HADOOP: Scalable, Flexible Data Storage and Analysis

By Mike Olson

Beginning in the early 2000s, Google faced a serious challenge. Its mission — to organize the world's information — meant that it was crawling, copying, and indexing the entire Internet continuously. As the number and size of websites grew and Google's service became more popular, the company was forced to digest an ever-increasing corpus more quickly. No commercially available software could handle the volume of data to be processed, and Google's early, custom-built infrastructure was reaching the limits of its ability to scale.

Google's engineers designed and built a new data processing infrastructure to solve this problem. The two key services in this system were the Google File System, or GFS, which provided fault-tolerant, reliable, and scalable storage, and MapReduce, a data processing system that allowed work to be split among large numbers of servers and carried out in parallel. GFS and MapReduce were designed from the very beginning to run on the commodity server hardware that Google used throughout its data center.

In 2004, Google published an academic paper[1] describing its work. Shortly thereafter, a well-known open source software developer named Doug Cutting decided to use the technique it described. Cutting was working on a web crawler called Nutch[2] and was having the same problems with data volumes and indexing speed that had driven Google to develop MapReduce. He replaced the data collection and processing infrastructure behind the crawler, basing his new implementation on MapReduce. He named the new software Hadoop, after a toy stuffed elephant that belonged to his young son.

Engineers at Yahoo!, a direct competitor of Google's in search, also read the paper and learned of Cutting's work. Eager to use the system for processing data of its own, Yahoo! decided to invest in the development of Hadoop. Yahoo!'s leadership, including Cutting, made a strategic decision: Hadoop would be an open source project, free to download and use, but also open to enhancement, improvement, and contributions from a worldwide pool of talented developers. By 2006, Hadoop was in production use at established and emerging web companies.

Hadoop is an open source project[3] and operates under the auspices of the Apache Software Foundation today.

Beyond the Internet

Cloudera was founded in 2008 to commercialize Apache Hadoop, with an awareness of three key trends that are changing the way the entire industry stores and uses data. These changes came first to consumer Internet properties, but loom large for commercial, defense, and intelligence applications as well.

First, data centers are built differently today than they were a decade ago. Instead of a large, centralized server to run applications, organizations typically buy and deploy a collection of rack-mountable commodity servers. Each has several processors and a few terabytes of local disk. Older applications that assume a central, shared-everything hardware platform run poorly in this environment. Modern applications like Hadoop are built to exploit it.

Second, the variety and complexity of available data is exploding. The simple, structured data managed by legacy relational database management system (RDBMS) offerings still exist, but are generally dwarfed by the volume of log, text, imagery, video, audio,

sensor readings, scientific modeling output, and other complex data types streaming into data centers today.

This difference is both qualitative and quantitative. Complex data cannot easily be used in row- or column-oriented database systems. More importantly, information of this kind is big, and the rate at which it is generated is growing. New sources coming online for defense and intelligence applications will help to drive that growth.

Finally, new data processing techniques must be applied to extract the value locked up in this data. Each of the individual data types in the list — audio, video, or imagery, for example — uses algorithms tuned to both the data and the questions that can be answered from it. A key insight among the early web companies was that *combinations* of these data types — putting web logs, which record user activity, alongside the free text of status update messages, for example — deliver insights that neither dataset offers on its own. A computing platform that allows analysts to combine structured and complex data, and to build processing and analytical applications across data sources and types, is a dramatic advance in information management.

These three trends — a shift to scalable, elastic computing infrastructure; an explosion in the complexity and variety of data available; and the power and value that come from combining disparate data for comprehensive analysis — make Hadoop a critical new platform for data-driven enterprises.

Cloudera sees these trends playing out among its customers and the global community of Hadoop users. In financial services, for example, many customers are interested in detecting fraudulent transactions quickly. Credit card companies have long used proprietary algorithms based on transaction history to flag suspicious activity. With the move to online commerce, these firms are able to improve their accuracy by combining transactional data with systems activity logs generated by web servers and other network-connected devices.

In the energy sector, the Tennessee Valley Authority has built a data collection and analysis system called the Open Phasor Data Collector, or OpenPDC, on Hadoop.[4] OpenPDC monitors data streaming at volume from sensors attached to power generation systems. These sensors, deployed around the power grid, report the health and status of each generation facility, and its response to loads on the power grid. Close monitoring and rapid response to changes in the state of the grid allow utilities to minimize or prevent blackouts, and to manage capacity better. This reduces operating costs and better controls greenhouse gas emissions and other environmental risks.

These stories are also repeated in other industries. While the details vary, understanding data is crucial for running a smart, efficient organization.

What is Hadoop?

Like the Google MapReduce system on which it was based, Hadoop consists of two major components: a file store and a distributed processing system.

HDFS

The file store is called the Hadoop Distributed File System, or HDFS. HDFS provides scalable,

Fault-tolerant Hadoop Distributed File System (HDFS)

Provides reliable, scalable, low-cost storage.

HDFS breaks incoming files into blocks and stores them redundantly across the cluster.

fault-tolerant storage at low cost. The HDFS software detects and compensates for hardware issues, including disk problems and server failure.

HDFS stores files across a collection of servers in a cluster. Files are decomposed into blocks, and each block is written to more than one (the number is configurable, but three is common) of the servers. This replication provides both fault-tolerance (loss of a single disk or server does not destroy a file) and performance (any given block can be read from one of several servers, improving system throughput).

HDFS ensures data availability by continually monitoring the servers in a cluster and the blocks that they manage. Individual blocks include checksums. When a block is read, the checksum is verified, and if the block has been damaged it will be restored from one of its replicas. If a server or disk fails, all of the data it stored is replicated to some other node or nodes in the cluster, from the collection of replicas. As a result, HDFS runs very well on commodity hardware. It tolerates, and compensate for, failures in the cluster. As clusters get large, even very expensive fault-tolerant servers are likely to fail. Because HDFS expects failure, organizations can spend less on servers and let software compensate for hardware issues.

The details of HDFS are beyond the scope of this article, but are described clearly in *Hadoop: The Definitive Guide*.[5]

MapReduce

HDFS delivers inexpensive, reliable, and available file storage. That service alone, though, would not be enough to create the level of interest, or to drive the rate of adoption, that characterize Hadoop over the past several years. The second major component of Hadoop is the parallel data processing system called MapReduce.

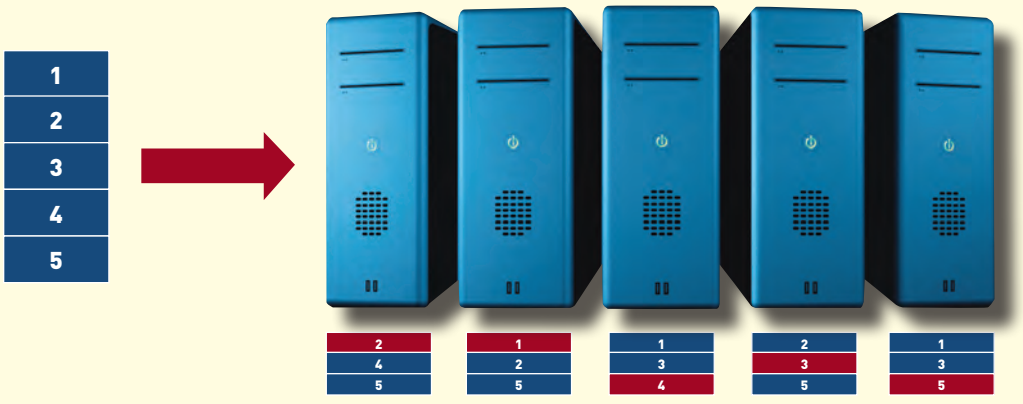
Conceptually, MapReduce is simple.

MapReduce includes a software component called the job scheduler. The job scheduler is responsible for choosing the servers that will run each user job, and for scheduling execution of multiple user jobs on a shared cluster. The job scheduler consults the NameNode for the location of all of the blocks that make up the file or files required by a job. Each of those servers is instructed to run the user's analysis code against its local block or blocks. The MapReduce processing infrastructure includes an abstraction called an *input split* that permits each block to be broken into individual records. There is special processing built in to reassemble records broken by block boundaries.

The user code that implements a map job can be virtually anything. MapReduce allows developers to write and deploy code that runs directly on each DataNode server in the cluster. That code understands the format of the data stored in each block in the file,

MapReduce Software Framework

Offers clean abstraction between data analysis tasks and the underlying systems challenges involved in ensuring reliable large-scale computation.



- Processes large jobs in parallel across many nodes and combines results.
- Eliminates the bottlenecks imposed by monolithic storage systems.
- Results are collated and digested into a single output after each piece has been analyzed.

The big web properties invented MapReduce, and built Hadoop, because they had a data problem that no existing commercial or research system could solve.

and can implement simple algorithms (count the number of occurrences of a single word, for example) or much more complex ones (e.g. natural language processing, pattern detection and machine learning, feature extraction, or face recognition).

At the end of the map phase of a job, results are collected and filtered by a *reducer*. MapReduce guarantees that data will be delivered to the reducer in sorted order, so output from all mappers is collected and passed through a *shuffle and sort* process. The sorted output is then passed to the reducer for processing. Results are typically written back to HDFS.

Because of the replication built into HDFS, MapReduce is able to provide some other useful features. For example, if one of the servers involved in a MapReduce job is running slowly — most of its peers have finished, but it is still working — the job scheduler can launch another instance of that particular task on one of the other servers in the cluster that stores the file block in question. This means that overloaded or failing nodes in a cluster need not stop, or even dramatically slow down, a MapReduce job.

As was true for HDFS, the details of MapReduce are beyond the scope of this article. Interested readers should consult *Hadoop: The Definitive Guide*[5] for an authoritative treatment.

The Character of MapReduce Jobs

It is important to understand the properties that characterize good MapReduce jobs.

First, algorithms must run well on the shared-nothing distributed infrastructure of Hadoop. If a processing job needs to communicate extensively among servers in the cluster, MapReduce is often a poor platform choice. At least in the map phase of any job, the best algorithms are able to examine single records, or a small number of adjacent records stored on the same DataNode, to compute a result. Some of the signature problems that characterize high-performance computing research over the last few decades — weather prediction or modeling nuclear explosions, for example — require extensive intra-node communication. These translate poorly to MapReduce.

That said, MapReduce is a conceptually much simpler parallel processing system than older HPC systems. Existing algorithms, altered somewhat to work on data as stored on HDFS, often run very well. MapReduce is able to execute Java code, but also to use software written in other languages, including high-level languages like C or C++ and scripting languages like PHP, Python, and Perl. Speaking practically, a surprisingly large number of algorithms can be expressed as MapReduce jobs.[6]

MapReduce also excels at exhaustive processing. If an algorithm must examine every single record in a file in order to compute a result, then MapReduce is an excellent choice. Jobs run in parallel on DataNodes in a cluster. The number of DataNodes in a cluster is directly proportional to the amount of data the cluster can store. As a result, the size of a dataset affects performance of a MapReduce job much less than the complexity of the algorithm it implements.

MapReduce is generally used for batch processing, not for the kind of interactive jobs that characterize many existing business intelligence applications. The key advantage of MapReduce is that it can process, literally, petabytes of data in reasonable time to answer a question. Users may have to wait minutes or even hours for a result, but can ask questions that were simply impossible to answer before MapReduce was available.

Why Hadoop?

The big web properties invented MapReduce, and built Hadoop, because they had a data problem that no existing commercial or research system could solve.

The platform is now used to support an enormous variety of applications. These applications are not necessarily characterized by enormous datasets. Instead, they need the three key properties that Hadoop offers.

First, Hadoop is a single, consolidated storage platform for all kinds of data. Of course there are outstanding file and relational storage products available on the market today, and those systems will remain in use for years to come, to solve exactly the problems for which they were designed. Hadoop complements them by delivering a new repository where structured data and complex data may be combined easily.

Second, because of the economies of shared-nothing commodity servers and open source software, Hadoop provides vastly more storage at much lower cost than legacy systems. Fault-tolerant, reliable storage under HDFS costs just a few hundred dollars per terabyte today, and rides the inevitable downward price curve as server and storage vendors improve performance and reduce costs.

Third, and finally, MapReduce exploits the distributed storage architecture of HDFS to deliver scalable, reliable parallel processing services for arbitrary algorithms. Users are not limited to a small set of algorithms delivered by an RDBMS or other vendor. In Hadoop, the storage system is programmable. Users can analyze data using processors attached directly to the disks on which it resides.

This combination — the consolidation of all data types on a low-cost, reliable storage platform that delivers fast parallel execution of powerful analytical algorithms — is new. Hadoop offers data-driven organizations ways to exploit data that they have simply never had before.

MapReduce and Hadoop were created when Google and Yahoo! undertook solving an engineering problem core to their businesses: building an index of more than one billion web pages. The technology has proven invaluable there. But it has proven even more valuable in other, unexpected areas like improving page layout, advertisement selection, spell checking, map rendering, and so on. A general-purpose tool that permits analysis of all data enables new analyses that weren't previously practical or even imagined. Giving all developers easy, powerful access to all data creates a surprising multitude of business improvements.

More data in and of itself generates higher quality information. If a person can access the data, patterns emerge that are not apparent with less data. In a world where information separates winners from losers, successful organizations will distinguish themselves by how they work with data. Some wonderful examples are described by their inventors in *Beautiful Data*.[7] This is no less true for intelligence and defense applications than for commercial ones. The need to capture, process, and analyze information at scale, quickly, is a defining characteristic of the post-9/11 world.

Mike Olson *is founder and CEO of Cloudera, Inc., the commercial Hadoop company. Prior to starting Cloudera, he was CEO of Sleepycat Software, makers of Berkeley DB, the open source embedded database engine. Mike spent two years at Oracle Corporation as Vice President for Embedded Technologies after Oracle's acquisition of Sleepycat in 2006. Prior to joining Sleepycat, Mike held technical and business positions at database vendors Britton Lee, Illustra Information Technologies, and Informix Software. Mike has Bachelor's and Master's degrees in Computer Science from the University of California at Berkeley.*

For more information on Cloudera, please visit ***www.cloudera.com****.*

REFERENCES

1 Dean, J. and Ghemawat, S., "MapReduce: Simplified Data Processing on Large Clusters." Appeared in *Proceedings of the Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, December, 2004. Available online at http://labs.google.com/papers/mapreduce.html, March 2010.

2 Apache Nutch project, http://lucene.apache.org/nutch/, March 2010.

3 Apache Hadoop project, http://hadoop.apache.org/, March 2010.

4 Patterson, J., "The Smart Grid: Hadoop at the Tennessee Valley Authority (TVA)," Cloudera Blog, http://www.cloudera.com/blog/2009/06/smart-grid-hadoop-tennessee-valley-authority-tva/, June 2, 2009.

5 White, T., *Hadoop: The Definitive Guide*. O'Reilly Media, May 2009.

6 Lin, J. and Dyer, C., *Data-Intensive Text Processing with MapReduce*. To be published by Morgan and Claypool. Pre-press edition available at http://www.umiacs.umd.edu/~jimmylin/MapReduce-book-20100219.pdf, February 2010.

7 Segaran, T. and Hammerbacher, J., *Beautiful Data*. O'Reilly Media, July 2009.