. pages 799–808, May 2014. doi: 10.1109/IPDPS.2014.87.

Sachin Bappalige. An introduction to Apache Hadoop for big data, August 2014. URL `https://opensource.com/life/14/8/intro-apache-hadoop-big-data`.

Frank Kane. What is Hadoop?, March 2018. URL `https://www.youtube.com/watch?v=DCaiZq3aBSc&t=1282s`.

J Jeffrey Hanson. An introduction to the Hadoop Distributed File System. *IBM - developerWorks*, page 8, January 2011.

Michael Malak. Data Locality: HPC vs. Hadoop vs. Spark, September 2014. URL `/content/data-locality-hpc-vs-hadoop-vs-spark`.

Tom White. *Hadoop: the definitive guide*. O'Reilly, Beijing, third edition edition, 2012. ISBN 978-1-4493-1152-0.

Dhruba Borthakur. HDFS Architecture Guide, 2018. URL `https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html`.