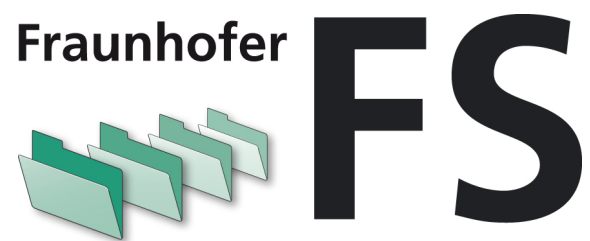


*FraunhoferFS*  
**User Guide**



## Table of Contents

1.	Introduction .....	3
1.1.	Key Benefits .....	3
1.2.	System Architecture .....	4
2.	Installation and Setup .....	5
3.	Tuning and advanced Configuration.....	5
3.1.	Native Infiniband Support .....	5
3.2.	Caching.....	6
3.3.	Striping .....	6
3.4.	Storage Tuning .....	6
3.5.	Network Tuning.....	7
4.	Miscellaneous .....	7
5.	About the current Version.....	9
5.1.	Additional Notes .....	9
6.	Support.....	9

# 1. Introduction

FraunhoferFS (short: FhGFS) is a high-performance parallel file system for Linux. It has been developed from scratch by the Fraunhofer Competence Center for High Performance Computing, which is part of the international Fraunhofer Gesellschaft (FhG). By combining latest research results and optimizations for HPC technologies, the file system has been designed to provide a scalable, flexible, and robust parallel architecture.

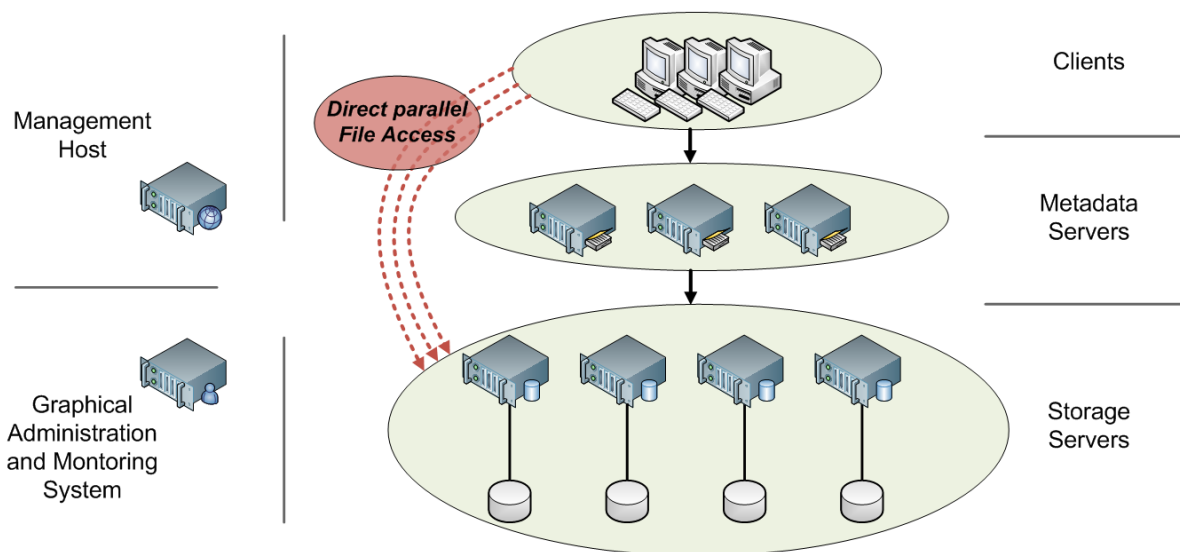
## 1.1. Key Benefits

- **Distributed file contents and metadata:** One of the most fundamental concepts of FhGFS is the strict avoidance of architectural bottle necks. Striping file contents across multiple storage servers is only one part of this concept.  
Another important aspect is the distribution of file system metadata (e.g. directory information) across multiple metadata servers. Large systems and metadata intensive applications in general can greatly profit from the latter feature.
- **HPC technologies:** Built on scalable multithreaded core components with native Infiniband support, file system nodes can serve Infiniband and Ethernet (or any other TCP-enabled network) connections at the same time and automatically switch to a redundant connection path in case any of them fails.
- **Easy to use:** FhGFS requires no kernel patches (the client is based on pre-built kernel modules), comes with cluster installation tools and allows you to add more clients and servers to the running system whenever you want it.
- **Client and servers on any machine:** No specific enterprise Linux distribution or other specific environment is required to run FhGFS. FhGFS client and servers can even run on the same machine to enable performance increases for small clusters or networks.  
FhGFS requires no dedicated file system partition on the servers - It uses existing partitions, formatted with any of the standard Linux file systems, e.g. XFS or ext4. For larger networks, it is also possible to create several distinct FhGFS file system partitions with different configurations.
- **Increased coherency (compared to simple remote file systems like NFS):** FhGFS provides a coherent mode, in which it is guaranteed that changes to a file or directory by one client are always immediately visible to other clients.

## 1.2. System Architecture

FhGFS combines multiple storage servers to provide a shared network storage resource with striped file contents. This way, the system overcomes the tight performance limitations of single servers, single network interconnects, a limited number of hard drives etc. In such a system, high throughput demands of large numbers of clients can easily be satisfied, but even a single client can profit from the aggregated performance of all the storage nodes in the system.

This is made possible by a separation of metadata and file contents. FhGFS clients have direct access to the storage servers and communicate with multiple servers simultaneously, giving your applications truly parallel access to the file data. To keep the metadata access latency (e.g. directory lookups) at a minimum, FhGFS allows you to also distribute the metadata across multiple servers. The following picture shows the system architecture of FhGFS. Note that although all roles in the picture are running on different nodes, it is also possible to run any combination of client and servers on the same machine.



Besides the three basic roles in FhGFS (clients, metadata servers, storage servers) there are two additional system services, that are part of FhGFS. The first one is the management daemon, that serves as a central point of information for client and node groups.

The second one is the administration and monitoring service (Admon), that provides a graphical web-browser frontend for installation and permanent system status monitoring.

## 2. Installation and Setup

FhGFS can be installed either automatically via a graphical Java interface or manually. A description of the installation and setup procedure can be found in the separate FhGFS Installation Guide:

[http://www.fhgfs.com/docs/FraunhoferFS\\_Installation\\_Guide.pdf](http://www.fhgfs.com/docs/FraunhoferFS_Installation_Guide.pdf)

## 3. Tuning and advanced Configuration

Many of the file system options can be set by editing the corresponding configuration files (`/etc/fhgfs/fhgfs_xxxx.conf`). If you haven't done so yet, you might want to take a look at these files first to find out which options you have in general.

If you have chosen the automatic installation process based on the Admon GUI, you typically won't edit the configuration files on the file system nodes directly, but instead change the settings via the GUI or modify the template files in `/opt/fhgfs` on the Admon node and then let the Admon distribute the configuration to the nodes.

The runtime configuration can be queried and changed with the FhGFS online configuration tool (located at `/opt/fhgfs/sbin/fhgfs_online_cfg` on the client nodes). On a client with an active FhGFS mount, several configuration options can also be queried and changed by accessing the `procs` entries (located at `/proc/fs/fhgfs`).

### 3.1. Native Infiniband Support

Native Infiniband support in FhGFS is based on the Open Fabrics Enterprise Distribution `ibverbs` API (<http://www.openib.org>). To enable native Infiniband support, the FhGFS Client OpenTk kernel module (`fhgfs_client_opentk.ko`) and the FhGFS OpenTk shared library (`libfhgfs_opentk.so`) have to be built for your specific OFED version. At OFED version 1.2 or higher is required to compile the kernel module and the library. The build process for both, kernel module and library, is part of the general Admon-based installation process.

At runtime, you can check whether your IB devices have been discovered by using the FhGFS online configuration tool. The tool mode `"LISTNODES"` will output a list of all running nodes and their configured network interfaces. The word `"RDMA"` will be appended to interfaces that are enabled for the native Infiniband protocol.

To check whether the clients are connecting to the servers via RDMA or whether they are falling back to TCP because of configuration problems, you can access the `proc` file system on a mounted client.

The `/proc/fs/fhgfs/<...>/xxxx_nodes` entries provide a list of the currently active connections of a client.

### ***3.2. Caching***

Caching is one of the major performance aspects of shared network resources, because it can significantly reduce the network traffic and the latency of operations. FhGFS has been designed to always provide a coherent system view for all the clients. However, the cache coherence protocol is still being adjusted and not included in the current release.

The clients currently use the non-coherent cache protocol by default. To disable caching and guarantee a coherent view for all the clients, set the option “`tuneCacheEnabled`” to `false` in the client config file (`/etc/fhgfs/fhgfs_client.conf`).

### ***3.3. Striping***

Striping in FhGFS can be configured on a per-directory and per-file basis. Each directory has a specific stripe pattern configuration, which will be derived to new subdirectories and applied to any file created inside a directory. There are currently two parameters that can be configured for the standard RAID-0 stripe pattern: the desired number of storage nodes for each file and the chunk size (or block size) for each file stripe.

There are two ways of configuring the stripe pattern parameters: A static configuration can be made in the configuration files of the metadata nodes (`/etc/fhgfs/fhgfs_meta.conf`). The options “`tuneDefaultChunksize`” and “`tuneDefaultNumStripeNodes`” will be applied to the root directory of the file system during the very first system startup. These settings are static, so they have no effect during later startups.

The second way of configuring the stripe pattern parameters is the FhGFS online configuration tool. It allows you to view or change the stripe pattern details of each file or directory in the file system while the file system is online.

### ***3.4. Storage Tuning***

FhGFS uses a pre-formatted partition to store its data. Since it only makes use of the very basic local file system operations on the servers, you are free to choose whatever standard Linux file system you prefer. Of course, each of the local file systems performs better or worse than the others in certain situations. In general, XFS is recommended as the underlying local file system on FhGFS servers, because it typically provides significantly better scalability and throughput results, especially for RAID systems.

For low-level I/O tuning, the Linux kernel 2.6 series comes with several I/O schedulers (aka elevators). The typical two choices here are the deadline scheduler and the anticipatory scheduler. Effectiveness of the schedulers varies for different setups and workloads, but in general setting the deadline scheduler for your metadata server devices and the anticipatory scheduler for your storage server devices is a good starting point. You can view or change the current scheduler by writing its name to the virtual file `/sys/block/<block_device>/queue/scheduler`. Enlarging the block queue size (`/sys/block/<block_device>/queue/nr_requests`) can also increase throughput and the effectiveness of the I/O scheduler, especially in high-concurrency situations. Enlarging the read-ahead size typically improves performance for sequential file access (`<...>/queue/read_ahead_kb`).

Another way of optimizing the storage behavior is to adjust the operating system memory cache of the servers. You might want to experiment with the files in `/proc/sys/vm` for this purpose.

### 3.5. Network Tuning

Network tuning can be applied to clients and servers, and is often based on adjustments to the common TCP settings. The most important settings here include the TCP window scaling and buffer size. These settings can be changed by writing to the files in `/proc/sys/net/ipv4`. Your distribution might also come with different TCP implementations, which provide significantly better optimizations for high-speed networks than the standard TCP implementation.

## 4. Miscellaneous

- **Node discovery:** FhGFS is designed to connect clients and storage nodes automatically by registering the corresponding nodes and their services at the FhGFS management daemon.
- **Communication retries:** FhGFS clients support an infinite retry mode for communications, which will make them keep on trying in case a server is temporarily unreachable. This behavior can be turned on or off by setting the “`connNumRetries`” option in the client configuration file. If the infinite retries mode is turned off, client I/O operations will fail with an error if a server is unreachable.  
Infinite retries can be stopped by interrupting the process waiting for I/O (e.g. by pressing CTRL+C on a console). Another way of interrupting infinite retries for all waiting processes is by writing “false” to the `procfs` entry `/proc/fs/fhgfs/<...>/conn_retries_enabled`.
- **Network interfaces:** By default, FhGFS clients and servers try to connect via any available network interface that supports TCP, SDP or OFED ibverbs (preferring ibverbs and SDP-enabled interfaces) and switch to a different one if the primary network fails. If you do not

want to allow this behavior, the config options “`connInterfacesFilename`” and “`connNetFilterFile`” can be used to allow only specific interfaces or to connect only to specific IP address ranges. By listing an allowed interface higher than the others in the interfaces file, you can also assign priority to specific interfaces. (E.g. if you want to prefer `eth1` over `eth0`, write `eth1` in the first line and `eth0` in the second line).

- **Preferred storage nodes:** Clients can prefer storage nodes if they, for instance, have a faster connection to them. These preferred nodes can be specified with the “`tunePreferredNodesFilename`” parameter in the client configuration file (`/etc/fhgfs/fhgfs_client.conf`). Only preferred nodes will be used to store new files.
- **Root node:** The current version of FhGFS relies on the (usually automatic) choice of a special root metadata node, which serves as an initial coordinator and as a guide to the system for clients. (The online configuration tool can be used to find out which node has been chosen to be the root node). The management daemon saves the root node ID when a root node was chosen, so the election of the root node only happens during the very first startup of the system.

Later FhGFS releases will optionally add redundancy for any kind of metadata or file data in the system to keep the system running even when the root node (or any other node) is down.

In case you do want the root node to be chosen automatically, you can use the “`sysForcedRoot`” and “`sysOverrideStoredRoot`” options in the configuration files to set the root node manually.

- **Storage directories:** FhGFS separates the metadata from the file contents. Hence, there are two major directories that can be configured. The configuration options “`storeMetaDirectory`” and “`storeStorageDirectory`” can be used to set the metadata directory and the directory for the striped file contents. Both directories may be on the same local file system partition, but do not specify the same directory to store metadata and file stripes.

## 5. About the current Version

FhGFS allows you to create a shared network file system that combines multiple storage servers to a large global namespace. The file system automatically stripes files across the servers and supports parallel access to the files for clients. This way, even a single client can profit from the aggregated performance of multiple storage servers, while multiple simultaneous clients profit from the distributed server load.

Clients mounting the file system can perform the typical file operations and users immediately see the performance increase that comes with the parallel access and, depending on the hardware, the other high-performance optimizations, e.g. the native Infiniband support. Clients can be configured to have a coherent view of the system, which means that any file operations performed by one client are always immediately visible by other clients.

In addition to the basic file and directory operations, memory mapping of files and symlinks are currently supported.

### 5.1. Additional Notes

Currently, FhGFS is designed primarily as a fast scratch file system for clusters. Hence, special backup and high availability features have not been implemented yet. However, these features have been planned since the beginning of FhGFS development and will be added to the system in a future release.

Besides that, some more advanced file access features like hardlinks or distributed user file locking are not enabled yet (`flock()` is a local operation, currently).

We are continuously extending the features of FhGFS and will provide new features on the website as soon as they are ready for release.

## 6. Support

FhGFS is currently available free of charge as a binary package. If you have any trouble installing or running the software, please send an e-mail to [support@fhgfs.com](mailto:support@fhgfs.com). Of course, you are also welcome to send any other kind of question, recommendations or benchmark results to this e-mail address.

Commercial support for FhGFS is also available. Support fees are based on the number of storage servers (clients are free). Contact [support@fhgfs.com](mailto:support@fhgfs.com) for additional information.