Hortonworks Data Platform

Using Apache Hadoop 2.0

(Aug 1, 2014)

docs.hortonworks.com

Hortonworks Data Platform: Using Apache Hadoop 2.0

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, Zookeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

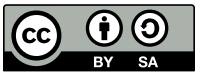

Table of Contents

1. Using Apache Hadoop

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

The project includes these modules:

- Hadoop Common: The set of common utilities that support other Hadoop modules
- Hadoop Distributed File System (HDFS): A distributed file system that provides high-throughput access to application data
- Hadoop YARN: Framework for job scheduling and cluster resource management
- Hadoop MapReduce: A YARN-based system for parallel processing of large data sets

To learn more about Apache Hadoop, see Apache Hadoop.

In this section:

- Hadoop Common
- Using Hadoop HDFS
- Using Hadoop MapReduce
- Using Hadoop YARN

1.1. Hadoop Common

Hadoop Common is the set of common utilities that support other Hadoop modules.

In this section:

- Setting up a Single Node Cluster
- Cluster Setup
- CLI MiniCluster
- Using File System (FS) Shell
- Hadoop Commands Reference

1.2. Using Hadoop HDFS

HDFS is a distributed file system that provides high-throughput access to data. It provides a limited interface for managing the file system to allow it to scale and provide high

throughput. HDFS creates multiple replicas of each data block and distributes them on computers throughout a cluster to enable reliable and rapid access.

Use the following resources to learn more about Hadoop HDFS:

- HDFS High Availability Using the Quorum Journal Manager
- HDFS High Availability with NFS
- HDFS Federation
- WebHDFS REST API

1.3. Using Hadoop MapReduce

In this section:

- Encrypted Shuffle
- Pluggable Shuffle and Pluggable Sort

1.4. Using Hadoop YARN

The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker i.e. resource management and job scheduling/monitoring, into separate daemons: a global **ResourceManager** and per-application **ApplicationMaster (AM)**.

The ResourceManager and per-node slave, the **NodeManager (NM)**, form the new, and generic, system for managing applications in a distributed manner. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system.

The per-application ApplicationMaster is, in effect, a framework specific entity and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the component tasks.

The ResourceManager has a pluggable **Scheduler**, which is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues etc. The Scheduler is a pure scheduler in the sense that it performs no monitoring or tracking of status for the application, offering no guarantees on restarting failed tasks either due to application failure or hardware failures. The Scheduler performs its scheduling function based on the resource requirements of the applications; it does so based on the abstract notion of a Resource Container which incorporates resource elements such as memory, CPU, disk, network etc. The NodeManager is the per-machine slave, which is responsible for launching the applications' containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager. The per-application ApplicationMaster has the responsibility of negotiating appropriate resource containers from the Scheduler, tracking their status and monitoring for progress. From the system perspective, the ApplicationMaster itself runs as a normal container.

Use the following resources to learn more about YARN:

- YARN architecture

- Writing YARN applications
- Capacity Scheduler
- Web Application Proxy
- Using YARN REST APIs
- Hadoop Auth