

MapR's Direct Access NFS vs. Hadoop FUSE

Executive Summary

The MapR Distribution for Apache Hadoop greatly enhances Hadoop through Direct Access NFS. This capability is enabled by MapR's replacement of the Hadoop Distributed File Service (HDFS) with a highly scalable, enterprise-class storage service layer that supports random reads and writes and allows users to mount the cluster over NFS so that standard file-based applications (command line utilities, file browsers, databases, etc.) can work directly with the data in the cluster, and application servers can write their data directly into the cluster. Other Hadoop distributions, unlike MapR, are based on HDFS, which is a read-only (or write-once) file system, with no support for NFS. MapR provides Direct Access NFS along with 100% compatibility at the API layer for MapReduce and HDFS so there is no need to change or recompile existing applications.

Other distributions claim that a "Hadoop FUSE" component provides similar functionality to MapR's NFS, but this is not true. First, their back-end storage service (HDFS) only supports sequential I/O, so a client application can fail at any point in time. Second, Hadoop FUSE does not provide the minimal consistency guarantees needed by traditional applications (a client cannot always read what it just wrote). Third, a client component must be installed and maintained on every client, and only Linux clients are supported. Finally, while MapR's NFS provides full wire-speed performance, Hadoop FUSE is extremely slow.

The following table provides a high-level comparison between MapR's Direct Access NFS feature and Hadoop FUSE.

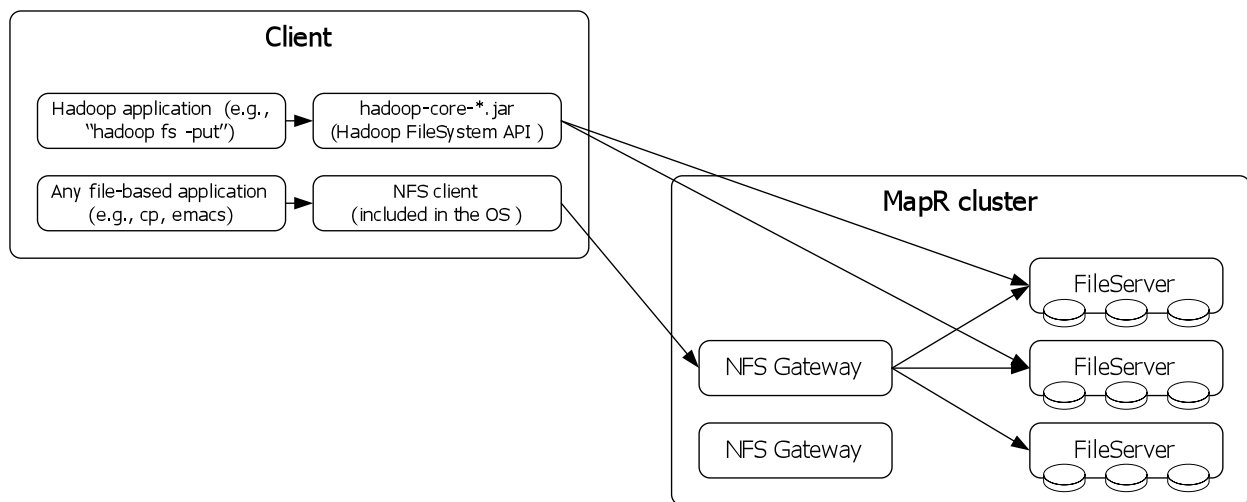
	NFS (MapR)	Hadoop FUSE	Hadoop FUSE exported over NFS
Supported client operating systems	Linux, Mac, Windows, Solaris, etc.	Linux	<i>Do not use! (see explanation below)</i>
No client software installation needed	Yes	No	
Random read/write	Yes	No (<u>arbitrary applications will fail</u>)	
Performance	Wire-speed	Slow	
"Self consistency" (read what you write)	Yes	No	
Stable	Yes	No (see explanation below)	
Recommended by MapR/Cloudera for direct writing from application server?	Yes	No	

MapR

FileSystem API and NFS

Like HDFS, MapR's storage services implement the Hadoop FileSystem API. However, in addition to the FileSystem API, MapR also provides an NFS interface so that clients can mount the cluster and access the data directly.

The following diagram shows how this works:



Each node in the cluster has a FileServer service, whose role is similar in many ways to the DataNode in HDFS. In addition, there can be one or more NFS Gateway services running in the cluster. In many deployments the NFS Gateway service runs on every node in the cluster, alongside the FileServer service.

A MapR cluster can be accessed either through the Hadoop FileSystem API or through NFS:

- **Hadoop FileSystem API.** To access a MapR cluster via the Hadoop FileSystem API, the MapR/Hadoop client must be installed on the client. MapR provides easy-to-install clients for Linux, Mac and Windows. The Hadoop FileSystem API is in Java, so in most cases client applications are developed in Java and linked to the hadoop-core-*.jar library.
- **NFS.** To access a MapR cluster over NFS, the client mounts any of the NFS Gateway services. There is no need to install any software on the client, because every common operating system includes an NFS client. In Windows, the MapR cluster becomes a letter drive (Y:, Z:, etc.), whereas in Linux and Mac the cluster is accessible as a directory in the local file system (e.g., /mapr). (Note that some low-end Windows versions do not include the NFS client.)

The Hadoop FileSystem API is designed for MapReduce (with functions such as `getFileBlockLocations`), so MapReduce jobs normally read and write data through that API. However, the NFS interface is often more suitable for applications that are not specific to Hadoop. For example, an application server can use the NFS interface to write its log files directly into the cluster. Text editors, command-line utilities and file browsers can also use the NFS interface to access the cluster.

Note that MapR provides high availability for NFS in the M5 edition. The administrator allocates a pool of virtual IP addresses (VIPs), which the cluster then automatically assigns to the NFS Gateways. A VIP automatically migrates from one NFS Gateway service to another in the event of a failure, so that all clients who mounted the cluster through that VIP can continue reading and writing data with no impact. In a typical deployment, a simple load-balancing scheme, such as DNS round-robin, is used to uniformly distribute clients among the different NFS Gateways (i.e., VIPs).

Random read/write

The MapR Distribution for Apache Hadoop includes an underlying storage system that supports random reads and writes (with support for multiple readers and writers simultaneously). This provides a significant advantage over other distributions, in which HDFS provides a write-once storage system (similar to FTP, or a CD-ROM).

Having support for random reads and writes is necessary in order to provide NFS access (and, more generally, any kind of access for non-Hadoop applications). NFS is a simple protocol in which the client sends the server requests to write (or read) n bytes at offset m in a given file. In a MapR cluster, the NFS Gateway service receives these requests from the client and translates them into the corresponding RPCs to the FileServer services. The server-side in the NFS protocol is mostly stateless – there is no concept of opening or closing files.

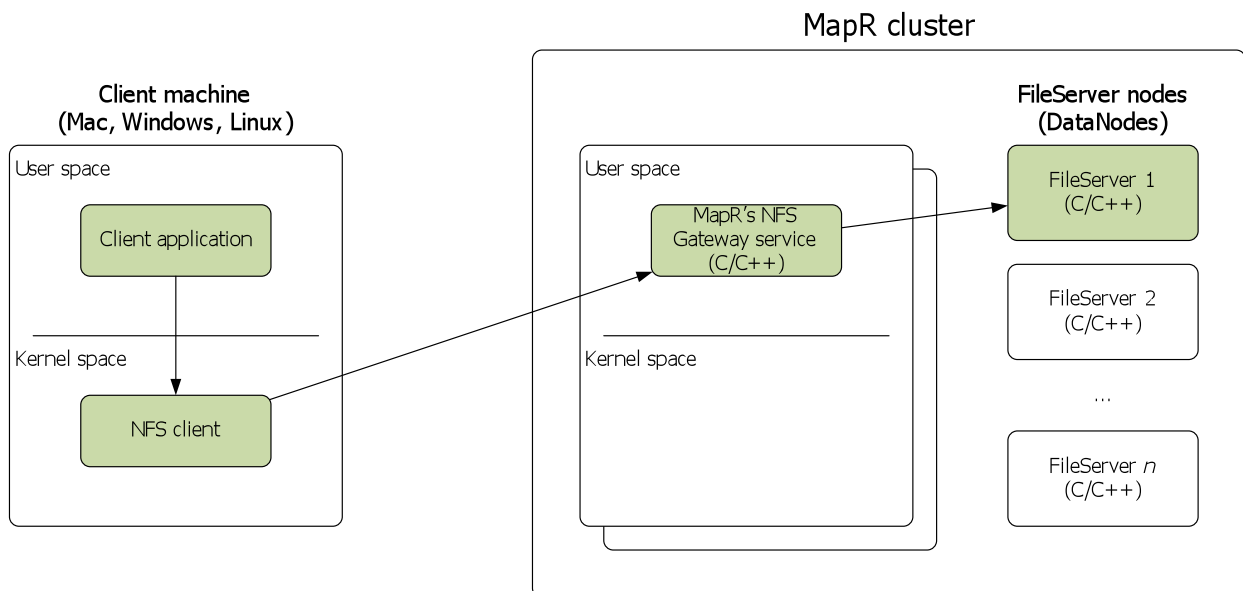
Mountable HDFS (Hadoop FUSE)

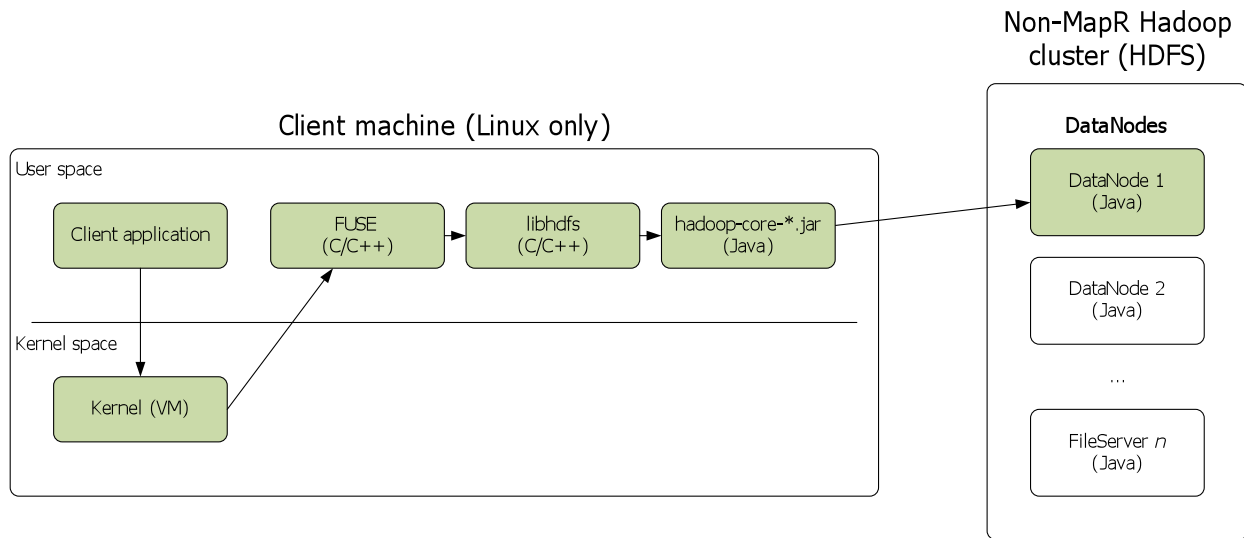
Other Hadoop distributions (e.g., CDH) cannot provide NFS access because the underlying storage system (HDFS) does not support random read/write semantics. A project called Hadoop FUSE was developed so that Hadoop clusters can be mounted on Linux systems. This approach has many problems:

- It requires a client installation. In other words, software must be installed on every machine that will read from or write to the Hadoop cluster. In addition, if the Hadoop API changes, the software must be upgraded on each client. Installing and maintaining Hadoop software on every client (e.g., every application server in the datacenter) can be painful.
- FUSE only works on Linux. Windows and Mac clients cannot mount the cluster.

- If the application doesn't write data sequentially, it receives an I/O error ("Operation not supported"). Not only do many applications not write sequentially, it is impossible to know how an application behaves without actually looking at its source code.
- It is very slow. Unlike NFS clients which are typically part of the OS kernel, a FUSE client runs in user space. In Hadoop FUSE, the FUSE implementation calls libhdfs, a C/C++ wrapper for hadoop-core-*.jar. The libhdfs library calls the corresponding Java FileSystem API method, which in turn communicates with the cluster. All these memory copies and transitions from kernel to user space and then the JVM impact performance.
- No "self consistency". If you write to a file and then immediately read the file, the read returns empty. You must wait several seconds after writing to the file before reading it. This behavior is obviously not something that applications expect, so the results can be catastrophic depending on the application.
- Poor stability and data loss. We tried Hadoop FUSE on multiple platforms: CentOS 5.4 and Ubuntu 10.10. In Ubuntu 10.10 (the latter using the Cloudera Demo VM). In Ubuntu 10.10, any time we tried to edit an existing file (e.g., in vi), not only was the new content not saved, but the existing content of the file was lost. It appears that when the client tried to rewrite the file, the NameNode didn't think that the client had opened the file and thus threw a LeaseExpiredException.

To further illustrate why MapR's Direct Access NFS provides much higher performance than Hadoop FUSE, the following diagrams outline how data flows from the client application to the cluster:





It's worth noting that Hadoop FUSE does work with MapR, because it uses the Hadoop FileSystem API internally, but we strongly advise against using it due to all these problems.

Export FUSE over NFS?

In theory, it is possible to mount HDFS on a Linux server using FUSE and then export that through NFS using the standard Linux NFS server. However, this should be avoided due to the following reasons:

- FUSE relies on the kernel's inode cache since FUSE is path-based and not inode-based like NFS. In other words, an NFS client specifies an inode number in a read/write request, and the server must be able to translate that inode into a path so it can be passed into the Hadoop FUSE implementation. However, the NFS server is generally stateless, so it has no way of knowing whether or not a client is keeping a reference to a file. As a result, it may flush an inode from its cache even though a client is still referencing the corresponding file, and the NFS server will no longer be able to translate the inode number into a path, so additional requests from the client will fail. (Note that FUSE provides a "noforget" option to mitigate this problem by never flushing inodes from the cache, but in the case of HDFS this is not practical because the NFS server will eventually run out of memory.)
- NFS reorders writes. So even if an application is writing sequentially, the write requests will be processed in random order, and the Hadoop FUSE client will fail because HDFS can only write sequentially.
- Exporting FUSE-mounted file systems over NFS is only possible in recent Linux kernels. It doesn't work with RHEL/CentOS 5.x.

See <http://wiki.apache.org/hadoop/MountableHDFS> and <https://github.com/fuse4x/fuse/blob/master/README.NFS> for more details.

Summary

The MapR Distribution for Apache Hadoop includes a robust, enterprise-class storage service that supports random reads and writes and exposes the standard NFS interface so that clients can mount the cluster and read and write data directly. This capability makes Hadoop much easier to use, and enables new classes of applications.

Other Hadoop distributions claim to support a “mountable HDFS” via FUSE. However, as demonstrated in this article, Hadoop FUSE simply does not work for most desired use cases, due to HDFS and FUSE limitations.