**Apache Hadoop** ( [/həˈduːp/](https://en.wikipedia.org/wiki/Help:IPA/English)) is a collection of [open-source](https://en.wikipedia.org/wiki/Open-source_software) 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](https://en.wikipedia.org/wiki/Software_framework) for [distributed storage](https://en.wikipedia.org/wiki/Clustered_file_system) and processing of [big data](https://en.wikipedia.org/wiki/Big_data) using the [MapReduce](https://en.wikipedia.org/wiki/MapReduce) [programming model](https://en.wikipedia.org/wiki/Programming_model). Hadoop was originally designed for [computer clusters](https://en.wikipedia.org/wiki/Computer_cluster) built from [commodity hardware](https://en.wikipedia.org/wiki/Commodity_hardware), which is still the common use.[[3]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-3) It has since also found use on clusters of higher-end hardware.[[4]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-4)[[5]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-homepage-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](https://en.wikipedia.org/wiki/JAR_(file_format)) into nodes to process the data in parallel. This approach takes advantage of [data locality](https://en.wikipedia.org/wiki/Data_locality),[[7]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-7) where nodes manipulate the data they have access to. This allows the dataset to be [processed](https://en.wikipedia.org/wiki/Distributed_processing) faster and more efficiently than it would be in a more conventional [supercomputer architecture](https://en.wikipedia.org/wiki/Supercomputer_architecture) that relies on a [parallel file system](https://en.wikipedia.org/wiki/Parallel_file_system) where computation and data are distributed via high-speed networking.[[8]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-8)[[9]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-10)[[11]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-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

The term *Hadoop* is often used for both base modules and sub-modules and also the *ecosystem*,[[12]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-12) or collection of additional software packages that can be installed on top of or alongside Hadoop, such as [Apache Pig](https://en.wikipedia.org/wiki/Pig_(programming_tool)), [Apache Hive](https://en.wikipedia.org/wiki/Apache_Hive), [Apache HBase](https://en.wikipedia.org/wiki/Apache_HBase), [Apache Phoenix](https://en.wikipedia.org/wiki/Apache_Phoenix), [Apache Spark](https://en.wikipedia.org/wiki/Apache_Spark), [Apache ZooKeeper](https://en.wikipedia.org/wiki/Apache_ZooKeeper), [Cloudera Impala](https://en.wikipedia.org/wiki/Cloudera_Impala), [Apache Flume](https://en.wikipedia.org/wiki/Apache_Flume), [Apache Sqoop](https://en.wikipedia.org/wiki/Apache_Sqoop), [Apache Oozie](https://en.wikipedia.org/wiki/Apache_Oozie), and [Apache Storm](https://en.wikipedia.org/wiki/Apache_Storm).[[13]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-13)

Apache Hadoop's MapReduce and HDFS components were inspired by [Google](https://en.wikipedia.org/wiki/Google) papers on [MapReduce](https://en.wikipedia.org/wiki/MapReduce) and [Google File System](https://en.wikipedia.org/wiki/Google_File_System).[[14]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-14)

The Hadoop framework itself is mostly written in the [Java programming language](https://en.wikipedia.org/wiki/Java_(programming_language)), with some native code in [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [command line](https://en.wikipedia.org/wiki/Command-line_interface) utilities written as [shell scripts](https://en.wikipedia.org/wiki/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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-15) Other projects in the Hadoop ecosystem expose richer user interfaces.



**Contents**

* [1History](https://en.wikipedia.org/wiki/Apache_Hadoop#History)
* [2Architecture](https://en.wikipedia.org/wiki/Apache_Hadoop#Architecture)
  + [2.1File systems](https://en.wikipedia.org/wiki/Apache_Hadoop#File_systems)
    - [2.1.1Hadoop distributed file system](https://en.wikipedia.org/wiki/Apache_Hadoop#Hadoop_distributed_file_system)
    - [2.1.2Other file systems](https://en.wikipedia.org/wiki/Apache_Hadoop#Other_file_systems)
  + [2.2JobTracker and TaskTracker: the MapReduce engine](https://en.wikipedia.org/wiki/Apache_Hadoop#JobTracker_and_TaskTracker:_the_MapReduce_engine)
    - [2.2.1Scheduling](https://en.wikipedia.org/wiki/Apache_Hadoop#Scheduling)
      * [2.2.1.1Fair scheduler](https://en.wikipedia.org/wiki/Apache_Hadoop#Fair_scheduler)
      * [2.2.1.2Capacity scheduler](https://en.wikipedia.org/wiki/Apache_Hadoop#Capacity_scheduler)
  + [2.3Difference between Hadoop 1 and Hadoop 2 (YARN)](https://en.wikipedia.org/wiki/Apache_Hadoop#Difference_between_Hadoop_1_and_Hadoop_2_(YARN))
  + [2.4Difference between Hadoop 2 and Hadoop 3](https://en.wikipedia.org/wiki/Apache_Hadoop#Difference_between_Hadoop_2_and_Hadoop_3)
  + [2.5Other applications](https://en.wikipedia.org/wiki/Apache_Hadoop#Other_applications)
* [3Prominent use cases](https://en.wikipedia.org/wiki/Apache_Hadoop#Prominent_use_cases)
* [4Hadoop hosting in the cloud](https://en.wikipedia.org/wiki/Apache_Hadoop#Hadoop_hosting_in_the_cloud)
* [5Commercial support](https://en.wikipedia.org/wiki/Apache_Hadoop#Commercial_support)
  + [5.1Branding](https://en.wikipedia.org/wiki/Apache_Hadoop#Branding)
* [6Papers](https://en.wikipedia.org/wiki/Apache_Hadoop#Papers)
* [7See also](https://en.wikipedia.org/wiki/Apache_Hadoop#See_also)
* [8References](https://en.wikipedia.org/wiki/Apache_Hadoop#References)
* [9Bibliography](https://en.wikipedia.org/wiki/Apache_Hadoop#Bibliography)
* [10External links](https://en.wikipedia.org/wiki/Apache_Hadoop#External_links)

History[[edit](https://en.wikipedia.org/w/index.php?title=Apache_Hadoop&action=edit&section=1)]

According to its co-founders, [Doug Cutting](https://en.wikipedia.org/wiki/Doug_Cutting) and [Mike Cafarella](https://en.wikipedia.org/wiki/Mike_Cafarella), the genesis of Hadoop was the Google File System paper that was published in October 2003.[[16]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-16)[[17]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-17) This paper spawned another one from Google – "MapReduce: Simplified Data Processing on Large Clusters".[[18]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-18) Development started on the [Apache Nutch](https://en.wikipedia.org/wiki/Apache_Nutch) project, but was moved to the new Hadoop subproject in January 2006.[[19]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-19) Doug Cutting, who was working at [Yahoo!](https://en.wikipedia.org/wiki/Yahoo!) at the time, named it after his son's toy elephant.[[20]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-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.

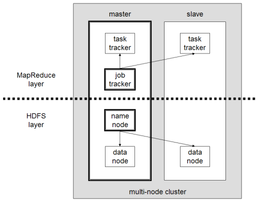
In March 2006, Owen O’Malley was the first committer to add to the Hadoop project;[[21]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-21) Hadoop 0.1.0 was released in April 2006.[[22]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-22) It continues to evolve through contributions that are being made to the project.[[23]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-23) The very first design document for the Hadoop Distributed File System was written by Dhruba Borthakur in 2007.[[24]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-24)

Architecture[[edit](https://en.wikipedia.org/w/index.php?title=Apache_Hadoop&action=edit&section=2)]

*See also:*[*Hadoop Distributed File System*](https://en.wikipedia.org/wiki/Apache_Hadoop#Hadoop_distributed_file_system)*,*[*Apache HBase*](https://en.wikipedia.org/wiki/Apache_HBase)*, and*[*MapReduce*](https://en.wikipedia.org/wiki/MapReduce)

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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-25) and the [Hadoop Distributed File System](https://en.wikipedia.org/wiki/Apache_Hadoop#Hadoop_distributed_file_system) (HDFS). The Hadoop Common package contains the [Java Archive (JAR)](https://en.wikipedia.org/wiki/JAR_(file_format)) 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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-26)

[](https://en.wikipedia.org/wiki/File:Hadoop_1.png)

A multi-node Hadoop cluster

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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-michael-noll.com_2-27)

Hadoop requires [Java Runtime Environment](https://en.wikipedia.org/wiki/JRE) (JRE) 1.6 or higher. The standard startup and shutdown scripts require that [Secure Shell](https://en.wikipedia.org/wiki/Secure_Shell) (SSH) be set up between nodes in the cluster.[[28]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-michael-noll.com_1-28)

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**[[edit](https://en.wikipedia.org/w/index.php?title=Apache_Hadoop&action=edit&section=3)]

**Hadoop distributed file system**[[edit](https://en.wikipedia.org/w/index.php?title=Apache_Hadoop&action=edit&section=4)]

The *Hadoop distributed file system* (HDFS) is a distributed, scalable, and portable [file system](https://en.wikipedia.org/wiki/Distributed_file_system) written in Java for the Hadoop framework. Some consider it to instead be a [data store](https://en.wikipedia.org/wiki/Distributed_data_store) due to its lack of [POSIX](https://en.wikipedia.org/wiki/POSIX) compliance,[[29]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-29) but it does provide shell commands and Java application programming interface (API) [methods](https://en.wikipedia.org/wiki/Method_(computer_programming)) that are similar to other file systems.[[30]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-32)

Hadoop cluster has nominally a single namenode plus a cluster of datanodes, although [redundancy](https://en.wikipedia.org/wiki/Redundancy_(engineering)) 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](https://en.wikipedia.org/wiki/TCP/IP) [sockets](https://en.wikipedia.org/wiki/Internet_socket) for communication. Clients use [remote procedure calls](https://en.wikipedia.org/wiki/Remote_procedure_call) (RPC) to communicate with each other.

HDFS stores large files (typically in the range of gigabytes to terabytes[[33]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-33)) across multiple machines. It achieves reliability by [replicating](https://en.wikipedia.org/wiki/Replication_(computer_science)) the data across multiple hosts, and hence theoretically does not require [redundant array of independent disks (RAID)](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Throughput) and support for non-POSIX operations such as Append.[[34]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-openlibrary1-34)

In May 2012, high-availability capabilities were added to HDFS,[[35]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-failover-35) letting the main metadata server called the NameNode manually fail-over onto a backup. The project has also started developing automatic [fail-overs](https://en.wikipedia.org/wiki/Fail-over).

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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-36)

HDFS was designed for mostly immutable files and may not be suitable for systems requiring concurrent write operations.[[34]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-openlibrary1-34)

HDFS can be [mounted](https://en.wikipedia.org/wiki/Mount_(computing)) directly with a [Filesystem in Userspace](https://en.wikipedia.org/wiki/Filesystem_in_Userspace) (FUSE) [virtual file system](https://en.wikipedia.org/wiki/Virtual_file_system) on [Linux](https://en.wikipedia.org/wiki/Linux) and some other [Unix](https://en.wikipedia.org/wiki/Unix) systems.

File access can be achieved through the native Java API, the [Thrift](https://en.wikipedia.org/wiki/Thrift_(protocol)) API (generates a client in a number of languages e.g. C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, [Cocoa](https://en.wikipedia.org/wiki/Cocoa_(API)), Smalltalk, and [OCaml](https://en.wikipedia.org/wiki/OCaml)), the [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface), the HDFS-UI [web application](https://en.wikipedia.org/wiki/Web_application) over [HTTP](https://en.wikipedia.org/wiki/HTTP), or via 3rd-party network client libraries.[[37]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-39) There are currently several monitoring platforms to track HDFS performance, including [Hortonworks](https://en.wikipedia.org/wiki/Hortonworks#Hortonworks_Data_Platform_(HDP)), [Cloudera](https://en.wikipedia.org/wiki/Cloudera), and [Datadog](https://en.wikipedia.org/wiki/Datadog).

**Other file systems**[[edit](https://en.wikipedia.org/w/index.php?title=Apache_Hadoop&action=edit&section=5)]

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]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-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](https://en.wikipedia.org/wiki/Operating_system).
* Apache Hadoop Ozone: HDFS-compatible object store targeting optimized for billions small files.
* [FTP](https://en.wikipedia.org/wiki/FTP) file system: This stores all its data on remotely accessible FTP servers.
* [Amazon S3 (Simple Storage Service)](https://en.wikipedia.org/wiki/Amazon_Simple_Storage_Service) object storage: This is targeted at clusters hosted on the [Amazon Elastic Compute Cloud](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/MapR).

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

**JobTracker and TaskTracker: the MapReduce engine**[[edit](https://en.wikipedia.org/w/index.php?title=Apache_Hadoop&action=edit&section=6)]

*Main article:*[*MapReduce*](https://en.wikipedia.org/wiki/MapReduce)

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](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Jetty_(web_server)) 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](https://en.wikipedia.org/wiki/Load_(computing)) 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.