Apache Hadoop

Apache Hadoop (/həˈduːp/) is a collection of open-source software utilities that facilitates using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model. Hadoop was originally designed for computer clusters built from commodity hardware, which is still the common use.[3] It has since also found use on clusters of higher-end hardware.[4][5] All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.[6]

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. This approach takes advantage of data locality,[7] where nodes manipulate the data they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.[8][9]

The base Apache Hadoop framework is composed of the following modules:

- *Hadoop Common* – contains libraries and utilities needed by other Hadoop modules;
- *Hadoop Distributed File System (HDFS)* – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- *Hadoop YARN* – (introduced in 2012) a platform responsible for managing computing resources in clusters and using them for scheduling users' applications;[10][11]
- *Hadoop MapReduce* – an implementation of the MapReduce programming model for large-scale data processing.
- *Hadoop Ozone* – (introduced in 2020) An object store for Hadoop

Apache Hadoop

Original author(s)	Doug Cutting, Mike Cafarella
Developer(s)	Apache Software Foundation
Initial release	April 1, 2006[1]
Stable release	**2.7.x** 2.7.7 / May 31, 2018[2] **2.8.x** 2.8.5 / September 15, 2018[2] **2.9.x** 2.9.2 / November 9, 2018[2] **2.10.x** 2.10.1 / September 21, 2020[2] **3.1.x** 3.1.4 / August 3, 2020[2] **3.2.x** 3.2.2 / January 9, 2021[2] **3.3.x** 3.3.1 / June 15, 2021[2]
Repository	Hadoop Repository (https://gitbox.apache.org/repos/asf?p=hadoop.git)
Written in	Java
Operating system	Cross-platform
Type	Distributed file system

License	Apache License 2.0
Website	hadoop.apache.org (https://hadoop.apache.org)

The term *Hadoop* is often used for both base modules and sub-modules and also the *ecosystem*,[12] or collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Phoenix, Apache Spark, Apache ZooKeeper, Cloudera Impala, Apache Flume, Apache Sqoop, Apache Oozie, and Apache Storm.[13]

Apache Hadoop's MapReduce and HDFS components were inspired by Google papers on MapReduce and Google File System.[14]

The Hadoop framework itself is mostly written in the Java programming language, with some native code in C and command line utilities written as shell scripts. Though MapReduce Java code is common, any programming language can be used with Hadoop Streaming to implement the map and reduce parts of the user's program.[15] Other projects in the Hadoop ecosystem expose richer user interfaces.

Contents

History

According to its co-founders, Doug Cutting and Mike Cafarella, the genesis of Hadoop was the Google File System paper that was published in October 2003.[16][17] This paper spawned another one from Google – "MapReduce: Simplified Data Processing on Large Clusters".[18] Development started on the Apache Nutch project, but was moved to the new Hadoop subproject in January 2006.[19] Doug Cutting, who was working at Yahoo! at the time, named it after his son's toy elephant.[20] The initial code that was factored out of Nutch consisted of about 5,000 lines of code for HDFS and about 6,000 lines of code for MapReduce.

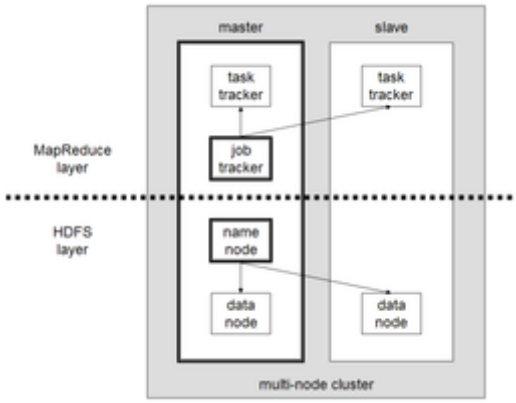
In March 2006, Owen O'Malley was the first committer to add to the Hadoop project;[21] Hadoop 0.1.0 was released in April 2006.[22] It continues to evolve through contributions that are being made to the project.[23] The very first design document for the Hadoop Distributed File System was written by Dhruba Borthakur in 2007.[24]

Architecture

Hadoop consists of the *Hadoop Common* package, which provides file system and operating system level abstractions, a MapReduce engine (either MapReduce/MR1 or YARN/MR2)[25] and the Hadoop Distributed File System (HDFS). The Hadoop Common package contains the Java Archive (JAR) files and scripts needed to start Hadoop.

For effective scheduling of work, every Hadoop-compatible file system should provide location awareness, which is the name of the rack, specifically the network switch where a worker node is. Hadoop applications can use this information to execute code on the node where the data is, and, failing that, on the same rack/switch to reduce backbone traffic. HDFS uses this method when replicating data for data redundancy across multiple racks. This approach reduces the impact of a rack power outage or switch failure; if any of these hardware failures occurs, the data will remain available.[26]

A small Hadoop cluster includes a single master and multiple worker nodes. The master node consists of a Job Tracker, Task Tracker, NameNode, and DataNode. A slave or *worker node* acts as both a DataNode and TaskTracker, though it is possible to have data-only and compute-only worker nodes. These are normally used only in nonstandard applications.[27]



A multi-node Hadoop cluster

Hadoop requires Java Runtime Environment (JRE) 1.6 or higher. The standard startup and shutdown scripts require that Secure Shell (SSH) be set up between nodes in the cluster.[28]

In a larger cluster, HDFS nodes are managed through a dedicated NameNode server to host the file system index, and a secondary NameNode that can generate snapshots of the namenode's memory structures, thereby preventing file-system corruption and loss of data. Similarly, a standalone JobTracker server can manage job scheduling across nodes. When Hadoop MapReduce is used with an alternate file system, the NameNode, secondary NameNode, and DataNode architecture of HDFS are replaced by the file-system-specific equivalents.

File systems

Hadoop distributed file system

The *Hadoop distributed file system* (HDFS) is a distributed, scalable, and portable file system written in Java for the Hadoop framework. Some consider it to instead be a data store due to its lack of POSIX compliance,[29] but it does provide shell commands and Java application programming interface (API) methods that are similar to other file systems.[30] A Hadoop instance is divided into HDFS and MapReduce. HDFS is used for storing the data and MapReduce is used for processing data. HDFS has five services as follows:

1. Name Node
2. Secondary Name Node
3. Job tracker
4. Data Node
5. Task Tracker

Top three are Master Services/Daemons/Nodes and bottom two are Slave Services. Master Services can communicate with each other and in the same way Slave services can communicate with each other. Name Node is a master node and Data node is its corresponding Slave node and can talk with each other.

Name Node: HDFS consists of only one Name Node that is called the Master Node. The master node can track files, manage the file system and has the metadata of all of the stored data within it. In particular, the name node contains the details of the number of blocks, locations of the data node that the data is stored in, where the replications are stored, and other details. The name node has direct contact with the client.

Data Node: A Data Node stores data in it as blocks. This is also known as the slave node and it stores the actual data into HDFS which is responsible for the client to read and write. These are slave daemons. Every Data node sends a Heartbeat message to the Name node every 3 seconds and conveys that it is alive. In this way when Name Node does not receive a heartbeat from a data node for 2 minutes, it will take that data node as dead and starts the process of block replications on some other Data node.

Secondary Name Node: This is only to take care of the checkpoints of the file system metadata which is in the Name Node. This is also known as the checkpoint Node. It is the helper Node for the Name Node. The secondary name node instructs the name node to create & send fsimage & editlog file, upon which the compacted fsimage file is created by the secondary name node.[31]

Job Tracker: Job Tracker receives the requests for Map Reduce execution from the client. Job tracker talks to the Name Node to know about the location of the data that will be used in processing. The Name Node responds with the metadata of the required processing data.

Task Tracker: It is the Slave Node for the Job Tracker and it will take the task from the Job Tracker. It also receives code from the Job Tracker. Task Tracker will take the code and apply on the file. The process of applying that code on the file is known as Mapper.[32]

Hadoop cluster has nominally a single namenode plus a cluster of datanodes, although redundancy options are available for the namenode due to its criticality. Each datanode serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses TCP/IP sockets for communication. Clients use remote procedure calls (RPC) to communicate with each other.

HDFS stores large files (typically in the range of gigabytes to terabytes[33]) across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence theoretically does not require redundant array of independent disks (RAID) storage on hosts (but to increase input-output (I/O) performance some RAID configurations are still useful). With the default replication value, 3, data is stored on three nodes: two on the same rack, and one on a different rack. Data nodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high. HDFS is not fully POSIX-

compliant, because the requirements for a POSIX file-system differ from the target goals of a Hadoop application. The trade-off of not having a fully POSIX-compliant file-system is increased performance for data throughput and support for non-POSIX operations such as Append.[34]

In May 2012, high-availability capabilities were added to HDFS,[35] letting the main metadata server called the NameNode manually fail-over onto a backup. The project has also started developing automatic fail-overs.

The HDFS file system includes a so-called *secondary namenode*, a misleading term that some might incorrectly interpret as a backup namenode when the primary namenode goes offline. In fact, the secondary namenode regularly connects with the primary namenode and builds snapshots of the primary namenode's directory information, which the system then saves to local or remote directories. These checkpointed images can be used to restart a failed primary namenode without having to replay the entire journal of file-system actions, then to edit the log to create an up-to-date directory structure. Because the namenode is the single point for storage and management of metadata, it can become a bottleneck for supporting a huge number of files, especially a large number of small files. HDFS Federation, a new addition, aims to tackle this problem to a certain extent by allowing multiple namespaces served by separate namenodes. Moreover, there are some issues in HDFS such as small file issues, scalability problems, Single Point of Failure (SPoF), and bottlenecks in huge metadata requests. One advantage of using HDFS is data awareness between the job tracker and task tracker. The job tracker schedules map or reduce jobs to task trackers with an awareness of the data location. For example: if node A contains data (a, b, c) and node X contains data (x, y, z), the job tracker schedules node A to perform map or reduce tasks on (a, b, c) and node X would be scheduled to perform map or reduce tasks on (x, y, z). This reduces the amount of traffic that goes over the network and prevents unnecessary data transfer. When Hadoop is used with other file systems, this advantage is not always available. This can have a significant impact on job-completion times as demonstrated with data-intensive jobs.[36]

HDFS was designed for mostly immutable files and may not be suitable for systems requiring concurrent write operations.[34]

HDFS can be mounted directly with a Filesystem in Userspace (FUSE) virtual file system on Linux and some other Unix systems.

File access can be achieved through the native Java API, the Thrift API (generates a client in a number of languages e.g. C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk, and OCaml), the command-line interface, the HDFS-UI web application over HTTP, or via 3rd-party network client libraries.[37]

HDFS is designed for portability across various hardware platforms and for compatibility with a variety of underlying operating systems. The HDFS design introduces portability limitations that result in some performance bottlenecks, since the Java implementation cannot use features that are exclusive to the platform on which HDFS is running.[38] Due to its widespread integration into enterprise-level infrastructure, monitoring HDFS performance at scale has become an increasingly important issue. Monitoring end-to-end performance requires tracking metrics from datanodes, namenodes, and the underlying operating system.[39] There are currently several monitoring platforms to track HDFS performance, including Hortonworks, Cloudera, and Datadog.

Other file systems

Hadoop works directly with any distributed file system that can be mounted by the underlying operating system by simply using a `file://` URL; however, this comes at a price – the loss of locality. To reduce network traffic, Hadoop needs to know which servers are closest to the data, information that Hadoop-

specific file system bridges can provide.

In May 2011, the list of supported file systems bundled with Apache Hadoop were:

- HDFS: Hadoop's own rack-aware file system.[40] This is designed to scale to tens of petabytes of storage and runs on top of the file systems of the underlying operating systems.
- Apache Hadoop Ozone: HDFS-compatible object store targeting optimized for billions of small files.
- FTP file system: This stores all its data on remotely accessible FTP servers.
- Amazon S3 (Simple Storage Service) object storage: This is targeted at clusters hosted on the Amazon Elastic Compute Cloud server-on-demand infrastructure. There is no rack-awareness in this file system, as it is all remote.
- Windows Azure Storage Blobs (WASB) file system: This is an extension of HDFS that allows distributions of Hadoop to access data in Azure blob stores without moving the data permanently into the cluster.

A number of third-party file system bridges have also been written, none of which are currently in Hadoop distributions. However, some commercial distributions of Hadoop ship with an alternative file system as the default – specifically IBM and MapR.

- In 2009, IBM discussed running Hadoop over the IBM General Parallel File System.[41] The source code was published in October 2009.[42]
- In April 2010, Parascale published the source code to run Hadoop against the Parascale file system.[43]
- In April 2010, Appistry released a Hadoop file system driver for use with its own CloudIQ Storage product.[44]
- In June 2010, HP discussed a location-aware IBRIX Fusion file system driver.[45]
- In May 2011, MapR Technologies Inc. announced the availability of an alternative file system for Hadoop, MapR FS, which replaced the HDFS file system with a full random-access read/write file system.

JobTracker and TaskTracker: the MapReduce engine

Atop the file systems comes the MapReduce Engine, which consists of one *JobTracker*, to which client applications submit MapReduce jobs. The JobTracker pushes work to available *TaskTracker* nodes in the cluster, striving to keep the work as close to the data as possible. With 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 in 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. The TaskTracker on each node spawns a separate Java virtual machine (JVM) process to prevent the TaskTracker itself from failing if the running job crashes its JVM. A heartbeat is sent from the TaskTracker to the JobTracker every few minutes to check its status. The Job Tracker and TaskTracker status and information is exposed by Jetty and can be viewed from a web browser.

Known limitations of this approach are:

1. The allocation of work to TaskTrackers is very simple. Every TaskTracker has a number of available *slots* (such as "4 slots"). Every active map or reduce task takes up one slot. The Job Tracker allocates work to the tracker nearest to the data with an available slot. There is no consideration of the current system load of the allocated machine, and hence its actual availability.

2. If one TaskTracker is very slow, it can delay the entire MapReduce job – especially towards the end, when everything can end up waiting for the slowest task. With speculative execution enabled, however, a single task can be executed on multiple slave nodes.

Scheduling

By default Hadoop uses FIFO scheduling, and optionally 5 scheduling priorities to schedule jobs from a work queue.[46] In version 0.19 the job scheduler was refactored out of the JobTracker, while adding the ability to use an alternate scheduler (such as the *Fair scheduler* or the *Capacity scheduler*, described next).[47]

Fair scheduler

The fair scheduler was developed by Facebook.[48] The goal of the fair scheduler is to provide fast response times for small jobs and Quality of service (QoS) for production jobs. The fair scheduler has three basic concepts.[49]

1. Jobs are grouped into pools.
2. Each pool is assigned a guaranteed minimum share.
3. Excess capacity is split between jobs.

By default, jobs that are uncategorized go into a default pool. Pools have to specify the minimum number of map slots, reduce slots, as well as a limit on the number of running jobs.

Capacity scheduler

The capacity scheduler was developed by Yahoo. The capacity scheduler supports several features that are similar to those of the fair scheduler.[50]

1. Queues are allocated a fraction of the total resource capacity.
2. Free resources are allocated to queues beyond their total capacity.
3. Within a queue, a job with a high level of priority has access to the queue's resources.

There is no preemption once a job is running.

Difference between Hadoop 1 and Hadoop 2 (YARN)

The biggest difference between Hadoop 1 and Hadoop 2 is the addition of YARN (Yet Another Resource Negotiator), which replaced the MapReduce engine in the first version of Hadoop. YARN strives to allocate resources to various applications effectively. It runs two dæmons, which take care of two different tasks: the *resource manager*, which does job tracking and resource allocation to applications, the *application master*, which monitors progress of the execution.

Difference between Hadoop 2 and Hadoop 3

There are important features provided by Hadoop 3. For example, while there is one single *namenode* in Hadoop 2, Hadoop 3 enables having multiple name nodes, which solves the single point of failure problem.

In Hadoop 3, there are containers working in principle of Docker, which reduces time spent on application development.

One of the biggest changes is that Hadoop 3 decreases storage overhead with erasure coding.

Also, Hadoop 3 permits usage of GPU hardware within the cluster, which is a very substantial benefit to execute deep learning algorithms on a Hadoop cluster.[51]

Other applications

The HDFS is not restricted to MapReduce jobs. It can be used for other applications, many of which are under development at Apache. The list includes the HBase database, the Apache Mahout machine learning system, and the Apache Hive data warehouse. Theoretically, Hadoop could be used for any workload that is batch-oriented rather than real-time, is very data-intensive, and benefits from parallel processing. It can also be used to complement a real-time system, such as lambda architecture, Apache Storm, Flink, and Spark Streaming.[52]

Commercial applications of Hadoop include:[53]

- Log or clickstream analysis
- Marketing analytics
- Machine learning and data mining
- Image processing
- XML message processing
- Web crawling
- Archival work for compliance, including of relational and tabular data

Prominent use cases

On 19 February 2008, Yahoo! Inc. launched what they claimed was the world's largest Hadoop production application. The Yahoo! Search Webmap is a Hadoop application that runs on a Linux cluster with more than 10,000 cores and produced data that was used in every Yahoo! web search query.[54] There are multiple Hadoop clusters at Yahoo! and no HDFS file systems or MapReduce jobs are split across multiple data centers. Every Hadoop cluster node bootstraps the Linux image, including the Hadoop distribution. Work that the clusters perform is known to include the index calculations for the Yahoo! search engine. In June 2009, Yahoo! made the source code of its Hadoop version available to the open-source community.[55]

In 2010, Facebook claimed that they had the largest Hadoop cluster in the world with 21 PB of storage.[56] In June 2012, they announced the data had grown to 100 PB[57] and later that year they announced that the data was growing by roughly half a PB per day.[58]

As of 2013, Hadoop adoption had become widespread: more than half of the Fortune 50 companies used Hadoop.[59]

Hadoop hosting in the cloud

Hadoop can be deployed in a traditional onsite datacenter as well as in the cloud.[60] The cloud allows organizations to deploy Hadoop without the need to acquire hardware or specific setup expertise.[61]

Commercial support

A number of companies offer commercial implementations or support for Hadoop.[62]

Branding

The Apache Software Foundation has stated that only software officially released by the Apache Hadoop Project can be called *Apache Hadoop* or *Distributions of Apache Hadoop*.[63] The naming of products and derivative works from other vendors and the term "compatible" are somewhat controversial within the Hadoop developer community.[64]

Papers

Some papers influenced the birth and growth of Hadoop and big data processing. Some of these are:

- Jeffrey Dean, Sanjay Ghemawat (2004) MapReduce: Simplified Data Processing on Large Clusters (https://www.usenix.org/legacy/publications/library/proceedings/osdi04/tech/full_papers/dean/dean_html/index.html), Google. This paper inspired Doug Cutting to develop an open-source implementation of the Map-Reduce framework. He named it Hadoop, after his son's toy elephant.
- Michael Franklin, Alon Halevy, David Maier (2005) From Databases to Dataspaces: A New Abstraction for Information Management (http://www.eecs.berkeley.edu/~franklin/Papers/dataspaceSR.pdf). The authors highlight the need for storage systems to accept all data formats and to provide APIs for data access that evolve based on the storage system's understanding of the data.
- Fay Chang et al. (2006) Bigtable: A Distributed Storage System for Structured Data (http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/archive/bigtable-osdi06.pdf), Google.
- Robert Kallman et al. (2008) H-store: a high-performance, distributed main memory transaction processing system (http://www.vldb.org/pvldb/vol1/1454211.pdf)

See also

- Apache Accumulo – Secure Bigtable[65]
- Apache Cassandra, a column-oriented database that supports access from Hadoop
- Apache CouchDB, a database that uses JSON for documents, JavaScript for MapReduce queries, and regular HTTP for an API
- Apache HCatalog, a table and storage management layer for Hadoop
- Big data
- Data Intensive Computing
- HPCC – LexisNexis Risk Solutions High Performance Computing Cluster
- Hypertable – HBase alternative
- Sector/Sphere – Open source distributed storage and processing
- Simple Linux Utility for Resource Management

References

1. "Hadoop Releases" (https://archive.apache.org/dist/hadoop/common/). *apache.org*. Apache

Software Foundation. Retrieved 28 April 2019.
2. "Apache Hadoop" (https://hadoop.apache.org/releases.html). Retrieved 7 September 2019.
3. Judge, Peter (22 October 2012). "Doug Cutting: Big Data Is No Bubble" (http://www.silicon.co.uk/workspace/doug-cutting-big-data-is-not-a-bubble-96694). *silicon.co.uk*. Retrieved 11 March 2018.
4. Woodie, Alex (12 May 2014). "Why Hadoop on IBM Power" (https://www.datanami.com/2014/05/12/hadoop-ibm-power/). *datanami.com*. Datanami. Retrieved 11 March 2018.
5. Hemsoth, Nicole (15 October 2014). "Cray Launches Hadoop into HPC Airspace" (https://www.hpcwire.com/2014/10/15/cray-launches-hadoop-hpc-airspace/). *hpcwire.com*. Retrieved 11 March 2018.
6. "Welcome to Apache Hadoop!" (http://hadoop.apache.org). *hadoop.apache.org*. Retrieved 25 August 2016.
7. "What is the Hadoop Distributed File System (HDFS)?" (https://www.ibm.com/analytics/hadoop/hdfs). *ibm.com*. IBM. Retrieved 12 April 2021.
8. Malak, Michael (19 September 2014). "Data Locality: HPC vs. Hadoop vs. Spark" (http://www.datascienceassn.org/content/data-locality-hpc-vs-hadoop-vs-spark). *datascienceassn.org*. Data Science Association. Retrieved 30 October 2014.
9. Wang, Yandong; Goldstone, Robin; Yu, Weikuan; Wang, Teng (October 2014). "Characterization and Optimization of Memory-Resident MapReduce on HPC Systems". *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE. pp. 799–808. doi:10.1109/IPDPS.2014.87 (https://doi.org/10.1109%2FIPDPS.2014.87). ISBN 978-1-4799-3800-1. S2CID 11157612 (https://api.semanticscholar.org/CorpusID:11157612).
10. "Resource (Apache Hadoop Main 2.5.1 API)" (https://web.archive.org/web/20141006090717/http://hadoop.apache.org/docs/r2.5.1/api/org/apache/hadoop/yarn/api/records/Resource.html#newInstance(int,%20int)). *apache.org*. Apache Software Foundation. 12 September 2014. Archived from the original (http://hadoop.apache.org/docs/r2.5.1/api/org/apache/hadoop/yarn/api/records/Resource.html#newInstance(int,%20int)) on 6 October 2014. Retrieved 30 September 2014.
11. Murthy, Arun (15 August 2012). "Apache Hadoop YARN – Concepts and Applications" (http://hortonworks.com/blog/apache-hadoop-yarn-concepts-and-applications/). *hortonworks.com*. Hortonworks. Retrieved 30 September 2014.
12. "Continuuity Raises $10 Million Series A Round to Ignite Big Data Application Development Within the Hadoop Ecosystem" (https://finance.yahoo.com/news/continuuity-raises-10-million-series-120500471.html). *finance.yahoo.com*. Marketwired. 14 November 2012. Retrieved 30 October 2014.
13. "Hadoop-related projects at" (http://hadoop.apache.org/). Hadoop.apache.org. Retrieved 17 October 2013.
14. *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data* (https://books.google.com/books?id=axruBQAAQBAJ&pg=PA300). John Wiley & Sons. 19 December 2014. p. 300. ISBN 9781118876220. Retrieved 29 January 2015.
15. "[nlpatumd] Adventures with Hadoop and Perl" (http://www.mail-archive.com/nlpatumd@yahoogroups.com/msg00570.html). Mail-archive.com. 2 May 2010. Retrieved 5 April 2013.
16. Cutting, Mike; Cafarella, Ben; Lorica, Doug (31 March 2016). "The next 10 years of Apache Hadoop" (https://www.oreilly.com/ideas/the-next-10-years-of-apache-hadoop). *O'Reilly Media*. Retrieved 12 October 2017.
17. Ghemawat, Sanjay; Gobioff, Howard; Leung, Shun-Tak (2003). "The Google File System" (http://research.google.com/archive/gfs.html). pp. 20–43.

18. Dean, Jeffrey; Ghemawat, Sanjay (2004). "MapReduce: Simplified Data Processing on Large Clusters" (http://research.google.com/archive/mapreduce.html). pp. 137–150.
19. Cutting, Doug (28 January 2006). "new mailing lists request: hadoop" (https://issues.apache.org/jira/browse/INFRA-700). *issues.apache.org*. "The Lucene PMC has voted to split part of Nutch into a new sub-project named Hadoop"
20. Vance, Ashlee (17 March 2009). "Hadoop, a Free Software Program, Finds Uses Beyond Search" (https://www.nytimes.com/2009/03/17/technology/business-computing/17cloud.html). *The New York Times*. Archived (https://web.archive.org/web/20110830130350/http://www.nytimes.com/2009/03/17/technology/business-computing/17cloud.html) from the original on 30 August 2011. Retrieved 20 January 2010.
21. Cutting, Doug (30 March 2006). "[RESULT] VOTE: add Owen O'Malley as Hadoop committer" (http://mail-archives.apache.org/mod_mbox/hadoop-common-dev/200603.mbox/%3C442B27A6.8080500@apache.org%3E). *hadoop-common-dev* (Mailing list).
22. "Index of /dist/hadoop/core" (https://archive.apache.org/dist/hadoop/core/). *archive.apache.org*. Retrieved 11 December 2017.
23. "Who We Are" (https://hadoop.apache.org/who.html). *hadoop.apache.org*. Retrieved 11 December 2017.
24. Borthakur, Dhruba (2006). "The Hadoop Distributed File System: Architecture and Design" (http://svn.apache.org/repos/asf/hadoop/common/tags/release-0.10.0/docs/hdfs_design.pdf) (PDF). *Apache Hadoop Code Repository*.
25. Chouraria, Harsh (21 October 2012). "MR2 and YARN Briefly Explained" (https://web.archive.org/web/20131022080058/http://blog.cloudera.com/blog/2012/10/mr2-and-yarn-briefly-explained/). *Cloudera.com*. Archived from the original (https://blog.cloudera.com/blog/2012/10/mr2-and-yarn-briefly-explained/) on 22 October 2013. Retrieved 23 October 2013.
26. "HDFS User Guide" (http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html). Hadoop.apache.org. Retrieved 4 September 2014.
27. "Running Hadoop on Ubuntu Linux System(Multi-Node Cluster)" (http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/).
28. "Running Hadoop on Ubuntu Linux (Single-Node Cluster)" (http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/#prerequisites). Retrieved 6 June 2013.
29. Evans, Chris (October 2013). "Big data storage: Hadoop storage basics" (http://www.computerweekly.com/feature/Big-data-storage-Hadoop-storage-basics). *computerweekly.com*. Computer Weekly. Retrieved 21 June 2016. "HDFS is not a file system in the traditional sense and isn't usually directly mounted for a user to view"
30. deRoos, Dirk. "Managing Files with the Hadoop File System Commands" (http://www.dummies.com/how-to/content/managing-files-with-the-hadoop-file-system-command.html). *dummies.com*. For Dummies. Retrieved 21 June 2016.
31. Balram. "Big Data Hadoop Tutorial for Beginners" (https://www.gyansetu.in/big-data-hadoop-tutorial-for-beginners). *www.gyansetu.in*. Retrieved 11 March 2021.
32. "Archived copy" (https://web.archive.org/web/20191023001222/http://hadoop.apache.org/docs/r2.7.5/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html). Archived from the original (https://hadoop.apache.org/DOCS/R2.7.5/HADOOP-PROJECT-DIST/HADOOP-HDFS/HDFSUSERGUIDE.HTML) on 23 October 2019. Retrieved 19 June 2020.
33. "HDFS Architecture" (http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Large_Data_Sets). Retrieved 1 September 2013.
34. Pessach, Yaniv (2013). "Distributed Storage" (Distributed Storage: Concepts, Algorithms, and Implementations ed.). OL 25423189M (https://openlibrary.org/books/OL25423189M).

35. "Version 2.0 provides for manual failover and they are working on automatic failover" (https://hadoop.apache.org/releases.html#23+May%2C+2012%3A+Release+2.0.0-alpha+available). Hadoop.apache.org. Retrieved 30 July 2013.
36. "Improving MapReduce performance through data placement in heterogeneous Hadoop Clusters" (http://www.eng.auburn.edu/~xqin/pubs/hcw10.pdf) (PDF). Eng.auburn.ed. April 2010.
37. "Mounting HDFS" (https://wiki.apache.org/hadoop/MountableHDFS). Retrieved 5 August 2016.
38. Shafer, Jeffrey; Rixner, Scott; Cox, Alan. "The Hadoop Distributed Filesystem: Balancing Portability and Performance" (http://www.jeffshafer.com/publications/papers/shafer_ispass10.pdf) (PDF). Rice University. Retrieved 19 September 2016.
39. Mouzakitis, Evan (21 July 2016). "How to Collect Hadoop Performance Metrics" (https://www.datadoghq.com/blog/monitor-hadoop-metrics/#toc-hdfs-metrics2). Retrieved 24 October 2016.
40. "HDFS Users Guide – Rack Awareness" (http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html#Rack_Awareness). Hadoop.apache.org. Retrieved 17 October 2013.
41. "Cloud analytics: Do we really need to reinvent the storage stack?" (http://www.usenix.org/events/hotcloud09/tech/full_papers/ananthanarayanan.pdf) (PDF). IBM. June 2009.
42. "HADOOP-6330: Integrating IBM General Parallel File System implementation of Hadoop Filesystem interface" (https://issues.apache.org/jira/browse/HADOOP-6330). IBM. 23 October 2009.
43. "HADOOP-6704: add support for Parascale filesystem" (https://issues.apache.org/jira/browse/HADOOP-6704). Parascale. 14 April 2010.
44. "HDFS with CloudIQ Storage" (https://web.archive.org/web/20140405044536/http://resources.appistry.com/news-and-events/press/06072010-appistry-cloudiq-storage-now-generally-available). Appistry,Inc. 6 July 2010. Archived from the original (http://resources.appistry.com/news-and-events/press/06072010-appistry-cloudiq-storage-now-generally-available) on 5 April 2014. Retrieved 10 December 2013.
45. "High Availability Hadoop" (http://www.slideshare.net/steve_l/high-availability-hadoop). HP. 9 June 2010.
46. "Commands Guide" (https://web.archive.org/web/20110817053520/http://hadoop.apache.org/common/docs/current/commands_manual.html#job). 17 August 2011. Archived from the original on 17 August 2011. Retrieved 11 December 2017.
47. "Refactor the scheduler out of the JobTracker" (https://issues.apache.org/jira/browse/HADOOP-3412). *Hadoop Common*. Apache Software Foundation. Retrieved 9 June 2012.
48. Jones, M. Tim (6 December 2011). "Scheduling in Hadoop" (http://www.ibm.com/developerworks/library/os-hadoop-scheduling/). *ibm.com*. IBM. Retrieved 20 November 2013.
49. "Hadoop Fair Scheduler Design Document" (https://svn.apache.org/repos/asf/hadoop/common/branches/MAPREDUCE-233/src/contrib/fairscheduler/designdoc/fair_scheduler_design_doc.pdf) (PDF). *apache.org*. Retrieved 12 October 2017.
50. "CapacityScheduler Guide" (http://hadoop.apache.org/docs/stable1/capacity_scheduler.html). *Hadoop.apache.org*. Retrieved 31 December 2015.
51. "How Apache Hadoop 3 Adds Value Over Apache Hadoop 2" (https://it.hortonworks.com/blog/hadoop-3-adds-value-hadoop-2/). *hortonworks.com*. 7 February 2018. Retrieved 11 June 2018.

52. Chintapalli, Sanket; Dagit, Derek; Evans, Bobby; Farivar, Reza; Graves, Thomas; Holderbaugh, Mark; Liu, Zhuo; Nusbaum, Kyle; Patil, Kishorkumar; Peng, Boyang Jerry; Poulosky, Paul (May 2016). "Benchmarking Streaming Computation Engines: Storm, Flink and Spark Streaming". *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE. pp. 1789–1792. doi:10.1109/IPDPSW.2016.138 (https://doi.org/10.1109%2FIPDPSW.2016.138). ISBN 978-1-5090-3682-0. S2CID 2180634 (https://api.semanticscholar.org/CorpusID:2180634).
53. " "How 30+ enterprises are using Hadoop", in DBMS2" (http://www.dbms2.com/2009/10/10/enterprises-using-hadoo/). Dbms2.com. 10 October 2009. Retrieved 17 October 2013.
54. "Yahoo! Launches World's Largest Hadoop Production Application" (https://web.archive.org/web/20160307081144/https://developer.yahoo.com/blogs/hadoop/yahoo-launches-world-largest-hadoop-production-application-398.html). *Yahoo*. 19 February 2008. Archived from the original (https://developer.yahoo.com/blogs/hadoop/yahoo-launches-world-largest-hadoop-production-application-398.html) on 7 March 2016. Retrieved 31 December 2015.
55. "Hadoop and Distributed Computing at Yahoo!" (http://developer.yahoo.com/hadoop/). Yahoo!. 20 April 2011. Retrieved 17 October 2013.
56. "HDFS: Facebook has the world's largest Hadoop cluster!" (http://hadoopblog.blogspot.com/2010/05/facebook-has-worlds-largest-hadoop.html). Hadoopblog.blogspot.com. 9 May 2010. Retrieved 23 May 2012.
57. "Under the Hood: Hadoop Distributed File system reliability with Namenode and Avatarnode" (https://www.facebook.com/notes/facebook-engineering/under-the-hood-hadoop-distributed-filesystem-reliability-with-namenode-and-avata/10150888759153920). Facebook. Retrieved 13 September 2012.
58. "Under the Hood: Scheduling MapReduce jobs more efficiently with Corona" (https://www.facebook.com/notes/facebook-engineering/under-the-hood-scheduling-mapreduce-jobs-more-efficiently-with-corona/10151142560538920). Facebook. Retrieved 9 November 2012.
59. "Altior's AltraSTAR – Hadoop Storage Accelerator and Optimizer Now Certified on CDH4 (Cloudera's Distribution Including Apache Hadoop Version 4)" (http://www.prnewswire.com/news-releases/altiors-altrastar---hadoop-storage-accelerator-and-optimizer-now-certified-on-cdh4-clouderas-distribution-including-apache-hadoop-version-4-183906141.html) (Press release). Eatontown, NJ: Altior Inc. 18 December 2012. Retrieved 30 October 2013.
60. "Hadoop - Microsoft Azure" (http://azure.microsoft.com/en-us/solutions/hadoop/). *azure.microsoft.com*. Retrieved 11 December 2017.
61. "Hadoop" (http://azure.microsoft.com/en-us/solutions/hadoop/). Azure.microsoft.com. Retrieved 22 July 2014.
62. "Why the Pace of Hadoop Innovation Has to Pick Up" (http://gigaom.com/cloud/why-we-need-more-hadoop-innovation/). Gigaom.com. 25 April 2011. Retrieved 17 October 2013.
63. "Defining Hadoop" (http://wiki.apache.org/hadoop/Defining%20Hadoop). Wiki.apache.org. 30 March 2013. Retrieved 17 October 2013.
64. "Defining Hadoop Compatibility: revisited" (http://mail-archives.apache.org/mod_mbox/hadoop-general/201105.mbox/%3C4DC91392.2010308@apache.org%3E). Mail-archives.apache.org. 10 May 2011. Retrieved 17 October 2013.
65. "Apache Accumulo User Manual: Security" (https://accumulo.apache.org/1.4/user_manual/Security.html). *apache.org*. Apache Software Foundation. Retrieved 3 December 2014.

Bibliography

- Lam, Chuck (28 July 2010). *Hadoop in Action* (https://books.google.com/books?id=iGq3PwAACAAJ) (1st ed.). Manning Publications. p. 325. ISBN 978-1-935-18219-1.

- Venner, Jason (22 June 2009). *Pro Hadoop* (https://web.archive.org/web/20101205204120/http://apress.com/book/view/1430219424) (1st ed.). Apress. p. 440. ISBN 978-1-430-21942-2. Archived from the original (http://www.apress.com/book/view/1430219424) on 5 December 2010. Retrieved 3 July 2009.
- White, Tom (16 June 2009). *Hadoop: The Definitive Guide* (http://oreilly.com/catalog/9780596521974) (1st ed.). O'Reilly Media. p. 524. ISBN 978-0-596-52197-4.
- Vohra, Deepak (October 2016). *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools* (https://www.apress.com/us/book/9781484221983) (1st ed.). Apress. p. 429. ISBN 978-1-4842-2199-0.
- Wiktorski, Tomasz (January 2019). *Data-intensive Systems* (https://www.springer.com/it/book/9783030046026). Cham, Switzerland: Springer. ISBN 978-3-030-04603-3.

External links

- Official website (https://hadoop.apache.org)