Redbooks

ibm.com/redbooks

# IBM Spectrum Scale (GPFS) for Linux on z Systems

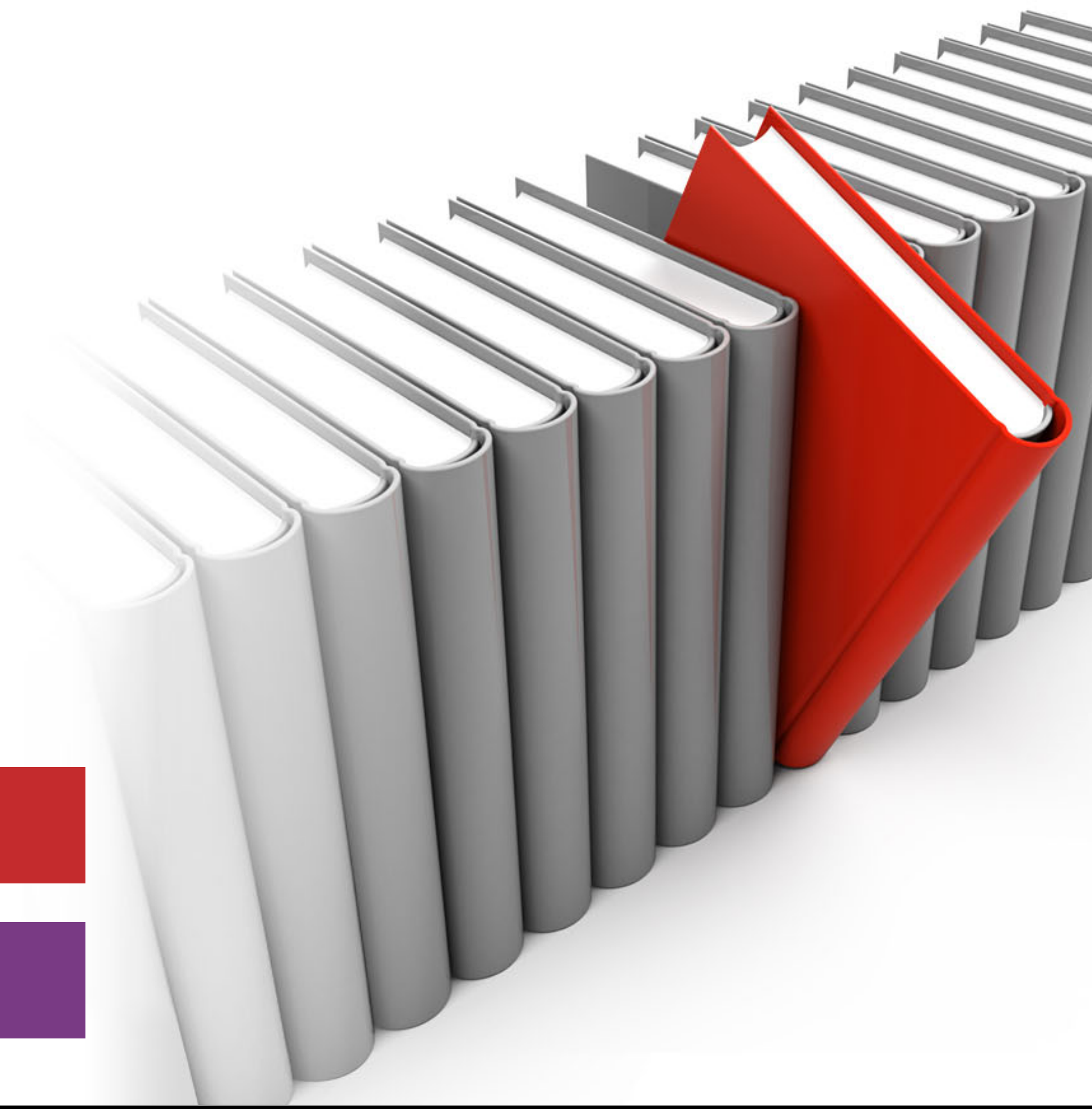

Octavian Lascu

Randy Brewster

Rich Conway

Andrei Socoliuc

Andrei Vlad

Cloud

z Systems

IBM

Redpaper

International Technical Support Organization

# IBM Spectrum Scale (GPFS) for Linux on z Systems

February 2016

REDP-5308-00

**Note:** Before using this information and the product it supports, read the information in “Notices” on page v.

**First Edition (February 2016)**

This edition applies to:

- IBM z13, driver level 22
- z/VM Version 6 Release 3.0, Service Level 1501 (64-bit)
- SUSE Linux Enterprise Server 12 (for IBM z Systems and x86_64)
- IBM Spectrum Scale 4.2 for Linux on z Systems
- IBM Spectrum Protect Server 7.1.4
- IBM Spectrum Protect Client for Linux on z Systems 7.1.4

# Contents



# Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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, MD-NC119, Armonk, NY 10504-1785, US*

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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 actual people or business enterprises 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.

# Trademarks

THIS PAGE INTENTIONALLY LEFT BLANK

# Preface

This IBM® Redpaper™ publication describes IBM Spectrum Scale™ for Linux on z Systems™.

This paper helps you install and configure IBM Spectrum Scale (formerly GPFS™) in a disaster recovery configuration. Scenario testing is described for various events: Site failure, storage failure, node failure. Recovery procedures from each tested scenario are provided.

This paper also provides an installation and configuration scenario for saving data stored in a Spectrum Scale file system by using IBM Spectrum Protect™ integration features. Multi-node backup usage is described.

## Authors

This paper was produced by a team of specialists from around the world working at the IBM International Technical Support Organization (ITSO), Poughkeepsie Center.

**Octavian Lascu** is a Senior IT Consultant for IBM Romania with over 25 years of experience. He specializes in designing, implementing, and supporting complex IT infrastructure environments (systems, storage, and networking), including high availability and disaster recovery solutions and high-performance computing deployments. He has developed materials for and taught over 50 workshops for technical audiences around the world. He has written several IBM Redbooks and IBM Redpaper™ publications.

**Randy Brewster** is a Software Test Specialist at IBM Poughkeepsie Center, New York. He has 18 years of experience in IBM pSeries® systems and clustering software. Before his work on pSeries systems, he spent over 10 years on IBM zSeries® systems. He has worked at IBM for over 33 years. His areas of expertise include clustering software, Spectrum Scale (formerly GPFS), PowerHA (formerly HACMP), IBM BlueGene®, and WebSphere MQ. He has written extensively on GPFS and IBM DB2®. He holds a Master's degree in Computer Science from the City University of New York.

**Rich Conway** is a senior IT specialist working for IBM Development Support Team (DST), Poughkeepsie Center. He has over 30 years of experience in all aspects of IBM MVS™ and z/OS as a systems programmer. He has worked extensively with IBM UNIX® System Services and WebSphere on z/OS. His areas of expertise also include AIX and cross-platform virtualization. He was a project leader for the ITSO for many Redbooks publications and has also provided technical advice and support for numerous Redbooks publications over the past 20 years.

**Andrei Socoliuc** is a Certified IT Specialist in Systems and Infrastructure, working in IBM Global Technology Services Romania. He has over 15 years of experience in IT infrastructure. Andrei holds a Master's degree in Computer Science from the Polytechnics University of Bucharest. He is a Certified Advanced Technical Expert for IBM Power Systems and a Certified Tivoli Storage Manager specialist. He is also a coauthor of several IBM Redbooks publications.

**Andrei Vlad** is a Client Technical Architect working for IBM Romania, Global Technology Services department since 2002. His areas of expertise include High Performance Computing and clustering, GPFS, CSM, xCAT, and virtualization on various platforms. He has implemented several HPC clusters and provided worldwide customer training. Currently, he is developing the Infrastructure as a Service and Cloud platforms in Romania. He is certified in AIX and Linux, and has a Master's degree in Electronic Engineering from Polytechnics University in Bucharest, Romania.

Thanks to the following people for their contributions to this project:

Ella Buslovich, Robert Haimowitz, Lydia Parziale, William G White
IBM International Technical Support Organization, Poughkeepsie Center

Steven Bermann
IBM Pittsburgh

Stefan Bender
IBM Germany

Susanne Wintenberger, Martin Kammerer
IBM Germany

Marian Gasparovic
IBM Slovakia

# Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

**ibm.com**/redbooks/residencies.html

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks® publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  **ibm.com**/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  https://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

# Overview and scenarios

IBM Spectrum Scale provides an outstanding highly available clustering solution, adding value to the strengths of Linux on IBM z Systems™ environments. Linux on z Systems can provide significant advantages over other Linux platforms, helping you control costs and providing high levels of quality of service.

Spectrum Scale, based on IBM GPFS technology, is a high performance shared-disk file management solution that provides fast, reliable access to data from multiple nodes in a cluster environment.

Spectrum Scale also allows data sharing in a mixed platform environment, which can provide benefits in cloud or analytics environments by eliminating the need of transferring data across platforms.

IBM Spectrum Scale for Linux on z Systems V4.2 is now available in three editions:

- Express
- Standard
- Advanced

IBM Spectrum Scale V4.2 provides the following enhancements for Linux on z Systems:

- Support of IBM Spectrum Protect (Tivoli Storage Manager) backup-archive and Space Management client

  IBM Spectrum Scale V4.2 Standard Edition extends support of backup and restore functions to protect data in your file system and also enables space management of your data.

- Asynchronous multisite disaster recovery

  IBM Spectrum Scale V4.2 Advanced Edition supports asynchronous disaster recovery (DR) at the fileset level. This enables you to create a primary (active)/secondary (passive) relationship at the fileset level.

For more information, see the IBM Spectrum Scale web page:

http://public.dhe.ibm.com/common/ssi/ecm/zs/en/zss03118usen/ZSS03118USEN.PDF

This chapter describes various configurations of Spectrum Scale cluster on Linux on z Systems. We provide considerations for achieving high availability and failure resiliency using Spectrum Scale and common infrastructure tools. The following topics are described in this chapter:

- Overview
- Single Central Processor Complex deployment
- Cross site cluster: Synchronous mirroring

> **Important:** We used the following products for the configurations:
>
> - Two IBM z13™ 2964-N63, driver level 22
> - IBM z/VM® Version 6 Release 3.0, Service Level 1501 (64-bit)
> - SUSE Linux Enterprise Server 12 (for IBM z Systems and x86_64)
> - IBM Spectrum Scale 4.2 for Linux on z Systems
> - IBM Spectrum Protect Server 7.1.4
> - IBM Spectrum Protect Client for Linux on z Systems 7.1.4
> - IBM DS8870 Storage subsystem for extended count key data (ECKD) storage
> - IBM Storwize V7000 for FCP attached storage

## 1.1 Overview

Depending on the intended use, Spectrum Scale cluster configurations can be implemented with several variations. From shared data access to high availability, high performance file system access configurations, following are the common infrastructure elements that must be considered when designing the cluster:

- Number and type of nodes
- Type of storage
- Communication infrastructure (LAN, WAN, and SAN)
- Backup and archiving

Planning, installing, and configuring Spectrum Scale on Linux on IBM z Systems is not fundamentally different from deployments on other platforms, aside from preparing the platform (there are specific actions for Linux on z Systems).

In this material, we focus on deploying a configuration suitable for certain disaster recovery (DR) environments: Spectrum Scale synchronous mirroring utilizing GPFS replication (for Linux on z Systems).

> **Tip:** GPFS replication does not require additional software or storage subsystem licensed features (for synchronous data replication).

Later in this chapter, we describe the most common methods for providing infrastructure availability (network and storage) in Spectrum Scale configurations for Linux on z Systems.

## Nodes (cluster members) and cluster topology

The number of cluster members and their roles are defined based on the following:

- **Applications requirements:** A cluster member can run an application on top of Spectrum Scale file system. File system access is provided through the file system device driver (kernel extension). The file system device driver accesses storage through the Network Shared Disk (NSD) layer (block device abstraction layer) provided by Spectrum Scale. Depending on the application I/O requirements, access to the NSD can be done either through the local disk device driver (direct-attached disk) or through the network (NSD via network). NSD access is provided to the nodes that do not have direct-attached storage via the cluster daemon network from the NSD server nodes (NSD server nodes have direct-attached disk).
- **Heterogeneous cluster**: Spectrum Scale can run on diverse platforms. This represents an attractive solution in multiplatform environments, where cross-platform data sharing is needed, simplifying the storage architecture and application deployment.
- **Redundancy and availability:** Spectrum Scale has built-in mechanisms for providing redundant disk access (NSD failover), cluster, and file system survivability (node quorum and file system quorum). Synchronous or asynchronous[1] replication can be used to protect data.

> **Note:** More information about node types and roles can be found in the manual *IBM Spectrum Scale V4.2: Concepts, Planning, and Installation Guide*, GA76-0441-03.

## Cluster storage

There are two ways in which Spectrum Scale for Linux on z Systems can access raw storage:

- FICON/IBM ECKD™ characteristics:
  - 1:1 mapping host subchannel:dasd
  - Serialization of I/Os per subchannel
  - I/O request queue in Linux
  - Disk blocks are 4 KB
  - High availability by IBM FICON® path groups
  - Load balancing by FICON path groups and parallel access volumes
- Fibre Channel Protocol (FCP)/Small Computer System Interface (SCSI) characteristics:
  - Several I/Os can be issued against a logical unit number (LUN) immediately
  - Queuing in the FICON Express card or in the storage server
  - Additional I/O request queue in Linux
  - Disk blocks are 512 bytes
  - High availability by Linux multipathing, type failover, or multibus
  - Load balancing by Linux multipathing, type multibus

For details, see the following web page:

http://public.dhe.ibm.com/software/dw/linux390/perf/ECKD_versus_SCSI.pdf

## Connectivity (internode)

There are multiple solutions for Spectrum Scale internode communication:

- Open Systems Adapter (OSA) direct attach: Shared or dedicated (virtualized) OSA channel-path identifier (CHPID) type OSD attached to the Linux guest.

[1] Active File Management (AFM): See the following web page:
http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/com.ibm.spectrum.scale.v4r2.adv.doc/bl1adv_afm.htm?lang=en

- Linux guest using virtual OSA through a virtual switch: The virtual switch is connected to an OSA (dedicated or shared) CHPID.
- IBM HiperSockets™: Is a function that emulates the Logical Link Control (LLC) layer of an OSA-Express QDIO interface (in memory). It is implemented in z Systems firmware.

### Backup and archiving

Spectrum Scale implements single namespace across multiple nodes (all nodes access the same data). In order to back up the data, multiple nodes can be used to send data in parallel to an IBM Spectrum Protect server (formerly IBM Tivoli Storage Manager).

## 1.2 Single Central Processor Complex deployment

In this section, we introduce some scenarios that can be used to deploy Spectrum Scale cluster inside the same IBM z Systems Central Processor Complex (CPC). Spectrum Scale can be deployed on the following nodes:

- Nodes running Linux in a logical partition (LPAR)
- Nodes running as Linux guests under a z Systems hypervisor such as z/VM

For availability purposes, nodes running as Linux guests can be deployed in two or more z/VM LPARs.

Storage for NSD servers can be deployed by using FICON/ECKD or FCP/SCSI. See “Cluster storage” on page 3. For high availability, multiple NSD servers (minimum two) for each NSD should be configured.

Internode network connectivity can be deployed as OSA direct, virtual OSA, or HiperSockets, as described in “Connectivity (internode)” on page 3. A combination of techniques is also possible, but not described in this document.

External network availability can be achieved with the following configurations:

- Virtual OSA connected to the virtual switch in z/VM. The virtual switch uplink (external connection) can be implemented with active/standby OSA configuration.
- If OSA direct attach is used, it is recommended to configure at least two OSA CHPIDs to each node configured for high availability using Linux Ethernet Bonding[2] Driver (not covered in this document).

[2] See https://www.kernel.org/doc/Documentation/networking/bonding.txt

## 1.2.1 Sample scenario

The scenario that is depicted in Figure 1-1 shows a deployment that provides high availability and continuous operation even if software maintenance is required for a z/VM LPAR or for Linux nodes. However, this scenario does not cover all possible situations in terms of internode connectivity or cluster topologies.

### HiperSockets replication

If all cluster nodes are inside the same z Systems CPC, GPFS replication can be done by using a HiperSockets network. The tiebreaker node can be reached via standard Ethernet (OSA) connectivity.

Although a HiperSockets network provides very high bandwidth and low latency, you should consider HiperSockets carefully and assess CPU consumption before committing into production.

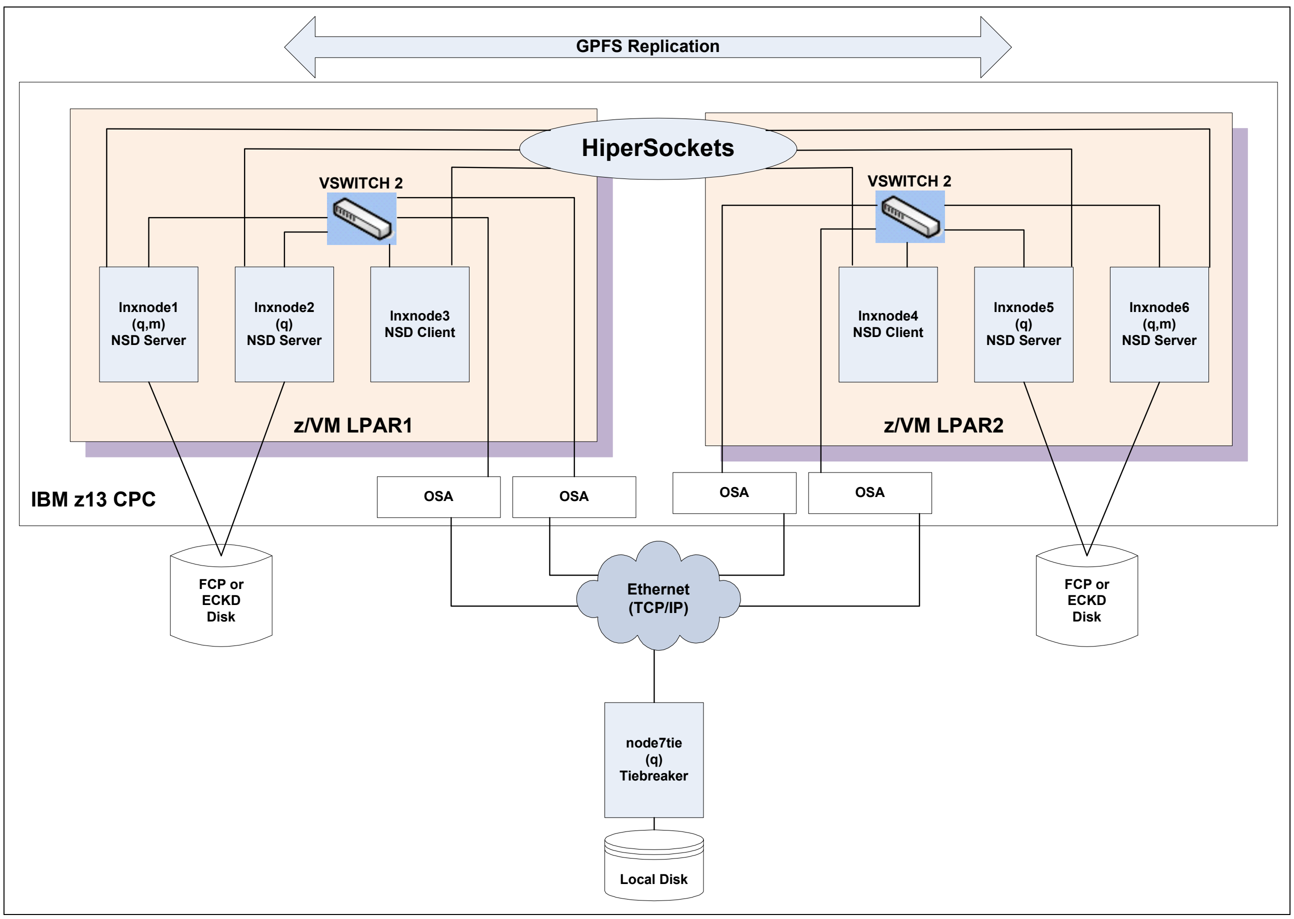



*Figure 1-1 Typical deployment for high availability (single CPC)*

## 1.3 Cross site cluster: Synchronous mirroring

In this section, we describe a configuration using GPFS synchronous replication, with a single GPFS cluster defined over three geographically distinct sites: Two production sites and a third tiebreaker site. One or more file systems are created, mounted, and accessed concurrently from the two production sites.

### 1.3.1 Scenario overview

This paper describes a disaster recovery solution using IBM Spectrum Scale with the following characteristics:

- Three geographically distributed locations: Site A, site B, and site C.
- A single Spectrum Scale cluster is defined across all nodes in the three sites.
- Nodes in site A and site B are for production.
- Site A, site B, and site C are connected using the Internet Protocol network.
- Site C provides quorum functions (for cluster and file systems). In case one production site fails, the surviving site together with site C can achieve cluster and file systems' quorum:
  - Site C can be a different architecture (IBM Power Systems or Intel™ based system running a supported OS)
  - Site C is to provide protection for a split-brain situation, where the connection between site A and site B is lost, but site C remains connected to either site A or site B.
  - Site C consists of a single server (can be physical or virtual) with a TCP/IP connection to both sites (A and B). It also requires one disk (or LUN) for each file system created on Spectrum Scale.
- Site A and site B replicate data using Spectrum Scale mechanisms over a TCP/IP connection. Nodes in both production sites access and update the data on the file system (this solution is considered an active-active DR solution). The replication used in this scenario is *synchronous*. For performance reasons, a fast and reliable network connection between site A and site B is recommended (1 Gbps or higher, depending on application requirements). For site C, a 100 Mbps or faster network connection is recommended (assuming site C is used only for quorum).
- If one of the production sites become unavailable, operation continues in the surviving site without operator intervention. When the unavailable site is recovered, manual intervention is required to restore proper data replication and disk access. In Chapter 2, "Implementing Spectrum Scale synchronous mirroring using GPFS replication on Linux on z Systems" on page 11, we describe the recovery procedures that we performed in our test environment.

## 1.3.2 Internode connectivity using z/VM virtual switch

Figure 1-2 presents a scenario with the following characteristics (in addition to the characteristics listed in 1.3.1, “Scenario overview” on page 6:

- Network connectivity high availability is implemented using z/VM virtual switch uplink redundancy (active/passive).
- It is also possible to use z/VM Link Aggregation by grouping two or more dedicated OSA ports together to form a logical “fat” pipe between a virtual switch and an external switch. z/VM configures all the OSA features in Exclusive Mode to disable its MIF port sharing capabilities. Other configuration types to increase bandwidth and availability are supported. For information, see the following site:

  http://www.vm.ibm.com/virtualnetwork/linkag.html
- The storage can be either FCP or ECKD. Disk access redundancy is implemented by using specific mechanisms (see “Cluster storage” on page 3).
- The replication is done by using the TCP/IP network. Proper sizing is required, as demanded by application.

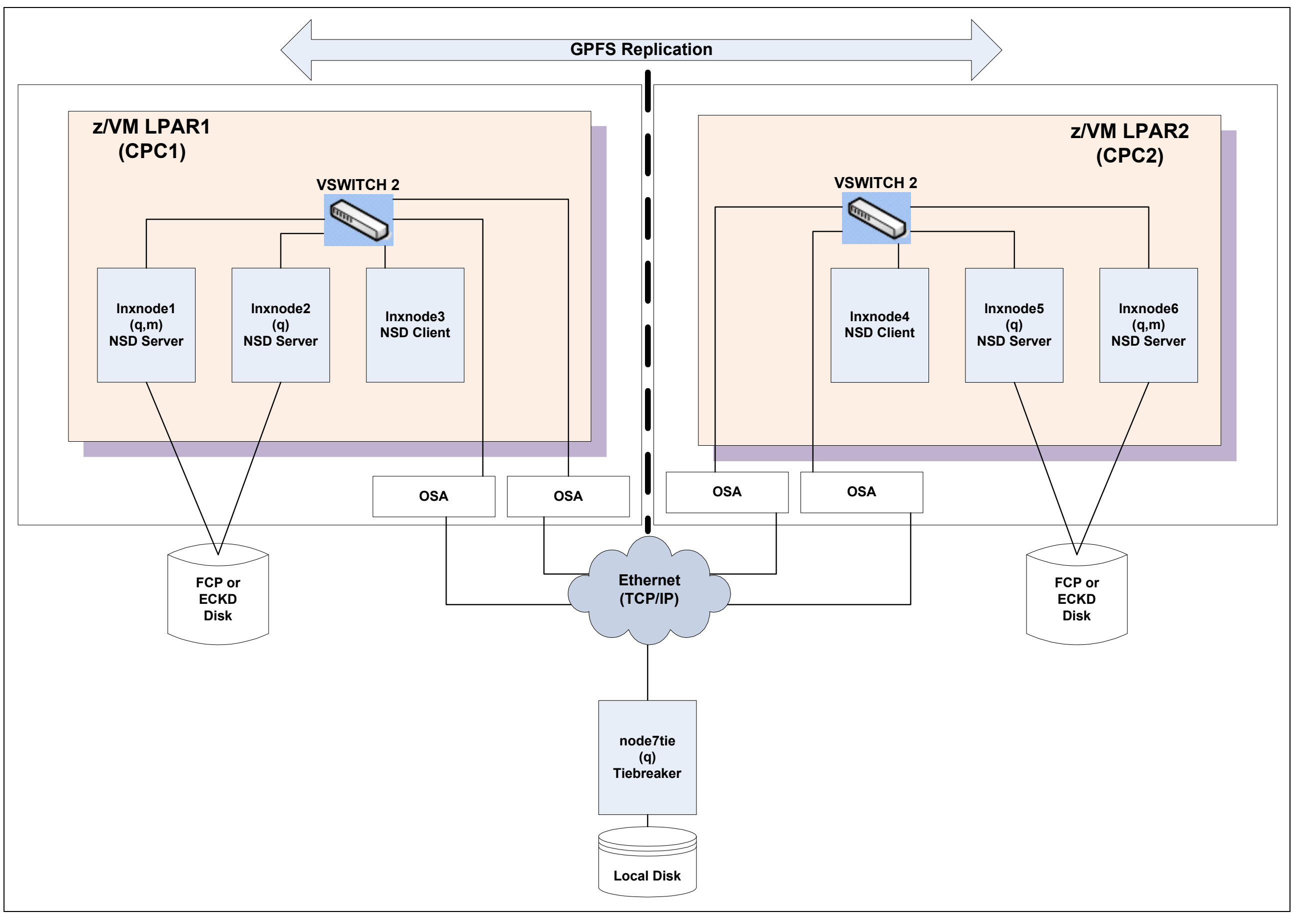



*Figure 1-2 Typical scenario using z/VM virtual switch*

## 1.3.3 Direct attached OSA

Figure 1-3 presents a scenario with the following characteristics (in addition to the characteristics listed in 1.3.1, "Scenario overview" on page 6):

- Internode connectivity high availability is implemented by using dual OSA CHPIDs defined to each Linux node and aggregated at the Linux level using Linux Ethernet Bonding Driver.
- OSAs can be shared or dedicated.
- The storage can be either FCP or ECKD. Disk access redundancy is implemented by using specific mechanisms (see "Cluster storage" on page 3).
- The replication is done by using the Internet Protocol network. Proper sizing is required, as demanded by application needs.

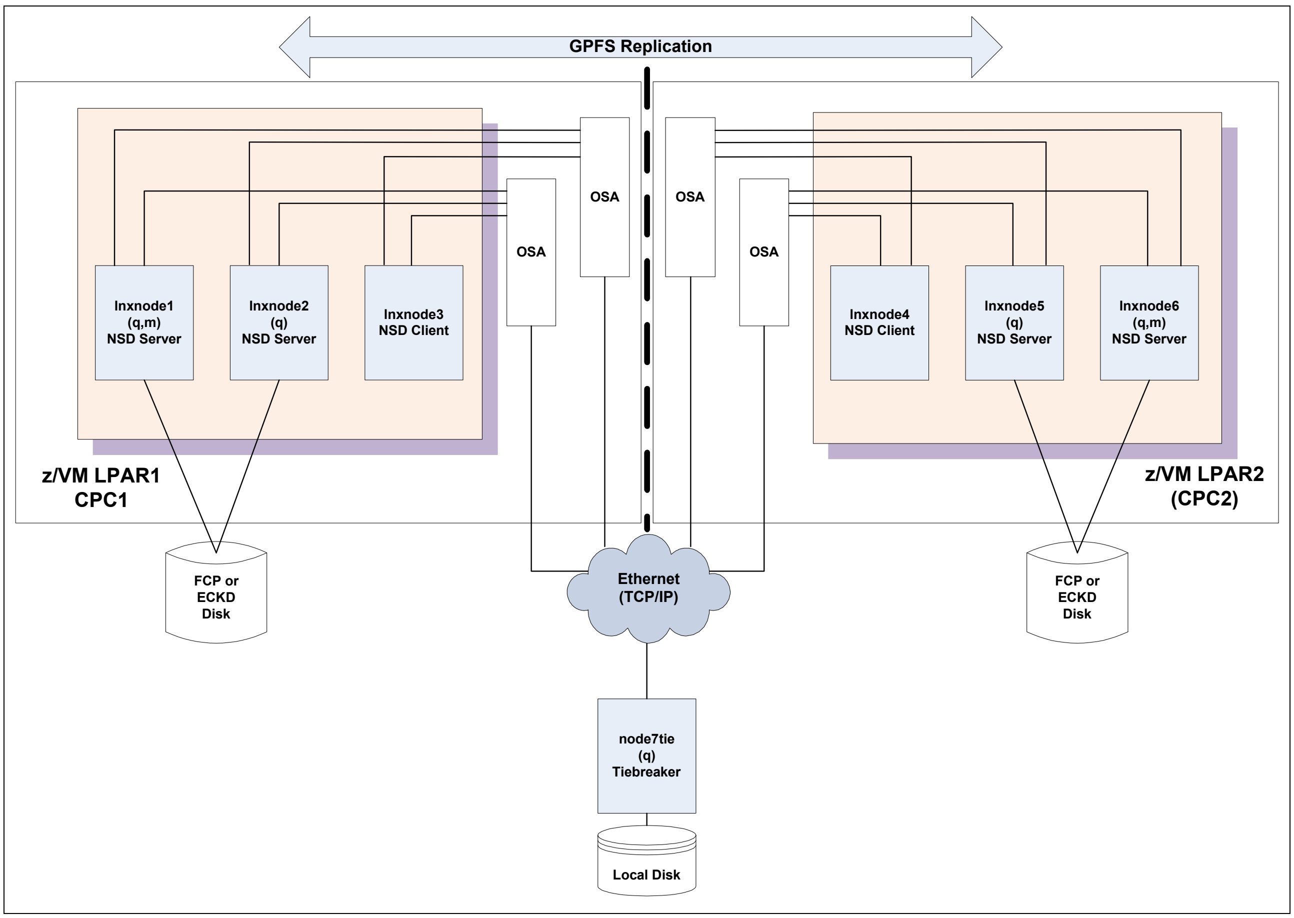



*Figure 1-3 Direct attached OSA: No z/VM switch*

## 1.4 Conclusion

There are multiple ways to exploit hardware configurations that can be used for deploying Spectrum Scale on IBM z Systems. When designing the solution, you should always consider the optimal solution for your environment. Starting from the application and drilling down to the platform is the correct approach. As a reminder, when deploying Spectrum Scale on IBM z Systems, some of the following aspects should be considered:

- Multiple OSAs can be used. Redundancy and performance can be managed either at the hypervisor (z/VM) or at Linux level.
- For synchronous replication, the cluster should be tested with actual (not simulated) cross-site connectivity. For distance between sites, check the latest frequently asked questions at:

  http://www.ibm.com/support/knowledgecenter/api/content/nl/en-us/STXKQY_4.2.0/gpfsclustersfaq.html
- Multiple networks can be used:
  - Data sharing
  - Management (for security reasons)
  - Client access
- Storage access
  - Disk access type (FCP/SCSI or FICON/ECKD)
  - Performance (striping, load balancing, multipathing)
  - Availability

2

# Implementing Spectrum Scale synchronous mirroring using GPFS replication on Linux on z Systems

In this chapter, we describe IBM Spectrum Scale for Linux on z Systems cluster configuration for two scenarios based on z Systems types of storage access (FICON and FCP). Both scenarios describe a Spectrum Scale active-active cluster using (native) synchronous replication across two sites[1] (site A, site B), with a tiebreaker configuration in a third site (site C). We use Spectrum Scale version 4.2.0.

The following topics are covered in this chapter:

- Platform preparation
- Scenario 1: Storage presented as directly attached SCSI disks (SCSI over FCP protocol)
- Scenario 2: Storage presented as extended count key data devices

1 For configuration details and supported replication distances, see:
https://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/com.ibm.spectrum.scale.v4r2.adv.doc/bl1adv_gpfsrep.htm
Also, see the most current frequently asked questions list at:
https://www.ibm.com/support/knowledgecenter/api/content/nl/en-us/STXKQY_4.2.0/gpfsclustersfaq.html

## 2.1 Platform preparation

The deployment consists of a seven-node cluster spanning across two production sites (active-active configuration) using native Spectrum Scale (GPFS) synchronous replication, with a tiebreaker site.

> **Cluster planning:** Cluster network should be designed considering the following elements that may affect performance:
>
> - Network bandwidth
> - Distance between sites (A <-> B <->C)

A diagram of our configuration is shown in Figure 2-1.

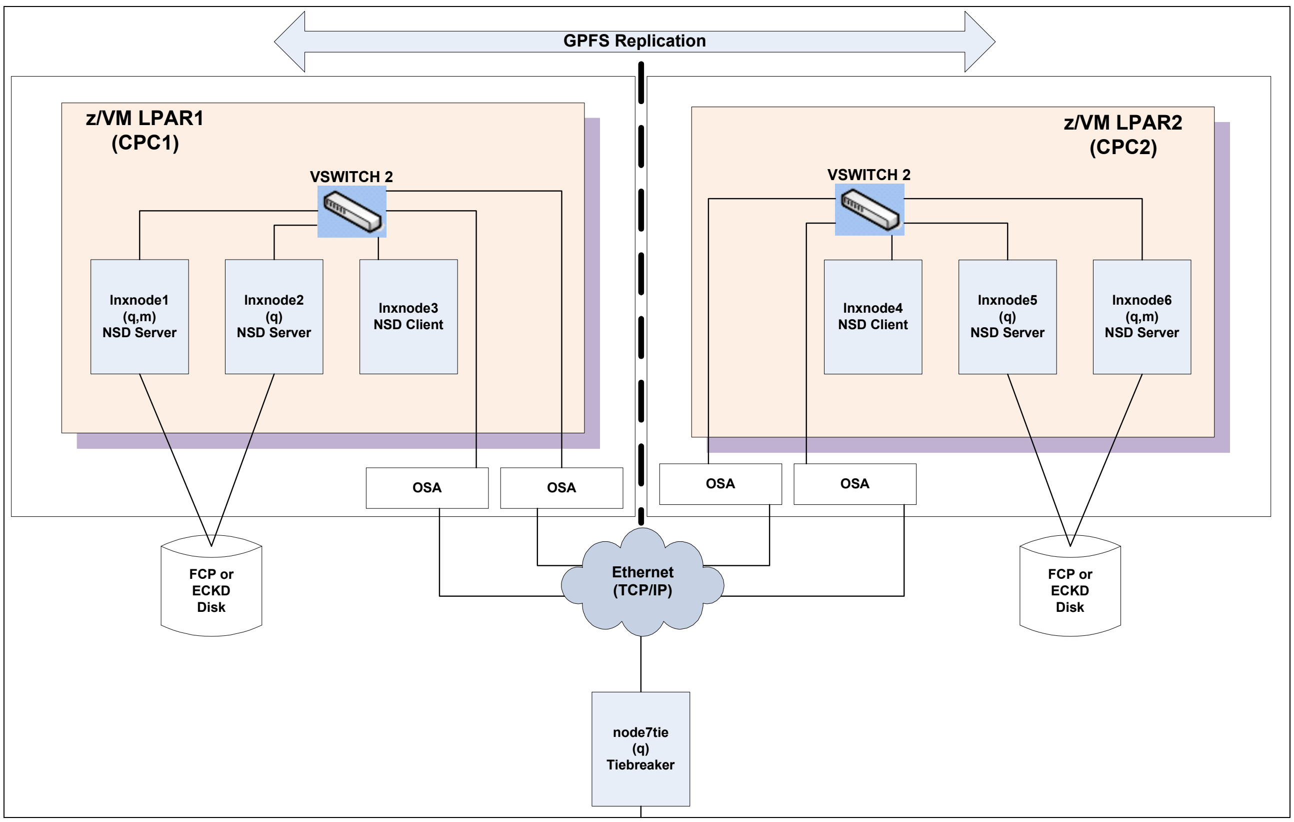



*Figure 2-1 Cluster diagram*

### *Cluster components*

The Spectrum Scale cluster elements used in our configuration are briefly introduced in this paragraph.

- Seven nodes, in the following setup:
  - Three nodes (lnxnode1, lnxnode2, lnxnode3) on primary location (site A), in an IBM z Systems CPC (IBM z13), where two nodes (lnxnode1, lnxnode2) access Small Computer System Interface (SCSI) disks via Fibre Channel Protocol (FCP), while the third node (lnxnode3) is a Network Shared Disk (NSD) client (multiple NSD servers for each NSD can be configured. See the **mmcrnsd** command **man** page).

– Three nodes in secondary location (site B) in a different z Systems CPC (IBM z13), where two have nodes (lnxnode5, lnxnode6) access SCSI disks via FCP, while the third node (lnxnode4) is an NSD client.

– One x86 - 64-bit architecture node (node7tie) in a separate (third) location (site C), used for the tiebreaker role.

- Four data disks (extended count key data (ECKD) or SCSI) for each of site A and site B (eight data disks in total).

  The disks in each site are in separate storage subsystems (one storage in each site). There is no storage replication mechanism or other replication software involved in this configuration (other than Spectrum Scale replication). It is highly recommended to use disks with similar performance characteristics in both sites.

- Redundant network configuration (redundancy provided by z Systems and IBM z/VM configuration).

## 2.1.1 Setting up the operating system

Installation and configuration of the two IBM z Systems, as well as z/VM installation are out of the scope of this material[2]. In our setup, we use the following configuration (high-level overview):

- Two IBM z13 CPCs (hardware) with connectivity to LAN (Open Systems Adapter (OSA)) and storage (ECKD, SCSI)
- Two logical partitions (LPARs), one in each z13 running z/VM 6.3
- Two DS8700 storage subsystems (providing both ECKD and SCSI devices)
- Linux guests are running SUSE Linux Enterprise Server 12 with the latest update available at the time of writing. For supported kernel levels (Spectrum Scale V4.2.0 requirements), see GPFS FAQs:

  https://www.ibm.com/support/knowledgecenter/api/content/nl/en-us/STXKQY_4.2.0/gpfsclustersfaq.html

- Guest OS (Linux) configuration:

  – Firewall service disabled. If you require a firewall, consult the Spectrum Scale documentation:

    https://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/ibmspectrumscale42_welcome.html

  – AppArmor service disabled

  – Network (recommended dedicated) for Spectrum Scale cluster. We use a flat Layer 2 network configuration, available on all sites (A, B, C). Additional networks can be configured (not described in this document), depending on your requirements (client access, backup, and so on)

  – High availability through multiple paths to access the storage

  – Secure Shell (OpenSSH) for cluster management

[2] See z/VM V6.3 documentation at:
http://www.ibm.com/support/knowledgecenter/SSB27U_6.3.0/com.ibm.zvm.v630/zvminfoc03.htm
and IBM Redbooks:
http://www.redbooks.ibm.com

**Note:** Network access and storage configuration details for Linux on z Systems are presented in 3.3.1, “Network devices (interfaces)” on page 78 and 3.3.2, “Disk storage” on page 83.

– The `/etc/hosts` file is the same on all nodes and is shown in Example 2-1

*Example 2-1 /etc/host file*

```
9.12.7.28       node7tie
# GPFS nodes on z
9.12.7.20       lnxnode1
9.12.7.21       lnxnode2
9.12.7.22       lnxnode3
9.12.7.24       lnxnode4
9.12.7.25       lnxnode5
9.12.7.26       lnxnode6
```

**Notes:** If each node is connected using multiple network interfaces, use the IP label[a] of the interface used for Spectrum Scale cluster management as the node name in the Spectrum Scale cluster.

Ensure that all nodes that will be part of the cluster resolve the names identically (either short or fully qualified domain name).

a. IP label: A name associated with an IP address (regardless of the name resolution method)

- Set up the boot parameter for SUSE Linux Enterprise Server 12 for every node (Linux on z Systems) as follows:
  a. Edit file `/etc/default/grub` and add the **`vmalloc=4096G`** parameter at the end of the line as shown in Example 2-2.

*Example 2-2 vmalloc parameter*

```
GRUB_CMDLINE_LINUX_DEFAULT="hvc_iucv=8 TERM=dumb osamedium=eth instmode=ftp x
crashkernel=206M-:103M cio_ignore=all,!ipldev,!condev vmalloc=4096G
```

  b. Run the "**`grub-mkconfig -o /boot/here grub2/grub.cfg`**" command.
  c. Reboot the node.

Also, see the IBM Knowledge Center, “For Linux on z Systems: Changing the kernel settings”:

https://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/com.ibm.spectrum.scale.v4r2.ins.doc/bl1ins_zlinux_kernel.htm

## 2.1.2 Internode communication for Spectrum Scale cluster

Internode communication for cluster configuration management and control requires a method of executing actions on all nodes from one of the cluster nodes. This section presents the remote command execution setup for our cluster.

**Important:** We use a Spectrum Scale cluster configuration with *adminMode*[a]*=allToAll* (shown by running the **`mmlsconfig`** command). All commands must be run as “root” user.

a. https://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/ibmspectrumscale42_welcome.html

1. We set Secure Shell (SSH) remote login without password prompt. We run the **ssh-keygen** command on every node in order to generate public and private keys. In Example 2-3, we show the commands for one node.

**SSH remote command:** There are other methods to configure remote login without password prompt, like using a single key for the entire cluster. However, these are outside the scope of this document.

*Example 2-3 ssh-keygen command*

```
lnxnode1:~ # ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
74:e7:4e:43:8b:38:86:bb:88:92:c2:56:41:ea:20:17 [MD5] root@lnxnode1
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|  E.             |
|  o.    . . o    |
|o...   o o = .   |
|+.  . . S . =    |
| . .   o . o .   |
|...   .     .    |
|+o . . .         |
|o.. . .          |
+--[MD5]----------+

lnxnode1:~ # ls -l ~/.ssh/
total 12
-rw------- 1 root root 1675 Nov 12 15:07 id_rsa
-rw-r--r-- 1 root root  395 Nov 12 15:07 id_rsa.pub
-rw-r--r-- 1 root root 1890 Nov 12 14:51 known_hosts
```

2. Create the authorization file, which includes the public keys of all Spectrum Scale cluster nodes, as shown in Example 2-4. The method of harvesting the public keys is the administrator's decision.

*Example 2-4 authorized_keys file*

```
lnxnode1:~ # cat ~/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC9QCoyNOj432WrUfYuOl8XWjAWQviTLDq2TfxXxUBrJXSnkyz
ZPX9CFI1KcTI5GTtCWFdSw3s32ILIMkuQdNSHobWE7uUeVBeYFqm23DgqONcWyDwGgU+OkFT22Pae4Y
yhTTPEJxZT7+UVPJYA+cwYvrvVv8PRdaXXuvOG9zxO5dQQWge1XT+bjyNPBXx1WR+cPWGMBDn+X685T
h7IEUKUkzS+G4B2oVkYDKHjngoteTLuZh4DFhz5YzrAROK8kXbxbcc3pKTDINmzGWVVO+9HoXjSgtCN
Yl41qRZLUdsCiAE25Sx2PKG9l3rO1EF8VtiijXxnXckdxgUbekZxSImz root@lnxnode1
ssh-rsa

...<< Snippet >>...

AAAAB3NzaC1yc2EAAAADAQABAAABAQDWDCfi4MBCY82mIGOcdx5bOuOyYkcbPX6dO+y94I2qmmhk5cV
VNaXa4jHZ9NrpJZcHDAltLzUiJOUh36pOy8xvpWpaEDLlmMXltQVUJaBrf753Jfov4FTm4q7yjyoWRi
fADmHaRBupuGj45JXKfsWZC+L4IaMzha4EEBWkZzlXWyG95Oo+R7wqDLdofh2uHTnlkg7e5EiUmQYDN
ye28FjLVqsYGekCPGh/IbgQjVWb7P9T6KzQE16i+GORd6bTqeSf2ST4SToL86z/eOaGuinrhnUj3YqO
uTGqphzOCO2GT5OKkRSrW4YcuR625+jRFMlPfk3eFAkwQOe6getaurK9 root@node7tie
```

3. Configure 0600 file permissions for the `authorized_keys` file by running:

   ```
   chmod 0600 ~/.ssh/authorized_keys
   ```

4. Distribute the files in the `~/.ssh` directory to every Spectrum Scale node. If the number of nodes is large, a script can help to distribute the files faster and consistently. We use the following commands:

   ```
   cd ~/.ssh/; scp authorized_keys  <nodename>:~/.ssh/
   ```

   where <nodename> is the host name for each node.

5. Create the `~/.ssh/known_hosts` file to include public keys of the ssh server on all nodes and distribute it on all cluster nodes. This helps avoid the confirmation required when connecting to a node. Our `known_hosts` file is shown in Example 2-5.

*Example 2-5 known_hosts file*

```
lnxnode1:~/.ssh # cat /root/.ssh/known_hosts
lnxnode1,9.12.7.20 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBGlBmRODmywIMlQ/d8v66Ut2Z4S
jO54ApABNyvZWg8TK1zfw3Suxf6wjwnpp6YVuvXXjROO2IDBapckOd9E/rFI=

...<< Snippet >>...

node7tie,9.12.7.28 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBuWFEVf8+fMpaag6x2GqOMjTQW
DT7OmtXPYiiSH+Ca5jAq4Bla2AO8S27vdcelpsDp+GQdy7xsj/2k7fhXz8Dk=
```

The `known_hosts` file can be distributed in the same way as the `authorized_keys` file (see Step 4).

6. Test remote connection to *each node* by running a remote command as shown in example Example 2-6 on page 17.

*Example 2-6 Test remote ssh*

```
lnxnode1:~/.ssh # ssh lnxnode1 date
Thu Nov 12 15:36:17 EST 2015
lnxnode1:~/.ssh # ssh lnxnode2 date
Thu Nov 12 15:36:20 EST 2015
lnxnode1:~/.ssh # ssh lnxnode3 date
Thu Nov 12 15:36:24 EST 2015
lnxnode1:~/.ssh # ssh lnxnode4 date
Thu Nov 12 15:36:30 EST 2015
lnxnode1:~/.ssh # ssh lnxnode5 date
Thu Nov 12 15:36:33 EST 2015
lnxnode1:~/.ssh # ssh date
Thu Nov 12 15:36:38 EST 2015
lnxnode1:~/.ssh # ssh node7tie date
Thu Nov 12 15:36:45 EST 2015
```

7. Set up time configuration (NTP client) on each cluster node. It is important that applications using files stored in a Spectrum Scale (shared) file system have a synchronized clock on all nodes. One of the cluster members can act as a time server for all other nodes.

   Setting up time synchronization across nodes helps cluster management by keeping all messages in log files in the correct time sequence.

### 2.1.3 Configuring the Spectrum Scale cluster

After the operating system is configured, we install the Spectrum Scale packages and prepare the platform for cluster configuration. Follow these steps:

1. The Spectrum Scale version used at the time of writing is 4.2.0.0. The code can be installed by using multiple methods, like zypper. However, we use RPM direct method, as shown in Example 2-7. Note that the tiebreaker node (located in Site C, x86 64 bit architecture) uses x86_64 packages.

   *Example 2-7 Spectrum Scale packages installation*

```
lnxnode6:/redp5308/GPFS-4.2.0.0 # rpm -ivh gpfs.base-4.2.0-0.s390x.rpm
gpfs.adv-4.2.0-0.s390x.rpm gpfs.crypto-4.2.0-0.s390x.rpm
gpfs.docs-4.2.0-0.noarch.rpm gpfs.ext-4.2.0-0.s390x.rpm
gpfs.gpl-4.2.0-0.noarch.rpm gpfs.gskit-8.0.50-47.s390x.rpm
gpfs.libsrc-4.2.0-0.noarch.rpm gpfs.msg.en_US-4.2.0-0.noarch.rpm
gpfs.src-4.2.0-0.noarch.rpm
Preparing...                          ################################# [100%]
Updating / installing...
   1:gpfs.base-4.2.0-0                ################################# [ 10%]
   2:gpfs.ext-4.2.0-0                 ################################# [ 20%]
   3:gpfs.adv-4.2.0-0                 ################################# [ 30%]
   4:gpfs.crypto-4.2.0-0              ################################# [ 40%]
   5:gpfs.gpl-4.2.0-0                 ################################# [ 50%]
   6:gpfs.src-4.2.0-0                 ################################# [ 60%]
   7:gpfs.msg.en_US-4.2.0-0           ################################# [ 70%]
   8:gpfs.libsrc-4.2.0-0              ################################# [ 80%]
   9:gpfs.gskit-8.0.50-47             ################################# [ 90%]
  10:gpfs.docs-4.2.0-0                ################################# [100%]
```

2. We compile the portability layer as shown in Example 2-8. The **--build-package** parameter creates an `rpm` file that contains the modules in binary format.

> **Tip:** For nodes with the same architecture and OS version, there is no need to recompile the portability layer on every node. You can distribute the resulted `rpm` package to the remainder nodes with the same architecture and kernel version, and install it directly.

*Example 2-8 Building the portability layer*

```
lnxnode6:~/ # /usr/lpp/mmfs/bin/mmbuildgpl --build-package -v
--------------------------------------------------------
mmbuildgpl: Building GPL module begins at Mon Nov 16 14:42:27 EST 2015.
--------------------------------------------------------
Verifying Kernel Header...
  kernel version = 31228004 (3.12.28-4-default, 3.12.28-4)
  module include dir = /lib/modules/3.12.28-4-default/build/include
  module build dir   = /lib/modules/3.12.28-4-default/build
  kernel source dir  = /usr/src/linux-3.12.28-4/include
  Found valid kernel header file under
/lib/modules/3.12.28-4-default/build/include
Verifying Compiler...
  make is present at /usr/bin/make
  cpp is present at /usr/bin/cpp
  gcc is present at /usr/bin/gcc
  g++ is present at /usr/bin/g++
  ld is present at /usr/bin/ld
Verifying rpmbuild...
Verifying Additional System Headers...
  Verifying linux-glibc-devel is installed ...
    Command: /bin/rpm -q linux-glibc-devel
    The required package linux-glibc-devel is installed
make World ...
make InstallImages ...
make rpm ...
Wrote:
/usr/src/packages/RPMS/s390x/gpfs.gplbin-3.12.28-4-default-4.2.0-0.s390x.rpm
--------------------------------------------------------
mmbuildgpl: Building GPL module completed successfully at Mon Nov 16 14:42:38
EST 2015.
```

3. Add the Spectrum Scale binaries path to the system PATH variable by editing the `/etc/profile` file and adding the following line:

   ```
   export PATH=$PATH:/usr/lpp/mmfs/bin
   ```

4. Define the Spectrum Scale cluster. We define the cluster nodes' type and roles in a text file (node descriptor file) as shown in Example 2-9 on page 19, with the following settings:

   – One manager node per site:

     This allows the election of a file system manager regardless which site is operational. (the manager node chooses a file system manager out of the quorum nodes).

   – Two quorum nodes per each site A and site B, and one quorum node on site C:

     This allows for quorum validation in case nodes one site (A or B) are not available (three out of five quorum nodes will be available).

*Example 2-9 Node descriptor file*

```
lnxnode1:/work # cat FCP-nodes
lnxnode1:manager-quorum
lnxnode2:quorum
lnxnode3:client
lnxnode4:client
lnxnode5:quorum
lnxnode6:manager-quorum
node7tie:quorum
```

5. Create the Spectrum Scale cluster by running the following command:

   `mmcrcluster -N FCP-nodes  -r /usr/bin/ssh -R /usr/bin/scp  -C FCP-Cluster`

   The result is shown in Example 2-10. Note the *default cluster configuration repository (CCR)* option. All quorum nodes maintain a copy of the cluster repository data.

*Example 2-10 Cluster configuration*

```
lnxnode1:~ # mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         FCP-Cluster.lnxnode1
  GPFS cluster id:           12427567213354573007
  GPFS UID domain:           FCP-Cluster.lnxnode1
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address  Admin node name  Designation
-------------------------------------------------------------------
   1   lnxnode1          9.12.7.20   lnxnode1         quorum-manager
   2   lnxnode2          9.12.7.21   lnxnode2         quorum
   3   lnxnode5          9.12.7.25   lnxnode5         quorum
   4   lnxnode6          9.12.7.26   lnxnode6         quorum-manager
   5   node7tie          9.12.7.28   node7tie         quorum
   6   lnxnode3          9.12.7.22   lnxnode3
   8   lnxnode4          9.12.7.24   lnxnode4
```

6. We start the Spectrum Scale cluster by running **`mmstartup -a`** (on any node).

   To check the status of the cluster, run **`mmgetstate -a`**, as shown in Example 2-11.

*Example 2-11 Cluster status*

```
lnxnode1:~ # mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      lnxnode1         active
       2      lnxnode2         active
       3      lnxnode5         active
       4      lnxnode6         active
       5      node7tie         active
       6      lnxnode3         active
       8      lnxnode4         active
```

### Spectrum Scale cluster review

We created the Spectrum Scale cluster in the following configuration:

- Three nodes in site A
- Three nodes in site B
- One node in site C (tie breaker)

At this time, the cluster is ready for file system configuration.

In the following sections, we test two types of storage configuration and file system replication (synchronous) between sites A and B:

- One file system using FCP disks (disks using SCSI protocol): Described in 2.2, “Scenario 1: Storage presented as directly attached SCSI disks (SCSI over FCP protocol)” on page 20.
- A separate file system using ECKD DASD: Described in 2.3, “Scenario 2: Storage presented as extended count key data devices” on page 41.

For each type of storage configuration, we perform the following tests:

1. Primary location becomes offline. We check if the cluster remains active in site B and that the file system is still accessible (through nodes in site B) without manual intervention. This test simulates a disaster in site A.
2. When site A is recovered and ready to rejoin the cluster, we perform the steps required to synchronize the data from site B to site A. This simulates a full recovery in site A. The steps are described in “Site A recovery” on page 33.

## 2.2 Scenario 1: Storage presented as directly attached SCSI disks (SCSI over FCP protocol)

In the first part of this section, we present the steps to configure the file system into an existing cluster. We use SCSI disks for storing data and metadata.

In the second part of this section, we perform testing by simulating a site failure and document the recovery steps.

### 2.2.1 File system configuration

This section presents the configuration steps for creating a cluster file system on provided storage (FCP/SCSI).

#### Network Shared Disk configuration

This section presents the steps used to configure the Network Shared Disk (NSD) in our cluster:

1. Set up a multipath configuration. Enable the service, if not already done, by using the **`yast`** or **`systemctl`** commands.

   By default, the SCSI disks are shown as worldwide ID (WWID), which is a long number and difficult to use. We define an alias for each disk. On SUSE Linux Enterprise Server 12 the `/etc/multipath.conf` file is not created by default. We create one as shown in Example 2-12 on page 21 and distribute to site A NSD server nodes (in our case, lnxnode1 and lnxnode2).

> **Attention:** The `multipath.conf` file shown in Example 2-12 is *only* for the NSD server nodes in site A. For the nodes in site B, the WWIDs are different because the LUNs are from a different storage. Thus, the `multipath.conf` file for nodes *lnxnode5* and *lnxnode6* is different from nodes in site A.

*Example 2-12 multipath.conf for NSD server nodes in site A*

```
lnxnode1:~ # cat /etc/multipath.conf
multipaths {
   multipath {
      wwid    360050768018305e120000000000000f4
      alias   fcp-disk-a
   }
   multipath {
      wwid    360050768018305e120000000000000f5
      alias   fcp-disk-b
   }
   multipath {
      wwid    360050768018305e120000000000000f6
      alias   fcp-disk-c
   }
   multipath {
      wwid    360050768018305e120000000000000f7
      alias   fcp-disk-d
   }
}
```

To activate the configuration run the following commands:

```
# systemctl stop multipathd.service to stop the multipathd service
# multipath -F to flush the tables
# systemctl start multipathd.service to start the multipathd service
```

After restarting the *multipathd* service, the disk alias names are visible and devices are created in the */dev/mapper* path, as shown in Example 2-13.

*Example 2-13 Multipath configuration details*

```
lnxnode2:~ # ll /dev/mapper/
total 0
crw------- 1 root root 10, 236 Nov 13 15:46 control
lrwxrwxrwx 1 root root       7 Nov 16 17:16 fcp-disk-a -> ../dm-0
lrwxrwxrwx 1 root root       7 Nov 16 17:16 fcp-disk-b -> ../dm-1
lrwxrwxrwx 1 root root       7 Nov 16 17:16 fcp-disk-c -> ../dm-2
lrwxrwxrwx 1 root root       7 Nov 16 17:16 fcp-disk-d -> ../dm-3

lnxnode2:~ # multipath -l
fcp-disk-d (360050768018305e120000000000000f7) dm-3 IBM,2145
size=30G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| |- 0:0:1:3 sdh  8:112  active undef running
| |- 0:0:3:3 sdp  8:240  active undef running
| |- 1:0:2:3 sdab 65:176 active undef running
| |- 1:0:3:3 sdaf 65:240 active undef running
| |- 2:0:2:3 sdar 66:176 active undef running
| |- 2:0:3:3 sdav 66:240 active undef running
```

```
| |- 3:0:2:3 sdbh 67:176 active undef running
| `- 3:0:3:3 sdbl 67:240 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 0:0:0:3 sdd  8:48   active undef running
  |- 0:0:2:3 sdl  8:176  active undef running
  |- 1:0:0:3 sdt  65:48  active undef running
  |- 1:0:1:3 sdx  65:112 active undef running
  |- 2:0:0:3 sdaj 66:48  active undef running
  |- 2:0:1:3 sdan 66:112 active undef running
  |- 3:0:0:3 sdaz 67:48  active undef running
  `- 3:0:1:3 sdbd 67:112 active undef running
fcp-disk-c (360050768018305e120000000000000f6) dm-2 IBM,2145
size=30G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| |- 0:0:0:2 sdc  8:32   active undef running
| |- 0:0:2:2 sdk  8:160  active undef running
| |- 1:0:0:2 sds  65:32  active undef running
| |- 1:0:1:2 sdw  65:96  active undef running
| |- 2:0:0:2 sdai 66:32  active undef running
| |- 2:0:1:2 sdam 66:96  active undef running
| |- 3:0:0:2 sday 67:32  active undef running
| `- 3:0:1:2 sdbc 67:96  active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 0:0:1:2 sdg  8:96   active undef running
  |- 0:0:3:2 sdo  8:224  active undef running
  |- 1:0:2:2 sdaa 65:160 active undef running
  |- 1:0:3:2 sdae 65:224 active undef running
  |- 2:0:2:2 sdaq 66:160 active undef running
  |- 2:0:3:2 sdau 66:224 active undef running
  |- 3:0:2:2 sdbg 67:160 active undef running
  `- 3:0:3:2 sdbk 67:224 active undef running
fcp-disk-b (360050768018305e120000000000000f5) dm-1 IBM,2145
size=30G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| |- 0:0:1:1 sdf  8:80   active undef running
| |- 0:0:3:1 sdn  8:208  active undef running
| |- 1:0:2:1 sdz  65:144 active undef running
| |- 1:0:3:1 sdad 65:208 active undef running
| |- 2:0:2:1 sdap 66:144 active undef running
| |- 2:0:3:1 sdat 66:208 active undef running
| |- 3:0:2:1 sdbf 67:144 active undef running
| `- 3:0:3:1 sdbj 67:208 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 0:0:0:1 sdb  8:16   active undef running
  |- 0:0:2:1 sdj  8:144  active undef running
  |- 1:0:0:1 sdr  65:16  active undef running
  |- 1:0:1:1 sdv  65:80  active undef running
  |- 2:0:0:1 sdah 66:16  active undef running
  |- 2:0:1:1 sdal 66:80  active undef running
  |- 3:0:0:1 sdax 67:16  active undef running
  `- 3:0:1:1 sdbb 67:80  active undef running
fcp-disk-a (360050768018305e120000000000000f4) dm-0 IBM,2145
size=30G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| |- 0:0:0:0 sda  8:0    active undef running
```

```
| |- 0:0:2:0 sdi  8:128  active undef running
| |- 1:0:1:0 sdu  65:64  active undef running
| |- 2:0:0:0 sdag 66:0   active undef running
| |- 2:0:1:0 sdak 66:64  active undef running
| |- 3:0:0:0 sdaw 67:0   active undef running
| `- 3:0:1:0 sdba 67:64  active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 0:0:1:0 sde  8:64   active undef running
  |- 0:0:3:0 sdm  8:192  active undef running
  |- 1:0:2:0 sdy  65:128 active undef running
  |- 1:0:3:0 sdac 65:192 active undef running
  |- 2:0:2:0 sdao 66:128 active undef running
  |- 2:0:3:0 sdas 66:192 active undef running
  |- 3:0:2:0 sdbe 67:128 active undef running
  `- 3:0:3:0 sdbi 67:192 active undef running
```

**Note:** This step has to executed also on the NSD server nodes in site B.

2. We create the Network Shared Disk stanza file, shown in Example 2-14, with the following characteristics:
   – All disks in site A and site B are holding data and metadata.
   – All disks in site A are defined as NSD failure group 1.
   – All disks in site B are defined as NSD failure group 2.
   – The disk in site C is defined as NSD failure group 3.
   – The disk in site C holds only a copy of the file system descriptor. No data or metadata is stored on the disk in site C (file system descriptor quorum).

*Example 2-14 NSD definition file*

```
lnxnode1:~ # cat /work/FCP-Disks
 %nsd:
 nsd=site_A_disk_1
 usage=dataAndMetadata
 failureGroup=1
 servers=lnxnode1,lnxnode2
 device=/dev/mapper/fcp-disk-a
 %nsd:
 nsd=site_A_disk_2
 usage=dataAndMetadata
 failureGroup=1
 servers=lnxnode2,lnxnode1
 device=/dev/mapper/fcp-disk-b
 %nsd:
 nsd=site_A_disk_3
 usage=dataAndMetadata
 failureGroup=1
 servers=lnxnode1,lnxnode2
 device=/dev/mapper/fcp-disk-c
 %nsd:
 nsd=site_A_disk_4
 usage=dataAndMetadata
 failureGroup=1
 servers=lnxnode2,lnxnode1
 device=/dev/mapper/fcp-disk-d
 %nsd:
```

```
nsd=site_B_disk_1
usage=dataAndMetadata
failureGroup=2
servers=lnxnode5,lnxnode6
device=/dev/mapper/fcp-disk-a
%nsd:
nsd=site_B_disk_2
usage=dataAndMetadata
failureGroup=2
servers=lnxnode6,lnxnode5
device=/dev/mapper/fcp-disk-b
%nsd:
nsd=site_B_disk_3
usage=dataAndMetadata
failureGroup=2
servers=lnxnode5,lnxnode6
device=/dev/mapper/fcp-disk-c
%nsd:
nsd=site_B_disk_4
usage=dataAndMetadata
failureGroup=2
servers=lnxnode6,lnxnode5
device=/dev/mapper/fcp-disk-d
%nsd:
nsd=site_C_disk_1
usage=descOnly
failureGroup=3
servers=node7tie
device=sdb
```

3. We create the NSDs by running the **mmcrnsd** command, as shown in Example 2-15.

*Example 2-15 Creating NSDs*

```
lnxnode1:/work # mmcrnsd -F FCP-Disks
mmcrnsd: Processing disk mapper/fcp-disk-a
mmcrnsd: Processing disk mapper/fcp-disk-b
mmcrnsd: Processing disk mapper/fcp-disk-c
mmcrnsd: Processing disk mapper/fcp-disk-d
mmcrnsd: Processing disk mapper/fcp-disk-a
mmcrnsd: Processing disk mapper/fcp-disk-b
mmcrnsd: Processing disk mapper/fcp-disk-c
mmcrnsd: Processing disk mapper/fcp-disk-d
mmcrnsd: Processing disk sdb
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

4. List the NSD disks as shown in Example 2-16. We alternated the primary and secondary NSD servers in rolling configuration so the NSD disks are balanced over the NSD servers (to balance I/O across NSD server nodes).

*Example 2-16 List the NSD disks*

```
lnxnode1:/work # mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
```

```
(free disk)   site_A_disk_1 lnxnode1,lnxnode2
(free disk)   site_A_disk_2 lnxnode2,lnxnode1
(free disk)   site_A_disk_3 lnxnode1,lnxnode2
(free disk)   site_A_disk_4 lnxnode2,lnxnode1
(free disk)   site_B_disk_1 lnxnode5,lnxnode6
(free disk)   site_B_disk_2 lnxnode6,lnxnode5
(free disk)   site_B_disk_3 lnxnode5,lnxnode6
(free disk)   site_B_disk_4 lnxnode6,lnxnode5
(free disk)   site_C_disk_1 node7tie
```

### File system definition

This section presents the steps that follow NSD configuration. Once NSDs are available to the cluster, file systems can be defined as follows:

5. We create a file system as shown in Example 2-17 with the following settings:
   - Use all NSDs defined in the previous step.
   - Automount the file system when the cluster is started.
   - Specify two maximum and default number of replicas for both data and metadata. These configurations ensure that any file created in this file system is replicated by default.

*Example 2-17 Create the file system*

```
lnxnode1:~ # mmcrfs fcp_fs1 -F /work/FCP-Disks  -A yes -m 2 -M 2 -n 10 -r 2
-R 2 -T /fcp_fs1

The following disks of fcp_fs1 will be formatted on node lnxnode6:
 site_A_disk_1: size 30720 MB
 site_A_disk_2: size 30720 MB
 site_A_disk_3: size 30720 MB
 site_A_disk_4: size 30720 MB
 site_B_disk_1: size 30720 MB
 site_B_disk_2: size 30720 MB
 site_B_disk_3: size 30720 MB
 site_B_disk_4: size 30720 MB
 site_C_disk_1: size 152627 MB
Formatting file system ...
Disks up to size 1.4 TB can be added to storage pool system.
Creating Inode File
  81 % complete on Tue Nov 17 15:35:25 2015
 100 % complete on Tue Nov 17 15:35:26 2015
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Completed creation of file system /dev/fcp_fs1.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

6. We mount the file system:

```
mmmount /fcp_fs1 -a
```

7. List the file system disk parameters as shown in Example 2-18. Note that the disk at site C does not hold any data or metadata.

*Example 2-18 File system disks*

```
lnxnode1:~ # mmlsdisk fcp_fs1
disk         driver   sector     failure holds    holds                              storage
name         type       size       group metadata data  status        availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
site_A_disk_1 nsd        512           1 Yes      Yes   ready         up           system
site_A_disk_2 nsd        512           1 Yes      Yes   ready         up           system
site_A_disk_3 nsd        512           1 Yes      Yes   ready         up           system
site_A_disk_4 nsd        512           1 Yes      Yes   ready         up           system
site_B_disk_1 nsd        512           2 Yes      Yes   ready         up           system
site_B_disk_2 nsd        512           2 Yes      Yes   ready         up           system
site_B_disk_3 nsd        512           2 Yes      Yes   ready         up           system
site_B_disk_4 nsd        512           2 Yes      Yes   ready         up           system
site_C_disk_1 nsd        512           3 No       No    ready         up           system
```

8. To check the active file system descriptors' distribution (one per failure group), we run the command shown in Example 2-19.

*Example 2-19 mmlsdisk (long format)*

```
lnxnode1:~ # mmlsdisk fcp_fs1 -L
disk         driver   sector     failure holds    holds                                    storage
name         type       size       group metadata data  status        availability disk id pool         remarks
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------- ------------ ---------
site_A_disk_1 nsd        512           1 Yes      Yes   ready         up                 1 system       desc
site_A_disk_2 nsd        512           1 Yes      Yes   ready         up                 2 system
site_A_disk_3 nsd        512           1 Yes      Yes   ready         up                 3 system
site_A_disk_4 nsd        512           1 Yes      Yes   ready         up                 4 system
site_B_disk_1 nsd        512           2 Yes      Yes   ready         up                 5 system       desc
site_B_disk_2 nsd        512           2 Yes      Yes   ready         up                 6 system
site_B_disk_3 nsd        512           2 Yes      Yes   ready         up                 7 system
site_B_disk_4 nsd        512           2 Yes      Yes   ready         up                 8 system
site_C_disk_1 nsd        512           3 No       No    ready         up                 9 system       desc
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
```

9. File system disk usage is shown in Example 2-20.

*Example 2-20 mmdf (empty file system)*

```
lnxnode1:~ # mmdf fcp_fs1
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 1.3 TB)
site_A_disk_1        31457280        1 Yes      Yes        28546560 ( 91%)           616 ( 0%)
site_A_disk_2        31457280        1 Yes      Yes        28545536 ( 91%)          2920 ( 0%)
site_A_disk_3        31457280        1 Yes      Yes        28545536 ( 91%)           616 ( 0%)
site_A_disk_4        31457280        1 Yes      Yes        28546048 ( 91%)           616 ( 0%)
site_B_disk_1        31457280        2 Yes      Yes        28546304 ( 91%)          2664 ( 0%)
site_B_disk_2        31457280        2 Yes      Yes        28545536 ( 91%)           600 ( 0%)
site_B_disk_3        31457280        2 Yes      Yes        28545536 ( 91%)           632 ( 0%)
site_B_disk_4        31457280        2 Yes      Yes        28546560 ( 91%)           616 ( 0%)
site_C_disk_1       156290904        3 No       No                0 (  0%)             0 ( 0%)
                -------------                       -------------------- -------------------
```

```
(pool total)       407949144                                  228367616 ( 56%)          9280 ( 0%)

                =============                         ==================== ===================
(total)            407949144                                  228367616 ( 56%)          9280 ( 0%)
```

10. We write some data on the file system and recheck the usage as shown in Example 2-21. The data written in the file system is distributed evenly on all disks in both site A and site B, but not in site C.

*Example 2-21 mmdf after write operation*

```
lnxnode1:~ # mmdf fcp_fs1
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 1.3 TB)
site_A_disk_1        31457280        1 Yes      Yes         27797504 ( 88%)          1048 ( 0%)
site_A_disk_2        31457280        1 Yes      Yes         27797504 ( 88%)          3368 ( 0%)
site_A_disk_3        31457280        1 Yes      Yes         27797760 ( 88%)          1048 ( 0%)
site_A_disk_4        31457280        1 Yes      Yes         27798016 ( 88%)           808 ( 0%)
site_B_disk_1        31457280        2 Yes      Yes         27797760 ( 88%)          3368 ( 0%)
site_B_disk_2        31457280        2 Yes      Yes         27797760 ( 88%)           792 ( 0%)
site_B_disk_3        31457280        2 Yes      Yes         27796992 ( 88%)          1320 ( 0%)
site_B_disk_4        31457280        2 Yes      Yes         27798016 ( 88%)          1048 ( 0%)
site_C_disk_1       156290904        3 No       No                 0 (  0%)             0 ( 0%)
                -------------                         -------------------- -------------------
(pool total)        407949144                                222381312 ( 55%)         12800 ( 0%)

                =============                         ==================== ===================
(total)             407949144                                222381312 ( 55%)         12800 ( 0%)
```

## 2.2.2 Activating SCSI-3 persistent reservation

If the storage used supports SCSI-3 Persistent Reservation (PR), this should be enabled for cluster use. Enabling SCSI-3 PR speeds up recovery actions in case of node failure.

11. Use SCSI-3 PR as the disk fencing mechanism.
    - Even though the disk subsystem might be capable of handling SCSI-3 PR, the cluster does not (by default) use the PR fencing mechanism, as shown in Example 2-22.

> **Tip:** The default disk fencing mechanism used in a Spectrum Scale cluster is *disk leasing*. This is implemented in Spectrum Scale software and does not require specific disk configuration. Disk leasing is used for storage subsystems that do not support the SCSI-3 PR mechanism.
>
> Check the Spectrum Scale FAQs for supported storage subsystems.

*Example 2-22 NSD disk w/o SCSI-3 PR*

```
lnxnode1:~ # mmlsnsd -X

 Disk name    NSD volume ID      Device         Devtype  Node name     Remarks
---------------------------------------------------------------------------------
 site_A_disk_1 0714090C564A62FB   /dev/dm-1      dmm      lnxnode1  server node
 site_A_disk_1 0714090C564A62FB   /dev/dm-0      dmm      lnxnode2  server node
```

```
site_A_disk_2 0715090C564A62FF   /dev/dm-0      dmm      lnxnode1  server node
site_A_disk_2 0715090C564A62FF   /dev/dm-1      dmm      lnxnode2  server node
site_A_disk_3 0714090C564A6300   /dev/dm-2      dmm      lnxnode1  server node
site_A_disk_3 0714090C564A6300   /dev/dm-2      dmm      lnxnode2  server node
site_A_disk_4 0715090C564A6304   /dev/dm-3      dmm      lnxnode1  server node
site_A_disk_4 0715090C564A6304   /dev/dm-4      dmm      lnxnode2  server node
site_B_disk_1 0719090C564A6306   /dev/dm-2      dmm      lnxnode5  server node
site_B_disk_1 0719090C564A6306   /dev/dm-1      dmm      lnxnode6  server node
site_B_disk_2 071A090C564A630B   /dev/dm-1      dmm      lnxnode5  server node
site_B_disk_2 071A090C564A630B   /dev/dm-0      dmm      lnxnode6  server node
site_B_disk_3 0719090C564A6310   /dev/dm-3      dmm      lnxnode5  server node
site_B_disk_3 0719090C564A6310   /dev/dm-2      dmm      lnxnode6  server node
site_B_disk_4 071A090C564A6314   /dev/dm-0      dmm      lnxnode5  server node
site_B_disk_4 071A090C564A6314   /dev/dm-3      dmm      lnxnode6  server node
site_C_disk_1 090C071C564A4A34   /dev/sdb       generic  node7tie  server node
```

12. Unmount the file system:

```
mmunmount fcp_fs1 -a
```

13. Shutdown the cluster:

```
mmshutdown -a
```

14. Change the **usePersistentReserve** parameter to the cluster configuration as shown in Example 2-23.

*Example 2-23 usePersistentReserve parameter*

```
lnxnode1:~ # mmchconfig usePersistentReserve=yes
Verifying GPFS is stopped on all nodes ...
mmchconfig: Processing disk site_A_disk_1
mmchconfig: Processing disk site_A_disk_2
mmchconfig: Processing disk site_A_disk_3
mmchconfig: Processing disk site_A_disk_4
mmchconfig: Processing disk site_B_disk_1
mmchconfig: Processing disk site_B_disk_2
mmchconfig: Processing disk site_B_disk_3
mmchconfig: Processing disk site_B_disk_4
mmchconfig: Command successfully completed
mmchconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

15. Set the failure detection time to 10 seconds by running the command:

```
mmchconfig failureDetectionTime=10
```

16. Start the cluster by running the mmstartup -a command.

17. Check that SCSI reservation is active (see Example 2-24).

*Example 2-24 NSD disk after SCSI reservation set up*

```
lnxnode1:~ # mmlsnsd -X

 Disk name    NSD volume ID      Device         Devtype  Node name            Remarks
-------------------------------------------------------------------------------------------
 site_A_disk_1 0714090C564A62FB   /dev/dm-1      dmm      lnxnode1  server node,pr=yes
 site_A_disk_1 0714090C564A62FB   /dev/dm-0      dmm      lnxnode2  server node,pr=yes
 site_A_disk_2 0715090C564A62FF   /dev/dm-0      dmm      lnxnode1  server node,pr=yes
 site_A_disk_2 0715090C564A62FF   /dev/dm-1      dmm      lnxnode2  server node,pr=yes
 site_A_disk_3 0714090C564A6300   /dev/dm-2      dmm      lnxnode1  server node,pr=yes
```

```
site_A_disk_3 0714090C564A6300   /dev/dm-2      dmm      lnxnode2  server node,pr=yes
site_A_disk_4 0715090C564A6304   /dev/dm-3      dmm      lnxnode1  server node,pr=yes
site_A_disk_4 0715090C564A6304   /dev/dm-4      dmm      lnxnode2  server node,pr=yes
site_B_disk_1 0719090C564A6306   /dev/dm-2      dmm      lnxnode5  server node,pr=yes
site_B_disk_1 0719090C564A6306   /dev/dm-1      dmm      lnxnode6  server node,pr=yes
site_B_disk_2 071A090C564A630B   /dev/dm-1      dmm      lnxnode5  server node,pr=yes
site_B_disk_2 071A090C564A630B   /dev/dm-0      dmm      lnxnode6  server node,pr=yes
site_B_disk_3 0719090C564A6310   /dev/dm-3      dmm      lnxnode5  server node,pr=yes
site_B_disk_3 0719090C564A6310   /dev/dm-2      dmm      lnxnode6  server node,pr=yes
site_B_disk_4 071A090C564A6314   /dev/dm-0      dmm      lnxnode5  server node,pr=yes
site_B_disk_4 071A090C564A6314   /dev/dm-3      dmm      lnxnode6  server node,pr=yes
site_C_disk_1 090C071C564A4A34   /dev/sdb       generic  node7tie  server node
```

## 2.2.3 Testing the cluster

In this section, we perform two tests on the configuration previously defined:

- Complete site failure (for site A).
- Site A loses connectivity to site B and site C. Nodes in site A continue to communicate with each other and have access to storage in site A.

We observe cluster behavior before and during site failure. Finally, we bring back site A online and perform recovery operations to bring file system to a consistent, replicated status.

### *Initial cluster and file system status*

Before we test the cluster, we verify the cluster status (as a baseline):

- List the manager node for the cluster as shown in Example 2-25.

*Example 2-25 File system manager and cluster manager roles*

```
lnxnode1:~ # mmlsmgr
file system      manager node
---------------- ------------------
fcp_fs1          9.12.7.26 (lnxnode6)

Cluster manager node: 9.12.7.26 (lnxnode6)
```

- List the file attributes as shown in Example 2-26. The replication factor for each file and its metadata is (2).

*Example 2-26 Files' attributes*

```
lnxnode6:~ # mmlsattr /fcp_fs1/*
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- ---------------
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD1.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD2.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD3.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-x86_64-GM-DVD1.iso
```

- Detailed configuration data can be displayed by running the **mmfsadm** command, as shown in Example 2-27.

*Example 2-27 Cluster management status (mmfsadm dump cfgmgr)*

```
lnxnode6:~ # mmfsadm dump cfgmgr
nClusters 1

Cluster Configuration [0] "FCP-Cluster.lnxnode1": id AC779689564A5CCF
ccUseCount 1 unused since (never) contactListRefreshMethod 0
Domain , myAddr <c0n0>, authIsRequired true, ccMaxFeatureLevel 1501, ccClusterFlags 0x0000000B
UID domain 0x80000885D60 (0x1FFC0000885D60)  Name "FCP-Cluster.lnxnode1"
  hold count 1 CredCacheHT 0x0 IDCacheHT 0x0
No of nodes: 7 total, 7 local, 5 quorum nodes, 16384 max

Authorized keys list:
clusterName              port cKeyGen nKeyGen cipherList
FCP-Cluster.lnxnode1        0       1      -1 AUTHONLY

Cluster info list:
clusterName              port cKeyGen nKeyGen cipherList
FCP-Cluster.lnxnode1     1191      -1      -1 AUTHONLY

node     node               primary          admin  OS --status---   join fail  SGs cnfs   rcksum   wcksum -lease-renewals--
--heartbeats-- other ip addrs,
  no  address host name      ip address        func     tr p    rpc  seqNo  cnt mngd  grp mismatch mismatch sent    processed received
pings last failure
---- -------- ------------- ------------- ------- --- ----------- ------ ---- ---- ---- -------- -------- -----------------
-------------- -------------------
   1  <c0n3> lnxnode1      9.12.7.20     qQm--l-- Lnx -- J     up      2    0   1     0        0        0 47764.59 47764.60 47764.60
   2  <c0n1> lnxnode2      9.12.7.21     qQ---l-- Lnx -- J     up      1    0   0     0        0        0 47764.91 47764.99 47764.99
   6  <c0n5> lnxnode3      9.12.7.22     -----l-- Lnx -- J     up      3    0   0     0        0        0 47763.53 47763.53 47763.53
   8  <c0n6> lnxnode4      9.12.7.24     -----l-- Lnx -- J     up      4    0   0     0        0        0 47764.19 47764.19 47764.19
   3  <c0n2> lnxnode5      9.12.7.25     qQ---l-- Lnx -- J     up      1    0   0     0        0        0 47764.85 47764.87 47764.87
   4  <c0n0> lnxnode6      9.12.7.26     qQm--l-- Lnx -- J     up      1    0   0     0        0        0          69471.23 69471.23
   5  <c0n4> node7tie      9.12.7.28     qQ---l-- Lnx -- J     up      1    0   0     0        0        0 47765.82 47766.03 47766.03

Current clock tick (seconds since boot): 43047766.29 (resolution 0.010) = 2015-11-18 14:49:37

Groupleader <c0n0> 0x00000000 (this node)
Cluster manager is <c0n0> (this node); pendingOps 0
group quorum formation time 2015-11-17 17:04:42
gid 564ba47a:090c071a elect <4:2> seq 4 pendingSeq 4, gpdIdle phase 0, joined
Node ready? yes
quorumMethod NodeCCR (ccClusterFlags B, ccrEnabled 1), ccrCommitSeqNum 54
GroupLeader: joined 1 gid <4:2>
lease config: dynamic yes failureDetectionTime 10.0 usePR yes recoveryWait 35 dmsTimeout 23
  leaseDuration 6.7/6.7 renewalInterval 3.3/3.3 renewalTimeout 3.3 fuzz 0.33/0.33
  missedPingTimeout 6x0.6=3.3 totalPingTimeout 216x0.6=120.0
lastFailedLeaseGranted: 0.00
lastLeaseObtained      43047764.91, 5.28 sec left (ok)
leaderLeaseQuorum 2
hasTiebreaker no
renewalList 2 of 2: (4 47765.82) (1 47764.91)
nextChallengeCheck      (none scheduled)
lastLeaderChangeTime: 42969471.23 (78295 seconds ago)
stats: nTemporaryLeaseLoss 0 nSupendIO 0 nLeaseOverdue 0 nTakeover 0 nPinging 0
Summary of lease renewal round-trip times:
  Number of keys = 3, total count 148519
  Min 0.0 Max 1.0, Most common 0.0 (109196)
  Mean 0.1, Median 0.0, 99th 1.0
ccSANergyExport no

Assigned stripe group managers:
  "fcp_fs1"  id 071A090C:564B8F94  cond 0x3FF8CEC0F50  sgiHold 1  SG mgr <c0n3> seq 1365965146
```

```
  Appointed at 1447799955.296562000; done recovery=true
    DMAPI disposition is not set for any event.
  sgStats: tmSpace 0 freeSpace 0 tmRequests 0 cpuUsage 0 totalTmRequests 0

...<< Snippet >>...

pcacheHashVersion 2
CrHashTable 0x3FF6C001CB0 n 1
  cmd sock 36 cookie 1359238349 owner 19034 id 0x3E50E015F5000010(#16) uses 1 type 14 start
1447876177.561533 flags 0x106 SG none line 'dump cfgmgr'
```

## Testing the configuration: Complete site A failure

**Test description:** In this test, the nodes in failed site A have no connectivity whatsoever (network), nor are they connected to their storage.

Note that we do not cover in detail the actions we perform to bring down Site A because the testing plan and methods should follow the guidelines and requirements of your IT environment.

### *Site A fails*

We simulate site down by stopping all nodes in site A:

1. We stop the nodes on site A by shutting down the guests.
2. We check the cluster status as shown in Example 2-28.

   Observe that nodes on site A appear in an “unknown” state as they have no network connectivity.

*Example 2-28 Site A down*

```
lnxnode6:~ # mmgetstate -aL

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
-------------------------------------------------------------------------------------
       1      lnxnode1          0        0          7        unknown     quorum node
       2      lnxnode2          0        0          7        unknown     quorum node
       3      lnxnode5          3        3          7        active      quorum node
       4      lnxnode6          3        3          7        active      quorum node
       5      node7tie          3        3          7        active      quorum node
       6      lnxnode3          0        0          7        unknown
       8      lnxnode4          3        3          7        active
```

3. List the disks on site A as shown in Example 2-29 to check the status of each disk.

*Example 2-29 Missing disks after site A disaster*

```
lnxnode6:~ # mmlsnsd -M

 Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------------
 site_A_disk_1 0714090C564A62FB   -              lnxnode1  (not found) server node
 site_A_disk_1 0714090C564A62FB   -              lnxnode2  (not found) server node
 site_A_disk_2 0715090C564A62FF   -              lnxnode2  (not found) server node
 site_A_disk_2 0715090C564A62FF   -              lnxnode1  (not found) server node
 site_A_disk_3 0714090C564A6300   -              lnxnode1  (not found) server node
 site_A_disk_3 0714090C564A6300   -              lnxnode2  (not found) server node
```

```
site_A_disk_4 0715090C564A6304   -             lnxnode2  (not found) server node
site_A_disk_4 0715090C564A6304   -             lnxnode1  (not found) server node
site_B_disk_1 0719090C564A6306   /dev/dm-2     lnxnode5  server node
site_B_disk_1 0719090C564A6306   /dev/dm-1     lnxnode6  server node
site_B_disk_2 071A090C564A630B   /dev/dm-1     lnxnode5  server node
site_B_disk_2 071A090C564A630B   /dev/dm-0     lnxnode6  server node
site_B_disk_3 0719090C564A6310   /dev/dm-3     lnxnode5  server node
site_B_disk_3 0719090C564A6310   /dev/dm-2     lnxnode6  server node
site_B_disk_4 071A090C564A6314   /dev/dm-0     lnxnode5  server node
site_B_disk_4 071A090C564A6314   /dev/dm-3     lnxnode6  server node
site_C_disk_1 090C071C564A4A34   /dev/sdb      node7tie  server node

mmlsnsd: The following nodes could not be reached:

lnxnode2
lnxnode1
lnxnode3
```

4. Check the files' attributes for the existing files as shown in Example 2-30.

   Observe that the *existing* files appear as *replicated*, which is true because they were replicated at the time site A nodes were up and running.

   *Example 2-30 Files' attributes*

```
lnxnode6:~ # mmlsattr /fcp_fs1/*
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- ---------------
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD1.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD2.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD3.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-x86_64-GM-DVD1.iso
```

5. We add new files to the file system while site A is down and check the files' attributes (see Example 2-31). The files created while site A is unavailable will not have data and metadata replicated.

*Example 2-31 mmlsattr showing new files*

```
node7tie:~ # mmlsattr /fcp_fs1/*
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- ---------------
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-s390x-GM-DVD1.iso       [dataupdatemiss,metaupdatemiss]
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-s390x-GM-DVD2.iso       [dataupdatemiss,metaupdatemiss]
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-s390x-GM-DVD3.iso       [dataupdatemiss,metaupdatemiss]
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-x86_64-GM-DVD1.iso      [dataupdatemiss,metaupdatemiss]
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD1.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD2.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD3.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-x86_64-GM-DVD1.iso
```

6. List the file system descriptors as shown in Example 2-32 on page 33.

   Observe that the disks in site A (failure group 1) are marked as "down" and the third active file system descriptor has moved to *site_B_disk_2* (compare to initial status shown in Example 2-19 on page 26).

*Example 2-32 File system descriptor status*

```
lnxnode6:~ # mmlsdisk  fcp_fs1 -L
disk         driver   sector    failure holds    holds                                   storage
name         type       size     group metadata data  status availability disk id pool    remarks
------------ -------- ------ --------- -------- ----- ------ ------------ ------- ------- -----
site_A_disk_1 nsd         512       1 Yes      Yes   ready    down          1 system
site_A_disk_2 nsd         512       1 Yes      Yes   ready    down          2 system
site_A_disk_3 nsd         512       1 Yes      Yes   ready    down          3 system
site_A_disk_4 nsd         512       1 Yes      Yes   ready    down          4 system
site_B_disk_1 nsd         512       2 Yes      Yes   ready    up            5 system       desc
site_B_disk_2 nsd         512       2 Yes      Yes   ready    up            6 system       desc
site_B_disk_3 nsd         512       2 Yes      Yes   ready    up            7 system
site_B_disk_4 nsd         512       2 Yes      Yes   ready    up            8 system
site_C_disk_1 nsd         512       3 No       No    ready    up            9 system       desc
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
```

**Note:** In this state, the Spectrum Scale cluster continues to provide data access even if site A is not available.

### *Site A recovery*

The following steps are taken for site A recovery:

7. We start the nodes in site A. Initially the nodes have status *down* (the operating system is started, but **mmfsd** is not running). When the mmfsd on nodes in site A starts, the nodes rejoin the cluster and change to *active* status, as shown in Example 2-33.

*Example 2-33 Nodes' status after starting site A*

```
lnxnode6:~ # mmgetstate -aL

 Node number  Node name      Quorum  Nodes up  Total nodes  GPFS state  Remarks
-------------------------------------------------------------------------------------
       1      lnxnode1          0        0          7       down        quorum node
       2      lnxnode2          0        0          7       down        quorum node
       3      lnxnode5          3        3          7       active      quorum node
       4      lnxnode6          3        3          7       active      quorum node
       5      node7tie          3        3          7       active      quorum node
       6      lnxnode3          0        0          7       down
       8      lnxnode4          3        3          7       active


lnxnode6:~ # mmstartup -N lnxnode1,lnxnode2,lnxnode3
Wed Nov 18 15:48:49 EST 2015: mmstartup: Starting GPFS ...

lnxnode6:~ # mmgetstate -aL

 Node number  Node name      Quorum  Nodes up  Total nodes  GPFS state  Remarks
-------------------------------------------------------------------------------------
       1      lnxnode1          3        5          7       active      quorum node
       2      lnxnode2          3        5          7       active      quorum node
       3      lnxnode5          3        5          7       active      quorum node
       4      lnxnode6          3        5          7       active      quorum node
       5      node7tie          3        5          7       active      quorum node
```

```
        6      lnxnode3            3         5           7        active
        8      lnxnode4            3         5           7        active
```

8. Even with all nodes up and running, the data belonging to the files that have changed while site A was unavailable are not yet replicated, as shown in Example 2-34.

*Example 2-34 File status before recovery start*

```
lnxnode6:~ # mmlsattr /fcp_fs1/*
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- ---------------
  2 (  2)   2 (  2) /fcp_fs1/dir1
  2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-s390x-GM-DVD1.iso   [dataupdatemiss,metaupdatemiss]
  2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-s390x-GM-DVD2.iso   [dataupdatemiss,metaupdatemiss]
  2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-s390x-GM-DVD3.iso   [dataupdatemiss,metaupdatemiss]
  2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-x86_64-GM-DVD1.iso  [dataupdatemiss,metaupdatemiss]
  2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD1.iso
  2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD2.iso
  2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD3.iso
  2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-x86_64-GM-DVD1.iso
```

9. We initiate the recovery of the file system. At this point, the NSD disks in site A are still *down* (see Example 2-32 on page 33) and remain *down* until the NSD disks are started manually, as shown in Example 2-35.

*Example 2-35 Starting disks for site A*

```
lnxnode6:~ # mmchdisk fcp_fs1 start -a
mmnsddiscover:  Attempting to rediscover the disks.  This may take a while ...
mmnsddiscover:  Finished.
lnxnode5:  Rediscovered nsd server access to site_B_disk_1.
lnxnode2:  Rediscovered nsd server access to site_A_disk_1.
lnxnode1:  Rediscovered nsd server access to site_A_disk_1.
lnxnode5:  Rediscovered nsd server access to site_B_disk_2.
lnxnode2:  Rediscovered nsd server access to site_A_disk_2.
lnxnode1:  Rediscovered nsd server access to site_A_disk_2.
lnxnode6:  Rediscovered nsd server access to site_B_disk_1.
lnxnode6:  Rediscovered nsd server access to site_B_disk_2.
node7tie:  Rediscovered nsd server access to site_C_disk_1.
lnxnode5:  Rediscovered nsd server access to site_B_disk_3.
lnxnode5:  Rediscovered nsd server access to site_B_disk_4.
lnxnode2:  Rediscovered nsd server access to site_A_disk_3.
lnxnode2:  Rediscovered nsd server access to site_A_disk_4.
lnxnode1:  Rediscovered nsd server access to site_A_disk_3.
lnxnode1:  Rediscovered nsd server access to site_A_disk_4.
lnxnode6:  Rediscovered nsd server access to site_B_disk_3.
lnxnode6:  Rediscovered nsd server access to site_B_disk_4.
Scanning file system metadata, phase 1 ...
  51 % complete on Wed Nov 18 15:57:43 2015
 100 % complete on Wed Nov 18 15:57:45 2015
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
   6.98 % complete on Wed Nov 18 15:58:19 2015  (    390921 inodes with total      16021 MB data processed)
```

```
   8.29 % complete on Wed Nov 18 15:58:41 2015  (    390921 inodes with total      19021 MB data processed)
   9.19 % complete on Wed Nov 18 15:59:07 2015  (    390921 inodes with total      21106 MB data processed)
  10.43 % complete on Wed Nov 18 15:59:29 2015  (    390921 inodes with total      23948 MB data processed)
  11.34 % complete on Wed Nov 18 15:59:51 2015  (    390921 inodes with total      26034 MB data processed)
  13.23 % complete on Wed Nov 18 16:00:18 2015  (    390921 inodes with total      30375 MB data processed)
  14.14 % complete on Wed Nov 18 16:00:53 2015  (    390921 inodes with total      32469 MB data processed)
  15.80 % complete on Wed Nov 18 16:01:13 2015  (    392711 inodes with total      36278 MB data processed)
  17.27 % complete on Wed Nov 18 16:01:38 2015  (    392711 inodes with total      39653 MB data processed)
 100.00 % complete on Wed Nov 18 16:01:47 2015  (    398848 inodes with total      39882 MB data processed)
Scan completed successfully.
```

**Note:** This process might take a while depending on the file system size and network bandwidth between the sites.

10. While the **mmchdisk** command is running, the NSD disks in site A are in a "recovering" state, as shown in Example 2-36.

*Example 2-36 Recovering disks*

```
node7tie:~ # mmlsdisk fcp_fs1
disk driver sector failure holds holds storage
name type size group metadata data status availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
site_A_disk_1 nsd 512 1 yes yes ready recovering system
site_A_disk_2 nsd 512 1 yes yes ready recovering system
site_A_disk_3 nsd 512 1 yes yes ready recovering system
site_A_disk_4 nsd 512 1 yes yes ready recovering system
site_B_disk_1 nsd 512 2 yes yes ready up system
site_B_disk_2 nsd 512 2 yes yes ready up system
site_B_disk_3 nsd 512 2 yes yes ready up system
site_B_disk_4 nsd 512 2 yes yes ready up system
site_C_disk_1 nsd 512 3 no no ready up system

node7tie:~ # mmlsdisk fcp_fs1 -L
disk driver sector failure holds holds storage
name type size group metadata data status availability disk id pool remarks
------------ -------- ------ ----------- -------- ----- ------------- ------------
site_A_disk_1 nsd 512 1 yes yes ready recovering 1 system desc
site_A_disk_2 nsd 512 1 yes yes ready recovering 2 system
site_A_disk_3 nsd 512 1 yes yes ready recovering 3 system
site_A_disk_4 nsd 512 1 yes yes ready recovering 4 system
site_B_disk_1 nsd 512 2 yes yes ready up 5 system desc
site_B_disk_2 nsd 512 2 yes yes ready up 6 system
site_B_disk_3 nsd 512 2 yes yes ready up 7 system
site_B_disk_4 nsd 512 2 yes yes ready up 8 system
site_C_disk_1 nsd 512 3 no no ready up 9 system desc
Number of quorum disks: 3
Read quorum value: 2
Write quorum value: 2
```

11. At the end of the recovery process, all files are fully replicated as shown in Example 2-37.

    Observe that all files have data and metadata replicated and the file system descriptor is back on failure group 1.

*Example 2-37 Normal operation restored*

```
lnxnode6:~ # mmlsattr /fcp_fs1/*
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- ---------------
      2 (  2)   2 (  2) /fcp_fs1/dir1
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-s390x-GM-DVD1.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-s390x-GM-DVD2.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-s390x-GM-DVD3.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-SDK-DVD-x86_64-GM-DVD1.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD1.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD2.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-s390x-GM-DVD3.iso
      2 (  2)   2 (  2) /fcp_fs1/SLE-12-Server-DVD-x86_64-GM-DVD1.iso


lnxnode6:~ # mmlsdisk fcp_fs1 -L
disk         driver  sector failure holds   holds                            storage
name         type    size   group  metadata data  status  availability disk id pool    remarks
------------ ------- ------ --------- -------- ----- ------- ------------ ------- ------------
---------
site_A_disk_1 nsd         512           1 Yes      Yes   ready   up           1 system    desc
site_A_disk_2 nsd         512           1 Yes      Yes   ready   up           2 system
site_A_disk_3 nsd         512           1 Yes      Yes   ready   up           3 system
site_A_disk_4 nsd         512           1 Yes      Yes   ready   up           4 system
site_B_disk_1 nsd         512           2 Yes      Yes   ready   up           5 system    desc
site_B_disk_2 nsd         512           2 Yes      Yes   ready   up           6 system
site_B_disk_3 nsd         512           2 Yes      Yes   ready   up           7 system
site_B_disk_4 nsd         512           2 Yes      Yes   ready   up           8 system
site_C_disk_1 nsd         512           3 No       No    ready   up           9 system    desc
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
```

### Testing the cluster: Site A cannot communicate with site B

In this section, we test the configuration previously defined by simulating network connectivity failure (for site A). We observe cluster and node behavior before and during the event. Finally, we restore network connectivity to site A online and perform recovery operations to bring the file system to a consistent, replicated status.

**Test description:** In this test, the nodes in site A continue to communicate with each other and have connectivity to local storage (disks in failure group 1), but cannot communicate with either site B or site C. Nodes in site B communicate with site C.

**Obs.:** We do not cover in detail how to induce network connectivity failure to site A.

### Site A loses network connectivity to site B and site C

We simulate site A network connectivity loss to the rest of the cluster:

1. The initial status has all nodes up and running, as shown in Example 2-38.

*Example 2-38 All nodes are active*

```
lnxnode6:~ # mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      lnxnode1         active
       2      lnxnode2         active
       3      lnxnode5         active
       4      lnxnode6         active
       5      node7tie         active
       6      lnxnode3         active
       8      lnxnode4         active
```

At this time, all NSDs are operational as shown in Example 2-39.

*Example 2-39 NSD disks are operational*

```
lnxnode6:~ # mmlsnsd -X

 Disk name    NSD volume ID      Device         Devtype  Node name                Remarks
---------------------------------------------------------------------------------------------------
 site_A_disk_1 0714090C564A62FB   /dev/dm-1      dmm      lnxnode1                 server node,pr=yes
 site_A_disk_1 0714090C564A62FB   /dev/dm-1      dmm      lnxnode2                 server node,pr=yes
 site_A_disk_2 0715090C564A62FF   /dev/dm-3      dmm      lnxnode1                 server node,pr=yes
 site_A_disk_2 0715090C564A62FF   /dev/dm-2      dmm      lnxnode2                 server node,pr=yes
 site_A_disk_3 0714090C564A6300   /dev/dm-0      dmm      lnxnode1                 server node,pr=yes
 site_A_disk_3 0714090C564A6300   /dev/dm-3      dmm      lnxnode2                 server node,pr=yes
 site_A_disk_4 0715090C564A6304   /dev/dm-2      dmm      lnxnode1                 server node,pr=yes
 site_A_disk_4 0715090C564A6304   /dev/dm-0      dmm      lnxnode2                 server node,pr=yes
 site_B_disk_1 0719090C564A6306   /dev/dm-2      dmm      lnxnode5                 server node,pr=yes
 site_B_disk_1 0719090C564A6306   /dev/dm-1      dmm      lnxnode6                 server node,pr=yes
 site_B_disk_2 071A090C564A630B   /dev/dm-1      dmm      lnxnode5                 server node,pr=yes
 site_B_disk_2 071A090C564A630B   /dev/dm-0      dmm      lnxnode6                 server node,pr=yes
 site_B_disk_3 0719090C564A6310   /dev/dm-3      dmm      lnxnode5                 server node,pr=yes
 site_B_disk_3 0719090C564A6310   /dev/dm-2      dmm      lnxnode6                 server node,pr=yes
 site_B_disk_4 071A090C564A6314   /dev/dm-0      dmm      lnxnode5                 server node,pr=yes
 site_B_disk_4 071A090C564A6314   /dev/dm-3      dmm      lnxnode6                 server node,pr=yes
 site_C_disk_1 090C071C564A4A34   /dev/sdb       generic  node7tie                 server node


lnxnode6:~ # mmlsdisk fcp_fs1 -M

Disk name     IO performed on node     Device            Availability
------------  -----------------------  ----------------- ------------
site_A_disk_1 lnxnode1                 /dev/dm-1         up
site_A_disk_2 lnxnode2                 /dev/dm-2         up
site_A_disk_3 lnxnode1                 /dev/dm-0         up
site_A_disk_4 lnxnode2                 /dev/dm-0         up
site_B_disk_1 localhost                /dev/dm-1         up
site_B_disk_2 localhost                /dev/dm-0         up
site_B_disk_3 localhost                /dev/dm-2         up
site_B_disk_4 localhost                /dev/dm-3         up
site_C_disk_1 node7tie                 /dev/sdb          up
```

**Tip:** The **mmlsdisk** command with the *-m* flag shows how the NSD devices are accessed from the local node. The *-M* flag shows all NSD devices and the nodes they are accessed through.

2. We disconnect the network (IP) connection between sites. Site A is considered down and site B is used as a recovery site in this case as it can communicate with site C (tiebreaker node is reachable from site B).
3. The logs on the cluster nodes on site B are shown in Example 2-40. The nodes from site A are fenced from accessing the disks (including site A disks) and expelled from the cluster due to loss of network communication. Also, the disks from site A are marked as *failed*.

*Example 2-40 Site B nodes' cluster logs (excerpt)*

```
Fri Nov 20 09:53:34.660 2015: [N] Node 9.12.7.21 (lnxnode2) lease renewal is overdue. Pinging to check if
it is alive
Fri Nov 20 09:53:37.981 2015: [E] Node 9.12.7.21 (lnxnode2) is being expelled because of an expired lease.
Pings sent: 6. Replies received: 0.
Fri Nov 20 09:53:41.972 2015: [N] Node 9.12.7.20 (lnxnode1) lease renewal is overdue. Pinging to check if
it is alive
Fri Nov 20 09:53:45.303 2015: [E] Node 9.12.7.20 (lnxnode1) is being expelled because of an expired lease.
Pings sent: 6. Replies received: 0.
Fri Nov 20 09:53:45.305 2015: [I] Recovering nodes: 9.12.7.20 9.12.7.21
Fri Nov 20 09:53:45.715 2015: [E] Disk failure.  Volume fcp_fs1. rc = 19. Physical volume site_A_disk_1.
Fri Nov 20 09:53:45.716 2015: [E] Disk failure.  Volume fcp_fs1. rc = 19. Physical volume site_A_disk_2.
Fri Nov 20 09:53:45.717 2015: [E] Disk failure.  Volume fcp_fs1. rc = 19. Physical volume site_A_disk_3.
Fri Nov 20 09:53:45.718 2015: [E] Disk failure.  Volume fcp_fs1. rc = 19. Physical volume site_A_disk_4.
Fri Nov 20 09:53:46.115 2015: [I] Recovery: fcp_fs1, delay 30 sec. for safe recovery.
Fri Nov 20 09:54:15.874 2015: [N] Node 9.12.7.22 (lnxnode3) lease renewal is overdue. Pinging to check if
it is alive
Fri Nov 20 09:54:19.205 2015: [E] Node 9.12.7.22 (lnxnode3) is being expelled because of an expired lease.
Pings sent: 6. Replies received: 0.
```

4. The logs on the cluster nodes on site A are shown in Example 2-41. The nodes on site A try to reconnect with the nodes on site B, but fail.

   When time-out is reached, each node detaches itself from the cluster (because it cannot contact enough quorum nodes) and unmounts the shared file system. After that, it keeps trying to contact the cluster manager.

*Example 2-41 Site A nodes' cluster logs*

```
Fri Nov 20 09:53:41.744 2015: [N] Node 9.12.7.26 (lnxnode6) lease renewal is
overdue. Pinging to check if it is alive
Fri Nov 20 09:55:41.726 2015: [I] Node 9.12.7.26 (lnxnode6): ping timed out. Pings
sent: 216. Replies received: 216.
Fri Nov 20 09:55:41.727 2015: Sending request to collect expel debug data to
lnxnode6 localNode
Fri Nov 20 09:55:41.728 2015: [I] Calling User Exit Script gpfsSendRequestToNodes:
event sendRequestToNodes, Async command /usr/lpp/mmfs/bin/mmcommon.
Fri Nov 20 09:55:42.287 2015: [I] Tracing in blocking mode
Trace started: Wait 20 seconds before cut and stop trace
Fri Nov 20 09:55:51.742 2015: [I] Lease is overdue. Probing cluster
FCP-Cluster.lnxnode1
Fri Nov 20 09:55:52.243 2015: [N] Connecting to 9.12.7.21 lnxnode2 <c0n1>
Fri Nov 20 09:55:52.244 2015: [N] Connecting to 9.12.7.25 lnxnode5 <c0n2>
Fri Nov 20 09:55:52.245 2015: [N] Connecting to 9.12.7.28 node7tie <c0n4>
```

```
Fri Nov 20 09:55:52.246 2015: [E] Unable to contact enough other quorum nodes
during cluster probe.
Fri Nov 20 09:55:52.247 2015: [E] Lost membership in cluster FCP-Cluster.lnxnode1.
Unmounting file systems.
Fri Nov 20 09:55:52.295 2015: [N] Close connection to 9.12.7.26 lnxnode6 <c0n0>
(Node failed)
Fri Nov 20 09:55:52.802 2015: [N] Connecting to 9.12.7.26 lnxnode6 <c0p0>
```

5. The nodes in site A switch to an *arbitrating* state as shown in Example 2-42. The nodes remain in this mode until the network connection to nodes in site B and site C is recovered.

*Example 2-42 Arbitrating mode*

```
lnxnode1:~ # mmgetstate

 Node number  Node name        GPFS state
------------------------------------------
       1      lnxnode1         arbitrating
```

#### *Network connectivity to site A restored: Recovery*

When all the nodes are reconnected, the nodes in site A join the cluster and the file system will be remounted on the joining nodes, as shown in Example 2-43.

*Example 2-43 Remounting the file system*

```
2015-11-20T10:10:24.715993-05:00 lnxnode1 mmfs: [N] Remounted fcp_fs1
2015-11-20T10:10:24.726078-05:00 lnxnode1 mmfsd ready
```

Recovery after nodes in site A rejoin the cluster is similar to the previous scenario. See“Site A recovery” on page 33.

## 2.2.4 Exporting a file system

In Spectrum Scale, exporting a file system means to stop using a specific file system in a cluster with the main purpose to migrate it on a different cluster (there could be other reasons, such as: System upgrade, nodes maintenance, physical cluster or storage relocation). When a file system is exported, the data stored on the disks is not deleted.

The **mmexportfs** command is used to perform the export. The **mmimportfs** command is used to import a previously exported file system.

When a file system is exported, a text file containing information about the file system is created. The information contains NSD configuration, file system settings, and so on.

For more information about file systems export and import, see this site:

https://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/com.ibm.spectrum.scale.v4r2.adv.doc/bl1adv_fsmexpcl.htm

In our test environment, for the first test we created a file system using FCP disks. In the second test, we used a different set of disks, extended count key data (z Systems specific). We decided to maintain the cluster definition (not remove the cluster).

To maintain the definition, we exported the file system based on FCP disks for further testing and created a new file system, based on ECKD direct access storage device (DASD):

1. We unmount the file system that we want to export from all nodes in the cluster. The **mmlsmount fcp_fs1 -L** command lists if and where the file system is still mounted (it may be still used on some nodes):

   ```
   mmunmount fcp_fs1 -a
   ```

2. Next, we export the file system from the cluster configuration. The export command and the output of the export text file are shown in Example 2-44.

   The text file contains vital information about the file system:

   – Version of the file system
   – Automount at boot (yes or no)
   – Mount point
   – NSD server for each disk

*Example 2-44 Export the fcp_fs1 file system*

```
lnxnode1:~ # mmexportfs fcp_fs1 -o fcp_fs1-export-file

mmexportfs: Processing file system fcp_fs1 ...
mmexportfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

lnxnode1:~ # cat fcp_fs1-export-file
%%9999%%:00_VERSION_LINE::1501:3:18::lc:lnxnode1:lnxnode6:0:/usr/bin/ssh:/usr/bin/scp:1242756721
3354573007:lc2:1447714006::FCP-Cluster.lnxnode1:1:1:1:3:A:::central:0.0:
%%home%%:10_NODESET_HDR:::8:TCP:AUTHONLY:1191:::9:1501:1501:L:4:::8:8::9:::::::
%%home%%:70_MMFSCFG::1:clusterName:FCP-Cluster.lnxnode1:::::::::::::::::::::
%%home%%:70_MMFSCFG::2:clusterId:12427567213354573007:::::::::::::::::::::
%%home%%:70_MMFSCFG::3:autoload:no:::::::::::::::::::::
%%home%%:70_MMFSCFG::4:dmapiFileHandleSize:32:::::::::::::::::::::
%%home%%:70_MMFSCFG::5:minReleaseLevel:1501 4.2.0.0:::::::::::::::::::::
%%home%%:70_MMFSCFG::6:ccrEnabled:yes:::::::::::::::::::::
%%home%%:70_MMFSCFG::7:cipherList:AUTHONLY:::::::::::::::::::::
%%home%%:70_MMFSCFG::8:usePersistentReserve:yes:::::::::::::::::::::
%%home%%:70_MMFSCFG::9:failureDetectionTime:10:::::::::::::::::::::
%%home%%:30_SG_HEADR:fcp_fs1::150::::0::::no:::::::::::::::
%%home%%:40_SG_ETCFS:fcp_fs1:1:%2Ffcp%5Ffs1:
%%home%%:40_SG_ETCFS:fcp_fs1:2: dev             = /dev/fcp_fs1
%%home%%:40_SG_ETCFS:fcp_fs1:3: vfs             = mmfs
%%home%%:40_SG_ETCFS:fcp_fs1:4: nodename        = -
%%home%%:40_SG_ETCFS:fcp_fs1:5: mount           = mmfs
%%home%%:40_SG_ETCFS:fcp_fs1:6: type            = mmfs
%%home%%:40_SG_ETCFS:fcp_fs1:7: account         = false
%%home%%:50_SG_MOUNT:fcp_fs1::rw:mtime:atime:::::::::::::::::::::
%%home%%:60_SG_DISKS:fcp_fs1:1:site_A_disk_1:0:1:dataAndMetadata:0714090C564A62FB:nsd:lnxnode1,l
nxnode2::prLinux::dmm:user::::::system:lnxnode1,lnxnode2:::::
%%home%%:60_SG_DISKS:fcp_fs1:2:site_A_disk_2:0:1:dataAndMetadata:0715090C564A62FF:nsd:lnxnode2,l
nxnode1::prLinux::dmm:user::::::system:lnxnode2,lnxnode1:::::
%%home%%:60_SG_DISKS:fcp_fs1:3:site_A_disk_3:0:1:dataAndMetadata:0714090C564A6300:nsd:lnxnode1,l
nxnode2::prLinux::dmm:user::::::system:lnxnode1,lnxnode2:::::
```

```
%%home%%:60_SG_DISKS:fcp_fs1:4:site_A_disk_4:0:1:dataAndMetadata:0715090C564A6304:nsd:lnxnode2,l
nxnode1::prLinux::dmm:user::::::system:lnxnode2,lnxnode1:::::
%%home%%:60_SG_DISKS:fcp_fs1:5:site_B_disk_1:0:2:dataAndMetadata:0719090C564A6306:nsd:lnxnode5,l
nxnode6::prLinux::dmm:user::::::system:lnxnode5,lnxnode6:::::
%%home%%:60_SG_DISKS:fcp_fs1:6:site_B_disk_2:0:2:dataAndMetadata:071A090C564A630B:nsd:lnxnode6,l
nxnode5::prLinux::dmm:user::::::system:lnxnode6,lnxnode5:::::
%%home%%:60_SG_DISKS:fcp_fs1:7:site_B_disk_3:0:2:dataAndMetadata:0719090C564A6310:nsd:lnxnode5,l
nxnode6::prLinux::dmm:user::::::system:lnxnode5,lnxnode6:::::
%%home%%:60_SG_DISKS:fcp_fs1:8:site_B_disk_4:0:2:dataAndMetadata:071A090C564A6314:nsd:lnxnode6,l
nxnode5::prLinux::dmm:user::::::system:lnxnode6,lnxnode5:::::
%%home%%:60_SG_DISKS:fcp_fs1:9:site_C_disk_1:0:3:descOnly:090C071C564A4A34:nsd:node7tie::other::
generic:user::::::system:node7tie:::::
```

**Attention:** When a file system is exported, the member NSDs are also removed from the cluster configuration, as shown in Example 2-45.

*Example 2-45 File system and NSDs removed from cluster configuration*

```
node7tie:~/work # mmlsfs all
mmlsfs: No file systems were found.
mmlsfs: Command failed. Examine previous error messages to determine cause.

node7tie:~/work # mmlsnsd -M
mmlsnsd: [I] No disks were found
```

## 2.3 Scenario 2: Storage presented as extended count key data devices

The scenario in this section describes how to configure a new file system using DASD storage in an existing cluster. Cluster configuration is similar to the one presented in 2.1.3, “Configuring the Spectrum Scale cluster” on page 17.

**Important:** Because we exported the file system defined on SCSI storage, and we use DASD storage for this scenario, the disk fencing mechanism must be reverted to disk leasing. Stop the cluster on all nodes and change the following parameters:

```
mmchconfig usePersistentReserve=no
mmchconfig failureDetectionTime=35
```

## 2.3.1 File system definition

In the second part of this, we perform testing by simulating a site failure, and document the recovery steps:

1. Check the DASD configuration. Because the DASDs are virtual devices, they might present identical names at the operating system level. In our scenario, there are four dedicated DASDs in site A and four different dedicated DASDs in site B (different storage). The **lsdasd** command output is shown in Example 2-46.

*Example 2-46 DASD configuration list*

```
lnxnode1:/work # lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0202   active      dasda     94:0    ECKD  4096   21128MB   5409000
0.0.0301   active      dasdb     94:4    ECKD  4096   21128MB   5409000
0.0.0302   active      dasdc     94:8    ECKD  4096   21128MB   5409000
0.0.0303   active      dasdd     94:12   ECKD  4096   21128MB   5409000
0.0.0304   active      dasde     94:16   ECKD  4096   21128MB   5409000

lnxnode6:~ # lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0202   active      dasda     94:0    ECKD  4096   21128MB   5409000
0.0.0301   active      dasdb     94:4    ECKD  4096   21128MB   5409000
0.0.0302   active      dasdc     94:8    ECKD  4096   21128MB   5409000
0.0.0303   active      dasdd     94:12   ECKD  4096   21128MB   5409000
0.0.0304   active      dasde     94:16   ECKD  4096   21128MB   5409000
```

Example 2-47 shows the DASD configuration in site A (lnxnode1 and lnxnode2). Devices with addresses 301, 302, 303, and 304 are used for Spectrum Scale. Node lnxnode3 does not have direct access to these devices.

*Example 2-47 lsdasd -l output - site A disks (shown on lnxnode1)*

```
lnxnode1:/work # lsdasd  -l
0.0.0202/dasda/94:0
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
  eer_enabled:  0
  erplog:       0
  uid:          IBM.750000000W9161.8971.0c.00000001000075620000000000000000

0.0.0301/dasdb/94:4
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
```

```
  eer_enabled:  0
  erplog:       0
  uid:          IBM.750000000W9161.8971.08.0000000100007562000000000000000

0.0.0302/dasdc/94:8
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
  eer_enabled:  0
  erplog:       0
  uid:          IBM.750000000W9161.8971.09.0000000100007562000000000000000

0.0.0303/dasdd/94:12
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
  eer_enabled:  0
  erplog:       0
  uid:          IBM.750000000W9161.8971.0a.0000000100007562000000000000000

0.0.0304/dasde/94:16
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
  eer_enabled:  0
  erplog:       0
  uid:          IBM.750000000W9161.8971.0b.0000000100007562000000000000000
```

Example 2-48 shows the DASD configuration in site B (lnxnode5 and lnxnode6). Devices with addresses 301, 302, 303, and 304 are used for Spectrum Scale. Node lnxnode4 does not have direct access to these devices.

*Example 2-48   DASD configuration for nodes in site B (shown on lnxnode6)*

```
lnxnode6:~ # lsdasd -l
0.0.0202/dasda/94:0
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
  eer_enabled:  0
```

```
  erplog:       0
  uid:          IBM.750000000W9162.8976.16.0000000100007562000000000000000

0.0.0301/dasdb/94:4
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
  eer_enabled:  0
  erplog:       0
  uid:          IBM.750000000W9162.8976.10.0000000100007562000000000000000

0.0.0302/dasdc/94:8
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
  eer_enabled:  0
  erplog:       0
  uid:          IBM.750000000W9162.8976.11.0000000100007562000000000000000

0.0.0303/dasdd/94:12
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
  eer_enabled:  0
  erplog:       0
  uid:          IBM.750000000W9162.8976.12.0000000100007562000000000000000

0.0.0304/dasde/94:16
  status:       active
  type:         ECKD
  blksz:        4096
  size:         21128MB
  blocks:       5409000
  use_diag:     0
  readonly:     0
  eer_enabled:  0
  erplog:       0
  uid:          IBM.750000000W9162.8976.13.0000000100007562000000000000000
```

2. Low-level format the DASDs and create a partition on each DASD. See Example 2-49. This step is needed to prepare the ECKD devices for Spectrum Scale. You have a choice of formatting the DASD using compatible disk layout (cdl) or Linux disk layout (ldl) layout. GPFS supports both layouts.

> **Note:** Low-level formatting destroys any previously existing data on the target DASD.

*Example 2-49 Preparing DASDs for Spectrum Scale*

```
lnxnode1:~ # cat dasd_prepare.sh
for i in b c d e
do
dasdfmt -d cdl -b 4096 /dev/dasd${i}
fdasd -a /dev/dasd${i}
done
```

3. Check the disks, as shown in Example 2-50.

*Example 2-50 Checking prepared disks*

```
lnxnode1:~ # for i in b c d e; do fdasd -ps /dev/dasd${i}; done
          /dev/dasdb1          2   450749   450748    1  Linux native
          /dev/dasdc1          2   450749   450748    1  Linux native
          /dev/dasdd1          2   450749   450748    1  Linux native
          /dev/dasde1          2   450749   450748    1  Linux native
```

4. We prepared the NSD disk stanza file as shown in Example 2-51.

   To balance the access to DASDs, we alternately define two NSD servers for each DASD, one primary and the other secondary.

   For synchronous replication, disks in each site must have defined a distinct failure group.

*Example 2-51 DASD-stanza file*

```
%nsd:
 nsd=site_A_dasd_1
 usage=dataAndMetadata
 failureGroup=1
 servers=lnxnode1,lnxnode2
 device=/dev/dasdb1
 %nsd:
 nsd=site_A_dasd_2
 usage=dataAndMetadata
 failureGroup=1
 servers=lnxnode2,lnxnode1
 device=/dev/dasdc1
 %nsd:
 nsd=site_A_dasd_3
 usage=dataAndMetadata
 failureGroup=1
 servers=lnxnode1,lnxnode2
 device=/dev/dasdd1
 %nsd:
 nsd=site_A_dasd_4
 usage=dataAndMetadata
 failureGroup=1
 servers=lnxnode2,lnxnode1
```

```
device=/dev/dasde1
%nsd:
nsd=site_B_dasd_1
usage=dataAndMetadata
failureGroup=2
servers=lnxnode5,lnxnode6
device=/dev/dasdb1
%nsd:
nsd=site_B_dasd_2
usage=dataAndMetadata
failureGroup=2
servers=lnxnode6,lnxnode5
device=/dev/dasdc1
%nsd:
nsd=site_B_dasd_3
usage=dataAndMetadata
failureGroup=2
servers=lnxnode5,lnxnode6
device=/dev/dasdd1
%nsd:
nsd=site_B_dasd_4
usage=dataAndMetadata
failureGroup=2
servers=lnxnode6,lnxnode5
device=/dev/dasde1
%nsd:
nsd=site_C_disk_2
usage=descOnly
failureGroup=3
servers=node7tie
device=sdc
```

5. We create the NSDs, as shown in Example 2-52.

*Example 2-52 Create NSD disk*

```
lnxnode1:/work # mmcrnsd -F DASD-Disks
mmcrnsd: Processing disk dasdb1
mmcrnsd: Processing disk dasdc1
mmcrnsd: Processing disk dasdd1
mmcrnsd: Processing disk dasde1
mmcrnsd: Processing disk dasdb1
mmcrnsd: Processing disk dasdc1
mmcrnsd: Processing disk dasdd1
mmcrnsd: Processing disk dasde1
mmcrnsd: Processing disk sdc
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

6. Each disk has primary and secondary NSD servers in alternating configuration. Therefore, the NSD disks are balanced between the NSD servers (see Example 2-53).

*Example 2-53 NSD list*

```
lnxnode1:/work # mmlsnsd

 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 (free disk)   site_A_dasd_1 lnxnode1,lnxnode2
 (free disk)   site_A_dasd_2 lnxnode2,lnxnode1
 (free disk)   site_A_dasd_3 lnxnode1,lnxnode2
 (free disk)   site_A_dasd_4 lnxnode2,lnxnode1
 (free disk)   site_B_dasd_1 lnxnode5,lnxnode6
 (free disk)   site_B_dasd_2 lnxnode6,lnxnode5
 (free disk)   site_B_dasd_3 lnxnode5,lnxnode6
 (free disk)   site_B_dasd_4 lnxnode6,lnxnode5
 (free disk)   site_C_disk_2 node7tie
```

### File system creation

Following NSD creation, we define a file system using the available NSDs:

7. We create a file system as shown in Example 2-54 with the following settings:
   – Use the NSDs defined in the previous step.
   – Automount the file system when the cluster is started.
   – Specify two maximum and default number of replicas for both data and metadata (enable default replication for all files). These configurations ensure that any file created in this file system is replicated by default.

*Example 2-54 Creating the file system*

```
lnxnode1:/work # mmcrfs dasd_fs1 -F /work/DASD-Disks  -A yes -m 2 -M 2 -n 10 -r
2 -R 2 -T  /dasd_fs1

The following disks of dasd_fs1 will be formatted on node lnxnode6:
    site_A_dasd_1: size 21128 MB
    site_A_dasd_2: size 21128 MB
    site_A_dasd_3: size 21128 MB
    site_A_dasd_4: size 21128 MB
    site_B_dasd_1: size 21128 MB
    site_B_dasd_2: size 21128 MB
    site_B_dasd_3: size 21128 MB
    site_B_dasd_4: size 21128 MB
    site_C_disk_2: size 152627 MB
Formatting file system ...
Disks up to size 1.5 TB can be added to storage pool system.
Creating Inode File
  64 % complete on Thu Dec  3 16:26:13 2015
 100 % complete on Thu Dec  3 16:26:16 2015
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Completed creation of file system /dev/dasd_fs1.
mmcrfs: Propagating the cluster configuration data to all
```

```
affected nodes.  This is an asynchronous process.
```

8. Mount the file system:

   `mmmount dasd_fs1 -a`

9. List file system disks to confirm that all settings are correct as shown in Example 2-55. Check the following configuration:

   – All disks are in *ready* status.
   – All disks are in *up* availability.
   – The NSD nodes are alternating as defined in the initial NSD file.
   – The file descriptors are one per failure group.

*Example 2-55 File system disk availability checks*

```
lnxnode1:/work # mmlsdisk dasd_fs1
disk         driver   sector     failure holds    holds                              storage
name         type       size       group metadata data  status        availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
site_A_dasd_1 nsd        4096           1 Yes      Yes   ready         up           system
site_A_dasd_2 nsd        4096           1 Yes      Yes   ready         up           system
site_A_dasd_3 nsd        4096           1 Yes      Yes   ready         up           system
site_A_dasd_4 nsd        4096           1 Yes      Yes   ready         up           system
site_B_dasd_1 nsd        4096           2 Yes      Yes   ready         up           system
site_B_dasd_2 nsd        4096           2 Yes      Yes   ready         up           system
site_B_dasd_3 nsd        4096           2 Yes      Yes   ready         up           system
site_B_dasd_4 nsd        4096           2 Yes      Yes   ready         up           system
site_C_disk_2 nsd         512           3 No       No    ready         up           system


lnxnode1:/work # mmlsnsd -m

 Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------------
 site_A_dasd_1 0714090C5660B23B   /dev/dasdb1    lnxnode1                 server node
 site_A_dasd_1 0714090C5660B23B   /dev/dasdb1    lnxnode2                 server node
 site_A_dasd_2 0715090C5660B241   /dev/dasdc1    lnxnode1                 server node
 site_A_dasd_2 0715090C5660B241   /dev/dasdc1    lnxnode2                 server node
 site_A_dasd_3 0714090C5660B242   /dev/dasdd1    lnxnode1                 server node
 site_A_dasd_3 0714090C5660B242   /dev/dasdd1    lnxnode2                 server node
 site_A_dasd_4 0715090C5660B248   /dev/dasde1    lnxnode1                 server node
 site_A_dasd_4 0715090C5660B248   /dev/dasde1    lnxnode2                 server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode5                 server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode6                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode5                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode6                 server node
 site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode5                 server node
 site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode6                 server node
 site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode5                 server node
 site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode6                 server node
 site_C_disk_2 090C071C56609A2F   /dev/sdc       node7tie                 server node

nxnode1:~ # mmlsdisk dasd_fs1 -L
disk         driver   sector     failure holds    holds                              storage
name         type       size       group metadata data  status        availability disk id pool         remarks
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------- ------------ ---------
site_A_dasd_1 nsd        4096           1 Yes      Yes   ready         up                 1 system       desc
site_A_dasd_2 nsd        4096           1 Yes      Yes   ready         up                 2 system
site_A_dasd_3 nsd        4096           1 Yes      Yes   ready         up                 3 system
site_A_dasd_4 nsd        4096           1 Yes      Yes   ready         up                 4 system
site_B_dasd_1 nsd        4096           2 Yes      Yes   ready         up                 5 system       desc
```

```
site_B_dasd_2 nsd        4096          2 Yes      Yes   ready         up               6 system
site_B_dasd_3 nsd        4096          2 Yes      Yes   ready         up               7 system
site_B_dasd_4 nsd        4096          2 Yes      Yes   ready         up               8 system
site_C_disk_2 nsd         512          3 No       No    ready         up               9 system       desc
Number of quorum disks: 3
Read quorum value:      2
Write quorum value:     2
```

## 2.3.2 Testing the cluster

We perform the following tests:

- Test 1: NSD server and cluster manager node failure
- Test 2: Site A: Complete site failure

### Test 1: NSD server and cluster manager node failure

If an NSD server fails and there are at least two NSD servers defined for each disk, the expected behavior is that the devices served by the failed NSD server are automatically moved over to the healthy NSD server as described in the following steps.

#### *Node failure: lnxnode1*

**Test description and expected behavior:** For this test case, we decided to test the worst case scenario: The NSD server that fails is also the cluster manager. Expected behavior is that a new cluster manager is elected before NSD access is recovered.

1. We check the cluster manager as shown in Example 2-56. If the cluster manager is stopped, recovery will take longer because a new cluster manager must be elected.

*Example 2-56 Cluster manager*

```
lnxnode6:~ # mmlsmgr
file system      manager node
---------------- ------------------
dasd_fs1         9.12.7.26 (lnxnode6)

Cluster manager node: 9.12.7.20 (lnxnode1)
```

2. We check the active NSD server for each NSD, as shown in Example 2-57.

*Example 2-57 NSD server before lnxnode1 failure*

```
lnxnode1:~ # mmlsnsd -f dasd_fs1 -m

 Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------------
 site_A_dasd_1 0714090C5660B23B   /dev/dasdb1    lnxnode1                 server node
 site_A_dasd_1 0714090C5660B23B   /dev/dasdb1    lnxnode2                 server node
 site_A_dasd_2 0715090C5660B241   /dev/dasdc1    lnxnode1                 server node
 site_A_dasd_2 0715090C5660B241   /dev/dasdc1    lnxnode2                 server node
 site_A_dasd_3 0714090C5660B242   /dev/dasdd1    lnxnode1                 server node
 site_A_dasd_3 0714090C5660B242   /dev/dasdd1    lnxnode2                 server node
 site_A_dasd_4 0715090C5660B248   /dev/dasde1    lnxnode1                 server node
 site_A_dasd_4 0715090C5660B248   /dev/dasde1    lnxnode2                 server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode5                 server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode6                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode5                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode6                 server node
```

```
site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode5                server node
site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode6                server node
site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode5                server node
site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode6                server node
site_C_disk_2 090C071C56609A2F   /dev/sdc       node7tie                server node
```

3. We disable cluster connection of the *lnxnode1* to simulate a failure. The cluster log content (/var/mmfs/gen/mmfslog) on the failing node is shown in Example 2-58 (excerpt).

   Note the following node statuses:

   – The failing (disconnected) node (*lnxnode1*) is the cluster manager at the time of failure, and it signals that the lease is overdue on the other nodes. When time-out is reached, the nodes are expelled from the cluster.

   – From the cluster manager perspective, the other nodes are failing, so these nodes are expelled one by one until the cluster manager reaches a point where the cluster quorum is not met (50% + 1 of the total number of quorum nodes) and it loses cluster membership. Once *linxnode1* loses cluster membership, the file system is unmounted.

*Example 2-58 Cluster log on lnxnode1 (disconnected node)*

```
Thu Dec  3 16:36:48.497 2015: [N] Node 9.12.7.21 (lnxnode2) lease renewal is overdue. Pinging to
check if it is alive
Thu Dec  3 16:36:49.669 2015: [N] Node 9.12.7.26 (lnxnode6) lease renewal is overdue. Pinging to
check if it is alive
Thu Dec  3 16:36:49.869 2015: [N] Node 9.12.7.25 (lnxnode5) lease renewal is overdue. Pinging to
check if it is alive
Thu Dec  3 16:36:50.169 2015: [N] Node 9.12.7.28 (node7tie) lease renewal is overdue. Pinging to
check if it is alive
Thu Dec  3 16:36:50.259 2015: [N] Node 9.12.7.22 (lnxnode3) lease renewal is overdue. Pinging to
check if it is alive
Thu Dec  3 16:36:50.380 2015: [N] Node 9.12.7.24 (lnxnode4) lease renewal is overdue. Pinging to
check if it is alive
Thu Dec  3 16:38:48.492 2015: Sending request to collect expel debug data to lnxnode2 localNode
Thu Dec  3 16:38:48.493 2015: [I] Calling User Exit Script gpfsSendRequestToNodes: event
sendRequestToNodes, Async command /usr/lpp/mmfs/bin/mmcommon.
Thu Dec  3 16:38:58.518 2015: [E] Node 9.12.7.21 (lnxnode2) is being expelled because of an
expired lease. Pings sent: 216. Replies received: 216.
Thu Dec  3 16:38:58.525 2015: [N] Expel data is not collected on any node. It was collected
recently at 2015-12-03 16:38:48.
Thu Dec  3 16:38:59.125 2015: [N] Expel data is not collected on any node. It was collected
recently at 2015-12-03 16:38:48.
Thu Dec  3 16:38:59.126 2015: [N] Expel data is not collected on any node. It was collected
recently at 2015-12-03 16:38:48.
Thu Dec  3 16:38:59.127 2015: [E] Node 9.12.7.25 (lnxnode5) is being expelled because of an
expired lease. Pings sent: 216. Replies received: 216.
Thu Dec  3 16:38:59.128 2015: [E] Node 9.12.7.26 (lnxnode6) is being expelled because of an
expired lease. Pings sent: 216. Replies received: 216.
Thu Dec  3 16:38:59.129 2015: [E] Not enough quorum nodes reachable: 2.
Thu Dec  3 16:38:59.130 2015: [E] Lost membership in cluster FCP-Cluster.lnxnode1. Unmounting
file systems.
Thu Dec  3 16:38:59.131 2015: [N] Expel data is not collected on any node. It was collected
recently at 2015-12-03 16:38:48.
Thu Dec  3 16:39:09.126 2015: [N] Close connection to 9.12.7.22 lnxnode3 <c0n5> (Node failed)
Thu Dec  3 16:39:09.127 2015: [N] Close connection to 9.12.7.24 lnxnode4 <c0n6> (Node failed)
Thu Dec  3 16:39:09.128 2015: [N] Close connection to 9.12.7.28 node7tie <c0n4> (Node failed)
```

```
Thu Dec  3 16:39:09.148 2015: [I] Quorum loss. Probing cluster FCP-Cluster.lnxnode1
Thu Dec  3 16:39:09.650 2015: [N] Connecting to 9.12.7.21 lnxnode2 <c0p1>
Thu Dec  3 16:39:09.651 2015: [N] Connecting to 9.12.7.25 lnxnode5 <c0p2>
Thu Dec  3 16:39:09.652 2015: [N] Connecting to 9.12.7.26 lnxnode6 <c0p0>
Thu Dec  3 16:39:09.653 2015: [N] Connecting to 9.12.7.28 node7tie <c0p4>
```

**Note:** The lnxnode1 node was disconnected, but the server itself was not stopped in order to collect the logs.

4. The remaining cluster nodes are running properly as shown in Example 2-59. Note the following node statuses:
   - The *lnxnode6* node notices that the lease from *lnxnode1* (cluster manager) is overdue and starts to ping the cluster manager.
   - *lnxnode6* receives no replies and declares the node *lnxnode1* dead.
   - *lnxnode6* assumes the role of cluster manager (as defined in the cluster configuration: lnxnode1 and lnxnode6 are the only two manager nodes).
   - *lnxnode6* rechecks the lease with the other nodes.

*Example 2-59 Cluster logs: Healthy node*

```
Thu Dec  3 16:36:49.526 2015: [N] Node 9.12.7.20 (lnxnode1) lease renewal is overdue. Pinging to check if
it is alive
Thu Dec  3 16:36:52.858 2015: [I] Node 9.12.7.20 (lnxnode1): ping timed out. Pings sent: 6. Replies
received: 0.
Thu Dec  3 16:36:52.859 2015: [I] Lease is overdue. Probing cluster FCP-Cluster.lnxnode1
Thu Dec  3 16:36:54.366 2015: [N] CCR: failed to connect to node 9.12.7.20:1191 (sock 36 err 1143)
Thu Dec  3 16:36:55.452 2015: [N] CCR: failed to connect to node 9.12.7.20:1191 (sock 37 err 1143)
Thu Dec  3 16:37:05.461 2015: [N] CCR: failed to connect to node 9.12.7.20:1191 (sock 37 err 1143)
Thu Dec  3 16:37:05.462 2015: [I] This node got elected. Sequence: 4
Thu Dec  3 16:37:05.463 2015: [D] Election completed. Details: OLLG: 44350206 OLLG delta: 8
Thu Dec  3 16:37:05.464 2015: [E] Node 9.12.7.20 (lnxnode1) is being expelled due to expired lease.
Thu Dec  3 16:37:05.465 2015: [I] Node 9.12.7.26 (lnxnode6) is now the Group Leader.
Thu Dec  3 16:37:05.503 2015: [I] Recovering nodes: 9.12.7.20
Thu Dec  3 16:37:05.504 2015: [N] This node (9.12.7.26 (lnxnode6)) is now Cluster Manager for
FCP-Cluster.lnxnode1.
Thu Dec  3 16:37:05.769 2015: [I] Recovery: dasd_fs1, delay 33 sec. for safe recovery.
Thu Dec  3 16:37:12.128 2015: [N] Node 9.12.7.22 (lnxnode3) lease renewal is overdue. Pinging to check if
it is alive
Thu Dec  3 16:37:12.129 2015: [N] Node 9.12.7.24 (lnxnode4) lease renewal is overdue. Pinging to check if
it is alive
Thu Dec  3 16:37:13.789 2015: [I] Node 9.12.7.22 (lnxnode3): lease request received late. Pings sent: 4.
Maximum pings missed: 0.
Thu Dec  3 16:37:13.967 2015: [I] Node 9.12.7.24 (lnxnode4): lease request received late. Pings sent: 4.
Maximum pings missed: 0.
Thu Dec  3 16:37:38.782 2015: [I] Recovered 1 nodes for file system dasd_fs1.
```

   - The NSD disks are now served only by *lnxnode2* in site A, as shown in Example 2-60.

*Example 2-60 NSD fail over*

```
lnxnode6:~ # mmlsnsd -m

 Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------------
 site_A_dasd_1 0714090C5660B23B   /dev/dasdb1    lnxnode2                 server node
 site_A_dasd_1 0714090C5660B23B   -              lnxnode1                 (not found) server node
 site_A_dasd_2 0715090C5660B241   /dev/dasdc1    lnxnode2                 server node
```

```
 site_A_dasd_2 0715090C5660B241   -              lnxnode1                 (not found) server node
 site_A_dasd_3 0714090C5660B242   /dev/dasdd1    lnxnode2                 server node
 site_A_dasd_3 0714090C5660B242   -              lnxnode1                 (not found) server node
 site_A_dasd_4 0715090C5660B248   /dev/dasde1    lnxnode2                 server node
 site_A_dasd_4 0715090C5660B248   -              lnxnode1                 (not found) server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode5                 server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode6                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode5                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode6                 server node
 site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode5                 server node
 site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode6                 server node
 site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode5                 server node
 site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode6                 server node
 site_C_disk_2 090C071C56609A2F   /dev/sdc       node7tie                 server node

mmlsnsd: The following nodes could not be reached:

lnxnode1
```

5. The *lnxnode3* node, who is an NSD client, notices that the disk served by lnxnode1 is missing (see output for **mmlsdisk dasd_fs1 -M**), but it does not change the I/O path to the backup NSD server (see **mmlsdisk dasd_fs1 -m**), as shown in Example 2-61.

*Example 2-61 I/O path not changed (no file system I/O performed)*

```
lnxnode3:~ # mmlsdisk dasd_fs1 -m

Disk name     IO performed on node     Device            Availability
------------  -----------------------  -----------------  ------------
site_A_dasd_1 lnxnode1                 -                 up
site_A_dasd_2 lnxnode2                 -                 up
site_A_dasd_3 lnxnode1                 -                 up
site_A_dasd_4 lnxnode2                 -                 up
site_B_dasd_1 lnxnode5                 -                 up
site_B_dasd_2 lnxnode6                 -                 up
site_B_dasd_3 lnxnode5                 -                 up
site_B_dasd_4 lnxnode6                 -                 up
site_C_disk_2 node7tie                 -                 up
lnxnode3:~ # mmlsdisk dasd_fs1 -M

Disk name     IO performed on node     Device            Availability
------------  -----------------------  -----------------  ------------
site_A_dasd_1 lnxnode1                 -                 up
site_A_dasd_2 lnxnode2                 /dev/dasdc1       up
site_A_dasd_3 lnxnode1                 -                 up
site_A_dasd_4 lnxnode2                 /dev/dasde1       up
site_B_dasd_1 lnxnode5                 /dev/dasdb1       up
site_B_dasd_2 lnxnode6                 /dev/dasdc1       up
site_B_dasd_3 lnxnode5                 /dev/dasdd1       up
site_B_dasd_4 lnxnode6                 /dev/dasde1       up
site_C_disk_2 node7tie                 /dev/sdc          up
```

6. When I/O requests are created on *lnxnode3*, the I/O paths to the NSD disks are moved to the backup server, as shown in Example 2-62. Now all NSD disks in site A are served by *lnxnode2* because *lnxnode1* is down.

*Example 2-62 I/O requests change the NSD access path*

```
lnxnode3:~ # dd if=/dasd_fs1/site-A-file4 of=/dev/null bs=1M count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB) copied, 0.69005 s, 152 MB/s
lnxnode3:~ # mmlsdisk dasd_fs1 -m

Disk name     IO performed on node     Device            Availability
------------  -----------------------  ----------------- ------------
site_A_dasd_1 lnxnode2                 -                 up
site_A_dasd_2 lnxnode2                 -                 up
site_A_dasd_3 lnxnode2                 -                 up
site_A_dasd_4 lnxnode2                 -                 up
site_B_dasd_1 lnxnode5                 -                 up
site_B_dasd_2 lnxnode6                 -                 up
site_B_dasd_3 lnxnode5                 -                 up
site_B_dasd_4 lnxnode6                 -                 up
site_C_disk_2 node7tie                 -                 up
lnxnode3:~ # mmlsdisk dasd_fs1 -M

Disk name     IO performed on node     Device            Availability
------------  -----------------------  ----------------- ------------
site_A_dasd_1 lnxnode2                 /dev/dasdb1       up
site_A_dasd_2 lnxnode2                 /dev/dasdc1       up
site_A_dasd_3 lnxnode2                 /dev/dasdd1       up
site_A_dasd_4 lnxnode2                 /dev/dasde1       up
site_B_dasd_1 lnxnode5                 /dev/dasdb1       up
site_B_dasd_2 lnxnode6                 /dev/dasdc1       up
site_B_dasd_3 lnxnode5                 /dev/dasdd1       up
site_B_dasd_4 lnxnode6                 /dev/dasde1       up
site_C_disk_2 node7tie                 /dev/sdc          up
```

**Important:** Until the node *lnxnode1* recovers, the cluster remains in this state.

### *Node recovery: lnxnode1*

Node *lnxnode1* rejoins the cluster by reconnecting the network. The cluster logs on *lnxnode1* are shown in Example 2-63:

Note the following node statuses:

- *lnxnode1* connects with all cluster members that are defined as quorum nodes.
- *lnxnode1* acknowledges that the cluster node manager is now *lnxnode6*.

*Example 2-63 Node lnxnode1 rejoins the cluster*

```
Thu Dec  3 17:02:48.678 2015: [I] Connected to 9.12.7.21 lnxnode2 <c0p1>
Thu Dec  3 17:02:48.692 2015: [I] Connected to 9.12.7.26 lnxnode6 <c0p0>
Thu Dec  3 17:02:48.697 2015: [I] Node 9.12.7.26 (lnxnode6) is now the Group Leader.
Thu Dec  3 17:02:48.958 2015: [I] Connected to 9.12.7.25 lnxnode5 <c0n2>
Thu Dec  3 17:02:49.056 2015: [I] Connected to 9.12.7.28 node7tie <c0n4>
Thu Dec  3 17:02:49.605 2015: [N] Remounted dasd_fs1
```

```
Thu Dec  3 17:02:49.621 2015: [N] mmfsd ready
Thu Dec  3 17:02:49 EST 2015: mmcommon mmfsup invoked. Parameters: 9.12.7.20 9.12.7.26 all
```

- The NSD disks can also be accessed by *lnxnode1* as shown in Example 2-64.

*Example 2-64 NSD path recovery*

```
nxnode6:~ # mmlsnsd -m

 Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------------
 site_A_dasd_1 0714090C5660B23B   /dev/dasdb1    lnxnode1                 server node
 site_A_dasd_1 0714090C5660B23B   /dev/dasdb1    lnxnode2                 server node
 site_A_dasd_2 0715090C5660B241   /dev/dasdc1    lnxnode1                 server node
 site_A_dasd_2 0715090C5660B241   /dev/dasdc1    lnxnode2                 server node
 site_A_dasd_3 0714090C5660B242   /dev/dasdd1    lnxnode1                 server node
 site_A_dasd_3 0714090C5660B242   /dev/dasdd1    lnxnode2                 server node
 site_A_dasd_4 0715090C5660B248   /dev/dasde1    lnxnode1                 server node
 site_A_dasd_4 0715090C5660B248   /dev/dasde1    lnxnode2                 server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode5                 server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode6                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode5                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode6                 server node
 site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode5                 server node
 site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode6                 server node
 site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode5                 server node
 site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode6                 server node
 site_C_disk_2 090C071C56609A2F   /dev/sdc       node7tie                 server node
```

On *lnxnode3* (NSD client), the missing paths (through *lnxnode1*) are still not updated. However, as soon as there are I/O requests on the file system, the paths are updated automatically, as shown in Example 2-65.

*Example 2-65 NSD path change after I/O*

```
lnxnode3:~ # mmlsdisk dasd_fs1 -m

Disk name     IO performed on node     Device            Availability
------------  -----------------------  -----------------  ------------
site_A_dasd_1 lnxnode2                 -                  up
site_A_dasd_2 lnxnode2                 -                  up
site_A_dasd_3 lnxnode2                 -                  up
site_A_dasd_4 lnxnode2                 -                  up
site_B_dasd_1 lnxnode5                 -                  up
site_B_dasd_2 lnxnode6                 -                  up
site_B_dasd_3 lnxnode5                 -                  up
site_B_dasd_4 lnxnode6                 -                  up
site_C_disk_2 node7tie                 -                  up
lnxnode3:~ # dd if=/dasd_fs1/site-A-file4 of=/dev/null bs=1M count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB) copied, 1.29444 s, 81.0 MB/s
lnxnode3:~ # mmlsdisk dasd_fs1 -m

Disk name     IO performed on node     Device            Availability
------------  -----------------------  -----------------  ------------
```

```
site_A_dasd_1 lnxnode1                 -                up
site_A_dasd_2 lnxnode2                 -                up
site_A_dasd_3 lnxnode1                 -                up
site_A_dasd_4 lnxnode2                 -                up
site_B_dasd_1 lnxnode5                 -                up
site_B_dasd_2 lnxnode6                 -                up
site_B_dasd_3 lnxnode5                 -                up
site_B_dasd_4 lnxnode6                 -                up
site_C_disk_2 node7tie                 -                up
```

### Test 2: Site A: Complete site failure

In case an entire site fails, the secondary site is still operational. The synchronous replication stops and it resumes when site A recovers. The following steps simulate an entire site failure:

1. Before site A fails the cluster status (healthy) is shown in Example 2-66. All files are replicated and all nodes are active.

*Example 2-66 Healthy cluster*

```
nxnode6:~ # mmlsattr /dasd_fs1/*
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- ---------------
      2 (  2)   2 (  2) /dasd_fs1/site-A-file1
      2 (  2)   2 (  2) /dasd_fs1/site-A-file2
      2 (  2)   2 (  2) /dasd_fs1/site-A-file3
      2 (  2)   2 (  2) /dasd_fs1/site-A-file4


lnxnode6:~ # mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      lnxnode1         active
       2      lnxnode2         active
       3      lnxnode5         active
       4      lnxnode6         active
       5      node7tie         active
       6      lnxnode3         active
       8      lnxnode4         active
```

2. We simulate site A failure by “powering off” all nodes in this site.
3. The cluster log entries on the *lnxnode6* node (site B) are shown in Example 2-67.

   Note the following node and disk statuses:

   – Nodes *lnxnode1*, *lnxnode2,* and *lnxnode3* have a lease overdue and are expelled from the cluster.

   – The disks in site A are marked as *failed* on all remaining nodes.

*Example 2-67 Site B cluster node logs*

```
Thu Dec  3 17:08:47.409 2015: [N] Node 9.12.7.20 (lnxnode1) lease renewal is overdue. Pinging to check if
it is alive
Thu Dec  3 17:08:50.740 2015: [E] Node 9.12.7.20 (lnxnode1) is being expelled because of an expired lease.
Pings sent: 6. Replies received: 0.
Thu Dec  3 17:08:50.743 2015: [I] Recovering nodes: 9.12.7.20
```

```
Thu Dec  3 17:08:50.942 2015: [I] Recovery: dasd_fs1, delay 31 sec. for safe recovery.
Thu Dec  3 17:08:56.911 2015: [N] Node 9.12.7.21 (lnxnode2) lease renewal is overdue. Pinging to check if
it is alive
Thu Dec  3 17:09:00.232 2015: [E] Node 9.12.7.21 (lnxnode2) is being expelled because of an expired lease.
Pings sent: 6. Replies received: 0.
Thu Dec  3 17:09:02.582 2015: [N] Node 9.12.7.22 (lnxnode3) lease renewal is overdue. Pinging to check if
it is alive
Thu Dec  3 17:09:05.913 2015: [E] Node 9.12.7.22 (lnxnode3) is being expelled because of an expired lease.
Pings sent: 6. Replies received: 0.
Thu Dec  3 17:09:06.095 2015: [I] Recovery: dasd_fs1, delay 31 sec. for safe recovery.
Thu Dec  3 17:09:21.943 2015: [E] Disk failure.  Volume dasd_fs1. rc = 5. Physical volume site_A_dasd_2.
Thu Dec  3 17:09:21.953 2015: [E] Disk failure.  Volume dasd_fs1. rc = 5. Physical volume site_A_dasd_3.
Thu Dec  3 17:09:21.954 2015: [E] Disk failure.  Volume dasd_fs1. rc = 5. Physical volume site_A_dasd_4.
Thu Dec  3 17:09:21.955 2015: [E] Disk failure.  Volume dasd_fs1. rc = 5. Physical volume site_A_dasd_1.
Thu Dec  3 17:09:21.991 2015: [E] Disk failure from node 9.12.7.24 (lnxnode4) Volume dasd_fs1. Physical
volume site_A_dasd_4.
Thu Dec  3 17:09:21.992 2015: [E] Disk failure from node 9.12.7.25 (lnxnode5) Volume dasd_fs1. Physical
volume site_A_dasd_1.
Thu Dec  3 17:09:21.993 2015: [E] Disk failure from node 9.12.7.25 (lnxnode5) Volume dasd_fs1. Physical
volume site_A_dasd_3.
Thu Dec  3 17:09:21.996 2015: [E] Disk failure from node 9.12.7.28 (node7tie) Volume dasd_fs1. Physical
volume site_A_dasd_1.
Thu Dec  3 17:09:21.998 2015: [E] Disk failure from node 9.12.7.24 (lnxnode4) Volume dasd_fs1. Physical
volume site_A_dasd_3.
Thu Dec  3 17:09:37.276 2015: [I] Recovered 1 nodes for file system dasd_fs1.
Thu Dec  3 17:09:37.277 2015: [I] Recovering nodes: 9.12.7.20 9.12.7.21 9.12.7.22
Thu Dec  3 17:09:37.440 2015: [I] Recovered 3 nodes for file system dasd_fs1.
```

4. The cluster status is shown in Example 2-68. The nodes on site A are in an *unknown* state because there is no information about the status (no lease requests, no ping replies).

*Example 2-68 Cluster status*

```
lnxnode6:~ # mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      lnxnode1         unknown
       2      lnxnode2         unknown
       3      lnxnode5         active
       4      lnxnode6         active
       5      node7tie         active
       6      lnxnode3         unknown
       8      lnxnode4         active
```

5. We create new files on the file system. Because site A is missing, the disks in site A are unavailable. Therefore, the new files are not replicated (see Example 2-69).

*Example 2-69 File attributes*

```
lnxnode6:~ # mmlsattr /dasd_fs1/*
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- ---------------
      2 (  2)   2 (  2) /dasd_fs1/site-A-file1
      2 (  2)   2 (  2) /dasd_fs1/site-A-file2
      2 (  2)   2 (  2) /dasd_fs1/site-A-file3
      2 (  2)   2 (  2) /dasd_fs1/site-A-file4
      2 (  2)   2 (  2) /dasd_fs1/site-B-file1   [dataupdatemiss,metaupdatemiss]
```

```
     2 (  2)   2 (  2) /dasd_fs1/site-B-file2   [dataupdatemiss,metaupdatemiss]
     2 (  2)   2 (  2) /dasd_fs1/site-B-file3   [dataupdatemiss,metaupdatemiss]
     2 (  2)   2 (  2) /dasd_fs1/site-B-file4   [dataupdatemiss,metaupdatemiss]
```

**Important:** The cluster remains in this stage until the nodes in site A rejoin the cluster.

6. We start the nodes in site A. The nodes rejoin the cluster as shown in Example 2-70.

*Example 2-70 Cluster nodes after site A recovery*

```
lnxnode6:~ # mmgetstate -a

 Node number  Node name       GPFS state
------------------------------------------
       1      lnxnode1        active
       2      lnxnode2        active
       3      lnxnode5        active
       4      lnxnode6        active
       5      node7tie        active
       6      lnxnode3        active
       8      lnxnode4        active
```

7. The disks in site A are down until they are manually started, as shown in Example 2-71.

*Example 2-71 Site A disks*

```
lnxnode6:~ # mmlsdisk dasd_fs1
disk         driver   sector     failure holds    holds                            storage
name         type       size       group metadata data  status        availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
site_A_dasd_1 nsd       4096           1 Yes      Yes   ready         down         system
site_A_dasd_2 nsd       4096           1 Yes      Yes   ready         down         system
site_A_dasd_3 nsd       4096           1 Yes      Yes   ready         down         system
site_A_dasd_4 nsd       4096           1 Yes      Yes   ready         down         system
site_B_dasd_1 nsd       4096           2 Yes      Yes   ready         up           system
site_B_dasd_2 nsd       4096           2 Yes      Yes   ready         up           system
site_B_dasd_3 nsd       4096           2 Yes      Yes   ready         up           system
site_B_dasd_4 nsd       4096           2 Yes      Yes   ready         up           system
site_C_disk_2 nsd        512           3 No       No    ready         up           system
```

8. We start the disks as shown in Example 2-72. Note that when the disks are rediscovered, the files replicated automatically.

*Example 2-72 Site A recover disks*

```
lnxnode6:~ # mmchdisk dasd_fs1 start -a
mmnsddiscover:  Attempting to rediscover the disks.  This may take a while ...
mmnsddiscover:  Finished.
lnxnode5:  Rediscovered nsd server access to site_B_dasd_1.
lnxnode5:  Rediscovered nsd server access to site_B_dasd_2.
lnxnode2:  Rediscovered nsd server access to site_A_dasd_1.
lnxnode2:  Rediscovered nsd server access to site_A_dasd_2.
lnxnode1:  Rediscovered nsd server access to site_A_dasd_1.
lnxnode1:  Rediscovered nsd server access to site_A_dasd_2.
lnxnode6:  Rediscovered nsd server access to site_B_dasd_1.
lnxnode6:  Rediscovered nsd server access to site_B_dasd_2.
node7tie:  Rediscovered nsd server access to site_C_disk_2.
```

```
lnxnode5:  Rediscovered nsd server access to site_B_dasd_3.
lnxnode5:  Rediscovered nsd server access to site_B_dasd_4.
lnxnode2:  Rediscovered nsd server access to site_A_dasd_3.
lnxnode2:  Rediscovered nsd server access to site_A_dasd_4.
lnxnode1:  Rediscovered nsd server access to site_A_dasd_3.
lnxnode1:  Rediscovered nsd server access to site_A_dasd_4.
lnxnode6:  Rediscovered nsd server access to site_B_dasd_3.
lnxnode6:  Rediscovered nsd server access to site_B_dasd_4.
Scanning file system metadata, phase 1 ...
  64 % complete on Thu Dec  3 17:42:46 2015
 100 % complete on Thu Dec  3 17:42:48 2015
Scan completed successfully.
Scanning file system metadata, phase 2 ...
Scan completed successfully.
Scanning file system metadata, phase 3 ...
Scan completed successfully.
Scanning file system metadata, phase 4 ...
Scan completed successfully.
Scanning user file metadata ...
   4.18 % complete on Thu Dec  3 17:43:11 2015  (    322048 inodes with total       6847 MB data
processed)
 100.00 % complete on Thu Dec  3 17:43:11 2015  (    322048 inodes with total       6847 MB data
processed)
Scan completed successfully.
```

9. The files and disk status are shown in Example 2-73:
   – There are no files with the [dataupdatemiss,metaupdatemiss] attribute.
   – The NSDs in site A are up and running.

*Example 2-73 Site A files and disk status*

```
lnxnode1:~ # mmlsattr /dasd_fs1/*
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- ---------------
      2 (  2)   2 (  2) /dasd_fs1/site-A-file1
      2 (  2)   2 (  2) /dasd_fs1/site-A-file2
      2 (  2)   2 (  2) /dasd_fs1/site-A-file3
      2 (  2)   2 (  2) /dasd_fs1/site-A-file4
      2 (  2)   2 (  2) /dasd_fs1/site-B-file1
      2 (  2)   2 (  2) /dasd_fs1/site-B-file2
      2 (  2)   2 (  2) /dasd_fs1/site-B-file3
      2 (  2)   2 (  2) /dasd_fs1/site-B-file4



lnxnode6:~ # mmlsnsd -m

 Disk name    NSD volume ID      Device         Node name                Remarks
---------------------------------------------------------------------------------------
 site_A_dasd_1 0714090C5660B23B   /dev/dasdb1    lnxnode1                 server node
 site_A_dasd_1 0714090C5660B23B   /dev/dasdb1    lnxnode2                 server node
 site_A_dasd_2 0715090C5660B241   /dev/dasdc1    lnxnode1                 server node
 site_A_dasd_2 0715090C5660B241   /dev/dasdc1    lnxnode2                 server node
 site_A_dasd_3 0714090C5660B242   /dev/dasdd1    lnxnode1                 server node
 site_A_dasd_3 0714090C5660B242   /dev/dasdd1    lnxnode2                 server node
```

```
 site_A_dasd_4 0715090C5660B248   /dev/dasde1    lnxnode1                 server node
 site_A_dasd_4 0715090C5660B248   /dev/dasde1    lnxnode2                 server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode5                 server node
 site_B_dasd_1 0719090C5660B24B   /dev/dasdb1    lnxnode6                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode5                 server node
 site_B_dasd_2 071A090C5660B252   /dev/dasdc1    lnxnode6                 server node
 site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode5                 server node
 site_B_dasd_3 0719090C5660B259   /dev/dasdd1    lnxnode6                 server node
 site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode5                 server node
 site_B_dasd_4 071A090C5660B260   /dev/dasde1    lnxnode6                 server node
 site_C_disk_2 090C071C56609A2F   /dev/sdc       node7tie                 server node

lnxnode6:~ # mmlsdisk dasd_fs1
disk         driver   sector     failure holds    holds                            storage
name         type       size       group metadata data  status        availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
site_A_dasd_1 nsd        4096           1 Yes      Yes   ready         up           system
site_A_dasd_2 nsd        4096           1 Yes      Yes   ready         up           system
site_A_dasd_3 nsd        4096           1 Yes      Yes   ready         up           system
site_A_dasd_4 nsd        4096           1 Yes      Yes   ready         up           system
site_B_dasd_1 nsd        4096           2 Yes      Yes   ready         up           system
site_B_dasd_2 nsd        4096           2 Yes      Yes   ready         up           system
site_B_dasd_3 nsd        4096           2 Yes      Yes   ready         up           system
site_B_dasd_4 nsd        4096           2 Yes      Yes   ready         up           system
site_C_disk_2 nsd         512           3 No       No    ready         up           system
lnxnode6:~ #
```

# 3

# Common administrative tasks

In this chapter, we describe some of the common administrative tasks that can be performed when deploying Spectrum Scale for Linux on z Systems:

- Backing up data: Using the mmbackup utility
- Striping and I/O balancing considerations
- Configuring devices on Linux on z (post-installation)

## 3.1 Backing up data: Using the mmbackup utility

This section describes how to use IBM Spectrum Protect (formerly IBM Tivoli® Storage Manager) to back up data from a cluster file system (IBM Spectrum Scale).

In our environment, we use Spectrum Scale (GPFS) Version 4.2.0 for Linux on z Systems. GPFS Version 4.2 now supports backing up data to a Tivoli Storage Manager server using the **mmbackup** utility. The **mmbackup** command can be used to back up some or all of the files of a Spectrum Scale file system to IBM Spectrum Protect (Tivoli Storage Manager) servers using the Tivoli Storage Manager backup-archive client.

The **mmbackup** command uses all the scalable, parallel processing capabilities of the **mmapplypolicy** command to determine which files need to be sent to backup in Tivoli Storage Manager, as well as which deleted files should be expired from Tivoli Storage Manager.

For more information about using the **mmbackup** utility, see the following site:

http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/com.ibm.spectrum.scale.v4r2.adm.doc/bl1adm_backupusingmmbackup.htm?lang=en

The Disaster Recovery (DR) solution that we implement and test (described in details in 1.3, "Cross site cluster: Synchronous mirroring" on page 6) protects against site failure and is based on synchronous mirroring using GPFS replication. For automation of the failover process, we use Spectrum Scale capabilities (cluster and file system quorum mechanisms) with two sites (A and B) providing application access to data. The third site (C) is used to fulfill the quorum requirements in order to maintain access to the file system without manual intervention, if one site (A or B) fails.

**Important: Design and sizing:** The backup solution we describe in this section is *for exemplification purposes* (as a concept). We do not provide sizing procedures because it is out of the scope of this document. If you intend to use parts of this solution (concept), proper design and sizing should be used to customize and to adapt it to your requirements.

The following sizing elements should be considered:

- Tivoli Storage Manager server sizing (CPU, memory, I/O bandwidth)
- Network bandwidth between sites (A<->C, A<->B and B<->C)
- Tivoli Storage Manager storage pools size and performance
- Backup elements (which data to back up) and timing (backup operations window)
- Recovery procedures

The cluster node in site C (tiebreaker) is not used for file system I/O. However, this node has access to the file system. Because this node is in a separate site, it could be used for saving (backup, archive) the data stored in the file system to a backup device (such as disk or tape) in site C (different location than sites A and B).

The backup (Tivoli Storage Manager) server can be a member in the Spectrum Scale cluster (as a node). By being a member in the Spectrum Scale cluster, the Tivoli Storage Manager server has direct access to the cluster file system. If the Tivoli Storage Manager server (with its attached storage devices) is in site C, it provides off-site backup (vaulting). Another option that we have in this scenario is to use the Tivoli Storage Manager server in site C for both functions: Tivoli Storage Manager backup server and Spectrum Scale tiebreaker node, thus reducing the hardware complexity of the environment while allowing the Tivoli Storage Manager server to participate in the IBM Spectrum Protect cluster.

In our environment, we cover two scenarios for backing up Spectrum Scale file system using IBM Spectrum Protect:

1. Spectrum Scale nodes in site A and B can send backup data using IBM Spectrum Protect (Tivoli Storage Manager) backup-archive (BA) client installed on each node. If multiple nodes are configured with Tivoli Storage Manager backup client, backup can be performed on multiple nodes (parallel backup), thus increasing the data throughput and reducing the backup window time.
2. Nodes running the TSM BA client can send data to any external TSM server. However, if the IBM Spectrum Protect (Tivoli Storage Manager) server is part of the Spectrum Scale cluster (cluster node) and also acts as a tiebreaker, backup can be performed directly to the server (on the same node), thus reducing the solution complexity. In such a configuration proper design must be exercised for solution sizing (network and I/O bandwidth, as well as backup operations timing).

> **Note:** In this document, we do not cover Tivoli Storage Manager server installation and configuration. We assume that the Tivoli Storage Manager server is already up and available. The Tivoli Storage Manager BA client software is a prerequisite for **`mmbackup`** and must be installed on all nodes participating in the backup operation.

## 3.1.1 Basic configuration for backup

In this section, we describe the basic procedure for backing up data. This section covers the following topics:

- Tivoli Storage Manager BA client setup
- Node registration and backup server basic configuration
- Backup tests: Single-node and multi-node

### Tivoli Storage Manager BA client configuration

The Tivoli Storage Manager BA client configuration is described in the following steps:

> **Important:** For Tivoli Storage Manager BA client-supported versions with Spectrum Scale V4.2, refer to:
>
> http://www.ibm.com/support/docview.wss?&uid=swg21066436#Version%207.1

1. Verify that the Tivoli Storage Manager BA client software (supported version) is installed on all nodes (that will be used to back up data), as shown in Example 3-1. The same Tivoli Storage Manager BA client version must be installed on all nodes regardless if multi-node backup (parallel backup from multiple nodes) is used or not.

   > **Restriction:** Parallel backup from multiple nodes (**`mmbackup -N`**) is only supported from the nodes with the same architecture and software version (OS, Tivoli Storage Manager BA client, Spectrum Scale).

   *Example 3-1 Backup/archive client packages*

```
node7tie:~ # WCOLL=/dasd_fs1/spectrum.nodes  mmdsh "rpm -qa | grep TIV "| sort
lnxnode1:  TIVsm-API64-7.1.4-0.s390x
lnxnode1:  TIVsm-BA-7.1.4-0.s390x
lnxnode2:  TIVsm-API64-7.1.4-0.s390x
lnxnode2:  TIVsm-BA-7.1.4-0.s390x
lnxnode3:  TIVsm-API64-7.1.4-0.s390x
```

```
lnxnode3:  TIVsm-BA-7.1.4-0.s390x
lnxnode4:  TIVsm-API64-7.1.4-0.s390x
lnxnode4:  TIVsm-BA-7.1.4-0.s390x
lnxnode5:  TIVsm-API64-7.1.4-0.s390x
lnxnode5:  TIVsm-BA-7.1.4-0.s390x
lnxnode6:  TIVsm-API64-7.1.4-0.s390x
lnxnode6:  TIVsm-BA-7.1.4-0.s390x
node7tie:  TIVsm-API64-7.1.4-0.x86_64
node7tie:  TIVsm-BA-7.1.4-0.x86_64
```

2. Verify that the client settings are correct on each cluster node that backs up data. The settings for *lnxnode1* are shown in Example 3-2 and in Example 3-3, with the following specific information:
   - *NODENAME* is the host name for each cluster node
   - *ASNODENAME* is a common node name for all Spectrum Scale cluster nodes performing backup because any cluster node can back up the same data from the cluster file system and perform restore to the file system on behalf of the same node, named *proxynode*. This is the Tivoli Storage Manager node that holds the client backup data.

*Example 3-2 /opt/tivoli/tsm/client/ba/bin/dsm.sys*

```
lnxnode1:~ # cat /opt/tivoli/tsm/client/ba/bin/dsm.sys
SErvername  TSMGPFS
   COMMMethod        TCPip
   TCPPort           1500
   TCPServeraddress   gpfstsm
   txnb 2g
   PASSWORDACCESS GENERATE
   NODENAME lnxnode1
   ASNODENAME gpfs_proxy
```

We also use the client option file (`dsm.opt`) configured as shown in Example 3-3.

*Example 3-3 /opt/tivoli/tsm/client/ba/bin/dsm.opt*

```
Servername TSMGPFS
```

For more information about using the Tivoli Storage Manager BA client with Spectrum Scale, see the manual *IBM Spectrum Scale V4.2: Administration and Programming Reference,* SA23-1452-03.

### Registering the nodes in the Tivoli Storage Manager server

Use the following steps to register the nodes in the Tivoli Storage Manager server:

3. We define the Spectrum Scale nodes to the backup server. On the IBM Spectrum Protect server, the nodes' definition is shown in Example 3-4 on page 65.

   We use the following command on the TSM server to register the cluster nodes as TSM clients (repeat for each node):

   – `register node <NODENAME> <password> domain=standard maxnummp=10`

   Register the proxy node that acts on behalf of the cluster nodes:

   – `register node gpfs_proxy <password> domain=standard maxnummp=10`

   Associate the cluster nodes with the proxy node:

   – `grant proxy target=gpfs_proxy ag=lnxnode1,lnxnode2,...`

*Example 3-4 Node definition (Tivoli Storage Manager server)*

```
tsm: TSMGPFS>q node

Node Name                 Platform Policy Domain  Days Sinc- Days Sinc- Locked?
                                   Name             e Last  e Passwor-
                                                     Access      d Set
------------------------- -------- -------------- ---------- ---------- -------
GPFSTSM                   Linux    STANDARD                6         29   No
                           x86-64
GPFS_PROXY                LinuxZ64 STANDARD               <1         29   No
LNXNODE1                  LinuxZ64 STANDARD                6         29   No
LNXNODE2                  LinuxZ64 STANDARD                6         29   No
LNXNODE3                  LinuxZ64 STANDARD               18         20   No
LNXNODE4                  LinuxZ64 STANDARD               20         20   No
LNXNODE5                  LinuxZ64 STANDARD                6         29   No
LNXNODE6                  LinuxZ64 STANDARD                6         29   No
NODE7TIE                  Linux    STANDARD               <1         <1   No
                           x86-64
```

4. Each node that sends data to the Tivoli Storage Manager server must be defined in the Tivoli Storage Manager server. Because we are backing up a clustered file system, also a proxy node must be defined. The *GPFS_PROXY* node is defined to act as a virtual node performing operations on behalf of the cluster nodes (agent nodes) for the Spectrum Scale file system.

   Example 3-5 shows the definition of the proxy node and the associated agent nodes.

   *Example 3-5 Proxynode definition*

```
tsm: TSMGPFS>q node GPFS_PROXY f=d
                           Node Name: GPFS_PROXY
                            Platform: LinuxZ64

...<< Snippet >>...

                           Node Type: Client

...<< Snippet >>...

                    Proxynode Target:
                     Proxynode Agent: GPFSTSM LNXNODE1 LNXNODE4 LNXNODE2
                                       LNXNODE5 LNXNODE6 LNXNODE3 NODE7TIE

...<< Snippet >>...
```

5. We verify the access from a client node to the Tivoli Storage Manager server by using the **dsmc** command (Tivoli Storage Manager BA client command-line interface (CLI)). See Example 3-6.

*Example 3-6 Backup client CLI*

```
lnxnode1:~ # dsmc
IBM Tivoli Storage Manager
Command Line Backup-Archive Client Interface
  Client Version 7, Release 1, Level 4.0
  Client date/time: 12/22/15   12:48:12
(c) Copyright by IBM Corporation and other(s) 1990, 2015. All Rights Reserved.

Node Name: LNXNODE1
Session established with server TSMGPFS: Linux/x86_64
  Server Version 7, Release 1, Level 4.0
  Server date/time: 12/22/15   12:47:16  Last access: 12/22/15   12:46:08

Accessing as node: GPFS_PROXY
tsm>
```

**Note:** Tivoli Storage Manager BA client on the nodes used for backup requires access to the Tivoli Storage Manager server without prompting for a password. Option PASSWORDACCESS must be set to GENERATE in the system option file (`dsm.sys`).

### Backing up data: Single node

We initiate the backup from one node by using the **mmbackup** command on the `/dasd_fs1` file system, as shown in Example 3-7.

**Attention:** The **mmbackup** command can be applied to a Spectrum Scale file system or to an independent file set. Do not use a subdirectory other than the root path to the mentioned object types because this can lead to inconsistent backups.

*Example 3-7 mmbackup command (basic usage)*

```
node7tie:/opt/tivoli/tsm/client/ba/bin # mmbackup /dasd_fs1/
--------------------------------------------------------
mmbackup: Backup of /dasd_fs1 begins at Wed Dec 16 10:29:40 EST 2015.
--------------------------------------------------------
Wed Dec 16 10:29:42 2015 mmbackup:Scanning file system dasd_fs1
Wed Dec 16 10:29:44 2015 mmbackup:Determining file system changes for dasd_fs1
[TSMGPFS].
Wed Dec 16 10:29:44 2015 mmbackup:changed=5990, expired=0, unsupported=0 for
server [TSMGPFS]
Wed Dec 16 10:29:44 2015 mmbackup:Sending files to the TSM server [5990 changed, 0
expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      5990
        Total number of objects backed up:      5990
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
        Total number of objects failed:         0
        Total number of objects encrypted:      0
```

```
        Total number of bytes inspected:        13464722472
        Total number of bytes transferred:      13464722472
----------------------------------------------------------
mmbackup: Backup of /dasd_fs1 completed successfully at Wed Dec 16 10:37:43 EST
2015.
```

On the backup server, we observe two sessions: One for control/metadata (session 37) and one for data transfer (session 38), as shown in Example 3-8.

*Example 3-8 Backup server sessions*

```
tsm: TSMGPFS>q ses

 Sess   Comm.   Sess     Wait    Bytes    Bytes   Sess     Platform   Client Name
Number  Method  State    Time     Sent    Recvd   Type
------  ------  ------  ------  -------  -------  -------  --------   ------------------
    21  Tcp/Ip  Run       0 S     4.0 K      660  Admin    Linux      ADMIN
                                                            x86-64
    37  Tcp/Ip  IdleW   1.6 M   247.9 K  109.6 K  Node     LinuxZ64   GPFS_PROXY
                                                                       (NODE7TIE)
    38  Tcp/Ip  RecvW     0 S       844    4.6 G  Node     LinuxZ64   GPFS_PROXY
                                                                       (NODE7TIE)
```

### Multi-node backup

You can run the **mmbackup** command with specific parameters to use Spectrum Scale parallelism. Note the following options for multi-threaded backup:

- **-N**: Specifies the list of nodes that run parallel instances of the backup process. When running parallel backup sessions from multiple nodes, more nodes send data to the backup server, which can increase data transfer speed. In the scenario where the backup server is in site C, both communication lines from site A and from site B can be used at the same time.
- **--backup-threads**: Specifies the number of worker threads permitted on each node to perform the backup operation. This is useful when backing up file systems with many files.

When the **mmbackup** command is started with **-N** or **--backup-threads** parameters, the files to be backed up are split in chunks and distributed to multiple **dsmc** processes running inside a node (**--backup-threads**) and on multiple nodes (**-N**), based on the following formula:

```
(number of nodes) x (number of threads) x 100
```

If the number of files to be backed up is lower than the result of the formula, less backup sessions (threads) are created. For example, if the number of nodes is four and the number of threads per node is also four, the minimum number of files that are needed to fully use the number of threads is 1600 (4 x 4 x 100).

> **Important:** The minimum number of files that can be specified per chunk can be set by using the **--max-backup-count** parameter with the range 100 - 8192. The value cannot be under (the default) 100 files per chunk.

1. For example, we create 1600 files in our cluster file system by using the script shown in Example 3-9.

*Example 3-9 Sample script used to create files*

```
for i in `seq -w 1 1600`
> do
> echo Create file file-backup-$i
> dd if=/dev/zero of=/dasd_fs1/file-backup-$i bs=2M count=1
> done
```

2. We initiate multi-threaded backup as shown in Example 3-10.

*Example 3-10 Multi-threaded mmbackup*

```
lnxnode1:~ # mmbackup dasd_fs1 -N lnxnode1,lnxnode2,lnxnode5,lnxnode6
--backup-threads=4  -t full
--------------------------------------------------------
mmbackup: Backup of /dasd_fs1 begins at Wed Dec 16 17:00:51 EST 2015.
--------------------------------------------------------
Wed Dec 16 17:00:55 2015 mmbackup:Scanning file system dasd_fs1
Wed Dec 16 17:00:56 2015 mmbackup:Determining file system changes for dasd_fs1
[TSMGPFS].
Wed Dec 16 17:00:57 2015 mmbackup:changed=5979, expired=0, unsupported=0 for
server [TSMGPFS]
Wed Dec 16 17:00:57 2015 mmbackup:Sending files to the TSM server [5979
changed, 0 expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      5979
        Total number of objects backed up:      5979
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
        Total number of objects failed:         0
        Total number of objects encrypted:      0
        Total number of bytes inspected:        6049622911
        Total number of bytes transferred:      6047829846
----------------------------------------------------------
mmbackup: Backup of /dasd_fs1 completed successfully at Wed Dec 16 17:04:42 EST
2015.
----------------------------------------------------------
```

On the backup server, we observe that multiple sessions are created for each node, as shown in Example 3-11.

*Example 3-11 Backup server sessions*

```
lnxnode1:~ # dsmadmc -id=admin -password=****** q ses
IBM Tivoli Storage Manager
Command Line Administrative Interface - Version 7, Release 1, Level 4.0
(c) Copyright by IBM Corporation and other(s) 1990, 2015. All Rights Reserved.

Session established with server TSMGPFS: Linux/x86_64
  Server Version 7, Release 1, Level 4.0
  Server date/time: 12/16/2015 14:50:58  Last access: 12/16/2015 14:46:11
```

```
ANS8000I Server command: 'q ses'.

 Sess   Comm.    Sess     Wait    Bytes    Bytes    Sess      Platform   Client Name
Number  Method   State     Time     Sent    Recvd   Type
------  ------   ------  ------  -------  -------  -------   --------   -------------------
   395  Tcp/Ip   IdleW      1 S    3.4 K    5.8 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE1)
   396  Tcp/Ip   IdleW      1 S    3.7 K   14.0 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE1)
   397  Tcp/Ip   IdleW      1 S    3.5 K    7.8 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE1)
   398  Tcp/Ip   IdleW      1 S    4.0 K   20.3 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE1)
   405  Tcp/Ip   Run        0 S    3.6 K   15.7 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE6)
   406  Tcp/Ip   IdleW      0 S    3.7 K   14.5 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE5)
   407  Tcp/Ip   IdleW      0 S    3.6 K   12.6 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE6)
   408  Tcp/Ip   IdleW      1 S    3.2 K    2.6 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE6)
   409  Tcp/Ip   IdleW      1 S    3.2 K    2.6 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE6)
   410  Tcp/Ip   Run        0 S    3.6 K   16.3 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE5)
   411  Tcp/Ip   Run        0 S    3.4 K   10.2 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE5)
   412  Tcp/Ip   Run        0 S      766   21.9 M  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE1)
   413  Tcp/Ip   IdleW      0 S    2.9 K    4.1 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE5)
   414  Tcp/Ip   IdleW      0 S    2.4 K    2.1 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE2)
   415  Tcp/Ip   Run        0 S    1.9 K    5.5 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE2)
   416  Tcp/Ip   IdleW      0 S    2.3 K    7.0 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE2)
   417  Tcp/Ip   IdleW      0 S    3.1 K    3.5 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE2)
   418  Tcp/Ip   Run        0 S      766    8.7 M  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE1)
   419  Tcp/Ip   Run        0 S      766  512.6 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE6)
   420  Tcp/Ip   Run        0 S      766    1.0 M  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE6)
   421  Tcp/Ip   Run        0 S      766  514.0 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE1)
   422  Tcp/Ip   MediaW     0 S      761    2.1 K  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE1)
   423  Tcp/Ip   Run        0 S      162      236  Admin     LinuxZ64   ADMIN
   424  Tcp/Ip   Run        0 S      761      866  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE2)
   425  Tcp/Ip   Run        0 S      761      855  Node      LinuxZ64   GPFS_PROXY
                                                                         (LNXNODE6)
   427  Tcp/Ip   Run        0 S      761      866  Node      LinuxZ64   GPFS_PROXY
```

```
                                                    (LNXNODE5)

ANS8002I Highest return code was 0.
```

#### *Considerations for running multi-threaded backups*

In order to run **mmbackup** properly, take into account the following considerations:

- The backup client negotiates how the files are distributed between the cluster nodes specified to the **mmbackup** command. As the files' size differs, the amount of data sent by each node to the backup server differs, as such the duration of each session will not be the same (not all nodes finish at once).
- For each backup session that **mmbackup** generates, the IBM Spectrum Protect server needs a dedicated mount point. If the destination storage pool is tape-based, the number of threads will not exceed the number of tape drives (in our example, the backup server needs up to 16 tape drives for this backup only). A better option for accommodating the multi-threaded backups is using a file device class where multiple threads can store data at the same time.

> **Important:** To accommodate the maximum number of sessions determined by the multi-thread parameters, increase the **maxnummp** parameter for the *proxy node* on the Tivoli Storage Manager server. The default value is 2 and you can change this value by using the **update node** command in an administrative client session (**dsmadmc**).
>
> **Tip:** For extensive use of multi-threaded backup operations and file pools, also consider using the preallocated volumes on the Tivoli Storage Manager server side to minimize the fragmentation impact on the server file system used to hold the client backup data. See the following site:
>
> https://www.ibm.com/support/knowledgecenter/SSGSG7_6.3.4/com.ibm.itsm.perf.doc/r_ptg_sysfragfile.html

Example 3-12 shows the file space generated during the backup operation on the backup server.

*Example 3-12 Filespace for the cluster file system*

```
tsm: TSMGPFS>q file

Node Name          Filespace     FSID    Platform    Filespac-    Is Filesp-        Capacity      Pct
                   Name                              e Type       ace Unico-                      Util
                                                                     de?
---------------    -----------   ----    --------    ---------    ----------    -----------    -----
GPFS_PROXY         /dasd_fs1       23    LinuxZ64    GPFS             No             165 GB     15.9
```

## 3.1.2 Using Tivoli Storage Manager compression and data deduplication

In a scenario with an IBM Spectrum Protect (Tivoli Storage Manager) server placed in a distinct site (third site that is site C) other than the primary and secondary NSD servers, the data is sent to the Tivoli Storage Manager server by using the TCP/IP intersite communication network.

Tivoli Storage Manager provides data size reduction mechanisms, such as client-side data deduplication and compression for efficient transfer of the data between the source nodes (Tivoli Storage Manager clients) and the Tivoli Storage Manager server.

**CPU consumption:** Both data deduplication and compression increase CPU consumption on the client. Client-side CPU consumption trade-off should be evaluated thoroughly. Considering application CPU requirements and backup schedule, using data deduplication or compression can produce economies in network bandwidth and storage space.

The data reduction depends on data pattern, so using both compression and data deduplication does not mean double reduction of data, and sometimes the benefits are not significant.

Data deduplication and compression on the Tivoli Storage Manager client side can be used independently or simultaneously.

Also, consider that data deduplication requires a specific server-side configuration with a target pool enabled for data deduplication on the Tivoli Storage Manager server. The data deduplication storage pool can be created on a disk device using the FILE device class on the Tivoli Storage Manager server.

Starting with Tivoli Storage Manager server 7.1.3, a new type of storage pool was introduced, the *container pool*, which enables the Tivoli Storage Manager server inline data deduplication. Both server storage pool options (based on FILE device class or container type pool) are supported for client-side data deduplication.

Also, compression can be performed by the target device (tape drive). However, this does not bring network bandwidth economies (for Tivoli Storage Manager client to server data transfer).

The following scenarios show various examples for using data deduplication and compression with a Tivoli Storage Manager server version 7.1.4 and a directory container target storage pool (created on the Tivoli Storage Manager server) for our Spectrum Storage clustered file system.

### Backup (mmbackup) using client-side data deduplication

In this example, we use one node in site A (lnxnode1) for backing up the clustered file system `/dasd_fs1`. We enable the client-side data deduplication by using the "*deduplication yes*" option in the client system options file, in our case `dsm.sys`.

We also ensure that the node definition on the Tivoli Storage Manager server has the *DEDUPLICATION* attribute set to "*ClientorServer*" (default). You can change this attribute by using the Tivoli Storage Manager **update node** command. See the following link for setting up client-side data deduplication:

https://www.ibm.com/support/knowledgecenter/SSGSG7_7.1.4/client/c_dedup.html?lang=en

For exemplification of the data reduction, we use two subsequent full backups using **mmbackup**:

- Initial full backup without data deduplication enabled for the `/dasd_fs1` file system. Example 3-13 shows a full backup of 5.63 GB (6,045,166,469 bytes) without data deduplication.

*Example 3-13 Full backup without data deduplication*

```
lnxnode1:~ # mmbackup /dasd_fs1 -t full
--------------------------------------------------------
mmbackup: Backup of /dasd_fs1 begins at Tue Dec 22 16:37:49 EST 2015.
--------------------------------------------------------
Tue Dec 22 16:37:52 2015 mmbackup:Scanning file system dasd_fs1
```

```
Tue Dec 22 16:37:53 2015 mmbackup:Determining file system changes for dasd_fs1
[TSMGPFS].
Tue Dec 22 16:37:53 2015 mmbackup:changed=5979, expired=0, unsupported=0 for
server [TSMGPFS]
Tue Dec 22 16:37:53 2015 mmbackup:Sending files to the TSM server [5979
changed, 0 expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      5979
        Total number of objects backed up:      5979
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
        Total number of objects failed:         0
        Total number of objects encrypted:      0
        Total number of bytes inspected:        6045166469
        Total number of bytes transferred:      6045166469
----------------------------------------------------------
mmbackup: Backup of /dasd_fs1 completed successfully at Tue Dec 22 16:40:35 EST
2015.
----------------------------------------------------------
```

- A second full backup is performed on the same file system, with client-side data deduplication enabled, as shown in Example 3-14.

*Example 3-14 Full backup with client-side data deduplication*

```
lnxnode1:~ # mmbackup /dasd_fs1 -t full
--------------------------------------------------------
mmbackup: Backup of /dasd_fs1 begins at Tue Dec 22 16:49:02 EST 2015.
--------------------------------------------------------
Tue Dec 22 16:49:05 2015 mmbackup:Scanning file system dasd_fs1
Tue Dec 22 16:49:06 2015 mmbackup:Determining file system changes for dasd_fs1
[TSMGPFS].
Tue Dec 22 16:49:06 2015 mmbackup:changed=5979, expired=0, unsupported=0 for
server [TSMGPFS]
Tue Dec 22 16:49:06 2015 mmbackup:Sending files to the TSM server [5979
changed, 0 expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      5979
        Total number of objects backed up:      5979
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
        Total number of objects failed:         0
        Total number of objects encrypted:      0
        Total number of bytes inspected:        6045166469
        Total number of bytes transferred:      1509949
----------------------------------------------------------
mmbackup: Backup of /dasd_fs1 completed successfully at Tue Dec 22 16:50:07 EST
2015.
----------------------------------------------------------
```

**Attention:** Data deduplication efficiency depends on the actual data. For evaluating data deduplication results, see the following link:

https://www.ibm.com/support/knowledgecenter/SSGSG7_7.1.1/com.ibm.itsm.perf.doc/t_dedup_process.html?lang=en

In our case, using data deduplication has resulted in significant network bandwidth spare. Your results will vary depending on your data pattern.

Detailed statistics on session transfer can also be obtained from the Tivoli Storage Manager server activity log (**q actlog**). See the following output (Example 3-15) for the previous backup session (Example 3-14 on page 72).

*Example 3-15 Server logs for full backup with data deduplication*

```
12/22/2015 16:49:08      ANR0403I Session 35 ended for node LNXNODE1 (LinuxZ64).
                          (SESSION: 35)
12/22/2015 16:49:08      ANE4952I (Session: 33, Node: GPFS_PROXY)  Total number of
                          objects inspected:       5,979  (SESSION: 33)
12/22/2015 16:49:08      ANE4954I (Session: 33, Node: GPFS_PROXY)  Total number of
                          objects backed up:       5,979  (SESSION: 33)
12/22/2015 16:49:08      ANE4958I (Session: 33, Node: GPFS_PROXY)  Total number of
                          objects updated:             0  (SESSION: 33)
12/22/2015 16:49:08      ANE4960I (Session: 33, Node: GPFS_PROXY)  Total number of
                          objects rebound:             0  (SESSION: 33)
12/22/2015 16:49:08      ANE4957I (Session: 33, Node: GPFS_PROXY)  Total number of
                          objects deleted:             0  (SESSION: 33)
12/22/2015 16:49:08      ANE4970I (Session: 33, Node: GPFS_PROXY)  Total number of
                          objects expired:             0  (SESSION: 33)
12/22/2015 16:49:08      ANE4959I (Session: 33, Node: GPFS_PROXY)  Total number of
                          objects failed:              0  (SESSION: 33)
12/22/2015 16:49:08      ANE4197I (Session: 33, Node: GPFS_PROXY)  Total number of
                          objects encrypted:           0  (SESSION: 33)
12/22/2015 16:49:08      ANE4982I (Session: 33, Node: GPFS_PROXY)  Total objects
                          deduplicated:              3,384  (SESSION: 33)
12/22/2015 16:49:08      ANE4914I (Session: 33, Node: GPFS_PROXY)  Total number of
                          objects grew:                0  (SESSION: 33)
12/22/2015 16:49:08      ANE4916I (Session: 33, Node: GPFS_PROXY)  Total number of
                          retries:                     0  (SESSION: 33)
12/22/2015 16:49:08      ANE4977I (Session: 33, Node: GPFS_PROXY)  Total number of
                          bytes inspected:          5.63 GB  (SESSION: 33)
12/22/2015 16:49:08      ANE4975I (Session: 33, Node: GPFS_PROXY)  Total number of
                          bytes processed:          1.29 MB  (SESSION: 33)
12/22/2015 16:49:08      ANE4984I (Session: 33, Node: GPFS_PROXY)  Total bytes
                          before deduplication:         5.63 GB  (SESSION: 33)
12/22/2015 16:49:08      ANE4198I (Session: 33, Node: GPFS_PROXY)  Total bytes
                          after deduplication:             0  B  (SESSION: 33)
12/22/2015 16:49:08      ANE4961I (Session: 33, Node: GPFS_PROXY)  Total number of
                          bytes transferred:        1.44 MB  (SESSION: 33)
12/22/2015 16:49:08      ANE4963I (Session: 33, Node: GPFS_PROXY)  Data transfer
                          time:                        0.05 sec  (SESSION: 33)
12/22/2015 16:49:08      ANE4966I (Session: 33, Node: GPFS_PROXY)  Network data
                          transfer rate:         26,678.33 KB/sec  (SESSION: 33)
12/22/2015 16:49:08      ANE4967I (Session: 33, Node: GPFS_PROXY)  Aggregate data
                          transfer rate:            25.54 KB/sec  (SESSION: 33)
```

```
12/22/2015 16:49:08      ANE4968I (Session: 33, Node: GPFS_PROXY)  Objects
                          compressed by:                        0%   (SESSION: 33)
12/22/2015 16:49:08      ANE4981I (Session: 33, Node: GPFS_PROXY)  Deduplication
                          reduction:                 100.00%   (SESSION: 33)
12/22/2015 16:49:08      ANE4976I (Session: 33, Node: GPFS_PROXY)  Total data
                          reduction ratio:              99.98%   (SESSION: 33)
12/22/2015 16:49:08      ANE4964I (Session: 33, Node: GPFS_PROXY)  Elapsed
                          processing time:              00:00:57  (SESSION: 33)
12/22/2015 16:49:08      ANR0399I Session 33 for node LNXNODE1 has ended a proxy
                          session for node GPFS_PROXY.  (SESSION: 33)
12/22/2015 16:49:08      ANR0403I Session 33 ended for node LNXNODE1 (LinuxZ64).
                          (SESSION: 33)
12/22/2015 16:49:08      ANR0399I Session 34 for node LNXNODE1 has ended a proxy
                          session for node GPFS_PROXY.  (SESSION: 34)
```

### Backup (mmbackup) using client-side data deduplication *and* compression

In this scenario, we use compression *and* data deduplication enabled on the client. We enable the compression (by setting "*compression yes*") and the data deduplication (by setting "*deduplication yes*") in the client system options file. Example 3-16 shows the client system options file.

*Example 3-16 dsm.sys with data deduplication and compression enabled*

```
lnxnode1:~ # cat $DSM_DIR/dsm.sys
SErvername  TSMGPFS
   COMMMethod         TCPip
   TCPPort            1500
   TCPServeraddress   gpfstsm
   txnb 2g
   PASSWORDACCESS GENERATE
   NODENAME lnxnode1
   ASNODENAME gpfs_proxy
   DEDUPLICATION YES
   COMPRESSION YES
```

We repeat the test and perform a full backup (**mmbackup** with **-t full** option), as shown in Example 3-17. Observe that the amount of data transferred to the Tivoli Storage Manager server is 1.24 MB (1,300,234 bytes). Results vary depending on the type of data.

*Example 3-17 Full backup*

```
lnxnode1:~ # mmbackup /dasd_fs1 -t full
--------------------------------------------------------
mmbackup: Backup of /dasd_fs1 begins at Tue Dec 22 17:42:03 EST 2015.
--------------------------------------------------------
Tue Dec 22 17:42:06 2015 mmbackup:Scanning file system dasd_fs1
Tue Dec 22 17:42:07 2015 mmbackup:Determining file system changes for dasd_fs1
[TSMGPFS].
Tue Dec 22 17:42:07 2015 mmbackup:changed=5979, expired=0, unsupported=0 for
server [TSMGPFS]
Tue Dec 22 17:42:07 2015 mmbackup:Sending files to the TSM server [5979 changed, 0
expired].
mmbackup: TSM Summary Information:
```

```
        Total number of objects inspected:      5979
        Total number of objects backed up:      5979
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
        Total number of objects failed:         0
        Total number of objects encrypted:      0
        Total number of bytes inspected:        6045166469
        Total number of bytes transferred:      1300234
----------------------------------------------------------
mmbackup: Backup of /dasd_fs1 completed successfully at Tue Dec 22 17:43:08 EST
2015.
----------------------------------------------------------
```

### *Multi-threaded backup*

The same options can be used with **mmbackup** and multi-node backup. In this case, the data deduplication and compression settings should be set in the same way on all nodes specified with the **-N** option.

Example 3-18 shows a case of running the **mmbackup** with compression and data deduplication enabled on two nodes (lnxnode1 in site A and lnxnode5 in site B).

*Example 3-18 Multi-threaded backup with compression and data deduplication*

```
lnxnode1:~ # mmbackup /dasd_fs1 -t full -N lnxnode1,lnxnode5
--------------------------------------------------------
mmbackup: Backup of /dasd_fs1 begins at Tue Dec 22 17:52:13 EST 2015.
--------------------------------------------------------
Tue Dec 22 17:52:16 2015 mmbackup:Scanning file system dasd_fs1
Tue Dec 22 17:52:17 2015 mmbackup:Determining file system changes for dasd_fs1
[TSMGPFS].
Tue Dec 22 17:52:18 2015 mmbackup:changed=5979, expired=0, unsupported=0 for
server [TSMGPFS]
Tue Dec 22 17:52:18 2015 mmbackup:Sending files to the TSM server [5979 changed, 0
expired].
mmbackup: TSM Summary Information:
        Total number of objects inspected:      5979
        Total number of objects backed up:      5979
        Total number of objects updated:        0
        Total number of objects rebound:        0
        Total number of objects deleted:        0
        Total number of objects expired:        0
        Total number of objects failed:         0
        Total number of objects encrypted:      0
        Total number of bytes inspected:        6045166557
        Total number of bytes transferred:      1309141
----------------------------------------------------------
mmbackup: Backup of /dasd_fs1 completed successfully at Tue Dec 22 17:53:02 EST
2015.
----------------------------------------------------------
```

The sessions opened during the **mmbackup** operations on the Tivoli Storage Manager server can be captured by using the **q ses** command (**dsmadmc**), as shown in Example 3-19.

*Example 3-19 Tivoli Storage Manager server session information*

```
tsm: TSMGPFS>q ses
Session established with server TSMGPFS: Linux/x86_64
  Server Version 7, Release 1, Level 4.0
  Server date/time: 12/22/2015 17:51:55  Last access: 12/22/2015 17:11:01


 Sess      Comm.      Sess        Wait       Bytes       Bytes      Sess        Platform     Client Name
Number     Method     State        Time        Sent        Recvd     Type
------     ------     ------     ------     -------     -------     -------     --------
------------------
    64     Tcp/Ip     IdleW       16 S     151.9 K      65.6 K     Node        LinuxZ64     GPFS_PROXY
                                                                                              (LNXNODE1)
    67     Tcp/Ip     IdleW       27 S     113.2 K      50.1 K     Node        LinuxZ64     GPFS_PROXY
                                                                                              (LNXNODE5)
    68     Tcp/Ip     Run          0 S         883       1.3 M     Node        LinuxZ64     GPFS_PROXY
                                                                                              (LNXNODE1)
    69     Tcp/Ip     IdleW       31 S         842         549     Node        LinuxZ64     GPFS_PROXY
                                                                                              (LNXNODE1)
    70     Tcp/Ip     Run          0 S         851       2.1 M     Node        LinuxZ64     GPFS_PROXY
                                                                                              (LNXNODE5)
    71     Tcp/Ip     IdleW        6 S     370.8 K     439.8 K     Node        LinuxZ64     GPFS_PROXY
                                                                                              (LNXNODE5)
    72     Tcp/Ip     Run          0 S         162         236     Admin       Linux        ADMIN
                                                                                 x86-64
```

## 3.2 Striping and I/O balancing considerations

In this section, we list some considerations on striping mechanisms. Spectrum Scale implements striping in the file system across disks (NSDs) in the same storage pool. Achieving high throughput to a single, large file requires striping data across multiple disks, multiple disk controllers, and nodes. For Spectrum Scale performance monitoring, see the manual *IBM Spectrum Scale: Advanced Administration Guide,* SC23-7032-03:

http://www.ibm.com/support/knowledgecenter/api/content/nl/en-us/STXKQY_4.2.0/c2370323.pdf

In addition to its own striping mechanisms, for achieving maximum performance and reliability a Spectrum Scale cluster must be designed employing load balancing and striping mechanisms for the following configurations:

- Storage built-in striping capabilities: Type of backing array and number of spindles

  Storage subsystems use built-in striping methods to maximize hardware performance and provide data protection. Various array types can be configured (storage subsystem dependent), each having distinct performance and availability characteristics. Storage chunks carved from these arrays are provided to Spectrum Scale to be used as NSDs for the file systems. Since Spectrum Scale also implements striping across disks (NSDs), optimal storage striping should consider both striping mechanisms to avoid overstriping.

  An optimal striping scheme should always start with application in mind: Data access pattern and data layout must be considered when designing the cluster storage.

- Storage access multipath (for node I/O load balancing) mechanisms

  Storage access multipath is configured at the system and storage subsystem level. A storage subsystem can be accessed through multiple channels (built in) for performance (load balancing) and path redundancy (availability). The cluster node (especially storage nodes) should also have multiple adapters (HBAs) to access the storage subsystem. The OS device drivers are capable of handling access through multiple paths to the storage subsystem and manage this access for I/O load balancing and access availability. A combination of storage subsystem capabilities and device driver (OS) implementation must be considered to achieve optimal I/O performance and availability for Spectrum Scale storage nodes.

- NSD I/O load balancing (Spectrum Scale configuration)

  For SCSI devices, to balance I/O across storage nodes, it is possible to configure NSD access through multiple nodes (alternate servers of NSDs defined to a file system). This method is in addition to node I/O load balancing (through device driver/OS multipath) and file system striping. This method also provides better performance for certain storage subsystems that do not provide simultaneous LUN access through multiple controllers (active/passive). See Example 2-14 on page 23.

# 3.3 Configuring devices on Linux on z (post-installation)

Default installation (for SUSE Linux Enterprise Server 12) blacklists devices that are not used during installation (security reasons). You need to identify appropriate (allocated and allowed) devices available in z/VM and configure them to Linux guest.

It is a good idea during OS installation (we used GUI via VNC) to save a list of devices that have been identified by the installation program. This serves as a reference later when you need to configure (make available) additional devices. We saved the discovered devices as a `hwinfo.out` file. See Example 3-20.

*Example 3-20 List of devices identified during system installation*

```
# ls -l ~/inst-sys/hwinfo.out
-rw-r--r-- 1 root root 128290 Nov 14 10:23 /root/inst-sys/hwinfo.out
```

The devices are blacklisted to avoid inadvertent access to the resources. Only the devices used during the installation process are enabled and used after first reboot. To configure additional (blacklisted) devices in Linux, these must be removed from the “blacklist”.

> **Important:** For Linux on IBM z Systems, the following package is *required* (SUSE Linux Enterprise Server 12, shown here):
>
> ```
> # rpm -qa s390-tools
> s390-tools-1.24.1-38.17.s390x
> ```

The **`cio_ignore`** command provides functions to query and modify the contents of the CIO device driver blacklist. This blacklist determines if Linux tries to make a device that is connected through the channel subsystem (CSS) available for use by Linux.

To identify the range of devices blacklisted, use the **cio_ignore** command, as shown in Example 3-21.

*Example 3-21 Listing blacklisted (ignored) devices*

```
# cio_ignore -l
Ignored devices:
=================
0.0.0000-0.0.0008
0.0.000a-0.0.0201
0.0.0203-0.0.301f
0.0.3023-0.0.ffff
0.1.0000-0.1.ffff
0.2.0000-0.2.ffff
0.3.0000-0.3.ffff
```

To list the non-blacklisted devices, see Example 3-22.

*Example 3-22 Non-blacklisted devices*

```
# cio_ignore -L
Devices that are not ignored:
=============================
0.0.0009
0.0.0202
0.0.3020-0.0.3022

# lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.3020 0.0.0000  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3021 0.0.0001  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3022 0.0.0002  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.0202 0.0.000b  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0009 0.0.001d  0000/00 3215/00 yes  80  80  ff   08000000 00000000
```

Next, you need to obtain a list of devices that can be configured to your Linux guest from your z/VM system administrator.

In the following sections, we configure the following types of devices to our Linux guest:

- Networking devices
  - Virtual NIC (OSA) that connects into a virtual switch
  - OSA connection direct NIC (not via a virtual switch)
- Storage devices
  - Extended count key data (ECKD)
  - FCP devices (HBA with NPIV feature enabled) with the associated LUNs (SCSI devices)

## 3.3.1 Network devices (interfaces)

This section describes how to configure the following types of network devices:

- Virtual network interface (virtual OSA) managed by z/VM
- Direct attached OSA

### Virtual NIC (OSA)

We list the virtual OSA devices available from the z/VM host, as shown in Example 3-23.

*Example 3-23 Initial NIC configuration and available devices (eth0 -- 0.0.3020-3022)*

```
# lsqeth
Device name                  : eth0
-------------------------------------------------------------------------
        card_type            : VSWITCH: SYSTEM VSW1 (Type: QDIO)
        cdev0                : 0.0.3020
        cdev1                : 0.0.3021
        cdev2                : 0.0.3022
        chpid                : 08
        online               : 1
        portno               : 0
        state                : UP (LAN ONLINE)
        priority_queueing    : always queue 2
        buffer_count         : 64
        layer2               : 1
        isolation            : none

# vmcp q virtual nic
Adapter 0700.P00 Type: QDIO      Name: UNASSIGNED  Devices: 3
  MAC: 02-00-07-00-00-2E         VSWITCH: SYSTEM VSW2
Adapter 0800.P00 Type: QDIO      Name: UNASSIGNED  Devices: 3
  MAC: 02-00-07-00-00-2F         VSWITCH: SYSTEM VSW3
Adapter 3020.P00 Type: QDIO      Name: UNASSIGNED  Devices: 3
  MAC: 02-00-07-00-00-2D         VSWITCH: SYSTEM VSW1
```

To “unblacklist” the NIC 0700 connected to virtual switch VSW2, use the commands shown in Example 3-24.

*Example 3-24 Adding device 0.0.0700-702*

```
# cio_ignore -r 0700
# cio_ignore -r 0701
# cio_ignore -r 0702

# lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.3020 0.0.0000  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3021 0.0.0001  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3022 0.0.0002  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.0700 0.0.0003  1732/01 1731/01      80  80  ff   09000000 00000000
0.0.0701 0.0.0004  1732/01 1731/01      80  80  ff   09000000 00000000
0.0.0702 0.0.0005  1732/01 1731/01      80  80  ff   09000000 00000000
0.0.0202 0.0.000b  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0009 0.0.001d  0000/00 3215/00 yes  80  80  ff   08000000 00000000
```

The QDIO device must now be configured as a Linux NIC (qeth) and made available (online), as shown in Example 3-25.

*Example 3-25 Device 0.0.0700-0702 configured as eth1*

```
# qeth_configure -l -t qeth 0.0.0700 0.0.0701 0.0.0702 1
(Layer2)

# lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.3020 0.0.0000  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3021 0.0.0001  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3022 0.0.0002  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.0700 0.0.0003  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0701 0.0.0004  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0702 0.0.0005  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0202 0.0.000b  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0009 0.0.001d  0000/00 3215/00 yes  80  80  ff   08000000 00000000

# lsqeth
Device name                 : eth1
-------------------------------------------------------------------------
        card_type           : VSWITCH: SYSTEM VSW2 (Type: QDIO)
        cdev0               : 0.0.0700
        cdev1               : 0.0.0701
        cdev2               : 0.0.0702
        chpid               : 09
        online              : 1
        portno              : 0
        state               : SOFTSETUP
        priority_queueing   : always queue 2
        buffer_count        : 64
        layer2              : 1
        isolation           : none

Device name                 : eth0
-------------------------------------------------------------------------
        card_type           : VSWITCH: SYSTEM VSW1 (Type: QDIO)
        cdev0               : 0.0.3020
        cdev1               : 0.0.3021
        cdev2               : 0.0.3022
        chpid               : 08
        online              : 1
        portno              : 0
        state               : UP (LAN ONLINE)
        priority_queueing   : always queue 2
        buffer_count        : 64
        layer2              : 1
        isolation           : none
# ip ad
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
```

```
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
group default qlen 1000
    link/ether 02:00:07:00:00:2d brd ff:ff:ff:ff:ff:ff
    inet 9.12.7.26/20 brd 9.12.15.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::7ff:fe00:2d/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen
1000
    link/ether 02:00:07:00:00:2e brd ff:ff:ff:ff:ff:ff
```

The *eth1* interface can now be configured with IP and started, as shown in Example 3-26.

*Example 3-26 IP configuration for eth1*

```
# cat /etc/sysconfig/network/ifcfg-eth1
BOOTPROTO='static'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR=''
MTU=''
NAME='OSA Express Network card (0.0.0700)'
NETMASK='255.255.255.0'
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
IPADDR_0='192.168.60.247/24'

# ifup eth1
eth1 up

# ip ad
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
group default qlen 1000
    link/ether 02:00:07:00:00:2d brd ff:ff:ff:ff:ff:ff
    inet 9.12.7.26/20 brd 9.12.15.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::7ff:fe00:2d/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
group default qlen 1000
    link/ether 02:00:07:00:00:2e brd ff:ff:ff:ff:ff:ff
    inet 192.168.60.247/24 brd 192.168.60.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::7ff:fe00:2e/64 scope link
       valid_lft forever preferred_lft forever
```

### Direct attached OSA (not via virtual switch)

Example 3-27 shows the OSA devices available to the Linux guest. We are looking for an OSA type “OSD” directly connected (not via a virtual switch).

*Example 3-27 Identifying OSA direct*

```
# vmcp q osa
.......<< Snippet >>.......
OSA  2D46 ON OSA   2D46 SUBCHANNEL = 0014
     2D46 DEVTYPE OSA         VIRTUAL CHPID 06 OSD REAL CHPID 06
     2D46 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
OSA  2D47 ON OSA   2D47 SUBCHANNEL = 0015
     2D47 DEVTYPE OSA         VIRTUAL CHPID 06 OSD REAL CHPID 06
     2D47 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
OSA  2D48 ON OSA   2D48 SUBCHANNEL = 0016
     2D48 DEVTYPE OSA         VIRTUAL CHPID 06 OSD REAL CHPID 06
     2D48 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
.......<< Snippet >>.......
```

We “unblacklist” the devices corresponding to this interface, as shown in Example 3-28.

*Example 3-28 Adding eth2 device*

```
# cio_ignore -r 2D46
# cio_ignore -r 2D47
# cio_ignore -r 2D48

# lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.3020 0.0.0000  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3021 0.0.0001  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3022 0.0.0002  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.0700 0.0.0003  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0701 0.0.0004  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0702 0.0.0005  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0202 0.0.000b  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.2d46 0.0.0014  1732/01 1731/01      80  80  ff   06000000 00000000
0.0.2d47 0.0.0015  1732/01 1731/01      80  80  ff   06000000 00000000
0.0.2d48 0.0.0016  1732/01 1731/01      80  80  ff   06000000 00000000
0.0.0009 0.0.001d  0000/00 3215/00 yes  80  80  ff   08000000 00000000
```

We define and configure online the (qeth) interface eth2 to Linux, as shown Example 3-29.

*Example 3-29 Configuring online the eth2 interface*

```
# qeth_configure -l -t qeth 0.0.2d46 0.0.2d47 0.0.2d48 1
(Layer2)
linux-cbo4:~ # ip ad
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
.......<< Snippet >>.......
```

```
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen
1000
    link/ether 02:00:00:bf:ba:99 brd ff:ff:ff:ff:ff:ff
```

The device can now be configured with IP, as shown in Example 3-30.

*Example 3-30 Configuring IP for eth2*

```
# cat /etc/sysconfig/network/ifcfg-eth2
BOOTPROTO='static'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR=''
MTU=''
NAME='OSA Express Network card (0.0.2D46)'
NETMASK='255.255.255.0'
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
IPADDR_0='10.1.1.247/24'

# ifup eth2
eth2            up

# ip ad
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
.......<< Snippet >>.......
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
group default qlen 1000
    link/ether 02:00:00:bf:ba:99 brd ff:ff:ff:ff:ff:ff
    inet 10.1.1.247/24 brd 10.1.1.255 scope global eth2
       valid_lft forever preferred_lft forever
    inet6 fe80::ff:febf:ba99/64 scope link
       valid_lft forever preferred_lft forever
```

## 3.3.2 Disk storage

This section describes how to configure storage to the Linux guests. The storage devices can be presented to the guests as:

- Extended count key data (ECKD) devices
- FCP devices with associated LUNs (SCSI devices)

### ECKD devices

We identify the DASDs available to the Linux guest, as shown in Example 3-31.

*Example 3-31 Identifying z/VM host storage (ECKD)*

```
# vmcp q dasd
DASD 0190 3390 LX7RES R/O       214 CYL ON DASD  1092 SUBCHANNEL = 001E
```

```
DASD 0191 3390 LX7UR1 R/W        100 CYL ON DASD  134C SUBCHANNEL = 000A
DASD 019D 3390 LX7RES R/O        292 CYL ON DASD  1092 SUBCHANNEL = 001F
DASD 019E 3390 LX7RES R/O        500 CYL ON DASD  1092 SUBCHANNEL = 0020
DASD 0202 3390 LXC616 R/W      30050 CYL ON DASD  C616 SUBCHANNEL = 000B
DASD 0301 3390 LXC610 R/W      30050 CYL ON DASD  C610 SUBCHANNEL = 000C
DASD 0302 3390 LXC611 R/W      30050 CYL ON DASD  C611 SUBCHANNEL = 000D
DASD 0303 3390 LXC612 R/W      30050 CYL ON DASD  C612 SUBCHANNEL = 000E
DASD 0304 3390 LXC613 R/W      30050 CYL ON DASD  C613 SUBCHANNEL = 000F
DASD 0401 3390 LX7RES R/O        292 CYL ON DASD  1092 SUBCHANNEL = 0022
DASD 0402 3390 LX7RES R/O        292 CYL ON DASD  1092 SUBCHANNEL = 0021
DASD 0592 3390 LX7RES R/O        140 CYL ON DASD  1092 SUBCHANNEL = 0009
```

We "unblacklist" the devices that we need, as shown in Example 3-32.

*Example 3-32 Removing ECKD devices blacklist*

```
# cio_ignore -r 0301
# cio_ignore -r 0302
# cio_ignore -r 0303
# cio_ignore -r 0304

# lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.3020 0.0.0000  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3021 0.0.0001  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3022 0.0.0002  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.0700 0.0.0003  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0701 0.0.0004  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0702 0.0.0005  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0202 0.0.000b  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0301 0.0.000c  3390/0c 3990/e9      f0  e0  ff   52565a5e 00000000
0.0.0302 0.0.000d  3390/0c 3990/e9      f0  e0  ff   52565a5e 00000000
0.0.0303 0.0.000e  3390/0c 3990/e9      f0  e0  ff   52565a5e 00000000
0.0.0304 0.0.000f  3390/0c 3990/e9      f0  e0  ff   52565a5e 00000000
0.0.2d46 0.0.0014  1732/01 1731/01 yes  80  80  ff   06000000 00000000
0.0.2d47 0.0.0015  1732/01 1731/01 yes  80  80  ff   06000000 00000000
0.0.2d48 0.0.0016  1732/01 1731/01 yes  80  80  ff   06000000 00000000
0.0.0009 0.0.001d  0000/00 3215/00 yes  80  80  ff   08000000 00000000
```

We activate the devices, as shown in Example 3-33.

*Example 3-33 Activating DASDs*

```
# dasd_configure 0.0.0301 1
# dasd_configure 0.0.0302 1
# dasd_configure 0.0.0303 1
# dasd_configure 0.0.0304 1

# lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.3020 0.0.0000  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3021 0.0.0001  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3022 0.0.0002  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.0700 0.0.0003  1732/01 1731/01 yes  80  80  ff   09000000 00000000
```

```
0.0.0701 0.0.0004  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0702 0.0.0005  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0202 0.0.000b  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0301 0.0.000c  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0302 0.0.000d  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0303 0.0.000e  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0304 0.0.000f  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.2d46 0.0.0014  1732/01 1731/01 yes  80  80  ff   06000000 00000000
0.0.2d47 0.0.0015  1732/01 1731/01 yes  80  80  ff   06000000 00000000
0.0.2d48 0.0.0016  1732/01 1731/01 yes  80  80  ff   06000000 00000000
0.0.0009 0.0.001d  0000/00 3215/00 yes  80  80  ff   08000000 00000000

# lsdasd
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
0.0.0202   active      dasda     94:0    ECKD  4096   21128MB   5409000
0.0.0301   active      dasdb     94:4    ECKD  4096   21128MB   5409000
0.0.0302   active      dasdc     94:8    ECKD  4096   21128MB   5409000
0.0.0303   active      dasdd     94:12   ECKD  4096   21128MB   5409000
0.0.0304   active      dasde     94:16   ECKD  4096   21128MB   5409000
```

Format the DASDs added in the previous step, as shown in Example 3-34.

> **Important:** In Example 3-33, *dasda* (/dev/dasda, Bus-ID 0.0.0202) is used by the operating system. Make sure that you *do not format* this device.
>
> **Avoid formatting multiple times:** If you are sharing the DASDs between multiple Linux guests, you need to format these devices only once (on a single Linux guest).

*Example 3-34 Formatting the previously added DASDs*

```
# for i in b c d e
> do
> dasdfmt -b 4096 -d cdl -y -f /dev/dasd${i} &
> done
[1] 39243
[2] 39244
[3] 39245
[4] 39246

# jobs
[1]   Running                 dasdfmt -b 4096 -y -f /dev/dasd${i} &
[2]   Running                 dasdfmt -b 4096 -y -f /dev/dasd${i} &
[3]-  Running                 dasdfmt -b 4096 -y -f /dev/dasd${i} &
[4]+  Running                 dasdfmt -b 4096 -y -f /dev/dasd${i} &

# Finished formatting the device.
Rereading the partition table... ok
Finished formatting the device.
Rereading the partition table... ok
Finished formatting the device.
Rereading the partition table... ok
Finished formatting the device.
Rereading the partition table... ok
```

```
[1]   Done                    dasdfmt -b 4096 -y -f /dev/dasd${i}
[2]   Done                    dasdfmt -b 4096 -y -f /dev/dasd${i}
[3]-  Done                    dasdfmt -b 4096 -y -f /dev/dasd${i}
[4]+  Done                    dasdfmt -b 4096 -y -f /dev/dasd${i}
```

### FCP device (HBA with NPIV feature enabled) with the associated LUNs (SCSI devices)

We identify the FCP devices that are available to our Linux guest, as shown in Example 3-35.

*Example 3-35 Identifying FCP devices with NPIV feature enabled*

```
# vmcp q v fcp
FCP  FC00 ON FCP   B604 CHPID 76 SUBCHANNEL = 0010
     FC00 DEVTYPE FCP         VIRTUAL CHPID 76 FCP REAL CHPID 76
     FC00 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
     FC00 DATA ROUTER ELIGIBLE
     WWPN C05076DD90002320
FCP  FC01 ON FCP   B605 CHPID 76 SUBCHANNEL = 0011
     FC01 DEVTYPE FCP         VIRTUAL CHPID 76 FCP REAL CHPID 76
     FC01 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
     FC01 DATA ROUTER ELIGIBLE
     WWPN C05076DD90002324
FCP  FD00 ON FCP   B704 CHPID 77 SUBCHANNEL = 0012
     FD00 DEVTYPE FCP         VIRTUAL CHPID 77 FCP REAL CHPID 77
     FD00 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
     FD00 DATA ROUTER ELIGIBLE
     WWPN C05076DD900023A8
FCP  FD01 ON FCP   B705 CHPID 77 SUBCHANNEL = 0013
     FD01 DEVTYPE FCP         VIRTUAL CHPID 77 FCP REAL CHPID 77
     FD01 QDIO-ELIGIBLE       QIOASSIST-ELIGIBLE
     FD01 DATA ROUTER ELIGIBLE
     WWPN C05076DD900023AC
```

We “unblacklist” devices that are highlighted (FC00, FC01, FD00, FD01) in Example 3-35, and configure them online, as shown in Example 3-36. To set the FCP adapter online, use the **zfcp_host_configure** command.

> **Tip:** It is recommended that you configure the LUN access (SAN zoning, LUN masking) before you configure FCP devices to Linux. Use the worldwide port names (WWPNs) from the previous command (**vmcp q v fcp**).

*Example 3-36 Unblacklisting FCP (NPIV) devices*

```
# cio_ignore -r FC00
# cio_ignore -r FC01
# cio_ignore -r FD00
# cio_ignore -r FD01

# lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.3020 0.0.0000  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.3021 0.0.0001  1732/01 1731/01 yes  80  80  ff   08000000 00000000
```

```
0.0.3022 0.0.0002  1732/01 1731/01 yes  80  80  ff   08000000 00000000
0.0.0700 0.0.0003  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0701 0.0.0004  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0702 0.0.0005  1732/01 1731/01 yes  80  80  ff   09000000 00000000
0.0.0202 0.0.000b  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0301 0.0.000c  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0302 0.0.000d  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0303 0.0.000e  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.0304 0.0.000f  3390/0c 3990/e9 yes  f0  e0  ff   52565a5e 00000000
0.0.fc00 0.0.0010  1732/03 1731/03      80  80  ff   76000000 00000000
0.0.fc01 0.0.0011  1732/03 1731/03      80  80  ff   76000000 00000000
0.0.fd00 0.0.0012  1732/03 1731/03      80  80  ff   77000000 00000000
0.0.fd01 0.0.0013  1732/03 1731/03      80  80  ff   77000000 00000000
0.0.2d46 0.0.0014  1732/01 1731/01 yes  80  80  ff   06000000 00000000
0.0.2d47 0.0.0015  1732/01 1731/01 yes  80  80  ff   06000000 00000000
0.0.2d48 0.0.0016  1732/01 1731/01 yes  80  80  ff   06000000 00000000
0.0.0009 0.0.001d  0000/00 3215/00 yes  80  80  ff   08000000 00000000

# zfcp_host_configure 0.0.fc00  1
# zfcp_host_configure 0.0.fc01  1
# zfcp_host_configure 0.0.fd00  1
# zfcp_host_configure 0.0.fd01  1
```

We check that the FCP devices have been set online and LUNs configured, as shown in Example 3-37.

> **Note:** With NPIV-enabled FCP devices, SUSE Linux Enterprise Server 12 uses automatic LUN scanning by default. For FCP devices that are not NPIV-enabled, or if automatic LUN scanning is disabled, add the LUNs (= SCSI devices) manually by using the **`zfcp_disk_configure`** command.

*Example 3-37 Checking FCP adapters and LUNs*

```
linux-cbo4:~ # lszfcp
0.0.fc00 host0
0.0.fc01 host1
0.0.fd00 host2
0.0.fd01 host3

# lsluns
Scanning for LUNs on adapter 0.0.fc00
        at port 0x500507680120bb91:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680120bc24:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680130bb91:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
```

```
                0x0003000000000000
        at port 0x500507680130bc24:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
Scanning for LUNs on adapter 0.0.fc01
        at port 0x500507680120bb91:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680120bc24:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680130bb91:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680130bc24:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
Scanning for LUNs on adapter 0.0.fd00
        at port 0x500507680120bb91:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680120bc24:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680130bb91:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680130bc24:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
Scanning for LUNs on adapter 0.0.fd01
        at port 0x500507680120bb91:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680120bc24:
```

```
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680130bb91:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
        at port 0x500507680130bc24:
                0x0000000000000000
                0x0001000000000000
                0x0002000000000000
                0x0003000000000000
```

Because LUNs (SCSI devices) are accessible via multiple paths, we need to enable the Linux multipath configuration, as shown in Example 3-38. We begin by checking if the "multipath-tools" package is installed. If not installed, install it (not shown here).

*Example 3-38 Enabling Linux multipath configuration*

```
# rpm -qa multipath-tools
multipath-tools-0.5.0-30.1.s390x

# systemctl enable multipathd
ln -s '/usr/lib/systemd/system/multipathd.service'
'/etc/systemd/system/sysinit.target.wants/multipathd.service'
# systemctl start multipathd

# systemctl status multipathd
multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled)
   Active: active (running) since Sat 2015-11-14 16:37:41 EST; 35s ago
  Process: 40066 ExecStartPre=/sbin/modprobe dm-multipath (code=exited,
status=0/SUCCESS)
 Main PID: 40071 (multipathd)
   Status: "running"
   CGroup: /system.slice/multipathd.service
           ··40071 /sbin/multipathd -d -s

# multipath -l
360050768018305e120000000000000f8 dm-0 IBM,2145
size=30G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| |- 0:0:0:0 sda  8:0    active undef running
| |- 0:0:1:0 sde  8:64   active undef running
| |- 1:0:0:0 sdq  65:0   active undef running
| |- 1:0:3:0 sdac 65:192 active undef running
| |- 2:0:0:0 sdag 66:0   active undef running
| |- 2:0:1:0 sdak 66:64  active undef running
| |- 3:0:0:0 sdaw 67:0   active undef running
| `- 3:0:1:0 sdba 67:64  active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 0:0:2:0 sdi  8:128  active undef running
  |- 0:0:3:0 sdm  8:192  active undef running
  |- 1:0:1:0 sdu  65:64  active undef running
```

```
  |- 1:0:2:0 sdy  65:128 active undef running
  |- 2:0:2:0 sdao 66:128 active undef running
  |- 2:0:3:0 sdas 66:192 active undef running
  |- 3:0:2:0 sdbe 67:128 active undef running
  `- 3:0:3:0 sdbi 67:192 active undef running
360050768018305e120000000000000fe dm-3 IBM,2145
size=30G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| |- 0:0:2:3 sdl  8:176  active undef running
| |- 0:0:3:3 sdp  8:240  active undef running
| |- 1:0:1:3 sdx  65:112 active undef running
| |- 1:0:2:3 sdab 65:176 active undef running
| |- 2:0:2:3 sdar 66:176 active undef running
| |- 2:0:3:3 sdav 66:240 active undef running
| |- 3:0:2:3 sdbh 67:176 active undef running
| `- 3:0:3:3 sdbl 67:240 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 0:0:0:3 sdd  8:48   active undef running
  |- 0:0:1:3 sdh  8:112  active undef running
  |- 1:0:0:3 sdt  65:48  active undef running
  |- 1:0:3:3 sdaf 65:240 active undef running
  |- 2:0:0:3 sdaj 66:48  active undef running
  |- 2:0:1:3 sdan 66:112 active undef running
  |- 3:0:0:3 sdaz 67:48  active undef running
  `- 3:0:1:3 sdbd 67:112 active undef running
360050768018305e120000000000000fd dm-2 IBM,2145
size=30G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| |- 0:0:0:2 sdc  8:32   active undef running
| |- 0:0:1:2 sdg  8:96   active undef running
| |- 1:0:0:2 sds  65:32  active undef running
| |- 1:0:3:2 sdae 65:224 active undef running
| |- 2:0:0:2 sdai 66:32  active undef running
| |- 2:0:1:2 sdam 66:96  active undef running
| |- 3:0:0:2 sday 67:32  active undef running
| `- 3:0:1:2 sdbc 67:96  active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 0:0:2:2 sdk  8:160  active undef running
  |- 0:0:3:2 sdo  8:224  active undef running
  |- 1:0:1:2 sdw  65:96  active undef running
  |- 1:0:2:2 sdaa 65:160 active undef running
  |- 2:0:2:2 sdaq 66:160 active undef running
  |- 2:0:3:2 sdau 66:224 active undef running
  |- 3:0:2:2 sdbg 67:160 active undef running
  `- 3:0:3:2 sdbk 67:224 active undef running
360050768018305e120000000000000fc dm-1 IBM,2145
size=30G features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=0 status=active
| |- 0:0:2:1 sdj  8:144  active undef running
| |- 0:0:3:1 sdn  8:208  active undef running
| |- 1:0:1:1 sdv  65:80  active undef running
| |- 1:0:2:1 sdz  65:144 active undef running
| |- 2:0:2:1 sdap 66:144 active undef running
| |- 2:0:3:1 sdat 66:208 active undef running
| |- 3:0:2:1 sdbf 67:144 active undef running
```

```
| `- 3:0:3:1 sdbj 67:208 active undef running
`-+- policy='service-time 0' prio=0 status=enabled
  |- 0:0:0:1 sdb  8:16   active undef running
  |- 0:0:1:1 sdf  8:80   active undef running
  |- 1:0:0:1 sdr  65:16  active undef running
  |- 1:0:3:1 sdad 65:208 active undef running
  |- 2:0:0:1 sdah 66:16  active undef running
  |- 2:0:1:1 sdal 66:80  active undef running
  |- 3:0:0:1 sdax 67:16  active undef running
  `- 3:0:1:1 sdbb 67:80  active undef running
```

Finally, we check SCSI disk capabilities (if the disk supports SCSI-3 Persistent Reservation), as shown Example 3-39.

*Example 3-39 Checking SCSI disk capabilities*

```
# sg_persist -d /dev/dm-0
>> No service action given; assume Persistent Reserve In command
>> with Read Keys service action
  IBM       2145             0000
  Peripheral device type: disk
  PR generation=0x0, there are NO registered reservation keys
```

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

## IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *The Virtualization Cookbook for IBM z Systems Volume 1: IBM z/VM 6.3*, SG24-8147
- *IBM Spectrum Scale (formerly GPFS)*, SG24-8254
- *Implementing IBM Spectrum Scale*, REDP-5254

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

**ibm.com**/redbooks

## Other publications

These publications are also relevant as further information sources:

- *IBM Spectrum Scale V4.2 Documentation web page*

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/ibmspectrumscale42_welcome.html?lang=en
- *IBM Spectrum Protect Documentation web page:*

  http://www.ibm.com/support/knowledgecenter/SSGSG7/landing/welcome_ssgsg7.html

## Online resources

These websites are also relevant as further information sources:

- Linux OS on IBM z Systems web page:

  http:/www.ibm.com/systems/z/linux
- IBM Spectrum Scale web page:

  http://www.ibm.com/systems/storage/spectrum/scale
- IBM Spectrum Scale Frequently Asked Questions web page:

  http://www.ibm.com/support/knowledgecenter/STXKQY_4.2.0/gpfsclustersfaq.html?lang=en-us
- IBM Spectrum Protect web page:

  http://www.ibm.com/software/tivoli/csi/backup-recovery

## Help from IBM

IBM Support and downloads

**ibm.com**/support

IBM Global Services

**ibm.com**/services

REDP-5308-00

ISBN 0738454990

Printed in U.S.A.