General Parallel File System
Version 3 Release 5.0.16

Documentation Update: GPFS Version 3 Release 5.0.16 Information (Applied to GPFS Version 3 Release 5.0.11 Information Units)

General Parallel File System
Version 3 Release 5.0.16

Documentation Update: GPFS Version 3 Release 5.0.16 Information (Applied to GPFS Version 3 Release 5.0.11 Information Units)

IBM

Contents

January 2014

This *Documentation Update* provides information about GPFS Version 3 Release 5.0.16 (applied to GPFS Version 3 Release 5.0.11 information units). These updates are divided into the following sections:

- "Changes and additions to *GPFS: Advanced Administration Guide* (SC23-5182-08)" on page 3
- "Changes and additions to *GPFS: Administration and Programming Reference* (SA23-2221-08)" on page 79

Within these sections, a vertical line to the left of text indicates technical changes or additions made to the previous edition of the information.

Changes and additions to *GPFS: Advanced Administration Guide* (SC23-5182-08)

This update provides new and changed information for the previously published GPFS Version 3 Release 5.0.11 *Advanced Administration Guide*.

GPFS Native RAID

Note: GPFS™ Native RAID is only available on the following:

- Power® 775 Disk Enclosure
- IBM® System x® GPFS Storage Server (GSS). GSS is a high-capacity, high-performance storage solution that combines IBM System x servers, storage enclosures, and drives; software, including GPFS Native RAID; and networking components. GSS uses a building-block approach to create highly scalable storage for use in a broad range of application environments.

GPFS Native RAID is a software implementation of storage RAID technologies within GPFS. Using conventional dual-ported disks in a JBOD configuration, GPFS Native RAID implements sophisticated data placement and error correction algorithms to deliver high levels of storage reliability, availability, and performance. Standard GPFS file systems are created from the NSDs defined through GPFS Native RAID.

This chapter describes the basic concepts, advantages, and motivations behind GPFS Native RAID: redundancy codes; end-to-end checksums; data declustering; and administrator configuration, including recovery groups, declustered arrays, virtual disks, and virtual disk NSDs.

Overview

GPFS Native RAID integrates the functionality of an advanced storage controller into the GPFS NSD server. Unlike an external storage controller, where configuration, LUN definition, and maintenance are beyond the control of GPFS, GPFS Native RAID takes ownership of a JBOD array to directly match LUN definition, caching, and disk behavior to GPFS file system requirements.

Sophisticated data placement and error correction algorithms deliver high levels of storage reliability, availability, serviceability, and performance. GPFS Native RAID provides a variation of the GPFS network shared disk (NSD) called a virtual disk, or *vdisk*. Standard NSD clients transparently access the vdisk NSDs of a file system using the conventional NSD protocol.

The features of GPFS Native RAID include:

- Software RAID: GPFS Native RAID runs on standard AIX® disks in a dual-ported JBOD array, which does not require external RAID storage controllers or other custom hardware RAID acceleration.
- Declustering: GPFS Native RAID distributes client data, redundancy information, and spare space uniformly across all disks of a JBOD. This distribution reduces the rebuild (disk failure recovery process) overhead compared to conventional RAID.
- Checksum: An end-to-end data integrity check, using checksums and version numbers, is maintained between the disk surface and NSD clients. The checksum algorithm uses version numbers to detect silent data corruption and lost disk writes.
- Data redundancy: GPFS Native RAID supports highly reliable 2-fault-tolerant and 3-fault-tolerant Reed-Solomon based parity codes and 3-way and 4-way replication.
- Large cache: A large cache improves read and write performance, particularly for small I/O operations.

- Arbitrarily sized disk arrays: The number of disks is not restricted to a multiple of the RAID redundancy code width, which allows flexibility in the number of disks in the RAID array.
- Multiple redundancy schemes: One disk array can support vdisks with different redundancy schemes, for example Reed-Solomon and replication codes.
- Disk hospital: A disk hospital asynchronously diagnoses faulty disks and paths, and requests replacement of disks by using past health records.
- Automatic recovery: Seamlessly and automatically recovers from primary server failure.
- Disk scrubbing: A disk scrubber automatically detects and repairs latent sector errors in the background.
- Familiar interface: Standard GPFS command syntax is used for all configuration commands; including, maintaining and replacing failed disks.
- Flexible hardware configuration: Support of JBOD enclosures with multiple disks physically mounted together on removable carriers.
- Configuration and data logging: Internal configuration and small-write data are automatically logged to solid-state disks for improved performance.

GPFS Native RAID features

This section introduces three key features of GPFS Native RAID and how they work: data redundancy using RAID codes, end-to-end checksums, and declustering.

RAID codes

GPFS Native RAID automatically corrects for disk failures and other storage faults by reconstructing the unreadable data using the available data redundancy of either a Reed-Solomon code or *N*-way replication. GPFS Native RAID uses the reconstructed data to fulfill client operations, and in the case of disk failure, to rebuild the data onto spare space. GPFS Native RAID supports 2- and 3-fault-tolerant Reed-Solomon codes and 3-way and 4-way replication, which respectively detect and correct up to two or three concurrent faults[1]. The redundancy code layouts supported by GPFS Native RAID, called *tracks*, are illustrated in Figure 1.

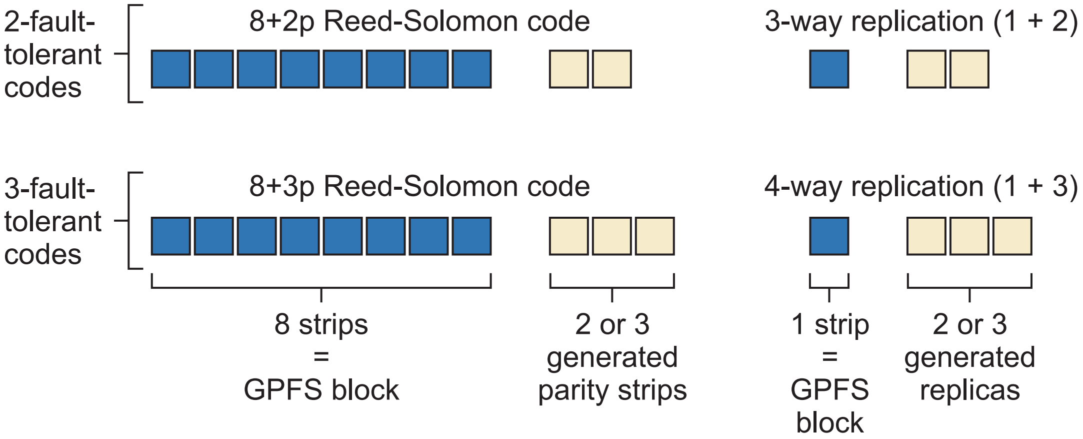

Figure 1. Redundancy codes supported by GPFS Native RAID. GPFS Native RAID supports 2- and 3-fault-tolerant Reed-Solomon codes, which partition a GPFS block into eight data strips and two or three parity strips. The *N*-way replication codes duplicate the GPFS block on *N* - 1 replica strips.

[1] An f-fault-tolerant Reed-Solomon code or a $(1 + f)$-way replication can survive the *concurrent* failure of f disks, read faults, or either. Also, if there are s equivalent spare disks in the array, an f-fault-tolerant array can survive the *sequential* failure of $f + s$ disks where disk failures occur between successful rebuild operations.

GPFS Native RAID automatically creates redundancy information depending on the configured RAID code. Using a Reed-Solomon code, GPFS Native RAID equally divides a GPFS block of user data into eight *data strips* and generates two or three redundant *parity strips*. This results in a stripe or track width of 10 or 11 strips and storage efficiency of 80% or 73% (excluding user configurable spare space for rebuild).

Using *N*-way replication, a GPFS data block is simply replicated *N* – 1 times, in effect implementing 1 + 2 and 1 + 3 redundancy codes, with the strip size equal to the GPFS block size. Thus, for every block/strip written to the disks, *N* replicas of that block/strip are also written. This results in track width of three or four strips and storage efficiency of 33% or 25%.

End-to-end checksum

Most implementations of RAID codes implicitly assume that disks reliably detect and report faults, hard-read errors, and other integrity problems. However, studies have shown that disks do not report some read faults and occasionally fail to write data, while actually claiming to have written the data. These errors are often referred to as silent errors, phantom-writes, dropped-writes, and off-track writes. To cover for these shortcomings, GPFS Native RAID implements an end-to-end checksum that can detect silent data corruption caused by either disks or other system components that transport or manipulate the data.

When an NSD client is writing data, a checksum of 8 bytes is calculated and appended to the data before it is transported over the network to the GPFS Native RAID server. On reception, GPFS Native RAID calculates and verifies the checksum. Then, GPFS Native RAID stores the data, a checksum, and version number to disk and logs the version number in its metadata for future verification during read.

When GPFS Native RAID reads disks to satisfy a client read operation, it compares the disk checksum against the disk data and the disk checksum version number against what is stored in its metadata. If the checksums and version numbers match, GPFS Native RAID sends the data along with a checksum to the NSD client. If the checksum or version numbers are invalid, GPFS Native RAID reconstructs the data using parity or replication and returns the reconstructed data and a newly generated checksum to the client. Thus, both silent disk read errors and lost or missing disk writes are detected and corrected.

Declustered RAID

Compared to conventional RAID, GPFS Native RAID implements a sophisticated data and spare space disk layout scheme that allows for arbitrarily sized disk arrays while also reducing the overhead to clients when recovering from disk failures. To accomplish this, GPFS Native RAID uniformly spreads or *declusters* user data, redundancy information, and spare space across all the disks of a declustered array. Figure 2 on page 6 compares a conventional RAID layout versus an equivalent declustered array.

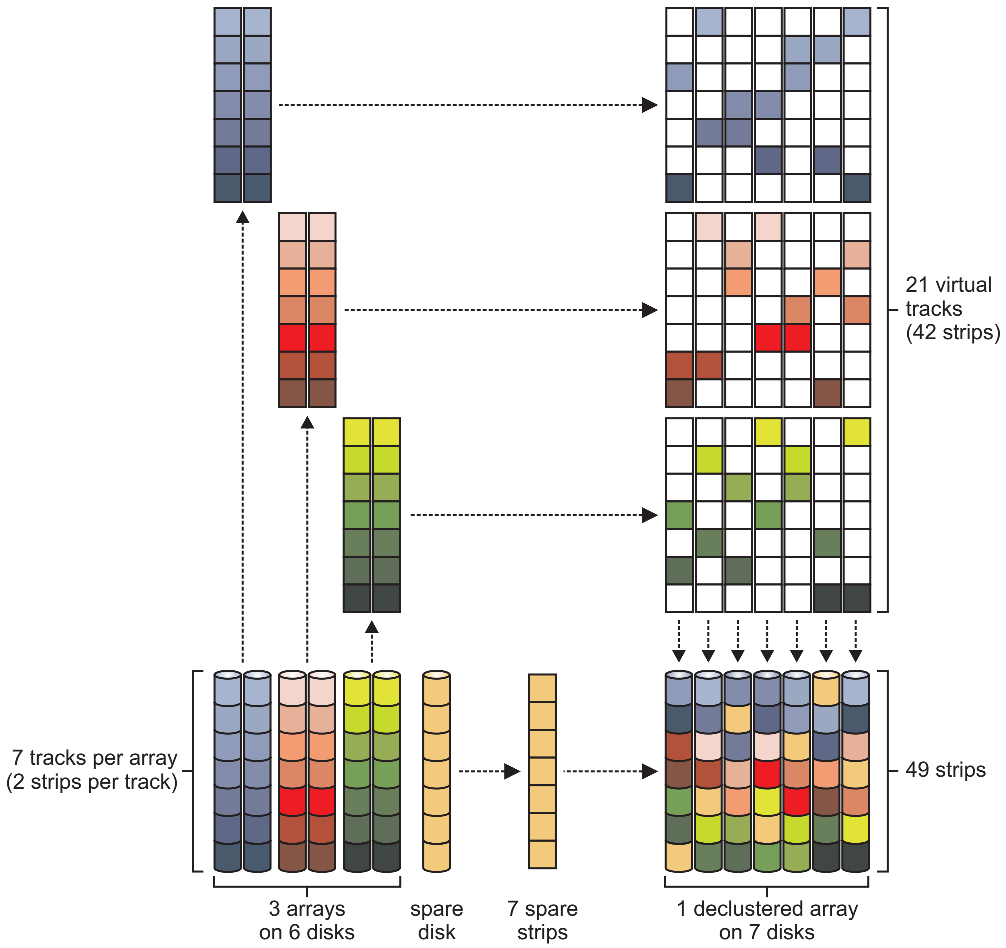
21 virtual tracks (42 strips)
7 tracks per array (2 strips per track)
49 strips
3 arrays on 6 disks
spare disk
7 spare strips
1 declustered array on 7 disks

Figure 2. Conventional RAID versus declustered RAID layouts. This figure is an example of how GPFS Native RAID improves client performance during rebuild operations by utilizing the throughput of all disks in the declustered array. This is illustrated here by comparing a conventional RAID of three arrays versus a declustered array, both using 7 disks. A conventional 1-fault-tolerant 1 + 1 replicated RAID array in the lower left is shown with three arrays of two disks each (data and replica strips) and a spare disk for rebuilding. To decluster this array, the disks are divided into seven tracks, two strips per array, as shown in the upper left. The strips from each group are then combinatorially spread across all seven disk positions, for a total of 21 virtual tracks, per the upper right. The strips of each disk position for every track are then arbitrarily allocated onto the disks of the declustered array of the lower right (in this case, by vertically sliding down and compacting the strips from above). The spare strips are uniformly inserted, one per disk.

As illustrated in Figure 3 on page 7, a declustered array can significantly shorten the time required to recover from a disk failure, which lowers the rebuild overhead for client applications. When a disk fails, erased data is rebuilt using all the operational disks in the declustered array, the bandwidth of which is greater than that of the fewer disks of a conventional RAID group. Furthermore, if an additional disk fault occurs during a rebuild, the number of impacted tracks requiring repair is markedly less than the previous failure and less than the constant rebuild overhead of a conventional array.

The decrease in declustered rebuild impact and client overhead can be a factor of three to four times less than a conventional RAID. Because GPFS stripes client data across all the storage nodes of a cluster, file system performance becomes less dependent upon the speed of any single rebuilding storage array.

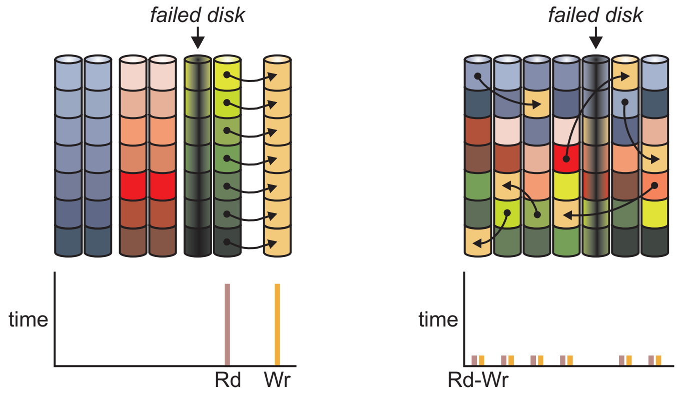

Figure 3. Lower rebuild overhead in conventional RAID versus declustered RAID. When a single disk fails in the 1-fault-tolerant 1 + 1 conventional array on the left, the redundant disk is read and copied onto the spare disk, which requires a throughput of 7 strip I/O operations. When a disk fails in the declustered array, all replica strips of the six impacted tracks are read from the surviving six disks and then written to six spare strips, for a throughput of 2 strip I/O operations. The bar chart illustrates disk read and write I/O throughput during the rebuild operations.

Disk configurations

This section describes recovery group and declustered array configurations.

Recovery groups

GPFS Native RAID divides disks into *recovery groups* where each is physically connected to two servers: primary and backup. All accesses to any of the disks of a recovery group are made through the active server of the recovery group, either the primary or backup.

Building on the inherent NSD failover capabilities of GPFS, when a GPFS Native RAID server stops operating because of a hardware fault, software fault, or normal shutdown, the backup GPFS Native RAID server seamlessly takes over control of the associated disks of its recovery groups.

Typically, a JBOD array is divided into two recovery groups controlled by different primary GPFS Native RAID servers. If the primary server of a recovery group fails, control automatically switches over to its backup server. Within a typical JBOD, the primary server for a recovery group is the backup server for the other recovery group.

Figure 4 on page 8 illustrates the ring configuration where GPFS Native RAID servers and storage JBODs alternate around a loop. A particular GPFS Native RAID server is connected to two adjacent storage JBODs and vice versa. The ratio of GPFS Native RAID server to storage JBODs is thus one-to-one. Load on servers increases by 50% when a server fails.

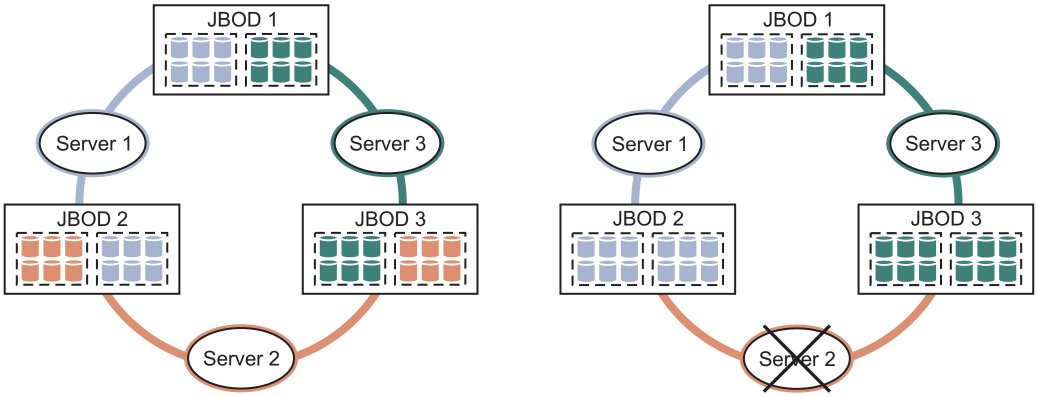

Figure 4. GPFS Native RAID server and recovery groups in a ring configuration. A recovery group is illustrated as the dashed-line enclosed group of disks within a storage JBOD. Server *N* is the primary controller of the first recovery group in JBOD *N* (and backup for its second recovery group), and the primary controller of the second recovery group in JBOD *N* + 1 (and backup for its first recovery group). As shown, when server 2 fails, control of the first recovery group in JBOD 2 is taken over by its backup server 1, and control of the second recovery group in JBOD 3 is taken over by its backup server 3. During the failure of server 2, the load on backup server 1 and 3 increases by 50% from two to three recovery groups.

For small configurations, Figure 5 illustrates a setup with two GPFS Native RAID servers connected to one storage JBOD. For handling server failures, this configuration can be less efficient for large clusters because it requires 2 × *N* servers each capable of serving two recovery groups, where *N* is the number of JBOD arrays. Conversely, the ring configuration requires 1 × *N* servers each capable of serving three recovery groups.

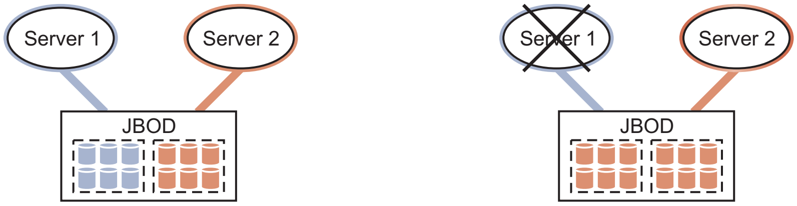

Figure 5. Minimal configuration of two GPFS Native RAID servers and one storage JBOD. GPFS Native RAID server 1 is the primary controller for the first recovery group and backup for the second recovery group. GPFS Native RAID server 2 is the primary controller for the second recovery group and backup for the first recovery group. As shown, when server 1 fails, control of the first recovery group is taken over by its backup server 2. During the failure of server 1, the load on backup server 2 increases by 100% from one to two recovery groups.

Declustered arrays

A declustered array is a subset of the physical disks (pdisks) in a recovery group across which data, redundancy information, and spare space are declustered. The number of disks in a declustered array is determined by the RAID code-width of the vdisks that will be housed in the declustered array. For more information, see “Virtual disks” on page 9. There can be one or more declustered arrays per recovery group. Figure 6 on page 9 illustrates a storage JBOD with two recovery groups, each with four declustered arrays.

A declustered array can hold one or more vdisks. Since redundancy codes are associated with vdisks, a declustered array can simultaneously contain both Reed-Solomon and replicated vdisks.

If the storage JBOD supports multiple disks physically mounted together on removable carriers, removal of a carrier temporarily disables access to all the disks in the carrier. Thus, pdisks on the same carrier should not be in the same declustered array, as vdisk redundancy protection would be weakened upon carrier removal.

Declustered arrays are normally created at recovery group creation time but new ones can be created or existing ones grown by adding pdisks at a later time.

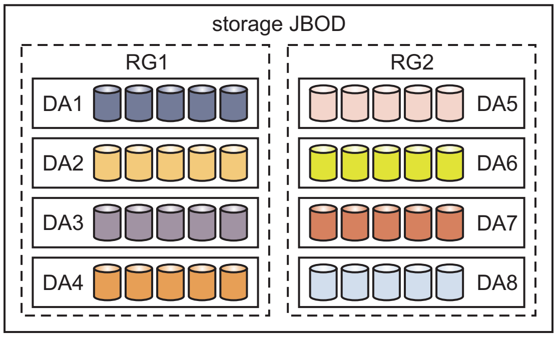

Figure 6. Example of declustered arrays and recovery groups in storage JBOD. This figure shows a storage JBOD with two recovery groups, each recovery group with four declustered arrays, and each declustered array with five disks.

Virtual and physical disks

A virtual disk (vdisk) is a type of NSD, implemented by GPFS Native RAID across all the physical disks (pdisks) of a declustered array. Multiple vdisks can be defined within a declustered array, typically Reed-Solomon vdisks for GPFS user data and replicated vdisks for GPFS metadata.

Virtual disks

Whether a vdisk of a particular capacity can be created in a declustered array depends on its redundancy code, the number of pdisks and equivalent spare capacity in the array, and other small GPFS Native RAID overhead factors. The **mmcrvdisk** command can automatically configure a vdisk of the largest possible size given a redundancy code and configured spare space of the declustered array.

In general, the number of pdisks in a declustered array cannot be less than the widest redundancy code of a vdisk plus the equivalent spare disk capacity of a declustered array. For example, a vdisk using the 11-strip-wide 8 + 3p Reed-Solomon code requires at least 13 pdisks in a declustered array with the equivalent spare space capacity of two disks. A vdisk using the 3-way replication code requires at least five pdisks in a declustered array with the equivalent spare capacity of two disks.

Vdisks are partitioned into virtual tracks, which are the functional equivalent of a GPFS block. All vdisk attributes are fixed at creation and cannot be subsequently altered.

Physical disks

A pdisk is used by GPFS Native RAID to store both user data and GPFS Native RAID internal configuration data.

A pdisk is either a conventional rotating magnetic-media disk (HDD) or a solid-state disk (SSD). All pdisks in a declustered array must have the same capacity.

It is assumed that pdisks are dual ported. This configuration means one or more paths are connected to the primary GPFS Native RAID server and one or more paths are connected to the backup server. There are typically two redundant paths between a GPFS Native RAID server and connected JBOD pdisks.

Solid-state disks

GPFS Native RAID assumes several solid-state disks (SSDs) in each recovery group in order to redundantly log changes to its internal configuration and fast-write data in non-volatile memory, which is accessible from either the primary or backup GPFS Native RAID servers after server failure. A typical GPFS Native RAID log vdisk might be configured as 3-way replication over a dedicated declustered array of 4 SSDs per recovery group.

Disk hospital

The disk hospital is a key feature of GPFS Native RAID that asynchronously diagnoses errors and faults in the storage subsystem. GPFS Native RAID times out an individual pdisk I/O operation after about ten seconds, thereby limiting the impact from a faulty pdisk on a client I/O operation. When a pdisk I/O operation results in a timeout, an I/O error, or a checksum mismatch, the suspect pdisk is immediately admitted into the disk hospital. When a pdisk is first admitted, the hospital determines whether the error was caused by the pdisk itself or by the paths to it. While the hospital diagnoses the error, GPFS Native RAID, if possible, uses vdisk redundancy codes to reconstruct lost or erased strips for I/O operations that would otherwise have used the suspect pdisk.

Health metrics

The disk hospital maintains the following internal health assessment metrics for each pdisk. When one of these metrics exceeds the threshold, the pdisk is marked for replacement according to the disk maintenance replacement policy for the declustered array.

relativePerformance

Characterizes response times. Values greater than one indicate that the disk is performing above average speed; values less than one indicate that the disk is performing below average speed. Values within a range of 0.800 to 1.250 are considered normal. Examples of typical values are: 0.932, 0.978, 1.039, and 1.095 If the **relativePerformance** of a disk falls below a particular threshold (the default setting is 0.667), the hospital adds “slow” to the pdisk state, and the disk is prepared for replacement.

dataBadness

Characterizes media errors (hard errors) and checksum errors.

The disk hospital logs selected Self-Monitoring, Analysis and Reporting Technology (SMART) data, including the number of internal sector remapping events for each pdisk.

Pdisk discovery

GPFS Native RAID discovers all connected pdisks when it starts up, and then regularly schedules a process that will rediscover a pdisk that becomes newly accessible to the GPFS Native RAID server. This allows pdisks to be physically connected or connection problems to be repaired without restarting the GPFS Native RAID server.

Disk replacement

The disk hospital keeps track of disks that require replacement according to the disk replacement policy of the declustered array, and it can be configured to report the need for replacement in a variety of ways. It records and reports the FRU number and physical hardware location of failed disks to help guide service personnel to the correct location with replacement disks.

When multiple disks are mounted on a removable carrier, each a member of a different declustered array, disk replacement requires the hospital to temporarily suspend other disks in the same carrier. In order to guard against human error, carriers are also not removable until GPFS Native RAID actuates a solenoid controlled latch. In response to administrative commands, the hospital quiesces the appropriate disks, releases the carrier latch, and turns on identify lights on the carrier adjacent to the disks that require replacement.

After one or more disks are replaced and the carrier is re-inserted, the hospital, in response to administrative commands, verifies that the repair has taken place and automatically adds any new disks to the declustered array, which causes GPFS Native RAID to rebalance the tracks and spare space across all the disks of the declustered array. If service personnel fail to re-insert the carrier within a reasonable period, the hospital declares the disks on the carrier as `missing` and starts rebuilding the affected data.

Managing GPFS Native RAID

This section describes, in more detail, the characteristics and behavior of GPFS Native RAID entities: recovery groups, pdisks, declustered arrays, and vdisks. Disk maintenance and overall GPFS Native RAID management are also described.

Recovery groups

A *recovery group* is the fundamental organizing structure employed by GPFS Native RAID. A recovery group is conceptually the internal GPFS equivalent of a hardware disk controller. Within a recovery group, individual JBOD disks are defined as *pdisks* and assigned to *declustered arrays*. Each pdisk belongs to exactly one declustered array within one recovery group. Within a declustered array of pdisks, *vdisks* are defined. The vdisks are the equivalent of the RAID LUNs for a hardware disk controller. One or two GPFS cluster nodes must be defined as the servers for a recovery group, and these servers must have direct hardware connections to the JBOD disks in the recovery group. Two servers are recommended for high availability server failover, but only one server will actively manage the recovery group at any given time. One server is the preferred and *primary* server, and the other server, if defined, is the *backup* server.

Multiple recovery groups can be defined, and a GPFS cluster node can be the primary or backup server for more than one recovery group. The name of a recovery group must be unique within a GPFS cluster.

Recovery group server parameters:

To enable a GPFS cluster node as a recovery group server, it must have the **mmchconfig** configuration parameter **nsdRAIDTracks** set to a nonzero value, and the GPFS daemon must be restarted on the node. The **nsdRAIDTracks** parameter defines the maximum number of vdisk track descriptors that the server can have in memory at a given time. The volume of actual vdisk data that the server can cache in memory is governed by the size of the GPFS pagepool on the server and the value of the **nsdRAIDBufferPoolSizePct** configuration parameter. The **nsdRAIDBufferPoolSizePct** parameter defaults to 50% of the pagepool on the server. A recovery group server should be configured with a substantial amount of pagepool, on the order of tens of gigabytes. A recovery group server becomes an NSD server after NSDs are defined on the vdisks in the recovery group, so the **nsdBufSpace** parameter also applies. The default for **nsdBufSpace** is 30% of the pagepool, and it can be decreased to its minimum value of 10% because the vdisk data buffer pool is used directly to serve the vdisk NSDs.

The vdisk track descriptors, as governed by **nsdRAIDTracks**, include such information as the RAID code, track number, and status. The descriptors also contain pointers to vdisk data buffers in the GPFS pagepool, as governed by **nsdRAIDBufferPoolSizePct**. It is these buffers that hold the actual vdisk data and redundancy information.

For more information on how to set the **nsdRAIDTracks** and **nsdRAIDBufferPoolSizePct** parameters, see "Planning considerations for GPFS Native RAID" on page 28.

For more information on the **nsdRAIDTracks**, **nsdRAIDBufferPoolSizePct**, and **nsdBufSpace** parameters, see the **mmchcon** command in the *GPFS: Administration and Programming Reference*.

Recovery group creation:

Recovery groups are created using the **mmcrrecoverygroup** command, which takes the following arguments:

- The name of the recovery group to create.

- The name of a stanza file describing the declustered arrays and pdisks within the recovery group.
- The names of the GPFS cluster nodes that will be the primary and, if specified, backup servers for the recovery group.

When a recovery group is created, the GPFS daemon must be running with the **nsdRAIDTracks** configuration parameter in effect on the specified servers.

For more information see the **mmcrrecoverygroup** command in the *GPFS: Administration and Programming Reference*.

Recovery group server failover:

When, as is recommended, a recovery group is assigned two servers, one server is the preferred and *primary* server for the recovery group and the other server is the *backup* server. Only one server can serve the recovery group at any given time; this server is known as the *active* recovery group server. The server that is not currently serving the recovery group is the *standby* server. If the active recovery group server is unable to serve a recovery group, it will relinquish control of the recovery group and pass it to the standby server, if available. The failover from the active to the standby server should be transparent to any GPFS file system using the vdisk NSDs in the recovery group. There will be a pause in access to the file system data in the vdisk NSDs of the recovery group while the recovery operation takes place on the new server. This server failover recovery operation involves the new server opening the component disks of the recovery group and playing back any logged RAID transactions.

The active server for a recovery group can be changed by the GPFS administrator using the **mmchrecoverygroup** command. This command can also be used to change the primary and backup servers for a recovery group. For more information, see the **mmchrecoverygroup** command in the *GPFS: Administration and Programming Reference*.

Pdisks

The GPFS Native RAID *pdisk* is an abstraction of a physical disk. A pdisk corresponds to exactly one physical disk, and belongs to exactly one declustered array within exactly one recovery group. Before discussing how declustered arrays collect pdisks into groups, it will be useful to describe the characteristics of pdisks.

A recovery group may contain a maximum of 512 pdisks. A declustered array within a recovery group may contain a maximum of 128 pdisks. The name of a pdisk must be unique within a recovery group; that is, two recovery groups may each contain a pdisk named `disk10`, but a recovery group may not contain two pdisks named `disk10`, even if they are in different declustered arrays.

A pdisk is usually created using the **mmcrrecoverygroup** command, whereby it is assigned to a declustered array within a newly created recovery group. In unusual situations, pdisks may also be created and assigned to a declustered array of an existing recovery group by using the **mmaddpdisk** command.

To create a pdisk, a stanza must be supplied to the **mmcrrecoverygroup** or **mmaddpdisk** commands specifying the pdisk name, the declustered array name to which it is assigned, and a block device special file name for the entire physical disk as it is configured by the operating system on the active recovery group server. The following is an example pdisk creation stanza:

```
%pdisk: pdiskName=c073d1
        da=DA1
        device=/dev/hdisk192
```

The device name for a pdisk must refer to the entirety of a single physical disk; pdisks should not be created using virtualized or software-based disks (for example, logical volumes, disk partitions, logical units from other RAID controllers, or network-attached disks). For a pdisk to be successfully created, the physical disk must be present and functional at the specified device name on the active server. The physical disk must also be present on the standby recovery group server, if one is configured (note that the physical disk block device special name on the standby server will almost certainly be different, and will automatically be discovered by GPFS).

The attributes of a pdisk include the physical disk's unique world wide name (WWN), its field replaceable unit (FRU) code, and its physical location code. Pdisk attributes may be displayed by the **mmlspdisk** command; of particular interest here are the pdisk device *paths* and the pdisk *states*.

Pdisks that have failed and been marked for replacement by the disk hospital are replaced using the **mmchcarrier** command. In unusual situations, pdisks may be added or deleted using the **mmaddpdisk** or **mmdelpdisk** commands. When deleted, either through replacement or the **mmdelpdisk** command, the pdisk abstraction will only cease to exist when all the data it contained has been rebuilt onto spare space (even though the physical disk may have been removed from the system).

Pdisks are normally under the control of GPFS Native RAID and the disk hospital. In some situations, however, the **mmchpdisk** command may be used to directly manipulate pdisks. For example, if a pdisk has to be temporarily removed to allow for hardware maintenance on other parts of the system, you can use the **mmchpdisk --begin-service-drain** command to drain the data before removing the pdisk. After bringing the pdisk back online, you can use the **mmchpdisk --end-service-drain** command to return the drained data to the pdisk.

Note: This process requires that there be sufficient spare space in the declustered array for the data that is to be drained. If the available spare space is insufficient, it can be increased with the **mmchrecoverygroup** command.

Pdisk paths:

To the operating system, physical disks are made visible as block devices with device special file names, such as `/dev/hdisk32`. To achieve high availability and throughput, the physical disks of a JBOD array are connected to each server by multiple (usually two) interfaces in a configuration known as *multipath* (or *dualpath*). When two operating system block devices are visible for each physical disk, GPFS Native RAID refers to them as the *paths* to the pdisk.

In normal operation, the paths to individual pdisks are automatically discovered by GPFS Native RAID. There are only two instances when a pdisk must be referred to by its explicit block device path name: during recovery group creation using the **mmcrrecoverygroup** command, and when adding new pdisks to an existing recovery group with the **mmaddpdisk** command. In both of these cases, only one of the block device path names as seen on the active server needs to be specified; any other paths on the active and standby servers will be automatically discovered.

The operating system may have the ability to internally merge multiple paths to a physical disk into a single block device. When GPFS Native RAID is in use, the operating system multipath merge function must be disabled because GPFS Native RAID itself manages the individual paths to the disk. For more information, see "Example scenario: Configuring GPFS Native RAID recovery groups" on page 32.

Pdisk states:

GPFS Native RAID maintains its view of a pdisk and its corresponding physical disk by means of a *pdisk state*. The pdisk state consists of multiple keyword flags, which may be displayed using the **mmlsrecoverygroup** or **mmlspdisk** commands. The pdisk state flags indicate how GPFS Native RAID is currently using or managing a disk.

In normal circumstances, the state of the vast majority of pdisks will be represented by the sole keyword `ok`. This means that GPFS Native RAID considers the pdisk to be healthy: the recovery group server is able to communicate with the disk, the disk is functioning normally, and the disk can be used to store data. The diagnosing flag will be present in the pdisk state when the GPFS Native RAID disk hospital suspects, or attempts to correct, a problem. If GPFS Native RAID is unable to communicate with a disk, the pdisk state will include the keyword missing. If a missing disk becomes reconnected and functions properly, its state will change back to ok. The readonly flag means that a disk has indicated that it can no longer safely write data. A disk can also be marked by the disk hospital as failing, perhaps due to an excessive number of media or checksum errors. When the disk hospital concludes that a disk is no longer operating effectively, it will declare the disk to be dead. If the number of dead pdisks reaches or exceeds the replacement threshold of their declustered array, the disk hospital will add the flag replace to the pdisk state, which indicates that physical disk replacement should be performed as soon as possible.

When the state of a pdisk indicates that it can no longer behave reliably, GPFS Native RAID will rebuild the pdisk's data onto spare space on the other pdisks in the same declustered array. This is called *draining* the pdisk. That a pdisk is draining or has been drained will be indicated by a keyword in the pdisk state flags. The flag systemDrain means that GPFS Native RAID has decided to rebuild the data from the pdisk; the flag adminDrain means that the GPFS administrator issued the **mmdelpdisk** command to delete the pdisk.

GPFS Native RAID stores both user (GPFS file system) data and its own internal recovery group data and vdisk configuration data on pdisks. Additional pdisk state flags indicate when these data elements are not present on a pdisk. When a pdisk starts draining, GPFS Native RAID first replicates the recovery group data and vdisk configuration data onto other pdisks. When this completes, the flags noRGD (no recovery group data) and noVCD (no vdisk configuration data) are added to the pdisk state flags. When the slower process of removing all user data completes, the noData flag will be added to the pdisk state.

To summarize, the vast majority of pdisks will be in the ok state during normal operation. The ok state indicates that the disk is reachable, functioning, not draining, and that the disk contains user data and GPFS Native RAID recovery group and vdisk configuration information. A more complex example of a pdisk state is dead/systemDrain/noRGD/noVCD/noData for a single pdisk that has failed. This set of pdisk state flags indicates that the pdisk was declared dead by the system, was marked to be drained, and that all of its data (recovery group, vdisk configuration, and user) has been successfully rebuilt onto the spare space on other pdisks.

In addition to those discussed here, there are some transient pdisk states that have little impact on normal operations; the complete set of states is documented in Table 1.

Table 1. Pdisk states

State	Description
ok	The disk is functioning normally.
dead	The disk failed.
missing	GPFS Native RAID is unable to communicate with the disk.
diagnosing	The disk is temporarily unusable while its status is determined by the disk hospital.
suspended	The disk is temporarily unusable as part of a service procedure.
readonly	The disk is no longer writeable.
failing	The disk is not healthy but not dead.
| slow	The disk's performance is far below expectations.
systemDrain	The disk is faulty, so data and configuration data must be drained.
adminDrain	An administrator requested that this pdisk be deleted.
| serviceDrain	The disk is being drained to be taken temporarily out of service.
noRGD	The recovery group data was drained from the disk.
noVCD	All vdisk configuration data was drained from the disk.
noData	All vdisk user data was drained from the disk.

Table 1. Pdisk states (continued)

State	Description
`undrainable`	There is not enough spare space in the declustered array to drain this disk.
`replace`	Replacement of the disk was requested.
`noPath`	There was no functioning path found to this disk.
`PTOW`	The disk is temporarily unusable because of a pending timed-out write.
`init`	The pdisk object is being initialized or removed.
`formatting`	Initial configuration data is being written to the disk.

Declustered arrays

Declustered arrays are disjoint subsets of the pdisks in a recovery group. Vdisks are created within declustered arrays, and vdisk tracks are declustered across all of an array's pdisks. A recovery group may contain up to 16 declustered arrays. A declustered array may contain up to 128 pdisks (but the total number of pdisks in all declustered arrays within a recovery group may not exceed 512). A pdisk may belong to only one declustered array. The name of a declustered array must be unique within a recovery group; that is, two recovery groups may each contain a declustered array named `DA3`, but a recovery group may not contain two declustered arrays named `DA3`. The pdisks within a declustered array must all be of the same size and should all have similar performance characteristics.

A declustered array is usually created together with its member pdisks and its containing recovery group through the use of the **mmchrecoverygroup** command. A declustered array may also be created using the **mmaddpdisk** command to add pdisks to a declustered array that does not yet exist in a recovery group. A declustered array may be deleted by deleting its last member pdisk, or by deleting the recovery group in which it resides. Any vdisk NSDs and vdisks within the declustered array must already have been deleted. There are no explicit commands to create or delete declustered arrays.

Declustered arrays serve two purposes:

- Segregating a small number of fast SSDs into their own group for storing the vdisk log (the RAID update and recovery group event log).
- Partitioning the disks of a JBOD enclosure into smaller subsets exclusive of a common point of failure, such as removable carriers that hold multiple disks.

The latter consideration comes into play when one considers that removing a disk carrier to perform disk replacement also temporarily removes some good disks, perhaps a number in excess of the fault tolerance of the vdisk NSDs. This would cause temporary suspension of file system activity until the disks are restored. To avoid this, each disk position in a removable carrier should be used to define a separate declustered array, such that disk position one defines `DA1`, disk position two defines `DA2`, and so on. Then when a disk carrier is removed, each declustered array will suffer the loss of just one disk, which is within the fault tolerance of any GPFS Native RAID vdisk NSD.

Declustered array parameters:

Declustered arrays have three parameters that may be changed using the **mmchrecoverygroup** command with the **--declustered-array** option. These are:

- The number of disks' worth of equivalent spare space. This defaults to one for arrays with nine or fewer pdisks, and two for arrays with 10 or more pdisks.
- The number of disks that must fail before the declustered array is marked as needing to have disks replaced. The default is the number of spares.
- The number of days over which all the vdisks in the declustered array are scrubbed for errors. The default is 14 days.

Declustered array size:

GPFS Native RAID distinguishes between large and small declustered arrays. A declustered array is considered *large* if at the time of its creation it contains at least 11 pdisks, including an equivalent spare space of two disks (or at least 10 pdisks, including an equivalent spare space of one disk). All other declustered arrays are considered *small*. At least one declustered array in each recovery group must be large, because only large declustered arrays have enough pdisks to safely store an adequate number of replicas of the GPFS Native RAID configuration data for the recovery group.

Because the narrowest RAID code that GPFS Native RAID supports is 3-way replication, the smallest possible declustered array contains four pdisks, including the minimum required equivalent spare space of one disk. The RAID code width of the intended vdisk NSDs and the amount of equivalent spare space also affect declustered array size; if Reed-Solomon 8 + 3p vdisks, which have a code width of 11, are required, and two disks of equivalent spare space is also required, the declustered array must have at least 13 member pdisks.

Spare space:

While operating with a failed pdisk in a declustered array, GPFS Native RAID continues to serve file system I/O requests by using redundancy information on other pdisks to reconstruct data that cannot be read, and by marking data that cannot be written to the failed pdisk as stale. Meanwhile, to restore full redundancy and fault tolerance, the data on the failed pdisk is rebuilt onto *spare space*, reserved unused portions of the declustered array that are declustered over all the member pdisks. The failed disk is thereby *drained* of its data by copying it to the spare space.

The amount of spare space in a declustered array is set at creation time and may be changed later. The spare space is expressed in whole units equivalent to the capacity of a member pdisk of the declustered array, but is spread among all the member pdisks. There are no dedicated spare pdisks. This implies that a number of pdisks equal to the specified spare space may fail, and the full redundancy of all the data in the declustered array can be restored through rebuild.

At minimum, each declustered array requires spare space equivalent to the size of one member pdisk. Because large declustered arrays have a greater probability of disk failure, the default amount of spare space depends on the size of the declustered array. A declustered array with nine or fewer pdisks defaults to having one disk of equivalent spare space. A declustered array with 10 or more disks defaults to having two disks of equivalent spare space. These defaults can be overridden, especially at declustered array creation. However, if at a later point too much of the declustered array is already allocated for use by vdisks, it may not be possible to increase the amount of spare space.

Declustered array free space:

The declustered array free space reported by the **mmlsrecoverygroup** command reflects the space available for creating vdisks. Spare space is not included in this value since it is not available for creating new vdisks.

Pdisk free space:

The pdisk free space reported by the **mmlsrecoverygroup** command reflects the actual number of unused data partitions on the disk. This includes spare space, so if a pdisk fails, these values will decrease as data is moved to the spare space.

Vdisks

Vdisks are created across the pdisks within a declustered array. Each recovery group requires a special *log home vdisk* to function (along with other log-type vdisks, as appropriate for specific environments); see “Log vdisks” on page 17. All other vdisks are created for use as GPFS file system NSDs.

A recovery group can contain at most 64 vdisks, and the first must be the log vdisk. Vdisks can be allocated arbitrarily among declustered arrays. Vdisks are created with the **mmcrvdisk** command. The **mmdelvdisk** command destroys vdisks and all their contained data.

When creating a vdisk, you must specify the RAID code, block size, vdisk size, and a name that is unique within the recovery group and the GPFS cluster. There are no adjustable parameters available for vdisks.

RAID code:

The type, performance, and space efficiency of the RAID codes used for vdisks, discussed in "RAID codes" on page 4, should be considered when choosing the RAID code for a particular set of user data. GPFS storage pools and policy-based data placement can be used to ensure data is stored with appropriate RAID codes.

Block size:

The vdisk block size must equal the GPFS file system block size of the storage pool where the vdisk is assigned. For replication codes, the supported block sizes are 256 KiB, 512 KiB, 1 MiB and 2 MiB. For Reed-Solomon codes, they are 1 MiB, 2 MiB, 4 MiB, 8 MiB and 16 MiB. See "Planning considerations for GPFS Native RAID" on page 28 for an overview of vdisk configuration considerations.

Vdisk size:

The minimum vdisk size is 1 GiB. The maximum vdisk size is the total space available on the pdisks in the declustered array, taking into account the overhead of the RAID code, minus spare space, minus vdisk configuration data, and minus a small amount of space reserved as a buffer for write operations. GPFS Native RAID will round up the requested vdisk size as required. When creating a vdisk, the user can specify to use all remaining space in the declustered array for that vdisk.

Log vdisks:

There are four types of log vdisks:

log home vdisk
Every recovery group requires one log home vdisk to function. The log home vdisk must be created before any other vdisks in the recovery group, and it can only be deleted after all other vdisks in the recovery group have been deleted. The log home vdisk is divided into four sublogs: long-term event log, short-term event log, metadata log, and fast-write log-to-log small writes.

log tip vdisk
The log tip vdisk (appropriate for certain environments, but not required for all) is a vdisk to which log records are initially written, then migrated to the log home vdisk. The intent is to use a small, high-performance device for the log tip, and a larger vdisk on conventional spinning disks for the log home vdisk. The fast writes to the log tip hide the latency of the spinning disks used for the main body of the log.

log tip backup vdisk
The log tip backup vdisk (appropriate for certain environments, but not required for all) is used as an additional replica of the log tip vdisk when the log tip vdisk is two-way replicated on nonvolatile RAM disks. Ideally, the log tip backup vdisk provides a level of performance between that of NVRAM disks and that of spinning disks.

log reserved vdisk
Log reserved vdisks are optional vdisks that are used when the log home disk is not allocated in its own declustered array. Log reserved vdisks have the same size as the log home vdisk and are used to equalize the space consumption on the data declustered arrays, but they are otherwise unused.

Typical configurations

The following are descriptions of typical vdisk configurations in various recovery group environments:

P775 Configuration

In this configuration, the log home vdisk is allocated on a declustered array made up of four SSDs. Only three-way and four-way replication codes are supported for the log home vdisk. In the typical system with four SSDs and with spare space equal to the size of one disk, the three-way replication code would be used for the log home vdisk

GPFS Storage Server with no non-volatile RAM disks

In this configuration, a three-way replicated log tip vdisk is allocated on a declustered array made up of three SSDs. A four-way replicated log home vdisk is allocated in the first declustered array of HDDs.

GPFS Storage Server with non-volatile RAM disks

In this configuration, a two-way replicated log tip vdisk is allocated on non-volatile RAM disks; one from each of the servers.

A log tip backup vdisk is allocated on a declustered array of one or more SSDs. This provides an additional copy of the log tip data when one of the NVRAM disks is unavailable. If only one SSD is used, then the log tip backup uses a raid code of **Unreplicated**.

A four-way replicated log home vdisk is allocated in the first declustered array of HDDs. A four-way replicated log reserved vdisk for each of the data declustered arrays that do not contain the log home vdisk.

The relationship between vdisks and NSDs:

After creating a vdisk with the **mmcrvdisk** command, NSDs are created from the vdisks by using the **mmcrnsd** command. The relationship between vdisks and NSDs is described as follows:

- GPFS file systems are built from vdisk NSDs in the same way as they are built from any other NSDs.
- While an NSD exists for a vdisk, that vdisk cannot be deleted.
- A node cannot serve both vdisk-based NSDs and non-vdisk-based NSDs.
- A file system cannot support both vdisk-based NSDs and non-vdisk-based NSDs.
- Vdisk NSDs should not be used as tiebreaker disks.

Vdisk states:

GPFS Native Raid reports vdisk states, which are displayed using the **mmlsvdisk** command and also on the **mmlsrecoverygroup** command if the **-L** option is used. Vdisks are normally in the **ok** state, which indicates that the vdisk is fully functional with full redundancy. When a pdisk failure affects a specific vdisk, the vdisk state will be reported as degraded until the affected tracks have been rebuilt onto spare space.

When enough pdisks have failed that the specific vdisk has no redundancy left, the vdisk state will be reported as `critical` until the critically affected tracks have been rebuilt onto spare space. If the system uses up all the available spare space while a vdisk is in the degraded or critical state, the state will be followed by `(need spare)`, which indicates that rebuild activity is stalled, and will resume once the failed pdisks have been replaced.

Table 2. Vdisk states

State	Description
`ok`	The vdisk is functioning normally.

Table 2. Vdisk states (continued)

State	Description
`m/n -degraded`	The vdisk is currently running in degraded mode: *m* is the number of pdisks that are draining *n* is the fault-tolerance of the vdisk.
`critical`	The vdisk is currently running in degraded mode and can tolerate no more pdisk losses.
`(need spare)`	Rebuild will resume when more spare space is available; applies to degraded and critical states.

Maintenance

Very large disk systems, with thousands or tens of thousands of disks and servers, will likely experience a variety of failures during normal operation. To maintain system productivity, the vast majority of these failures must be handled automatically: without loss of data, without temporary loss of access to the data, and with minimal impact on the performance of the system. Some failures require human intervention, such as replacing failed components with spare parts or correcting faults that cannot be corrected by automated processes.

Disk diagnosis:

The disk hospital was introduced in “Disk hospital” on page 10. When an individual disk I/O operation (read or write) encounters an error, GPFS Native RAID completes the NSD client request by reconstructing the data (for a read) or by marking the unwritten data as stale and relying on successfully written parity or replica strips (for a write), and starts the disk hospital to diagnose the disk. While the disk hospital is diagnosing, the affected disk will not be used for serving NSD client requests.

Similarly, if an I/O operation does not complete in a reasonable time period, it is timed out, and the client request is treated just like an I/O error. Again, the disk hospital will diagnose what went wrong. If the timed-out operation is a disk write, the disk remains temporarily unusable until a pending timed-out write (`PTOW`) completes.

The disk hospital then determines the exact nature of the problem. If the cause of the error was an actual media error on the disk, the disk hospital marks the offending area on disk as temporarily unusable, and overwrites it with the reconstructed data. This cures the media error on a typical HDD by relocating the data to spare sectors reserved within that HDD.

If the disk reports that it can no longer write data, the disk is marked as `readonly`. This can happen when no spare sectors are available for relocating in HDDs, or the flash memory write endurance in SSDs was reached. Similarly, if a disk reports that it cannot function at all, for example not spin up, the disk hospital marks the disk as `dead`.

The disk hospital also maintains various forms of *disk badness*, which measure accumulated errors from the disk. If the badness level is high, the disk can be marked `dead`. For less severe cases, the disk can be marked `failing`.

Finally, the GPFS Native RAID server might lose communication with a disk. This can either be caused by an actual failure of an individual disk, or by a fault in the disk interconnect network. In this case, the disk is marked as `missing`.

If a disk would have to be marked dead or missing, and the problem affects only individual disks, not a large set of disks, the disk hospital attempts to recover the disk. If the disk reports that it is not started, the disk hospital attempts to start the disk. If nothing else helps, the disk hospital power-cycles the disk, and then waits for the disk to return online.

Before actually reporting an individual disk as missing, the disk hospital starts a search for that disk by polling all disk interfaces to locate the disk. Only after that fast poll fails is the disk actually declared missing.

If a large set of disks has faults, the GPFS Native RAID server can continue to serve read and write requests, provided that the number of failed disks does not exceed the fault tolerance of either the RAID code for the vdisk or the GPFS Native RAID internal configuration data. When any disk fails, the server begins rebuilding its data onto spare space. If the failure is not considered *critical*, rebuilding is throttled when user workload is present. This ensures that the performance impact to user workload is minimal. A failure might be considered critical if a vdisk has no remaining redundancy information, for example three disk faults for 4-way replication and 8 + 3p or two disk faults for 3-way replication and 8 + 2p. During a critical failure, critical rebuilding will run as fast as possible because the vdisk is in imminent danger of data loss, even if that impacts the user workload. Since the data is declustered, or spread out over many disks, and all disks in the declustered array participate in rebuilding, a vdisk will remain in critical rebuild only for short periods (2-3 minutes for a typical system). A double or triple fault is extremely rare, so the performance impact of critical rebuild is minimized.

In a multiple fault scenario, the server might not have enough disks to fulfill a request. More specifically, such a scenario occurs if the number of unavailable disks exceeds the fault tolerance of the RAID code. If some of the disks are only temporarily unavailable, and are expected back online soon, the server will stall the client I/O and wait for the disk to return to service. Disks can be temporarily unavailable for any of the following reasons:

- The disk hospital is diagnosing an I/O error.
- A timed-out write operation is pending.
- A user intentionally suspended the disk, perhaps it is on a carrier with another failed disk that has been removed for service.

If too many disks become unavailable for the primary server to proceed, it will fail over. In other words, the whole recovery group is moved to the backup server. If the disks are not reachable from the backup server either, then all vdisks in that recovery group become unavailable until connectivity is restored.

A vdisk will suffer data loss when the number of permanently failed disks exceeds the vdisk fault tolerance. This data loss is reported to NSD clients when the data is accessed.

Background tasks:

While GPFS Native RAID primarily performs NSD client read and write operations in the foreground, it also performs several long-running maintenance tasks in the background, which are referred to as *background tasks*. The background task that is currently in progress for each declustered array is reported in the long-form output of the **mmlsrecoverygroup** command. Table 3 describes the long-running background tasks.

Table 3. Background tasks

Task	Description
`repair-RGD/VCD`	Repairing the internal recovery group data and vdisk configuration data from the failed disk onto the other disks in the declustered array.
`rebuild-critical`	Rebuilding virtual tracks that cannot tolerate any more disk failures.
`rebuild-1r`	Rebuilding virtual tracks that can tolerate only one more disk failure.
`rebuild-2r`	Rebuilding virtual tracks that can tolerate two more disk failures.

Table 3. Background tasks (continued)

Task	Description
`rebuild-offline`	Rebuilding virtual tracks where failures exceeded the fault tolerance.
`rebalance`	Rebalancing the spare space in the declustered array for either a `missing` pdisk that was discovered again, or a new pdisk that was added to an existing array.
`scrub`	Scrubbing vdisks to detect any silent disk corruption or latent sector errors by reading the entire virtual track, performing checksum verification, and performing consistency checks of the data and its redundancy information. Any correctable errors found are fixed.

Server failover:

If the primary GPFS Native RAID server loses connectivity to a sufficient number of disks, the recovery group attempts to fail over to the backup server. If the backup server is also unable to connect, the recovery group becomes unavailable until connectivity is restored. If the backup server had taken over, it will relinquish the recovery group to the primary server when it becomes available again.

Data checksums:

GPFS Native RAID stores checksums of the data and redundancy information on all disks for each vdisk. Whenever data is read from disk or received from an NSD client, checksums are verified. If the checksum verification on a data transfer to or from an NSD client fails, the data is retransmitted. If the checksum verification fails for data read from disk, the error is treated similarly to a media error:

- The data is reconstructed from redundant data on other disks.
- The data on disk is rewritten with reconstructed good data.
- The disk badness is adjusted to reflect the silent read error.

Disk replacement:

When one disk fails, the system will rebuild the data that was on the failed disk onto spare space and continue to operate normally, but at slightly reduced performance because the same workload is shared among fewer disks. With the default setting of two spare disks for each large declustered array, failure of a single disk would typically not be a sufficient reason for maintenance.

When several disks fail, the system continues to operate even if there is no more spare space. The next disk failure would make the system unable to maintain the redundancy the user requested during vdisk creation. At this point, a service request is sent to a maintenance management application that requests replacement of the failed disks and specifies the disk FRU numbers and locations.

In general, disk maintenance is requested when the number of failed disks in a declustered array reaches the disk replacement threshold. By default, that threshold is identical to the number of spare disks. For a more conservative disk replacement policy, the threshold can be set to smaller values using the **mmchrecoverygroup** command.

Disk maintenance is performed using the **mmchcarrier** command with the **--release** option, which:

- Suspends all functioning disks on the carrier that is shared with the disk being replaced.
- Powers down all the disks on that carrier.
- Turns on indicators on the disk enclosure and carrier to help locate and identify the disk that requires replacement.
- Unlocks the carrier for disk replacement.

After the disk is replaced and the carrier reinserted, another **mmchcarrier** command with the **--replace** option powers on the disks.

Other hardware service:

While GPFS Native RAID can easily tolerate a single disk fault with no significant impact, and failures of up to three disks with various levels of impact on performance and data availability, it still relies on the vast majority of all disks being functional and reachable from the server. If a major equipment malfunction prevents both the primary and backup server from accessing more than that number of disks, or if those disks are actually destroyed, all vdisks in the recovery group will become either unavailable or suffer permanent data loss. As GPFS Native RAID cannot recover from such catastrophic problems, it also does not attempt to diagnose them or orchestrate their maintenance.

In the case that a GPFS Native RAID server becomes permanently disabled, a manual failover procedure exists that requires recabling to an alternate server (see the **mmchrecoverygroup** command in the *GPFS: Administration and Programming Reference*). If both the primary and backup GPFS Native RAID servers for a recovery group fail, the recovery group is unavailable until one of the servers is repaired.

Component configuration in the GPFS Storage Server

The function described in this topic is only available with the IBM System x GPFS Storage Server.

GPFS uses component information in the cluster configuration data to perform configuration and validation tasks. For example, when a disk needs to be replaced, the system reports the rack location of the enclosure containing the disk along with a description of the disk's slot within the enclosure.

A component in this context refers to some resource, usually hardware, that is part of the GPFS cluster; for example, **rack** or **storage enclosure**. Each component has an associated component specification that is identified by its part number. You can use the **mmlscompspec** command to see the defined component specifications.

You will normally configure components when deploying a cluster or when adding new hardware to a cluster. The cluster must exist before you can run any of the component-related commands. The basic configuration steps are as follows:

1. Add components to the configuration using **mmdiscovercomp** and **mmaddcomp**.
2. Define the location of storage enclosures and servers using **mmchcomploc**.
3. Set the two-digit ID visible on the back of some storage enclosures using **mmsyncdisplayid**.
4. Update the component names and other attributes using **mmchcomp**.

The following commands use or change the cluster's component configuration:

mmaddcomp
Adds new components

mmchcomp
Change component attributes

mmchcomploc
Change component locations

mmdelcomp
Delete components

mmdiscovercomp
Discover components and add them to the configuration

mmlscomp
List components.

mmlscomploc
List component locations

mmlscompspec
List component specifications

mmsyncdisplayid
Synchronize enclosure display IDs with the cluster configuration

The options that these commands support are documented in the Commands section of *GPFS: Administration and Programming Reference*. Most commands allow you to make a single change using command-line arguments. They also support stanza file input for making any number of changes with a single command.

Adding components to the cluster’s configuration:

The function described in this topic is only available with the IBM System x GPFS Storage Server.

Note: Sample output in this topic may not match the output produced on your system.

This section discusses how to add components to the cluster's configuration. Before adding any components, you may want to view the available component specifications. To do so, you can use **mmlscompspec**, which will list the defined component specifications, identified by part number.

```
$ mmlscompspec

    Rack Specifications

Part Number  Height  Description
-----------  ------  --------------------------------------------------
1410HEA          42  Intelligent Cluster 42U 1200mm Deep Expansion Rack
1410HPA          42  Intelligent Cluster 42U 1200mm Deep Primary Rack

    Server Specifications

Part Number  Height  Description
-----------  ------  -------------------
7915AC1           2  IBM System x3650 M4

    Storage Enclosure Specifications

Part Number  Height  Description                  Vendor ID  Product ID  Drive Slots  Has Display ID
-----------  ------  ---------------------------  ---------  ----------  -----------  --------------
181880E           4  DCS3700 Expansion Enclosure  IBM        DCS3700              60               1
```

If this was the first component command run on the cluster, you will see some initialization messages before the command output. The list of component specifications will be short since it only includes supported GSS hardware.

You can use **mmdiscovercomp** to define some of the cluster's components. This command finds supported storage enclosures attached to any of the cluster nodes and adds them to the cluster's configuration. (In your version of GPFS, **mmdiscovercomp** may find additional component types.) The output below resembles a cluster that includes two GSS 24 units.

```
$ mmdiscovercomp all
Adding enclosure: serial number = SV12345001; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345007; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345003; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345005; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345004; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345002; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345008; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345006; vendor ID = IBM; product ID = DCS3700

Storage Enclosures
```

```
Status  Comp ID  Component Type    Part Number  Serial Number  Name               Display ID
------  -------  ----------------  -----------  -------------  ------------------  ----------
new           1  storageEnclosure  181880E      SV12345001     181880E-SV12345001  00
new           2  storageEnclosure  181880E      SV12345007     181880E-SV12345007  00
new           3  storageEnclosure  181880E      SV12345003     181880E-SV12345003  00
new           4  storageEnclosure  181880E      SV12345005     181880E-SV12345005  00
new           5  storageEnclosure  181880E      SV12345004     181880E-SV12345004  00
new           6  storageEnclosure  181880E      SV12345002     181880E-SV12345002  00
new           7  storageEnclosure  181880E      SV12345008     181880E-SV12345008  00
new           8  storageEnclosure  181880E      SV12345006     181880E-SV12345006  00
```

In this example, **mmdiscovercomp** found eight storage enclosures that were not already in the cluster configuration. Notice that each component gets a default name. Below we will discuss how you can change the name to some identifier that is more suitable for your environment. You may run **mmdiscovercomp** a second time, in which case it should not find any new components.

Note: The serial numbers used by GPFS to identify 181880E storage enclosures do not match the product serial numbers shown on the outside of the enclosure. You may want to record this visible serial number as part of the enclosure's name. For instance, you could rename an enclosure to **G1E3-SX98760044**, which would stand for GSS 1 (G1), enclosure 3 (E3), with product serial number SX98760044. For details about how to change the name of a component using the **mmchcomp** command, see "Updating component attributes" on page 26.

The other essential components are racks to hold the storage enclosures. Use the **mmlscompspec** command to see the available rack part numbers, then use the **mmaddcomp** command to define the racks. This example shows defining an "Intelligent Cluster™ 42U 1200mm Deep Primary Rack" and giving it the name **R01C01**.

```
$ mmaddcomp 1410HPA --name R01C01
$ mmlscomp

    Rack Components

Comp ID  Part Number  Serial Number  Name
-------  -----------  -------------  ------
      9  1410HPA                     R01C01

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  181880E      SV12345001     181880E-SV12345001
      2  181880E      SV12345007     181880E-SV12345007
      3  181880E      SV12345003     181880E-SV12345003
      4  181880E      SV12345005     181880E-SV12345005
      5  181880E      SV12345004     181880E-SV12345004
      6  181880E      SV12345002     181880E-SV12345002
      7  181880E      SV12345008     181880E-SV12345008
      8  181880E      SV12345006     181880E-SV12345006
```

Defining component locations:

The function described in this topic is only available with the IBM System x GPFS Storage Server.

Note: Sample output in this topic may not match the output produced on your system.

You use the **mmchcomploc** command to place a component in a rack or change its location. This example puts the storage enclosure SV12345003 in rack R01C01 at U location 13.

```
$ mmchcomploc SV12345003 R01C01 13
$ mmlscomploc
Component           Location
------------------  ------------------
181880E-SV12345003  Rack R01C01 U13-16
```

The first argument, which identifies the component, can be either the component ID, serial number, or name. The second argument is the rack (container) and the third argument is the U position in the rack.

Changing one component at a time can be slow because each change is propagated to all nodes in the cluster. You may want to use a stanza file to make multiple changes with a single command. The following example uses a stanza file to position the remaining enclosures:

```
$ cat comploc.stanza
%compLoc: compId=SV12345007 containerId=R01C01 position=1
%compLoc: compId=SV12345005 containerId=R01C01 position=5
%compLoc: compId=SV12345003 containerId=R01C01 position=13
%compLoc: compId=SV12345008 containerId=R01C01 position=17
%compLoc: compId=SV12345001 containerId=R01C01 position=21
%compLoc: compId=SV12345002 containerId=R01C01 position=25
%compLoc: compId=SV12345006 containerId=R01C01 position=33
%compLoc: compId=SV12345004 containerId=R01C01 position=37

$ mmchcomploc -F comploc.stanza
$ mmlscomploc
Component           Location
------------------  ------------------
181880E-SV12345007  Rack R01C01 U01-04
181880E-SV12345005  Rack R01C01 U05-08
181880E-SV12345003  Rack R01C01 U13-16
181880E-SV12345008  Rack R01C01 U17-20
181880E-SV12345001  Rack R01C01 U21-24
181880E-SV12345002  Rack R01C01 U25-28
181880E-SV12345006  Rack R01C01 U33-36
181880E-SV12345004  Rack R01C01 U37-40
```

Synchronizing display IDs:

The function described in this topic is only available with the IBM System x GPFS Storage Server.

Note: Sample output in this topic may not match the output produced on your system.

The GSS disk enclosures have a two-digit display on their ESMs. This display is visible from the back of the enclosure. When configured correctly, the two-digit display will be the same as the U location of the enclosure within its rack. **mmsyncdisplayid** sets the display on the ESMs to match the locations of the enclosure as defined by the component configuration. Here is an example:

```
$ mmsyncdisplayid all
$ mmlscomp --type storageenclosure

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  181880E      SV12345001     181880E-SV12345001  21
      2  181880E      SV12345007     181880E-SV12345007  01
      3  181880E      SV12345003     181880E-SV12345003  13
      4  181880E      SV12345005     181880E-SV12345005  05
      5  181880E      SV12345004     181880E-SV12345004  37
      6  181880E      SV12345002     181880E-SV12345002  25
      7  181880E      SV12345008     181880E-SV12345008  17
      8  181880E      SV12345006     181880E-SV12345006  33
```

With the display ID set on the actual hardware, you should now verify that the locations defined in the cluster's configuration actually match the enclosure's position in its rack. If you find that the enclosures are not numbered 01, 05, 13, ... starting from the bottom of the rack, then use the **mmchcomploc** command to correct the configuration. Then rerun **mmsyncdisplayid** and recheck the hardware.

Updating component attributes:

The function described in this topic is only available with the IBM System x GPFS Storage Server.

Note: Sample output in this topic may not match the output produced on your system.

You can change some component attributes such as the name or serial number. This example updates the name of the third enclosure so that it includes the product serial number as recommended:

```
$ mmchcomp 3 --name G1E3-SX98760044
$ mmlscomp --type storageenclosure

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  181880E      SV12345001     181880E-SV12345001  21
      2  181880E      SV12345007     181880E-SV12345007  01
      3  181880E      SV12345003     G1E3-SX98760044     13
      4  181880E      SV12345005     181880E-SV12345005  05
      5  181880E      SV12345004     181880E-SV12345004  37
      6  181880E      SV12345002     181880E-SV12345002  25
      7  181880E      SV12345008     181880E-SV12345008  17
      8  181880E      SV12345006     181880E-SV12345006  33
```

The **mmchcomp** *Component* argument may be either the component ID, the component serial number, or the component name. This example uses the component ID (the value 3).

Changing one component at a time can be slow, so you can use a stanza file to make multiple changes with a single command. As an aid, **mmlscomp** can list components in stanza format. You can capture this output into a file, edit the file to make the desired updates, then use the stanza file as input to **mmchcomp**. The following example uses this procedure to rename the remaining enclosures:

```
$ mmlscomp --format stanza > rename.stanza
$ cat rename.stanza
%comp:
  compId=9
  compType=rack
  partNumber=1410HEA

%comp:
  compId=1
  compType=storageEnclosure
  displayId=21
  name='181880E-SV12345001'
  partNumber=181880E
  serialNumber=SV12345001

%comp:
  compId=2
  compType=storageEnclosure
  displayId=1
  name='181880E-SV12345007'
  partNumber=181880E
  serialNumber=SV12345007

%comp:
  compId=3
  compType=storageEnclosure
  displayId=13
  name='G1E3-SX98760044'
  partNumber=181880E
  serialNumber=SV12345003

...
```

Edit `rename.stanza` to define new enclosure names, then apply the changes using **mmchcomp**.

```
$ vi rename.stanza
$ cat rename.stanza
%comp:
  compId=9
  compType=rack
  name=R01C01
  partNumber=1410HEA

%comp:
  compId=1
  compType=storageEnclosure
  displayId=21
  name='G1E5-SX98760048'
  partNumber=181880E
  serialNumber=SV12345001

%comp:
  compId=2
  compType=storageEnclosure
  displayId=1
  name='G1E1-SX98760044'
  partNumber=181880E
  serialNumber=SV12345007

%comp:
  compId=3
  compType=storageEnclosure
  displayId=13
  name='G1E3-SX98760041'
  partNumber=181880E
  serialNumber=SV12345003

...

$ mmchcomp -F comp.stanza
$ mmlscomp --sort name

    Rack Components

Comp ID  Part Number  Serial Number  Name
-------  -----------  -------------  ------
      9  1410HEA                     R01C01

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      2  181880E      SV12345007     G1E1-SX98760044     01
      4  181880E      SV12345005     G1E2-SX98760049     05
      3  181880E      SV12345003     G1E3-SX98760041     13
      7  181880E      SV12345008     G1E4-SX98760036     17
      1  181880E      SV12345001     G1E5-SX98760048     21
      6  181880E      SV12345002     G1E6-SX98760050     25
      8  181880E      SV12345006     G1E7-SX98760039     33
      5  181880E      SV12345004     G1E8-SX98760052     37
```

Overall management of GPFS Native RAID

This section summarizes how to plan and monitor a GPFS Native RAID system. For an example of setting up a GPFS Native RAID system, see "GPFS Native RAID setup and disk replacement on the IBM Power 775 Disk Enclosure" on page 32.

Planning considerations for GPFS Native RAID:

Planning a GPFS Native RAID implementation requires consideration of the nature of the JBOD arrays being used, the required redundancy protection and usable disk capacity, the required spare capacity and maintenance strategy, and the ultimate GPFS file system configuration. This section is a set of best-practice recommendations for using GPFS Native RAID.

- Assign a primary and backup server to each recovery group.

  Each JBOD array should be connected to two servers to protect against server failure. Each server should also have two independent paths to each physical disk to protect against path failure and provide higher throughput to the individual disks.

  Define multiple recovery groups on a JBOD array, if the architecture suggests it, and use mutually reinforcing primary and backup servers to spread the processing evenly across the servers and the JBOD array.

  Recovery group server nodes can be designated GPFS quorum or manager nodes, but they should otherwise be dedicated to GPFS Native RAID and not run application workload.
- Configure recovery group servers with a large vdisk track cache and a large pagepool.

  The **nsdRAIDTracks** configuration parameter tells GPFS Native RAID how many vdisk track descriptors, not including the actual track data, to cache in memory.

  In general, a large number of vdisk track descriptors should be cached. The **nsdRAIDTracks** value for the recovery group servers should be 10000 - 60000. If the expected vdisk NSD access pattern is random across all defined vdisks and within individual vdisks, a larger value for **nsdRAIDTracks** might be warranted. If the expected access pattern is sequential, a smaller value can be sufficient.

  The amount of actual vdisk data (including user data, parity, and checksums) that can be cached depends on the size of the GPFS pagepool on the recovery group servers and the percentage of pagepool reserved for GPFS Native RAID. The **nsdRAIDBufferPoolSizePct** parameter specifies what percentage of the pagepool should be used for vdisk data. The default is 50%, but it can be set as high as 90% or as low as 10%. Because a recovery group server is also an NSD server and the vdisk buffer pool also acts as the NSD buffer pool, the configuration parameter **nsdBufSpace** should be reduced to its minimum value of 10%.

  As an example, to have a recovery group server cache 20000 vdisk track descriptors (**nsdRAIDTracks**), where the data size of each track is 4 MiB, using 80% (**nsdRAIDBufferPoolSizePct**) of the pagepool, an approximate pagepool size of 20000 * 4 MiB * (100/80) ≈ 100000 MiB ≈ 98 GiB would be required. It is not necessary to configure the pagepool to cache all the data for every cached vdisk track descriptor, but this example calculation can provide some guidance in determining appropriate values for **nsdRAIDTracks** and **nsdRAIDBufferPoolSizePct**.
- Define each recovery group with at least one large declustered array.

  A large declustered array contains enough pdisks to store the required redundancy of GPFS Native RAID vdisk configuration data. This is defined as at least nine pdisks plus the effective spare capacity. A minimum spare capacity equivalent to two pdisks is strongly recommended in each large declustered array. The code width of the vdisks must also be considered. The effective number of non-spare pdisks must be at least as great as the largest vdisk code width. A declustered array with two effective spares where 11 is the largest code width (8 + 3p Reed-Solomon vdisks) must contain at least 13 pdisks. A declustered array with two effective spares where 10 is the largest code width (8 + 2p Reed-Solomon vdisks) must contain at least 12 pdisks.
- Place the log vdisk in a separate declustered array of solid-state disks (SSDs).

  The SSDs in the JBOD array should be used for the log vdisk of each recovery group. These SSDs should be isolated in a small log declustered array, and the log vdisk should be the only vdisk defined there. One pdisk of spare capacity should be defined, which is the default for a small declustered array. For example, if the log declustered array contains four physical SSDs, it should have one spare defined and the log vdisk should use 3-way replication. The recommended track size for the log vdisk is 1MiB, and the recommended total size is 2 - 4 GiB.

- Determine the declustered array maintenance strategy.

  Disks will fail and need replacement, so a general strategy of deferred maintenance can be used. For example, failed pdisks in a declustered array are only replaced when the spare capacity of the declustered array is exhausted. This is implemented with the replacement threshold for the declustered array set equal to the effective spare capacity. This strategy is useful in installations with a large number of recovery groups where disk replacement might be scheduled on a weekly basis. Smaller installations can have GPFS Native RAID require disk replacement as disks fail, which means the declustered array replacement threshold can be set to one.

- Choose the vdisk RAID codes based on GPFS file system usage.

  The choice of vdisk RAID codes depends on the level of redundancy protection required versus the amount of actual space required for user data, and the ultimate intended use of the vdisk NSDs in a GPFS file system.

  Reed-Solomon vdisks are more space efficient. An 8 + 3p vdisk uses approximately 27% of actual disk space for redundancy protection and 73% for user data. An 8 + 2p vdisk uses 20% for redundancy and 80% for user data. Reed-Solomon vdisks perform best when writing whole tracks (the GPFS block size) at once. When partial tracks of a Reed-Solomon vdisk are written, parity recalculation must occur.

  Replicated vdisks are less space efficient. A vdisk with 3-way replication uses approximately 67% of actual disk space for redundancy protection and 33% for user data. A vdisk with 4-way replication uses 75% of actual disk space for redundancy and 25% for user data. The advantage of vdisks with *N*-way replication is that small or partial write operations can complete faster.

  For file system applications where write performance must be optimized, the preceding considerations make replicated vdisks most suitable for use as GPFS file system **metadataOnly** NSDs, and Reed-Solomon vdisks most suitable for use as GPFS file system **dataOnly** NSDs. The volume of GPFS file system metadata is usually small (1% - 3%) relative to file system data, so the impact of the space inefficiency of a replicated RAID code is minimized. The file system metadata is typically written in small chunks, which takes advantage of the faster small and partial write operations of the replicated RAID code. Applications are often tuned to write file system user data in whole multiples of the file system block size, which works to the strengths of the Reed-Solomon RAID codes both in terms of space efficiency and speed.

  When segregating vdisk NSDs for file system **metadataOnly** and **dataOnly** disk usage, the **metadataOnly** replicated vdisks can be created with a smaller block size and assigned to the GPFS file system storage pool. The **dataOnly** Reed-Solomon vdisks can be created with a larger block size and assigned to GPFS file system data storage pools. When using multiple storage pools, a GPFS placement policy must be installed to direct file system data to non-system storage pools.

  When write performance optimization is not important, it is acceptable to use Reed-Solomon vdisks as **dataAndMetadata** NSDs for better space efficiency.

  When assigning the failure groups to vdisk NSDs in a GPFS file system, the JBOD array should be considered the common point of failure. All vdisks within all recovery groups in a given JBOD array should be assigned the same failure group number.

Monitoring GPFS Native RAID:

To monitor GPFS Native RAID during normal operation, use the **mmlsrecoverygroup**, **mmlspdisk**, and **mmpmon** commands. Pay particular attention to the GPFS Native RAID event log, which is visible using the **mmlsrecoverygroupevents** command.

Consider using GPFS Native RAID user exits to notify an automated system management tool if critical events, such as disk failures, occur during normal operation. For more information, see the **mmaddcallback** command in the *GPFS: Administration and Programming Reference*.

If disk maintenance is indicated, use the **mmchcarrier** command to release the failed disk, replace the failed drive, and use the **mmchcarrier** command again to inform GPFS Native RAID that the failed disk has been replaced.

Displaying vdisk I/O statistics:

To display vdisk I/O statistics, run **mmpmon** with the following command included in the input file:

```
vio_s [f [rg RecoveryGroupName [ da DeclusteredArrayName [ v VdiskName]]]] [reset]
```

This request returns strings containing vdisk I/O statistics as seen by that node. The values are presented as total values for the node, or they can be filtered with the **f** option. The reset option indicates that the statistics should be reset after the data is sampled.

If the **-p** option is specified when running **mmpmon**, the vdisk I/O statistics are provided in the form of keywords and values in the **vio_s** response. Table 4 lists and describes these keywords in the order in which they appear in the output.

Table 4. Keywords and descriptions of values provided in the **mmpmon vio_s** *response*

Keyword	Description
_n_	The IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The name by which GPFS knows the node.
_rc_	The reason or error code. In this case, the reply value is 0 (OK).
_t_	The current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	The microseconds part of the current time of day.
_rg_	The name of the recovery group.
_da_	The name of the declustered array.
_r_	The total number of read operations.
_sw_	The total number of short write operations.
_mw_	The total number of medium write operations.
pfw	The total number of promoted full track write operations.
ftw	The total number of full track write operations.
fuw	The total number of flushed update write operations.
fpw	The total number of flushed promoted full track write operations.
_m_	The total number of migrate operations.
_s_	The total number of scrub operations.
_l_	The total number log write operations.
_fc_	The total number of force consistency operations.
fix	The total number of buffer range fix operations.
ltr	The total number of log tip read operations.
lhr	The total number of log home read operations.
rgd	The total number of recovery group descriptor write operations.
meta	The total number metadata block write operations.

To display these statistics, use the sample script `/usr/lpp/mmfs/samples/vdisk/viostat`. The following shows the usage of the `viostat` script:

```
viostat  [-F NodeFile | [--recovery-group RecoveryGroupName
                        [--declustered-array  DeclusteredArrayName
                        [--vdisk VdiskName]]]]
         [Interval [Count]]
```

| For more information, see the following *GPFS: Advanced Administration* topic:

- Monitoring GPFS I/O performance with the mmpmon command (not included in this *Documentation Update*)

.

GPFS Native RAID callbacks:

GPFS Native RAID introduces 12 new GPFS callbacks for events that can occur during recovery group operations. These callbacks can be installed by the system administrator using the **mmaddcallback** command.

The callbacks are provided primarily as a method for system administrators to take notice when important GPFS Native RAID events occur. For example, a GPFS administrator can use the **pdReplacePdisk** callback to send an e-mail to notify system operators that the replacement threshold for a declustered array was reached and that pdisks must be replaced. Similarly, the **preRGTakeover** callback can be used to inform system administrators of a possible server failover.

As notification methods, no real processing should occur in the callback scripts. GPFS Native RAID callbacks should not be installed for synchronous execution; the default of asynchronous callback execution should be used in all cases. Synchronous or complicated processing within a callback might delay GPFS daemon execution pathways and cause unexpected and undesired results, including loss of file system availability.

Table 5 lists the callbacks and their corresponding parameters available through the **mmaddcallback** command:

Table 5. GPFS Native RAID callbacks and parameters

Callbacks	Parameters
preRGTakeover	**myNode**, **rgName**, **rgErr**, **rgCount**, **rgReason**
postRGTakeover	**myNode**, **rgName**, **rgErr**, **rgCount**, **rgReason**
preRGRelinquish	**myNode**, **rgName**, **rgErr**, **rgCount**, **rgReason**
postRGRelinquish	**myNode**, **rgName**, **rgErr**, **rgCount**, **rgReason**
rgOpenFailed	**myNode**, **rgName**, **rgErr**, **rgReason**
rgPanic	**myNode**, **rgName**, **rgErr**, **rgReason**
pdFailed	**myNode**, **rgName**, **daName**, **pdName**, **pdLocation**, **pdFru**, **pdWwn**, **pdState**
pdRecovered	**myNode**, **rgName**, **daName**, **pdName**, **pdLocation**, **pdFru**, **pdWwn**
pdReplacePdisk	**myNode**, **rgName**, **daName**, **pdName**, **pdLocation**, **pdFru**, **pdWwn**, **pdState**, **pdPriority**
pdPathDown	**myNode**, **rgName**, **daName**, **pdName**, **pdPath**, **pdLocation**, **pdFru**, **pdWwn**
daRebuildFailed	**myNode**, **rgName**, **daName**, **daRemainingRedundancy**
nsdCksumMismatch	**myNode**, **ckRole**, **ckOtherNode**, **ckNSD**, **ckReason**, **ckStartSector**, **ckDataLen**, **ckErrorCountClient**, **ckErrorCountServer**, **ckErrorCountNSD**, **ckReportingInterval**

All GPFS Native RAID callbacks are local, which means that the event triggering the callback occurs only on the involved node or nodes, in the case of **nsdCksumMismatch**, rather than on every node in the GPFS cluster. The nodes where GPFS Native RAID callbacks should be installed are, by definition, the recovery group server nodes. An exception is the case of **nsdCksumMismatch**, where it makes sense to install the callback on GPFS client nodes as well as recovery group servers.

For more information about GPFS Native RAID callbacks, see the *GPFS: Administration and Programming Reference* topic "**mmaddcallback** command".

GPFS Native RAID setup and disk replacement on the IBM Power 775 Disk Enclosure

Example scenario: Configuring GPFS Native RAID recovery groups

This topic provides a detailed example of configuring GPFS Native RAID using the JBOD SAS disks on the Power 775 Disk Enclosure. The example considers one fully populated Power 775 Disk Enclosure cabled to two recovery group servers, and shows how the architecture of the Power 775 Disk Enclosure determines the structure of the recovery groups. Throughout this topic, it may be helpful to have Power 775 Disk Enclosure documentation at hand.

Preparing recovery group servers:

Disk enclosure and HBA cabling

The Power 775 Disk Enclosure should be cabled to the intended recovery group servers according to the Power 775 Disk Enclosure hardware installation instructions. The fully populated Power 775 Disk Enclosure consists of 8 STORs of 48 disks, for a total of 384 JBOD disks. Each STOR provides redundant left and right port cards for host server HBA connections (STOR is short for *physical storage group*, meaning the part of the disk enclosure controlled by a pair of port cards). To ensure proper multi-pathing and redundancy, each recovery group server must be connected to each port card using different HBAs. For example, STOR 1 has port cards P1-C4 and P1-C5. Server 1 may be connected to P1-C4 using HBA `hba1` and to P1-C5 using HBA `hba2`; similarly for server 2 and its respective HBAs `hba1` and `hba2`.

GPFS Native RAID provides system administration tools for verifying the correct connectivity of the Power 775 Disk Enclosure, which will be seen later during the operating system preparation.

When the port cards of the Power 775 Disk Enclosure have been cabled to the appropriate HBAs of the two recovery group servers, the Power 775 Disk Enclosure should be powered on and the recovery group servers should be rebooted.

Initial operating system verification

Preparation then continues with the operating system, which must be either AIX 7.1 or Red Hat Enterprise Linux 6.1, and which must be the same on both recovery group servers. It is not necessary to do a complete verification of the Power 775 Disk Enclosure connectivity at this point. Logging in to the servers to perform a quick check that at least some disks have been detected and configured by the operating system will suffice. The operating system device configuration should be examined for the Power 775 Disk Enclosure VPD enclosure type, which is 78AD.001.

One way to quickly verify that AIX has configured devices with enclosure type 78AD.001 for the Power 775 Disk Enclosure is:

```
# lsdev -t ses -F 'name physloc parent' | grep 78AD.001
```

The output should include lines resembling the following:

```
ses12 U78AD.001.000DE37-P1-C4 sas3
```

This is the SAS expander device on port card P1-C4 of the Power 775 Disk Enclosure with serial number 000DE37, together with the SAS protocol device driver `sas3` under which it has been configured. To see what disks have been detected by the SAS protocol driver, use:

```
# lsdev -p sas3
```

The output should include all the disks and port card expanders that successfully configured under the sas3 SAS protocol driver (which corresponds to the HBA device mpt2sas3).

If AIX has not configured any port card expanders of enclosure type 78AD.001, the hardware installation of the server HBAs and the Power 775 Disk Enclosure must be reexamined and corrected.

One way to quickly verify that Red Hat Enterprise Linux has configured devices with enclosure type 78AD.001 for the Power 775 Disk Enclosure is:

```
# grep 78AD.001 /proc/scsi/scsi
```

The output should include lines resembling the following:

```
Vendor: IBM      Model: 78AD-001         Rev: 0150

Further examination of the /proc/scsi/scsi file should reveal contents similar to:
Host: scsi7 Channel: 00 Id: 394 Lun: 00
  Vendor: IBM      Model: ST9600204SS      Rev: 631C
  Type:   Direct-Access                    ANSI  SCSI revision: 06
Host: scsi7 Channel: 00 Id: 395 Lun: 00
  Vendor: IBM      Model: 78AD-001         Rev: 0150
  Type:   Enclosure                        ANSI  SCSI revision: 04
```

The above indicates that a Power 775 Disk Enclosure port card SAS expander and a disk drive have been configured on SCSI host bus 7 (the HBA corresponding to scsi7).

As with AIX, if Linux has not configured any port card expanders of enclosure type 78AD.001, the hardware installation of the server HBAs and the Power 775 Disk Enclosure must be reexamined and corrected.

Disabling operating system multi-pathing

Once it has been verified that at least some of the Power 775 Disk Enclosure has been configured by the operating system, the next step is to disable any operating system multi-pathing. Since GPFS Native RAID performs its own disk multi-pathing, AIX MPIO (Multiple Path I/O) and Linux DMM (Device Mapper Multipath) must be disabled as appropriate.

To disable AIX MPIO for SAS disks, use:

```
# manage_disk_drivers -d SAS_SCSD -o AIX_non_MPIO
```

To disable Linux DMM, use:

```
# chkconfig --del multipathd
```

Note: This blanket disabling of operating system multi-pathing is appropriate because a Power 775 Disk Enclosure installation provides the only available disk devices to the recovery group servers. Once operating system multi-pathing has been disabled, the recovery group servers should be rebooted.

Operating system device attributes

For best performance, the operating system disk device driver should be configured to allow GPFS Native RAID I/O operations to be made with one disk access, rather than being fragmented. Under AIX this is controlled by the **max_transfer** attribute of the HBAs and disk devices. Under Red Hat Enterprise Linux, this is controlled by the disk block device **max_sectors_kb** attribute.

The disk I/O size performed by GPFS Native RAID depends on the strip size of the RAID code of the vdisk NSD. This in turn is related to the vdisk track size and its corresponding GPFS file system block size. The operating system I/O size should be equal to or greater than the largest strip size of the planned vdisk NSDs.

Because GPFS Native RAID stores checksums with each strip, strips have an additional 4 KiB or 8 KiB than might be expected just from the user data (strips containing 2 MiB of user data have an additional 8 KiB; all smaller strips have an additional 4 KiB). The strip size for a replicated vdisk RAID code is equal to the vdisk track size plus the size of the checksum. The strip size for a Reed-Solomon vdisk RAID code is equal to one-eighth of the vdisk track size plus the size of the checksum.

The default **max_transfer** value of 1 MiB under AIX is suitable for GPFS Native RAID vdisk strip sizes under 1 MiB.

The default **max_sectors_kb** value of 512 KiB sectors under Red Hat Enterprise Linux is suitable for GPFS Native RAID vdisk strip sizes under 512 KiB.

However, for vdisk strip sizes greater than 1 MiB under AIX or greater than 512 KiB under Linux, the operating system disk device driver I/O size should be increased for best performance.

The following table indicates the relationship between file system NSD block size, vdisk track size, vdisk RAID code, vdisk strip size, and the non-default operating system I/O size for all permitted GPFS Native RAID vdisks. The AIX **max_transfer** attribute is specified in hexadecimal, and the only allowable values greater than the 1 MiB default are 0x200000 (2 MiB) and 0x400000 (4 MiB). Linux **max_sectors_kb** is specified as the desired I/O size in KiB.

Table 6. NSD block size, vdisk track size, vdisk RAID code, vdisk strip size, and non-default operating system I/O size for permitted GPFS Native RAID vdisks

NSD block size	vdisk track size	vdisk RAID code	RAID code strip size	AIX max_transfer	Linux max_sectors_kb
256 KiB	256 KiB	3- or 4-way replication	260 KiB	default	260
512 KiB	512 KiB	3- or 4-way replication	516 KiB	default	516
1 MiB	1 MiB	3- or 4-way replication	1028 KiB	0x200000	1028
2 MiB	2 MiB	3- or 4-way replication	2056 KiB	0x400000	2056
512 KiB	512 KiB	8 + 2p or 8 + 3p	68 KiB	default	default
1 MiB	1 MiB	8 + 2p or 8 + 3p	132 KiB	default	default
2 MiB	2 MiB	8 + 2p or 8 + 3p	260 KiB	default	260
4 MiB	4 MiB	8 + 2p or 8 + 3p	516 KiB	default	516
8 MiB	8 MiB	8 + 2p or 8 + 3p	1028 KiB	0x200000	1028
16 MiB	16 MiB	8 + 2p or 8 + 3p	2056 KiB	0x400000	2056

If the largest strip size of all the vdisk NSDs planned for a GPFS Native RAID installation exceeds the operating system default I/O size, the operating system I/O size should be changed.

AIX Under AIX, this involves changing the HBA and disk device **max_transfer** size. For an HBA to accommodate an increased **max_transfer** size, the **max_commands** for the HBA will also need to be decreased. With a 0x400000 **max_transfer** size, the four-port HBA requires a **max_commands** value of 124 and the two-port HBA requires a **max_commands** value of 248.

To change the **max_transfer** attribute to 4 MiB for the HBA `mpt2sas0` under AIX, use the following command:

```
# chdev -P -l mpt2sas0 -a max_transfer=0x400000
```

To change the **max_commands** value to 124 for the four-port HBA `mpt2sas0` (AIX device type 001072001410f60), use the following command:

```
# chdev -P -l mpt2sas0 -a max_commands=124
```

To change the **max_commands** value to 248 for the two-port HBA `mpt2sas0` (AIX device type 001072001410ea0), use the following command:

```
# chdev -P -l mpt2sas0 -a max_commands=248
```

Repeat the previous commands for each HBA.

Changing the hdisk **max_transfer** attribute requires changing its default value for AIX device type nonmpioscsd disks. It is not sufficient to change the **max_transfer** for individual hdisks, because performing disk replacement deletes and recreates hdisk objects.

To change the default hdisk **max_transfer** attribute for type nonmpioscsd hdisks, use the following command:

```
# chdef -a max_transfer=0x400000 -c disk -s sas -t nonmpioscsd
```

The new **max_transfer** and **max_commands** values will not take effect until AIX reconfigures the HBAs and hdisks. This can be done either by rebooting the recovery group server, or by deconfiguring (but not removing) and then reconfiguring the affected HBAs. The **max_transfer** and **max_commands** attribute are recorded in the CuAt ODM class and will persist across reboots.

Linux Under Linux, the **max_sectors_kb** I/O size is reset to the default each time a disk block device is configured. This means that upon reboot and upon adding or replacing disks, a desired nondefault I/O size must be explicitly set. Since **max_sectors_kb** is dynamically reconfigurable, GPFS Native RAID manages this setting under Linux.

The value of **max_sectors_kb** under Linux is set on all the disks in a recovery group when GPFS Native RAID begins serving the recovery group, and on disks that are configured as part of GPFS Native RAID disk replacement.

The default **max_sectors_kb** set by GPFS Native RAID is 4096, which is large enough for any of the strip sizes listed in Table 6 on page 34. It can be changed to exactly match the largest strip size in use by GPFS Native RAID by setting the GPFS **nsdRAIDBlockDeviceMaxSectorsKB** configuration parameter.

To set the **max_sectors_kb** value used by GPFS Native RAID to 2056, use the following command:

```
# mmchconfig nsdRAIDBlockDeviceMaxSectorsKB=2056
```

The new value will take effect the next time GPFS is started. To have the new value take effect immediately, append the **-i** option to the above command.

For optimal performance, additional device attributes may need to be changed (for example, the HBA and block device command queue depths); consult the operating system documentation for the device attributes.

Verifying that a Power 775 Disk Enclosure is configured correctly

Once a superficial inspection indicates that the Power 775 Disk Enclosure has been configured on the recovery group servers, and especially once operating system multi-pathing has been disabled, it is necessary to perform a thorough discovery of the disk topology on each server.

To proceed, GPFS must be installed on the recovery group servers, and they should be members of the same GPFS cluster. Consult the *GPFS: Administration and Programming Reference* for instructions for creating a GPFS cluster.

GPFS Native RAID provides tools in /usr/lpp/mmfs/samples/vdisk for collecting and collating information on any attached Power 775 Disk Enclosure and for verifying that the detected topology is correct. The **mmgetpdisktopology** command examines the operating system's list of connected devices and produces a colon-delimited database with a line for each discovered Power 775 Disk Enclosure physical disk, port card expander device, and HBA. **mmgetpdisktopology** should be run on each of the two intended recovery group server nodes, and the results examined to verify that the disk enclosure hardware and software configuration is as expected. An additional tool called **topsummary** concisely summarizes the output of the **mmgetpdisktopology** command.

Create a directory in which to work, and then capture the output of the **mmgetpdisktopology** command from each of the two intended recovery group server nodes:

```
# mkdir p7ihde
# cd p7ihde
# ssh server1 /usr/lpp/mmfs/samples/vdisk/mmgetpdisktopology > server1.top
# ssh server2 /usr/lpp/mmfs/samples/vdisk/mmgetpdisktopology > server2.top
```

Then view the summary for each of the nodes (server1 example shown):

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top
P7IH-DE enclosures found: DE00022
Enclosure DE00022:
Enclosure DE00022 STOR P1-C4/P1-C5 sees both portcards: P1-C4 P1-C5
Portcard P1-C4: ses0[0150]/mpt2sas0/24 diskset "37993" ses1[0150]/mpt2sas0/24 diskset "18793"
Portcard P1-C5: ses4[0150]/mpt2sas1/24 diskset "37993" ses5[0150]/mpt2sas1/24 diskset "18793"
Enclosure DE00022 STOR P1-C4/P1-C5 sees 48 disks
Enclosure DE00022 STOR P1-C12/P1-C13 sees both portcards: P1-C12 P1-C13
Portcard P1-C12: ses8[0150]/mpt2sas2/24 diskset "40657" ses9[0150]/mpt2sas2/24 diskset "44382"
Portcard P1-C13: ses12[0150]/mpt2sas3/24 diskset "40657" ses13[0150]/mpt2sas3/24 diskset "44382"
Enclosure DE00022 STOR P1-C12/P1-C13 sees 48 disks
Enclosure DE00022 STOR P1-C20/P1-C21 sees both portcards: P1-C20 P1-C21
Portcard P1-C20: ses16[0150]/mpt2sas4/24 diskset "04091" ses17[0150]/mpt2sas4/24 diskset "31579"
Portcard P1-C21: ses20[0150]/mpt2sas5/24 diskset "04091" ses21[0150]/mpt2sas5/24 diskset "31579"
Enclosure DE00022 STOR P1-C20/P1-C21 sees 48 disks
Enclosure DE00022 STOR P1-C28/P1-C29 sees both portcards: P1-C28 P1-C29
Portcard P1-C28: ses24[0150]/mpt2sas6/24 diskset "64504" ses25[0150]/mpt2sas6/24 diskset "62361"
Portcard P1-C29: ses28[0150]/mpt2sas7/24 diskset "64504" ses29[0150]/mpt2sas7/24 diskset "62361"
Enclosure DE00022 STOR P1-C28/P1-C29 sees 48 disks
Enclosure DE00022 STOR P1-C60/P1-C61 sees both portcards: P1-C60 P1-C61
Portcard P1-C60: ses30[0150]/mpt2sas7/24 diskset "10913" ses31[0150]/mpt2sas7/24 diskset "52799"
Portcard P1-C61: ses26[0150]/mpt2sas6/24 diskset "10913" ses27[0150]/mpt2sas6/24 diskset "52799"
Enclosure DE00022 STOR P1-C60/P1-C61 sees 48 disks
Enclosure DE00022 STOR P1-C68/P1-C69 sees both portcards: P1-C68 P1-C69
Portcard P1-C68: ses22[0150]/mpt2sas5/24 diskset "50112" ses23[0150]/mpt2sas5/24 diskset "63400"
Portcard P1-C69: ses18[0150]/mpt2sas4/24 diskset "50112" ses19[0150]/mpt2sas4/24 diskset "63400"
Enclosure DE00022 STOR P1-C68/P1-C69 sees 48 disks
Enclosure DE00022 STOR P1-C76/P1-C77 sees both portcards: P1-C76 P1-C77
Portcard P1-C76: ses14[0150]/mpt2sas3/23 diskset "45948" ses15[0150]/mpt2sas3/24 diskset "50856"
Portcard P1-C77: ses10[0150]/mpt2sas2/24 diskset "37258" ses11[0150]/mpt2sas2/24 diskset "50856"
Enclosure DE00022 STOR P1-C76/P1-C77 sees 48 disks
Enclosure DE00022 STOR P1-C84/P1-C85 sees both portcards: P1-C84 P1-C85
Portcard P1-C84: ses6[0150]/mpt2sas1/24 diskset "13325" ses7[0150]/mpt2sas1/24 diskset "10443"
Portcard P1-C85: ses2[0150]/mpt2sas0/24 diskset "13325" ses3[0150]/mpt2sas0/24 diskset "10443"
Enclosure DE00022 STOR P1-C84/P1-C85 sees 48 disks
Carrier location P1-C79-D4 appears only on the portcard P1-C77 path
Enclosure DE00022 sees 384 disks

mpt2sas7[1005470001] U78A9.001.9998884-P1-C1 DE00022 STOR 4 P1-C29 (ses28 ses29) STOR 5 P1-C60 (ses30 ses31)
mpt2sas6[1005470001] U78A9.001.9998884-P1-C3 DE00022 STOR 4 P1-C28 (ses24 ses25) STOR 5 P1-C61 (ses26 ses27)
mpt2sas5[1005470001] U78A9.001.9998884-P1-C5 DE00022 STOR 3 P1-C21 (ses20 ses22) STOR 6 P1-C68 (ses21 ses23)
mpt2sas4[1005470001] U78A9.001.9998884-P1-C7 DE00022 STOR 3 P1-C20 (ses16 ses17) STOR 6 P1-C69 (ses18 ses19)
```

```
mpt2sas3[1005470001] U78A9.001.9998884-P1-C9 DE00022 STOR 2 P1-C13 (ses12 ses13) STOR 7 P1-C76 (ses14 ses15)
mpt2sas2[1005470001] U78A9.001.9998884-P1-C11 DE00022 STOR 2 P1-C12 (ses8 ses9) STOR 7 P1-C77 (ses10 ses11)
mpt2sas1[1005470001] U78A9.001.9998884-P1-C13 DE00022 STOR 1 P1-C5 (ses4 ses5) STOR 8 P1-C84 (ses6 ses7)
mpt2sas0[1005470001] U78A9.001.9998884-P1-C15 DE00022 STOR 1 P1-C4 (ses0 ses1) STOR 8 P1-C85 (ses2 ses3)
```

In the preceding output, the Power 775 Disk Enclosure with serial number DE00022 is discovered, together with its eight individual STORs and the component port cards, port card expanders (with their firmware levels in brackets), and physical disks. One minor discrepancy is noted: The physical disk in location P1-C79-D4 is only seen over one of the two expected HBA paths. This can also be seen in the output for the STOR with port cards P1-C76 and P1-C77:

```
Enclosure DE00022 STOR P1-C76/P1-C77 sees both portcards: P1-C76 P1-C77
Portcard P1-C76: ses14[0150]/mpt2sas3/23 diskset "45948" ses15[0150]/mpt2sas3/24 diskset "50856"
Portcard P1-C77: ses10[0150]/mpt2sas2/24 diskset "37258" ses11[0150]/mpt2sas2/24 diskset "50856"
Enclosure DE00022 STOR P1-C76/P1-C77 sees 48 disks
```

Here the connection through port card P1-C76 sees just 23 disks on the expander `ses14` and all 24 disks on the expander `ses15`, while the connection through port card P1-C77 sees all 24 disks on each of the expanders `ses10` and `ses11`. The “disksets” that are reached over the expanders are identified by a checksum of the unique SCSI WWNs of the physical disks that are present; equal disksets represent the same collection of physical disks.

The preceding discrepancy can either be corrected or ignored, as it is probably due to a poorly seated or defective port on the physical disk. The disk is still present on the other port.

If other discrepancies are noted (for example, physical disks that are expected but do not show up at all, or SSDs or HDDs in the wrong locations), they should be corrected before proceeding.

The HBAs (firmware levels in brackets) are also listed with their slot location codes to show the cabling pattern. Each HBA sees two STORs, and each STOR is seen by two different HBAs, which provides the multiple paths and redundancy required by a correct Power 775 Disk Enclosure installation.

This output can be compared to the hardware cabling specification to verify that the disk enclosure is connected correctly.

The `server2.top` topology database should also be examined with the **topsummary** sample script and verified to be correct.

Once the Power 775 Disk Enclosure topologies are verified to be correct on both intended recovery group server nodes, the recommended recovery group configuration can be created using GPFS Native RAID commands.

Creating recovery groups on a Power 775 Disk Enclosure:

Configuring GPFS nodes to be recovery group servers

Before a GPFS node can create and serve recovery groups, it must be configured with a vdisk track cache. This is accomplished by setting the **nsdRAIDTracks** configuration parameter.

nsdRAIDTracks is the GPFS configuration parameter essential to define a GPFS cluster node as a recovery group server. It specifies the number of vdisk tracks of which the attributes will be held in memory by the GPFS daemon on the recovery group server.

The actual contents of the vdisk tracks, the user data and the checksums, are stored in the standard GPFS pagepool. Therefore, the size of the GPFS pagepool configured on a recovery group server should be considerable, on the order of tens of gigabytes. The amount of pagepool dedicated to hold vdisk track data is governed by the **nsdRAIDBufferPoolSizePct** parameter, which defaults to 50%. In practice, a recovery group server will not need to use the GPFS pagepool for any significant amount of standard file caching, and the **nsdRAIDBufferPoolSizePct** value can be increased to 80%. Also applicable, since a recovery group server is by definition an NSD server, is the **nsdBufSpace** parameter, which defaults to 30% of pagepool. Since the vdisk buffer pool doubles as the NSD buffer spool, the **nsdBufSpace** parameter should be decreased to its minimum of 10%. Together these values leave only 10% of the pagepool for application program file cache, but this should not be a problem as a recovery group server should not be running application programs.

In this example, the recovery group servers will be configured to cache the information on 16384 vdisk tracks and to have 64 GiB of pagepool, of which 80% will be used for vdisk data. Once the configuration changes are made, the servers will need to be restarted.

```
# mmchconfig nsdRAIDTracks=16384,nsdRAIDBufferPoolSizePct=80,nsdBufSpace=10,pagepool=64G -N server1,server2
# mmshutdown -N server1,server2
# mmstartup -N server1,server2
```

Defining the recovery group layout

The definition of recovery groups on a Power 775 Disk Enclosure is dictated by the architecture and cabling of the disk enclosure. Two servers sharing a Power 775 Disk Enclosure implies two recovery groups; one is served by one node and one by the other, and each server acts as the other's backup. Half the disks in each STOR should belong to one recovery group, and half to the other. One recovery group will therefore be defined on the disks and carriers in the top halves of the eight STORs, and one on the bottom halves. Since the disks in a STOR are placed four to a removable carrier, thereby having a common point of failure, each disk in a carrier should belong to one of four different declustered arrays. Should a carrier fail or be removed, then each declustered array will only suffer the loss of one disk. There are four SSDs distributed among the top set of carriers, and four in the bottom set of carriers. These groups of four SSDs will make up the vdisk log declustered arrays in their respective halves.

GPFS Native RAID provides a tool that understands the layout of the Power 775 Disk Enclosure and will automatically generate the **mmcrrecoverygroup** stanza files for creating the top and bottom recovery groups. **/usr/lpp/mmfs/samples/vdisk/mkp7rginput**, when supplied with output of the **mmgetpdisktopology** command, will create recovery group stanza files for the top and bottom halves of each Power 775 Disk Enclosure found in the topology.

Each recovery group server, though it may see the same functional disk enclosure topology, will almost certainly differ in the particulars of which disk device names (e.g., `/dev/rhdisk77` on AIX or `/dev/sdax` on Linux) refer to which physical disks in what disk enclosure location.

There are two possibilities then for creating the recovery group stanza files and the recovery groups themselves:

Alternative 1:

Generate the recovery group stanza files and create the recovery groups from the perspective of just one of the servers as if that server were to be primary for both recovery groups, and then use the **mmchrecoverygroup** command to swap the primary and backup servers for one of the recovery groups

Alternative 2:

Generate the recovery group stanza files for each server's primary recovery group using the primary server's topology file.

This example will show both alternatives.

Creating the recovery groups, alternative 1

To create the recovery group input stanza files from the perspective of `server1`, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkp7rginput server1.top
```

This will create two files for each disk enclosure present in the server1 topology; in this case, DE00022TOP.server1 for the top half of disk enclosure DE00022 and DE00022BOT.server2 for the bottom half. (An extra file, DEXXXXXbad, may be created if any discrepancies are present in the topology; if such a file is created by **mkp7rginput**, it should be examined and the discrepancies corrected.)

The recovery group stanza files will follow the recommended best practice for the Power 775 Disk Enclosure of defining in each half of the disk enclosure a separate declustered array of 4 SSDs for recovery group transaction logging, and four file system data declustered arrays using the regular HDDs according to which of the four disk enclosure carrier slots each HDD resides in.

The defaults are accepted for other recovery group declustered array parameters such as scrub duration, spare space, and disk replacement policy.

The stanza file will look something like this:

```
# head DE00022TOP.server1
%pdisk: pdiskName=c081d1
        device=/dev/hdisk10
        da=DA1
%pdisk: pdiskName=c065d1
        device=/dev/hdisk211
        da=DA1
%pdisk: pdiskName=c066d1
        device=/dev/hdisk259
        da=DA1
%pdisk: pdiskName=c067d1
```

All the pdisk stanzas for declustered array DA1 will be listed first, followed by those for DA2, DA3, DA4, and the LOG declustered array. The pdisk names will indicate the carrier and disk location in which the physical disk resides. Notice that only one block device path to the disk is given; the second path will be discovered automatically soon after the recovery group is created.

Now that the DE00022TOP.server1 and DE00022BOT.server1 stanza files have been created from the perspective of recovery group server node server1, these two recovery groups can be created using two separate invocations of the **mmcrrecoverygroup** command:

```
# mmcrrecoverygroup DE00022TOP -F DE00022TOP.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmcrrecoverygroup DE00022BOT -F DE00022BOT.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Note that both recovery groups were created with server1 as primary and server2 as backup. It is now necessary to swap the primary and backup servers for DE00022BOT using the **mmchrecoverygroup** command:

```
# mmchrecoverygroup DE00022BOT --servers server2,server1
mmchrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

GPFS Native RAID will automatically discover the appropriate disk devices on server2.

Creating the recovery groups, alternative 2

To create the recovery groups from the start with the intended primary and backup servers, the stanza files from both server topologies will need to be created.

To create the server1 recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkp7rginput server1.top
```

To create the server2 recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkp7rginput server2.top
```

These two commands will result in four stanza files: DE00022TOP.server1, DE00022BOT.server1, DE00022TOP.server2, and DE00022BOT.server2. (As in alternative 1, if any files named DEXXXXXbad are created, they should be examined and the errors within should be corrected.)

The DE00022TOP recovery group must then be created using server1 as the primary and the DE00022TOP.server1 stanza file. The DE00022BOT recovery group must be created using server2 as the primary and the DE00022BOT.server2 stanza file.

```
# mmcrrecoverygroup DE00022TOP -F DE00022TOP.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmcrrecoverygroup DE00022BOT -F DE00022BOT.server2 --servers server2,server1
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Since each recovery group was created using the intended primary server and the stanza file for that server, it is not necessary to swap the primary and backup servers.

Verifying recovery group creation

Use the **mmlsrecoverygroup** command to verify that each recovery group was created:

```
# mmlsrecoverygroup DE00022TOP -L

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 DE00022TOP                   5       0     192

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            0      47       2          2      24 TiB   14 days  inactive    0%  low
 DA2          no            0      47       2          2      24 TiB   14 days  inactive    0%  low
 DA3          no            0      47       2          2      24 TiB   14 days  inactive    0%  low
 DA4          no            0      47       2          2      24 TiB   14 days  inactive    0%  low
 LOG          no            0       4       1          1     558 GiB   14 days  inactive    0%  low

                                         declustered
 vdisk               RAID code              array     vdisk size  remarks
 ------------------  ------------------  -----------  ----------  -------

 active recovery group server                     servers
 -----------------------------------------------  -------
 server1                                          server1,server2

# mmlsrecoverygroup DE00022BOT -L

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 DE00022BOT                   5       0     192

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            0      47       2          2      24 TiB   14 days  inactive    0%  low
 DA2          no            0      47       2          2      24 TiB   14 days  inactive    0%  low
 DA3          no            0      47       2          2      24 TiB   14 days  inactive    0%  low
 DA4          no            0      47       2          2      24 TiB   14 days  inactive    0%  low
 LOG          no            0       4       1          1     558 GiB   14 days  inactive    0%  low

                                         declustered
 vdisk               RAID code              array     vdisk size  remarks
 ------------------  ------------------  -----------  ----------  -------
 active recovery group server                     servers
 -----------------------------------------------  -------
 server1                                          server2,server1
```

Notice that the vdisk sections of the newly created recovery groups are empty; the next step is to create the vdisks.

Defining and creating the vdisks

Once the recovery groups are created and being served by their respective servers, it is time to create the vdisks using the **mmcrvdisk** command.

Each recovery group requires a single log vdisk for recording RAID updates and diagnostic information. This is internal to the recovery group, cannot be used for user data, and should be the only vdisk in the LOG declustered array. The log vdisks in this example use 3-way replication in order to fit in the LOG declustered array, which contains 4 SSDs and spare space equivalent to one disk.

Data vdisks are required to be defined in the four data declustered arrays for use as file system NSDs. In this example, each of the declustered arrays for file system data is divided into two vdisks with different characteristics: one using 4-way replication and a 1 MiB block size and a total vdisk size of 250 GiB suitable for file system metadata, and one using Reed-Solomon 8 + 3p encoding and an 16 MiB block size suitable for file system data. The vdisk size is omitted for the Reed-Solomon vdisks, meaning that they will default to use the remaining non-spare space in the declustered array (for this to work, any vdisks with specified total sizes must of course be defined first).

The possibilities for the vdisk creation stanza file are quite great, depending on the number and type of vdisk NSDs required for the number and type of file systems desired, so the vdisk stanza file will need to be created by hand, possibly following a template.

In this example, a single stanza file, mmcrvdisk.DE00022ALL, is used. The single file contains the specifications for all the vdisks in both the DE00022TOP and DE00022BOT recovery groups. Here is what the example stanza file for use with **mmcrvdisk** should look like:

```
# cat mmcrvdisk.DE00022ALL
%vdisk: vdiskName=DE00022TOPLOG
        rg=DE00022TOP
        da=LOG
        blocksize=1m
        size=4g
        raidCode=3WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=DE00022BOTLOG
        rg=DE00022BOT
        da=LOG
        blocksize=1m
        size=4g
        raidCode=3WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=DE00022TOPDA1META
        rg=DE00022TOP
        da=DA1
        blocksize=1m
        size=250g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=22
        pool=system
%vdisk: vdiskName=DE00022TOPDA1DATA
        rg=DE00022TOP
        da=DA1
        blocksize=16m
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=22
        pool=data
%vdisk: vdiskName=DE00022BOTDA1META
        rg=DE00022BOT
        da=DA1
        blocksize=1m
        size=250g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=22
        pool=system
%vdisk: vdiskName=DE00022BOTDA1DATA
        rg=DE00022BOT
        da=DA1
        blocksize=16m
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=22
        pool=data
[DA2, DA3, DA4 vdisks omitted.]
```

Notice how the file system metadata vdisks are flagged for eventual file system usage as **metadataOnly** and for placement in the `system` storage pool, and the file system data vdisks are flagged for eventual **dataOnly** usage in the `data` storage pool. (After the file system is created, a policy will be required to allocate file system data to the correct storage pools; see "Creating the GPFS file system" on page 43.)

Importantly, also notice that block sizes for the file system metadata and file system data vdisks must be specified at this time, may not later be changed, and must match the block sizes supplied to the eventual **mmcrfs** command.

Notice also that the eventual `failureGroup=22` value for the NSDs on the file system vdisks is the same for vdisks in both the `DE00022TOP` and `DE00022BOT` recovery groups. This is because the recovery groups, although they have different servers, still share a common point of failure in the disk enclosure `DE00022`, and GPFS should be informed of this through a distinct failure group designation for each disk enclosure. It is up to the GPFS system administrator to decide upon the failure group numbers for each Power 775 Disk Enclosure in the GPFS cluster.

To create the vdisks specified in the `mmcrvdisk.DE00022ALL` file, use the following **mmcrvdisk** command:

```
# mmcrvdisk -F mmcrvdisk.DE00022ALL
mmcrvdisk: [I] Processing vdisk DE00022TOPLOG
mmcrvdisk: [I] Processing vdisk DE00022BOTLOG
mmcrvdisk: [I] Processing vdisk DE00022TOPDA1META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA1DATA
mmcrvdisk: [I] Processing vdisk DE00022TOPDA2META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA2DATA
mmcrvdisk: [I] Processing vdisk DE00022TOPDA3META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA3DATA
mmcrvdisk: [I] Processing vdisk DE00022TOPDA4META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA4DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA1META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA1DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA2META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA2DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA3META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA3DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA4META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA4DATA
mmcrvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Creation of the vdisks may be verified through the **mmlsvdisk** command (the **mmlsrecoverygroup** command may also be used):

```
# mmlsvdisk
                                                          declustered  block size
 vdisk name          RAID code        recovery group         array      in KiB    remarks
 ------------------  ---------------  ------------------  -----------  ----------  -------
 DE00022BOTDA1DATA   8+3p             DE00022BOT          DA1               16384
 DE00022BOTDA1META   4WayReplication  DE00022BOT          DA1                1024
 DE00022BOTDA2DATA   8+3p             DE00022BOT          DA2               16384
 DE00022BOTDA2META   4WayReplication  DE00022BOT          DA2                1024
 DE00022BOTDA3DATA   8+3p             DE00022BOT          DA3               16384
 DE00022BOTDA3META   4WayReplication  DE00022BOT          DA3                1024
 DE00022BOTDA4DATA   8+3p             DE00022BOT          DA4               16384
 DE00022BOTDA4META   4WayReplication  DE00022BOT          DA4                1024
 DE00022BOTLOG       3WayReplication  DE00022BOT          LOG                1024  log
 DE00022TOPDA1DATA   8+3p             DE00022TOP          DA1               16384
 DE00022TOPDA1META   4WayReplication  DE00022TOP          DA1                1024
 DE00022TOPDA2DATA   8+3p             DE00022TOP          DA2               16384
 DE00022TOPDA2META   4WayReplication  DE00022TOP          DA2                1024
 DE00022TOPDA3DATA   8+3p             DE00022TOP          DA3               16384
 DE00022TOPDA3META   4WayReplication  DE00022TOP          DA3                1024
 DE00022TOPDA4DATA   8+3p             DE00022TOP          DA4               16384
 DE00022TOPDA4META   4WayReplication  DE00022TOP          DA4                1024
 DE00022TOPLOG       3WayReplication  DE00022TOP          LOG                1024  log
```

Creating NSDs from vdisks

The **mmcrvdisk** command rewrites the input file so that it is ready to be passed to the **mmcrnsd** command that creates the NSDs from which GPFS builds file systems. To create the vdisk NSDs, run the **mmcrnsd** command on the rewritten **mmcrvdisk** stanza file:

```
# mmcrnsd -F mmcrvdisk.DE00022ALL
mmcrnsd: Processing disk DE00022TOPDA1META
mmcrnsd: Processing disk DE00022TOPDA1DATA
mmcrnsd: Processing disk DE00022TOPDA2META
mmcrnsd: Processing disk DE00022TOPDA2DATA
mmcrnsd: Processing disk DE00022TOPDA3META
mmcrnsd: Processing disk DE00022TOPDA3DATA
mmcrnsd: Processing disk DE00022TOPDA4META
mmcrnsd: Processing disk DE00022TOPDA4DATA
mmcrnsd: Processing disk DE00022BOTDA1META
mmcrnsd: Processing disk DE00022BOTDA1DATA
mmcrnsd: Processing disk DE00022BOTDA2META
mmcrnsd: Processing disk DE00022BOTDA2DATA
mmcrnsd: Processing disk DE00022BOTDA3META
mmcrnsd: Processing disk DE00022BOTDA3DATA
mmcrnsd: Processing disk DE00022BOTDA4META
mmcrnsd: Processing disk DE00022BOTDA4DATA
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Notice how the recovery group log vdisks are omitted from NSD processing.

The **mmcrnsd** command then once again rewrites the stanza file in preparation for use as input to the **mmcrfs** command.

Creating the GPFS file system

Run the **mmcrfs** command to create the file system:

```
# mmcrfs gpfs -F mmcrvdisk.DE00022ALL -B 16m --metadata-block-size 1m -T /gpfs -A no

The following disks of gpfs will be formatted on node c250f09c01ap05.ppd.pok.ibm.com:
    DE00022TOPDA1META: size 262163456 KB
    DE00022TOPDA1DATA: size 8395522048 KB
    DE00022TOPDA2META: size 262163456 KB
    DE00022TOPDA2DATA: size 8395522048 KB
    DE00022TOPDA3META: size 262163456 KB
    DE00022TOPDA3DATA: size 8395522048 KB
    DE00022TOPDA4META: size 262163456 KB
    DE00022TOPDA4DATA: size 8395522048 KB
    DE00022BOTDA1META: size 262163456 KB
    DE00022BOTDA1DATA: size 8395522048 KB
    DE00022BOTDA2META: size 262163456 KB
    DE00022BOTDA2DATA: size 8395522048 KB
    DE00022BOTDA3META: size 262163456 KB
    DE00022BOTDA3DATA: size 8395522048 KB
    DE00022BOTDA4META: size 262163456 KB
    DE00022BOTDA4DATA: size 8395522048 KB
Formatting file system ...
Disks up to size 2.5 TB can be added to storage pool 'system'.
Disks up to size 79 TB can be added to storage pool 'data'.
Creating Inode File
Creating Allocation Maps
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool 'system'
Formatting Allocation Map for storage pool 'data'
Completed creation of file system /dev/gpfs.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Notice how the 16 MiB data block size is specified with the traditional **-B** parameter and the 1 MiB metadata block size is specified with the **--metadata-block-size** parameter. Since a file system with different metadata and data block sizes requires the use of multiple GPFS storage pools, a file system placement policy is needed to direct user file data to the data storage pool. In this example, the file placement policy is very simple:

```
# cat policy
rule 'default' set pool 'data'
```

The policy must then be installed in the file system using the **mmchpolicy** command:

```
# mmchpolicy gpfs policy -I yes
Validated policy `policy': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules
Policy `policy' installed and broadcast to all nodes.
```

If a policy is not placed in a file system with multiple storage pools, attempts to place data into files will return ENOSPC as if the file system were full.

This file system, built on a Power 775 Disk Enclosure using two recovery groups, two recovery group servers, eight file system metadata vdisk NSDs and eight file system data vdisk NSDs, may now be mounted and placed into service:

```
# mmmount gpfs -a
```

Example scenario: Replacing failed disks in a Power 775 Disk Enclosure recovery group

The scenario presented here shows how to detect and replace failed disks in a recovery group built on a Power 775 Disk Enclosure.

Detecting failed disks in your enclosure

Assume a fully populated Power 775 Disk Enclosure (serial number 000DE37) on which the following two recovery groups are defined:

- 000DE37TOP containing the disks in the top set of carriers
- 000DE37BOT containing the disks in the bottom set of carriers

Each recovery group contains the following:

- one log declustered array (LOG)
- four data declustered arrays (DA1, DA2, DA3, DA4)

The data declustered arrays are defined according to Power 775 Disk Enclosure best practice as follows:

- 47 pdisks per data declustered array
- each member pdisk from the same carrier slot
- default disk replacement threshold value set to 2

The replacement threshold of 2 means that GPFS Native RAID will only require disk replacement when two or more disks have failed in the declustered array; otherwise, rebuilding onto spare space or reconstruction from redundancy will be used to supply affected data.

This configuration can be seen in the output of **mmlsrecoverygroup** for the recovery groups, shown here for 000DE37TOP:

```
# mmlsrecoverygroup 000DE37TOP -L

                     declustered
 recovery group        arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                   5       9     192

 declustered   needs                            replace                scrub      background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       63%  low
 DA2          no            2      47       2          2    3072 MiB   14 days  scrub       19%  low
 DA3          yes           2      47       2          2       0   B   14 days  rebuild-2r  48%  low
 DA4          no            2      47       2          2    3072 MiB   14 days  scrub       33%  low
 LOG          no            1       4       1          1     546 GiB   14 days  scrub       87%  low

                                         declustered
 vdisk               RAID code              array     vdisk size  remarks
 ------------------  ------------------  -----------  ----------  -------
 000DE37TOPLOG       3WayReplication     LOG            4144 MiB  log
 000DE37TOPDA1META   4WayReplication     DA1             250 GiB
 000DE37TOPDA1DATA   8+3p                DA1              17 TiB
 000DE37TOPDA2META   4WayReplication     DA2             250 GiB
 000DE37TOPDA2DATA   8+3p                DA2              17 TiB
 000DE37TOPDA3META   4WayReplication     DA3             250 GiB
 000DE37TOPDA3DATA   8+3p                DA3              17 TiB
 000DE37TOPDA4META   4WayReplication     DA4             250 GiB
 000DE37TOPDA4DATA   8+3p                DA4              17 TiB

 active recovery group server                     servers
 -----------------------------------------------  -------
 server1                                          server1,server2
```

The indication that disk replacement is called for in this recovery group is the value of yes in the needs service column for declustered array DA3.

The fact that DA3 (the declustered array on the disks in carrier slot 3) is undergoing rebuild of its RAID tracks that can tolerate two strip failures is by itself not an indication that disk replacement is required; it merely indicates that data from a failed disk is being rebuilt onto spare space. Only if the replacement threshold has been met will disks be marked for replacement and the declustered array marked as needing service.

GPFS Native RAID provides several indications that disk replacement is required:

- entries in the AIX error report or the Linux syslog
- the GPFS **pdReplacePdisk** callback, which can be configured to run an administrator-supplied script at the moment a pdisk is marked for replacement

- the POWER7® cluster event notification TEAL agent, which can be configured to send disk replacement notices when they occur to the POWER7 cluster EMS
- the output from the following commands, which may be performed from the command line on any GPFS cluster node (see the examples that follow):
  1. **mmlsrecoverygroup** with the **-L** flag shows `yes` in the `needs service` column
  2. **mmlsrecoverygroup** with the **-L** and **--pdisk** flags; this shows the states of all pdisks, which may be examined for the `replace` pdisk state
  3. **mmlspdisk** with the **--replace** flag, which lists only those pdisks that are marked for replacement

Note: Because the output of **mmlsrecoverygroup -L --pdisk** for a fully-populated disk enclosure is very long, this example shows only some of the pdisks (but includes those marked for replacement).

```
# mmlsrecoverygroup 000DE37TOP -L --pdisk

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                   5       9     192

 declustered   needs                            replace                scrub      background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       63%  low
 DA2          no            2      47       2          2    3072 MiB   14 days  scrub       19%  low
 DA3          yes           2      47       2          2       0   B   14 days  rebuild-2r  68%  low
 DA4          no            2      47       2          2    3072 MiB   14 days  scrub       34%  low
 LOG          no            1       4       1          1     546 GiB   14 days  scrub       87%  low

                     number of    declustered
 pdisk              active paths     array     free space  state
 -----------------  ------------  -----------  ----------  -----
 [...]
 c014d1                   2       DA1               23 GiB  ok
 c014d2                   2       DA2               23 GiB  ok
 c014d3                   2       DA3              510 GiB  dead/systemDrain/noRGD/noVCD/replace
 c014d4                   2       DA4               12 GiB  ok
 [...]
 c018d1                   2       DA1               24 GiB  ok
 c018d2                   2       DA2               24 GiB  ok
 c018d3                   2       DA3              558 GiB  dead/systemDrain/noRGD/noVCD/noData/replace
 c018d4                   2       DA4               12 GiB  ok
 [...]
```

The preceding output shows that the following pdisks are marked for replacement:

- `c014d3` in `DA3`
- `c018d3` and `DA3`

The naming convention used during recovery group creation indicates that these are the disks in slot 3 of carriers 14 and 18. To confirm the physical locations of the failed disks, use the **mmlspdisk** command to list information about those pdisks in declustered array `DA3` of recovery group `000DE37TOP` that are marked for replacement:

```
# mmlspdisk 000DE37TOP --declustered-array DA3 --replace
pdisk:
   replacementPriority = 1.00
   name = "c014d3"
   device = "/dev/rhdisk158,/dev/rhdisk62"
   recoveryGroup = "000DE37TOP"
   declusteredArray = "DA3"
   state = "dead/systemDrain/noRGD/noVCD/replace"
   freeSpace = 513745775616
   fru = "74Y4936"
   location = "78AD.001.000DE37-C14-D3"
   WWN = "naa.5000C5001D90E17F"
   server = "server1"
   reads = 986
   writes = 293368
   bytesReadInGiB = 0.826
   bytesWrittenInGiB = 286.696
   IOErrors = 0
   IOTimeouts = 12
   mediaErrors = 0
   checksumErrors = 0
   pathErrors = 0
   timeBadness = 33.278
   dataBadness = 0.000
pdisk:
   replacementPriority = 1.00
   name = "c018d3"
   device = "/dev/rhdisk630,/dev/rhdisk726"
   recoveryGroup = "000DE37TOP"
   declusteredArray = "DA3"
   state = "dead/systemDrain/noRGD/noVCD/noData/replace"
   freeSpace = 599147937792
   fru = "74Y4936"
   location = "78AD.001.000DE37-C18-D3"
   WWN = "naa.5000C5001DC5DF3F"
   server = "server1"
   reads = 1104
   writes = 379053
   bytesReadInGiB = 0.844
   bytesWrittenInGiB = 370.717
   IOErrors = 0
   IOTimeouts = 0
   mediaErrors = 0
   checksumErrors = 10
   pathErrors = 0
   timeBadness = 0.000
   dataBadness = 10.000
```

The preceding location code attributes confirm the pdisk naming convention:

Disk	Location code	Interpretation
pdisk `c014d3`	78AD.001.000DE37-C14-D3	Disk 3 in carrier 14 in the disk enclosure identified by enclosure type 78AD.001 and serial number 000DE37
pdisk `c018d3`	78AD.001.000DE37-C18-D3	Disk 3 in carrier 18 in the disk enclosure identified by enclosure type 78AD.001 and serial number 000DE37

Replacing the failed disks in a Power 775 Disk Enclosure recovery group

Note: In this example, it is assumed that two new disks with the appropriate Field Replaceable Unit (FRU) code, as indicated by the `fru` attribute (`74Y4936` in this case), have been obtained as replacements for the failed pdisks `c014d3` and `c018d3`.

Replacing each disk is a three-step process:

1. Using the **mmchcarrier** command with the **--release** flag to suspend use of the other disks in the carrier and to release the carrier.
2. Removing the carrier and replacing the failed disk within with a new one.
3. Using the **mmchcarrier** command with the **--replace** flag to resume use of the suspended disks and to begin use of the new disk.

GPFS Native RAID assigns a priority to pdisk replacement. Disks with smaller values for the **replacementPriority** attribute should be replaced first. In this example, the only failed disks are in DA3 and both have the same **replacementPriority**.

Disk c014d3 is chosen to be replaced first.

1. To release carrier 14 in disk enclosure 000DE37:

   ```
   # mmchcarrier 000DE37TOP --release --pdisk c014d3
     [I] Suspending pdisk c014d1 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D1.
     [I] Suspending pdisk c014d2 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D2.
     [I] Suspending pdisk c014d3 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D3.
     [I] Suspending pdisk c014d4 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D4.
     [I] Carrier released.

       - Remove carrier.
       - Replace disk in location 78AD.001.000DE37-C14-D3 with FRU 74Y4936.
       - Reinsert carrier.
       - Issue the following command:

           mmchcarrier 000DE37TOP --replace --pdisk 'c014d3'

         Repair timer is running.  Perform the above within 5 minutes
         to avoid pdisks being reported as missing.
   ```

   GPFS Native RAID issues instructions as to the physical actions that must be taken. Note that disks may be suspended only so long before they are declared missing; therefore the mechanical process of physically performing disk replacement must be accomplished promptly.

   Use of the other three disks in carrier 14 has been suspended, and carrier 14 is unlocked. The identify lights for carrier 14 and for disk 3 are on.

2. Carrier 14 should be unlatched and removed. The failed disk 3, as indicated by the internal identify light, should be removed, and the new disk with FRU 74Y4936 should be inserted in its place. Carrier 14 should then be reinserted and the latch closed.

3. To finish the replacement of pdisk c014d3:

   ```
   # mmchcarrier 000DE37TOP --replace --pdisk c014d3
   [I] The following pdisks will be formatted on node server1:
       /dev/rhdisk354
   [I] Pdisk c014d3 of RG 000DE37TOP successfully replaced.
   [I] Resuming pdisk c014d1 of RG 000DE37TOP.
   [I] Resuming pdisk c014d2 of RG 000DE37TOP.
   [I] Resuming pdisk c014d3#162 of RG 000DE37TOP.
   [I] Resuming pdisk c014d4 of RG 000DE37TOP.
   [I] Carrier resumed.
   ```

When the **mmchcarrier --replace** command returns successfully, GPFS Native RAID has resumed use of the other 3 disks. The failed pdisk may remain in a temporary form (indicated here by the name c014d3#162) until all data from it has been rebuilt, at which point it is finally deleted. The new replacement disk, which has assumed the name c014d3, will have RAID tracks rebuilt and rebalanced onto it. Notice that only one block device name is mentioned as being formatted as a pdisk; the second path will be discovered in the background.

This can be confirmed with **mmlsrecoverygroup -L --pdisk**:

```
# mmlsrecoverygroup 000DE37TOP -L --pdisk

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                   5       9     193

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       63%  low
```

```
DA2          no            2      47       2          2    3072 MiB   14 days  scrub       19%  low
DA3          yes           2      48       2          2       0   B   14 days  rebuild-2r  89%  low
DA4          no            2      47       2          2    3072 MiB   14 days  scrub       34%  low
LOG          no            1       4       1          1     546 GiB   14 days  scrub       87%  low

                    number of    declustered
pdisk              active paths     array     free space  state
-----------------  ------------  -----------  ----------  -----
[...]
c014d1                  2        DA1              23 GiB  ok
c014d2                  2        DA2              23 GiB  ok
c014d3                  2        DA3             550 GiB  ok
c014d3#162              0        DA3             543 GiB  dead/adminDrain/noRGD/noVCD/noPath
c014d4                  2        DA4              23 GiB  ok
[...]
c018d1                  2        DA1              24 GiB  ok
c018d2                  2        DA2              24 GiB  ok
c018d3                  2        DA3             558 GiB  dead/systemDrain/noRGD/noVCD/noData/replace
c018d4                  2        DA4              23 GiB  ok
[...]
```

Notice that the temporary pdisk c014d3#162 is counted in the total number of pdisks in declustered array DA3 and in the recovery group, until it is finally drained and deleted.

Notice also that pdisk c018d3 is still marked for replacement, and that DA3 still needs service. This is because GPFS Native RAID replacement policy expects all failed disks in the declustered array to be replaced once the replacement threshold is reached. The replace state on a pdisk is not removed when the total number of failed disks goes under the threshold.

Pdisk c018d3 is replaced following the same process.

1. Release carrier 18 in disk enclosure 000DE37:

   ```
   # mmchcarrier 000DE37TOP --release --pdisk c018d3
     [I] Suspending pdisk c018d1 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D1.
     [I] Suspending pdisk c018d2 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D2.
     [I] Suspending pdisk c018d3 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D3.
     [I] Suspending pdisk c018d4 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D4.
     [I] Carrier released.

       - Remove carrier.
       - Replace disk in location 78AD.001.000DE37-C18-D3 with FRU 74Y4936.
       - Reinsert carrier.
       - Issue the following command:

           mmchcarrier 000DE37TOP --replace --pdisk 'c018d3'

         Repair timer is running.  Perform the above within 5 minutes
         to avoid pdisks being reported as missing.
   ```

2. Unlatch and remove carrier 18, remove and replace failed disk 3, reinsert carrier 18, and close the latch.
3. To finish the replacement of pdisk c018d3:

   ```
   # mmchcarrier 000DE37TOP --replace --pdisk c018d3

     [I] The following pdisks will be formatted on node server1:
         /dev/rhdisk674
     [I] Pdisk c018d3 of RG 000DE37TOP successfully replaced.
     [I] Resuming pdisk c018d1 of RG 000DE37TOP.
     [I] Resuming pdisk c018d2 of RG 000DE37TOP.
     [I] Resuming pdisk c018d3#166 of RG 000DE37TOP.
     [I] Resuming pdisk c018d4 of RG 000DE37TOP.
     [I] Carrier resumed.
   ```

Running **mmlsrecoverygroup** again will confirm the second replacement:

```
# mmlsrecoverygroup 000DE37TOP -L --pdisk

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                   5       9     192

 declustered   needs                          replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       64%  low
 DA2          no            2      47       2          2    3072 MiB   14 days  scrub       22%  low
 DA3          no            2      47       2          2    2048 MiB   14 days  rebalance   12%  low
 DA4          no            2      47       2          2    3072 MiB   14 days  scrub       36%  low
 LOG          no            1       4       1          1     546 GiB   14 days  scrub       89%  low

                     number of    declustered
 pdisk              active paths     array     free space  state
 -----------------  ------------  -----------  ----------  -----
 [...]
 c014d1                  2        DA1              23 GiB  ok
 c014d2                  2        DA2              23 GiB  ok
 c014d3                  2        DA3             271 GiB  ok
 c014d4                  2        DA4              23 GiB  ok
 [...]
 c018d1                  2        DA1              24 GiB  ok
 c018d2                  2        DA2              24 GiB  ok
 c018d3                  2        DA3             542 GiB  ok
 c018d4                  2        DA4              23 GiB  ok
 [...]
```

Notice that both temporary pdisks have been deleted. This is because `c014d3#162` has finished draining, and because pdisk `c018d3#166` had, before it was replaced, already been completely drained (as evidenced by the `noData` flag). Declustered array `DA3` no longer needs service and once again contains 47 pdisks, and the recovery group once again contains 192 pdisks.

GPFS Native RAID setup and maintenance on the IBM System x GPFS Storage Server (GSS)

The functions described in these topics are only available with the IBM System x GPFS Storage Server (GSS).

Updating firmware on enclosures and drives

After creating a GPFS cluster, you can configure the GSS.

The configuration process requires that you upgrade the enclosure and drive firmware.

Note: Before upgrading the enclosure and drive firmware, ensure that the `gpfs.gss.firmware` rpm has been installed on all GSS servers. `gpfs.gss.firmware` is a standard rpm that can be installed using the **rpm** command. It contains the latest updates of the following types of supported firmware for a GSS configuration:

- System BIOS updates
- Adapter firmware
- Enclosure firmware
- Driver firmware
- Firmware loading tools

To upgrade the enclosure firmware, follow these steps:

1. To confirm that the enclosures have appropriate redundancy and that firmware updates have been found, issue this command:

   ```
   mmchfirmware --type storage-enclosure -N Node[,Node...] --dry-run
   ```

2. To load the firmware, issue this command:

   Note: If the existing firmware levels are less than 0375 (as shown by the **mmlsfirmware --type storage-enclosure** command issued in the previous step), include the **--stop-on-error no** option.

   ```
   mmchfirmware --type storage-enclosure -N Node[,Node...][--stop-on-error no]
   ```

To upgrade the drive firmware, follow these steps:

1. To confirm that the drives can be updated, issue this command:

   ```
   mmchfirmware --type drive -N Node[,Node...] --dry-run
   ```

2. To load the firmware, issue this command:

   ```
   mmchfirmware --type drive -N Node[,Node...]
   ```

Example scenario: Configuring GPFS Native RAID recovery groups on the GSS

This topic provides a detailed example of configuring GPFS Native RAID on a GSS building block. The example considers one GSS-24 building block, consisting of two recovery group servers cabled to four DCS3700 JBOD disk enclosures, and shows how recovery groups are defined on the disk enclosures. Throughout this topic, it may be helpful to have IBM System x GPFS Storage Server (GSS) documentation at hand.

Preparing GSS recovery group servers:

Disk enclosure and HBA cabling

The GSS-24 DCS3700 JBOD disk enclosures should be cabled to the intended recovery group servers according to the IBM System x GPFS Storage Server (GSS) hardware installation instructions. The GSS-24 building block contains four disk enclosures. Each disk enclosure contains five disk drawers. In each disk enclosure, drawers 2, 3, and 4 contain 12 HDDs. In disk enclosures 1 through 3, drawers 1 and 5 have one SSD and 11 HDDs. In disk enclosure 4 (and, in the case of GSS-26, 5 and 6), drawers 1 and 5 have 11 HDDs and 1 empty slot. In total, a GSS-24 building block has 4 enclosures, 20 drawers, 232 HDDs, 6 SSDs, and 2 empty slots. (For comparison, a GSS-26 building block has 6 enclosures, 30 drawers, 348 HDDs, 6 SSDs, and 6 empty slots.) Each disk enclosure provides redundant A and B ESM port connections for host server HBA connections. To ensure proper multi-pathing and redundancy, each recovery group server must be connected to each of the A and B ESMs using different HBAs. The IBM System x GPFS Storage Server (GSS) hardware installation documentation describes precisely which HBA ports must be connected to which disk enclosure ESM ports.

GPFS Native RAID provides system administration tools for verifying the correct connectivity of GSS-24 and GSS-26 disk enclosures, which will be seen later during the operating system preparation.

When the ESM ports of the GSS-24 disk enclosures have been cabled to the appropriate HBAs of the two recovery group servers, the disk enclosures should be powered on and the recovery group servers should be rebooted.

Verifying that the GSS-24 building block is configured correctly

Preparation then continues at the operating system level on the recovery group servers, which must be installed with the Red Hat Enterprise Linux 6.3 distribution provided with the GSS. Once the cabling has been performed and the servers have been rebooted, it is necessary to perform a thorough discovery of the disk topology on each server.

To proceed, GPFS must be installed on the recovery group servers, but they need not yet be added to a GPFS cluster. Consult *GPFS: Administration and Programming Reference* for instructions about installing GPFS.

GPFS Native RAID provides tools in /usr/lpp/mmfs/samples/vdisk for collecting and collating information on any attached GSS-24 disk enclosures and for verifying that the detected topology is correct. The **mmgetpdisktopology** command examines the list of connected devices for the operating system and produces a colon-delimited database with a line for each discovered GSS-24 physical disk, disk enclosure ESM expander device, and HBA. **mmgetpdisktopology** should be run on each of the two intended recovery group server nodes, and the results examined to verify that the disk enclosure hardware and software configuration is as expected. An additional tool called **topsummary** concisely summarizes the output of the **mmgetpdisktopology** command.

Create a directory in which to work, and then capture the output of the **mmgetpdisktopology** command from each of the two intended recovery group server nodes:

```
# mkdir gss-24
# cd gss-24
# ssh server1 /usr/lpp/mmfs/samples/vdisk/mmgetpdisktopology > server1.top
# ssh server2 /usr/lpp/mmfs/samples/vdisk/mmgetpdisktopology > server2.top
```

Then view the summary for each of the nodes; the following example is for server1:

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top
DCS3700 enclosures found: SV21311179 SV21313978 SV21314035 SV21419522
Enclosure SV21314035 (number 1):
Enclosure SV21314035 ESM A sg243[0363][scsi8 port 4] ESM B sg62[0363][scsi7 port 4]
Enclosure SV21314035 Drawer 1 ESM sg243 12 disks diskset "11082" ESM sg62 12 disks diskset "11082"
Enclosure SV21314035 Drawer 2 ESM sg243 12 disks diskset "16380" ESM sg62 12 disks diskset "16380"
Enclosure SV21314035 Drawer 3 ESM sg243 12 disks diskset "11864" ESM sg62 12 disks diskset "11864"
Enclosure SV21314035 Drawer 4 ESM sg243 12 disks diskset "31717" ESM sg62 12 disks diskset "31717"
Enclosure SV21314035 Drawer 5 ESM sg243 12 disks diskset "11824" ESM sg62 12 disks diskset "11824"
Enclosure SV21314035 sees 60 disks
Enclosure SV21419522 (number 2):
Enclosure SV21419522 ESM A sg63[0363][scsi7 port 3] ESM B sg426[0363][scsi9 port 4]
Enclosure SV21419522 Drawer 1 ESM sg63 12 disks diskset "60508" ESM sg426 12 disks diskset "60508"
Enclosure SV21419522 Drawer 2 ESM sg63 12 disks diskset "29045" ESM sg426 12 disks diskset "29045"
Enclosure SV21419522 Drawer 3 ESM sg63 12 disks diskset "08276" ESM sg426 12 disks diskset "08276"
Enclosure SV21419522 Drawer 4 ESM sg63 12 disks diskset "28750" ESM sg426 12 disks diskset "28750"
Enclosure SV21419522 Drawer 5 ESM sg63 12 disks diskset "34265" ESM sg426 12 disks diskset "34265"
Enclosure SV21419522 sees 60 disks
Enclosure SV21313978 (number 3):
Enclosure SV21313978 ESM A sg365[0363][scsi9 port 3] ESM B sg244[0363][scsi8 port 3]
Enclosure SV21313978 Drawer 1 ESM sg365 12 disks diskset "36679" ESM sg244 12 disks diskset "36679"
Enclosure SV21313978 Drawer 2 ESM sg365 12 disks diskset "18808" ESM sg244 12 disks diskset "18808"
Enclosure SV21313978 Drawer 3 ESM sg365 12 disks diskset "55525" ESM sg244 12 disks diskset "55525"
Enclosure SV21313978 Drawer 4 ESM sg365 12 disks diskset "51485" ESM sg244 12 disks diskset "51485"
Enclosure SV21313978 Drawer 5 ESM sg365 12 disks diskset "24274" ESM sg244 12 disks diskset "24274"
Enclosure SV21313978 sees 60 disks
Enclosure SV21311179 (number 4):
Enclosure SV21311179 ESM A sg184[0363][scsi8 port 2] ESM B sg3[0363][scsi7 port 2]
Enclosure SV21311179 Drawer 1 ESM sg184 11 disks diskset "59007" ESM sg3 11 disks diskset "59007"
Enclosure SV21311179 Drawer 2 ESM sg184 12 disks diskset "01848" ESM sg3 12 disks diskset "01848"
Enclosure SV21311179 Drawer 3 ESM sg184 12 disks diskset "49359" ESM sg3 12 disks diskset "49359"
Enclosure SV21311179 Drawer 4 ESM sg184 12 disks diskset "62742" ESM sg3 12 disks diskset "62742"
Enclosure SV21311179 Drawer 5 ESM sg184 11 disks diskset "62485" ESM sg3 11 disks diskset "62485"
Enclosure SV21311179 sees 58 disks
DCS3700 configuration: 4 enclosures, 6 SSDs, 2 empty slots, 238 disks total
scsi7[14.00.01.00] 0000:11:00.0 [P2 SV21311179 ESM B (sg3)] [P3 SV21419522 ESM A (sg63)] [P4 SV21314035 ESM B (sg62)]
scsi8[14.00.01.00] 0000:8b:00.0 [P2 SV21311179 ESM A (sg184)] [P3 SV21313978 ESM B (sg244)] [P4 SV21314035 ESM A (sg243)]
scsi9[14.00.01.00] 0000:90:00.0 [P3 SV21313978 ESM A (sg365)] [P4 SV21419522 ESM B (sg426)]
```

This output shows a correctly cabled and populated GSS-24 installation. Four disk enclosures with serial numbers SV21311179, SV21313978, SV21314035, and SV21419522 are found cabled in the correct order and with the correct numbers of disks.

The section for each enclosure begins with the enclosure number as determined by the cabling, followed by which ESM and HBA ports provide connectivity to the enclosure; the following line indicates that /dev/sg184 is the SCSI/SES expander device representing the enclosure's A ESM, which is at firmware level 0363 and is connected to port 2 of the HBA that the operating system configured as SCSI host 8:

```
Enclosure SV21311179 ESM A sg184[0363][scsi8 port 2] ESM B sg3[0363][scsi7 port 2]
```

Each drawer of an enclosure is represented by a line indicating how many disks are seen over each ESM and a "diskset" checksum of the disk WWNs, which is used to verify that each ESM connection sees the same set of disks over different device paths.

If the cabling were incorrect in any way, one or more of the enclosures would indicate "number undetermined," and the following message would be printed:

```
Unable to determine enclosure order; check the HBA to enclosure cabling.
```

If any slots are unexpectedly empty or contain the wrong type of disk, the **topsummary** output will include additional messages such as these:

```
Location SV21311179-2-10 appears only on the sg184 path
Location SV21313978-1-3 appears empty but should have an SSD
Location SV21311179-5-12 has an HDD but should be empty
```

The HBAs are also listed (firmware levels in brackets) with their bus addresses and port connections. For example, [P2 SV21311179 ESM B (sg3)] indicates that HBA port 2 is connected to ESM B of enclosure SV21311179; ESM B is represented by the /dev/sg3 SCSI/SES expander device. The HBA bus addresses and port connections are standard in a GSS building block and are used by **topsummary** to validate the correctness of the cabling.

If the cabling is called out as incorrect and the enclosure order cannot be determined, or if other discrepancies are noted (for example, physical disks that are expected but do not show up at all, or SSDs or HDDs in the wrong locations), corrections should be made and verified before proceeding. This would probably involve shutting down the servers and the disk enclosures and fixing the cabling and disk placement.

The server2.top topology database should also be examined with the **topsummary** sample script and verified to be correct. It should also be verified to have found the same enclosures with the same serial numbers and in the same order. One way to do this is to run the following commands and verify that they give exactly the same output:

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top | grep number
# /usr/lpp/mmfs/samples/vdisk/topsummary server2.top | grep number
```

Once the GSS-24 disk enclosure topologies are verified to be correct on both intended recovery group server nodes, it is highly recommended that the disk enclosures be entered into the GPFS Native RAID component database. This allows the system administrator to provide a meaningful name for each component and to record each of their physical rack locations in a manner intended to make maintenance actions easier.

To proceed (either to populate the GPFS Native RAID component database or to create recovery groups), the recovery group servers must now be members of the same GPFS cluster. Consult *GPFS: Administration and Programming Reference* for instructions for creating a GPFS cluster.

Once the intended recovery group servers are added to a GPFS cluster, the component database can be updated to identify the equipment rack and the U compartments in which the disk enclosures reside.

In this scenario, a GSS-24 building block will occupy an Intelligent Cluster 42U Primary Rack. Using the component database, the four disk enclosures and the rack in which they reside may be given meaningful names and associated with each other.

To create the component database entry for an Intelligent Cluster 42U Primary Rack, which has part number 1410HPA, and to give it the descriptive name BB1, use the **mmaddcomp** command:

```
# mmaddcomp 1410HPA --name BB1
```

Do the same for each of the four disk enclosures as identified by the **topsummary** command. In this example, the enclosures were identified according to the cabling in this order:

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top | grep number
Enclosure SV21314035 (number 1):
Enclosure SV21419522 (number 2):
Enclosure SV21313978 (number 3):
Enclosure SV21311179 (number 4):
```

Suppose this is GSS-24 building block 1, and the desired descriptive names for the disk enclosures are BB1ENC1, BB1ENC2, BB1ENC3, and BB1ENC4. To define these to the component database, use the disk enclosure part number 181880E as follows:

```
# mmaddcomp 181880E --serial-number SV21314035 --name BB1ENC1
# mmaddcomp 181880E --serial-number SV21419522 --name BB1ENC2
# mmaddcomp 181880E --serial-number SV21313978 --name BB1ENC3
# mmaddcomp 181880E --serial-number SV21311179 --name BB1ENC4
```

You may also use the **mmdiscovercomp** command in place of **mmaddcomp**, then use **mmchcomp** to assign descriptive names to the enclosures.

At this point, the building block rack and disk enclosures can be listed from component database using the **mmlscomp** command:

```
# mmlscomp
Rack Components
Comp ID Part Number Serial Number Name
------- ----------- ------------- ------
      1 1410HPA                   BB1

Storage Enclosure Components

Comp ID Part Number Serial Number Name    Display ID
------- ----------- ------------- ------- ----------
      2 181880E     SV21314035    BB1ENC1
      3 181880E     SV21419522    BB1ENC2
      4 181880E     SV21313978    BB1ENC3
      5 181880E     SV21311179    BB1ENC4
```

Each of the defined components has an ID. The location of the enclosures can be defined according to the four U compartments they occupy in the rack. To do this, use the **mmchcomploc** command to specify the component IDs of the enclosures, the rack that contains them, and the starting U compartment (position) within the rack. The syntax of the **mmchcomploc** command is:

mmchcomploc *Component*
Container Position

To define enclosure BB1ENC1 as occupying U compartments 1 through 4 in rack BB1, use this command:

```
# mmchcomploc BB1ENC1 BB1 1
```

Suppose BB1ENC2, BB1ENC3, and BB1ENC4 have starting U compartments 5, 13, and 17, respectively, in rack BB1. Use the following commands to define these relationships:

```
# mmchcomploc BB1ENC2 BB1 5
# mmchcomploc BB1ENC3 BB1 13
# mmchcomploc BB1ENC4 BB1 17
```

The defined component locations can be listed with the **mmlscomploc** command:

```
# mmlscomploc
Component Location
--------- ------------------
BB1ENC1   Rack BB1 U01-04
BB1ENC2   Rack BB1 U05-08
BB1ENC3   Rack BB1 U13-16
BB1ENC4   Rack BB1 U17-20
```

(U compartments 9 - 10 and 11 - 12 are presumably occupied by the recovery group servers. They are omitted here as extraneous to this set of examples. They may be named and defined if desired by using component part number 7915AC1 with the **mmaddcomp** command, and then using their component IDs and starting U compartments with the **mmchcomploc** command, as was done with the disk enclosures.)

With the above component names and locations having been entered into the GPFS Native RAID component database, a common maintenance action such as replacing a disk will be able to supply meaningful direction to the user in locating the equipment in a crowded machine room.

Creating recovery groups on the IBM System x GPFS Storage Server (GSS):

Configuring GPFS nodes to be recovery group servers

Having verified the disk enclosure connectivity of the GSS-24 building block, and having optionally also created a component database to associate names and machine room locations with the storage hardware, GPFS Native RAID must be enabled on the intended recovery group servers.

To proceed, the recovery group servers must be members of the same GPFS cluster. If this has not already been done in preparing a GPFS Native RAID component database, consult *GPFS: Administration and Programming Reference* for instructions for adding nodes to or creating a GPFS cluster.

Before a GPFS node can create and serve recovery groups, it must be configured with a vdisk track cache. This is accomplished by setting the **nsdRAIDTracks** configuration parameter.

nsdRAIDTracks is the GPFS configuration parameter essential to define a GPFS cluster node as a recovery group server. It specifies the number of vdisk tracks of which the attributes will be held in memory by the GPFS daemon on the recovery group server.

The actual contents of the vdisk tracks, the user data and the checksums, are stored during runtime in the standard GPFS pagepool. Therefore, the size of the GPFS pagepool configured on a recovery group server should be considerable, on the order of tens of gigabytes. The amount of pagepool dedicated to hold vdisk track data is governed by the **nsdRAIDBufferPoolSizePct** parameter, which defaults to 50%. In practice, a recovery group server will not need to use the GPFS pagepool for any significant amount of standard file caching, and the **nsdRAIDBufferPoolSizePct** value can be increased to 80%. This implies that the **prefetchPct** pagepool parameter can be reduced to as little as 5%. Since a recovery group server is also an NSD server, the vdisk buffer pool doubles as the NSD buffer pool. To achieve good NSD server performance, a large number of dedicated NSD service threads should be configured by setting **nsdMinWorkerThreads** and **nsdMaxWorkerThreads** to the same value, on the order of a few thousand.

In this example, the recovery group servers will be configured to cache the information on 131072 vdisk tracks and to have 38 GiB of pagepool, of which 80% will be used for vdisk data and 5% for application file caching. The servers are also configured to have 3072 NSD service threads. Once the configuration changes are made, the servers will need to be restarted.

```
# mmchconfig nsdRAIDTracks=131072,nsdRAIDBufferPoolSizePct=80,prefetchPct=5,pagepool=38G -N server1,server2
# mmchconfig nsdMinWorkerThreads=3072,nsdMaxWorkerThreads=3072 -N server1,server2
# mmshutdown -N server1,server2
# mmstartup -N server1,server2
```

Defining the recovery group layout

The definition of recovery groups on a GSS-24 building block is accomplished by dividing the drawers of the enclosures into left and right halves. The sharing of GSS-24 disk enclosures by two servers implies two recovery groups; one is served by one node and one by the other, and each server acts as the other's backup. Half the disks in each enclosure and drawer should belong to one recovery group, and half to the other. One recovery group will therefore be defined on the disks in the left half of each drawer, slots 1 though 6, and one on the disks in the right half of each drawer, slots 7 through 12. The three SSDs in drawer 1 slot 3 of the first three enclosures will make up the vdisk log declustered array for the left recovery group, and the three SSDs in drawer 5 slot 12 of the first three enclosures will make up the vdisk log declustered array of the right recovery group. The remaining 116 HDDs are divided into two vdisk data declustered arrays of 58 disks.

GPFS Native RAID provides a tool that understands the layout of GSS building blocks and will automatically generate the **mmcrrecoverygroup** stanza files for creating the left and right recovery groups. `/usr/lpp/mmfs/samples/vdisk/mkrginput`, when supplied with output of the **mmgetpdisktopology** command, will create recovery group stanza files for the left and right sets of disks as described in the previous paragraph.

Each recovery group server, though it may see the same functional disk enclosure topology, will almost certainly differ in the particulars of which disk device names (for example, `/dev/sdax`) refer to which physical disks in what disk enclosure location.

There are therefore two possibilities for creating the recovery group stanza files and the recovery groups themselves:

Alternative 1:
Generate the recovery group stanza files and create the recovery groups from the perspective of just one of the servers as if that server were to be primary for both recovery groups, and then use the **mmchrecoverygroup** command to swap the primary and backup servers for one of the recovery groups

Alternative 2:
Generate the recovery group stanza files for each server's primary recovery group using the primary server's topology file.

The following example shows both alternatives.

Creating the recovery groups, alternative 1

To create the recovery group input stanza files from the perspective of `server1`, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkrginput server1.top
```

This will create two files, one for the left set of disks and one for the right set of disks found in the `server1` topology. The files will be named after the serial number of the enclosure determined to be first in the topology, but each will contain disks from all four enclosures. In this case, the resulting stanza files will be `SV21314035L.server1` for the left half and `SV21314035R.server1` for the right half.

The recovery group stanza files will follow the recommended best practice for a GSS-24 building block of defining in each half a separate declustered array of three SSDs for recovery group transaction logging, and two file system data declustered arrays using the regular HDDs. One of the data declustered arrays will contain the disks from the left or right halves of enclosures 1 and 2, and the other will contain the disks from the left or right halves of enclosures 3 and 4. The vdisk log declustered array will be named `LOG`, and data declustered arrays will be named `DA1` and `DA2`.

Except for the `LOG` declustered array, the defaults are accepted for recovery group declustered array parameters such as scrub duration, spare space, and disk replacement policy. For the `LOG` declustered array, the spare space will be defined to be 0. This will permit the log vdisk to later be created with three-way replication, which is a two-fault tolerant RAID code.

The stanza file will look something like this:

```
# head -14 SV21314035L.server1
%da: daName=LOG
     spares=0
%pdisk: pdiskName=e1d1s03log
        device=/dev/sdbk
        da=LOG
%pdisk: pdiskName=e2d1s03log
        device=/dev/sdbw
        da=LOG
%pdisk: pdiskName=e3d1s03log
        device=/dev/sdis
        da=LOG
%pdisk: pdiskName=e1d1s01
        device=/dev/sdbi
        da=DA1
```

All the pdisk stanzas for declustered array LOG will be listed first, followed by those for DA1 and DA2. The pdisk names will indicate the enclosure number, the drawer number, and the slot within the drawer in which the physical disk resides. The names of the pdisks in the LOG declustered array will have "log" appended to make them stand out. Notice that only one block device path to the disk is given; the second path will be discovered automatically by GPFS Native RAID soon after the recovery group is created.

If, as was recommended, the GPFS Native RAID component database was used to provide meaningful names for the GSS-24 building block components, the names of the recovery groups should be chosen to follow that convention. In this example, the building block rack was named BB1 and the enclosures were named BB1ENC1 through BB1ENC4. It would make sense then to name the left and right recovery groups BB1RGL and BB1RGR. Other conventions are of course possible.

With the left and right recovery group stanza files both created from the disk device perspective of server node server1, these two recovery groups can be created using two separate invocations of the **mmcrrecoverygroup** command:

```
# mmcrrecoverygroup BB1RGL -F SV21314035L.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
# mmcrrecoverygroup BB1RGR -F SV21314035R.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

Note that both recovery groups were created with server1 as primary and server2 as backup. It is now necessary to swap the primary and backup servers for BB1RGR using the **mmchrecoverygroup** command:

```
# mmchrecoverygroup BB1RGR --servers server2,server1
mmchrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

GPFS Native RAID will automatically discover the appropriate disk devices on server2.

Creating the recovery groups, alternative 2

To create the recovery groups from the start with the intended primary and backup servers, the stanza files from both server topologies will need to be created.

To create the server1 recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkrginput server1.top
```

To create the server2 recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkrginput server2.top
```

These two commands will result in four stanza files: SV21314035L.server1, SV21314035R.server1, SV21314035L.server2, and SV21314035R.server2.

The BB1RGL recovery group must then be created using server1 as the primary and the SV21314035L.server1 stanza file. The BB1RGR recovery group must be created using server2 as the primary and the SV21314035R.server2 stanza file.

```
# mmcrrecoverygroup BB1RGL -F SV21314035L.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
# mmcrrecoverygroup BB1RGR -F SV21314035R.server2 --servers server2,server1
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

Since each recovery group was created using the intended primary server and the stanza file for that server, it is not necessary to swap the primary and backup servers.

Verifying recovery group creation

Use the **mmlsrecoverygroup** command to verify that each recovery group was created:

```
# mmlsrecoverygroup BB1RGL -L

                  declustered
recovery group    arrays      vdisks pdisks
----------------- ----------- ------ ------
BB1RGL                      3      0    119
declustered needs                         replace              scrub      background activity
array       service vdisks pdisks spares threshold free space duration task   progress  priority
----------- ------- ------ ------ ------ --------- ---------- -------- ------------------------
LOG         no           0      3      0         1    558 GiB  14 days inactive     0%  low
DA1         no           0     58      2         2    101 TiB  14 days inactive     0%  low
DA2         no           0     58      2         2    101 TiB  14 days inactive     0%  low

                                      declustered                       checksum
vdisk              RAID code             array    vdisk size block size granularity remarks
------------------ ------------------ ----------- ---------- ---------- ----------- -------

active recovery group server                     servers
----------------------------------------------- -------
server1                                         server1,server2

# mmlsrecoverygroup BB1RGR -L
                  declustered
recovery group      arrays    vdisks pdisks
----------------- ----------- ------ ------
BB1RGR                      3      0    119
declustered needs                         replace              scrub    background activity
array       service vdisks pdisks spares threshold free space duration task   progress  priority
----------- ------- ------ ------ ------ --------- ---------- -------- ------------------------
LOG         no           0      3      0         1    558 GiB  14 days inactive     0%  low
DA1         no           0     58      2         2    101 TiB  14 days inactive     0%  low
DA2         no           0     58      2         2    101 TiB  14 days inactive     0%  low

                                      declustered                        checksum
vdisk             RAID code              array    vdisk size block size granularity remarks
------------------ ------------------ ----------- ---------- ---------- ----------- -------

active recovery group server                     servers
----------------------------------------------- -------
server1                                        server2,server1
```

Notice that the vdisk sections of the newly created recovery groups are empty; the next step is to create the vdisks.

Defining and creating the vdisks

Once the recovery groups are created and being served by their respective servers, it is time to create the vdisks using the **mmcrvdisk** command.

Each recovery group requires a single log vdisk for recording RAID updates and diagnostic information. This is internal to the recovery group, cannot be used for user data, and should be the only vdisk in the LOG declustered array. The log vdisks use three-way replication in order to be fully two-fault tolerant in the LOG declustered array, which contains three SSDs and no spare space.

Data vdisks are required to be defined in the two data declustered arrays for use as file system NSDs. In this example, each of the declustered arrays for file system data is divided into two vdisks with different characteristics:

- one using 4-way replication and a 1 MiB block size and a total vdisk size of 1024 GiB suitable for file system metadata
- one using Reed-Solomon 8 + 3p encoding and an 16 MiB block size suitable for file system data

The vdisk size is omitted for the Reed-Solomon vdisks, meaning that they will default to use the remaining non-spare space in the declustered array (for this to work, any vdisks with specified total sizes must of course be defined first).

The possibilities for the vdisk creation stanza file are quite great, depending on the number and type of vdisk NSDs required for the number and type of file systems desired, so the vdisk stanza file will need to be created by hand, possibly following a template. The example vdisk stanza file that is supplied in /usr/lpp/mmfs/samples/vdisk/vdisk.stanza may be used for this purpose and adapted to specific file system requirements.

For GSS, it is suggested that each recovery group use the option of defining a log tip vdisk together with the mandatory log home vdisk. These vdisks record RAID updates and diagnostic information, are internal to each recovery group, and cannot be used for user data. The log tip vdisk is a fast staging area for RAID updates. When a log tip vdisk is used, it must be the only vdisk in the LOG declustered array, and must be the first vdisk defined in each recovery group. When log tip is used, the log home vdisk must be the second vdisk defined in each recovery group, and should be placed in the first data declustered array (in this example, DA1). Any additional data declustered arrays may not contain log vdisks. As a staging area, the log tip vdisk is relatively small in size at 128 MiB.

In this example, a single stanza file, mmcrvdisk.BB1, is used. The single file contains the specifications for all the vdisks in both the BB1RGL and BB1RGR recovery groups. Here is what the example stanza file for use with **mmcrvdisk** should look like:

```
# cat mmcrvdisk.BB1
%vdisk: vdiskName=BB1RGLLOGTIP
        rg=BB1RGL
        da=LOG
        blocksize=1m
        size=128m
        raidCode=3WayReplication
        diskUsage=vdiskLogTip
%vdisk: vdiskName=BB1RGRLOGTIP
        rg=BB1RGR
        da=LOG
        blocksize=1m
        size=128m
        raidCode=3WayReplication
        diskUsage=vdiskLogTip
%vdisk: vdiskName=BB1RGLLOGHOME
        rg=BB1RGL
        da=DA1
        blocksize=1m
        size=40g
        raidCode=4WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=BB1RGRLOGHOME
        rg=BB1RGR
        da=DA1
        blocksize=1m
        size=40g
        raidCode=4WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=BB1RGLMETA1
        rg=BB1RGL
        da=DA1
        blocksize=1m
        size=256g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=1
        pool=system
%vdisk: vdiskName=BB1RGRMETA1
        rg=BB1RGR
        da=DA1
        blocksize=1m
        size=256g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=1
        pool=system
%vdisk: vdiskName=BB1RGLDATA1
        rg=BB1RGL
        da=DA1
        blocksize=16m
        size=8192g
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=1
        pool=data
%vdisk: vdiskName=BB1RGRDATA1
        rg=BB1RGR
        da=DA1
        blocksize=16m
        size=8192g
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=1
        pool=data
[DA2 vdisks omitted.]
```

Notice how the file system metadata vdisks are flagged for eventual file system usage as **metadataOnly** and for placement in the system storage pool, and the file system data vdisks are flagged for eventual **dataOnly** usage in the data storage pool. (After the file system is created, a policy will be required to allocate file system data to the correct storage pools; see “Creating the GPFS file system” on page 61.)

Importantly, also notice that block sizes for the file system metadata and file system data vdisks must be specified at this time, may not later be changed, and must match the block sizes supplied to the eventual **mmcrfs** command.

Notice also that the eventual **failureGroup=1** value for the NSDs on the file system vdisks is the same for vdisks in both the `BB1RGL` and `BB1RGR` recovery groups. This is because the recovery groups, although they have different servers, still share a common point of failure in the four GSS-24 disk enclosures, and GPFS should be informed of this through a distinct failure group designation for each disk enclosure. It is up to the GPFS system administrator to decide upon the failure group numbers for each GSS building block in the GPFS cluster. In this example, the failure group number 1 has been chosen to match the example building block number.

To create the vdisks specified in the mmcrvdisk.BB1 file, use the following **mmcrvdisk** command:

```
# mmcrvdisk -F mmcrvdisk.BB1
mmcrvdisk: [I] Processing vdisk BB1RGLLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGRLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGLLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGRLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGLMETA1
mmcrvdisk: [I] Processing vdisk BB1RGRMETA1
mmcrvdisk: [I] Processing vdisk BB1RGLDATA1
mmcrvdisk: [I] Processing vdisk BB1RGRDATA1
mmcrvdisk: [I] Processing vdisk BB1RGLMETA2
mmcrvdisk: [I] Processing vdisk BB1RGRMETA2
mmcrvdisk: [I] Processing vdisk BB1RGLDATA2
mmcrvdisk: [I] Processing vdisk BB1RGRDATA2
mmcrvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Creation of the vdisks may be verified through the **mmlsvdisk** command (the **mmlsrecoverygroup** command may also be used):

```
# mmlsvdisk
                                                          declustered  block size
 vdisk name          RAID code        recovery group         array      in KiB    remarks
 ------------------  ---------------  ------------------  -----------  ----------  -------
 BB1RGLDATA1         8+3p             BB1RGL              DA1               16384
 BB1RGLDATA2         8+3p             BB1RGL              DA2               16384
 BB1RGLLOGHOME       4WayReplication  BB1RGL              DA1                1024  log
 BB1RGLLOGTIP        3WayReplication  BB1RGL              LOG                1024  logTip
 BB1RGLMETA1         4WayReplication  BB1RGL              DA1                1024
 BB1RGLMETA2         4WayReplication  BB1RGL              DA2                1024
 BB1RGRDATA1         8+3p             BB1RGR              DA1               16384
 BB1RGRDATA2         8+3p             BB1RGR              DA2               16384
 BB1RGRLOGHOME       4WayReplication  BB1RGR              DA1                1024  log
 BB1RGRLOGTIP        3WayReplication  BB1RGR              LOG                1024  logTip
 BB1RGRMETA1         4WayReplication  BB1RGR              DA1                1024
 BB1RGRMETA2         4WayReplication  BB1RGR              DA2                1024
```

Creating NSDs from vdisks

The **mmcrvdisk** command rewrites the input file so that it is ready to be passed to the **mmcrnsd** command that creates the NSDs from which GPFS builds file systems. To create the vdisk NSDs, run the **mmcrnsd** command on the rewritten **mmcrvdisk** stanza file:

```
# mmcrnsd -F mmcrvdisk.BB1
mmcrnsd: Processing disk BB1RGLMETA1
mmcrnsd: Processing disk BB1RGRMETA1
mmcrnsd: Processing disk BB1RGLDATA1
mmcrnsd: Processing disk BB1RGRDATA1
mmcrnsd: Processing disk BB1RGLMETA2
mmcrnsd: Processing disk BB1RGRMETA2
mmcrnsd: Processing disk BB1RGLDATA2
mmcrnsd: Processing disk BB1RGRDATA2
mmcrnsd: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

Notice how the recovery group log vdisks are omitted from NSD processing.

The **mmcrnsd** command then once again rewrites the stanza file in preparation for use as input to the **mmcrfs** command.

Creating the GPFS file system

Run the **mmcrfs** command to create the file system:

```
# mmcrfs gpfsbb1 -F mmcrvdisk.BB1 -B 16m --metadata-block-size 1m -T /gpfsbb1 -n 256
The following disks of gpfsbb1 will be formatted on node server1:
    BB1RGLMETA1: size 269213696 KB
    BB1RGRMETA1: size 269213696 KB
    BB1RGLDATA1: size 8593965056 KB
    BB1RGRDATA1: size 8593965056 KB
    BB1RGLMETA2: size 269213696 KB
    BB1RGRMETA2: size 269213696 KB
    BB1RGLDATA2: size 8593965056 KB
    BB1RGRDATA2: size 8593965056 KB
Formatting file system ...
Disks up to size 3.3 TB can be added to storage pool system.
Disks up to size 82 TB can be added to storage pool data.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
98 % complete on Sun Nov 18 13:27:00 2012
100 % complete on Sun Nov 18 13:27:00 2012
Formatting Allocation Map for storage pool data
85 % complete on Sun Nov 18 13:27:06 2012
100 % complete on Sun Nov 18 13:27:06 2012
Completed creation of file system /dev/gpfsbb1.
mmcrfs: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

(The **-n 256** parameter specifies that the allocation maps should account for 256 nodes mounting the file system. This is an example only and should be adjusted to actual cluster expectations.)

Notice how the 16 MiB data block size is specified with the traditional **-B** parameter and the 1 MiB metadata block size is specified with the **--metadata-block-size** parameter. Since a file system with different metadata and data block sizes requires the use of multiple GPFS storage pools, a file system placement policy is needed to direct user file data to the data storage pool. In this example, the file placement policy is very simple:

```
# cat policy
rule 'default' set pool 'data'
```

The policy must then be installed in the file system using the **mmchpolicy** command:

```
# mmchpolicy gpfsbb1 policy -I yes
Validated policy `policy': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules
Policy `policy' installed and broadcast to all nodes.
```

If a policy is not placed in a file system with multiple storage pools, attempts to place data into files will return ENOSPC as if the file system were full.

This file system, built on a GSS-24 building block using two recovery groups, two recovery group servers, four file system metadata vdisk NSDs and four file system data vdisk NSDs, may now be mounted and placed into service:

```
# mmmount gpfsbb1 -a
```

Differences in GSS 1.5 with LogTip NVRAM partitions:

Disk enclosure and HBA cabling

GSS 1.5 servers are configured with six small (2000 MiB) disk partitions on the mirrored boot disk. These disk partitions are write-back storage for very fast NVRAM cache in the server HBAs. On each server in a GSS building block, two of these NVRAM write-back partitions are used for the log tip declustered array for GNR RAID updates. (Event logs are only written to the log home vdisk on spinning disks, not the log tip vdisk.)

The **mmgetpdisktopology** command acquires information about NVRAM partitions on the server on which it is run. The **topsummary** command reports the existence of NVRAM partitions in the **mmgetpdisktopology** output.

For example, in the following output, the line "GSS configuration" indicates that there are six NVRAM partitions in this server topology. This is the expected output for a GSS 1.5 server in a GSS building block.

```
# ssh server1 /usr/lpp/mmfs/bin/mmgetpdisktopology > server1.top
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top
GSS enclosures found: SV21311179 SV21313978 SV21314035 SV21419522
Enclosure SV21314035 (number 1):
Enclosure SV21314035 ESM A sg248[0393][scsi8 port 4] ESM B sg67[0393][scsi7 port 4]
Enclosure SV21314035 Drawer 1 ESM sg248 12 disks diskset "13336" ESM sg67 12 disks diskset "13336"
Enclosure SV21314035 Drawer 2 ESM sg248 12 disks diskset "30460" ESM sg67 12 disks diskset "30460"
Enclosure SV21314035 Drawer 3 ESM sg248 12 disks diskset "55034" ESM sg67 12 disks diskset "55034"
Enclosure SV21314035 Drawer 4 ESM sg248 12 disks diskset "26111" ESM sg67 12 disks diskset "26111"
Enclosure SV21314035 Drawer 5 ESM sg248 12 disks diskset "34371" ESM sg67 12 disks diskset "34371"
Enclosure SV21314035 sees 60 disks
Enclosure SV21419522 (number 2):
Enclosure SV21419522 ESM A sg68[0393][scsi7 port 3] ESM B sg431[0393][scsi9 port 4]
Enclosure SV21419522 Drawer 1 ESM sg68 12 disks diskset "05350" ESM sg431 12 disks diskset "05350"
Enclosure SV21419522 Drawer 2 ESM sg68 12 disks diskset "29524" ESM sg431 12 disks diskset "29524"
Enclosure SV21419522 Drawer 3 ESM sg68 12 disks diskset "12828" ESM sg431 12 disks diskset "12828"
Enclosure SV21419522 Drawer 4 ESM sg68 12 disks diskset "52289" ESM sg431 12 disks diskset "52289"
Enclosure SV21419522 Drawer 5 ESM sg68 12 disks diskset "57634" ESM sg431 12 disks diskset "57634"
Enclosure SV21419522 sees 60 disks
Enclosure SV21313978 (number 3):
Enclosure SV21313978 ESM A sg370[0393][scsi9 port 3] ESM B sg249[0393][scsi8 port 3]
Enclosure SV21313978 Drawer 1 ESM sg370 12 disks diskset "09002" ESM sg249 12 disks diskset "09002"
Enclosure SV21313978 Drawer 2 ESM sg370 12 disks diskset "27263" ESM sg249 12 disks diskset "27263"
Enclosure SV21313978 Drawer 3 ESM sg370 12 disks diskset "14213" ESM sg249 12 disks diskset "14213"
Enclosure SV21313978 Drawer 4 ESM sg370 12 disks diskset "36478" ESM sg249 12 disks diskset "36478"
Enclosure SV21313978 Drawer 5 ESM sg370 12 disks diskset "31136" ESM sg249 12 disks diskset "31136"
Enclosure SV21313978 sees 60 disks
Enclosure SV21311179 (number 4):
Enclosure SV21311179 ESM A sg189[0393][scsi8 port 2] ESM B sg8[0393][scsi7 port 2]
Enclosure SV21311179 Drawer 1 ESM sg189 11 disks diskset "18272" ESM sg8 11 disks diskset "18272"
Enclosure SV21311179 Drawer 2 ESM sg189 12 disks diskset "61983" ESM sg8 12 disks diskset "61983"
Enclosure SV21311179 Drawer 3 ESM sg189 12 disks diskset "36150" ESM sg8 12 disks diskset "36150"
Enclosure SV21311179 Drawer 4 ESM sg189 12 disks diskset "62228" ESM sg8 12 disks diskset "62228"
Enclosure SV21311179 Drawer 5 ESM sg189 11 disks diskset "05314" ESM sg8 11 disks diskset "05314"
Enclosure SV21311179 sees 58 disks

GSS configuration: 4 enclosures, 6 SSDs, 2 empty slots, 238 disks total, 6 NVRAM partitions

scsi7[15.00.00.00] 0000:11:00.0 [P2 SV21311179 ESM B (sg8)] [P3 SV21419522 ESM A (sg68)]
 [P4 SV21314035 ESM B (sg67)]
scsi8[15.00.00.00] 0000:8b:00.0 [P2 SV21311179 ESM A (sg189)] [P3 SV21313978 ESM B (sg249)]
 [P4 SV21314035 ESM A (sg248)]
scsi9[15.00.00.00] 0000:90:00.0 [P3 SV21313978 ESM A (sg370)] [P4 SV21419522 ESM B (sg431)]
```

Defining the recovery group layout

The left and right recovery groups in a GSS building block without server NVRAM partitions contain a LOG declustered array built on SSDs and two (GSS-24) or three (GSS-26) data declustered arrays (DA1, DA2, and DA3) built on HDDs.

In a GSS 1.5 building block with server NVRAM partitions, there will be an NVR declustered array built on NVRAM partitions, an SSD declustered array built on SSDs, and two or three data declustered arrays (DA1, DA2, and DA3) built on HDDs.

This introduces a new method of storing GNR RAID updates and diagnostics that is both faster and has increased redundancy, and a new set of vdisks for GNR internal logging.

The log tip vdisk will now reside in the faster NVR declustered array.

The SSD declustered array will hold a new log backup vdisk.

The log home vdisk will reside in the first data declustered array (DA1).

A new vdisk of type "log reserved" will be placed in the second and third data declustered arrays. This will be the same size as the log home vdisk, with the intent of making all the data declustered arrays have the same amount of usable space for filesystem vdisks. (The log reserved vdisks may in the future be used to store additional internal GNR data.)

The NVR declustered array will contain two pdisks. These will be a write-back NVRAM partition from each server, one of which is accessed over the network (rather than through a common SAS fabric).

Since the NVR pdisks are local to each server, recovery groups can no longer be created from the isolated perspective of a single server. The stanza file used to create a recovery group must contain information about the local NVRAM partitions on each of the two servers in a GSS 1.5 building block. The syntax of the **mkrginput** command has been extended to accept two topology files, one from each server in the building block.

To create the recovery group creation stanza files for the left and right recovery groups in a GSS 1.5 building block, provide the names of the topology files for both servers to the **mkrginput** command:

```
# /usr/lpp/mmfs/samples/vdisk/mkrginput server1.top server2.top
```

This will create two stanza files, one for the left recovery group and one for the right. The files will be named after the serial number of the first numbered enclosure in the building block, with an added 'L' or 'R' and a .stanza suffix:

```
# ls *.stanza
SV21314035L.stanza  SV21314035R.stanza
```

The left and right recovery groups must now be created by specifying both servers, using one server as primary for the left and backup for the right, and the other as primary for the right and backup for the left:

```
# mmcrrecoverygroup BB1RGL -F SV21314035L.stanza --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
# mmcrrecoverygroup BB1RGR -F SV21314035R.stanza --servers server2,server1
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Verifying recovery group creation

The **mmlsrecoverygroup** command can now be used to show the NVR, SSD, and data declustered arrays:

```
# mmlsrecoverygroup BB1RGL -L

                    declustered
 recovery group       arrays     vdisks  pdisks  format version
 -----------------  -----------  ------  ------  --------------
 BB1RGL                       4       0     121  3.5.0.13

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 SSD           no           0       3       0          1     558 GiB   14 days  repair-RGD/VCD 10%  low
 NVR           no           0       2       0          1    3744 MiB   14 days  repair-RGD/VCD 10%  low
 DA1           no           0      58       2          2     101 TiB   14 days  repair-RGD/VCD 10%  low
 DA2           no           0      58       2          2     101 TiB   14 days  repair-RGD/VCD 10%  low

                                         declustered                           checksum
 vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
 ------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------

 active recovery group server                     servers
 -----------------------------------------------  -------
 server1                                          server1,server2

# mmlsrecoverygroup BB1RGR -L

                    declustered
 recovery group       arrays     vdisks  pdisks  format version
 -----------------  -----------  ------  ------  --------------
 BB1RGR                       4       0     121  3.5.0.13

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 SSD           no           0       3       0          1     558 GiB   14 days  repair-RGD/VCD 10%  low
 NVR           no           0       2       0          1    3744 MiB   14 days  repair-RGD/VCD 10%  low
 DA1           no           0      58       2          2     101 TiB   14 days  repair-RGD/VCD 10%  low
 DA2           no           0      58       2          2     101 TiB   14 days  repair-RGD/VCD 10%  low

                                         declustered                           checksum
 vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
 ------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------

 active recovery group server                     servers
 -----------------------------------------------  -------
 server2                                          server2,server1
```

Defining and creating the vdisks

Each of the left and right recovery groups will now require:

- A log tip vdisk (type `vdiskLogTip`) in the `NVR` declustered array
- A log tip backup vdisk (type `vdiskLogTipBackup`) in the `SSD` declustered array
- A log home vdisk (type `vdiskLog`) in the `DA1` declustered array
- A log reserved vdisk (type `vdiskLogReserved` in the `DA2` declustered array (and in the `DA3` declustered array, in the case of GSS-26)

These will all need to be specified in a vdisk creation stanza file. The vdisk creation example for GSS without NVRAM partitions combined the GNR log vdisks and the filesystem vdisks in the same stanza file. In the case of GSS 1.5 with NVRAM partitions, it will be simpler to perform vdisk creation in two steps with two separate stanza files, one for the log vdisks and one for the filesystem vdisks.

The log vdisk creation file for a GSS-24 building block with NVRAM partitions will look like this:

```
# cat mmcrvdisklog.BB1
%vdisk:
  vdiskName=BB1RGLLOGTIP
  rg=BB1RGL
  daName=NVR
  blocksize=2m
  size=48m
  raidCode=2WayReplication
  diskUsage=vdiskLogTip

%vdisk:
  vdiskName=BB1RGLLOGTIPBACKUP
  rg=BB1RGL
  daName=SSD
  blocksize=2m
  size=48m
  raidCode=3WayReplication
  diskUsage=vdiskLogTipBackup

%vdisk:
  vdiskName=BB1RGLLOGHOME
  rg=BB1RGL
  daName=DA1
  blocksize=2m
  size=20g
  raidCode=4WayReplication
  diskUsage=vdiskLog
  longTermEventLogSize=4m
  shortTermEventLogSize=4m
  fastWriteLogPct=90

%vdisk:
  vdiskName=BB1RGLDA2RESERVED
  rg=BB1RGL
  daName=DA2
  blocksize=2m
  size=20g
  raidCode=4WayReplication
  diskUsage=vdiskLogReserved

%vdisk:
  vdiskName=BB1RGRLOGTIP
  rg=BB1RGR
  daName=NVR
  blocksize=2m
  size=48m
  raidCode=2WayReplication
  diskUsage=vdiskLogTip

%vdisk:
  vdiskName=BB1RGRLOGTIPBACKUP
  rg=BB1RGR
  daName=SSD
  blocksize=2m
  size=48m
  raidCode=3WayReplication
  diskUsage=vdiskLogTipBackup

%vdisk:
  vdiskName=BB1RGRLOGHOME
  rg=BB1RGR
  daName=DA1
  blocksize=2m
  size=20g
  raidCode=4WayReplication
  diskUsage=vdiskLog
  longTermEventLogSize=4m
  shortTermEventLogSize=4m
  fastWriteLogPct=90

%vdisk:
  vdiskName=BB1RGRDA2RESERVED
  rg=BB1RGR
  daName=DA2
  blocksize=2m
  size=20g
  raidCode=4WayReplication
  diskUsage=vdiskLogReserved
```

The parameters chosen for `size`, `blocksize`, `raidCode`, `fastWriteLogPct`, and the event log sizes are standard and have been carefully calculated and should not be changed. The only difference in the vdisk log stanza files between two building blocks will be in the recovery group and vdisk names. (In the case of a GSS-26 building block with NVRAM partitions, there will be an additional `vdiskLogReserved` for `DA3`, with parameters otherwise identical to the `DA2` log reserved vdisk.)

The log vdisks for the example GSS 1.5 building block `BB1` can now be created with the **mmcrvdisk** command:

```
# mmcrvdisk -F mmcrvdisklog.BB1
mmcrvdisk: [I] Processing vdisk BB1RGLLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGLLOGTIPBACKUP
mmcrvdisk: [I] Processing vdisk BB1RGLLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGLDA2RESERVED
mmcrvdisk: [I] Processing vdisk BB1RGRLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGRLOGTIPBACKUP
mmcrvdisk: [I] Processing vdisk BB1RGRLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGRDA2RESERVED
mmcrvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Creation of the GSS 1.5 log vdisks may be verified through the **mmlsvdisk** and **mmlsrecoverygroup** commands:

```
# mmlsvdisk
                                                         declustered  block size
 vdisk name          RAID code        recovery group        array       in KiB    remarks
 ------------------  ---------------  ------------------  -----------  ----------  -------
 BB1RGLDA2RESERVED   4WayReplication  BB1RGL              DA2                2048  logRsvd
 BB1RGLLOGHOME       4WayReplication  BB1RGL              DA1                2048  log
 BB1RGLLOGTIP        2WayReplication  BB1RGL              NVR                2048  logTip
 BB1RGLLOGTIPBACKUP  3WayReplication  BB1RGL              SSD                2048  logTipBackup
 BB1RGRDA2RESERVED   4WayReplication  BB1RGR              DA2                2048  logRsvd
 BB1RGRLOGHOME       4WayReplication  BB1RGR              DA1                2048  log
 BB1RGRLOGTIP        2WayReplication  BB1RGR              NVR                2048  logTip
 BB1RGRLOGTIPBACKUP  3WayReplication  BB1RGR              SSD                2048  logTipBackup

# mmlsrecoverygroup BB1RGL -L

                     declustered
 recovery group        arrays     vdisks  pdisks  format version
 -----------------  -----------  ------  ------  --------------
 BB1RGL                       4       4     121  3.5.0.13

 declustered   needs                            replace                scrub      background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 SSD          no            1       3       0          1     558 GiB   14 days  scrub        4%  low
 NVR          no            1       2       0          1    3648 MiB   14 days  scrub        4%  low
 DA1          no            1      58       2          2     101 TiB   14 days  scrub        0%  low
 DA2          no            1      58       2          2     101 TiB   14 days  scrub        0%  low

                                         declustered                           checksum
 vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
 ------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------
 BB1RGLLOGTIP        2WayReplication     NVR              48 MiB       2 MiB          512  ok    logTip
 BB1RGLLOGTIPBACKUP  3WayReplication     SSD              48 MiB       2 MiB          512  ok    logTipBackup
 BB1RGLLOGHOME       4WayReplication     DA1              20 GiB       2 MiB          512  ok    log
 BB1RGLDA2RESERVED   4WayReplication     DA2              20 GiB       2 MiB          512  ok    logRsvd

 active recovery group server                     servers
 -----------------------------------------------  -------
 server1                                          server1,server2

# mmlsrecoverygroup BB1RGR -L

                    declustered
 recovery group       arrays     vdisks  pdisks  format version
 -----------------  -----------  ------  ------  --------------
 BB1RGR                       4       4     121  3.5.0.13

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 SSD          no            1       3       0          1     558 GiB   14 days  scrub        4%  low
 NVR          no            1       2       0          1    3648 MiB   14 days  scrub        4%  low
 DA1          no            1      58       2          2     101 TiB   14 days  scrub        1%  low
 DA2          no            1      58       2          2     101 TiB   14 days  scrub        1%  low

                                         declustered                           checksum
 vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
 ------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------
 BB1RGRLOGTIP        2WayReplication     NVR              48 MiB       2 MiB       512      ok    logTip
 BB1RGRLOGTIPBACKUP  3WayReplication     SSD              48 MiB       2 MiB       512      ok    logTipBackup
 BB1RGRLOGHOME       4WayReplication     DA1              20 GiB       2 MiB       512      ok    log
 BB1RGRDA2RESERVED   4WayReplication     DA2              20 GiB       2 MiB       512      ok    logRsvd

 active recovery group server                     servers
 -----------------------------------------------  -------
 server2                                          server2,server1
```

The creation of filesystem vdisks, NSDs, and of a filesystem can now proceed just as in the example for GSS without NVRAM partitions, by omitting the log vdisks from the stanza file and including only the desired filesystem vdisks.

Example scenario: Replacing failed disks in a GSS recovery group

The scenario presented here shows how to detect and replace failed disks in a recovery group built on a GSS building block.

Note: When replacing a failed disk, the **mmchcarrier** command will attempt to install the latest firmware on the disk by invoking the sample callback script /usr/lpp/mmfs/samples/vdisk/tspreparenewpdiskforuse (if present). To use this callback script, copy it to /usr/lpp/mmfs/bin/tspreparenewpdiskforuse and make it executable.

Detecting failed disks in your GSS enclosure

Assume a GSS-24 building block on which the following two recovery groups are defined:

- BB1RGL, containing the disks in the left side of each drawer
- BB1RGR, containing the disks in the right side of each drawer

Each recovery group contains the following:

- one log declustered array (LOG)
- two data declustered arrays (DA1, DA2)

The data declustered arrays are defined according to GSS-24 best practice as follows:

- 58 pdisks per data declustered array
- default disk replacement threshold value set to 2

The replacement threshold of 2 means that GPFS Native RAID will only require disk replacement when two or more disks have failed in the declustered array; otherwise, rebuilding onto spare space or reconstruction from redundancy will be used to supply affected data. This configuration can be seen in the output of **mmlsrecoverygroup** for the recovery groups, shown here for BB1RGL:

```
# mmlsrecoverygroup BB1RGL -L

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 BB1RGL                       3       5     119

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 LOG          no            1       3       0          1     534 GiB   14 days  scrub        1%  low
 DA1          yes           2      58       2          2       0   B   14 days  rebuild-1r   4%  low
 DA2          no            2      58       2          2    1024 MiB   14 days  scrub       27%  low

                                         declustered                           checksum
 vdisk               RAID code              array     vdisk size  block size  granularity  remarks
 ------------------  ------------------  -----------  ----------  ----------  -----------  -------
 BB1RGLLOG           3WayReplication     LOG            8192 MiB    256 KiB       512      log
 BB1RGLMETA1         4WayReplication     DA1            1024 GiB      1 MiB     32 KiB
 BB1RGLDATA1         8+3p                DA1              70 TiB     16 MiB     32 KiB
 BB1RGLMETA2         4WayReplication     DA2            1024 GiB      1 MiB     32 KiB
 BB1RGLDATA2         8+3p                DA2              70 TiB     16 MiB     32 KiB

 active recovery group server                     servers
 -----------------------------------------------  -------
 server1                                          server1,server2
```

The indication that disk replacement is called for in this recovery group is the value of yes in the needs service column for declustered array DA1.

The fact that DA1 is undergoing rebuild of its RAID tracks that can tolerate one strip failure is by itself not an indication that disk replacement is required; it merely indicates that data from a failed disk is being rebuilt onto spare space. Only if the replacement threshold has been met will disks be marked for replacement and the declustered array marked as needing service.

GPFS Native RAID provides several indications that disk replacement is required:

- entries in the Linux syslog
- the GPFS **pdReplacePdisk** callback, which can be configured to run an administrator-supplied script at the moment a pdisk is marked for replacement
- the output from the following commands, which may be performed from the command line on any GPFS cluster node (see the examples that follow):
  1. **mmlsrecoverygroup** with the **-L** flag shows yes in the needs service column
  2. **mmlsrecoverygroup** with the **-L** and **--pdisk** flags; this shows the states of all pdisks, which may be examined for the replace pdisk state
  3. **mmlspdisk** with the **--replace** flag, which lists only those pdisks that are marked for replacement

Note: Because the output of **mmlsrecoverygroup -L --pdisk** is very long, this example shows only some of the pdisks (but includes those marked for replacement).

```
# mmlsrecoverygroup BB1RGL -L --pdisk

                    declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 BB1RGL                       3       5     119

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 LOG          no            1       3       0          1     534 GiB   14 days  scrub        1%  low
```

```
DA1          yes           2      58       2          2       0   B   14 days  rebuild-1r   4%  low
DA2          no            2      58       2          2    1024 MiB   14 days  scrub       27%  low

                    number of    declustered
pdisk              active paths     array     free space  state
-----------------  ------------  -----------  ----------  -----
[...]
e1d4s06                 2        DA1              62 GiB  ok
e1d5s01                 0        DA1              70 GiB  slow/noPath/systemDrain/noRGD/noVCD/replace
e1d5s02                 2        DA1              64 GiB  ok
e1d5s03                 2        DA1              63 GiB  ok
e1d5s04                 0        DA1              64 GiB  failing/noPath/systemDrain/noRGD/noVCD/replace
e1d5s05                 2        DA1              63 GiB  ok
[...]
```

The preceding output shows that the following pdisks are marked for replacement:

- e1d5s01 in DA1
- e1d5s04 in DA1

The naming convention used during recovery group creation indicates that these are the disks in Enclosure 1 Drawer 5 Slot 1 and Enclosure 1 Drawer 5 Slot 4. To confirm the physical locations of the failed disks, use the **mmlspdisk** command to list information about those pdisks in declustered array DA1 of recovery group BB1RGL that are marked for replacement:

```
# mmlspdisk BB1RGL --declustered-array DA1 --replace
pdisk:
   replacementPriority = 0.98
   name = "e1d5s01"
   device = ""
   recoveryGroup = "BB1RGL"
   declusteredArray = "DA1"
   state = "slow/noPath/systemDrain/noRGD/noVCD/replace"
   freeSpace = 75161927680
   fru = "90Y8597"
   location = "SV21314035-5-1"
   WWN = "naa.5000CCA01B510EFC"
   server = "server1"
   reads = 32750112
   writes = 53114318
   bytesReadInGiB = 15585.770
   bytesWrittenInGiB = 25117.229
   IOErrors = 0
   IOTimeouts = 41
   mediaErrors = 0
   checksumErrors = 0
   pathErrors = 0
   relativePerformance = 0.661
   dataBadness = 0.000
   rgIndex = 26
   userLocation = "Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 1"
pdisk:
   replacementPriority = 0.98
   name = "e1d5s04"
   device = ""
   recoveryGroup = "BB1RGL"
   declusteredArray = "DA1"
   state = "failing/noPath/systemDrain/noRGD/noVCD/replace"
   freeSpace = 68719476736
   fru = "90Y8597"
   location = "SV21314035-5-4"
   WWN = "naa.5000CCA01B4B702C"
   server = "server1"
   reads = 51176754
   writes = 80568342
   bytesReadInGiB = 24385.029
   bytesWrittenInGiB = 38243.908
   IOErrors = 11
   IOTimeouts = 422
   mediaErrors = 0
   checksumErrors = 0
   pathErrors = 3
   relativePerformance = 0.624
   dataBadness = 0.000
   rgIndex = 29
   userLocation = "Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 4"
```

The physical locations of the failed disks are confirmed to be consistent with the pdisk naming convention and with the GPFS Native RAID component database:

```
-------------------------------------------------------------------------------------
Disk          Location         User Location
-------------------------------------------------------------------------------------
pdisk e1d5s01  SV21314035-5-1  Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 1
-------------------------------------------------------------------------------------
pdisk e1d5s04  SV21314035-5-4  Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 4
-------------------------------------------------------------------------------------
```

This shows how the component database provides an easier-to-use location reference for the affected physical disks. The pdisk name e1d5s01 signifies very simply “Enclosure 1 Drawer 5 Slot 1.” The location additionally provides the serial number of enclosure 1, SV21314035, with the drawer and slot number. But the user location that has been defined in the component database can be used to precisely locate the disk in a given equipment rack and a named disk enclosure: This is the disk enclosure labelled “BB1ENC1,” found in compartments U01 - U04 of the rack labelled “BB1,” and the disk is in drawer 5, slot 1 of that enclosure.

The relationship between the enclosure serial number and the user location can be seen with the **mmlscomp** command:

```
# mmlscomp --serial-number SV21314035

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name     Display ID
-------  -----------  -------------  -------  ----------
      2  181880E      SV21314035     BB1ENC1
```

Replacing failed disks in a GSS-24 recovery group

Note: In this example, it is assumed that two new disks with the appropriate Field Replaceable Unit (FRU) code, as indicated by the fru attribute (90Y8597 in this case), have been obtained as replacements for the failed pdisks e1d5s01 and e1d5s04.

Replacing each disk is a three-step process:

1. Using the **mmchcarrier** command with the **--release** flag to inform GPFS to locate the disk, suspend it, and allow it to be removed.
2. Locating and removing the failed disk and replacing it with a new one.
3. Using the **mmchcarrier** command with the **--replace** flag to begin use of the new disk.

GPFS Native RAID assigns a priority to pdisk replacement. Disks with smaller values for the **replacementPriority** attribute should be replaced first. In this example, the only failed disks are in DA1 and both have the same **replacementPriority**.

Disk e1d5s01 is chosen to be replaced first.

1. To release pdisk e1d5s01 in recovery group BB1RGL:

```
# mmchcarrier BB1RGL --release --pdisk e1d5s01
[I] Suspending pdisk e1d5s01 of RG BB1RGL in location SV21314035-5-1.
[I] Location SV21314035-5-1 is Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 1.
[I] Carrier released.

  - Remove carrier.
  - Replace disk in location SV21314035-5-1 with FRU 90Y8597.
  - Reinsert carrier.
  - Issue the following command:

      mmchcarrier BB1RGL --replace --pdisk 'e1d5s01'

    Repair timer is running.  Perform the above within 5 minutes
    to avoid pdisks being reported as missing.
```

GPFS Native RAID issues instructions as to the physical actions that must be taken, and repeats the user-defined location to assist in finding the disk. As disk replacement in a GSS-24 building block affects only the disk being replaced, the five-minute warning serves mostly as a reminder to act promptly.

2. To allow the enclosure BB1ENC1 with serial number SV21314035 to be located and identified, GPFS Native RAID will turn on the enclosure's amber "service required" LED. The enclosure's bezel should be removed. This will reveal that the amber "service required" and blue "service allowed" LEDs for drawer 5 have been turned on.

   Drawer 5 should then be unlatched and pulled open. The disk in slot 1 will be seen to have its amber and blue LEDs turned on.

   Unlatch and pull up the handle for the identified disk in slot 1. Lift the failed disk out and set it aside. The drive LEDs will turn off when the slot is empty. A new disk with FRU 90Y8597 should be lowered in place and have its handle pushed down and latched.

   Since the second disk replacement in this example is also in drawer 5 of the same enclosure, leave the drawer open and the enclosure bezel off. If the next replacement were in a different drawer, the drawer would be closed; and if the next replacement were in a different enclosure, the enclosure bezel would be replaced.

3. To finish the replacement of pdisk e1d5s01:

```
# mmchcarrier BB1RGL --replace --pdisk e1d5s01
[I] The following pdisks will be formatted on node server1:
    /dev/sdmi
[I] Pdisk e1d5s01 of RG BB1RGL successfully replaced.
[I] Resuming pdisk e1d5s01#026 of RG BB1RGL.
[I] Carrier resumed.
```

   When the **mmchcarrier --replace** command returns successfully, GPFS Native RAID will begin rebuilding and rebalancing RAID strips onto the new disk, which has assumed the pdisk name e1d5s01. The failed pdisk may remain in a temporary form (indicated here by the name e1d5s01#026) until all data from it has been rebuilt, at which point it is finally deleted. Notice that only one block device name is mentioned as being formatted as a pdisk; the second path will be discovered in the background.

   Disk e1d5s04 will still be marked for replacement, and DA1 of BBRGL will still need service. This is because GPFS Native RAID replacement policy expects all failed disks in the declustered array to be replaced once the replacement threshold is reached.

Pdisk e1d5s04 is then replaced following the same process.

1. Release pdisk e1d5s04 in recovery group BB1RGL:

```
# mmchcarrier BB1RGL --release --pdisk e1d5s04
[I] Suspending pdisk e1d5s04 of RG BB1RGL in location SV21314035-5-4.
[I] Location SV21314035-5-4 is Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 4.
[I] Carrier released.

  - Remove carrier.
  - Replace disk in location SV21314035-5-4 with FRU 90Y8597.
  - Reinsert carrier.
  - Issue the following command:

      mmchcarrier BB1RGL --replace --pdisk 'e1d5s04'

    Repair timer is running.  Perform the above within 5 minutes
    to avoid pdisks being reported as missing.
```

2. Find the enclosure and drawer, unlatch and remove the disk in slot 4, place a new disk in slot 4, push in the drawer, and replace the enclosure bezel.
3. To finish the replacement of pdisk e1d5s04:

```
# mmchcarrier BB1RGL --replace --pdisk e1d5s04
[I] The following pdisks will be formatted on node server1:
    /dev/sdfd
[I] Pdisk e1d5s04 of RG BB1RGL successfully replaced.
[I] Resuming pdisk e1d5s04#029 of RG BB1RGL.
[I] Carrier resumed.
```

The disk replacements may be confirmed with **mmlsrecoverygroup -L --pdisk**:

```
# mmlsrecoverygroup BB1RGL -L --pdisk

                   declustered
 recovery group      arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 BB1RGL                       3       5     121

 declustered   needs                            replace                scrub      background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 LOG          no            1       3       0          1     534 GiB   14 days  scrub        1%  low
 DA1          no            2      60       2          2    3647 GiB   14 days  rebuild-1r   4%  low
 DA2          no            2      58       2          2    1024 MiB   14 days  scrub       27%  low

                    number of    declustered
 pdisk             active paths    array     free space  state
 -----------------  ------------  -----------  ----------  -----
 [...]
 e1d4s06                 2        DA1             62 GiB  ok
 e1d5s01                 2        DA1           1843 GiB  ok
 e1d5s01#026             0        DA1             70 GiB  slow/noPath/systemDrain/adminDrain/noRGD/noVCD
 e1d5s02                 2        DA1             64 GiB  ok
 e1d5s03                 2        DA1             63 GiB  ok
 e1d5s04                 2        DA1           1853 GiB  ok
 e1d5s04#029             0        DA1             64 GiB  failing/noPath/systemDrain/adminDrain/noRGD/noVCD
 e1d5s05                 2        DA1             62 GiB  ok
 [...]
```

Notice that the temporary pdisks (e1d5s01#026 and e1d5s04#029) representing the now-removed physical disks are counted toward the total number of pdisks in the recovery group BB1RGL and the declustered array DA1. They will exist until RAID rebuild completes the reconstruction of the data that they carried onto other disks (including their replacements). When rebuild completes, the temporary pdisks will disappear, and the number of disks in DA1 will once again be 58, and the number of disks in BBRGL will once again be 119.

Example scenario: Replacing failed GSS storage enclosure components

The scenario presented here shows how to detect and replace failed storage enclosure components in a GSS building block.

Detecting failed storage enclosure components

The **mmlsenclosure** command can be used to show you which enclosures need service along with the specific component. A best practice is to run this command each day to check for failures.

```
# mmlsenclosure all -L --not-ok

                    needs  nodes
 serial number     service
 -------------     ------- ------
 SV21313971        yes     c45f02n01-ib0.gpfs.net

 component type  serial number     component id    failed value   unit   properties
 --------------  -------------     ------------    ------ -----   ----   ----------
 fan             SV21313971        1_BOT_LEFT      yes            RPM    FAILED
```

This indicates that enclosure SV21313971 has a failed fan. Refer to *IBM GPFS Storage Server Hardware Installation and Service Guide*.

Once you are ready to replace the failed component, use the **mmchenclosure** command to identify whether it is safe to complete the repair action or whether GPFS needs to be shut down first:

```
# mmchenclosure SV21313971 --component fan --component-id 1_BOT_LEFT

mmenclosure: Proceed with the replace operation.
```

The fan can now be replaced.

Special note about detecting failed enclosure components

In the following example, only the enclosure itself is being called out as having failed; the specific component that has actually failed is not identified. This typically means that there are drive “Service Action Required (Fault)” LEDs that have been turned on in the drawers. In such a situation, the **mmlspdisk all --not-ok** command can be used to check for dead or failing disks.

```
mmlsenclosure all -L --not-ok

                    needs  nodes
 serial number     service
 -------------     ------- ------
 SV13306129        yes     c45f01n01-ib0.gpfs.net

 component type  serial number     component id    failed value   unit   properties
 --------------  -------------     ------------    ------ -----   ----   ----------
 enclosure       SV13306129        ONLY            yes                   NOT_IDENTIFYING,FAILED
```

Example scenario: Replacing a failed GSS storage enclosure drawer

The scenario presented here shows how to use the **chdrawer** sample script as a service aid in replacing a GSS enclosure drawer while GPFS is active.

The **chdrawer** sample script is intended to be used in GSS-26 configurations where the declustered arrays have been laid out across all the drawers so that removing a drawer affects only two disks in each declustered array.

Notes:

1. In this layout, follow these recommendations:
   - Run 8+3p or 4-way mirroring only
   - Always have two spares per declustered array.

   You can use 8+2p or 3-way mirroring, but this means that once the drawer is removed, every declustered array will be running in critical mode. Therefore, one more sector error or disk failure will result in data being unavailable or worse.
2. A drawer can be replaced in a GSS-24 configuration laid out such that removing a drawer affects three disks in each declustered array, but *only* in the case where the vdisks were using 8+3p or 4-way mirroring.

The syntax of the **chdrawer** sample script is:

```
chdrawer EnclosureSerialNumber DrawerNumber
     {--release [--force-release] | --replace [--force-replace]} [--dry-run]
```

where:

EnclosureSerialNumber
Specifies the enclosure serial number, as displayed by **mmlsenclosure**.

DrawerNumber
Specifies the drawer number to be replaced. Drawers are numbered from top to bottom in an enclosure.

--release
Prepares the drawer for disk replacement by draining all the pdisks.

--replace
Resumes all the pdisks once the drawer has been replaced (with all the former pdisks in their corresponding slots).

--dry-run
Runs the command without actually changing the pdisk states; checks whether there is enough redundancy to safely replace the drawer on a live system.

--force-release
Forces the **--release** option. Changes the pdisk states to `serviceDrain` despite warnings.

--force-replace
Forces the **--replace** option. Ends the service drain state despite warnings.

To replace drawer `3` in enclosure `SV21106537`, follow these steps:

1. Start the release sequence using the following command:
   ```
   /usr/lpp/mmfs/samples/vdisk/chdrawer SV21106537 3 --release
   ```
   The command performs the following actions:
   a. Discovers all the pdisks that are in the drawer.
   b. Checks that each declustered array has two spares.
   c. Checks that there are no other disks down that might prevent a successful rebuild.
   d. Places the pdisks in `serviceDrain` state, which will drain all the data from the affected pdisks.
   e. Waits until all the data is drained.
2. Record the location of the drives and label them, as you will later need to replace them in the corresponding slots of the new drawer.
3. Disconnect the drawer, repair it, and put it back in. (See *IBM GPFS Storage Server Hardware Installation and Service Guide* for physical replacement instructions.)
4. Complete the replacement sequence using the following command:
   ```
   /usr/lpp/mmfs/samples/vdisk/chdrawer SV21106537 3 --replace
   ```
   The command performs the following actions:
   a. Verifies that the expected number of pdisks have been found.
   b. Removes the `serviceDrain` state, which will start rebalancing data back onto the pdisks.

Example scenario: Replacing a failed GSS storage enclosure

The scenario presented here shows how to replace a failed GSS storage enclosure.

To replace a filed storage enclosure, follow these steps:

1. Shut down GPFS across the cluster by issuing the following command:
   ```
   mmshutdown -a
   ```

2. Record the drawer and slot locations of the drives and label them, as you will need to move them to the corresponding slots of the new drawers of the new enclosure.
3. Remove the SAS connections in the rear of the enclosure.
4. Remove the enclosure.
5. Install the new enclosure.
6. Transfer the disks to the new drawers of the new enclosure, referring to the information saved in step 2.
7. Connect the SAS connections in the rear of the new enclosure.
8. Power up the enclosure.
9. Verify the SAS topology on the servers to ensure that all drives from the new storage enclosure are present.
10. Update the necessary firmware on the new storage enclosure as needed.
11. Startup GPFS across the cluster

    ```
    mmstartup -a
    ```

Example scenario: Checking the health of a GSS configuration

The scenario presented here shows how to use the **gnrhealthcheck** sample script to check the general health of a GSS configuration.

The syntax of the **gnrhealthcheck** sample script is:

```
gnrhealthcheck [--topology] [--enclosure] [--rg] [--pdisk]
               [--perf-dd] [--local]
```

where:

--topology
Checks the operating system topology. Runs **mmgetpdisktopology** and **topsummary** to look for cabling and path issues.

--enclosure
Checks enclosures. Runs **mmlsenclosure** to look for failures.

--rg
Checks recovery groups. Runs **mmlsrecoverygroup** to check whether all recovery groups are active and whether the active server is the primary server. Also checks for any recovery groups that need service.

--pdisk
Checks pdisks. Runs **mmlspdisk** to check that each pdisk has two paths.

--perf-dd
Checks basic performance of disks. Runs a dd read to each potential GNR disk drive for a GB and reports basic performance stats. Reads are done six disks at a time. These statistics will only be meaningful if run on an idle system. Available on Linux only.

--local
Runs tests only on the invoking node.

The default is to check everything *except* **--perf-dd** arguments on all NSD server nodes.

Output return codes:

0 No problems were found.

1 Problems were found, and information was displayed.

Note: The default is to display to stdout. There may be a large amount of data, so it is recommended that you pipe the output to a file.

Examples

1. To run a health check on the local server nodes and place output in `/tmp/gnrhealthcheck.out`, issue the following command:

   ```
   gnrhealthcheck --local | tee /tmp/gnrhealthcheck.out
   ```

   In this example all checks are successful.

   The system displays information similar to this:

   ```
   ################################################################
   # Beginning topology checks.
   ################################################################
   Topology checks successful.

   ################################################################
   # Beginning enclosure checks.
   ################################################################
   Enclosure checks successful.

   ################################################################
   # Beginning recovery group checks.
   ################################################################
   Recovery group checks successful.

   ################################################################
   # Beginning pdisk checks.
   ################################################################

   Pdisk group checks successful.
   ```

2. To run a health check on the local server nodes and place output in `/tmp/gnrhealthcheck.out`, issue the following command:

   ```
   gnrhealthcheck --local | tee /tmp/gnrhealthcheck.out
   ```

   In this example, there are numerous issues to be investigated.

   The system displays information similar to this:

   ```
   ################################################################
   # Beginning topology checks.
   ############################################################
   Found topology problems on node c45f01n01-ib0.gpfs.net

   DCS3700 enclosures found: 0123456789AB SV11812206 SV12616296 SV13306129
   Enclosure 0123456789AB (number 1):
   Enclosure 0123456789AB ESM A sg244[0379][scsi8 port 4] ESM B sg4[0379][scsi7 port 4]
   Enclosure 0123456789AB Drawer 1 ESM sg244 12 disks diskset "19968" ESM sg4 12 disks diskset "19968"
   Enclosure 0123456789AB Drawer 2 ESM sg244 12 disks diskset "11294" ESM sg4 12 disks diskset "11294"
   Enclosure 0123456789AB Drawer 3 ESM sg244 12 disks diskset "60155" ESM sg4 12 disks diskset "60155"
   Enclosure 0123456789AB Drawer 4 ESM sg244 12 disks diskset "03345" ESM sg4 12 disks diskset "03345"
   Enclosure 0123456789AB Drawer 5 ESM sg244 11 disks diskset "33625" ESM sg4 11 disks diskset "33625"
   Enclosure 0123456789AB sees 59 disks

   Enclosure SV12616296 (number 2):
   Enclosure SV12616296 ESM A sg63[0379][scsi7 port 3] ESM B sg3[0379][scsi9 port 4]
   Enclosure SV12616296 Drawer 1 ESM sg63 11 disks diskset "51519" ESM sg3 11 disks diskset "51519"
   Enclosure SV12616296 Drawer 2 ESM sg63 12 disks diskset "36246" ESM sg3 12 disks diskset "36246"
   Enclosure SV12616296 Drawer 3 ESM sg63 12 disks diskset "53750" ESM sg3 12 disks diskset "53750"
   Enclosure SV12616296 Drawer 4 ESM sg63 12 disks diskset "07471" ESM sg3 12 disks diskset "07471"
   Enclosure SV12616296 Drawer 5 ESM sg63 11 disks diskset "16033" ESM sg3 11 disks diskset "16033"
   Enclosure SV12616296 sees 58 disks

   Enclosure SV11812206 (number 3):
   Enclosure SV11812206 ESM A sg66[0379][scsi9 port 3] ESM B sg6[0379][scsi8 port 3]
   Enclosure SV11812206 Drawer 1 ESM sg66 11 disks diskset "23334" ESM sg6 11 disks diskset "23334"
   Enclosure SV11812206 Drawer 2 ESM sg66 12 disks diskset "16332" ESM sg6 12 disks diskset "16332"
   Enclosure SV11812206 Drawer 3 ESM sg66 12 disks diskset "52806" ESM sg6 12 disks diskset "52806"
   Enclosure SV11812206 Drawer 4 ESM sg66 12 disks diskset "28492" ESM sg6 12 disks diskset "28492"
   Enclosure SV11812206 Drawer 5 ESM sg66 11 disks diskset "24964" ESM sg6 11 disks diskset "24964"
   Enclosure SV11812206 sees 58 disks
   ```

```
Enclosure SV13306129 (number 4):
Enclosure SV13306129 ESM A sg64[0379][scsi8 port 2] ESM B sg353[0379][scsi7 port 2]
Enclosure SV13306129 Drawer 1 ESM sg64 11 disks diskset "47887" ESM sg353 11 disks diskset "47887"
Enclosure SV13306129 Drawer 2 ESM sg64 12 disks diskset "53906" ESM sg353 12 disks diskset "53906"
Enclosure SV13306129 Drawer 3 ESM sg64 12 disks diskset "35322" ESM sg353 12 disks diskset "35322"
Enclosure SV13306129 Drawer 4 ESM sg64 12 disks diskset "37055" ESM sg353 12 disks diskset "37055"
Enclosure SV13306129 Drawer 5 ESM sg64 11 disks diskset "16025" ESM sg353 11 disks diskset "16025"
Enclosure SV13306129 sees 58 disks

DCS3700 configuration: 4 enclosures, 1 SSD, 7 empty slots, 233 disks total
Location 0123456789AB-5-12 appears empty but should have an SSD
Location SV12616296-1-3 appears empty but should have an SSD
Location SV12616296-5-12 appears empty but should have an SSD
Location SV11812206-1-3 appears empty but should have an SSD
Location SV11812206-5-12 appears empty but should have an SSD

scsi7[07.00.00.00] 0000:11:00.0 [P2 SV13306129 ESM B (sg353)] [P3 SV12616296 ESM A (sg63)] [P4 0123456789AB ESM B (sg4)]
scsi8[07.00.00.00] 0000:8b:00.0 [P2 SV13306129 ESM A (sg64)] [P3 SV11812206 ESM B (sg6)] [P4 0123456789AB ESM A (sg244)]
scsi9[07.00.00.00] 0000:90:00.0 [P3 SV11812206 ESM A (sg66)] [P4 SV12616296 ESM B (sg3)]

################################################################
# Beginning enclosure checks.
################################################################
Enclosure checks successful.

################################################################
# Beginning recovery group checks.
################################################################
Found recovery group BB1RGR, primary server is not the active server.

################################################################
# Beginning pdisk checks.
################################################################
Found recovery group BB1RGL pdisk e4d5s06 has 0 paths.
```

Changes and additions to *GPFS: Administration and Programming Reference* (SA23-2221-08)

This update provides new and changed information for the previously published GPFS Version 3 Release 5.0.11 *Administration and Programming Reference*.

mmchpdisk command

Changes GPFS Native RAID pdisk states. This command is to be used only in extreme situations under the guidance of IBM service personnel.

Synopsis

```
mmchpdisk RecoveryGroupName --pdisk PdiskName
      {--kill | --revive | --suspend | --resume | --diagnose | --identify {on|off} |
       --begin-service-drain | --end-service-drain}
```

Description

The **mmchpdisk** command changes the states of pdisks.

Attention: This command is to be used only in extreme situations under the guidance of IBM service personnel.

Parameters

RecoveryGroupName
Specifies the recovery group containing the pdisk for which one or more states are to be changed.

--pdisk *PdiskName*
Specifies the target pdisk.

--kill
Forces the failure of a pdisk by setting the `dead` pdisk state flag.

Attention: This option must be used with caution; if the total number of failures in a declustered array exceeds the fault tolerance of any vdisk in that array, permanent data loss might result.

--revive
Attempts to make a failed disk usable again by removing `dead`, `failing`, and `readonly` pdisk state flags. Data allocated on the disk that has not been rebuilt onto spare space can become readable again; however, any data that has already been reported as lost cannot be recovered.

--suspend
Suspends I/O to the pdisk until a subsequent resume command is given. If a pdisk remains in the `suspended` state for longer than a predefined timeout period, GPFS Native RAID begins rebuilding the data from that pdisk into spare space.

Attention: This option is to be used with caution and only when performing maintenance on disks manually, bypassing the automatic system provided by **mmchcarrier**.

If a pdisk is removed using this option, vdisks that store data on the removed pdisk will become temporarily degraded, requiring data that was stored on the removed pdisk to be rebuilt from redundant data. Also, if you try to remove more pdisks than the redundancy level of the least redundant vdisk in that declustered array, data will become inaccessible.

Therefore, when preparing to remove a pdisk, use the **--begin-service-drain** and **--end-service-drain** options instead of this option.

--resume
Cancels a previously given **mmchpdisk --suspend** command and resumes use of the pdisk.

Use this option only when performing maintenance on disks manually and bypassing the automatic system provided by **mmchcarrier**.

--diagnose
Runs basic tests on the pdisk. If no problems are found, the pdisk state automatically returns to `ok`.

--identify {on | off}
Turns on or off the disk identify light, if available.

--begin-service-drain
Starts draining the specified pdisk so that it can be temporarily removed. After issuing the command with this option, wait until the pdisk has been completely drained before removing the pdisk.

Note: This process requires that there be sufficient spare space in the declustered array for the data that is to be drained. If the available spare space is insufficient, it can be increased with the **mmchrecoverygroup** command.

--end-service-drain
Returns drained data to a pdisk after the pdisk has been brought back online.

Exit status

0 Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmchpdisk** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *GPFS: Administration and Programming Reference* topic:

- Requirements for administering a GPFS file system (not included in this *Documentation Update*)

Examples

The following command example shows how to tell GPFS to attempt to revive the failed pdisk `c036d3` in recovery group `000DE37BOT`:

```
mmchpdisk 000DE37BOT --pdisk c036d3 --revive
```

See also

See also the following *GPFS: Administration and Programming Reference* topics:

- mmaddpdisk command (not included in this *Documentation Update*)
- mmchcarrier command (not included in this *Documentation Update*)
- mmdelpdisk command (not included in this *Documentation Update*)
- mmlspdisk command (not included in this *Documentation Update*)
- “mmlsrecoverygroup command” on page 93

Location

```
/usr/lpp/mmfs/bin
```

mmchrecoverygroup command

Changes GPFS Native RAID recovery group and declustered array attributes.

Synopsis

```
mmchrecoverygroup RecoveryGroupName {--declustered-array DeclusteredArrayName {[--spares NumberOfSpares]
                                    [--scrub-duration NumberOfDays] [--replace-threshold NumberOfDisks]}
                                    [--version {VersionString | LATEST}]}
```

or

```
mmchrecoverygroup RecoveryGroupName --active ServerName
```

or

```
mmchrecoverygroup RecoveryGroupName --servers Primary[,Backup] [-v {yes | no}]
```

Description

The **mmchrecoverygroup** command changes recovery group and declustered array attributes.

Parameters

RecoveryGroupName
: Specifies the name of the recovery group being changed.

--declustered-array *DeclusteredArrayName*
: Specifies the name of the declustered array being changed.

--spares *NumberOfSpares*
: Specifies the number of disks' worth of spare space to set aside in the declustered array. This space is used to rebuild the declustered arrays when physical disks fail.

--scrub-duration *NumberOfDays*
: Specifies the number of days, from 1 to 60, for the duration of the scrub. The default value is 14.

--replace-threshold *NumberOfDisks*
: Specifies the number of pdisks that must fail in the declustered array before **mmlsrecoverygroup** will report that service (disk replacement) is needed.

--version {*VersionString* **| LATEST}**
: Specifies the recovery group version.

  Valid version strings are the following:

  3.5.0.5
  : Denotes the version with all features supported prior to GSS v1.5.

  3.5.0.13
  : Denotes the version with all features supported by GSS v1.5.

  LATEST
  : Denotes the latest supported version. New recovery groups will default to the latest version.

--active *ServerName*
: Changes the active server for the recovery group.

--servers *Primary***[,***Backup***]**
: Changes the defined list of recovery group servers.

  Note: To use this option, all file systems that use this recovery group must be unmounted.

-v {yes | no}
: Specifies whether the new server or servers should verify access to the pdisks of the recovery group.

The default is **-v yes**. Use **-v no** to specify that configuration changes should be made without verifying pdisk access; for example, you could use this if all the servers were down.

Exit status

0 Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmchrecoverygroup** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *GPFS: Administration and Programming Reference* topic:

- Requirements for administering a GPFS file system (not included in this *Documentation Update*)

Examples

1. The following command example shows how to change the number of spares to one and the replacement threshold to one for declustered array DA4 in recovery group 000DE37TOP:

   ```
   mmchrecoverygroup 000DE37TOP --declustered-array DA4 --spares 1 --replace-threshold 1
   ```

2. The following command example shows how to change the scrub duration to one day for declustered array DA2 of recovery group 000DE37BOT:

   ```
   mmchrecoverygroup 000DE37BOT --declustered-array DA2 --scrub-duration 1
   ```

3. The following command example shows how to replace the servers for a recovery group. In this example, assume that the two current servers for recovery group RG1 have been taken down for extended maintenance. GPFS is shut down on these current RG1 servers. The new servers have already been configured to be recovery group servers, with **mmchconfig** parameter **nsdRAIDTracks** set to a nonzero value. The disks for RG1 have not yet been connected to the new servers, so verification of disk availability must be disabled by specifying **-v no** as shown here:

   ```
   mmchrecoverygroup RG1 --servers newprimary,newbackup -v no
   ```

See also

See also the following *GPFS: Administration and Programming Reference* topics:

- "mmchpdisk command" on page 80
- mmdelrecoverygroup command (not included in this *Documentation Update*)
- "mmcrrecoverygroup command" on page 84
- "mmlsrecoverygroup command" on page 93

Location

```
/usr/lpp/mmfs/bin
```

mmcrrecoverygroup command

Creates a GPFS Native RAID recovery group and its component declustered arrays and pdisks and specifies the servers.

Synopsis

```
mmcrrecoverygroup RecoveryGroupName -F StanzaFile --servers {Primary[,Backup]}
                  [--version {VersionString | LATEST}]
                  [-v {yes | no}]
```

Description

The **mmcrrecoverygroup** command is used to define a cluster-wide recovery group for use by GPFS Native RAID. A recovery group is a set of physical disks shared by up to two server nodes. The set of disks must be partitioned into one or more declustered arrays.

See the topic about managing GPFS Native RAID in the *GPFS: Advanced Administration Guide*.

The pdisk stanzas assigned to a recovery group must contain at least one declustered array that meets the definition of *large*.

While the **mmcrrecoverygroup** command creates the declustered arrays and pdisks and defines the servers, further processing with the **mmcrvdisk** and **mmcrnsd** commands is necessary to create GPFS Native RAID vdisk NSDs within the recovery group.

Note: The recovery group servers must be active to run this command.

Parameters

RecoveryGroupName
Name of the recovery group being created.

-F *StanzaFile*
Specifies a file that includes pdisk stanzas and declustered array stanzas that are used to create the recovery group. The declustered array stanzas are optional.

Pdisk stanzas look like the following:

```
%pdisk: pdiskName=PdiskName
        device=BlockDeviceName
        da=DeclusteredArrayName
```

where:

pdiskName=*PdiskName*
Specifies the name of a pdisk.

device=*BlockDeviceName*
Specifies the name of a block device. The value provided for *BlockDeviceName* must refer to the block device as configured by the operating system on the primary recovery group server or have the node name prefixed to the device block name.

Example values for *BlockDeviceName* are `hdisk3`, `/dev/hdisk3`, and `//nodename/dev/hdisk3`.

Only one *BlockDeviceName* needs to be used, even if the device uses multipath and has multiple device names.

da=*DeclusteredArrayName*
Specifies the *DeclusteredArrayName* in the pdisk stanza, which implicitly creates the declustered array with default parameters.

Declustered array stanzas look like the following:

```
%da: daName=DeclusteredArrayName
  spares=Number
  replaceThreshold=Number
  scrubDuration=Number
  auLogSize=Number
  nspdEnable={yes|no}
```

where:

daName=*DeclusteredArrayName*
Specifies the name of the declustered array for which you are overriding the default values.

spares=*Number*
Specifies the number of disks' worth of spare space to set aside in the declustered array. The number of spares can be 1 or higher. The default values are the following:

1 for arrays with 9 or fewer disks

2 for arrays with 10 or more disks

replaceThreshold=*Number*
Specifies the number of pdisks that must fail in the declustered array before **mmlspdisk** will report that pdisks need to be replaced. The default is equal to the number of spares.

scrubDuration=*Number*
Specifies the length of time (in days) by which the scrubbing of entire array must be completed. Valid values are 1 to 60. The default value is 14 days.

auLogSize
Specifies the size of the atomic update log. This value must be chosen very carefully and typically only needs to be set for a declustered array consisting of nonvolatile RAM disks. See the *GPFS: Advanced Administration Guide* example scenario for configuring GPFS Native RAID recovery groups on the GSS.

nspdEnable {yes|no}
Specifies whether this declustered array should be enabled for network shared pdisks. Declustered arrays that contain non-volatile RAM disks must set **nspdEnable=yes**. The default value is no. See the *GPFS: Advanced Administration Guide* example scenario for configuring GPFS Native RAID recovery groups on the GSS.

--servers {*Primary***[,***Backup***]}**
Specifies the primary server and, optionally, a backup server.

--version {*VersionString* **| LATEST}**
Specifies the recovery group version.

The valid version strings are:

3.5.0.5
Denotes the version with all features supported prior to GSS v1.5.

3.5.0.13
Denotes the version with all features supported by GSS v1.5.

LATEST
Denotes the latest supported version. New recovery groups will default to the latest version.

-v {yes | no}
Verification flag that specifies whether each pdisk in the stanza file should only be created if it has not been previously formatted as a pdisk (or NSD). The default is **-v yes**. Use **-v no** to specify that the disks should be created regardless of whether they have been previously formatted or not.

Exit status

0 Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmcrrecoverygroup** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Examples

Assume that input stanza file 000DE37BOT contains the following lines:

```
%pdisk: pdiskName=c034d1
        device=/dev/hdisk316
        da=DA1
%pdisk: pdiskName=c034d2
        device=/dev/hdisk317
        da=DA2
%pdisk: pdiskName=c034d3
        device=/dev/hdisk318
        da=DA3
%pdisk: pdiskName=c034d4
        device=/dev/hdisk319
        da=DA4
%pdisk: pdiskName=c033d1
        device=/dev/hdisk312
        da=LOG
[...]
```

The following command example shows how to create recovery group 000DE37BOT using stanza file 000DE37BOT, with c250f10c08ap01-hf0 as the primary server and c250f10c07ap01-hf0 as the backup server:

```
mmcrrecoverygroup 000DE37BOT -F 000DE37BOT --servers c250f10c08ap01-hf0,c250f10c07ap01-hf0
```

The system displays output similar to the following:

```
mmcrrecoverygroup: Propagating the cluster configuration data to all
   affected nodes.  This is an asynchronous process.
```

See also

See also the following *GPFS: Administration and Programming Reference* topics:

- "mmchrecoverygroup command" on page 82
- "mmcrvdisk command" on page 87
- mmdelrecoverygroup command (not included in this *Documentation Update*)
- "mmlsrecoverygroup command" on page 93

Location

```
/usr/lpp/mmfs/bin
```

mmcrvdisk command

Creates a vdisk within a declustered array of a GPFS Native RAID recovery group.

Synopsis

```
mmcrvdisk -F StanzaFile [-v{yes | no}]
```

Description

The **mmcrvdisk** command creates one or more vdisks. Upon successful completion of the **mmcrvdisk** command, the vdisk stanza file is rewritten in a form that can be used as input to the **mmcrnsd** command.

The first vdisk that is created in a recovery group must be a log vdisk, which is indicated by a disk usage of **vdiskLog**.

Note: The recovery group must be active to run this command.

Parameters

-F *StanzaFile*

Specifies a file that includes vdisk stanzas identifying the vdisks to be created.

Vdisk stanzas look like the following:

```
%vdisk: vdiskName=VdiskName
        rg=RecoveryGroupName
        da=DeclusteredArrayName
        blocksize=BlockSize
        size=Size
        raidCode=RaidCode
        diskUsage=DiskUsage
        failureGroup=FailureGroup
        pool=StoragePool
        longTermEventLogSize=LongTermEventLogSize
        shortTermEventLogSize=ShortTermEventLogSize
        fastWriteLogPct=fastWriteLogPct
```

where:

vdiskName=*VdiskName*

Specifies the name you wish to assign to the vdisk NSD to be created. This name must not already be used as another GPFS disk name, and it must not begin with the reserved string 'gpfs'.

Note: This name can contain only the following characters:
Uppercase letters 'A' through 'Z'
Lowercase letters 'a' through 'z'
Numerals '0' through '9'
Underscore character '_'

All other characters are not valid.

rg=*RecoveryGroupName*

Specifies the recovery group where the vdisk is to be created.

da=*DeclusteredArrayName*

Specifies the declustered array within the recovery group where the vdisk is to be created.

blocksize=*BlockSize*

Specifies the size of the data blocks for the vdisk. This must match the block size planned for the file system. The valid values are:

For nWayReplicated RAID codes:
256 KiB, 512 KiB, 1 MiB, 2 MiB

For 8+2p and 8+3p RAID codes:
512 KiB, 1 MiB, 2 MiB, 4 MiB, 8 MiB, 16MiB

Specify this value with the character **K** or **M** (for example, **512K**).

If you are using the system pool as metadata only and placing your data in a separate pool, you can specify your metadata-only vdisks with one block size and your data vdisks with a different block size. Since a file system with different metadata and data block sizes requires the use of multiple GPFS storage pools, a file system placement policy is needed to direct user file data to the data storage pool.

size=*Size*
Specifies the size of the vdisk. If **size**=*Size* is omitted, it defaults to using all the available space in the declustered array. The requested vdisk size is equally allocated across all of the pdisks within the declustered array.

raidCode=*RaidCode*
Specifies the raid code to be used for the vdisk. Valid codes are the following:

Unreplicated
Indicates no replication. This should only be used for a log tip backup vdisk.

2WayReplication
Indicates two-way replication. This should only be used for a log tip vdisk.

3WayReplication
Indicates three-way replication.

4WayReplication
Indicates four-way replication.

8+2p
Indicates Reed-Solomon 8 + 2p.

8+3p
Indicates Reed-Solomon 8 + 3p.

diskUsage=*DiskUsage*
Specifies a disk usage or accepts the default. With the exception of the **vdiskLog** value, this field is ignored by the **mmcrvdisk** command and is passed unchanged to the output stanza file.

Possible values are the following:

dataAndMetadata
Indicates that the disk contains both data and metadata. This is the default for disks in the system pool.

dataOnly
Indicates that the disk contains data and does not contain metadata. This is the default for disks in storage pools other than the system pool.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

descOnly
Indicates that the disk contains no data and no file metadata. Such a disk is used solely to keep a copy of the file system descriptor, and can be used as a third failure group in certain disaster recovery configurations. For more information, see the *GPFS: Advanced Administration Guide* and search on *Synchronous mirroring utilizing GPFS replication*.

vdiskLog

Indicates that this is the log home vdisk for the recovery group.

The log home vdisk must be created before any vdisks that are exposed through NSD.

vdiskLogTip

Indicates that this is the log tip vdisk for the recovery group.

vdiskLogTipBackup

Indicates that this is the log tip backup vdisk for the recovery group.

vdiskLogReserved

Indicates a vdisk that should have the same size and raid code as the log home vdisk but is otherwise unused.

This is used to equalize the space consumption in the declustered arrays that do not contain the log home vdisk.

failureGroup=*FailureGroup*

This field is ignored by the **mmcrvdisk** command and is passed unchanged to the output stanza file. It identifies the failure group to which the vdisk NSD belongs.

pool=*StoragePool*

This field is ignored by the **mmcrvdisk** command and is passed unchanged to the output stanza file. It specifies the name of the storage pool to which the vdisk NSD is assigned.

longTermEventLogSize=*LongTermEventLogSize*

Specifies the size of the long-term event log, rounding up to the block size if not aligned. The default value is 4MB.

longTermEventLogSize applies only to the log vdisk.

shortTermEventLogSize=*ShortTermEventLogSize*

Specifies the size of the short-term event log, rounding up to the block size if not aligned. The default value is 1MB.

shortTermEventLogSize applies only to the log vdisk.

fastWriteLogPct=*FastWriteLogPct*

Specifies the fraction of the remaining log space to be used for the fast-write log, after allocating space for the event logs. The default value is 90%.

fastWriteLogPct applies only to the log vdisk.

-v {yes | no }

Verification flag; a value of **yes** specifies that vdisk creation will only start if the buffer space requirements of other recovery groups can be verified. The default value is **yes**.

Exit status

0 Successful completion.

nonzero

A failure has occurred.

Security

You must have root authority to run the **mmcrvdisk** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *GPFS: Administration and Programming Reference* topic:

- Requirements for administering a GPFS file system (not included in this *Documentation Update*)

Examples

Assume that input stanza file 000DE37TOP.vdisk contains the following lines:

```
%vdisk: vdiskName=000DE37TOPLOG
        rg=000DE37TOP
        da=LOG
        blocksize=1m
        size=4g
        raidCode=3WayReplication
        diskUsage=vdiskLog
        longTermEventLogSize=4M
        shortTermEventLogSize=1M
        fastWriteLogPct=90
%vdisk: vdiskName=000DE37TOPDA1META
        rg=000DE37TOP
        da=DA1
        blocksize=1m
        size=250g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=37
        pool=system
%vdisk: vdiskName=000DE37TOPDA1DATA
        rg=000DE37TOP
        da=DA1
        blocksize=8m
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=37
        pool=data
[...]
```

The following command example shows how to create the vdisks described in the stanza file 000DE37TOP.vdisk:

```
mmcrvdisk -F 000DE37TOP.vdisk
```

The system displays output similar to the following:

```
mmcrvdisk: [I] Processing vdisk 000DE37TOPLOG
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA1META
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA1DATA
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA2META
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA2DATA
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA3META
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA3DATA
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA4META
mmcrvdisk: [I] Processing vdisk 000DE37TOPDA4DATA
mmcrvdisk: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

See also

See also the following *GPFS: Administration and Programming Reference* topics:

- "mmcrrecoverygroup command" on page 84
- "mmdelvdisk command" on page 91
- mmlsvdisk command (not included in this *Documentation Update*)

Location

```
/usr/lpp/mmfs/bin
```

mmdelvdisk command

Deletes vdisks from a declustered array in a GPFS Native RAID recovery group.

Synopsis

```
mmdelvdisk {"VdiskName[;VdiskName...]" | -F StanzaFile} [-p | --recovery-group RecoveryGroupName]
```

Description

The **mmdelvdisk** command is used to delete vdisks from the declustered arrays of recovery groups. There must be no NSD defined on the specified vdisk; if necessary, use the **mmdelnsd** command prior to running this command.

The log home vdisk in a recovery group cannot be deleted until all the other vdisks in the recovery group have been deleted. Log tip and log tip backup vdisks may be created and deleted at any time, even while work is active on those or other vdisks.

Parameters

VdiskName[;*VdiskName*...]
Specifies the vdisks to be deleted.

-F *StanzaFile*
Specifies the name of a stanza file in which stanzas of the type **%vdisk** identify the vdisks to be deleted. Only the vdisk name is required to be included in the vdisk stanza; however, for a complete description of vdisk stanzas, see the following *GPFS: Administration and Programming Reference* topic: "mmcrvdisk command" on page 87.

-p Indicates that the recovery group is permanently damaged, and therefore the vdisk information should be removed from the GPFS cluster configuration data. This option can be used when the GPFS daemon is down on the recovery group servers.

--recovery-group *RecoveryGroupName*
Specifies the name of the recovery group that contains the vdisks. This option is for the rare case where the vdisks have been removed from the GPFS cluster configuration data but are still present in the recovery group.

You can see the vdisks in the recovery group by issuing either of the following commands:

```
mmlsvdisk --recovery-group RecoveryGroupName
```

or

```
mmlsrecoverygroup RecoveryGroupName -L
```

Exit status

0 Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmdelvdisk** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *GPFS: Administration and Programming Reference* topic:

- Requirements for administering a GPFS file system (not included in this *Documentation Update*)

Examples

The following command example shows how to delete vdisk `000DE37BOTDA4DATA`:

```
mmdelvdisk 000DE37BOTDA4DATA
```

The system displays output similar to the following:

```
mmdelvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

See also

See also the following *GPFS: Administration and Programming Reference* topics:

- "mmcrvdisk command" on page 87
- mmdelpdisk command (not included in this *Documentation Update*)
- mmdelrecoverygroup command (not included in this *Documentation Update*)
- mmlsvdisk command (not included in this *Documentation Update*)

Location

`/usr/lpp/mmfs/bin`

mmlsrecoverygroup command

Lists information about GPFS Native RAID recovery groups.

Synopsis

mmlsrecoverygroup [*RecoveryGroupName* [**-L** [**--pdisk**]]]

Description

The **mmlsrecoverygroup** command lists information about recovery groups. The command displays various levels of information, depending on the parameters specified.

Parameters

RecoveryGroupName
Specifies the recovery group for which the information is being requested. If no other parameters are specified, the command displays only the information that can be found in the GPFS cluster configuration data.

-L Displays more detailed runtime information for the specified recovery group.

--pdisk
Indicates that pdisk information is to be listed for the specified recovery group.

Exit status

0 Successful completion.

nonzero
A failure has occurred.

Security

You must have root authority to run the **mmlsrecoverygroup** command.

The node on which the command is issued must be able to execute remote shell commands on any other node in the cluster without the use of a password and without producing any extraneous messages. For additional details, see the following *GPFS: Administration and Programming Reference* topic:

- Requirements for administering a GPFS file system (not included in this *Documentation Update*)

Examples

1. The following command example shows how to list all the recovery groups in the GPFS cluster:

```
mmlsrecoverygroup
```

The system displays output similar to the following:

```
                    declustered
                    arrays with
 recovery group       vdisks     vdisks  servers
 ------------------  -----------  ------  -------
 000DE37BOT                    5       9  c07ap01.pok.ibm.com,c08ap01.pok.ibm.com
 000DE37TOP                    5       9  c08ap01.pok.ibm.com,c07ap01.pok.ibm.com
```

2. The following command example shows how to list the basic non-runtime information for recovery group 000DE37BOT:

```
mmlsrecoverygroup 000DE37BOT
```

The system displays output similar to the following:

```
                    declustered
                    arrays with
 recovery group       vdisks     vdisks  servers
 ------------------  -----------  ------  -------
 000DE37BOT                    5       9  c08ap01.pok.ibm.com,c07ap01.pok.ibm.com
```

```
declustered array
    with vdisks     vdisks
------------------  ------
DA1                      2
DA2                      2
DA3                      2
DA4                      2
LOG                      1

                                        declustered
vdisk               RAID code              array     remarks
------------------  ------------------  -----------  -------
000DE37BOTDA1DATA   8+3p                DA1
000DE37BOTDA1META   4WayReplication     DA1
000DE37BOTDA2DATA   8+3p                DA2
000DE37BOTDA2META   4WayReplication     DA2
000DE37BOTDA3DATA   8+3p                DA3
000DE37BOTDA3META   4WayReplication     DA3
000DE37BOTDA4DATA   8+3p                DA4
000DE37BOTDA4META   4WayReplication     DA4
000DE37BOTLOG       3WayReplication     LOG                 log
```

3. The following command example shows how to display the runtime status of recovery group BB1RGL:

   ```
   mmlsrecoverygroup BB1RGL -L
   ```

   The system displays output similar to the following:

   ```
   BB1RGL -L

   The system displays output similar to the following:

                      declustered
    recovery group       arrays     vdisks  pdisks  format version
    -----------------  -----------  ------  ------  --------------
    BB1RGL                       4       8     119  3.5.0.13

    declustered   needs                             replace               scrub      background activity
       array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
    -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
    DA1          no            3      58       2          2      50 TiB   14 days  scrub       52%  low
    DA2          yes           3      58       2          2      50 TiB   14 days  scrub       52%  low
    NVR          no            1       2       0          1    3648 MiB   14 days  scrub       33%  low
    SSD          no            1       1       0          1     186 GiB   14 days  scrub       25%  low

                                            declustered                               checksum
    vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
    ------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------
    ltip_BB1RGL         2WayReplication     NVR              48 MiB       2 MiB       512       ok    logTip
    ltbackup_BB1RGL     Unreplicated        SSD              48 MiB       2 MiB       512       ok    logTipBackup
    lhome_BB1RGL        4WayReplication     DA1              20 GiB       2 MiB       512       ok    log
    reserved1_BB1RGL    4WayReplication     DA2              20 GiB       2 MiB       512       ok    logRsvd
    BB1RGLMETA1         4WayReplication     DA1             750 GiB       1 MiB     32 KiB     ok
    BB1RGLDATA1         8+3p                DA1              35 TiB      16 MiB     32 KiB     ok
    BB1RGLMETA2         4WayReplication     DA2             750 GiB       1 MiB     32 KiB     ok
    BB1RGLDATA2         8+3p                DA2              35 TiB      16 MiB     32 KiB     ok

    active recovery group server                     servers
    -----------------------------------------------  -------
    c45f01n01-ib0.gpfs.net                           c45f01n01-ib0.gpfs.net,c45f01n02-ib0.gpfs.net
   ```

4. The following example shows how to include pdisk information for BB1RGL:

   ```
   mmlsrecoverygroup BB1RGL -L --pdisk
   ```

   The system displays output similar to the following:

   ```
                     declustered
    recovery group       arrays     vdisks  pdisks  format version
    -----------------  -----------  ------  ------  --------------
    BB1RGL                       4       8     119  3.5.0.13

    declustered   needs                             replace               scrub      background activity
       array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
    -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
    DA1          no            3      58       2          2      50 TiB   14 days  scrub       52%  low
    DA2          yes           3      58       2          2      50 TiB   14 days  scrub       52%  low
   ```

```
NVR          no            1       2       0          1    3648 MiB   14 days  scrub       33%  low
SSD          no            1       1       0          1     186 GiB   14 days  scrub       25%  low

                    number of    declustered                user
pdisk              active paths     array     free space  condition state
-----------------  ------------  -----------  ----------  --------- -----
e1d1s01                 2        DA1             955 GiB  Normal    ok
e1d1s02                 2        DA1             955 GiB  Normal    ok
e1d1s03ltbackup         2        SSD             186 GiB  Normal    ok
e1d1s04                 2        DA2             939 GiB  Normal    ok
.                       .        .               .        .         .
.                       .        .               .        .         .
.                       .        .               .        .         .
e4d5s06                 0        DA2            1862 GiB  Replaceable dead/noPath/systemDrain/noRGD/noVCD/noData/replace
nv00log                 1        NVR            1824 MiB  Normal    ok
nv01log                 1        NVR            1824 MiB  Normal    ok

                                        declustered                           checksum
vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------
ltip_BB1RGL         2WayReplication     NVR              48 MiB      2 MiB       512      ok    logTip
ltbackup_BB1RGL     Unreplicated        SSD              48 MiB      2 MiB       512      ok    logTipBackup
lhome_BB1RGL        4WayReplication     DA1              20 GiB      2 MiB       512      ok    log
reserved1_BB1RGL    4WayReplication     DA2              20 GiB      2 MiB       512      ok    logRsvd
BB1RGLMETA1         4WayReplication     DA1             750 GiB      1 MiB     32 KiB     ok
BB1RGLDATA1         8+3p                DA1              35 TiB     16 MiB     32 KiB     ok
BB1RGLMETA2         4WayReplication     DA2             750 GiB      1 MiB     32 KiB     ok
BB1RGLDATA2         8+3p                DA2              35 TiB     16 MiB     32 KiB     ok

active recovery group server                     servers
-----------------------------------------------  -------
c45f01n01-ib0.gpfs.net                           c45f01n01-ib0.gpfs.net,c45f01n02-ib0.gpfs.net
```

See also

See also the following *GPFS: Administration and Programming Reference* topics:

- "mmchrecoverygroup command" on page 82
- "mmcrrecoverygroup command" on page 84
- mmdelrecoverygroup command (not included in this *Documentation Update*)
- mmlspdisk command (not included in this *Documentation Update*)
- mmlsrecoverygroupevents command (not included in this *Documentation Update*)
- mmlsvdisk command (not included in this *Documentation Update*)

Location

/usr/lpp/mmfs/bin

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road,
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment or a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Intel is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Product Number: 5765-G66
5765-XA3
5641-A07
5724-N94

Printed in USA