# HADOOP 2
## What's New

SANJAY RADIA AND SURESH SRINIVAS

Sanjay is co-founder and architect at Hortonworks, and an Apache Hadoop committer and member of the Apache Hadoop Project Management Commitee (PMC). Prior to co-founding Hortonworks, Sanjay was the chief architect of core-Hadoop at Yahoo and part of the team that created Hadoop. In Hadoop he has focused mostly on HDFS, MapReduce schedulers, high availability, compatibility, etc. He has also held senior engineering positions at Sun Microsystems and INRIA, where he developed software for distributed systems and grid/utility computing infrastructures. Sanjay has a Ph.D. in Computer Science from the University of Waterloo in Canada.
Follow Sanjay on Twitter: @srr
sanjay@hortonworks.com

Suresh is an Apache Hadoop committer and member of the Apache Hadoop Project Management Commitee (PMC). He is a long-term active contributor to the Apache Hadoop project. Prior to co-founding Hortonworks, he served as a software architect at Yahoo! working on Apache Hadoop HDFS, where he developed features and supported some of the largest installations of Hadoop clusters. Suresh also worked for Sylantro Systems in various senior technical leadership roles and developed scalable real-time infrastructure for hosted communications services.
Follow Suresh on Twitter: @suresh_m_s
suresh@hortonworks.com

Hadoop 2 contains fundamental changes in the architecture that significantly extend the platform, taking the compute platform beyond MapReduce and introducing new application paradigms. Similarly, the storage subsystem has been generalized to support other frameworks besides HDFS. The new version significantly improves scalability and performance in both the compute and storage layers, with disk performance up to five times faster and the compute layer scaling to clusters with more than 100k concurrent tasks. Automatic failover of master servers now provides high availability. We cover all these and other key Hadoop 2 features in this article.

### Quick Background

Apache Hadoop is a scalable framework for storing and processing data on a cluster of commodity hardware nodes. Hadoop is designed to scale up from a single node to thousands of nodes. Hadoop has two main components: a computing framework and Hadoop Distributed File System (HDFS). HDFS uses the commodity server nodes and JBOD (Just a Bunch Of Disks) storage drives to store the data and provide large aggregated I/O bandwidth to data. The compute framework uses the same set of server nodes for computation. The key idea is to move computation to where the data is. This enables scalable and efficient ways of storing and processing data. Storage capacity, compute capacity, and I/O bandwidth can be scaled by adding more servers.

HDFS has had a single master server for storing the file system metadata called the NameNode. The files stored on HDFS are split into one or more blocks, typically of size 128 MB. These blocks are stored on slave nodes called DataNodes. To ensure data reliability, multiple replicas of blocks are stored on a set of DataNodes. A client performs file system operations such as creating, modifying, and deleting files at the NameNode. The NameNode records these transactions in a journal, and the data for the files are written by the clients at the DataNodes. The NameNode actively monitors the DataNodes, so if a replica of a block is lost due to disk failures or node failures, new replicas are created.

Computation framework has a master server that manages the compute resources in the slave nodes. It supports parallel, distributed programming paradigms over which a vast amount of data can be processed in a reliable and fault-tolerant manner. Typically, the data is processed in parallel using multiple tasks where each task processes a subset of the data. Traditionally, the MapReduce paradigm is used to process the data in parallel. Hadoop 2.0 has a new compute framework called YARN, which supports MapReduce and other programming paradigms.

### Hadoop 2 Improvements

#### *Architectural Evolution*

Hadoop 2 has made fundamental architectural changes for both the compute and storage sides of the platform.

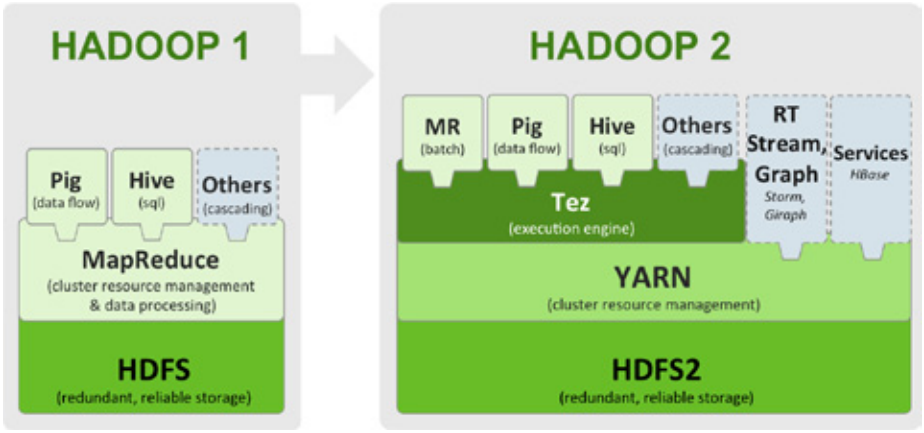


**Figure 1:** Comparison of Hadoop 1 and Hadoop 2 architectures

of execution. Tez allows Hive and Pig programs to be executed more naturally as a single job instead of multiple MapReduce phases, resulting in many orders of magnitude performance improvements. New breeds of applications for stream processing, such as Samza and Storm on YARN, also run as first-class applications (Figure 1). This allows a consolidation of clusters and compute resources to run heterogeneous applications, resulting in less resource fragmentation and more efficient utilization. For the IT department, this means improved hardware capitalization and simplified management.

## YARN—ARCHITECTURAL EVOLUTION IN THE COMPUTE LAYER

Previously, compute resources on Hadoop were only available to MapReduce programs and forced non-MapReduce applications to be modeled as MapReduce. The YARN component generalizes the compute layer to execute not just MapReduce style but other application frameworks. As a result, YARN allows analytics queries to be executed significantly more efficiently and has allowed a new breed of applications, such as stream processing, to be supported in a first-class manner. The new architecture is more decentralized and allows Hadoop clusters to be scaled significantly to more cores and servers. Further, it promises to offer another significant future improvement: the IT department will be able to consolidate other non-Hadoop clusters (such as HPC or virtualization clusters) with the Hadoop cluster.

### Decentralized Resource Management

Hadoop 1 had a single master server called a JobTracker to manage both the compute resources and the jobs that use the resources. YARN splits that function so that a Resource Manager (RM) focuses on managing the cluster resources and an Application Master (AM), one-per-running-application, manages each running application (such as a MapReduce job). The AM requests resources from the RM based on the needs and characteristics of the application being run. For example, a MapReduce application needs compute resources close to the data and has Map and Reduce functions that can be scheduled in phases. On the other hand, an MPI job may be compute-intensive and requires all resources to be scheduled together.

### First-Class Support for Different Application Types

Because the AM is separate from the RM, it can be customized per application type. Hadoop 2 has a specialized AM for MapReduce and another more generalized application framework called Tez that allows generic directed-acyclic-graphs (DAGs)

### Generalized Resource Modeling

Whereas Hadoop 1 modeled compute resources as Map or Reduce slots, YARN allows a more generalized notion of resources. YARN has started with the memory resource (because it was closest to Hadoop 1's "slot") but will soon be extended to support CPU, I/O, and network resources.

## ARCHITECTURAL EVOLUTION IN THE STORAGE LAYER

Hadoop cluster's storage resources were previously available only to HDFS. Similar to YARN, the new storage architecture generalizes the block storage layer so that it can be used not only by HDFS but also other storage services. The first use of this feature is HDFS federation, which allows multiple instances of HDFS namespaces to share the underlying storage. In future versions of Hadoop, other storage services (such as key-value storage) will use the same storage layer.

Another fundamental storage change that is being worked on is support for heterogeneous storage. Hadoop 1 treated all storage devices (be it spinning disks or SSDs) on a DataNode as a single uniform pool; although one could store data on an SSD, one could not control which data. Heterogeneous storage will be part of the 2014 release of Hadoop 2.x, where the system will distinguish between storage types and also make the storage type information available to frameworks and applications so that they can take advantage of storage properties. Indeed, the approach is general enough to allow us to treat even memory as a storage tier for cached and temporary data.

### *Classic Enterprise Features*

Hadoop was initially adopted by Web companies for large-scale data processing. With Hadoop crossing the chasm, different use cases and enterprises are migrating to Hadoop. With that comes the expectation of support for features that enterprise users have come to expect.

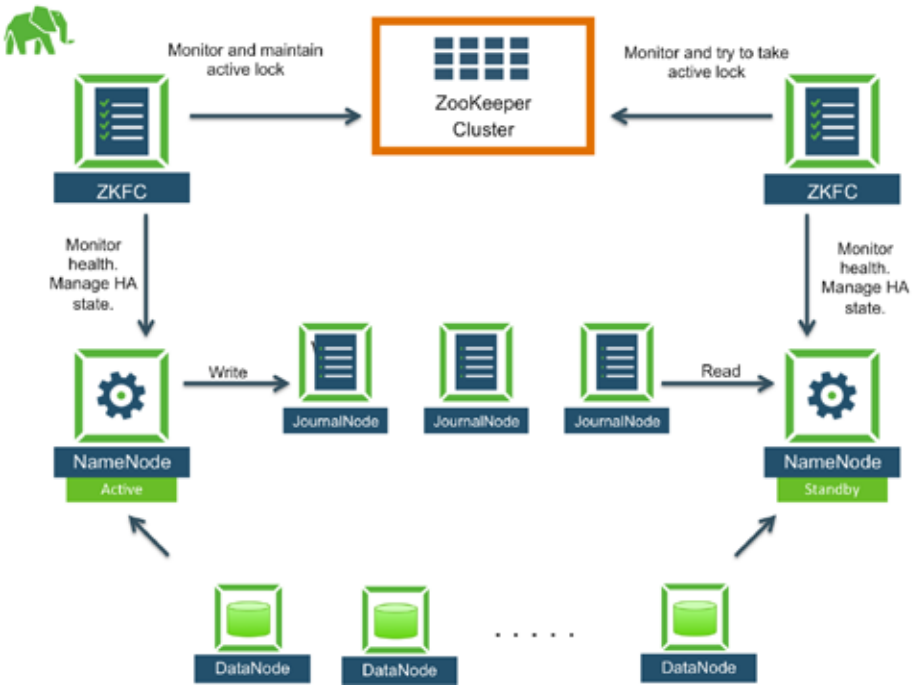


**Figure 2:** ZKFCs monitor NameNodes, and while the active NameNode writes to JournalNodes, the standby NameNode reads from JournalNodes to keep its state updated.

### NAMENODE HIGH AVAILABILITY

While the raw storage layer in HDFS (the block storage layer) is fully distributed and fault-tolerant, the file system metadata was stored in a single master server called the NameNode. When the NameNode is brought down for planned maintenance, or on rare software or hardware failure, the cluster would be unavailable until the NameNode is restarted. Distributions such as Hortonworks Data Platform (HDP) had cold failover for the NameNode using industry standard frameworks, such as Linux HA and VMware vSphere, in their Hadoop 1 distribution. Hadoop 2 adds support for a hot standby NameNode along with a journal service. In case of failure of the active NameNode, automatic failover is triggered and the standby NameNode becomes active.

### FAILOVER CONTROLLER

A new watchdog daemon called the ZKFC (ZooKeeper-based Failover Controller) manages failover of NameNodes. This daemon runs on each of the NameNodes and maintains a session with the ZooKeeper. Using ZooKeeper for coordination, one of the ZKFC becomes the leader and elects the local NameNode as active. The ZKFC performs a periodic health check of the NameNode. When the active NameNode fails health check, the local ZKFC resigns as the leader. Similarly, when the active NameNode machine fails, ZooKeeper detects the loss and removes the ZKFC from the failed node as the leader. This results in automatic failover; the ZKFC running on standby becomes the leader and makes the local standby NameNode active.

### QUORUM JOURNAL MANAGER

In Hadoop 2, the file system journal no longer needs external NAS storage. The NameNode writes the journal to external daemons called Journal Nodes. The Quorum Journal Manager ensures every transaction is written to a quorum number of journal nodes using a distributed commit protocol based on Multi-Paxos. Because only one NameNode can successfully write to a quorum number of Journal Nodes, the corruption due to a split-brain condition is avoided. This ensures that the redundant and consistent copies of journal are persisted on the journal nodes. The standby NameNode reads the transactions from the journal and updates its state to stay in sync with the active NameNode.

### NFS SUPPORT

Access to HDFS is usually done through the HDFS client library or over HTTP REST APIs. Lack of seamless integration with the client's file system makes it difficult for users and impossible for some applications to access HDFS. Hadoop 2 adds NFS version 3 support to make this integration easy.

NFS access is enabled using stateless NFS gateways. The NFS gateway's main functionality is translation of the NFS protocol to the HDFS native protocol. The gateway is started as a daemon on the slave nodes in a Hadoop cluster. The gateway supports three services: rpcbind (or portmap), mountd, and nfsd. HDFS is mounted on the client system, and applications can access HDFS through the local file system.

#### HDFS SNAPSHOTS

Hadoop 2 adds support for file system snapshots. A snapshot is a point-in-time image of the entire file system or a subtree of a file system. A snapshot has many uses:

- Protection against user errors: An admin can set up a process to take snapshots periodically. If a user accidentally deletes files, these can be restored from the snapshot that contains the files.
- Backup: If an admin wants to back up the entire file system or a subtree in the file system, the admin takes a snapshot and uses it as the starting point of a full backup. Incremental backups are then taken by copying the difference between two snapshots.
- Disaster recovery: Snapshots can be used for copying consistent point-in-time images over to a remote site for disaster recovery.

The snapshots feature supports read-only snapshots; it is implemented only in the NameNode, and no copy of data is made when the snapshot is taken. Snapshot creation is instantaneous. All the changes made to the snapshotted directory are tracked using modified persistent data structures to ensure efficient storage on the NameNode.

#### RPC IMPROVEMENTS AND WIRE COMPATIBILITY

Hadoop 2 has several improvements to the RPC layer shared by HDFS, YARN, and MapReduce v2. The on-the-wire protocol now uses protocol buffers and is no longer based on Java serialization. This helps in extending the protocol in the future without breaking the wire protocol compatibility. RPC also adds support for client-side retries of the operation, a key functionality for supporting highly available server implementation. These improvements help in running different versions of daemons within the cluster, paving the way for rolling upgrades.

### *Other HDFS Improvements*

#### I/O IMPROVEMENTS

Improvements to HDFS speed and efficiency are added on an ongoing basis. There are many improvements to HDFS interfaces. A better short-circuit interface based on UNIX Domain Sockets allows clients to read from the local file system directly instead of inefficiently over a socket from the DataNode. This interface also now supports zero copy reads. The CRC checksum calculation done during both reads and writes is now optimized using the Intel SSE4.2 CRC32 instruction. All of these improvements have made I/O 2.5 to 5 times faster than the previous releases.

#### APPEND SUPPORT

Hadoop 1 required HDFS files to be immutable once they were created. Hadoop 2 allows one to append data to a previously created file.

### *Expanding the Community and Use Cases*

YARN in Hadoop 2 expands the ecosystem beyond MapReduce and allows new kinds of applications to run on the cluster. Similarly the generalization of storage promises Hadoop storage to be used beyond HDFS.

#### WINDOWS SUPPORT

Hadoop was originally developed to support the UNIX family of operating systems. With Hadoop 2, the Windows operating system is natively supported. This work is simplified by the fact that Hadoop was written in Java. The dependencies on UNIX for compute and storage resource control now has been generalized to support Windows. This extends the reach of Hadoop significantly to a sizable Windows Server market.

#### OPENSTACK CLOUD SUPPORT

There is a growing trend to run Hadoop-on-demand and shared infrastructure. Hadoop 2 supports the OpenStack Swift file system, and it has topology improvements for virtualized environments. With OpenStack support for spinning Hadoop clusters up and down, Hadoop can now be run on virtualized hardware, both in public and private datacenter clouds.

### *Next Steps*

Hadoop, which was originally designed around batch processing using commodity disks and servers, is changing in the face of a number of trends. The Big Data application space and Hadoop usage pattern, along with the underlying hardware technology and platform, are rapidly evolving. Further, the increasing prevalence of cloud infrastructure, both public and private, is influencing Hadoop development. Hadoop is evolving to deal with changes in how clusters are being built. HDFS and YARN architecture are growing to adapt to such changes.

Hadoop has become the de facto kernel for the Big Data platform. These exciting developments are being driven by a dedicated Apache community, all in the open. People interested in participating in this technological revolution are welcome to visit http://hadoop.apache.org.