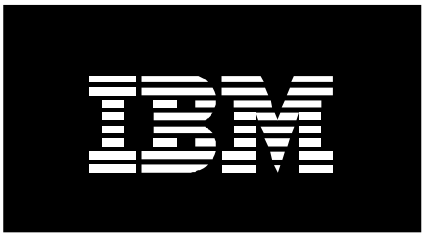

# Disaster Recovery with General Parallel File System

*August 2004*

# Abstract

*The ability to detect and quickly recover from a massive-scale hardware failure is of paramount importance to businesses that make use of real-time data processing systems. General Parallel File System (GPFS) provides a number of features that facilitate the implementation of highly-available GPFS environments capable of withstanding catastrophic hardware failures. By maintaining a redundant replica of the file system's data at another (geographically separated) location, we allow the system to sustain its processing using the secondary replica of the data in the event of a total failure in the prime environment. In this paper, we present an overview of the disaster recovery features available in the 2.2 release of GPFS and provide detailed hands-on guidance on implementing the various types of disaster-tolerant configurations supported in this release.*


# Overview and terminology

Businesses that depend on real-time data processing systems are vulnerable to a profound negative impact from natural or unnatural disasters such as fires, tornadoes, earthquakes, or power failures. A catastrophe can permanently disable the customer's data processing infrastructure and, without proper backup procedures, result in a permanent loss of business-critical data. To help minimize the impact from such unplanned hardware outages, many businesses are implementing preventive measures enabling them to quickly recover from a disastrous failure and enable the near continual availability of mission-critical applications and the associated data. GPFS 2.2 provides you with a number of features for the support of your disaster recovery solution.

On a very high level, a disaster-resilient GPFS cluster environment is typically made up of two (sometimes three) distinct hardware *sites* that operate in a coordinated fashion and are separated by a substantial geographic distance. Each site houses some number of GPFS nodes and a storage resource holding a complete replica of the file system. In the event of a catastrophic hardware failure that disables the operation of an entire site, GPFS is designed to failover to the remaining subset of the cluster and continue serving the data using the replica of the file system that survived the disaster. In this paper, we examine several methods for maintaining the secondary replica, which include synchronous mirroring (through the use of IBM TotalStorage® Enterprise Storage Server® (ESS) Peer-to-Peer Remote Copy (PPRC) or logical GPFS replication) along with a non-synchronous approach that utilizes ESS FlashCopy.

Certain high-end storage subsystems such as ESS implement support for volume-level geographic mirroring of data. For example, the PPRC feature of the ESS enables users to establish a persistent mirroring relationship between pairs of Logical Units (LUNs) on two subsystems connected over an ESCON or a fiber-channel link. All updates performed on the set of *primary* (or *source*) LUNs appear in the same order on the *secondary* (*target*) disks in the target subsystem. The PPRC mechanism provides that if the source volume fails, the target holds an exact bitwise replica of the source's content as seen at the time of the failure. This solution is described in the section *Active/passive GPFS configurations with ESS Peer-to-Peer Remote Copy*

The existing data and metadata replication features of GPFS can be similarly used to implement synchronous mirroring between a pair of geographically separated sites. The usage of logical replication-based mirroring can be seen as an attractive alternative to PPRC, in particular because it offers a generic solution that relies on no specific support from the disk subsystem

beyond the basic ability to read and write data blocks. This solution is described in the section *Active/active GPFS configurations employing logical replication and quorum tiebreakers*

The primary advantage of both synchronous mirroring methods lies in the minimization of the risk of permanent data loss. Both methods provide us with two consistent up-to-date replicas of the file system, each available for recovery should the other one fail. However, inherent to all solutions that synchronously mirror data over a wide-area network link is the latency penalty necessarily induced by the replicated write I/Os. This makes both mirroring methods prohibitively inefficient for certain types of performance-oriented applications.

An alternative technique involves taking periodic point-in-time copies of the file system using a facility such as the ESS FlashCopy®. The copy is subsequently transferred to a remote back-up location using PPRC and/or written to tape. The key difference between this and the synchronous mirroring techniques is that the creation of the secondary replica takes place asynchronously with respect to the regular file system activity against the primary replica, effectively eliminating the write penalty associated with synchronous mirroring. This solution is described in the section *Online backup with ESS FlashCopy*

This paper is written for Information Technology professionals who have some experience with GPFS, ESS PPRC, and FlashCopy. For more information on these technologies, please refer to the sources listed in the References section. The reader is assumed to have some familiarity with the standard administrative concepts of GPFS and understand the node quorum rules enforced by GPFS to help protect the integrity of on-disk data in the event of a network partition.

# Active/active GPFS configurations employing logical replication and quorum tiebreakers

## Overview of active/active dispersed GPFS clusters

In an active/active GPFS configuration, a single GPFS cluster is defined over two geographically separated physical clusters, or *sites*. One or more file systems are created; each to be mounted and accessed concurrently from both locations. The *data* and *metadata replication* features of GPFS are used to maintain a secondary copy of each file system block, relying on the concept of *disk failure groups* to control the physical placement of the individual copies. The high-level organization of a replicated active/active GPFS cluster is shown in Figure 1.

We partition the set of available disk volumes into two failure groups (one defined for each site) and create a standard replicated file system (specifying the replication factor of two for both data and metadata). When allocating new file system blocks, GPFS always assigns replicas of the same block to distinct failure groups. This allows for a sufficient level of redundancy allowing each site to continue operating independently should the other site fail.

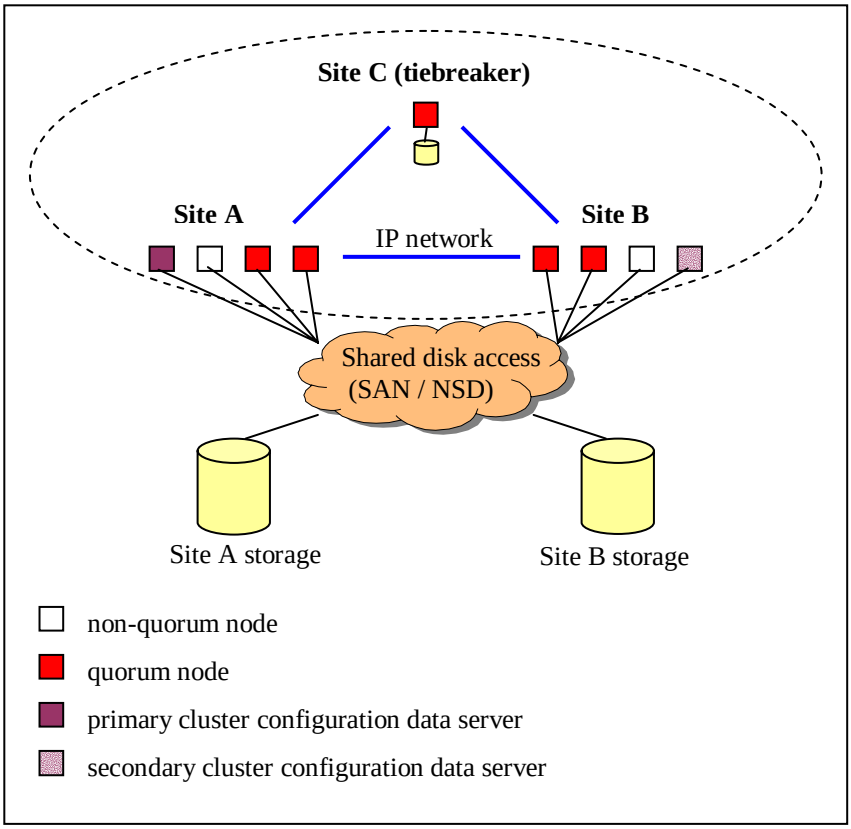


**Figure 1. High-level organization of a replicated geographically dispersed GPFS cluster**

As described in *Concepts, Planning and Installation Guide*, GPFS enforces a node quorum rule to prevent multiple nodes from assuming the role of the file system manager in the event of a network partition. Thus, a majority of quorum nodes must remain active in order for the cluster to sustain normal file system usage (*multi-node* quorum). Furthermore, GPFS uses a quorum replication algorithm to maintain the content of the file system descriptor (one of the central elements of the GPFS metadata). When formatting the file system, GPFS assigns some number of disks (usually three) as the *descriptor replica holders* that are responsible for maintaining an up-to-date copy of the descriptor. Similar to the node quorum requirement, a majority of the replica holder disks must remain available at all times to sustain normal file system operations.

Considering the above quorum constraints, we suggest introducing a third site into the configuration to fulfill the role of a tiebreaker for the node and the file system descriptor quorum decisions. Typically, the tiebreaker site would accommodate a single quorum node and a single GPFS disk (we suggest using a logical volume built on top of the tiebreaker node's internal disk) with the disk usage defined as a *descriptor only* disk. The **`descOnly`** option for disk usage instructs GPFS to exclude the respective disk from the regular block allocation activity and avoid using this disk for the storage of data and metadata. This disk serves no function other than to provide an additional replica of the file system descriptor needed to sustain quorum should a disaster cripple one of the other descriptor replica disks. The tiebreaker disk is accessed over NSD with the tiebreaker node defined as its primary NSD server.

Since the tiebreaker node does not normally access the file system, direct SAN connectivity between that node and the subsystems that supply the actual storage is unnecessary. However, we must instruct GPFS to disregard all disk access errors on the tiebreaker by enabling the **`unmountOnDiskFail`** configuration parameter through the **mmchconfig** command for that node. If enabled, this parameter forces the tiebreaker node to treat the lack of disk connectivity as a local error (resulting in a failure to mount the file system) rather that reporting this condition to the file system manager as a disk failure.

The three-site configuration proposed here is resilient to a complete failure of any single hardware site. Should all disks volumes in one of the failure groups become unavailable as a result of a disaster, GPFS performs a transparent failover to the remaining set of disks and continues serving the data to the surviving subset of nodes with no administrative intervention. Note that while nothing prevents users from placing the tiebreaker resources at one of the active sites, we suggest installing the tiebreakers at a third geographically distinct location to minimize the risk of double-site failures.

## Setting up an active/active dispersed GPFS cluster

This is a step-by-step example illustrating the creation of an active/active geographically dispersed GPFS cluster and a single replicated file system. We assume the following configuration:

**Site A**
- Node hostnames: `nodeA001, nodeA002, nodeA003, nodeA004`
- Disk volume names: `hdisk11, hdisk12, hdisk13, hdisk14`

**Site B**
- Node hostnames: `nodeB001, nodeB002, nodeB003, nodeB004`
- Disk volume names: `hdisk15, hdisk16, hdisk17, hdisk18`

**Site C (tiebreaker)**
- Node hostnames: `nodeTiebr`
- Disk volume names: `tielv`

Disks `hdisks11-hdisk18` are SAN-attached and accessible from all nodes at sites A and B. `tielv` is a logical volume defined over `nodeTiebr`'s internal disk.

1. Create a GPFS cluster of type **lc** selecting two nodes, one at each site, as the primary and seconday cluster data server nodes:
   ```
   mmcrcluster –t lc –n NodeDescFile –p nodeA001 –s nodeB001
   ```

2. Create and configure the GPFS nodeset:
   ```
   mmconfig –a
   ```

3. Disable the GPFS `autoload` option (see below):
   ```
   mmchconfig autoload=no
   ```

4. Define the set of network shared disks for the cluster where disks at sites A and B are assigned to failure groups 1 and 2, respectively. The tiebreaker disk is assigned to failure group 3:
   `mmcrnsd –F DiskDescFile`

5. Enable `unmountOnDiskFail` on the tiebreaker to prevent this node from falsely declaring the disks as having failed:
   `mmchconfig unmountOnDiskFail=yes nodeTiebr`

6. Start the GPFS daemon on all nodes:
   `mmstartup -a`

7. Create a replicated file system and mount it on all nodes in the cluster:
   `mmcrfs /gpfs/fs0 fs0 –F DiskDescFile –m 2 –M 2 –r 2 –R 2`
   `mount /gpfs/fs0`

```
nodeA001:nonquorum-client
nodeA002:nonquorum-client
nodeA003:nonquorum-client
nodeA004:quorum-manager
nodeB001:nonquorum-client
nodeB002:nonquorum-client
nodeB003:nonquorum-client
nodeB004:quorum-manager
nodeTiebr:quorum-client
```

**Sample NodeDescFile**

```
#/dev/hdisk11:::dataAndMetadata:1
gpfs1nsd:::dataAndMetadata:1
#/dev/hdisk12:::dataAndMetadata:1
gpfs2nsd:::dataAndMetadata:1
#/dev/hdisk13:::dataAndMetadata:1
gpfs3nsd:::dataAndMetadata:1
#/dev/hdisk14:::dataAndMetadata:1
gpfs4nsd:::dataAndMetadata:1
#/dev/hdisk15:::dataAndMetadata:2
gpfs5nsd:::dataAndMetadata:2
#/dev/hdisk16:::dataAndMetadata:2
gpfs6nsd:::dataAndMetadata:2
#/dev/hdisk17:::dataAndMetadata:2
gpfs7nsd:::dataAndMetadata:2
#/dev/hdisk18:::dataAndMetadata:2
gpfs8nsd:::dataAndMetadata:2
#/dev/tielv:nodeTiebr::descOnly:3
gpfs9nsd:::descOnly:3
```

**Sample DiskDescFile**

## Steps to take after a disaster

The procedure provided in *Setting up an active/active dispersed GPFS cluster* creates a disaster-resilient GPFS file system that masks failures of a single geographic site (assuming that the other site and the tiebreaker remain operational). That is, if a catastrophic failure disables site A, GPFS detects this failure, mark site A's disks as *down*, and continue serving the data using the replica located at site B. The failover to the surviving replica is performed automatically and requires no

administrative intervention although, depending on the specifics of the user application, manual steps may need to be taken to transfer the application's activity to the nodes at the surviving site.

Later, depending on the nature of the disaster, we may want to permanently remove the failed resources from the GPFS configuration or, if the outage is of a temporary nature, preserve the original configuration and simply restart GPFS on the affected nodes after the repair. In the case of permanent loss of hardware assets, execute the following procedure to remove all references to the failed nodes and disks from the GPFS configuration:

(Note: We assume that Site A has become unavailable while B and the tiebreaker survived the disaster).

1. As the primary cluster data server node, `nodeA001` has become unavailable. Migrate the server to a node at site B:

   ```
   mmchcluster –p nodeB002
   ```

   (Alternatively, if the secondary cluster data server node has become unavailable as a result of a disaster, run **mmchcluster –s** to migrate the secondary server to the surviving site).

2. Delete the failed disks from the GPFS configuration:

   ```
   mmdeldisk fs0 “gpfs1nsd;gpfs2nsd;gpfs3nsd;gpfs4nsd”
   mmdelnsd “gpfs1nsd;gpfs2nsd;gpfs3nsd;gpfs4nsd”
   ```

3. Shut down GPFS on all surviving nodes:

   ```
   mmshutdown –a
   ```

4. Delete the failed nodes from the GPFS configuration:

   ```
   mmdelnode nodeA001,nodeA002,nodeA003,nodeA004
   mmdelnode –c
   mmdelcluster nodeA001,nodeA002,nodeA003,nodeA004
   ```

5. Start the GPFS daemon on the remaining nodes and remount the file system on all nodes:

   ```
   mmstartup -a
   mount /gpfs/fs0
   ```

On the other hand, if the nature of the disaster is such that the resources at the failed site are left physically intact but temporarily inaccessible (for example, as a result of a power outage), no changes to the GPFS configuration are needed at this stage. The GPFS cluster enters the *failover* state, in which it will remain until a decision is made to restore the operation of the affected site by executing the *failback* procedure as described in the section *Transition to the active/active state after the repair*

Please note: We suggest users not to make any changes to the GPFS configuration (with commands like **`mmchconfig`**, **`mmchfs`**, etc.) while the cluster is in the failover state. These commands require both cluster configuration servers to be operational and while the servers can be migrated to the surviving site, it is best to avoid this step if the disaster does not leave the affected site permanently disabled. The reason is that the migration and the subsequent changes to the configuration would leave the two sites with distinct (and possibly inconsistent) copies of the GPFS configuration data file (**mmsdrfs**).

If it becomes absolutely necessary to modify the GPFS configuration while in failover mode, please ensure that all nodes at the affected site are left in a stable inactive state (e.g. powered down) and will remain in such state until the decision is made to execute the failback procedure. As a means of precaution, we suggest disabling the GPFS `autoload` option on all nodes to prevent GPFS from bringing itself up automatically on the affected nodes should they come up spontaneously at some point after a disaster.

During failback, please make sure to run the **`mmchcluster –p LATEST`** command to resynchronize the **mmsdrfs** data across the entire cluster before restarting the GPFS daemon in the disaster-affected site.

## Transition to the active/active state after the repair

Once the physical operation of site A has been restored, an administrative procedure is executed to resume file system access at the recovered site and rebalance the replicated file system data across the two locations. The following two cases must be considered:

If the disaster resulted in a permanent loss of hardware resources (nodes and/or disks) and if these resources have subsequently been replaced, the procedure would involve defining these new resources as part of the GPFS cluster. In the following example, we are augmenting the existing GPFS configuration with 4 new nodes (`nodeA005-nodeA008`) and 4 disks (`hdisk11-hdisk14`) located at site A:

1. Add the new nodes to the GPFS cluster and the nodeset:

   ```
   mmaddcluster nodeA005:nonquorum-client,nodeA006:nonquorum-client,
   nodeA007:nonquorum-client,nodeA008:quorum-manager

   mmaddnode nodeA005,nodeA006,nodeA007,nodeA008
   ```

2. Migrate the primary cluster data server back to site A:

   ```
   mmchcluster –p nodeA005
   ```

   (Alternatively, if the secondary cluster data server was migrated after a disaster, run **mmchcluster –s** to migrate the secondary server back to the recovered site).

3. Start GPFS on the newly added nodes and mount the file system:

   ```
   mmstartup –w nodeA005,nodeA006,nodeA007,nodeA008
   ```

`mount /gpfs/fs0` (on nodes `nodeA005,nodeA006,nodeA007, nodeA008`)

4. Create the new NSDs:

   ```
   mmcrnsd –F NewDiskDescFile
   ```

5. Add the disks to the GPFS file system and rebalance it to restore the initial replication properties:

   ```
   mmadddisk fs0 –F NewDiskDescFile –r  -a
   ```

   This command will perform a full scan of the file system's metadata and rebalance the files to make use of the new disks. This is a potentially time-consuming task and we suggest running the above command with the **–a** option to move the rebalancing operation into asynchronous mode. For the same reason, we recommend executing this command only once, specifying all new disks in the *NewDiskDescFile*, as opposed to running it once for every new disk.

```
#/dev/hdisk11:::dataAndMetadata:1
gpfs10nsd:::dataAndMetadata:1
#/dev/hdisk12:::dataAndMetadata:1
gpfs11nsd:::dataAndMetadata:1
#/dev/hdisk13:::dataAndMetadata:1
gpfs12nsd:::dataAndMetadata:1
#/dev/hdisk14:::dataAndMetadata:1
gpfs13nsd:::dataAndMetadata:1
```

**Sample NewDiskDescFile describing the disks at site A**

On the other hand, if the GPFS cluster is in the failover state (with the affected resources still part of the configuration), the *failback* procedure is executed to resume the operation of GPFS across the entire cluster. The procedure is as follows:

1. Ensure that all nodes have the latest copy of the **mmsdrfs** file:

   ```
   mmchcluster –p LATEST
   ```

2. Start GPFS on all nodes at site A and mount the file system on these nodes:

   ```
   mmstartup –w nodeA001,nodeA002,nodeA003,nodeA004
   ```
   `mount /gpfs/fs0` (on nodes `nodeA001,nodeA002,nodeA003, nodeA004`)

3a. If the data on disks at Site A is intact (meaning that no unexpected modifications to the content of these disks have taken place during or after the disaster), bring the disks on-line:

   ```
   mmchdisk fs0 start –a
   ```

   This command also restripes the file system to restore the initial replication properties.

3b. Otherwise, delete the affected disks from the file system, recreate the NSDs, and add the disks back:

   ```
   mmdeldisk fs0 "gpfs1nsd;gpfs2nsd;gpfs3nsd;gpfs4nsd"
   ```

```
mmdelnsd “gpfs1nsd;gpfs2nsd;gpfs3nsd;gpfs4nsd”
mmcrnsd –F NewDiskDescFile
mmadddisk fs0 –F NewDiskDescFile –r -a
```

The **mmadddisk** command with the **–r** option also restripes the file system to restore the initial replication properties.

# Active/passive GPFS configurations with ESS Peer-to-Peer Remote Copy

## Overview of active/passive PPRC environments

In an active/passive environment, two GPFS clusters are set up in two geographically distinct locations (the *production* and the *recovery sites).* We refer to these as *peer GPFS clusters*. A GPFS file system is defined over a set of disk volumes located at the production site and these disks are mirrored using PPRC to a secondary set of volumes located at the recovery site. During normal operation, only the nodes in the production GPFS cluster would mount and access the main replica of the file system residing on the production site’s storage resource.

In the event of a catastrophe in the production cluster, the *PPRC failover* task is executed to enable access to the secondary replica of the file system located on the target PPRC disks. (Please refer to the *IBM ESS Redbook* for a detailed explanation of *PPRC failover* and *failback* and the means of invoking these procedures in your environment).

The secondary replica is then mounted on nodes in the recovery cluster as a regular GPFS file system, thus allowing the processing of data to resume at the recovery site. At a latter point, after restoring the physical operation of the production site, we execute the *failback* procedure to resynchronize the content of the PPRC volume pairs between the two clusters and re-enable access to the file system in the production environment.

A schematic view of an active/passive GPFS environment with PPRC is shown in Figure 2. Note that in such environments, only one of the sites accesses the GPFS file system at any given time, which is one of the primary differences between a configuration of this type and the active/active model described in the previous section.

After defining the set of NSDs and creating the GPFS file system in the primary cluster, we will use the **`mmfsctl syncFSconfig`** command to import the file system’s definition into the peer recovery cluster. This command extracts the relevant configuration data from the local **mmsdrfs** file, contacts one of the nodes in the peer recovery cluster, and imports the extracted data into that cluster’s **mmsdrfs** file to make the file system *known* to the recovery nodes. The **`mmfsctl`** command is available for GPFS beginning with APAR IY59339 .

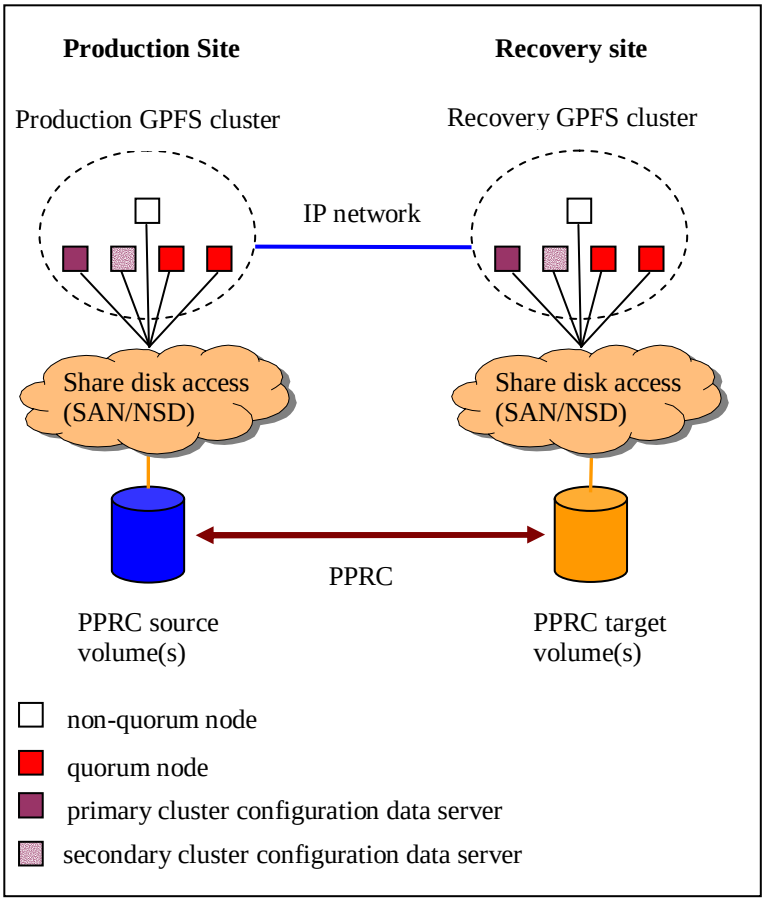


**Figure 2. An active/passive disaster-tolerant GPFS environment using PPRCmirroring**

**Syntax**

mmfsctl {*Device* | all} syncFSconfig *-n PeerNodesFile* [-S *PeerChangeSpecFile*]

**Purpose**

Synchronizes the configuration state of a GPFS file system between the local cluster and its peer

**Parameters**

| | |
|---|---|
| *Device* or `all` | The device name of the GPFS file system whose configuration is being resynchronized. If the **all** option is used, this operation is performed on all GPFS file systems defined in the local cluster. |
| `-n` *PeerNodeFile* | The list of contact nodes (one hostname per line) for the peer cluster. For efficiency reasons, we suggest specifying the identities of the peer cluster's primary and secondary configuration data servers. If none of the contact nodes are reachable, the command will report an error and no changes will take place. |
| `-S` *PeerChangeSpecFile* | The description of changes to be made to the file system in the peer cluster during the import step. The format of this file is identical to that of the *ChangeSpecFile* used as input to the **`mmimportfs`** command. This option can be used, for example, to define the assignment of the NSD servers for use in the peer cluster. |

**Comments**

The primary configuration data server of the peer cluster must be available and accessible via remote shell and remote copy at the time of the execution of this command. Additionally, the peer GPFS clusters should be defined to use identical remote shell and remote copy mechanisms (ssh/scp or rsh/rcp) and these mechanisms should be set up to allow nodes in peer clusters to communicate without the use of a password.

At all times, the peer clusters must see a consistent image of the mirrored file system's configuration state contained in the **mmsdrfs** file. You must ensure that after the initial creation of the file system, all subsequent updates to the local configuration data are propagated and imported into the peer cluster. The **`mmfsctl syncFSconfig`** command should be executed to resynchronize the configuration state between the peer clusters after each of the following actions in the primary GPFS cluster:

1) Additions, removals, and replacements of disks (**`mmadddisk`** / **`mmdeldisk`** / **`mmrpldisk`**)
2) Modifications to disk attributes (**`mmchdisk`**)
3) Changes to the file system's mountpoint (**`mmchfs -T`**)

Furthermore, we suggest users run the **mmfsctl `syncFSconfig`** command as part of the failback procedure to propagate any changes made in failover state back into the primary cluster's **mmsdrfs** file.

The **`/var/mmfs/etc/syncfsconfig`** user exit provides a means of automating this resynchronization step. This exit is invoked after the initial creation of the file system in the local cluster, as well as after the execution of all administrative commands that modify the file system's configuration state.

**Syntax**

`/var/mmfs/etc/syncfsconfig` *Device*

**Purpose**

Invoked after every administrative command that modifies the state of the file system in **mmsdrfs** on the node from which the command was issued. In disaster recovery environments, this exit is typically used for the propagation of administrative changes to the peer recovery cluster.

**Parameters**

| | |
|---|---|
| Device | The device name of the GPFS file system whose configuration has been modified. |

Unless some form of specialized administrative processing is implemented in your GPFS environment, we suggest defining this user exit to call **`mmfsctl syncFSconfig`**, for instance as follows:

```
mmfsctl $1 syncFSconfig –n recoveryNodeFile –S recoverySpecFile
```

**Sample contents of `/var/mmfs/etc/syncfsconfig`**

## Setting up an active/passive PPRC-mirrored GPFS environment

This example describes the creation of an active/passive GPFS environment with PPRC mirroring. We assume the following configuration:

| **Production site** | |
|---|---|
| Node hostnames: | `nodeP001, nodeP002, nodeP003, nodeP004` |
| Storage subsystems: | `ESS P`; logical subsystem `LSS P` |
| LUN ids and disk volume names: | `lunP1 (hdisk11), lunP2 (hdisk12), lunP3 (hdisk13), lunP4 (hdisk14)` |

| **Recovery site** | |
|---|---|
| Node hostnames: | `nodeR001, nodeR002, nodeR003, nodeR004` |
| Storage subsystems: | `ESS R`; logical subsystem `LSS R` |
| LUN ids and disk volume names: | `lunR1 (hdisk11), lunR2 (hdisk12)`<br>`lunR3 (hdisk13), lunR4 (hdisk14)` |

All disks are SAN-attached and are directly accessible from all local nodes.

1. Create a dual-active *ESS copy services domain*; assign `ESS P` as ServerA and `ESS R` as ServerB.

2. Establish a PPRC logical path from `LSS P` to `LSS R`. Enable the *consistency group* option. See the section *Data integrity and the use of PPRC consistency groups*.

3. Establish synchronous PPRC volume pairs:

   `lunP1-lunR1` (source-target)
   `lunP2-lunR2` (source-target)
   `lunP3-lunR3` (source-target)
   `lunP4-lunR4` (source-target)

   Use the *copy entire volume* option and leave the *permit read from secondary* option disabled.

At the recovery site:

4. Define the recovery GPFS cluster and the nodeset:

```
mmcrcluster –t lc –n NodeDescFileR –p nodeR001 –s nodeR002
mmconfig –a
```

At the production site:

5. Define the production GPFS cluster and the nodeset:

```
mmcrcluster –t lc –n NodeDescFileP –p nodeP001 –s nodeP002
mmconfig –a
```

6. Automate the propagation of the configuration state to the recovery cluster by invoking the **`mmfsctl syncFSconfig`** command from the **`syncfsconfig`** user exit:

```
echo ‘mmfsctl $1 syncFSconfig –n NodeDescFileR’ >
    /var/mmfs/etc/syncfsconfig
chmod 744 /var/mmfs/etc/syncfsconfig
```

7. Start the GPFS daemon on all nodes in the production cluster:

```
mmstartup -a
```

8. Define the NSDs:

```
mmcrnsd –F DiskDescFileP
```

9. Create the GPFS file system and mount it on all nodes:

```
mmcrfs /gpfs/fs0 fs0 –F DiskDescFileP
mount /gpfs/fs0
```

```
nodeP001
nodeP002
nodeP003
nodeP004
```

**Sample NodeDescFileP**

```
nodeR001
nodeR002
nodeR003
nodeR004
```

**Sample NodeDescFileR**

```
#/dev/hdisk11
gpfs10nsd:::dataAndMetadata:-1
#/dev/hdisk12
gpfs11nsd:::dataAndMetadata:-1
#/dev/hdisk13
gpfs12nsd:::dataAndMetadata:-1
#/dev/hdisk14
gpfs13nsd:::dataAndMetadata:-1
```

**Sample DiskDescFile**

## Failover to the recovery site and subsequent failback

The following steps can restore access to the file system at the recovery site after a disastrous node and/or subsystem failure in the production location:

Failover to the recovery site

1. From a node in the primary cluster, stop the GPFS daemon on all surviving nodes in the production cluster (if not already stopped):

   ```
   mmshutdown –a
   ```

2. Perform PPRC failover. This step involves establishing synchronous PPRC pairs `lunR1-lunP1, lunR2-lunP2`, etc. with the *failover* option. After the completion of this step, the volumes on `ESS R` should enter the *suspended source* PPRC state.

3. Start the daemon on all nodes in the recovery cluster and mount the file system:

   ```
   mmstartup –a          (on a node in the recovery cluster)
   mount /gpfs/fs0       (on all nodes in the recovery cluster)
   ```

Once the physical operation of the production site has been restored, execute the failback procedure to transfer the file system activity back to the production GPFS cluster. The failback operation is a two-step process:

1. Resynchronize the paired volumes by establishing a temporary PPRC pair with `lunR*` acting as the sources for `lunP*`. The PPRC modification bitmap is traversed to find the mismatching disk tracks, whose content is then copied from `lunR*` to `lunP*`.

2. Stop GPFS on all nodes in the recovery cluster and reverse the disk roles (the original primary disks become the primaries again), effectively bringing the PPRC environment to the initial (pre-disaster) state. Additionally, if the state of the file system's configuration changed while in failover mode, issue the **`mmfsctl syncFSConfig`** command to resynchronize the content of the **mmsdrfs** file between the two clusters.

Failback to site A - resynchronization

1. Remove the existing PPRC path from `LSS P` to `LSS R`. Use the *force removal* option.

2. Establish a PPRC path from `LSS R` to `LSS P`. Enable the *consistency group* option.

3. Establish synchronous PPRC volume pairs `lunR1-lunP1`, `lunR2-lunP2`, etc. with the *failback* option. Leave the *permit read from secondary* option disabled.

4. Wait for all PPRC pairs to reach the *duplex* (fully synchronized) state.

5. If necessary, update the GPFS configuration data in the production cluster to propagate the changes made while in failover mode:

   `mmfsctl fs0 syncFSconfig –n NodeDescFileP` (on a node at the recovery site)

Failback to site A – role reversal

6. Stop the GPFS daemon on all nodes in the recovery cluster:

   `mmshutdown –a` (on a node in the recovery cluster)

7. Establish synchronous PPRC volume pairs `lunP1-lunR1`, `lunP2-lunR2`, etc. with the *PPRC failover* option.

8. Remove the existing PPRC path from `LSS R` to `LSS P`. Use the *force removal* option.

9. Establish a PPRC path from `LSS P` to `LSS R`. Enable the *consistency group* option.

10. Establish synchronous PPRC volume pairs `lunP1-lunR1`, `lunP2-lunR2`, etc. with the *PPRC failback* option. Leave the *permit read from secondary* option disabled.

11. Start the daemon on all nodes in the production cluster and mount the file system:

```
mmstartup –a          (on a node in the primary cluster)
mount /gpfs/fs0       (on all nodes in the primary cluster)
```

## Data integrity and the use of PPRC consistency groups

The integrity of the post-disaster replica of the file system (contained on the secondary PPRC disk volumes) depends on the assumption that the order of dependent write operations is preserved by the PPRC mirroring mechanism. That is, given two PPRC volume pairs P1-S1 and P2-S2, if two sequential write requests, first to P1 and later to P2 are issued by the application, we expect PPRC to commit these updates to the secondary disks in the exact same order: S1 followed by S2.

While the synchronous nature of PPRC enables such ordering during periods of stability, certain types of rolling failure scenarios require additional consideration. By default (without the *PPRC consistency group* option), if the primary ESS detects the loss of physical PPRC connectivity or the failure of one of the secondary disks, it responds by moving the corresponding primary volume(s) to the *suspended* state but continues to process the subsequent I/O requests as normal. The subsystem does not report this condition to the driver and, as a result, the application continues normal processing without any knowledge of the lost updates. GPFS relies on log recovery techniques to provide for the atomicity of updates to the file system's metadata, which is why such behavior would expose GPFS to a serious data integrity risk.

Figure 3 illustrates this problem. Consider a set-up with two primary and two secondary subsystems, each providing access to two LUNs. The four primary LUNs make up the primary replica of the file system. At some point, GPFS attempts to issue a sequence of four write requests, one to each primary disk, with the expectation that the updates appear in the exact order they were issued. (That is, if we number the writes 1 through 4, 1-2-3-4, 1-2-3, 1-2, or 1 could all be considered as consistent outcomes, but for example 1-2-4 would not be). If PPRC path 1 breaks before the start of the first write, the recovery site receives updates 3 and 4, but not necessarily 1 and 2 – a result that violates the write dependency rule and renders the target replica of the file system unusable.

The *PPRC consistency group* option determines the behavior of the primary subsystem in situations where a sudden failure of the secondary volume (or a failure of the inter-site interconnect) makes it impossible to sustain the normal synchronous mirroring process. If the PPRC path between the pair has been defined with the *consistency group* option, the primary volume enters a *long busy* state, and the subsystem reports this condition back to the host system with the **QUEUE FULL (QF) SCSI** status byte code. Typically, the driver makes several attempts to requeue the request and ultimately report the failure to the requestor. This, in turn, allows GPFS to execute the appropriate action in response to the failure (marking the disk *down* or panicking the file system).

To ensure the proper ordering of updates to the recovery copy of the file system, we urge users to always enable the *consistency group* option when configuring the PPRC paths between the peer logical subsystems.

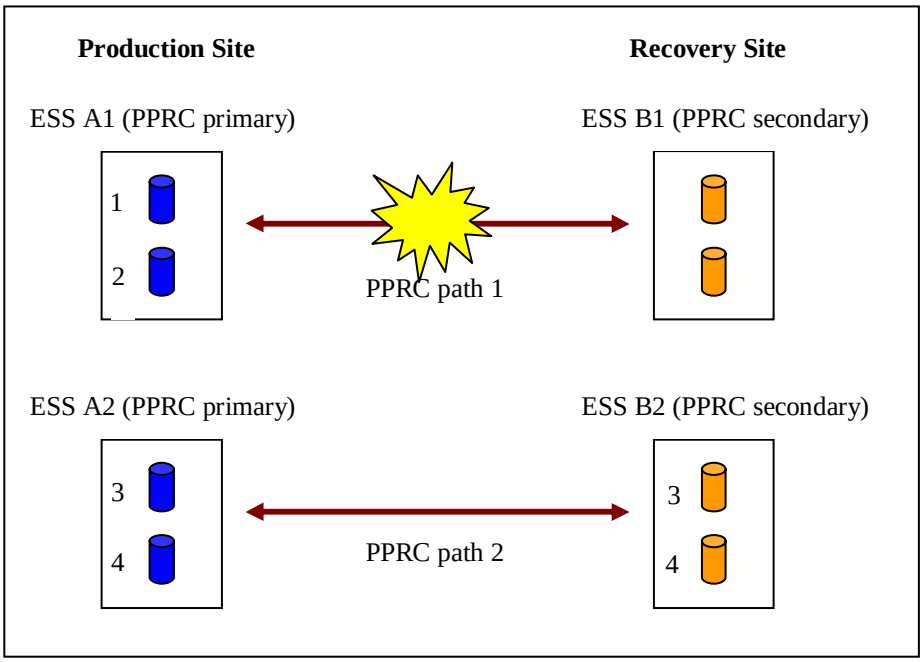


**Figure 3. Violation of write ordering without the use of a PPRC consistency group**

# Online backup with ESS FlashCopy

## Creating a FlashCopy image of a GPFS file system

The FlashCopy feature of the Enterprise Storage Server allows users to make a point-in-time copy of an ESS disk volume to another volume in the same logical subsystem. This function is designed to provide an near instantaneous (time-zero) copy or a *view* of the original data on the target disk, while the actual transfer of data takes place asynchronously and is fully transparent to the user.

The intended use of FlashCopy is to produce secondary copies of the production data quickly and with minimal application downtime. In a GPFS cluster, the FlashCopy technology can be used to produce point-in-time copies of a GPFS file system, allowing for an easy implementation of an online backup mechanism.

Similar to the PPRC mirroring scenarios, it is crucially important to consider the issues of data consistency when FlashCopy is used. To prevent the appearance of out-of-order updates, all disk volumes that make up the file system must be brought to same logical point in time prior to taking the FlashCopy image. Two methods may be used to enable data consistency in the FlashCopy image of your GPFS file system:

1. Generate a FlashCopy image using a consistency group.

With FlashCopy v2, you can use the *FlashCopy consistency group* mechanism to effectively *freeze* the source disk volume at the logical instant at which its image appears on the target disk.

When issuing the FlashCopy creation command, enable the *freeze FlashCopy consistency group* option, which will instruct the ESS to suspend all I/O activity against the source volume. From that point on, all I/O requests directed to this volume will be responded to with the **QUEUE FULL (QF) SCSI** status code. Once all disk volumes belonging to your file system have been flashed, run the *FlashCopy consistency created* task to release the consistency group and resume the normal processing of I/O. For details on the use of FlashCopy consistency groups, please refer to *IBM TotalStorage Enterprise StorageServer Implementing ESS Copy Services in Open Environments*. Note that the consistency group feature is available only with FlashCopy v2.

2. Generate a FlashCopy image using file system-level suspension.

You may temporarily quiesce the I/O activity at the GPFS file system level (as opposed to the disk volume level) through the use of the **`mmfsctl suspend`** and **`mmfsctl resume`** commands. The **`mmfsctl`** command is available for GPFS beginning with APAR IY59339. The **`mmfsctl suspend`** command instructs GPFS to flush its data buffers on all nodes, write the cached metadata structures to disk, and suspend the execution of all subsequent I/O requests. This command leaves the on-disk copy of the file system in a fully consistent state, ready to be flashed and copied onto a set of back-up disks. **`mmctlfs resume`** is the converse operation that resumes normal file system I/O after its suspension.

**Syntax**

mmfsctl Device {suspend | resume}

**Purpose**

Prior to generating a FlashCopy image of the file system, run `mmfsctl suspend` to temporarily quiesce all I/O activity and flush the GPFS buffers on all nodes that mount the file system. All subsequent requests are suspended until `mmctlfs resume` is issued to restore the normal processing.

**Parameters**

| | |
|---|---|
| Device | The device name of the GPFS file system |
| `suspend` | Instructs GPFS to flush its internal buffers on all nodes, bring the file system to a consistent state on disk, and suspend the execution of all subsequent application I/O requests. |
| `resume` | Instructs GPFS to resume the normal processing of I/O requests on all nodes. |

These examples illustrate the creation of a FlashCopy image to enable data consistency. We assume the following configuration:

| | |
|---|---|
| Storage subsystems: | `ESS 1`; logical subsystem `LSS 1` |
| LUN ids and disk volume names: | `lunS1 (hdisk11), lunS2 (hdisk12), lunT1, lunT2` |

`lunS1` and `lunS2` are the FlashCopy source volumes. These disks are SAN-connected and appear on the GPFS nodes as `hdisk11` and `hdisk12`, respectively. A single GPFS file system *fs0* has been defined over these two disks.

`lunT1` and `lunT2` are the FlashCopy target volumes. None of the GPFS nodes have direct connectivity to these disks.

Generating a FlashCopy image using a consistency group

1. Run the *establish FlashCopy pair* task with the *freeze FlashCopy consistency group* option. Create the following volume pairs:
   `lunS1 – lunT1` (source-target)
   `lunS2 – lunT2` (source-target)

2. When the previous task completes, run the *consistency created* task on all logical subsystems that hold any of the FlashCopy disks (only `LSS1` in this example).

Generating a FlashCopy image using file system-level suspension

1. Suspend all file system activity and flush the GPFS buffers on all nodes:
   `mmfsctl fs0 suspend` (run on any node in the GPFS cluster)

2. Run the e*stablish FlashCopy pair* task to create the following volume pairs:
   `lunS1 – lunT1` (source-target)
   `lunS2 – lunT2` (source-target)

3. Resume the file system activity:
   `mmfsctl fs0 resume` (run on any node in the GPFS cluster)

Both techniques allow for the consistency of the FlashCopy image by the means of temporary suspension of I/O, but either can be seen as the preferred method depending on your specific requirements and the nature of your GPFS client application.

While the use of FlashCopy consistency groups provide for the proper ordering of updates, this method does not by itself suffice to guarantee the atomicity of updates as seen from the point of view of the user application. If the application process is actively writing data to GPFS, the on-disk content of the file system may, at any point in time, contain some number of incomplete data record updates (those that have not been fully written to disk), and possibly some number of in-progress updates to the GPFS metadata. These would appear as partial updates in the FlashCopy image of the file system, which must be dealt before enabling the image for normal file system use.

While the use of metadata logging techniques enables GPFS to detect and recover from partial updates to the file system's metadata, ensuring the atomicity of updates to the actual data remains the responsibility of the user application. Consequently, the use of FlashCopy consistency groups is suitable only for applications that implement proper mechanisms for the recovery from incomplete updates to their data.

On the other hand, by quiescing the I/O activity at the file system level (with the **`mmfsctl suspend`** and **`mmfsctl resume`** commands), we permit GPFS to fully flush its data buffers, thus allowing it to complete all outstanding updates to the user data, as well as the GPFS metadata. This, in turn, prevents the appearance of incomplete updates in the FlashCopy image. For this

reason, we suggest file system-level suspension as the general-purpose method for protecting the integrity of your FlashCopy images.

Several uses of the FlashCopy replica after its initial creation can be considered. For example, if your primary operating environment suffers a permanent loss or a corruption of data, you may choose to flash the target disks back onto the originals to quickly restore access to a copy of the file system as seen at the time of the previous snapshot. Before restoring the file system from a FlashCopy, please make sure to suspend the activity of the GPFS client processes and unmount the file system on all GPFS nodes.

To help protect your data against site-wide disasters, you may chose to instead transfer the replica off-site to a remote location using PPRC or any other type of bulk data transfer technology. Alternatively, you may choose to mount the FlashCopy image as a separate file system on your backup processing server and transfer the data to tape storage. To enable regular file system access to your FlashCopy replica, create a single-node GPFS cluster on your backup server node and execute the **`mmfsctl syncFSconfig`** command to import the definition of the file system from your production GPFS cluster.

Please note:

1. The primary copy of a GPFS file system and its FlashCopy image cannot coexist in the same GPFS cluster. A node can mount either the original copy of the file system or one of its FlashCopy replicas, but not both. This restriction has to do with the current implementation of the NSD-to-LUN mapping mechanism, which scans all locally-attached disks, searching for a specific value (the NSD id) at a particular location on disk. If both the original volume and its FlashCopy image are visible to a particular node, these disks would appear to GPFS as distinct devices with identical NSD ids.

For this reason, we ask users to zone their SAN configurations such that at most one replica of any given GPFS disk is visible from any node. That is, the nodes in your production cluster should have access to the disks that make up the actual file system but should not see the disks holding the FlashCopy image(s), whereas the backup server should see the FlashCopy targets but not the originals.

Alternatively, you can use the **`/var/mmfs/etc/nsddevices`** user exit to explicitly define the subset of the locally visible disks to be accessed during the NSD device scan on the local node. (Consult the GPFS documentation errata for details on the use of **`nsddevices`**).

2. We suggest generating a new FlashCopy replica immediately after every administrative change to the state of the file system (for example, the additions or removals of disks). This would help eliminate the risk of a discrepancy between the GPFS configuration data contained in the **mmsdrfs** file and the on-disk content of the replica.

# References

- *GPFS documentation* at http://publib.boulder.ibm.com/clresctr/windows/public/gpfsbooks.html

- *GPFS FAQ* at http://publib.boulder.ibm.com/clresctr/library/gpfs_faqs.html

- *GPFS Architecture and Performance at* http://www.ibm.com/servers/eserver/clusters/whitepapers/gpfs_aix.html

- *IBM Enterprise Storage Server*, SG24-5465, at http://www.redbooks.ibm.com/redbooks/pdfs/sg245465.pdf

- *IBM Enterprise Storage Server User's Guide*, SC26-7445, at http://publibfp.boulder.ibm.com/epubs/pdf/f2bug05.pdf

- *IBM TotalStorage Enterprise Storage Server Implementing ESS Copy Services in Open Environments* , SG24-5757, at http://www.redbooks.ibm.com/redbooks/pdfs/sg245757.pdf

- *IBM TotalStorage Enterprise Storage Server Web Interface User's Guide*, SC26-7448, at http://publibfp.boulder.ibm.com/epubs/pdf/f2bui05.pdf

- *IBM Enterprise Storage Server Command-Line Interfaces User's Guide*, SC26-7994, at http://publibfp.boulder.ibm.com/epubs/pdf/f2bcli04.pdf

- *ESS Copy Services Tasks: A Possible Naming Convention*, at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0289.html?Open

- *Valid Combinations of ESS Copy Services* at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0311.html?Open

- *Connectivity Guidelines for Synchronous PPRC*, at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0211.html?Open

- *A Disaster Recovery Solution Selection Methodology*, http://www.redbooks.ibm.com/redpapers/pdfs/redp3847.pdf

- *IBM TotalStorage Solutions for Disaster Recovery*, SG26-6547, at http://www.redbooks.ibm.com/redbooks/pdfs/sg246547.pdf

- *Seven Tiers of Disaster Recovery* at http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/tips0340.html?Open


IBM Corporation
Marketing Communications
Systems and Technology Group
Route 100
Somers, New York 10589

Produced in the United States of America
August 2004