# High Availability in the Hadoop Ecosystem

## Table of Contents

- **Executive Summary** ............................................................... 2
- **Background** ........................................................................... 3
- **Storage** .................................................................................. 4
  - MapR ......................................................................................... 4
  - Other Distributions .................................................................... 5
    - Apache Hadoop 0.21 and Prior Versions (including CDH) .......... 5
    - Linux HA / DRBD ................................................................. 6
    - AvatarNode .......................................................................... 6
    - Apache Hadoop 0.23/0.24 (NameNode HA and HDFS Federation) .... 6
- **MapReduce** ............................................................................ 7
  - MapR ......................................................................................... 7
  - Other Distributions .................................................................... 8
    - Apache Hadoop 0.21 and Prior Versions (including CDH) .......... 8
    - Apache Hadoop 0.23 (MapReduce 2.0) .................................... 8
- **Conclusion** ............................................................................. 8
Executive Summary

The MapR Distribution for Apache Hadoop provides high availability with no single points of failure across the entire stack. In the storage layer, MapR’s Distributed NameNode HA™ architecture provides high availability with self-healing and support for multiple, simultaneous failures, with no additional hardware whatsoever. In the MapReduce layer, MapR’s JobTracker HA makes JobTracker failures transparent to applications – the currently running tasks continue to execute during the failover process. In the NFS layer, MapR automatically manages virtual IP addresses and balances them between the nodes so that failures are transparent to clients that are reading and writing data via NFS. (Note that NFS support is not available in any other distribution.)

These capabilities, combined with MapR’s unique capabilities for data protection (snapshots) and disaster recovery (mirroring), position MapR as the only distribution that provides business continuity. By comparison, Apache Hadoop 0.21 and prior versions, and the corresponding commercial distributions such as Cloudera’s Distribution including Apache Hadoop (CDH) also do not provide any HA capabilities. Furthermore, future versions of Apache Hadoop through 2012 will provide only limited HA, with many architectural issues and limitations. The following table outlines the key differences in business continuity between MapR, Apache Hadoop 0.20 and the planned releases of Apache Hadoop 0.23/0.24:

<table>
<thead>
<tr>
<th></th>
<th>MapR</th>
<th>Apache Hadoop 0.21 and Prior Versions</th>
<th>Future Release: Apache Hadoop 0.23/0.24</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Storage HA</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HA architecture</td>
<td>Distributed</td>
<td>None (single point of failure)</td>
<td>Active / Passive</td>
</tr>
<tr>
<td># of failures tolerated</td>
<td>Unlimited</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Self-healing</td>
<td>Scalability (files)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Dedicated hardware needed for metadata</td>
<td>0</td>
<td>2 servers (NameNode, Checkpoint/Backup Node) + NAS (e.g., NetApp)</td>
<td>2+ servers + NAS (e.g., NetApp)</td>
</tr>
<tr>
<td>Additional servers needed to store 1 billion files</td>
<td>0</td>
<td>Can’t scale beyond 50-100M files in a cluster!</td>
<td>20-40 high-end servers w/ HDFS Federation</td>
</tr>
<tr>
<td>Automated rolling upgrades</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>MapReduce HA</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HA architecture</td>
<td>Active / Passive</td>
<td>None (single point of failure)</td>
<td>Active / Passive</td>
</tr>
<tr>
<td>Task failures</td>
<td>No Tasks Fail</td>
<td>All Tasks Fail</td>
<td>Running tasks fail</td>
</tr>
<tr>
<td><strong>NFS HA</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HA architecture</td>
<td>Distributed (automatic VIP failover)</td>
<td>No NFS access to data</td>
<td>No NFS access to data</td>
</tr>
<tr>
<td><strong>Data Protection</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HA architecture</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Snapshots for recovery from user/application errors</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Mirroring for disaster recovery and remote backup</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Available Today</td>
<td>Yes</td>
<td>Yes</td>
<td>No — expected 2012 release</td>
</tr>
</tbody>
</table>
BACKGROUND

One of the capabilities that differentiate MapR from other Hadoop distributions is high availability (HA). With the MapR Distribution for Hadoop, there are no single points of failure, so any component can fail with no impact to the user, jobs or administrator. This document describes the HA capabilities of MapR’s storage services and MapReduce framework. Note that MapR also provides complete HA for the NFS access layer, as well as HBase.

We asked many Hadoop users about their HA requirements. The following requirements were the most common:

1) **No downtime or data/job loss on failure.** In other words, an individual hardware or software failure should not have any impact on applications.

2) **Self-healing.** The cluster should automatically heal from a failure so that it can tolerate additional failures in the future. No human intervention should be required after a failure. Some failures (e.g., disk failure) obviously require intervention, but in a self-healing architecture, intervention is not required in order to recover from a failure and restore to a highly available state, assuming there are enough physical resources available.

3) **Tolerate multiple failures.** The cluster should be able to tolerate more than one simultaneous failure. Ideally, the degree of tolerance should be configurable so that the administrator or user can choose the appropriate resource level to support how many failures can be tolerated (e.g., production data is probably more important than temporary scratchpad data).

4) **No additional hardware.** There should be no need to purchase, deploy and maintain additional hardware (e.g., nodes).

5) **100% commodity hardware.** There should be no dependency on non-commodity hardware, such as hardware load balancers or network-attached storage (e.g., NetApp).

6) **Easy to set up.** It should be trivial to set up HA. For example, the administrator should not need to install any third-party software or hardware (commercial or open source).

The driving factors behind these requirements are a combination of increasing the level of protection (mean time to failure, or MTTF) and making it easy to deploy and maintain. Hadoop users love the fact that they don’t have to attend to a DataNode failure until they have time, and expect that type of guarantee for every service.

The following sections describe the high availability architecture of the MapR distribution, and how it compares with the current state and roadmap of other distributions and projects.
**Storage**

**MapR**

MapR’s Lockless Storage Services feature a distributed HA architecture:

- The metadata is distributed across the entire cluster. Every node stores and serves a portion of the metadata.

- Every portion of the metadata is replicated on three different nodes (this number can be increased by the administrator). For example, the metadata corresponding to all the files and directories under `/project/advertising` would exist on three nodes. The three replicas are consistent at all times except, of course, for a short time after a failure.

- The metadata is persisted to disk, just like the data.

The following illustration shows how metadata is laid out in a MapR cluster (in this case, a small 8-node cluster). Each colored triangle represents a portion of the overall metadata; or in MapR terminology, the metadata of a single volume:

![Metadata Layout Illustration](image)

In this example each portion of the metadata is on three different nodes, so the system can tolerate two failures without a problem. Increasing the number of replicas for the metadata of a volume is as simple as modifying the value in a textbox in the MapR Control System, or running a command via the CLI or REST API.

If a node fails, the metadata that was on that node is quickly re-replicated to other nodes in the cluster so that the replication factor can quickly hit the configured level again. This is what makes MapR’s HA self-healing. Note that all nodes in the cluster can participate in the healing process, making it extremely efficient. The following list outlines one possible chain of events that could take place if FileServer 6 (in the preceding example) crashes:

1) FileServer 1 copies the “yellow” metadata from FileServer 3
2) FileServer 2 copies the “green” metadata from FileServer 4
3) FileServer 3 copies the “purple” metadata from FileServer 7

It is worth noting that enabling HA in MapR’s lockless storage services does not require any effort. By default, all metadata is replicated three times, just like the regular data. In addition, there is no need to deploy additional nodes or any non-commodity hardware.
Other distributions

MapR is currently the only distribution that provides HA. There have been a number of projects that attempted to increase the availability of Hadoop clusters, but neither these projects nor the existing roadmap address the majority of the preceding HA requirements. This section provides more details on the existing Hadoop distributions as well as three related projects: AvatarNode, Linux HA/DRBD and NameNode HA (the latter has not been implemented).

Apache Hadoop 0.21 and prior versions (including CDH)

In other Hadoop distributions, all HDFS metadata (e.g., namespace, block locations) is kept in memory on a single node, called the NameNode. In addition, the NameNode typically stores the namespace information on both a local disk and a non-commodity NAS system (e.g., NetApp), while block locations are maintained solely in memory.

If the NameNode fails, the entire cluster is down. Furthermore, since the NameNode doesn’t keep block locations on disk, the NameNode must reconstruct the information from all the DataNodes. This can take hours.

The following illustration outlines the architecture of HDFS. Each colored triangle represents a portion of the metadata, and the grey rectangles represent block locations. As you can see, the metadata is stored on a single machine, and it must fit entirely in memory.

Note that HDFS includes a service called the Secondary NameNode. This service runs on a separate node, and its job is to merge the NameNode’s edit log into the fsimage so that the edit log does not grow indefinitely (this would cause a NameNode restart to take forever). It is important to keep in mind that the Secondary NameNode cannot become a Primary NameNode, and thus does not provide any form of HA. Because the term Secondary NameNode is misleading, this service was renamed Checkpoint Node in Hadoop 0.21. In addition, an alternative service called Backup Node was introduced in 0.21. The Backup Node serves the same role as the Checkpoint Node, but is able to checkpoint more efficiently (because the edits are continuously streamed from the NameNode to the Backup Node, instead of being downloaded periodically).
**Linux HA/DRBD**

Linux HA and DRBD have been used in the past by a few Hadoop users to improve the availability of their Hadoop clusters. This solution employs a standby NameNode that is ready to assume responsibility when the active NameNode fails. However, this approach has several problems. For example, the standby NameNode is cold, so the cluster could be down for a significant amount of time (e.g., over an hour) following a NameNode failure, waiting for the backup NameNode to read (and merge) the file system image and edit log, and then waiting to receive a block report from every DataNode in the cluster.

Although this solution was originally described in Cloudera’s own blog, the company later commented that “… we want to clarify our position that we do not officially support nor fully endorse this method of doing NameNode HA. DRBD is hard to get right, and you can easily corrupt your fsimage file with a subtle mistake. … our recommendation is to design your application around HDFS’s availability semantics. This might include using Flume for reliable delivery, or having mirrored HDFS systems.”

To summarize, the Linux HA/DRBD solution does not meet any of the six preceding HA requirements.

**AvatarNode**

AvatarNode, developed by Facebook, makes it possible for an administrator to switch a live Hadoop cluster’s NameNode from one node to another node so that the administrator can then perform maintenance on the node. The failover must be manually initiated by an administrator, so it doesn’t provide protection from software or hardware failures, and does not meet any of the six preceding HA requirements.

With that said, AvatarNode is important to Facebook because the company is maintaining its own internal version of Hadoop, and has the need to upgrade the Hadoop software on a frequent basis. The AvatarNode project makes it possible to perform minor upgrades to the NameNode with no downtime.

**Apache Hadoop 0.23/0.24 (NameNode HA and HDFS Federation)**

NameNode HA is another HA-related project that aims to add high availability to HDFS. It is described in umbrella JIRA HDFS-1623. NameNode HA is still in the design phase. At a high level, the idea is to actively replicate all NameNode state required to quickly restart the process. There are still many open questions, so it’s not clear what the design will be, but it will likely be an extension of AvatarNode. None of the proposals provide self-healing or the ability to tolerate multiple failures. In addition, the proposals all require a dedicated standby server, and most likely a commercial NAS system. The illustration to the right outlines the proposed architecture.
Note that even under the proposed NameNode HA architecture, all metadata must reside in memory on the NameNode. As a result, a cluster with one Primary NameNode and one Standby NameNode can only support about 50-100M files (similarly to an Apache Hadoop 0.20/0.21 cluster). To address this limitation, a project called HDFS Federation is being developed. HDFS Federation allows administrators to divide the namespace between multiple NameNodes. HDFS Federation doesn’t provide any HA (in fact, it introduces multiple single points of failure), but it does provide some relief for the “# of files” limit in Hadoop. However, it does so at an extremely high cost:

- HDFS Federation introduces multiple single points of failure (each of the NameNodes is a single point of failure), unless the customer deploys a dedicated Standby NameNode for each of those NameNodes. However, that doubles the hardware and associated costs. For example, to store 1B files in a cluster, one would need to deploy 10-20 Primary NameNodes, and 10-20 Standby NameNodes, for a total of 20-40 dedicated NameNode servers! A cluster with 10B files would require 200-400 dedicated NameNode servers. Compare that to a MapR cluster, which would require zero additional servers to scale to many billions of files (and beyond).

- Any time a new NameNode is added, the configuration on every client and DataNode must be updated. This is because HDFS Federation requires clients to share a common mount table.

**MapReduce**

**MapR**

MapR is the only distribution that provides JobTracker high availability. When the JobTracker fails, the system automatically restarts the JobTracker on a different node, and the TaskTrackers automatically reconnect to the new JobTracker. All MapReduce jobs continue to run during this time. In fact, even the tasks that are in flight are not affected by the failure. This is important because losing all the running tasks in a MapReduce job can be almost as disruptive as losing the entire job (i.e., both the tasks that have completed, and the ones that are still running). MapR’s JobTracker HA is designed to meet all six of the preceding HA requirements. During the deployment phase, the administrator can decide which machines are candidates for running the JobTracker. At any point in time, one of the available candidates serves as the active JobTracker. Therefore, the MapReduce layer can tolerate an arbitrary number of simultaneous failures, and the administrator does not need to intervene. Also, no additional hardware is needed, and enabling JobTracker HA is as simple as running a single command on each candidate node.

In terms of high availability, MapReduce 2.0 offers a subset of the capabilities of MapR’s JobTracker HA. When the MapReduce ApplicationMaster (i.e., the job-specific JobTracker) fails, it is restarted automatically, but all incomplete map/reduce tasks must be restarted.
Other distributions

Apache Hadoop 0.21 and prior versions (including CDH)
Apache Hadoop 0.20 and 0.21 do not have any high availability for the JobTracker. If the JobTracker fails, all running jobs are terminated and their progress is lost. This can be a serious problem especially in a scenario in which jobs must complete within a specific timeframe.

Apache Hadoop 0.23 (MapReduce 2.0)
MapReduce 2.0 is a new architecture for MapReduce that was proposed by Yahoo!, and closely resembles the architecture that Google uses internally. MapReduce 2.0 provides various benefits, such as scalability, backward compatibility and higher cluster utilization (because it eliminates the need for predefined map and reduce slots). MapR is currently integrating MapReduce 2.0 into the MapR Distribution for Apache Hadoop, so that MapR users can take advantage of these benefits as soon as the community-driven work on the MapReduce 2.0 framework is completed. MapReduce 2.0 will be available in the MapR distribution within four weeks of its general availability in Apache Hadoop or CDH (these four weeks will allow MapR the time to run the system through the company’s rigorous QA processes).

Conclusion
The MapR Distribution for Apache Hadoop, unlike all other distributions, was designed from the ground up to provide high availability and other business continuity capabilities, such as data protection and disaster recovery. MapR is the only distribution with no single points of failure. Other distributions are expected to introduce limited HA capabilities in 2012, but due to fundamental architectural limitations they are unable to address the most important availability and ease of use requirements. In addition, they will require significant hardware and operating expenses just to tolerate a single failure.