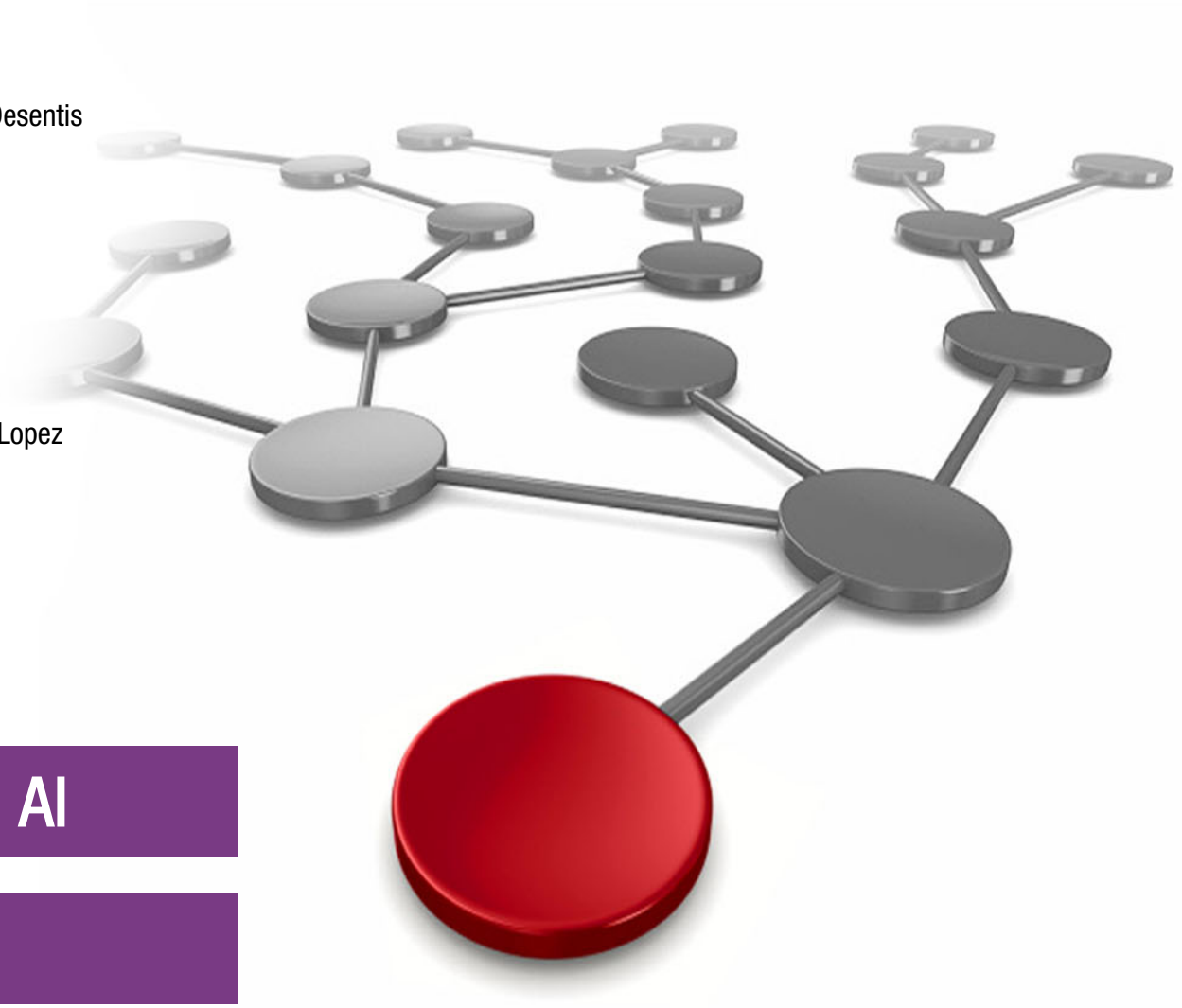
Redbooks
ibm.com/redbooks

IBM Storage Scale System 6000 with NVIDIA DGX SuperPOD Deployment Guide

Chris Maestas
Ana Gabriela Iturbe Desentis
Phillip Gerrard
Kiran Ghag
Nikhil Khandelwal
Matthew Klos
John Lewars
Jesus Daniel Munoz Lopez
Roger E. Sanders
Sanjay Sudam
Lindsay Todd
Joanna Wong



Data and AI

Storage

IBM

Redpaper

IBM Redbooks

IBM Storage Scale System 6000 with NVIDIA DGX SuperPOD Deployment Guide

March 2025

REDP-5746-00

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

(March 2025) First Edition

This edition applies to Version 5.1 of IBM Storage Scale.

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

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at https://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

AIX®
Db2®
IBM®
IBM Cloud®
IBM Elastic Storage®
IBM FlashCore®
IBM Spectrum®
IBM Z®
Redbooks®
Redbooks (logo)  ®

The following terms are trademarks of other companies:

Intel, Intel Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat, Ansible, OpenShift, are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

The business world is being transformed rapidly by artificial intelligence (AI), high-performance computing (HPC), analytics, and hybrid cloud technologies. Unlike traditional applications that work with structured data that is stored in databases, these new workloads operate on a vast ocean of unstructured data, such as documents, images, videos, audio recordings, log files, or any other information that can be generated by users, applications, or even machines.

As a result, organizations are having to re-think their data storage strategies and adapt to new ways of working, including how to use AI to unlock the value of their data, wherever it might be. Information technology (IT) leaders face many new challenges:

- Accessing and analyzing data that is scattered across the globe
- The increasing time and resources that are needed by AI training and inferencing workloads
- The cost and scarcity of resources, such as NVIDIA graphics processing units (GPUs)

Addressing these challenges requires specialized software and hardware:

- IBM Storage Scale: Software-defined file and object storage for both structured and unstructured data.
- IBM Storage Scale System 6000: A hardware implementation of IBM Storage Scale software that is optimized for the most demanding AI, HPC, analytics, and hybrid cloud workloads.
- IBM Storage Scale System 3500: A hardware implementation of IBM Storage Scale software that is designed for organizations requiring enterprise-ready, entry-level, or mid-level storage system.

This IBM Redbooks® publication is designed to provide information that is related to the implementation and deployment of the NVIDIA DGX SuperPOD architecture by using IBM Storage Scale System hardware. It is suitable for anyone looking to learn more about the overall solution. It provides direct examples of solution implementation.

Authors

This paper was produced by a team of specialists from around the world working at IBM Redbooks, Tucson Center.

Chris Maestas is the worldwide Chief Technical Officer for Data and AI Storage Solutions. He has over 25 years of experience deploying and designing IT systems for clients in many industries. His experience in scaling acceleration and resiliency with various file system technologies came from developing benchmark frameworks to test systems for reliability and performance. He has led global sessions discussing how best to position unstructured software-defined storage for file, object, and tape solutions in cloud, on-premises, hybrid, and now at the edge.

Ana Gabriela Iturbe Desentis is a software performance analyst and scrum master for the IBM Storage Scale Performance teams in Guadalajara, Mexico. With over 14 years of experience, she has authored and contributed to hundreds of technical documents that were published internally and externally for the IBM Storage Scale Out Network Attached Storage product. She has been the leader of the upgrade team that is responsible for more than a hundred maintenance windows, and resolved critical situations both remotely and onsite for customers. As a scrum master, she is experienced in leading worldwide teams, removing blockers and impediments, and arranging and optimizing the team backlog. She has experience in performance evaluation on IBM Storage Scale components such as Active File Management (AFM) and disaster recovery (DR). Ana Gabriela holds a degree in Mechatronics Engineering from Universidad del Valle de Mexico, Mexico.

Phillip Gerrard is a Project Leader for the International Technical Support Organization who is based in Beaverton, Oregon. As part of IBM® for over 15 years, he has authored and contributed to hundreds of technical documents to IBM.com and worked directly with IBM's largest customers to resolve critical situations. As a team lead and SME for the IBM Spectrum® Protect support team, he is experienced in leading and growing international teams of talented IBM employees, developing and implementing team processes, and creating and delivering education. Phillip holds a degree in computer science and business administration from Oregon State University.

Kiran Ghag has been an Advisory Delivery Consultant at IBM Technology Expert Labs, India for a decade. Kiran has over 20 years of experience in storage systems design, deployment, operations, and optimization. Kiran delivers IBM solutions for customers and helps them achieve business goals with IBM technology in the field. His current focus areas are software-defined storage (SDS) and HPC by using the IBM Spectrum product family. Kiran holds a bachelors degree in computer engineering from Mumbai University, India.

Matthew Klos is a Senior Solutions Architect working across many industries, including financial services, research organizations, higher education, automotive, healthcare and life sciences, and many more. He has been administering and designing scale and scale system solutions for over 10 years internally and externally. Matt holds a degree in Information Technology from Marist University.

John Lewars is a Senior Technical Staff Member and Performance Architect for IBM Storage Scale (formerly GPFS). With over 25+ years at IBM, he has worked on some of IBM’s largest HPC systems before shifting his focus to parallel file system development. John specializes in performance optimization, network resiliency, and large-scale customer deployments. He also co-led the development of the first IBM Storage Scale public cloud and container-support deliverables, enabling scalable and flexible storage solutions for modern cloud and container environments. John continues to drive scalable storage performance innovation with an emphasis on artificial intelligence (AI)-driven and next-generation HPC workloads, collaborating closely with customers to optimize IBM Storage Scale for performance, scalability, and evolving AI-driven storage needs.

Jesus Daniel Munoz Lopez is a software engineer and member of the IBM Storage Scale System deployment team at Guadalajara, Mexico. He has more than 12 years of experience in multiple areas of the software development lifecycle, such as testing, DevOps, and build, development, and release engineering, by working for multiple projects within IBM. His focus areas include virtualization and containerization technologies, Red Hat Enterprise Linux, automation, Python, quality assurance, continuous integration and continuous delivery, source control, and scripting. Daniel holds a degree in computer engineering from Universidad de Guadalajara, Mexico.

Roger E. Sanders a Principal Learning Content Development – Data, AI, and Storage specialist in the US. He is the author of 25 books on IBM Db2®, one book on Open Database Connectivity, and one book on technical writing. He also authored the “Distributed DBA” column in *IBM Data Magazine* (formerly *Db2 Magazine*) for 10 years and has written articles for publications like *Certification Magazine*, *International Db2 User’s Group (IDUG) Solutions Journal*, and *Database Trends and Applications*. Roger has also authored tutorials and articles for the IBM Developer website, presented at many IDUG conferences, taught numerous courses on Db2 Fundamentals and Db2 Database Administration, and participated in the development of 25 Db2 certification exams. Over the past year, he has developed and taught courses on IBM Storage Scale and IBM Elastic Storage® and IBM Storage Scale System. From 2008 to 2015, Roger was recognized as an IBM Champion for his contributions to the IBM Data Management community. In 2012, he was recognized as an IBM Developer Master Author, Level 2 (for his contributions to the IBM Developer community). In 2021, he was recognized as an IBM Redbooks Platinum Author.

Sanjay Sudam is the Senior Storage Sales Specialist for Strategic AI Solutions at IBM. He works with clients and partners to develop strategic solutions that deliver business value and technical excellence. He presents frequently at conferences and events about how to adopt leading-edge technology in enterprises.

Lindsay Todd is a Principal Storage Technical Specialist with the IBM Advanced Technology Group. He has a PhD in Computer Science from Rensselaer Polytechnic Institute, where he also worked as an adjunct faculty member and systems programmer supporting high-performance and research computing and becoming familiar with GPFS (now IBM Storage Scale). Lindsay has been with IBM since 2014. He still explores the usage of and builds innovative architectures for IBM Storage Scale, helping customers understand and use it to build solutions to their business problem.

Joanna Wong is a Principal Storage Sales Specialist with IBM Americas. She has extensive experience in HPC application optimization, large systems scalability performance, and solution architecture implementation. She recently has focused on Storage for Data and AI for IBM Storage Scale, which is a global data platform for application modernization. She has an AB degree from Princeton University, a PhD degree in Physics from Cornell University, and a MBA degree from the University of California, Berkeley.

A Special Thank you to the following for their assistance and contributions:

Nikhil Khandelwal - Software Engineer IBM Storage Scale

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 IBM Redbooks 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, IBM Redbooks
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on LinkedIn:

  https://www.linkedin.com/groups/2130806

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

  https://www.redbooks.ibm.com/subscribe

- Stay current on recent Redbooks publications with RSS Feeds:

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

1

Introduction and technical overview

IBM Storage Scale has a long history of supporting NVIDIA solutions. In 2018, IBM and NVIDIA delivered the Summit and Sierra SuperComputer systems that powered 27,648 Tesla GPUs that were fed from a 200 PB IBM Storage Scale namespace at over 2.5 TBps. Later, IBM collaborated with NVIDIA to support the DGX-2H and A100 SuperPOD Reference Architectures (RAs). IBM continues to support updated SuperPOD RA environments with H100, H200, and B200 components.

This paper seeks to document the necessary information, architecture deployment, and tuning that are needed for IBM Storage Solutions with NVIDIA DGX SuperPOD environments.

This chapter describes the following topics:

1.1 What is IBM Storage Scale

IBM Storage Scale, formerly known as IBM Spectrum IBM Storage Scale or General Parallel File System (GPFS), is a highly scalable, software-defined, and Portable Operating System Interface (POSIX)-compliant distributed file storage solution that provides high-performance concurrent access to a single file system or set of file systems from multiple nodes. The nodes can be Storage Area Network (SAN)-attached, network-attached, or a combination of the two. This approach enables high-performance access to a common set of data to support scale-out or scale-up data storage solutions with native high availability (HA). Thus, IBM Storage Scale enables organizations to build an enterprise-resilient, global data platform for artificial intelligence (AI), high-performance computing (HPC), intensive data analytics, or other demanding workloads.

Originally released in 1998, IBM Storage Scale makes it possible to store a “single copy of data” that can be accessed through industry standard protocols, including high-speed POSIX file access, the Network File System (NFS) protocol, the Common Internet File System (CIFS) and Server Message Block (SMB) protocol, Apache Hadoop Distributed File System (HDFS), and Amazon Simple Storage Service (S3) interfaces. Beyond global data access, IBM Storage Scale also delivers global data abstraction services that seamlessly connect many data sources (including third-party sources) across multiple locations. With this approach, users can consolidate various sources of data into one global namespace (A *namespace* is the tree structure of a file system; in an operating system, an example of a namespace is a directory). This namespace also enables efficient space usage because as data does not need to be moved (or copied) from one location to another one to facilitate different data access methods.

IBM Storage Scale can be deployed on many hardware platforms, including x86, IBM Power, IBM Z® mainframes, ARM-based clients, virtual machines, and Kubernetes. IBM Storage Scale also provides the foundation for IBM Fusion (formerly known as IBM Storage Fusion) and the IBM Fusion HCI appliance, both of which enable organizations to quickly and flexibly deploy Red Hat OpenShift applications and IBM watsonx. Users can create a cluster of IBM AIX® nodes, Linux nodes, Windows server nodes, or a mixture of the three. IBM Storage Scale may run on virtualized instances that provide common data access in environments and use logical partitioning. Multiple IBM Storage Scale clusters can share data within a single, central location or across wide area network (WAN) connections.

Here are some of the key features of IBM Storage Scale:

- Highly scalable performance and capacity: Enables both scale-out and scale-up solutions on-premises, in the cloud, or as a hybrid.
- Plug-and-play data services: Users can use software services that are simple to configure and manage while providing the capabilities that are needed to handle AI, HPC, and enterprise-level workloads.
- Cyber resilient data: Users can use comprehensive resiliency while protecting data with IBM SafeGuarded Copy and incorporating the optional IBM CyberVault component.
- Native NVIDIA support: IBM Storage Scale supports NVIDIA GPUDirect Storage (GDS), NVIDIA DGX SuperPOD, and NVIDIA AI solutions through simple configurations.
- Native Cloudera support: IBM Storage Scale users can make the most of an IBM Storage Scale Hadoop connector that supports Cloudera Analytics and Apache Spark.

- High-performance object storage: Applications and users have a choice of interfaces to access the data that they need with a high-performance S3 interface for file and object data.
- Kubernetes and containers: IBM Storage Scale users have access to a high-performance interface that makes it simple to deploy and grow storage for containers.

1.1.1 Basic structure of IBM Storage Scale

IBM Storage Scale can be deployed as software-defined storage, as an integrated hardware and software solution (IBM Storage Scale System), or as a cloud service. This versatile offering can be deployed on Network Attached Storage (NAS), Storage Area Network (SAN) storage, Internet Small Computer System Interface (iSCSI) SAN storage (which uses Transmission Control Protocol or (TCP) to transmit SCSI commands, enabling remote storage to connect to servers as though they were local disks), Non-Volatile Memory Express over Fabrics (NVMe-oF), or storage-rich servers like NVIDIA DGX. Storage can also be pooled to create a common global namespace, simplifying access to storage options and providing data protection through erasure coding or replication.

IBM Storage Scale presents itself as a clustered file system that is defined over one or more nodes. Each node that is used to form the cluster runs three basic components: administration commands, a kernel extension, and a multithreaded daemon. Interaction between nodes at the file system level is limited by locks and control flows to maintain data and metadata integrity in the parallel environment.

The administration commands are programs and scripts that control the configuration and operation of IBM Storage Scale. These commands typically begin with the letters “mm.” For example, the administrative command that is used to display file system attributes is `mmlsfs`. IBM Storage Scale administration commands can be issued from any node in a cluster, If the execution of a command requires tasks to be performed on other nodes, the order to perform those tasks are automatically sent to those nodes.

The kernel extension provides the interfaces to the operating system that are needed to register IBM Storage Scale as a native file system. When applications make file system calls to the operating system, those calls are forwarded to the IBM Storage Scale file system kernel extension. The kernel extension either fulfills these calls by using resources that are already available, or it sends a message to the IBM Storage Scale daemon to complete the request.

The multithreaded daemon performs all input/output (I/O) operations and buffer management for IBM Storage Scale, including *read-ahead* for sequential reads and *write-behind* for all non-synchronous writes. I/O operations are protected by IBM Storage Scale token management to help ensure consistency of data across all nodes in the cluster. In addition to managing local I/O, the daemon communicates with instances of the daemon on other nodes to coordinate configuration changes, data recovery, and parallel updates of the same data structures.

The Network Shared Disk (NSD) component of IBM Storage Scale provides a method for cluster-wide disk naming and high-speed access to data for applications running on nodes that do not have direct access to disks. NSDs can be physically attached to every node in a cluster, or they can have their data served through an NSD server (node), which provides a virtual connection. Up to eight NSD servers can be specified for each NSD, so if one server fails, the next server on the list automatically takes control.

1.1.2 IBM Storage Scale fundamental components

Central to understanding IBM Storage Scale are the concepts of *node* and *cluster*. An IBM Storage Scale cluster is a group of systems, which are called nodes, which are configured to work together. Every node has a common view of their data. Nodes in a cluster are tightly coupled, that is, they trust each other nodes' authentication of users. Clusters may have file systems and may share access to file systems with other authenticated remote clusters. Most importantly, the cluster helps ensure that every node has a consistent view of the file systems and file system data, even when processes on more than one node are actively accessing a file system or even the same file. The processes all access the file system with the same consistency as though they all ran on a single gigantic system.

Figure 1-1 shows the relationship of clusters and nodes.

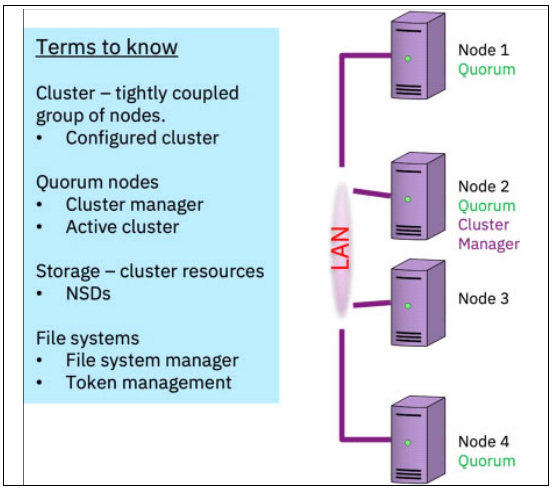


Figure 1-1 Clusters, nodes, and node roles

Each node requires the IBM Storage Scale software to be installed on them. The local state for each node is kept in the directory `/var/mmfs`. A kernel extension, which is tied into the operating system's vnode and Virtual File System (VFS) interfaces, intercepts POSIX API system calls and either satisfies them directly or forwards them to a local daemon that is named `mmfsd`. Some of the services that the `mmfsd` daemon performs are as follows:

- I/O and buffer management
- Reading-ahead for inferred I/O patterns (such as sequential and striped reads)
- Write-behind for files that are not specified as being written synchronously

It is not enough to install IBM Storage IBM Storage Scale software on nodes: The nodes must also be configured into a cluster to communicate successfully. Once the IBM Storage Scale software has been installed, configured, and is running, the nodes communicate with each other to form a running *active cluster*. Nodes in the active cluster may actively use the cluster's file systems. One of the nodes in the active cluster serves as the *cluster manager* and tracks the membership of the active cluster.

Because failures happen, the cluster manager uses *disk leases* to enable access for active cluster nodes to storage for a limited time. Nodes must renew their disk leases regularly, and if a node does not renew its lease before it expires, that node is expelled from the active cluster and loses its access to the storage. Generally, failure to renew a disk lease indicates a network problem, but it might also be indicative of a failed or over-committed node.

When a failure occurs, the cluster also needs a way to recover from a situation where there is no cluster manager running. Rather than configure a single node to be the sole cluster manager, several cluster nodes are configured to be *quorum nodes*. The quorum nodes communicate with each other and elect one of their number to be the cluster manager. Generally, a small, odd number of quorum nodes are configured, say 3, 5, or 7. Most of the quorum nodes must be able to communicate to hold this election. The cluster protocol helps ensure that there will never be more than one cluster manager running at any time. In addition to electing the cluster manager, the quorum nodes also persistently maintain a copy of the configuration state for the entire cluster.

IBM Storage Scale is a parallel clustered POSIX file system:

POSIX The cluster follows the Institute of Electrical and Electronics Engineers (IEEE) Std 1000.3 standard's operating system interface and environment and the file system semantics that are described.

Clustered All nodes in the cluster may work with the same storage volumes concurrently, with each participating in the implementation of the file system.

Parallel File systems and even files may be striped over multiple storage volumes, and operations may engage more than a single volume concurrently. IBM Storage Scale stripes file system metadata over multiple storage volumes.

Note: POSIX is a set of standard operating system interfaces that are based on the UNIX operating system (OS) and designed to maintain compatibility among different OSs. The most recent POSIX specification (IEEE Standard 1003.1-2017) defines a standard interface and environment that can be used by an OS to provide access to POSIX-compliant applications. The standard also defines a command interpreter (shell) and common utility programs. Essentially, any software application that conforms to POSIX standards should be compatible with and capable of running on any operating system that adheres to POSIX standards.

Storage volumes that are owned by a cluster are called NSDs for historical reasons. An NSD can be provided in many ways: a disk in a storage-rich server, a disk partition, a SAN-attached shared volume or LUN, and others. For the IBM Storage Scale System, NSDs are formed from vDisks, which are a software-RAID abstraction that is created by an IBM Storage Scale RAID.

Systems that can directly access NSDs can share those NSDs over a TCP or Remote Direct Memory Access (RDMA) (InfiniBand or RDMA over Converged Ethernet (RoCE)) network with the other systems in the IBM Storage Scale clusters. Systems sharing NSDs are called *NSD servers*, and systems accessing such NSDs are called *NSD clients* (or clients). The data nodes in an IBM Storage Scale System, for example, are NSD servers sharing their vDisks with the NSD clients in the cluster. An NSD server is an IBM Storage Scale node sharing NSDs. This ability is native to IBM Storage Scale nodes, and being an NSD server does not confer any special capabilities to the server. In particular, NSD servers are not file servers, and operations like replication, compression, and others are not performed on the NSD server.

Figure 1-2 shows how NSD servers make an NSD available to client systems. A single NSD might be accessible through more than one NSD server (in this case, there is redundancy), so if one of the NSD servers is down because of failure or maintenance, the data remains available through another NSD server. An IBM Storage Scale System always has redundancy because each IBM Storage Scale System building block always consists of a pair of data nodes.

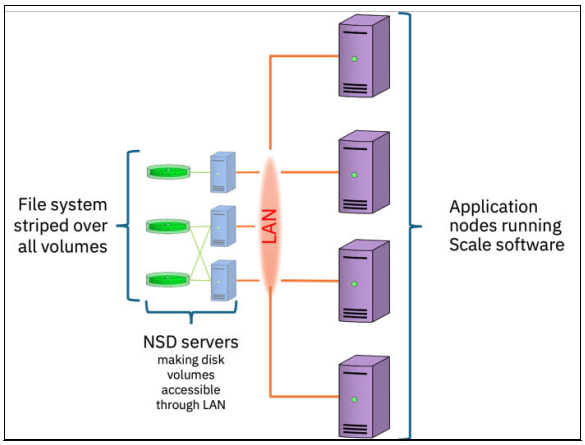


Figure 1-2 NSD server example

An IBM Storage Scale file system is configured from a set of NSDs and mounted on the nodes of the cluster. File data is striped over all the NSDs composing that file system, and even a single file may be striped over multiple NSDs. A node writing a large file potentially engages many NSDs concurrently, and reading large files potentially engages many NSDs concurrently to satisfy that request. This concurrency (or parallelism) is the key to the IBM Storage Scale outstanding performance. Uniquely for parallel file systems, IBM Storage Scale can stripe file system metadata over multiple NSDs.

NSDs can be added or removed from file systems even when they are in use. Adding NSDs increases the capacity of the file system. Adding NSDs also increases the opportunities for parallelism, enabling performance to also increase.

Note: Removing NSDs might cause data availability problems if the removal is not properly planned.

An IBM Storage Scale cluster may have more than one file system, each striped over a different set of NSDs. When a file system is mounted on nodes in the cluster, a *file system manager* is appointed for each file system on one of the nodes that is designated as a *manager*. Together, all the manager nodes serve as *token managers* for every file system, and work with all the cluster nodes to implement the IBM Storage Scale *distributed lock manager* (DLM). The DLM is what gives all the nodes of the cluster a single-system file consistency, even when many nodes are accessing the same file.

1.1.3 Storage and client clusters

An IBM Storage Scale cluster can provide controlled access to file systems that it is hosting to other clusters. Each cluster can be administered and updated separately, but still share access to the file systems as though they were one single, larger cluster. The DLM handles the authentication between clusters for multicluster environments. Each IBM Storage Scale cluster has a key pair for authentication between clusters nodes. For multi-cluster configurations, the administrators of the clusters must exchange the public key of the key pair to enable the clusters to communicate with each other.

The cluster that shares its file systems with other file systems is called the *owning cluster* or the *storage cluster*. A cluster that accesses those file systems is an *accessing cluster* or A *consumer* or *client cluster*.

Figure 1-3 shows an owning cluster and a pair of accessing clusters.

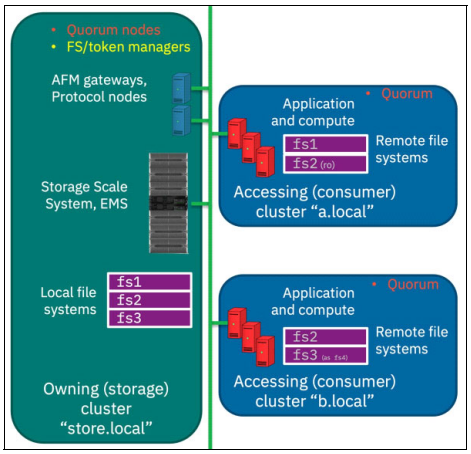


Figure 1-3 Sharing file systems to remote clusters

Owning clusters can specify which file systems are available to which accessing clusters, and whether that access is full or read-only. Access configuration can be customized and can be restricted down to a file system or file set directory level.

1.1.4 IBM Storage Scale Global Data Platform services

The IBM Storage Scale Global Data Platform is fundamental to supporting AI and machine learning (ML) workloads because it provides four key sets of capabilities:

- Data Access Services
- Data Acceleration Services
- Data Abstraction Services
- Data Assurance Services

These four services each provide a key set of features and functions that enable greater storage efficiency that correlates to greater AI and ML efficiency.

Figure 1-4 shows an overview of IBM Storage Scale Data Access.

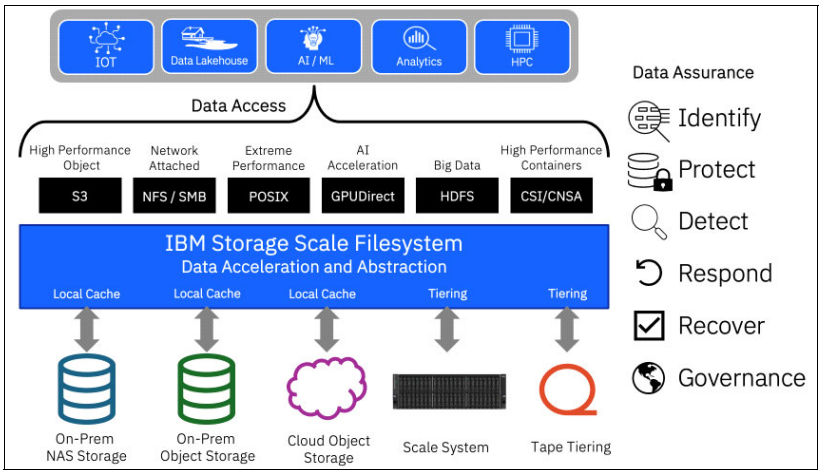


Figure 1-4 IBM Storage Scale Data Access overview

Data Access Services

IBM Storage Scale Data Access Services enable an organization to maintain a “single source of truth” for datasets within the IBM Storage Scale file system namespace, even when multiple workloads and applications access the same files through different protocols. IBM Storage Scale supports multi-protocol data access through different protocols, including the native IBM Storage Scale POSIX file system client and the GPUDirect Storage capability. Cluster Export Services (CES) can be enabled to export IBM Storage Scale through the Network File System (NFS) v3 or v4 protocol, the Common Internet File System (CIFS) and Server Message Block (SMB) protocol that is used generally with Microsoft Windows, the Amazon Simple Storage Service (S3) object-based protocol, and the IBM Storage Scale Apache Hadoop Distributed File System (HDFS) transparency layer. Also, IBM Storage Scale supports a Container Storage Interface (CSI) driver that enables IBM Storage Scale to be accessed by Kubernetes clusters alongside Container Native Storage Access (CNSA) that is used with Red Hat OpenShift immutable operating system clusters.

Data Acceleration Services

In addition to providing robust multi-protocol data access, the IBM Storage Scale parallel architecture is designed to provide linear scaling of both performance and capacity. As a result, it can deliver submillisecond latency, multiple terabytes per second (TBps) of throughput, and tens of millions of input/output operations per second (IOPS) while addressing datasets in the exabyte range. This approach helps eliminate data silos because it enables multiple application programming interfaces (APIs) to access the same data concurrently. It also enables AI and analytics workloads to use the output from one application to drive results to another application.

Also, IBM Storage Scale has a technology preview of its Data Acceleration Tier (DAT), which uses NVMeoF technology to provide a high-IOPS storage tier that is designed specifically for AI and ML workloads.

Data Abstraction Services

IBM Storage Scale provides a unique and powerful capability that is called Active File Management (AFM), which enables IBM Storage Scale to serve as a data caching and abstraction layer to external file and object data sources. This unique capability allows for IBM Storage Scale to transparently access data sources and accelerate slower external data sources where data access might be bottlenecked by the media type, network speed, or distance. AFM can cache data from on-premises or in the cloud and eliminate the need to copy data, which can result in multiple file versions and unnecessary redundancy. Therefore, IBM Storage Scale can provide connectivity from multiple data sources and multiple locations, bringing together data from IBM and other storage environments.

This data abstraction capability can extend the IBM Storage Scale file system namespace over geographically separated environments due to AFM's asynchronous data movement. Data centers can access remotely stored data faster by transparently caching that data locally as needed, increasing application agility from the edge to core to cloud. Also, organizations can rapidly grow their storage infrastructure by adding storage capabilities where they can be most efficiently deployed without regard to location. More importantly, organizations can protect their existing IT infrastructure investments and lower costs from now on by using IBM Storage Scale to create an open ecosystem of storage options that use multi-vendor and multi-cloud resources.

Data orchestration adds a layer of intelligent data management and mobility for both AFM caching relationships and Information Lifecycle Management (ILM) data tiering within the IBM Storage Scale file system namespace. Data orchestration enables the integration of a metadata catalog such as IBM Fusion Data Catalog, adding the ability to scan external data sources and provide a list of pertinent files based on tags and metadata attributes. This catalog can be integrated with AFM to cache only the required datasets from external sources into the IBM Storage Scale high-performance file system layer. Also, data orchestration can integrate with ILM, enabling files on colder storage to be tiered up to faster storage, such as from hard disk drives to Non-Volatile Memory Express (NVMe) or tapes to NVMe. The ability to add intelligence to AFM and ILM is critical for AI and ML workloads where specific data sets are needed for training, tuning, or inference. Also, data orchestration can help optimize data storage costs by enabling older, colder files to be intelligently tiered down to more cost-effective tiers or integrated with IBM Storage Scale file system compression for data reduction savings.

Data Assurance Services

Ensuring that data is protected within the IBM Storage Scale file system namespace is critical to supporting mission-critical applications. IBM Storage Scale employs a multitude of data resiliency capabilities protecting against system and site failures, data corruption, and malicious activity. IBM Storage Scale follows the NIST framework for security with specific capabilities for each step of the framework (Identify, Protect, Detect, Respond, Recover, and Governance).

Because data resilience is fundamental to the IBM Storage Scale architecture, it supports various multi-site configurations by using synchronous or asynchronous communication. These configurations can be clustered together to form a larger cluster that serves as a global data platform with exabytes of capacity.

With AFM, any single IBM Storage Scale cluster can be configured by using nodes and storage across two or more data centers. To enhance data resilience, a single IBM Storage Scale cluster can be configured to use nodes and storage that are “stretched” across two separate data centers. In this case, the cluster components are connected by a wide area network (WAN) that enables file systems in the cluster to be concurrently available and accessible at both sites. These file systems can use “failure group” replication to help ensure that both sites have a current replica of all data so that one site always remains available, even if the other site (or link to the site) fails. The result is a highly available file system with active-active synchronous replication.

IBM Storage Scale also includes the IBM Safeguarded Copy capability, which can be used to create isolated, immutable snapshots of data on a regular schedule so that operational data can be rapidly recovered in the event of a cyberattack or other potential loss of data.

Also, IBM Storage Scale offers File Audit Logging and Watch Folders that can be integrated into security information and event management (SIEM) for analysis of known malicious access patterns, and then SIEM can initiate an automated workflow that reverts to a previously known-good state (snapshot).

1.1.5 Cluster Export Services

IBM Storage Scale can provide a global namespace because it supports many Data Access protocols that can be used to bring different types of storage together. Cluster Export Services (CES) is the function that enables the use of Network File System (NFS), Simple Storage Service (S3), Server Message Block (SMB), and Hadoop Distributed File System (HDFS) protocols. Because CES has specific hardware and software requirements, its code must be installed on nodes in an IBM Storage Scale cluster that are designated to run the CES software stack. (These nodes are referred to as CES nodes or protocol nodes.) CES nodes are not attached to external storage; instead, they interact with storage that is remotely mounted.

The CES infrastructure is responsible for the following processes:

- Managing the setup for HA clustering that is used by the NFS, S3, SMB, Swift Object, or HDFS protocols. Regardless of which protocols are chosen, all are accessible through a set of floating internet protocol (IP) addresses that are stored in a CES address pool. This pool is considered floating because each address can move independently among all designated CES nodes. In the event of a CES node failure, accessibility to all protocols is maintained as the IP addresses automatically move from the failed CES node to a healthy one.
- Monitoring the health of these protocols on the protocol nodes and raising events or alerts during failures. Monitoring helps ensure that CES IP addresses are assigned to the appropriate node and that the processes that implement the services in the CES cluster are running correctly. On node failure detection, CES marks the node as temporarily unable to participate in the CES cluster and reassigns its IP addresses to another node.
- Managing the IP addresses that are used for accessing these protocols by including failover and failback of these addresses in the event of CES node failures. To avoid impacting the clients of other protocols during an HDFS failover, clients that use NFS, S3, SMB, and Swift Object protocols should not share an IP address with clients that are used by the HDFS service.

CES is used to support workflows that require NFS, S3, SMB, or HDFS access. Common use cases include the following ones:

- Data ingest, where the workload consists primarily of write operations. Examples include Internet of Things (IoT) sensor data collection, collection of streaming data from cameras (traffic cameras, security cameras, and other), genome sequencing, and Extract, Transform, and Load (ETL) operations.
- Data export, where the workload consists primarily of read operations. For example, moving processed data to an external location.
- Interactive access to data, where the workload consists primarily of read operations. Examples include reading analysis results or performing data visualizations.
- Low-performance data analysis, where the workload can be either predominately read or predominately write, depending on the application. For example, batch processing or grid computing.

To successfully use CES, IBM Storage Scale users must consider function prerequisites, setup and configuration, failover/failback policies, and other management and administration requirements.

1.1.6 Container Storage Interface

Although containers were originally conceived as having storage that was stateless, it soon became apparent that containerized applications needed to retain data for longer periods. So, many ways for meeting the containers persistent storage needs (largely represented by Docker and the container orchestrator Kubernetes) were developed. The most used method was a Kubernetes volume plug-in, which extended the Kubernetes volume interface to support a block or file storage system.

Initially, Kubernetes volume plug-ins were “in-tree,” meaning that their code was part of the core Kubernetes code and included with the core Kubernetes binary files. As a result, vendors that wanted to add support for their storage systems were forced to align with the Kubernetes release process. Unfortunately, third-party storage code caused reliability and security issues in the core Kubernetes binary files. Moreover, the third-party code that was provided was often difficult (and sometimes impossible) for Kubernetes developers to test and maintain.

CSI is a standardized mechanism for exposing arbitrary block and file storage systems to containerized workloads in Container Orchestration Systems (COS) like Kubernetes and Red Hat OpenShift. With the adoption of CSI, the Kubernetes volume layer became truly extensible; third-party storage providers could write and deploy plug-ins exposing storage systems in Kubernetes without having to touch the core Kubernetes code. This approach gave (and continues to give) Kubernetes users more options for storage and makes the system more secure and reliable. CSI decouples persistent storage development efforts from core cluster management tools, enabling the rapid development and iteration of storage drivers across the cloud native ecosystem.

Essentially, CSI is an application programming interface (API) specification that enables developers to build custom drivers that handle the provisioning, attaching, and mounting of volumes in containerized workloads. If a driver correctly implements the CSI API specification, it can be used in any supported COS. CSI was first released in December 2017, and 2 years later it became the primary volume plug-in system for Kubernetes. (It was introduced as alpha in Kubernetes 1.9, promoted to beta with Kubernetes 1.10, and became generally available in Kubernetes 1.13).

The IBM Storage Scale CSI driver enables IBM Storage Scale to be used as persistent storage for stateful applications running in Kubernetes clusters. Through this driver, Kubernetes persistent volumes (PVs) can be provisioned from IBM Storage Scale storage, which makes it possible for containerized applications and stateful microservices like containerized databases (MongoDB, PostgreSQL, and others) to use everything IBM Storage Scale has to offer.

The IBM Storage Scale CSI driver provides the following functions:

- Static provisioning: The ability to use existing directories and file datasets as persistent volumes.
- Lightweight dynamic provisioning: The ability to create directory-based volumes dynamically.
- File set-based dynamic provisioning: The ability to create fileset-based volumes dynamically.
- Multiple file systems support: Volumes can be created across multiple file systems.
- Remote mount support: Volumes can be created on a remotely mounted file system.
- Operator support for simpler deployment, upgrades, and cleanup.
- IBM Storage Scale volume access mode support: ReadWriteMany (RWX) and ReadWriteOnce (RWO).

The IBM Storage Scale CSI driver also makes Representational State Transfer (REST) API calls on an IBM Storage Scale Storage System to perform provisioning requests.

Active File Management (and AFM-DR)

Active File Management (AFM) enables data sharing across storage clusters, even if the network is unreliable or has high latency. Essentially, it is technology that makes data that physically resides “over there” appear to be “over here”, that is, it enables applications and users to see data that is physically stored at a remote location in such a way that it appears to be local.

IBM Storage Scale supports the concept of multiple clusters sharing resources. Thus, an IBM Storage Scale cluster can mount and export one or more file systems to selected remote clusters, and those clusters can then mount and access the file systems that are exported. Often, IBM Storage Scale clients that use POSIX have high-performance access to the remote file systems. However, the greater the distance between clusters, the greater the amount of latency. AFM reduces this latency by caching remote data into the local IBM Storage Scale file system.

AFM defines a “Home” as the source for the data in am AFM relationship and the “Cache” as the cached version of data in the IBM Storage Scale file system from the home. In these caching relationships, AFM supports different caching modes:

- Read-Only: The Cache is a read-only version of the Home data and cannot be modified.
- Local-Updates: Similar to read-only, but the data in the Cache can be modified. Changes are NOT sent back to the Home.
- Single-Writer: Data in the Cache can be modified and be sent to the Home. Changes on the Home are never sent back to the Cache.

- Independent-Writer: Changes in both the Cache and Home are asynchronously updated so that if a file is changed on the Home, it appears in the Cache and vice-versa. There is no cache consistency or locking between the Home and the Cache.
- Manual Updates: This caching mode works only for object storage. In this mode, files must explicitly be marked for upload or download to help ensure that egress fees can be contained and that only required data is cached.

To illustrate, suppose that an organization has an IBM Storage Scale compute cluster with a file system that is named `/scale/fs1` at its headquarters (primary site) that is used to store company data. They also have a manufacturing facility (secondary site) that has it sown IBM Storage Scale compute cluster with a file system that is named `/scale/fs2` that receives data being streamed from Internet of Things (IoT) sensors that are placed on equipment throughout the manufacturing facility. Because the sites are several hundred kilometers apart, the compute clusters are connected to each other through a WAN. So, if the remote site needs data from the `/scale/fs1` file system, the latency between the sites can cause problems.

By using AFM, the organization can create cache file sets at the secondary site to cache the parts of the `/scale/fs1` file system it needs by using NFS version 3 or 4 or Network Shared Disks (NSDs) as the transport protocol. (File sets provide a way to partition a file system to organize data into sets or enable administrative operations to be performed at a finer granularity.) Once the cache file sets are set-defined, everything that is needed in the `/scale/fs1` file system is visible and accessible to the remote system at the secondary site. However, only files that are actively being used to consume space in the cache. (The data is retrieved on demand; only when an application attempts to work with a file that is stored on the primary site is the requested data brought into the cache.) Usually, AFM periodically checks to ensure that the cached data is not stale and refreshes it, if necessary. There are several AFM configuration settings that are available to fine-tune this “revalidation” mechanism.

The caching capabilities of AFM are not limited to working with files on another IBM Storage Scale cluster file system. AFM can also cache object storage by using S3 endpoints, and these endpoints can be on-premises or in the cloud; the effect is to provide a file system interface for object data. Thus, AFM enables users to connect to Amazon S3 containers, IBM Cloud®, Microsoft Azure, and Google Cloud Platform, and any S3-compatible object storage.

AFM can be used to replicate data for disaster recovery (DR) purposes. AFM-based asynchronous disaster recovery (AFM-DR) provides IBM Storage Scale users with a file set-level replication DR solution that can perform the following functions:

- Provide business stability during a disaster.
- Restore business stability after a disaster.
- Endure multiple disasters.
- Minimize data loss due to a disaster.

The idea behind AFM-DR is that the primary site contains a writeable primary file set that is associated with a read-only file set at the secondary site. Data that is written to the primary file set is continuously replicated to the secondary file set asynchronously. The normal AFM updating mechanism populates the “read-only” file set on the secondary site. However, because these tasks can take a significant amount of time to complete, an alternative is to “truck” data (by using a non-AFM mechanism like `rsync` or tape) to pre-populate the secondary file set and establish the AFM relationship afterward. Thereafter, AFM helps ensure that the secondary file set is always kept consistent with the primary file set.

But what happens if an event occurs at the primary site that causes replication to stop? In that event, the secondary site can be converted into a functional primary site and applications can be redirected to it. Although the work of redirecting applications is outside the scope of IBM Storage Scale or any storage solution, AFM-DR performs the tasks that are required to make the “read-only” DR copy of the data “writeable” and it tracks all changes that are made.

In a true DR scenario, an organization might run their business for days, weeks, or even months from the secondary (failover) site. But usually, the original primary site eventually comes back online. At that point, the organization has two options:

- Reverse the primary and secondary roles so that the original secondary site (which is now the acting primary) replicates its data to the original primary site (which now acts as the secondary). Often, this way is the fastest way to restore business stability after a disaster.
- The original primary site asks the original secondary site (now the acting primary) for all the updates it has missed. (This situation can happen repeatedly.) Then, when both sites are fully synchronized, the acting primary can fail back to the original primary (requiring applications to be redirected once more) and the original replication relationship can be restored.

Optionally, a peer snapshot can be taken at regular intervals, based on an organization's recovery point objectives (a time-based measurement of the maximum amount of data loss that is tolerable). Then, when the secondary site becomes the acting primary, the active file set can be rolled back to the previous snapshot. This approach is useful when there is a risk that out-of-order updates might corrupt the acting primary.

Information Lifecycle Management

ILM technology enables IBM Storage Scale to physically store data on the most appropriate storage media possible, over the course of its life. With the ILM toolkit, IBM Storage Scale users can manage sets of files and pools of storage, and automate the movement of data from one location to another one. A “storage pool” is a collection of disks or a redundant array of independent disks (RAID) array. Storage pools enable users to create storage tiers by grouping storage devices together, based on performance, locality, or reliability characteristics. For example, one pool might be an enterprise class storage system that hosts high-performance Non-Volatile Memory Express (NVMe) drives, and another pool might consist of disk controllers that host a large set of economical, lower-performing Serial Advanced Technology Attachment (SATA) disks.

Generally, applications are not concerned about where data is (or when it is created) if it is available when needed. Once data is written to an IBM Storage Scale file system, when an application or user attempts to retrieve it, IBM Storage Scale provides the transparency that is required to make the data retrieval operation seamless, regardless of where the data physically is. More importantly because IBM Storage Scale knows exactly where data is, it can move it from one location to another one without having to inform applications and users that the data has been moved.

ILM provides a means to automate file management by using a set of user-defined policies and rules. A policy is a set of rules that describes the lifecycle of data, based on file attributes. A rule is a Structured Query Language (SQL)-like statement that tells IBM Storage Scale what to do with the data for a file in a specific storage pool if the file meets specific criteria. For example, when a data file is created, a placement rule can be used to control the storage pool the data should be created in.

Note: A default pool that is named `system` is where data and metadata is stored by default if no placement rules apply.

As data changes over its lifetime, migration rules can be applied to relocate it, or deletion rules can be applied to remove it. For example, older data that is not frequently accessed but must be kept to comply with data retention policies might be moved to tape when it has not been touched in the last 90 days. Regardless of where data is stored initially or relocated to, applications can continue to access it seamlessly. This approach is how ILM helps ensure that premium storage resources are used efficiently and that a proper balance of premium and less expensive storage resources is maintained.

1.2 IBM Storage Scale System

The business world is being transformed rapidly by artificial intelligence (AI), HPC, analytics, and hybrid cloud technologies. Unlike traditional applications that work with structured data that is stored in databases, these new workloads operate on a vast ocean of unstructured data, such as documents, images, videos, audio recordings, log files, or any other information that can be generated by users, applications, or even machines.

As a result, organizations are having to rethink their data storage strategies and adapt to new ways of working, including how to use AI to unlock the value of their data wherever it is. Information technology (IT) leaders face many new challenges:

- Accessing and analyzing data that is scattered across the globe.
- The increasing time and resources that are needed by AI training and inferencing workloads.
- The cost and scarcity of resources, like NVIDIA graphics processing units (GPUs).

Addressing these challenges requires specialized software and hardware:

- IBM Storage Scale: Software-defined file and object storage for both structured and unstructured data.
- IBM Storage Scale System 6000: A hardware implementation of IBM Storage Scale software, optimized for the most demanding AI, HPC, analytics, and hybrid cloud workloads.
- IBM Storage Scale System 3500: A hardware implementation of IBM Storage Scale software, which is designed for organizations requiring an enterprise-ready, entry-level, or mid-level Storage System

1.2.1 What is the IBM Storage Scale System

IBM Storage Scale System (formerly known as IBM Elastic Storage System (ESS)) combines IBM Storage Scale software with a pre-tested, storage hardware platform that is the fastest and most flexible way for organizations to build and deploy a global data platform around their file and object data. It uses the power of IBM Storage Scale together with Non-Volatile Memory Express (NVMe) flash technology to deliver high-performance storage for data analytics, AI, and high-performance file and object use cases.

To optimize performance, the IBM Storage Scale System is designed to scale to thousands of nodes and yottabytes (YBs) of capacity. IBM Storage Scale System runs IBM Storage Scale RAID erasure coding, which provides a petabyte-capable storage solution that deploys quickly, lessens the impact of large drive failures, lowers drive rebuild times, and provides consistent high performance. In millions of hours of production usage, IBM Storage Scale System has reliably demonstrated five 9s (99.999%) of availability. Installations and updates are delivered by containerized software, which speeds and simplifies the maintenance process.

IBM Storage Scale System runs IBM Storage Scale RAID erasure coding, which provides data efficiency, consistent high performance, mitigation of storage hardware failures, intelligent monitoring, and dynamic tuning of IBM Storage Scale System and IBM Storage Scale data. Installations and updates are delivered by containerized software that speeds and simplifies the maintenance process.

1.2.2 IBM Storage Scale System 6000 overview

The IBM Storage Scale System 6000 is a high-performance storage solution that is based on the IBM Storage Scale file system technology, which enables the highest levels of performance in a compact and efficient form factor. The IBM Storage Scale System 6000 is a scalable building block for deploying IBM Storage Scale, enabling the ability to scale up the capacity with the addition of SAS-attached Just a Bunch of Disks (JBODs) or scale out the capacity and performance with the addition of additional IBM Storage Scale System 6000 building blocks. The IBM Storage Scale System 6000 uses a central x86 management server that provides a GUI, APIs, health monitoring, call-home, system management, and performance monitoring capabilities. The IBM Storage Scale System 6000 is designed to support the most demanding of workloads, including AI and ML model training, inference and fine-tuning, and traditional HPC workloads.

Figure 1-5 shows the IBM Storage Scale System 6000 front view.

Figure 1-5 IBM Storage Scale System 6000 front view

Key features of the IBM Storage Scale System 6000

Here are the key features of the IBM Storage Scale System 6000:

- A 4U chassis with redundant active-active controllers and hardware to maximize uptime with up to five 9s of availability
- Up to 330 GBps read and 155 GBps write throughput with low latency
- Up to 13 million IOPS by using NVMe-oF Data Acceleration Tier (DAT)
- Up to 1 PB usable NVMe or up to 3.4 petabytes-effective (PBe) by using IBM FlashCore® Modules
- Support for both InfiniBand and Ethernet (RoCE and TCP/IP) networking
- Configurable in all-Flash, all-hard disk drive (HDD), or hybrid NVMe and HDD configurations

IBM Storage Scale System 6000 technical details

The IBM Storage Scale System 6000 features redundant active-active I/O canisters that share a centralized mid-plane, which provides redundant I/O paths to both the 48x dual-ported NVMe drives on the front of the chassis along with any additional SAS JBOD enclosures.

To accelerate the processing of globally distributed data, IBM Storage Scale System supports the NVIDIA GPUDirect Storage (GDS) protocol. GDS enables a direct data path for direct memory access (DMA) transfers between GPU memory and storage, avoiding the need for a “bounce buffer” through the central processing unit (CPU). This direct path helps ensure that the GPU, rather than the CPU, has the first and last touch on data that moves between storage and the GPU. The result is an increase in system bandwidth, a decrease in latency, and a reduction in the performance impact and dependence on CPUs to process storage data transfers.

Each of the I/O canisters contains the following components:

- Dual AMD EPYC Genoa 48-Core CPUs that provide a total of 96 cores per canister and 192 CPU cores in the entire IBM Storage Scale System 6000 building-block.
- 768 GB or 1536 GB of DDR4 ECC memory for a system total of either 1.5 TB or 3 TB.
- Up to 4x NVIDIA Mellanox ConnectX-7 Networking adapters (Figure 1-6) in either a single-port octal small form factor pluggable (OSFP) NDR InfiniBand Dual-Ported QSFP112 Virtual Protocol Interconnect (VPI) NDR200/HDR InfiniBand, or 200/100 Gb Ethernet adapters supporting TCP/IP or RoCE.
- Up to 4x SAS4 Host Bus Adapters (HBA) for connecting up to 9x 91-Drive JBODs.

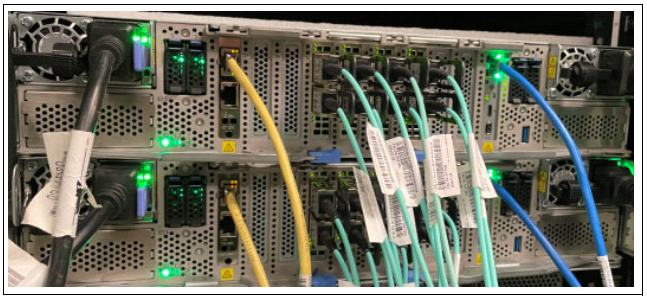


Figure 1-6 Dual-Port ConnectX-7 VPI Networking adapters

The IBM Storage Scale System 6000 supports several solid-state drive types and capacities so that you can customize capacity and performance. The IBM Storage Scale System can be configured with 24x or 48x commodity dual-ported PCIe Gen4 or Gen5 NVMe drives in 3.84 TB, 7.68 TB, 15.36 TB, or 30.72 TB capacity to provide up to 1 PB of usable NVMe capacity. Also, the IBM Storage Scale System 6000 supports IBM FlashCore Module 4 (FCM4) technology, which provides FPGA-based hardware compression on each drive. 24x or 48x FCM4 modules with 19.2 TB or 38.4 TB each can be configured to provide greater than 1 PB of capacity and enable data reduction, enabling an effective capacity of up to 3.4 PBe, assuming a 3:1 data compression ratio.

The IBM Storage Scale System 6000 requires 4x C19/C20 power connections and requires up to 3.2 kW of power under load.

1.2.3 The IBM Storage Scale System 6000 Expansion Enclosure

Note: IBM FlashCore Modules provide an effective capacity that varies based on file compressibility. Also, the performance of FCM drives is lower than commodity NVMe. For the highest performance, use commodity NVMe drives.

The IBM Storage Scale Expansion Enclosure is an enterprise-class, fully redundant storage expansion enclosure that is optimized for the IBM Storage Scale System 6000. Each enclosure can contain up to ninety-one 20 TB or 22 TB self-encrypting SAS HDDs. Up to nine enclosures can be attached to the IBM Storage Scale System 6000 head unit. A single disk enclosure can deliver up to 1.5 PB of usable HDD storage capacity for a system total of approximately 14 PB usable HDD capacity in the max 9x enclosure configuration. Additional disk enclosures can be added after initial deployment if needed. Integration with the ILM policy engine enables you to intelligently tier data between high-performance NVMe storage and high-capacity, lower-cost HDD storage to optimize storage efficiency.

The IBM Storage Scale System 4U91 JBOD expansion enclosure (Figure 1-7) requires 2x C19/C20 power connections and draws approximately 1.4 kW.

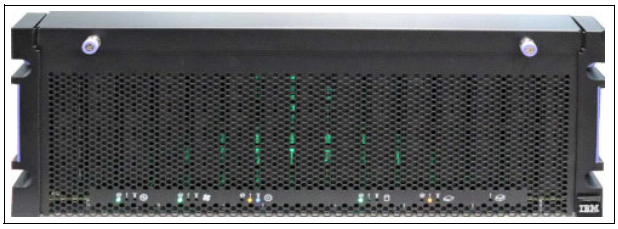


Figure 1-7 4U91 JBOD enclosure

Note: The 4U91 JBOD enclosure uses 24 Gb SAS4 interconnects.

1.2.4 IBM Storage Scale System 3500 overview

The IBM Storage Scale System 3500 is a high-performance storage solution that is based on the IBM Storage Scale file system technology, which enables high-performance storage in a compact and efficient form factor. The IBM Storage Scale System 3500 is a scalable building block for deploying IBM Storage Scale to scale up capacity with the addition of SAS attached JBODs or scale out the capacity and performance with the addition of additional IBM Storage Scale System 3500 building blocks. The IBM Storage Scale System 3500 uses a central x86 management server that provides a GUI, APIs, health monitoring, call-home, system management, and performance monitoring capabilities. The IBM Storage Scale System 3500 was designed to support the most demanding of workloads, including AI and ML model training, inference, fine-tuning, and traditional HPC workloads.

Figure 1-8 shows the IBM Storage Scale 3500 front view.

Figure 1-8 IBM Storage Scale 3500 front view

Key features of the IBM Storage Scale System 3500

Here are the key features of the IBM Storage Scale System 3500:

- 2U chassis with redundant active-active controllers and hardware to maximize uptime with up to five 9s of availability
- Up to 125 GBps read and 55 GBps write throughput with low latency
- Up to 1.2 million IOPS
- Up to 500 TB usable NVMe
- Support for both InfiniBand and Ethernet (RoCE and TCP/IP) networking
- Configurable in all-Flash, all-HDD, or hybrid NVMe and HDD configurations

To accelerate the processing of globally distributed data, IBM Storage Scale System supports the NVIDIA GPUDirect Storage (GDS) protocol. GDS enables a direct data path for direct memory access (DMA) transfers between GPU memory and storage, avoiding the need for a “bounce buffer” through the central processing unit (CPU). This direct path help ensure that the GPU, rather than the CPU, has the first and last touch on data that moves between storage and the GPU. The result is an increase in system bandwidth, a decrease in latency, and a reduction in the performance impact and dependence on CPUs to process storage data transfers.

IBM Storage Scale System 3500 technical details

The IBM Storage Scale System 3500 features redundant active-active I/O canisters that share a centralized mid-plane that provides redundant I/O paths to both the 24x dual-ported NVMe drives on the front of the chassis with any additional SAS JBOD enclosures.

Each of the I/O canisters contains the following components:

- A single AMD EPYC 7642 Rome 48-Core CPU that provides a total of 96 CPU cores in the entire IBM Storage Scale System 3500 building block.
- 512 GB or 768 GB of DDR4 ECC memory for a system total of either 1 TB or 1.5 TB.
- Up to 4x NVIDIA Mellanox ConnectX-7 Dual-Ported QSFP112 VPI NDR200/HDR InfiniBand (Figure 1-9) or 200/100 Gb Ethernet adapters supporting TCP/IP or RoCE.
- Up to 2x SAS4 Host Bus Adapters (HBAs) for connecting up to 8x 102-Drive JBODs.

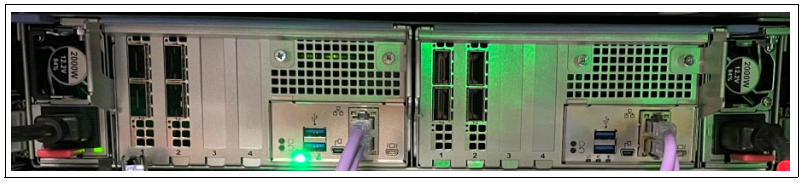


Figure 1-9 Dual-Port ConnectX-7 VPI Networking adapters

The IBM Storage Scale System 3500 supports several solid-state drive types and capacities, which you can use to customize the capacity and performance. The IBM Storage Scale System can be configured with 12x or 24x commodity dual-ported PCIe Gen4 NVMe drives in 3.84 TB, 7.68 TB, 15.36 TB, or 30.72 TB capacity to provide up to 500 TB of usable NVMe capacity.

The IBM Storage Scale System 3500 requires 2x C13/C14 power connections and requires up to 1.5 kW of power under load.

The IBM Storage Scale System 3500 Expansion Enclosure

The IBM Storage Scale Expansion Enclosure is an enterprise-class, fully redundant storage expansion enclosure that is optimized for the IBM Storage Scale System 3500. Each enclosure can contain up to 102x 10 TB, 14 TB, 18 TB, 20 TB, or 22 TB SAS HDDs (the 10 TB, 20 TB, and 22 TB HDDs offer a self-encrypting drive capability). Up to eight enclosures can be attached to the IBM Storage Scale System 3500 head unit. A single disk enclosure can deliver up to 1.7 PB of usable HDD storage capacity for a system total of approximately 14 PB of usable HDD capacity in the max 8x enclosure configuration. Additional disk enclosures can be added after initial deployment if needed. By integrating the ILM policy engine, you can intelligently tier data between high-performance NVMe storage and high-capacity lower cost HDD storage to optimize storage efficiency.

The IBM Storage Scale System 4U102 JBOD expansion enclosure (Figure 1-10) requires 2x C13/C14 power connections and draws approximately 1 kW (1030 W).

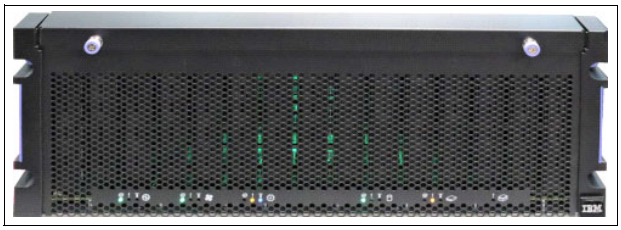


Figure 1-10 4U102 JBOD enclosure

1.2.5 IBM Storage Scale System Utility Node overview

There are several use cases where additional nodes are needed in addition to the IBM Storage Scale System storage itself. These use cases include management of the IBM Storage Scale System building blocks; serving protocols through Cluster Export Services (CES); serving as Active File Management (AFM) gateways for caching use cases; and serving as key management servers. To meet the needs of these use cases, use the IBM Storage System Utility Node (Figure 1-11).

Figure 1-11 IBM Storage Scale System Utility Node

Figure 1-12 shows the rear view of the IBM Storage Scale System Utility Node.

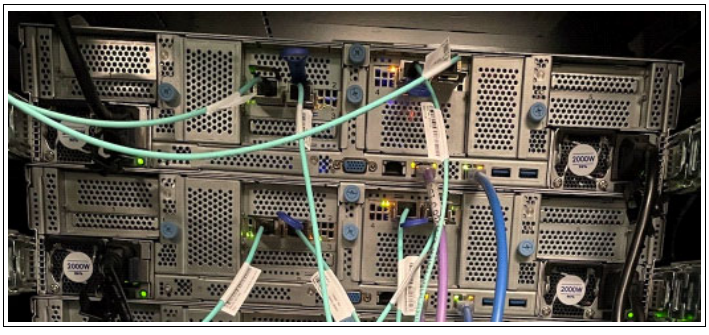


Figure 1-12 IBM Storage Scale System Utility Node rear view

Key features of the IBM Storage Scale System Utility Node

Here are the key features of the IBM Storage Scale System Utility Node:

- A 2U chassis with redundant hardware to maximize uptime
- Customizable memory options based on the use case
- Support for both InfiniBand and Ethernet (RoCE and TCP/IP) networking
- Deployment by using the IBM Storage Scale System deployment toolkit

The IBM Storage Scale System Utility Node contains the following components:

- A Dual AMD EPYC 7313 16-Core CPU that provides a total of 32 CPU cores.
- 128 GB - 1 TB of DDR4 ECC memory (32x 32 GB DIMMs).
- Up to 3x NVIDIA Mellanox ConnectX-7 Dual-Ported QSFP112 VPI NDR200/HDR InfiniBand or 200/100Gb Ethernet adapters supporting TCP/IP or RoCE.
- One 4-Port 10 GbE RJ45 Network Adapter.

The IBM Storage Scale System Utility Node requires 2x C13/C14 power connections and draws approximately 1 kW under load.

1.3 NVIDIA DGX SuperPOD software components

IBM Storage Scale is a parallel file system, and its optimal performance is achieved when multiple processes or workloads are running in parallel. This section describes some of the methods of orchestrating workloads to run in parallel on the IBM Storage Scale 6000, focusing on the NVIDIA DGX SuperPOD architecture, which provides a complete high-performance computing (HPC) and AI cluster solution.

1.3.1 SLURM

The SLURM Workload Manager (previously known as Simple Linux Utility for Resource Management), is an open-source job scheduler that is widely used on some of the world's most powerful clusters. SLURM is responsible for managing the assignment of user workloads to cluster compute resources, enabling users to start, stop, and monitor workloads. In an NVIDIA DGX SuperPOD solution, SLURM integrates with the high-level workload orchestration that is managed by the NVIDIA Base Command software, which handles data set preparation, training, and monitoring. The most common workloads that are managed by SLURM are typically parallel jobs, such as Message Passing Interface (MPI) jobs.

1.3.2 Message Passing Interface

MPI provides a high-level protocol that enables communication between processes running in parallel on a distributed architecture system. MPI is commonly used in HPC and AI computing to provide a standardized and largely hardware-neutral way for processes to communicate over lower-level network or shared memory protocols. MPI processes can send and receive messages, enabling multiple processes to coordinate work in parallel while running on both CPU and GPU resources.

1.3.3 GPUDirect Storage

AI and HPC are driving the usage of fast GPUs in a wider range of applications and on a multitude of platforms, ranging from edge devices to commodity hardware to high-performance supercomputers. As the datasets that are needed for AI and HPC continue to increase in size, the time that is spent loading data for an application can strain application performance. This strain happens because the process of moving data from storage to GPUs has been controlled by the central processing unit (CPU). Traditional reads rely on a server's CPU copying data from a storage resource like Direct Attached Storage (DAS), Storage Area Network (SAN), or Network-Attached Storage (NAS) into a memory buffer, and then writing it out to a network interface for onward transmission. Thus, the data "bounces" from storage to a memory buffer before going to its destination. Traditional writes follow the same route in reverse. This process is shown in Figure 1-13.

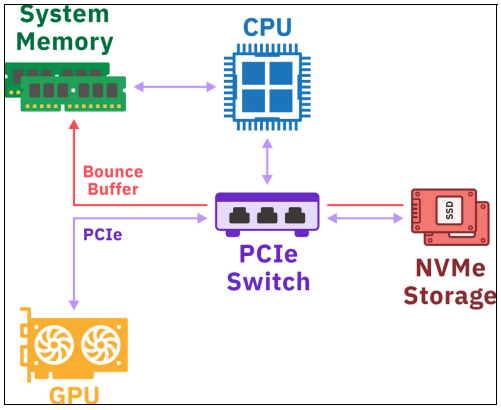


Figure 1-13 Example of the standard path between GPU memory and NVMe drives

Unfortunately, the movement of data through a "bounce buffer" is the leading cause of input/output (I/O) bandwidth bottlenecks. Therefore, as computing shifts from slower CPUs to faster GPUs, GPUs are increasingly being "starved" by I/O operations.

NVIDIA Magnum I/O GPUDirect Storage (GDS) is designed specifically to accelerate data transfers between GPU memory and local or remote storage in a way that bypasses the CPU. It bypasses the bounce buffer stage and sends data directly from storage devices like Non-Volatile Memory Express (NVMe) and NVMe-oF to the destination GPU system's memory. This process is shown in Figure 1-14.

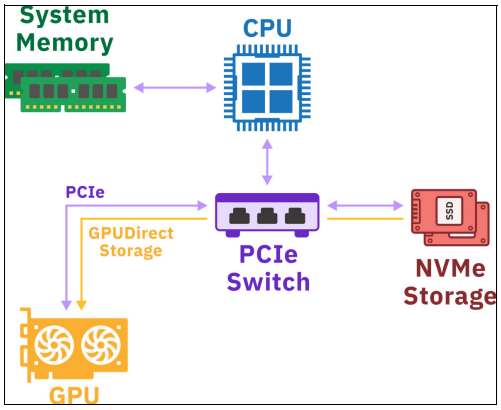


Figure 1-14 Example showing GPUDirect Storage

The benefits are similar to Remote Direct Memory Access (RDMA): higher throughput, lower latency, and reduced CPU impact in data-intensive applications.

GDS is enabled through a direct memory access (DMA) engine near the network interface card (NIC) or storage that transfers data into or out of GPU memory. If a DMA engine in an NVMe drive or elsewhere, near storage can be used instead of the GPU's DMA engine (there is no interference in the path between the CPU and the GPU).

For GDS, storage location does not matter. Storage can be inside an enclosure, within a rack, or connected over the network. One of the major benefits of GDS storage is that fast data access is additive across the various sources that are used. So, where the bandwidth from CPU system memory to GPUs in an NVIDIA DGX-2 is limited to 50 GBps, the bandwidth from many local drives and many NICs can be combined to achieve an upper bandwidth limit of nearly 200 GBps in the same NVIDIA DGX-2.

Note: At the time of writing GDS is supported on Linux x86-64 distributions. It is not supported on Windows.

1.4 Networking

This section describes the following topics:

- Introducing network fabrics in NVIDIA DGX SuperPOD deployments
- Compute fabric
- In-band management
- Out-of-band management network
- Storage fabric
- Interconnect choices for the storage fabric

1.4.1 Introducing network fabrics in NVIDIA DGX SuperPOD deployments

In the NVIDIA DGX SuperPOD, there are four separate network fabrics:

- The compute fabric
- The in-band management network
- The out-of-band management network
- The storage fabric

Although the first three networks are briefly described in this section, the primary focus is on the storage fabric, which is critical for achieving optimal performance with the IBM Storage Scale 6000.

1.4.2 Compute fabric

The compute fabric is a high-performance InfiniBand network that is used by compute nodes and GPUs to support workloads. The NVIDIA architecture employs the concept of a Scalable Unit (SU) to define modular building blocks that can scale based on the system's size. Each SU consists of 32 DGX compute nodes that are aligned on a rail and directly connected to a single leaf switch. Communication between nodes within the same SU occurs with minimal latency, and communication between nodes in different SUs requires additional network hops through the spine layer. For more information, see the NVIDIA documentation about the compute fabric.

1.4.3 In-band management

The in-band management network is used for data tier NFS mounts and managing essential services, such as Base Command Manager, SLURM, and run:ai. Also, it provides access to external services, including code repositories, the NGC registry, and data sources.

1.4.4 Out-of-band management network

The out-of-band Ethernet fabric is a separate network that is not accessible to users. It connects the management ports of all devices, including NVIDIA DGX SuperPOD systems, management servers, storage, networking equipment, and rack power distribution units (PDUs).

1.4.5 Storage fabric

NVIDIA supports a high-performance network fabric that can deliver high bandwidth to each NVIDIA DGX SuperPOD, requiring at least 40 GBps per node. Both InfiniBand and Ethernet networks are supported, and it is a best practice to enable RDMA in both configurations. For more information about NVIDIA recommendations, see their documentation about the storage fabric.

A well-designed network should address potential bottlenecks in the switch fabric during the planning phase. NVIDIA DGX SuperPOD deployment guidance suggests that while no oversubscription should be done for the server nodes, some degree of oversubscription might be an option for the NVIDIA DGX SuperPOD clients. However, any such oversubscription should be carefully considered. This section provides some general guidance about oversubscription.

To achieve optimal performance, consider implementing a network that approaches a theoretically non-blocking network. A non-blocking network is defined as one with full bisection bandwidth, meaning that if the endpoints are evenly divided into two groups, both groups can communicate without bandwidth limitations that are imposed by the switch network.

Figure 1-15 shows an example network that has full bisection bandwidth (all connections that are depicted are 200 Gbps).

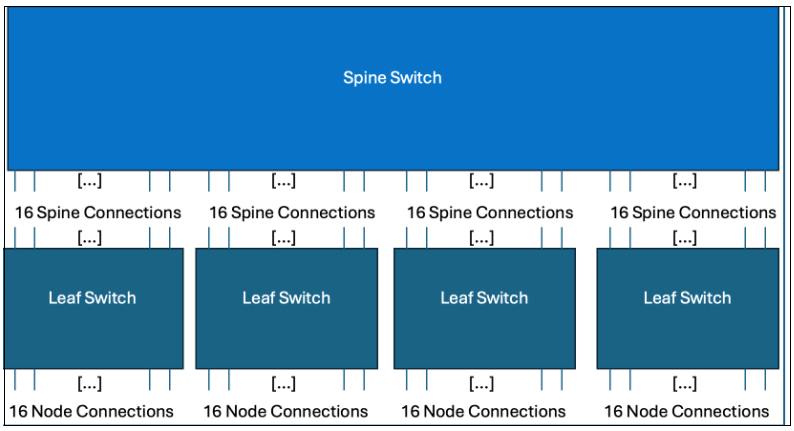


Figure 1-15 Non-blocking/not oversubscribed networking example

In a leaf-spine topology, switch-to-switch connections should provide sufficient bandwidth to support the required endpoint throughput. For NVIDIA DGX SuperPOD deployments, an NVIDIA best practice is per-node I/O exceeding 40 GBps.

An oversubscribed network is one that lacks full bisection bandwidth, meaning that communication between two evenly divided groups of endpoints might encounter bandwidth limitations that are imposed by the switch network. For example, a network with 3:1 oversubscription has three times as many node connections as leaf-to-spine switch connections, as shown in Figure 1-16. In this case, all connections are 200 Gbps.

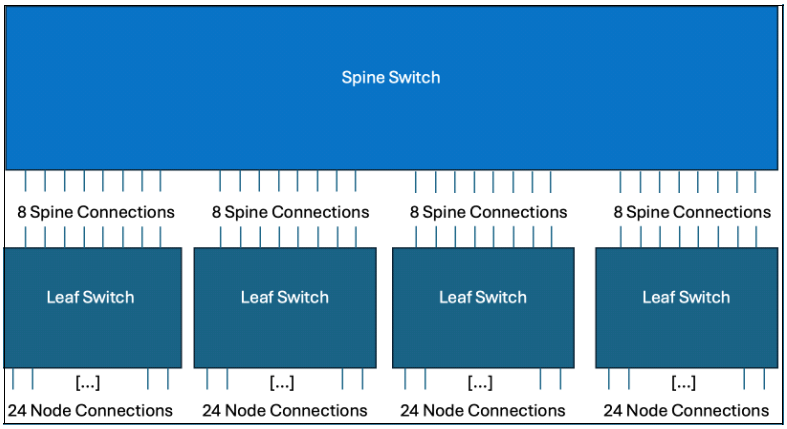


Figure 1-16 3:1 oversubscribed (not fully non-blocking) networking example

The tradeoff between deploying a fully non-blocking network and an oversubscribed network typically comes down to cost. Sometimes, the expense of a fully non-blocking network might not be justified, particularly if the endpoints cannot fully use the available network bandwidth. For example, a configured IBM Storage Scale 6000 can support up to 400 GBps of network bandwidth, but limitations in disk and memory bandwidth restrict the maximum achievable read throughput to about 330 - 340 GBps. Although some capacity greater than 330 - 340 GBps is a best practice for network transfers, the IBM Storage Scale 6000 cannot achieve 400 GBps of network throughput solely through read requests to client nodes.

To help ensure system stability and good performance, IBM Storage Scale requires reliable communication between all nodes in an IBM Storage Scale cluster. When installing a new system, it is a best practice to conduct network verification tests to identify and replace defective components or connections. Faulty adapters, cables, or switch port connections can cause packet loss, negatively affecting performance and system stability. Endpoint and switch configuration might be required to minimize packet loss and optimize throughput and latency.

Also, when nodes in a remote client cluster access another cluster's home file system by using the IBM Storage Scale remote mount feature, the nodes in the client cluster and the home cluster must have network connectivity between each other. When multiple client clusters mount a remote cluster's file system, they do not need connectivity with each other. Instead, each client cluster needs full connectivity only to all nodes in the remote cluster that serves the file system. For more information about on remote mounts, see Mounting a remote GPFS file system.

1.4.6 Interconnect choices for the storage fabric

Note: NVIDIA supports InfiniBand and Ethernet networks, and the best practice is to enable RDMA in both configurations.

InfiniBand

Designed for high-speed, low-latency communications, InfiniBand has long been a great choice for delivering optimal performance for HPC and AI applications. InfiniBand supports data rates from 10 Gbps to over 400 Gbps with low latency, which is critical for workloads such as HPC, AI, and large-scale simulations.

RDMA over InfiniBand

InfiniBand supports native RDMA, and applications seeking optimal throughput and latency typically use RDMA over an InfiniBand network. Although integrating and configuring an InfiniBand network into an existing environment might be more of a challenge than integrating Ethernet, the task of enabling RDMA for optimal throughput and latency is typically less involved than enabling RDMA over Converged Ethernet (RoCE). For this reason, and to achieve optimal performance, it is a best practice that RDMA is enabled for all IBM Storage Scale 6000 and NVIDIA DGX SuperPOD deployments on InfiniBand networks.

IP over InfiniBand

While RDMA delivers the best performance for applications running on an InfiniBand network, TCP/IP support is essential for communication with endpoints outside the InfiniBand network. Also, the TCP protocol support must be configured on all nodes in an IBM Storage Scale cluster regardless of whether RDMA is enabled because it is a requirement for IBM Storage Scale.

Ethernet

Traditionally less expensive and simpler to deploy and manage than InfiniBand, Ethernet solutions are another great choice for IBM Storage Scale 6000 and NVIDIA DGX SuperPOD deployments. Although InfiniBand has been the traditional choice for performance reasons in the past, Ethernet performance is becoming increasingly competitive. To achieve the best performance that IBM Storage Scale 6000 is capable of, enable RoCE.

RDMA over Converged Ethernet

While not as widely adopted as RDMA on InfiniBand networks, RoCE has become increasingly common and simpler to configure in recent years. When properly configured with mechanisms like Priority Flow Control (PFC) and Explicit Congestion Notification (ECN), RoCE can deliver performance close to that of InfiniBand. Although achieving near-lossless connections on Ethernet networks can be more challenging, advancements in technology, such as NVIDIA Spectrum-4 Ethernet switches combined with Bluefield adapters, have improved RoCE's ability to approach InfiniBand-level performance.

As RoCE configuration has become more accessible for customers and offers superior performance compared to TCP/IP, it is a NVIDIA best practice to enable RoCE for Ethernet configuration. It is also a best practice that RoCE is used in IBM Storage Scale 6000 and NVIDIA DGX SuperPOD deployments on Ethernet networks.

IP over Ethernet

Because most customers are familiar with configuring Ethernet networks, deploying TCP/IP over Ethernet is typically straightforward. Although not as performant as RoCE, TCP/IP provides a simpler alternative with fewer dependencies because it does not require PFC support in switches. Although RDMA offers the best performance for applications running on Ethernet networks, TCP/IP remains essential for communication with endpoints outside the Ethernet network where RoCE is enabled. Also, TCP protocol support must be configured on all nodes in an IBM Storage Scale cluster regardless of whether RDMA is enabled because this support is a requirement for IBM Storage Scale. Although configuring TCP/IP is required, a best practice is to enable RoCE for optimal usage of CPU and memory bandwidth on both the IBM Storage Scale 6000 server nodes and the compute cluster.

2

Architecture

This chapter provides information that is related to the overall system architecture of the NVIDIA DGX SuperPOD deployment that uses the IBM Storage Scale System. This information includes all hardware components, racking and networking considerations, and the usage of Active File Management (AFM) to seamlessly migrate into the newly configured environment.

This chapter describes the following topics:

2.1 Environment and rack layout

One NVIDIA DGX SuperPOD Scalable Unit (SU) is composed of multiple DGX systems and networking components. IBM Storage Scale System 6000 provides the capacity layer for the solution.

Data center designers should be mindful of the requirements and upper limits for physical deployment in the environment. Physical limitations for the following items should always be considered when planning for each SU:

- Electrical power per rack
- Cooling type and specifications
- Static weight and point load limit
- Contiguous rack space
- Seismic requirements
- Rack orientation
- Cable routing

Careful data center planning is required to stay within the deployment goals that are provided by the target data center facility. Concurrently, design specifications and the cable length radius of the SU components must be maintained.

This chapter provides specifications for NVIDIA DGX SuperPOD and IBM Storage Scale System 6000 integration.

2.1.1 NVIDIA DGX SuperPOD inventory (1 SU)

The NVIDIA DGX SuperPOD and IBM Storage Scale 6000 components must be mounted in standard EIA-310-D compliant racks. One NVIDIA DGX SuperPOD SU requires a minimum of 11 full-height racks, as shown in Table 2-1.

Table 2-1 NVIDIA DGX SuperPOD 1 SU rack space requirements

Type of rack	No. of racks	No. of units	Component
Compute	8	31	NVIDIA DGX systems
		24	Raritan power distribution units (PDUs)
Network	2	8 compute leaf switches	NVIDIA Quantum QM9700
		4 compute spine switches	NVIDIA Quantum QM9700
		6 storage switches	NVIDIA Quantum QM9700
		4 in-band management switches	NVIDIA SN4600c
		2 out-of-band management switches	NVIDIA SN2201 (48x 1 Gbps ports)
		2 appliances	NVIDIA Unified Fabric Manager Appliance 3.1
		4	Raritan PDUs
Storage	1	1 IBM Storage Scale 6000 Building Block	IBM Storage Scale 6000

Note: An IBM Storage Scale System 6000 building block can be placed in a network rack if disk enclosures are not installed. Capacity expansion requires a new rack, and the physical connectivity must be changed. A separate rack should be installed to avoid this situation.

In one SU block, one DGX node is removed to accommodate NVIDIA Unified Fabric Manager (UFM) appliance connectivity. Therefore, the cluster has 31 DGX nodes in total.

2.1.2 NVIDIA DGX SuperPOD inventory (4 SU)

The NVIDIA DGX SuperPOD and IBM Storage Scale 6000 components must be mounted in standard EIA-310-D compliant racks. One NVIDIA DGX SuperPOD SU requires a minimum of
39 full-height racks, as shown in Table 2-2.

Table 2-2 NVIDIA DGX SuperPOD 4 SU rack space requirements

Type of rack	No. of Racks	No. of Units	Component
Compute	32	127	NVIDIA DGX systems
		96	Raritan PDUs
Network	6	32 compute leaf switches	NVIDIA Quantum QM9700
		16 compute spine switches	NVIDIA Quantum QM9700
		16 storage switches	NVIDIA Quantum QM9700
		8 in-band management switches	NVIDIA SN4600c
		8 out-of-band management switches	NVIDIA SN2201 (48x1Gbps ports)
		4 appliances	NVIDIA Unified Fabric Manager Appliance 3.1
		4 BCM management servers	NVIDIA Base Command Manager
		12	Raritan PDUs
Storage	1	1 IBM Storage Scale 6000 building block with 9 disk enclosures	IBM Storage Scale 6000

Note: When more than one SU is put together, all but the last SU is populated with 32 DGX nodes. The last SU is populated with 31 DGX nodes to accommodate UFM appliance connectivity. Therefore, the cluster has 127 DGX nodes in total.

2.1.3 DGX compute nodes

NVIDIA DGX systems (Figure 2-1) are the compute element of the NVIDIA DGX SuperPOD SU. This node is a fourth-generation architecture node with an eight-GPU configuration. The GPUs are built on the NVIDIA Hopper or Blackwell architecture.

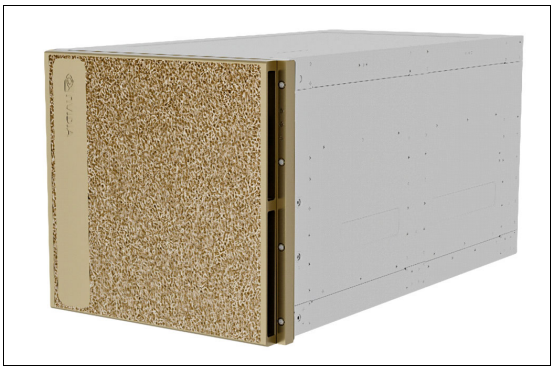


Figure 2-1 NVIDIA DGX B200

The DGX B200 provides rich compute capacity to deliver substantial performance enhancements to large artificial intelligence (AI) and machine learning (ML) workloads. It has the following features:

- Up to 144 petaflops for AI inference
- Dual 56-core 4 Gen Intel Xeon capable processors with PCIe 5.0 support and DDR5 memory
- Networking and storage connectivity @ 400 Gbps InfiniBand or Ethernet with NVIDIA ConnectX-7 smart network interface cards (SmartNICs)
- 1.5X higher bandwidth per GPU @ 900 GBps with fourth-generation NVIDIA NVLinks
- 1440 GB of aggregated HBM3 memory with 24 TBps of aggregate memory bandwidth

2.1.4 IBM Storage Scale System 6000 storage

The IBM Storage Scale System 6000 (Figure 2-2) can deliver up to 310 GBps throughput, up to 13 M IOPS, and up to 3.4 PBe (effective capacity) in a 4U rack.

Figure 2-2 The IBM Storage Scale System 6000

The IBM Storage Scale System 6000 uses a simple building-block approach with performance that scales linearly. A cluster of 10 IBM Storage Scale System 6000 system is capable of more than a 3 TBps throughput. It supports up to nine SAS hard disk drive (HDD) expansion enclosures.

Here are the key features of the IBM Storage Scale System 6000:

- A single 4U node with active-active controllers and redundant hardware to maximize uptime.
- Up to 310 GBps throughput with low latency.
- Up to 13 millions IOPS by using NVMeoF.
- Up to 3.4 PBe (effective capacity) in a standard 4U rack space.
- Supports up to forty-eight 3.84 TB, 7.68 TB, 15.36 TB, or 30 TB 2.5” Non-Volatile Memory Express (NVMe) flash drives.
- Supports 19.2 TB and 38.4 TB FlashCore Module 4 NVMe drives.

2.1.5 Compute rack layout

Each computer rack typically holds up to four DGX B200 nodes and their associated PDUs.

Figure 2-3 shows an example of a row of compute racks.

Figure 2-3 Compute rack that is configured with 4x DGX B200 nodes

Compute racks can also be configured to hold one or two DGX B200 nodes if the available power or space per rack is not sufficient. Although this configuration is supported, the total solution must be evaluated to maintain a cable length radius of 50 meters for the InfiniBand fabric.

All components of the NVIDIA DGX SuperPOD SU and IBM Storage Scale 6000 are non-disruptively serviceable for most physical changes. It is a best practice that empty space within the racks should not be used to host any other equipment because it can interfere with the serviceability and maintenance of the NVIDIA DGX SuperPOD.

2.1.6 Network rack layout

A network rack holds all the networking switches for compute and management. It also holds the UFM appliances for system management. The connectivity is defined in terms of topology and port mapping to all components within the SU. The uplink is also provided here to connect the SU to the customer infrastructure and other SUs.

The network cables for NVIDIA DGX SuperPOD SU connectivity fan out from this rack to all compute and storage racks. This rack provides uplink connectivity to the customer network infrastructure.

Figure 2-4 shows the rack layout of the NVIDIA DGX SuperPOD networking components.

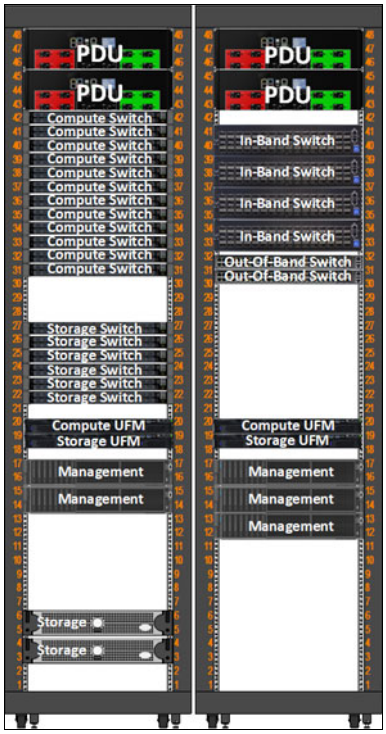


Figure 2-4 Sample rack SU configuration

2.1.7 Environmental requirements: Cooling

Table 2-3 shows the environmental requirements for cooling for the architecture.

Table 2-3 Environmental requirements for cooling

Feature	Specification
Operating temperature	5* C to 30* C (41* F to 86* F)
Relative humidity	20% - 80% non-condensing

Feature	Specification
Airflow	1105 CFM front to back @ 80% fan PMW
Heat output	33557 BTU/hr

2.1.8 Power consumption

Table 2-4 shows the power consumption for the various models.

Table 2-4 Power consumption

Product	kVA	Amps	Power supply outlets	Inlet	Watts	BTU/hr	Weight (lbs)
5141-FN2	1.550	7.75	2	C14	1500	5118	59
5147-102	1.34	6.70	2	C14	1300	4436	278
5149-091	3.00	13.00	2	C20	2400	7077	260
5149-23E	0.618	3.06	2	C14	600	2046	35.3
5149-F48	4.50	22.50	4	C20	4400	15013	153

2.1.9 Environmental requirements: Power

Both the NVIDIA DGX SuperPOD SU and IBM Storage Scale 6000 are based on the building block concept. Multiple units or blocks can be combined to scale up the deployment per the customer requirements. Power, cooling, and space requirements scale up linearly as more units or blocks are added.

Table 2-5 lists the power specifications of the NVIDIA DGX SuperPOD components.

Table 2-5 Environmental requirements for power

		Servers				Switches			
		Compute	Storage	Mgmt	Fabric	Compute	Storage	In-band mgmt	Out-of-band mgmt
	Model	DGX B200	IBM Storage Scale 6000	X86	NVIDIA UFM 3.1	QM9700	QM9700	SN4600c	SN2201
	Qty	127	1	5	4	48	16	8	8
EAP	Each	10.2 kW	3.2 kW	704	600	1.4 kW	1.4 kW	0.5 kW	98 W
	Subtotal	12,95,400	3200	3,520	2,400	66,048	22,016	3,728	784
EPP (watts)	Each	10,200	4800	880	750	1,720	1,720	820	135
	Subtotal	12,95,400	4800	4,400	3,000	82,560	27,520	6,560	1,080
Peak Heat Load (BTU/h)	Each	34,804		3,003	2,559	5,869	5,869	2,798	461
	Subtotal	44,20,108		15,015	10,236	2,81,712	93,904	22,384	3,688

2.1.10 Environmental requirements: Thermal

The NVIDIA DGX SuperPOD nodes and IBM Storage Scale 6000 use airflow cooling for standard operation. Sufficient airflow and temperature should be maintained in the environment for optimal function.

Table 2-6 provides the thermal operating limits for DGX nodes and IBM Storage Scale 6000 storage.

Table 2-6 Component operating temperature range

Component	Operating temperature range	Humidity range
DGX H100 systems	15* to 32* C (59* to 89.6* F)	20 - 80% RH
IBM Storage Scale 6000	5* to 35* C (41* to 95* F)	20 - 80% RH
Common range for all	15* to 32* C (59* to 89.6* F)	20 - 80% RH

Considerations for air flow direction

The airflow for the NVIDIA DGX SuperPOD components determines the rack-mounting orientation. It also influences the cable lengths and the cable routing solution to use for the data center.

Using aisle containment or aisle separation is a best practice to keep hot air from mixing with cold air, which improves cooling efficiency. The air flow direction of all components must be matched to support the environment.

DGX nodes have a front-to-rear airflow, that is, cold air enters from front of the chassis and hot air is expelled from rear. This air flow cannot be changed, so DGX nodes should be mounted to face the cold air intake in the rack.

Network switches and appliances in the NVIDIA DGX SuperPOD support front-to-back and back-to-front airflow. This configuration is selected during bill of materials. preparation and cannot be changed in the field.

An incorrectly ordered device can be mounted in a reversed direction, but it might not support the power or network cabling routing. Therefore, it is important to review data center provisions in advance.

2.1.11 Electrical specifications

The NVIDIA DGX SuperPOD and the IBM Storage Scale 6000 support international standards for voltage and power provisioning. A DGX H100 power supply system uses components that are certified for 200 - 240 VAC deployed worldwide. Single-phase and three-phase power supplies can be used by the PDUs. The PDUs divide the three-phase input power to obtain three single-phase circuits for operation.

Figure 2-5 on page 37 and Figure 2-6 on page 37 provide common supply and distribution voltages and currents that can support the defined SU deployment patterns.

Phase	Distribution Voltage	Line Voltage	Amps	Breaker Derating	Rack Power Capacity kW* (Two Circuits Provisioned)	Maximum Supported DGX B200 Systems per Rack**	Peak Server Demand per Circuit kW**
3Φ	380	219	32	80.0%	32	2	28.6
3Φ	400	230	32	80.0%	34	2	28.6
3Φ	415	240	32	80.0%	35	2	28.6
3Φ	415	240	60	80.0%	66	4	57.2
3Φ	415	240	63	80.0%	69	4	57.2
3Φ	380	219	32	100.0%	40	2	28.6
3Φ	400	230	32	100.0%	42	2	28.6
3Φ	415	240	32	100.0%	44	2	28.6
3Φ	415	240	60	100.0%	82	4	57.2
3Φ	415	240	63	100.0%	86	4	57.2

0.95 power factor.

* Based on a two circuit N power provisioning scheme.

Figure 2-5 Common power distribution schemes that are compatible with DGX B200 racks

Table 5. Power rating specifications per power module

Model and type	PSU	Input power requirements	Maximum input current	Maximum power output
4U IBM Storage Scale System 6000 (MTM 5149-F48), 48-drives configuration	2400 W (4)	200 V to 240 V single phase AC At a frequency of 50 Hz or 60 Hz IEC C20 standardized	13A (x4)	2400 W

Figure 2-6 IBM Storage Scale 6000 Power module specifications

2.2 Network design

The NVIDIA DGX SuperPOD and IBM Storage Scale 6000 both use a modular networking approach, and both fit nicely together. This design makes it simple to scale performance nearly linearly with large SU deployments.

The NVIDIA DGX SuperPOD design separates network traffic into physically independent networks. Each network carries specific data traffic without affecting each other. Similarly, the IBM Storage Scale 6000 mandates isolated networks for traffic segmentation. This section describes how to connect the NVIDIA DGX SuperPOD network fabrics to the IBM Storage Scale 6000.

2.2.1 Fabrics that are used by NVIDIA DGX SuperPOD

NVIDIA DGX SuperPOD has four network fabrics, each serving a specific role. This section describes the connectivity that is required by the IBM Storage Scale System 6000 and IBM Storage Scale cluster to integrate with the NVIDIA DGX SuperPOD.

Figure 2-7 shows an example of the connectivity that is required by two IBM Storage Scale System 6000 systems along with the ESS Management Server (EMS), and the protocol nodes that are required for NFS user home shares. Also shown is an optional AFM or additional protocol node that connects to an external network for extending the storage platform to external storage or users.

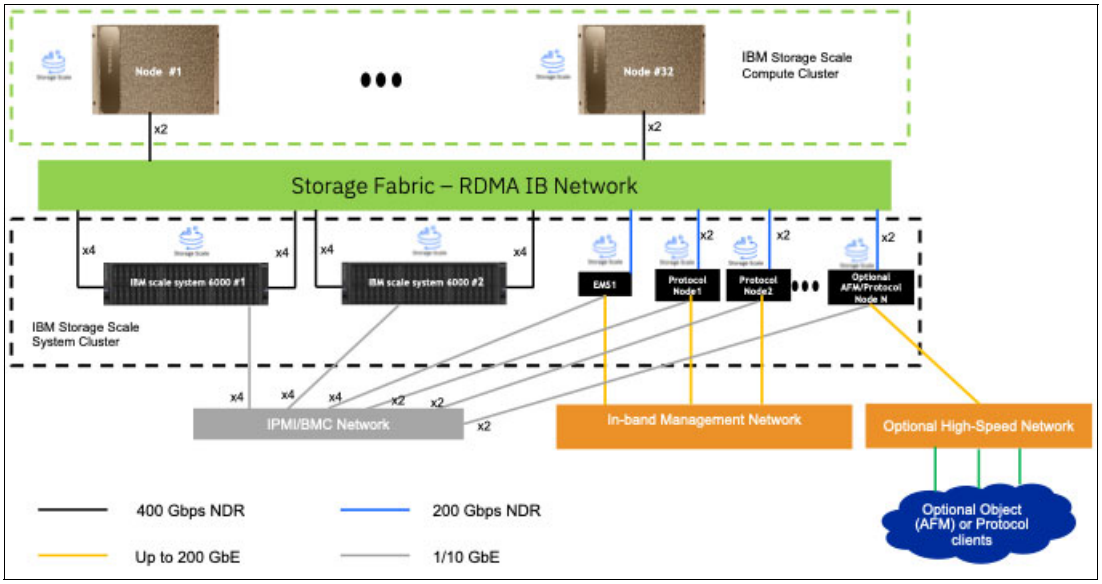


Figure 2-7 Sample network diagram with optional AFM connectivity

Compute fabric

The NVIDIA DGX SuperPOD compute fabric is used by DGX nodes for inter-node communication for exchanging compute workload. Mellanox InfiniBand switches are connected in a full-fat tree topology to form the network. The network is a rail-managed fabric to enable optimum bandwidth to different traffic flows between multiple DGX nodes.

Storage fabric

The NVIDIA DGX SuperPOD storage fabric is an independent fabric that connects the DGX compute nodes and the IBM Storage Scale System 6000 storage. The network uses a spine-and-leaf design to enable expandability and adjustments based on system requirements. Ports that are used by this network are not shared by any other fabric.

In Figure 2-8 on page 39, you see the logical storage fabric design for a NVIDIA DGX SuperPOD deployment, which includes the IBM Storage Scale System management node and an optional AFM or protocol node. Additional AFM or protocol nodes may be added as needed.

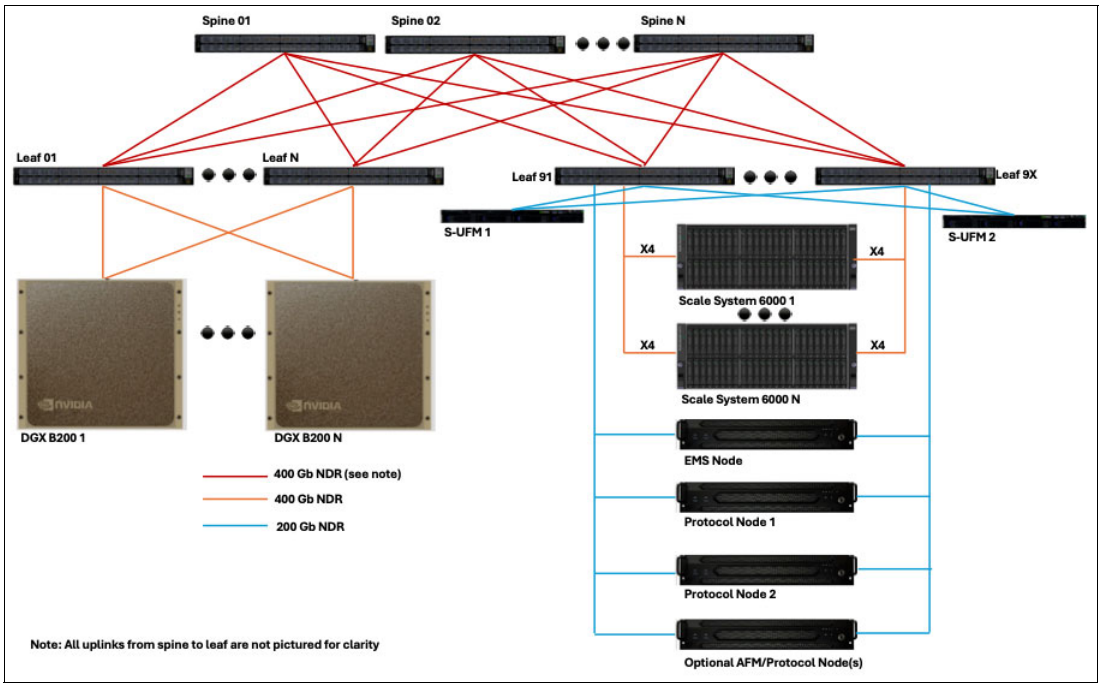


Figure 2-8 Logical storage network fabric design

It is a best practice to have a 1:1 ratio of uplinks from the leaf switches that are attached to the IBM Storage Scale 6000 storage (Leaf 91 - Leaf 9X in Figure 2-8) and ports that are connected to the IBM Storage Scale System 6000. So, for example, if two IBM Storage Scale System 6000 building blocks are connected, for a total of eight ports to each leaf switch, there should be a minimum of eight ports from each leaf switch that are connected to the spine switches.

In-band management network

The IBM Storage Scale cluster uses the in-band management network for two purposes:

- To connect to the EMS node to manage the cluster. The EMS node provides GUI and CLI access to the IBM Storage Scale System 6000, and it is used to configure, upgrade, manage, and monitor the system. This access can be routed to a customer network if external access is required. Each IBM Storage Scale System 6000 also uses a low-speed connection to this network for provisioning and management.
- To provide NFS access to the IBM Storage Scale file system. NFS access is not used for computing jobs, but is used by the NVIDIA DGX SuperPOD for home directories and management tasks.

Figure 2-9 shows the logical network design in which the protocol nodes are attached to NVIDIA SN5600 switches for NFS access. The EMS node is attached to SN2201 switches for access to the GUI and management of the system. The customer network typically attaches to the in-band management network to allow remote management of the system.

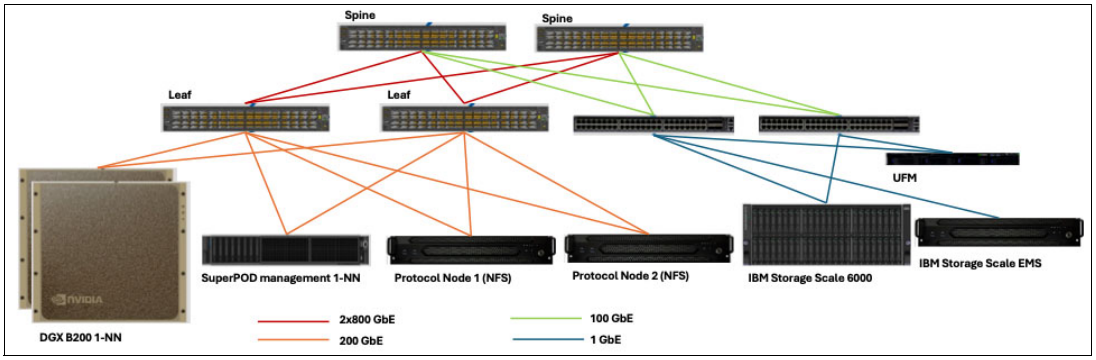


Figure 2-9 Logical in-band management network design

Out-of-band management network

The final network that is used in the NVIDIA DGX SuperPOD is the out-of-band management network. This low-speed network is used for Baseboard Management Controller (BMC) access. This network can be configured on the out-of-band management switches on the NVIDIA DGX SuperPOD. A separate VLAN should be created to avoid any conflicts and logically isolate this network from the BMC network on the DGX or any other servers.

Each EMS node and each IBM Storage Scale System 6000 building block requires two 1 GbE connections to this network. Any IBM Storage Scale Utility Nodes that are used as either a protocol or AFM node also require one 1 GbE connection to this network for provisioning.

Figure 2-10 shows an out-of-band network example.

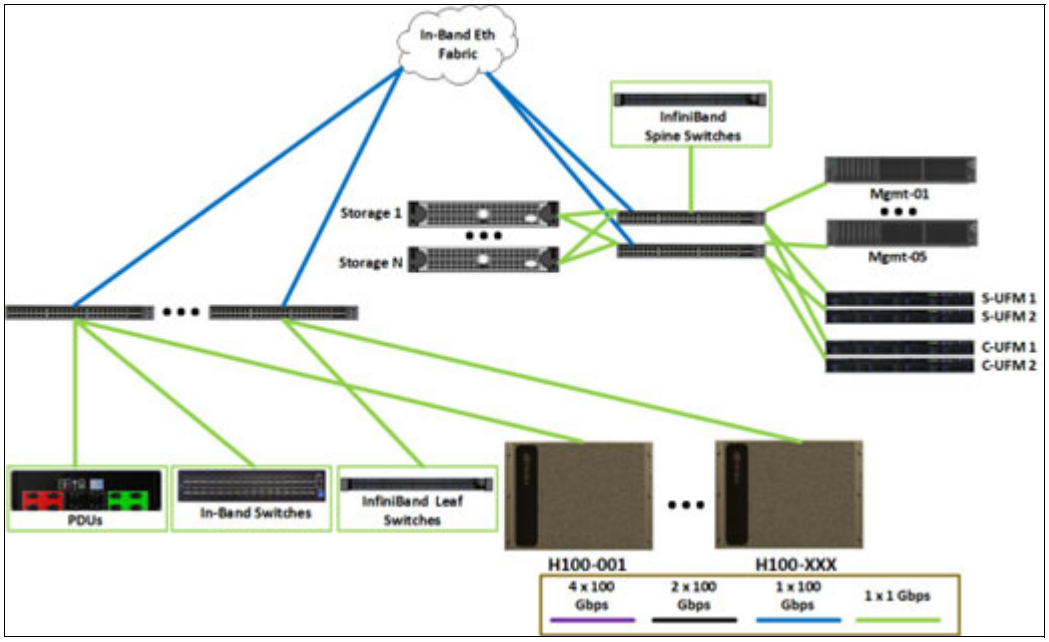


Figure 2-10 Out-of-band network example

2.2.2 Fabrics that are used by NVIDIA DGX SuperPOD SU

The following section provides details for each type of fabric that is required to build a NVIDIA DGX SuperPOD SU. We supply examples for each of the management and data networks for the NVIDIA DGX SuperPOD.

2.3 Fabric connectivity to the IBM Storage Scale 6000

This section describes the types of connectivity from the NVIDIA DGX SuperPOD SU fabric to the IBM Storage Scale 6000.

Figure 2-11 shows these fabrics at a glance.

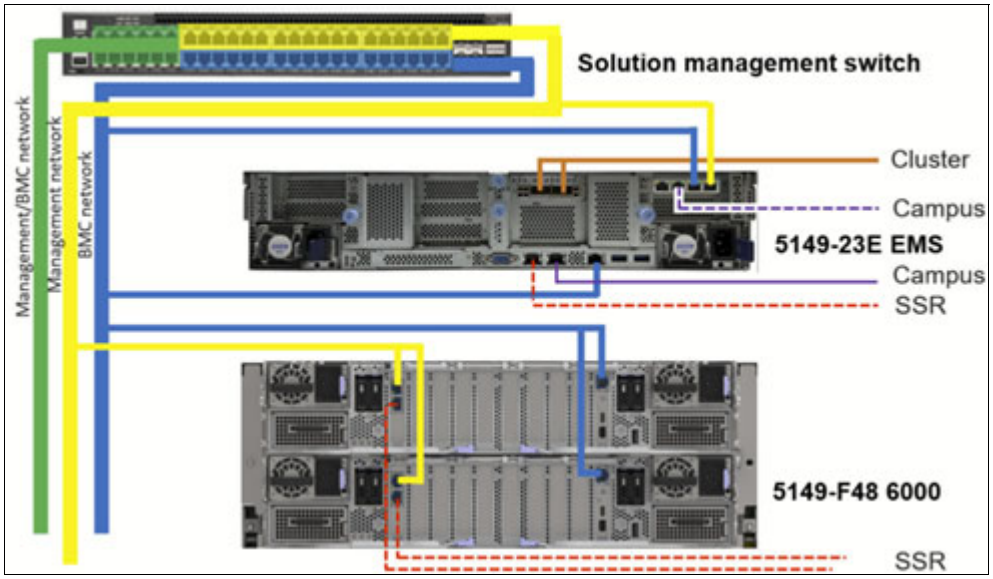


Figure 2-11 NIVIDIA SuperPOD SU fabrics

File Service Protocol and BMC network

This network is connected to the out-of-band management network on the NVIDIA DGX SuperPOD. The I/O nodes of IBM Storage Scale 6000 storage use BMC ports to enable remote access during initial provisioning and system management. These ports are 1 GbE onboard ports on the I/O nodes.

The I/O nodes should be logically isolated from the remainder of the out-of-band management network by using VLANs on the out-of-band switches.

Management network

The IBM Storage Scale 6000 system is provisioned by using 10 GbE ports on the I/O nodes. These ports connect directly to the in-band management network. Typically, this network uplinks to the campus or external network to provide external management access to the cluster.

Campus or external network

The campus network is an optional 10 GbE network connection for each I/O node of the IBM Storage Scale 6000. When this network is used, it provides external management access to the cluster and can isolate the management network. The management network is then used only for provisioning and upgrades.

High-speed network

The IBM Storage Scale 6000 provides dedicated slots to install high-speed storage access to multiple building blocks and external POSIX and CES clients. This network is connected directly to the storage fabric on the NVIDIA DGX SuperPOD. Outside of NVIDIA DGX SuperPOD deployments, IBM Storage Scale 6000 supports 100/200/400 Gb Ethernet and InfiniBand 100/200/400 Gbps connectivity.

InfiniBand or Virtual Protocol Interconnect (VPI) cards (in InfiniBand mode) must be used for NVIDIA DGX SuperPOD storage fabric connectivity.

IBM Storage Scale system integration

Figure 2-12 shows a combined network topology when NVIDIA DGX SuperPOD and IBM Storage Scale 6000 are used together.

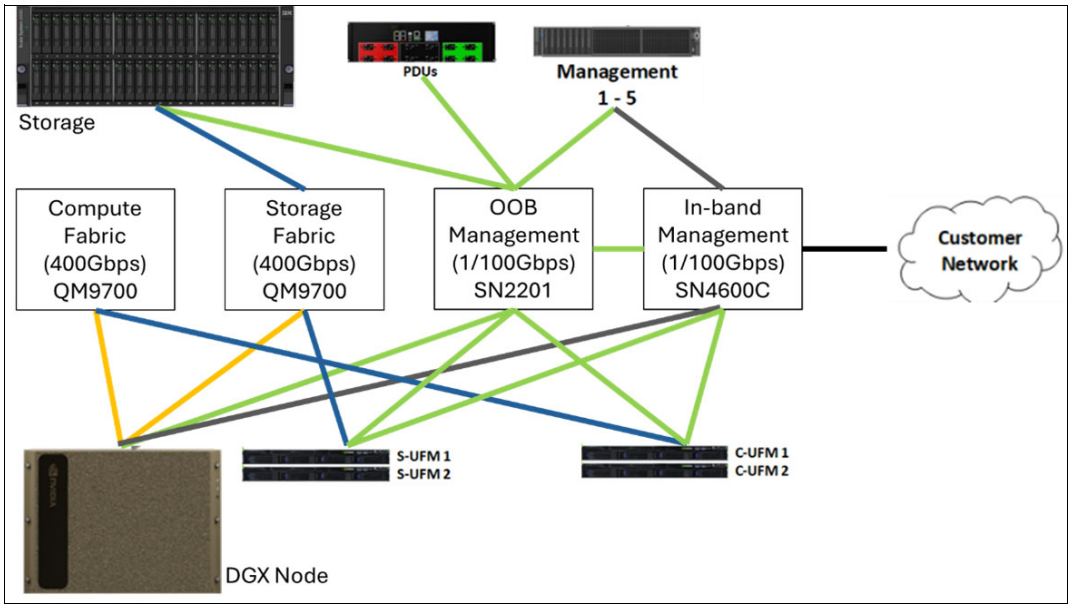


Figure 2-12 SU networking fabric example

Port reservations on DGX nodes

Each DGX node uses dedicated ports for connecting compute, storage, and management fabrics.

Figure 2-13 on page 43 shows the rear panel of the DGX H100 and outlines the ports for each type of network reservation.

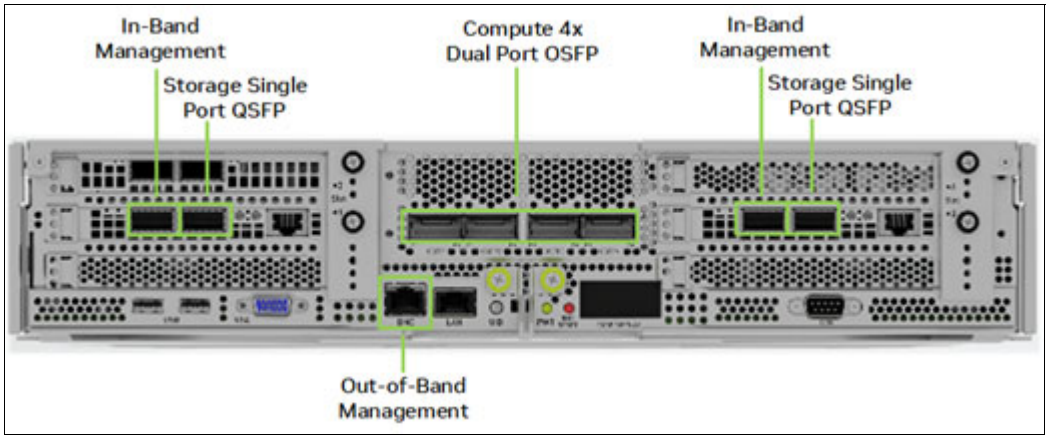


Figure 2-13 DGX H100 rear panel diagram

Table 2-7 shows the speed of each port and its target switch. Each node connects to all eight leaf switches in their NVIDIA DGX SuperPOD SUs. This configuration enables direct connectivity to all nodes within the SU.

Table 2-7 DGX H100 port information table

Fabric	Number of ports	Speed	Interface	Switch
Compute	4	2x 400 Gbps each	InfiniBand octal small form factor pluggable (OSFP)	MQM9700-NS2F (Leaf)
Storage	2	400 Gbps each	InfiniBand QSFP	MQM9700-NS2F (Leaf)
In-band management	2	200 Gbps each	InfiniBand QSFP	SN4600C (Leaf)
Out-of-band management	1	1 Gbps	Copper GbE	SN2201 (Leaf)

2.3.1 Control and communication: Bastion host

One desktop host is required to access management tools for NVIDIA DGX SuperPOD devices. A dedicated host can be provided to simplify firewall access between the Bastion host and the NVIDIA DGX SuperPOD. The Bastion host can be a stand-alone server or a virtual machine with the following minimum configuration:

- Four CPUs or vCPUs
- 16 GB of memory
- 100 GB of available disk space
- Windows or Linux OS
- SSH client
- SCP client
- Web browser

2.4 Active File Management

IBM Storage Scale AFM is an optional capability that may be deployed as part of a NVIDIA DGX SuperPOD solution. AFM enables data sharing across multiple types of storage, even if the network is unreliable or has high latency. Using AFM, it is possible to create associations to seamlessly move and share data between different IBM Storage Scale clusters or between an IBM Storage Scale cluster and a Network File System (NFS) data source or Simple Storage Service (S3) buckets. AFM provides users and administrators with several unique capabilities, some of which are highlighted in this section

Associating a file system with external S3, NFS, or other IBM Storage Scale clusters enables the following functions:

- Parallel file transfers move unused datasets quickly to lower cost storage, freeing space on the IBM Storage Scale System 6000 for additional data.
- Existing applications can read/write new data to common S3 or NFS endpoints. This data can quickly and automatically be transferred to the IBM Storage Scale 6000 for deep learning or AI models to access.
- Results from models or other jobs can quickly and automatically be transferred to external systems, enabling applications and users to access them without requiring direct access to the NVIDIA DGX SuperPOD cluster itself.

AFM also enables IBM Storage Scale clients to quickly stand up a cluster in a cloud environment. (This task may be done by using CloudKit, which is a CLI that provides the automated installation, deployment, and configuration of IBM Storage Scale on public clouds.) A cloud deployment that uses AFM can connect to on-premises storage or existing object storage to quickly provide access to existing data.

AFM, acting as an abstraction layer, enables users to extend the file namespace into the cloud, and see the same data from on-premises and cloud environments. The client moves only the data that is needed to the cloud, so not all the data would have to be moved.

Another common use case for AFM is disaster recovery (DR). AFM-based asynchronous disaster recovery (AFM-DR) provides IBM Storage Scale users with a file set-level replication DR solution that has the following capabilities:

- Providing business stability during a disaster
- Restoring business stability after the disaster is over
- Enduring multiple disasters

2.4.1 AFM concepts

To use the capabilities of AFM, a solution requires two or more AFM gateway nodes. These nodes are responsible for managing AFM relationships, communicating with the remote storage system, and copying data and metadata between the storage system that is attached to the NVIDIA DGX SuperPOD and the remote storage system. Two nodes are required at a minimum for redundancy if there are hardware failures.

AFM relationships are defined on file sets on the IBM Storage Scale file system. To a user, a file set appears as a directory, but it has special administrative properties that in some ways behave like a unique file system. Among those properties is the ability to set user and data quotas, perform scans, and define AFM relationships. An administrator can create thousands of file sets on a file system.

In AFM, each file set may be a unique data set, project, or user. These relationships can each point to different data sources, for example, a different NFS mount point, a unique object storage bucket, or a unique remote IBM Storage Scale cluster. When configuring AFM, consider how the data is organized, where it is, and what datasets it belongs to in order to create a directory and file set hierarchy that best serves the needs of the users.

Every AFM relationship has a home site, where the data is, and a cache site, which may temporarily cache the data to access it. Often, the storage that is attached to the NVIDIA DGX SuperPOD cluster is considered the cache, where it is stored while being used for training or analytics jobs. Once the data is not required, it can be evicted from the cache so that other jobs can use the storage space.

For other AFM use cases, such as temporarily moving a compute workload to a cloud provider, the relationship may be reversed, with the cloud cluster as the cache while the data is on storage that is attached to the NVIDIA DGX SuperPOD.

2.4.2 AFM sizing considerations

When sizing an AFM deployment, the two primary factors are the gateway nodes and the network connectivity. Usually, each AFM gateway node has three network connections:

- Two connections to the storage fabric
- One connection to the external network that communicates with the external storage source
- A BMC and management connection to the out-of-band management network for deployment and management

For deployment and support, using IBM Storage Scale Utility nodes as AFM gateway nodes is a best practice. However, AFM can run on any x86-based or IBM Power server running a version of Linux that is supported by IBM Storage Scale. Each node should have a minimum of 128 GB of memory, and each gateway node can support up to 20 file sets. When sizing the number of nodes, additional factors such as the speed of the external storage source, the amount of data to cache, and the change rate of the data should be considered. Using NVIDIA ConnectX adapters to connect to the storage fabric is a best practice, and additional ports should be configured as needed to communicate with the external storage source. For more information about sizing see “Planning for AFM” in *IBM Storage Scale 5.2.1 Concepts, Planning, and Installation Guide*, SC28-3906-01 and IBM Storage Scale Frequently Asked Questions and Answers.

2.4.3 AFM Gateway node connectivity

This section describes the connectivity for an AFM Gateway node with an IBM Storage Scale Utility node. Similar guidelines can be used if you use other hardware. For more information about slot and adapter placement, see the hardware manufacturer documentation.

The IBM Storage Scale Utility node supports up to 1 - 3 NVIDIA ConnectX-6 or ConnectX-7 dual-port VPI adapters, providing Ethernet or InfiniBand connectivity. Two of these ports (labeled 2 and 4 in Figure 2-14) are typically used for connectivity to the NVIDIA DGX SuperPOD storage fabric, providing high-speed access to the IBM Storage Scale 6000 storage. The remaining adapter (labeled 5) can optionally be populated with a 4-port 10 GbE adapter or an additional ConnectX-6 or ConnectX-7 adapter. These remaining high-speed ports can be used for connectivity to an external network, which provides access to the external storage system.

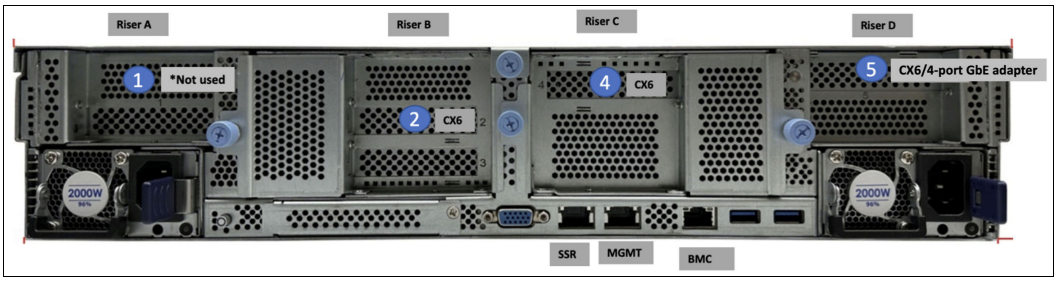


Figure 2-14 IBM Storage Scale Utility Node rear view

The MGMT and BMC connections in Figure 2-14 are used for management access and provisioning, and they are attached to the out-of-band management network.

3

Deployment

This chapter provides a step-by-step overview for the deployment of an NVIDIA DGX SuperPOD Scalable Unit (SU) by using the IBM Storage Scale 6000 in a NVIDIA DGX SuperPOD cluster. The chapter covers details that are related to configuring each component in the SU and the IBM Storage Scale storage cluster, and configuring the Cluster Export Services (CES), the Container Interface Driver, and Active File Management (AFM).

This chapter includes details based on an H100 deployment and the steps to configure the hardware and networking components. These steps also apply to the latest B200-based deployment.

This chapter describes the following topics:

3.1 Deploying an IBM Storage Scale System 6000 for use with a NVIDIA DGX SuperPOD

A successful implementation of the IBM Storage Scale System 6000 solution for the NVIDIA DGX SuperPOD starts with a detailed solution for the high-speed IBM Storage Scale InfiniBand network. This solution considers the network between the IBM Storage Scale client cluster on the DGX nodes, which you use to remotely mount the IBM Storage Scale file systems from the IBM Storage Scale 6000 storage cluster. Also, for the Base Command Manager (BCM), you can configure the Ethernet network for CES-NFS exported IBM Storage Scale file systems for BCM command and control. You may also configured the AFM gateway network to connect to external data sources. If you are deploying Kubernetes (k8S) (for example, for run:ai), you must include the IBM Storage Scale GUI network for the Representational State Transfer (REST) API in the network for integrated solution planning.

The implementation starts with completing the IBM Storage Scale Systems network installation worksheet, which includes the details that are required by the IBM Service Support Representative (IBM SSR) team to complete the initial configuration (code20) and the other networks (high-speed network, CES-NFS, and AFM) that are required for completing the logical deployments.

An example of the installation worksheet for a solution with two IBM Storage Scale 6000 building blocks and one NVIDIA DGX SuperPOD Scalable Unit (SU) DGX solution (32 DGX servers) is included in Appendix B, “IBM Storage Scale System NVIDIA DGX SuperPOD Solution Network Installation Worksheet” on page 119.

Figure 3-1 shows the flowchart of the deployment steps of the IBM Storage Scale 6000 solution for the NVIDIA DGX SuperPOD.

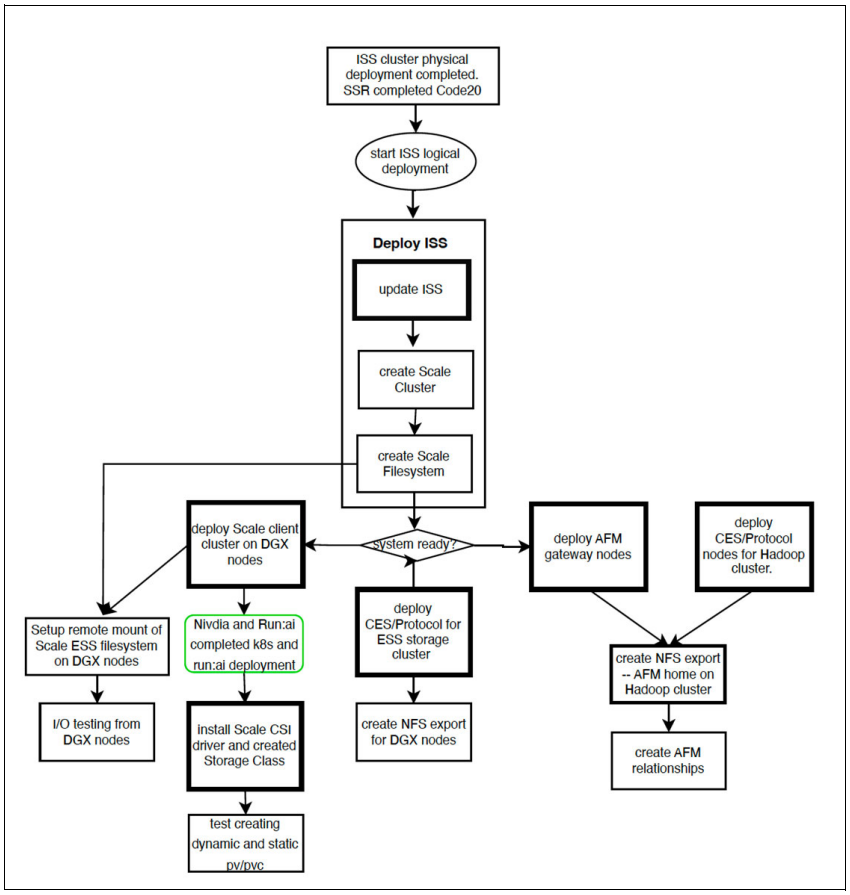


Figure 3-1 Overall workflow for the logical deployment of the IBM Storage Scale Systems solution for the NVIDIA DGX SuperPOD

Figure 3-2 shows the flowchart of the steps in Figure 3-1 on page 49.

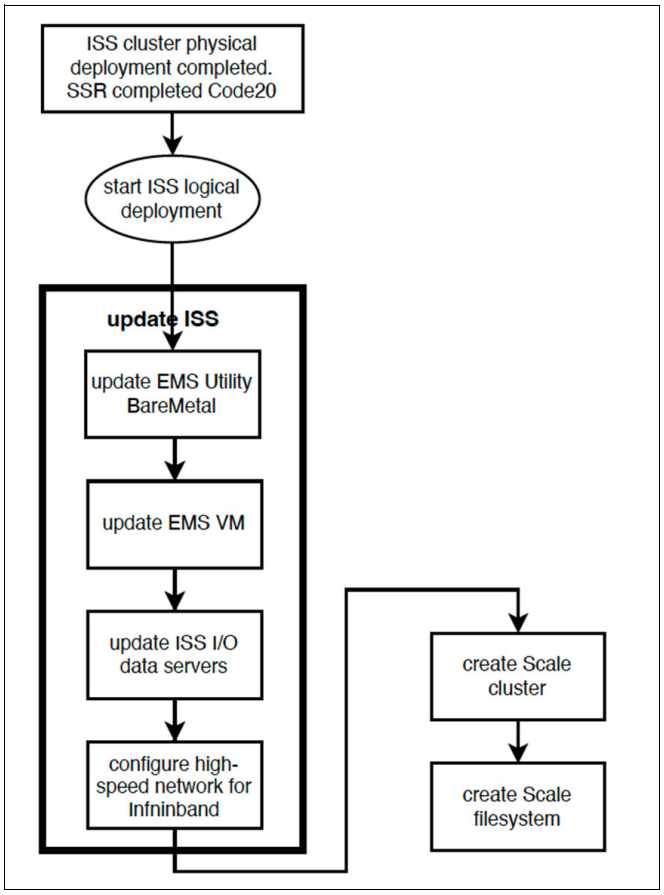


Figure 3-2 IBM Storage Scale 6000 logical deployment workflow

3.2 IBM Storage Scale System: The essrun command

Each IBM Storage Scale System with one or more building blocks requires an ESS Management Server (EMS). The EMS type that supports the IBM Storage Scale System 6000 is the IBM Storage Scale System Utility Node. The EMS node itself is a virtual machine running inside the IBM Storage Scale System Utility Node. The EMS node serves as a third IBM Storage Scale quorum node for configurations with one building block.

The EMS node in an IBM Storage Scale System 6000 cluster provides system management functions. It runs a container with Ansible playbooks that can provide orchestration of complex tasks, such as cluster configuration, file system creation, code updates, and network bonds creation.

The `essrun` command, which is available inside the container, runs the available tasks. The `essrun` utility handles almost the entire deployment process, and only minimal initial user input is required during deployment.

However, you must complete the /etc/hosts file by entering the information for the nodes that will be part of the cluster. These nodes will be managed by Ansible through the essrun command. This file must contain the low-speed (management) and high-speed (cluster) IP addresses, fully qualified domain names (FQDNs), and short names. The high-speed names for each node must contain a suffix to the corresponding low-speed names (For example, essio1-hs (high-speed name) to essio1 (low-speed name), as shown in Example 3-1).

Example 3-1 /etc/hosts file example

```
127.0.0.1 localhost localhost.localdomain.local localhost4 localhost4.localdomain4
## Management IP addresses 192.168.45.0/24
192.168.45.20 ems1.localdomain.local ems1
192.168.45.21 essio1.localdomain.local essio1
192.168.45.22 essio2.localdomain.local essio2
## High-speed IP addresses 10.0.11.0/24
10.0.11.1 ems1-hs.localdomain.local ems1-hs
10.0.11.2 essio1-hs.localdomain.local essio1-hs
10.0.11.3 essio2-hs.localdomain.local essio2-hs
```

You can run a DNS instead of using the /etc/hosts file if the container can properly resolve all the nodes correctly during the essrun run time. It is a best practice to use the /etc/hosts file.

Note: For more information about the IBM Storage Scale System deployment, see the IBM Storage Scale System Quick Deployment Guide.

3.2.1 Loading the configuration

To begin the configuration, run essrun config load. With this option, essrun performs a basic configuration for the nodes and determines the node information based on virtual product data. It also exchanges the SSH keys across all nodes. This task is the first task that must be done because the configuration and information that is gathered during the running of this command is needed and used by other downstream essrun tasks.

The NODE-LIST is a comma-separated list of the IBM Storage Scale System nodes. Run the essrun config load command against all the storage nodes (including the EMS), as shown in Example 3-2.

Example 3-2 The essrun config load command

```
# essrun -N allESSNodes(management hostnames) config load -p
```

Note: For new IBM Storage Scale Systems deployments, you must update the nodes. To do so, run essrun --precheck, and then run the actual update after essrun config load.

3.2.2 Defining a network

The essrun command has a network option that you can use to create the high-speed network for the IBM Storage Scale System. Create the network bonds before cluster creation so that the cluster is created over the high-speed network.

Example 3-3 shows the command to define the network.

Example 3-3 The essrun network command

```
# essrun -N allESSNodes(management hostnames) network --suffix=YourSuffix
```

You can also create network bonds manually from any IBM Storage Scale System node by using the `essgennetworks` command. This command is applicable to any network with Ethernet and InfiniBand interfaces. This command analyzes the `/etc/hosts` file in the EMS node and (based on the input that is provided in that file) creates the bonded connection on the high-speed interfaces in the target node.

This command also supports Virtual Protocol Interconnect (VPI) adapter interface switching between InfiniBand and Ethernet. You can use the `--change` option and select a target interface type of either InfiniBand or Ethernet and select the target port that needs conversion to a different type of protocol. There is an option that is named `--device` that you can use to select a particular device to perform the conversion on. Once complete, restart the host to enable the new interface type. Make sure that the Mellanox Tools Services (MTS) is running on the node before doing any type of interface type conversion.

IBM Storage Scale System 6000 supports the following NVIDIA CX7 cards:

- 400 Gb single port (InfiniBand only) x16 Gen5
- 200 Gb VPI dual port (InfiniBand/Ethernet) x16 Gen5

With these cards, you may use the **`essgennetworks`** command to switch the VPI card between InfiniBand and Ethernet.

Note: The **`essgennetworks`** command has many options. For basic configurations, it is a best practice to use **`essrun`**. For more information about the `essgennetworks` command, see the IBM Storage Scale System: Command Reference.

3.2.3 Creating a cluster

To create a cluster, use the essrun command, as shown in Example 3-4. Create the cluster with the I/O nodes, and then add the EMS to the cluster.

Example 3-4 The essrun cluster command

```
# essrun -N IONodes(management hostnames) cluster --suffix=YourSuffix
# essrun -N oneIONode(management hostname) cluster --add-ems emsNode
--suffix=YourSuffix
```

3.2.4 Creating a file system

To create the file system in the newly created cluster, run `essrun filesystem`. As with the cluster creation, the file system is created against the I/O nodes, but in this case you do not need to run an additional command to include the EMS.

The `essrun filesystem` command creates combined meta-data + data VDisk sets by using a default RAID code and block size. You can use additional flags to customize or use the `mmvdisk` command directly for advanced configurations.

For the IBM Storage Scale System 6000 file system, an 8 MB or larger block size, 8+2p raid code, and an 80% set-size are best practices.

Note: Before creating a recovery group or a file system, check whether a self-encrypted drive (SED) is required. You cannot enroll recovery groups for the SED support after they are created.

Example 3-5 shows the `essrun filesystem` command.

Example 3-5 The essrun filesystem command

```
# essrun -N ESSIONodesInCluster(management hostname) filesystem --suffix=-hs
```

3.3 Configuring Cluster Export Services

To deploy the NVIDIA DGX cluster by using the NVIDIA Base Command Manager (BCM), the user home directories (`home`) and shared data (`cm_shared`) directories must be on NFS file systems that are shared with the head node. You configure and export these NFS shares from the IBM Storage cluster by using the CES/Protocol nodes.

You can configure the IBM Storage Scale System Solution with the IBM Storage Scale System Utility Nodes (IBM 5149-23E) for CES/Protocol nodes. These servers are similar to the IBM Storage Scale System EMS (has the same IBM part number) with two CPU sockets.

Sometimes, organizations have restrictions on the Linux servers and the operating systems that can be installed on these servers for the CES/Protocol nodes. Users may deploy the CES/Protocol nodes on their own servers that run an operating system that IBM Storage Scale supports.

For a list of the supported operating systems, see A2.1 in the IBM Storage Scale FAQ.

You can deploy The CES/Protocol nodes of the IBM Storage Scale storage cluster by following a manual method or by using the Installation Toolkit. Here is an example of deploying and configure the CES/Protocol nodes by using the manual method and the following software:

- Supported operating system on the CES/Protocol node: Red Hat Enterprise Linux 9.2 (RHEL)
- IBM Storage Scale 5.2.1-0
- OpenFabrics Enterprise Distribution (OFED) for the NVIDIA ConnectX-6 InfiniBand adapters: `MLNX_OFED_LINUX-23.10-2.1.3.1` (`MLNX_OFED_LINUX-23.10-2.1.3.1-rhel9.2-x86_64.iso`)

3.3.1 Preparing the environment on Linux nodes

All the hostnames for the GPFS daemon interface (for GPFS daemon-to-daemon communication) of both the IBM Storage Scale storage and the DGX client clusters; the Cluster Export Services (CES) IP interfaces; the AFM gateway; and the AFM target must be resolvable. These hostnames are maintained in the `/etc/hosts` file locally on all the servers with fully qualified hostnames.

The `yum` repository was created so that any missing operating system prerequisites for the IBM Storage Scale installation could be installed.

For an air-gapped environment, you can set up the `yum` repository for RHEL 9.2 (for example, `/etc/yum.repos.d/rhel92-local.repo`) by downloading the RHEL 9.2 full image onto the CES/Protocol node and mounting the iso image locally as a DVD.

For more information about how to set up the yum repository of a locally mounted DVD on RHEL 9, see Need to set up yum repository for locally mounted DVD on Red Hat Enterprise Linux 9.

Installation prerequisite for MLNX_OFED

After you create the yum repository, install the prerequisite for OFED, as shown in Example 3-6.

Example 3-6 Installing the prerequisite software

```
yum clean all
yum repolist
dnf install tk gcc-gfortran perl
```

The ISO image `MLNX_OFED_LINUX-23.10-2.1.3.1-rhel9.2-x86_64.iso` was downloaded on the CES/Protocol node and locally mounted as a DVD, as shown in Example 3-7.

Example 3-7 Mounting and installing OFED software

```
mkdir /mnt/mofed
mount -o loop MLNX_OFED_LINUX-23.10-2.1.3.1-rhel9.2-x86_64.iso /mnt/mofed
cd /mnt/mofed
./mlnxofedinstall
reboot
```

Note: After installing the Mellanox OFED, restart the system to help ensure that the drivers and configurations are loaded into the kernel and take effect.

Creating the IP network and passwordless SSH

In this example, the IBM Storage Scale storage cluster of the IBM Storage Scale 6000 is defined with a daemon network on IP over InfiniBand (IPoIB). If the InfiniBand adapter of the CES/Protocol node has more than a single port, such as the dual-port NVIDIA ConnectX-6 InfiniBand adapter, you must set up the bonded IPoIB for IBM Storage Scale daemon network by using the network manager tools, such as `nmtui` or `nmcli`, and then restart the system.

GPFS requires password-less SSH for the root user from the “administration nodes” to all nodes in the IBM Storage Scale storage cluster. In this example, all nodes in the IBM Storage Scale storage cluster are “administration nodes” and a `sudo` wrapper is not used, as shown in Example 3-8.

Example 3-8 The hostnamectl command to configure passwordless SSH

```
hostnamectl set-hostname ces01 ssh-keygen
ssh-copy-id root@ces01-ib ; ssh-copy-id root@ces02-ib
ssh-copy-id root@ems01-ib
ssh-copy-id root@n01a-ib ;  ssh-copy-id root@n01b-ib
ssh-copy-id root@n02a-ib ; ssh-copy-id root@n02b-ib
```

Installing the IBM Storage Scale software

Any prerequisite software that is missing from the operating system for IBM Storage Scale must be installed. To do so, run the command in Example 3-9 to check and install the software by using the images in the `yum` repository that you set up.

Example 3-9 Prerequisite software check

```
dnf install -y  ksh kernel-devel kernel-headers gcc-c++
```

After the IBM Storage Scale 5.2.1 installation image (`Storage_IBM Storage Scale_Data_Management-5.2.1.0-x86_64-Linux-install`) is downloaded locally on the CES/Protocol node in `/tmp`, extract the IBM Storage Scale software packages and then install the required IBM Storage Scale software, as shown in Example 3-10. By default, the software is extracted to the directory `/usr/lpp/mmfs/<version>`, where <version> is the IBM Storage Scale version, which in this case is `/usr/lpp/mmfs/5.2.1`. You can change the directory by using the `–dir` option. Also, the license agreement can be accepted automatically if you use the `–silent` option.

Example 3-10 IBM Storage Scale component installation

```
cd /tmp
/bin/sh  ./Storage_IBM Storage Scale_Data_Management-5.2.1.0-x86_64-Linux-install
–silent

cd /usr/lpp/mmfs/5.2.1.0/gpfs_rpms/
dnf install -y gpfs.adv-5.2.1-0.x86_64.rpm gpfs.base-5.2.1-0.x86_64.rpm \
gpfs.compression-5.2.1-0.x86_64.rpm gpfs.crypto-5.2.1-0.x86_64.rpm \
gpfs.docs-5.2.1-0.noarch.rpm gpfs.gpl-5.2.1-0.noarch.rpm \
gpfs.gskit-8.0.55-19.1.x86_64.rpm gpfs.license.dm-5.2.1-0.x86_64.rpm \
gpfs.msg.en_US-5.2.1-0.noarch.rpm
```

In this example, the additional IBM Storage Scale software for deploying Cluster Export Services is installed before you complete the steps to configure GPFS on the node, as shown in Example 3-11.

Example 3-11 Cluster Export Service installation

```
cd /usr/lpp/mmfs/5.2.1.0/ganesha_rpms/rhel9
dnf install -y  gpfs.nfs-ganesha-5.7-ibm022.00.el9.x86_64.rpm \
gpfs.nfs-ganesha-gpfs-5.7-ibm022.00.el9.x86_64.rpm \
gpfs.nfs-ganesha-utils-5.7-ibm022.00.el9.x86_64.rpm
```

Note: If you plan to install and configure CES, all protocols that are available on your platform must be installed. Installing a subset of available protocols is not supported.

The RPMs for CES SMB and CES S3, with the prerequisite for CES S3 bzip2 and boost-thread, are installed, but only the NFS service that is needed for the NVIDIA DGX SuperPOD is configured, as shown in Example 3-12.

Example 3-12 Additional installation for the CES components

```
cd /usr/lpp/mmfs/5.2.1.0/smb_rpms/rhel9/
dnf install -y gpfs.smb-4.19.7_gpfs_3-2.el9.x86_64.rpm gpfs_3-2.el9.x86_64.rpm

dnf install -y bzip2 boost-thread

cd /usr/lpp/mmfs/5.2.1.0/s3_rpms/rhel9
dnf install -y gpfs.mms3-5.2.1-0.el9.x86_64.rpm
noobaa-core-5.15.4-20240704.el9.x86_64.rpm
```

As a best practice, enable IBM Storage Scale health and performance monitoring on the CES/Protocol nodes. The IBM Storage Scale performance monitoring packages are also installed, as shown in Example 3-13.

Example 3-13 IBM Storage Scale health monitor installation

```
cd /usr/lpp/mmfs/5.2.1.0/zimon_rpms/rhel9
dnf install -y gpfs.gss.pmsensors-5.2.1-0.el9.x86_64.rpm \
gpfs.pm-ganesha-10.0.0-2.el9.x86_64.rpm
```

3.3.2 Adding a node to an IBM Storage Scale storage cluster on the IBM Storage Scale storage solution

Note: If the absolute path of the IBM Storage Scale CLI commands that start with “mm” are not explicitly stated in the examples, the directory for the IBM Storage Scale “mm” commands (`/usr/lpp/mmfs/bin`) is assumed to be in the `PATH` environment variable.

The GPFS portability layer is a loadable kernel module that must be built and loaded for GPFS to start. It enables the GPFS daemon to interact with the operating system. The GPFS kernel module must be rebuilt every time that the Linux kernel is updated.

To add the CES/Protocol nodes to the IBM Storage Scale storage cluster, the protocol nodes must have the “server” type GPFS license. In addition, a new nodeclass (`ces`) can be defined to simplify administration management on the CES nodes, such as setting and activating the tuning of the IBM Storage Scale configuration. All the GPFS commands are run from one of the administration nodes in the existing IBM Storage Scale storage cluster. When GPFS is started and active on the CES nodes, it confirms that the GPFS portability layer was successfully built and installed on the nodes.

Example 3-14 shows adding an IBM Storage Scale node to the cluster.

Example 3-14 Adding an IBM Storage Scale node to the cluster

```
#Running from the EMS node:
mmaddnode -N ces01-ib, ces02-ib
mmchlicense server –accept -N ces01-ib, ces02-ib
mmcrnodeclass ces  -N ces01-ib, ces02-ib
mmstartup -N ces01-ib, ces02-ib
```

The GPFS configuration parameters on the `ces nodelcass` were adjusted for tuning the CES/Protocol node before configuring the CES services, as shown in Example 3-15.

Example 3-15 GPFS configuration

```
mmchconfig verbsRdma=enable -N ces mmchconfig verbsRdmaSend=yes -N ces
mmchconfig verbsPorts=" mlx5_0/1 mlx5_3/1" -N ces mmchconfig
ignorePrefetchLUNCount=yes -N ces
mmchconfig numaMemoryInterleave=yes -N ces
mmchconfig nsdClientCksumTypeLocal=ck64 -N
ces mmchconfig nsdClientCksumTypeRemote=ck64 -N ces
mmchcoofig workerThreads=1024 -N ces
mmchconfig pagePool=48G -N ces
mmchconfig maxFilesToCache=4M -N ces
mmchconfig maxStatCache=2M -N ces
mmshutdown -N ces mmstartup -N ces
```

3.3.3 Configuring CES on the CES/Protocol nodes

The CES requires that the CES node have access to the path that is configured for the CES shared root (`cesSharedRoot`). This shared root can be a dedicated GPFS file system, a file set, or a directory on the GPFS file system that is mounted on all CES nodes. It is used to store the CES shared configuration data for protocol-specific purposes, including recovery.

The size for the CES shared root file system should be greater than or equal to 4 GB; a starting inode limit of 5000; and a block size of 256 KB. In Example 3-16, the `cesShareRoot` is a dedicated GPFS file system that is mounted at `/gpfs/cesroot`.

Example 3-16 CES configuration

```
mmchconfig cesSharedRoot=/gpfs/cesroot -i
mmchnode --ces-enable -N ces

mmces address add --ces-ip 10.70.10.94 --ces-node ces01-ib

mmces address add --ces-ip 10.70.10.95 --ces-node ces02-ib
```

The CES final configuration can be verified by running the following command:

```
mmlscluster --ces
```

3.3.4 Enabling the CES NFS protocol

When the NFS or SMB protocol is used to access files on the IBM Storage Scale file systems, the authentication and ID mapping must be setup before enabling the protocol.

In Example 3-17, the user-defined ID mapping is configured. The NFS protocol support for both 4.0 and 4.1 are enabled and the NFS service is started.

Example 3-17 Enabling the NFS service

```
mmuserauth service create --data-access-method file --type userdefined
mmces service enable NFS
mmnfs export change MINOR_VERSIONS=0,1 mmces service start NFS
mmces state show -a
```

3.3.5 Exporting the required NFS shares for BCM

The BCM deployment of the NVIDIA DGX SuperPOD requires two NFS exported file systems: `home` and `cm-shared`. The GPFS file systems `/gpfs/home` and `/gpfs/cm-shared` are created and the NFS exports are created. These tasks can be accomplished by using the IBM Storage Scale GUI.

In Example 3-18, the Ethernet network of DGX login nodes and the DGX H100 nodes are in different network subnets (10.70.10.64/26 and 10.70.10.0/26). There are two NFS exports for each of the file systems.

Example 3-18 List of NFS exports

```
mmnfs export list

Path        Delegations  Clients           Access_Type  Protocols  Transports
Squash          Anonymous_uid  Anonymous_gid  SecType  PrivilegedPort
DefaultDelegations  Manage_Gids  NFS_Commit  Pseudo Path

----------  -----------  ----------------  -----------  ---------  ----------
--------------  -------------  -------------  -------  --------------
------------------  -----------  ----------  -----------

/gpfs/home  NONE         10.70.10.64/26  RW           3          TCP
NO_ROOT_SQUASH  -2           -2           SYS     FALSE          NONE
FALSE        FALSE       /home

/gpfs/home  NONE         10.70.10.0/26   RW           3          TCP
NO_ROOT_SQUASH  -2           -2           SYS     FALSE          NONE
FALSE        FALSE       /home



Path            Delegations  Clients          Access_Type  Protocols  Transports
Squash          Anonymous_uid  Anonymous_gid  SecType  PrivilegedPort
DefaultDelegations  Manage_Gids  NFS_Commit  Pseudo Path

--------------  -----------  ----------------  -----------  ---------  ----------
--------------  -------------  -------------  -------  --------------
------------------  -----------  ----------  -----------

/gpfs/cm-shared  NONE         10.70.10.64/26  RW           3          TCP
NO_ROOT_SQUASH  -2           -2           SYS     FALSE          NONE
FALSE        FALSE       /cm-shared

/gpfs/cm-shared  NONE         10.70.10.0/26   RW           3          TCP
NO_ROOT_SQUASH  -2           -2           SYS     FALSE          NONE
FALSE        FALSE       /cm-shared
```

Reference:

For more information about the topics in 3.3, “Configuring Cluster Export Services” on page 53, see the following resources:

- Quick Reference on Installation and protocols deployment on Linux nodes for IBM Storage Scale 5.2.1
- Installing Cluster Export Services as part of installing IBM Storage Scale on Linux systems
- Managing protocol user authentication for IBM Storage Scale multi-protocol access: POSIX, NFS, SMB, and object

3.4 NVIDIA DGX SuperPOD

The NVIDIA Base Command Manager (BCM) is the cluster manager software that built on major Linux distributions. It contains tools and applications to facilitate the installation, provisioning, administration, and monitoring of a cluster, and the installation of third-party software.

The step-by-step deployment of the NVIDIA DGX SuperPOD cluster is modified to include the deployment of the IBM Storage Scale client cluster on the NVIDIA DGX SuperPOD cluster by following the manual installation methods for deploying and configuring IBM Storage Scale and the IBM Storage Scale Container Storage Interface (CSI).

For the NVIDIA DGX SuperPOD, IBM Storage Scale 5.2.1.0 is installed on the DGX cluster running Ubuntu 22.04, which is the supported operating system for NVIDIA DGX OS6.

3.4.1 Preparing the environment on the NVIDIA DGX SuperPOD

The IBM Storage Scale cluster can be set up where the administration command (`mmstartup`) can be run only from a subset of the nodes (the “administration” nodes). Passwordless SSH must be set up between all nodes in the cluster to enable communication between the “administration” nodes and other cluster nodes by using FQDNs, IP addresses, and hostnames.

The `/etc/hosts` file for the software images in BCM for the NVIDIA DGX SuperPOD cluster (the login nodes and the DGX compute nodes) is later modified to include the hostnames for the IBM Storage Scale daemon node names (for IBM Storage Scale daemon-to-daemon communication) of both the DGX IBM Storage Scale client cluster and the IBM Storage Scale cluster components (the IBM Storage Scale, the Cluster Export Services (CES) IP interfaces, and the AFM gateway).

For this example, the passwordless SSH for the root user on the DGX cluster among all nodes in the cluster (the NVIDIA DGX SuperPOD login nodes, the DGX nodes, and the run:ai nodes) should be set up before deploying the IBM Storage Scale client cluster.

This section outlines the procedures to install IBM Storage Scale and software that is required to create the client IBM Storage Scale cluster on the DGX cluster. After the login nodes and the DGX nodes are reprovisioned with the image, including the IBM Storage Scale software, the IBM Storage Scale client cluster is created. The following steps are required so that the IBM Storage Scale client cluster configuration is preserved between cluster restarts.

Note: The steps that are outlined in this example were adapted from a successful customer deployment by a joint IBM and NVIDIA team.

To begin, install the prerequisite software for IBM Storage Scale deployment, which includes the following items:

- Operating Systems software requirements for Ubuntu Linux packages:
  - `linux-generic`.
  - `cpp`.
  - `gcc`.
  - `g++`.
  - `binutils`.
  - `libelf-dev`.
- Prerequisites for IBM Storage Scale health and performance monitoring:
  - Python 3.
  - Sqlite3.
  - Ubuntu 22.04: libsqlite3-0.
- For the IBM Storage Scale management GUI, which is required for use with the REST API for IBM Storage Scale CSI driver:
  - A PostgreSQL server with a contrib package. The PostgreSQL server must be specific to the Linux distribution.
  - OpenSSL for HTTPS certificate management.

The DGX cluster can be deployed by using the NVIDIA Base Command Manager (BCM). To do so, follow these guidelines from Bright Computing.

Note: In this instance, the Login Node was deployed with the 5.15.0 kernel-based image for Ubuntu 22.04 instead of the 5.19 Hardware Enablement kernel that is in the default DGX deployment because IBM Storage Scale is supported on the 5.15 LTS release kernel but not the 5.19 Hardware Enablement kernel.

To begin, request the kernel 5.15.0-based `slogin-image.tar.gz` file from NVIDIA BCM Support and copy it to `/tmp` on the head node. Complete the following steps:

1. Create a directory in `/cm/images` for the image `login-5.15.0-image` (Example 3-19).

Example 3-19 Creating a directory for the image login-5.15.0-image

```
cd /cm/images/login-5.15.0-image
tar xvf /tmp/slogin-image.tar.gz
```

2. Import the image into the BCM `cmsh` shell (Example 3-20).

Example 3-20 Importing the image into the BCM cmsh shell

```
cmsh
headnode]% softwareimage
[headnode->softwareimage]% add login-5.15.0-image
[headnode->softwareimage[login-5.15.0-image]]% set kernelversion 5.15.0-107-generic
[headnode->softwareimage[login-5.15.0-image*]]% commit
```

Wait until the `initrd` is generated, and then exit `cmsh`.

3. Install OFED into the login node image (Example 3-21).

Example 3-21 Installing OFED into the login node image

```
root@headnode:~#
/cm/local/apps/mlnx-ofed23.10/current/bin/mlnx-ofed23.10-install.sh
--softwareimage login-5.15.0-image
```

4. Update the SSH key (Example 3-22).

Example 3-22 Updating the SSH key

```
root@headnode:~# cp -p /cm/images/dgx-os-6.2-h100-image/root/.ssh/authorized_keys
/cm/images/login-5.15-0-image/root/authorized_keys
```

Note: On the DGX cluster, each of the login, DGX, and run:ai nodes have two IPoIB interfaces that are assigned IP addresses in the same subnet, per the NVIDIA BasePOD/NVIDIA DGX SuperPOD deployment reference. The fix `net.ipv4.conf.interface.arp_filter` was applied to counter the instability of the GPFS status on the DGX node.

5. Fix the `sysctl` setting so all the interfaces respond to ping (Example 3-23).

Example 3-23 Fixing the sysctl setting

```
root@headnode:~# vim /cm/images/login-5.15-0-image/etc/sysctl.d/90-cm-sysctl.conf
Change net.ipv4.conf.all.arp_filter = 1
to
net.ipv4.conf.all.arp_filter = 0
```

Note: The kernel can respond to ARP requests with addresses from other interfaces, which increase the chance of successful communication. IP addresses are owned by the complete host on Linux, not by specific interfaces.

6. Set the login nodes to use the new image (Example 3-24).

Example 3-24 Setting the login nodes to use the new image

```
cmsh
headnode]% category use slogin
[headnode->category[slogin]]% set softwareimage login-5.15-0-image
[headnode->category[slogin*]]% commit
```

7. Restart the login nodes.

3.4.2 Installing GPFS on the node images

To install GPFS on the head node, complete the following steps:

1. Copy the IBM Storage Scale installation image file `Storage_IBM Storage Scale_Data_Management-5.2.1-0-x86_64-Linux-install` to `/tmp` on the head node.
2. Extract the installation (`*.deb` and `*.rpm`) files to a temporary directory (Example 3-25).

Example 3-25 Extracting the installation files to a temporary directory

```
root@headnode:~# sh. /tmp/Storage_IBM Storage
Scale_Data_Management-5.2.1-0-x86_64-Linux-install --dir /tmp/gpfs
--silent
```

To install GPFS on the login node, complete the following steps:

1. Clone the current image in `cmsh: [headnode]% softwareimage` (Example 3-26).

Example 3-26 Cloning the current image in cmsh: [headnode]% softwareimage

```
headnode]% softwareimage
[headnode->softwareimage]% clone login-5.15-0-image login-5.15-0-gpfs-image
[headnode->softwareimage*]% commit
```

2. Collect the selected Ubuntu installation files into the temporary directory `/tmp/gpfs/gpfs_debs` on the head node and copy the following files from `/tmp/gpfs/gpfs_debs` to `/tmp` onto the login node `slogin-01`:

   ```
   gpfs.adv_5.2.1-0_amd64.deb
   gpfs.base_5.2.1-0_amd64.deb
   gpfs.compression_5.2.1-0_amd64.deb
   gpfs.crypto_5.2.1-0_amd64.deb
   gpfs.docs_5.2.1-0_all.deb
   gpfs.gpl_5.2.1-0_all.deb
   gpfs.gskit_8.0.55-19.1_amd64.deb
   gpfs.license.dm_5.2.1-0_amd64.deb
   gpfs.msg.en-us_5.2.1-0_all.deb
   ```

3. Use SSH to access `slogin01`, install the packages, and build the kernel modules (Example 3-27).

Example 3-27 Building the kernel modules

```
root@slogin-01:/# apt-get install ksh93u+m
root@slogin-01:/# cd /tmp
root@slogin-01:/tmp# dpkg -i *.deb
root@slogin-01:/tmp# rm -f *.deb
root@slogin-01:/tmp# /usr/lpp/mmfs/bin/mmbuildgpl --build-package
root@slogin-01:/tmp# dpkg -i gpfs.gplbin-5.15.0-107-generic_5.2.1-0_amd64.deb
```

4. Restart the login nodes. The required IBM Storage Scale software is installed and ready for the next step, which is creating the IBM Storage Scale client cluster.
5. Go to `cmsh` and grab the image. Set the login nodes in `cmsh` to use the new image (Example 3-28 on page 63).

Example 3-28 Setting the login nodes in cmsh to use the new image

```
[headnode]% device
[headnode->device]% grabimage -w -i login-5.15-0-gpfs-image slogin-01
[headnode->device*]% commit
[headnode]% category
[headnode->category]% use slogin
[headnode->category[slogin]]% set softwareimage login-5.15-0-gpfs-image
[headnode->category[slogin*]]% commit
```

To install GPFS on the DGX nodes, complete the following steps:

1. Clone the current image in the BCM cmsh shell (Example 3-29).

Example 3-29 Cloning the current image in the BCM cmsh shell

```
[headnode->softwareimage]% clone dgx-os-6.2-h100-image dgx-os-6.2-h100-gpfs-image
[headnode->softwareimage]% commit
```

2. Copy the following files from `/tmp/gpfs/gpfs_debs` to `/tmp` on the DGX node `dgx-01`: `gpfs.adv_5.2.1-0_amd64.deb`:

   ```
   gpfs.base_5.2.1-0_amd64.deb
   gpfs.compression_5.2.1-0_amd64.deb
   gpfs.crypto_5.2.1-0_amd64.deb
   gpfs.docs_5.2.1-0_all.deb
   gpfs.gpl_5.2.1-0_all.deb
   gpfs.gskit_8.0.55-19.1_amd64.deb
   gpfs.license.dm_5.2.1-0_amd64.deb
   gpfs.msg.en-us_5.2.1-0_all.deb
   ```

3. Use SSH to access dgx-01, install the packages, and build the kernel modules (Example 3-30).

Example 3-30 Building the kernel modules

```
root@dgx-01:/# apt-get install ksh93u+m
root@dgx-01:/# cd /tmp
root@dgx-01:/tmp# dpkg -i *.deb
root@dgx-01:/tmp# rm -f *.deb
root@dgx-01:/tmp# /usr/lpp/mmfs/bin/mmbuildgpl --build-package
root@dgx-01:/tmp# dpkg -i gpfs.gplbin-5.15.0-1046-nvidia_5.2.1-0_amd64.deb
```

4. In `cmsh`, grab and commit the image (Example 3-31).

Example 3-31 Committing the image

```
[headnode]% device
[headnode->device]% grabimage -w -i dgx-os-6.2-h100-gpfs-image dgx-01
[headnode->device*]% commit
[headnode]% category
[headnode->category]% use dgx-h100
[headnode->category[dgx-h100]]% set softwareimage dgx-os-6.2-h100-gpfs-image
[headnode->category[dgx-h100*]]% commit
```

3.4.3 Setting the DGX nodes in cmsh to use the new image

The default configuration for BCM is that all files under `/var` are not preserved between node restarts. Because the IBM Storage Scale configuration and log files are created and stored in directories under `/var/mmfs`, the BCM configuration must be customized to exclude the IBM Storage Scale (GPFS) storage from being deleted between node restarts.

Note: For more information, see How do I integrate GPFS with Bright?

Ensure that the DGX nodes restart after the required IBM Storage Scale software is installed so that they are ready for the next step of creating the IBM Storage Scale client cluster.

Complete the following steps:

1. Add the files in Example 3-32 to `cmsh`.

Example 3-32 Adding files to cmsh

```
[headnode]% category
[headnode->category]% use slogin
[headnode->category[slogin]]% set excludelistsyncinstall
```

#Add the following entries:

```
no-new-files: - /var/mmfs/*
no-new-files: - /var/adm/ras/*
no-new-files: - /opt/IBM/zimon/data/*
no-new-files: - /opt/IBM/zimon/*
no-new-files: - /var/lib/postgresql/*
no-new-files: - /var/lib/mmfs/*
no-new-files: - /var/log/*

[headnode->category[slogin*]]% set excludelistgrab
```

#Add the following entries:

```
/var/lib/postgresql/*
/var/mmfs/callhome/*
/var/mmfs/ces/*
/var/mmfs/ccr/*
/var/mmfs/etc/*
/var/mmfs/gen/*
/var/mmfs/mmbackup/*
/var/mmfs/mmpmon/*
/var/mmfs/run
/var/mmfs/ssl/*
/var/mmfs/tmp/*
/var/mmfs/mmsysmon/mmsysmonitor.socket
/var/mmfs/mmsysmon/mmsysmon.json
/var/mmfs/mmsysmon/_perfmon.keys
/var/adm/ras/*
/opt/IBM/zimon/data/*
/opt/IBM/zimon/*
/var/lib/mmfs/*
/var/log/*
```

```
[headnode->category[slogin*]]% set excludelistgrabnew
```

#Add the following entries:

```
/var/lib/postgresql/*
/var/mmfs/callhome/*
/var/mmfs/ces/*
/var/mmfs/ccr/*
/var/mmfs/etc/*
/var/mmfs/gen/*
/var/mmfs/mmbackup/*
/var/mmfs/mmpmon/*
/var/mmfs/run
/var/mmfs/ssl/*
/var/mmfs/tmp/*
/var/mmfs/mmsysmon/mmsysmonitor.socket
/var/mmfs/mmsysmon/mmsysmon.json
/var/mmfs/mmsysmon/_perfmon.keys
/var/adm/ras/*
/opt/IBM/zimon/data/*
/opt/IBM/zimon/*
/var/lib/mmfs/*
/var/log/*
```

```
[headnode->category[slogin*]]% set excludelistupdate
```

Add the following entries:

```
no-new-files: - /var/mmfs/*
no-new-files: - /var/adm/ras/*
no-new-files: - /opt/IBM/zimon/data/*
no-new-files: - /opt/IBM/zimon/*
no-new-files: - /var/lib/postgresql/*
no-new-files: - /var/lib/mmfs/*
no-new-files: - /var/log/*
```

```
[headnode->category[slogin*]]% commit
```

2. Perform the same actions for the `dgx-h100` category.
3. Create SSH keys for root access between nodes.
4. Log in to `slogin-01`.
5. Create an SSH key, as shown in Example 3-33.

Example 3-33 Creating an SSH key

```
root@slogin-01:/# ssh-keygen -t ecdsa
Generating public/private ecdsa key pair.
Enter the file in which to save the key (/root/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter the same passphrase again:
Your identification has been saved in /root/.ssh/id_ecdsa
Your public key has been saved in /root/.ssh/id_ecdsa.pub
root@slogin-01:~# cat ~/.ssh/id_ecdsa.pub >> ~/.ssh/authorized_keys
```

6. Copy the keys into the GPFS images, as shown in Example 3-34.

Example 3-34 Copying the keys into the GPFS images

```
root@headnode:~# cd /cm/images/login-5.15.0-gpfs-image/root
root@headnode:~# rsync -av slogin-01:/root/.ssh/ .ssh/
root@headnode:~# cd /cm/images/dgx-os-6.2-h100-gpfs-image/root
root@headnode:~# rsync -av slogin-01:/root/.ssh/ .ssh/
```

7. Restart the nodes (or synchronize the `.ssh` directory to the nodes) to install the keys.

Important: The steps in this section must be performed after IBM Storage Scale cluster creation so that the information is preserved between node restarts.The entry `/var/lib/postgresql/*` is specifically for the DGX nodes with the operating system Ubuntu 22.04, and might be different for other operating systems.

3.4.4 Creating an IBM Storage Scale client cluster on a DGX cluster

The IBM Storage Scale client cluster is created on the DGX cluster with the login nodes and the DGX GPU nodes. Node quorum, which is the default quorum algorithm, is used in the IBM Storage Scale cluster definition for the DGX IBM Storage Scale client cluster. An odd number of quorum nodes, with a maximum of nine quorum nodes, is required.

After the DGX IBM Storage Scale client cluster is created, the remote cluster is defined and the IBM Storage Scale file system from the IBM Storage Scale storage cluster is remotely mounted on the DGX IBM Storage Scale client cluster for testing. The validation is completed before installing k8S and deploying run:ai on the DGX cluster. For the validation testing, three quorum nodes are defined. All the quorum nodes must have a server license.

Creating an IBM Storage Scale Client cluster on slogin2

Complete the following steps:

1. Check the firewall status on the DGX nodes to confirm that the firewall is disabled by running the following command:

   ```
   for i in 81 82 83 `seq 1 32` ; do ssh 10.40.1.$i "ufw status"; done
   ```

Note: If `slogin1` (`slogin1-ib.superpod.company.com`) was initially used as the temporary NFS server to provide NFS shares for the BCM, then the DGX IBM Storage Scale client cluster is first created while excluding `slogin1`. Later, this node is added to the DGX IBM Storage Scale client cluster after the NFS shares are changed to use the IBM Storage Scale file systems that are exported from the IBM Storage Scale CES/protocol nodes.

2. Create two nodefiles on the logged in node:
   – The list of the quorum nodes (`spod.nodelist.1`):

   ```
   ======================================================
   slogin1-ib.superpod.company.com:quorum-manager
   slogin2-ib.superpod.company.com:quorum-manager
   slogin3-ib.superpod.company.com:quorum-manager
   ======================================================
   ```

– The nodefile of the non-quorum nodes (`spod.nodelist.2`):

```
=====================================================
dgx01-ib.superpod.company.com
dgx02-ib.superpod.company.com
dgx03-ib.superpod.company.com
dgx04-ib.superpod.company.com
dgx05-ib.superpod.company.com
dgx06-ib.superpod.company.com
dgx07-ib.superpod.company.com
dgx08-ib.superpod.company.com
dgx09-ib.superpod.company.com
dgx10-ib.superpod.company.com
dgx11-ib.superpod.company.com
dgx12-ib.superpod.company.com
dgx13-ib.superpod.company.com
dgx14-ib.superpod.company.com
dgx15-ib.superpod.company.com
dgx16-ib.superpod.company.com
dgx17-ib.superpod.company.com
dgx18-ib.superpod.company.com
dgx19-ib.superpod.company.com
dgx20-ib.superpod.company.com
dgx21-ib.superpod.company.com
dgx22-ib.superpod.company.com
dgx23-ib.superpod.company.com
dgx24-ib.superpod.company.com
dgx25-ib.superpod.company.com
dgx26-ib.superpod.company.com
dgx27-ib.superpod.company.com
dgx28-ib.superpod.company.com
dgx29-ib.superpod.company.com
dgx30-ib.superpod.company.com
dgx31-ib.superpod.company.com
dgx32-ib.superpod.company.com
=====================================================
```

3. Modify the `/etc/hosts` file on `slogin1-ib` and ensure that the `/etc/hosts` file on the NVIDIA DGX SuperPOD login nodes and the DGX nodes contains the hostnames for the IBM Storage Scale daemon nodes (the IPoIB interfaces), as shown in Example 3-35.

Example 3-35 Modifying the /etc/hosts file on slogin1-ib

```
# for i in 82 83 `seq 1 16` ; do ssh 100.40.1.$i "hostname; scp -p
root@100.40.1.81:/etc/hosts /etc/hosts"; done
Running from slogin1-ib.superpod (slogin1): #/usr/lpp/mmfs/bin/mmcrcluster -N
spod.nodelist.1 #/usr/lpp/mmfs/bin/mmchlicense server --accept -N spod.nodelist.1
#/usr/lpp/mmfs/bin/mmstartup -a
#/usr/lpp/mmfs/bin/mmgetstate -a #/usr/lpp/mmfs/bin/mmaddnode -N spod.nodelist.2
#/usr/lpp/mmfs/bin/mmchlicense server --accept -N spod.nodelist.2
#/usr/lpp/mmfs/bin/mmstartup -a
#/usr/lpp/mmfs/bin/mmchcluster -C DEMO-dgx

root@slogi12:~# /usr/lpp/mmfs/bin/mmgetstate -a

Node number   Node name   GPFS state
-------------------------------------------------
```

```
1              slogin1-ib  active
2              slogin2-ib  active
3              slogin3-ib  active
4              dgx01-ib    active
5              dgx02-ib    active
6              dgx03-ib    active
7              dgx04-ib    active
8              dgx05-ib    active
9              dgx06-ib    active
10             dgx07-ib    active
11             dgx08-ib    active
12             dgx09-ib    active
13             dgx10-ib    active
14             dgx11-ib    active
15             dgx12-ib    active
16             dgx13-ib    active
17             dgx14-ib    active
18             dgx15-ib    active
19             dgx16-ib    active
20             dgx17-ib    active
21             dgx18-ib    active
22             dgx19-ib    active
23             dgx20-ib    active
24             dgx21-ib    active
25             dgx22-ib    active
26             dgx23-ib    active
27             dgx24-ib    active
28             dgx25-ib    active
29             dgx26-ib    active
30             dgx27-ib    active
31             dgx28-ib    active
32             dgx29-ib    active
33             dgx30-ib    active
34             dgx31-ib    active
35             dgx32-ib    active

root@slogin2:~# mmlscluster
GPFS cluster information
========================

  GPFS cluster name:         DEMO-dgx.superpod.company.com
  GPFS cluster ID:          1169196794576787503
  GPFS UID domain:           DEMO-dgx.superpod.company.com
  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     slogin1-ib.superpod.company.com  10.40.1.81 slogin1-ib.superpod.company.com
quorum-manager
```

```
2     slogin2-ib.superpod.company.com  10.40.1.82 slogin2-ib.superpod.company.com
quorum-manager
3     slogin3-ib.superpod.company.com  10.40.1.83 slogin3-ib.superpod.company.com
quorum-manager
4     dgx01-ib.superpod.company.com  10.40.1.1    dgx01-ib.superpod.company.com
5     dgx02-ib.superpod.company.com 10.40.1.2     dgx02-ib.superpod.company.com
6     dgx04-ib.superpod.company.com   10.40.1.4   dgx04-ib.superpod.company.com
7     dgx05-ib.superpod.company.com   10.40.1.5   dgx05-ib.superpod.company.com
8     dgx06-ib.superpod.company.com   10.40.1.6   dgx06-ib.superpod.company.com
9     dgx07-ib.superpod.company.com   10.40.1.7   dgx07-ib.superpod.company.com
10   dgx08-ib.superpod.company.com   10.40.1.8   dgx08-ib.superpod.company.com
11   dgx09-ib.superpod.company.com   10.40.1.9   dgx09-ib.superpod.company.com
12   dgx10-ib.superpod.company.com   10.40.1.10  dgx10-ib.superpod.company.com
13   dgx11-ib.superpod.company.com   10.40.1.11  dgx11-ib.superpod.company.com
14   dgx12-ib.superpod.company.com   10.40.1.12  dgx12-ib.superpod.company.com
15   dgx13-ib.superpod.company.com   10.40.1.13  dgx13-ib.superpod.company.com
16   dgx14-ib.superpod.company.com   10.40.1.14  dgx14-ib.superpod.company.com
17   dgx15-ib.superpod.company.com   10.40.1.15  dgx15-ib.superpod.company.com
18   dgx16-ib.superpod.company.com   10.40.1.16  dgx16-ib.superpod.company.com
19   dgx17-ib.superpod.company.com   10.40.1.17  dgx17-ib.superpod.company.com
20   dgx18-ib.superpod.company.com   10.40.1.18  dgx18-ib.superpod.company.com
21   dgx19-ib.superpod.company.com   10.40.1.19  dgx19-ib.superpod.company.com
22   dgx20-ib.superpod.company.com   10.40.1.20  dgx20-ib.superpod.company.com
23   dgx21-ib.superpod.company.com   10.40.1.21  dgx21-ib.superpod.company.com
24   dgx22-ib.superpod.company.com   10.40.1.22  dgx12-ib.superpod.company.com
25   dgx23-ib.superpod.company.com   10.40.1.23  dgx23-ib.superpod.company.com
26   dgx24-ib.superpod.company.com   10.40.1.24  dgx24-ib.superpod.company.com
27   dgx25-ib.superpod.company.com   10.40.1.25  dgx25-ib.superpod.company.com
28   dgx26-ib.superpod.company.com   10.40.1.26  dgx26-ib.superpod.company.com
29   dgx27-ib.superpod.company.com   10.40.1.27  dgx27-ib.superpod.company.com
30   dgx28-ib.superpod.company.com   10.40.1.28  dgx28-ib.superpod.company.com
31   dgx29-ib.superpod.company.com   10.40.1.29  dgx29-ib.superpod.company.com
32   dgx30-ib.superpod.company.com   10.40.1.30  dgx30-ib.superpod.company.com
33   dgx31-ib.superpod.company.com   10.40.1.31  dgx31-ib.superpod.company.com
34   dgx32-ib.superpod.company.com   10.40.1.32  dgx12-ib.superpod.company.com
```

4. Using the same nodesfiles, create a node class (by running `mmcrnodeclass`) for the DGX login nodes:
   - Nodefile to create IBM Storage Scale nodeclass spod.dgx-login:

     ```
     ========================================
     slogin1-ib.superpod.company.com
     slogin2-ib.superpod.company.com
     slogin3-ib.superpod.company.com
     ========================================
     ```

   - Nodefile to create IBM Storage Scale nodeclass dgxh100 for the DGX H100 nodes (spod.dgx-h100):

     ```
     ========================================
     dgx01-ib.superpod.company.com
     dgx02-ib.superpod.company.com
     dgx03-ib.superpod.company.com
     dgx04-ib.superpod.company.com
     dgx05-ib.superpod.company.com
     dgx06-ib.superpod.company.com
     ```

```
dgx07-ib.superpod.company.com
dgx08-ib.superpod.company.com
dgx09-ib.superpod.company.com
dgx10-ib.superpod.company.com
dgx11-ib.superpod.company.com
dgx12-ib.superpod.company.com
dgx13-ib.superpod.company.com
dgx14-ib.superpod.company.com
dgx15-ib.superpod.company.com
dgx16-ib.superpod.company.com
dgx17-ib.superpod.company.com
dgx18-ib.superpod.company.com
dgx19-ib.superpod.company.com
dgx20-ib.superpod.company.com
dgx21-ib.superpod.company.com
dgx22-ib.superpod.company.com
dgx23-ib.superpod.company.com
dgx24-ib.superpod.company.com
dgx25-ib.superpod.company.com
dgx26-ib.superpod.company.com
dgx27-ib.superpod.company.com
dgx28-ib.superpod.company.com
dgx29-ib.superpod.company.com
dgx30-ib.superpod.company.com
dgx31-ib.superpod.company.com
dgx32-ib.superpod.company.com
========================================
#/usr/lpp/mmfs/bin/mmcrnodeclass  login  -N spod.dgx-login
#/usr/lpp/mmfs/bin/mmcrnodeclass  dgxh100  -N spod.dgx-h100
```

5. You can tune the DGX IBM Storage Scale client cluster by using the commands that are shown in Example 3-36.

Example 3-36 Tuning commands for the DGX IBM Storage Scale client cluster

```
/usr/lpp/mmfs/bin/mmchconfig  verbsRdma=enable
/usr/lpp/mmfs/bin/mmchconfig  verbsRdmaSend=yes
/usr/lpp/mmfs/bin/mmchconfig  maxblocksize=16M
/usr/lpp/mmfs/bin/mmchconfig  maxFilesToCache=128K
/usr/lpp/mmfs/bin/mmchconfig  workerThreads=1024
/usr/lpp/mmfs/bin/mmchconfig  ignorePrefetchLUNCount=yes
/usr/lpp/mmfs/bin/mmchconfig  nsdClientCksumTypeLocal=ck64
/usr/lpp/mmfs/bin/mmchconfig  nsdClientCksumTypeRemote=ck64
/usr/lpp/mmfs/bin/mmchconfig  numaMemoryInterleave=yes
/usr/lpp/mmfs/bin/mmchconfig  maxMBpS=20000
/usr/lpp/mmfs/bin/mmchconfig  maxMBpS=100000=-N dgxh100
/usr/lpp/mmfs/bin/mmchconfig  verbsPorts="mlx5_0/1 mlx5_1/1" -N login
```

The configuration data for the cluster `DEMO-dgx.superpod.company.com` is shown in Example 3-37.

Example 3-37 Configuration data for the cluster DEMO-dgx.superpod.company.com

```
-----------------------------------------------------------
autoload no
dmapiFileHandleSize 32
minReleaseLevel 5.2.0.0
```

```
tscCmdAllowRemoteConnections no
ccrEnabled yes
sdrNotifyAuthEnabled yes
verbsRdma enable
verbsRdmaSend yes
maxblocksize 16M
maxFilesToCache 128K
workerThreads 1024
ignorePrefetchLUNCount yes
nsdClientCksumTypeLocal ck64
nsdClientCksumTypeRemote ck64
numaMemoryInterleave yes
[login]
verbsPorts mlx5_0/1 mlx5_1/1
[dgxh100]
verbsPorts mlx5_1/1 mlx5_7/1
[common]
maxMBpS 20000
[dgxh100]
maxMBpS 100000
[login]
pagepool 16G
[dgxh100]
pagepool 32G
[common]
clusterName DEMO-dgx.superpod.company.com
clusterId 11691967945767875033
cipherList AUTHONLY
[dgxh100,login]
maxStatCache 128K
[common]
adminMode central

File systems in cluster DEMO-dgx.superpod.company.com:
------------------------------------------------------
(none)
```

The IBM Storage Scale client cluster is now deployed on the DGX. This cluster includes three DGX login nodes (`slogin1`, `slogin2`, and `slogin3`) and 32 DGX H100 nodes.

6. Remotely mount the IBM Storage Scale file system from IBM Storage Scale with the native IBM Storage Scale Network Shared Disk (NSD) (a POSIX protocol), as described in “Mounting a remote GPFS file system” on page 72.

Mounting a remote GPFS file system

Table 3-1 summarizes an example of the commands to run to set up the remote cluster mount. All “mm” commands are in the directory `/usr/lpp/mmfs/bin`. The “mm” commands are cluster-wide commands, and can be run on any “administration” node with passwordless SSH for root access to other nodes in the IBM Storage Scale cluster. The IBM Storage Scale file system is `gpfs0`.

Table 3-1 Remote cluster mount command list

DGX IBM Storage Scale client cluster (accessingCluster)	IBM Storage Scale storage cluster (owningCluster)
	`mmauth genkey new`
	`mmauth update . -I AUTHONLY`
`mmauth genkey new`	
`mmauth update . -I AUTHONLY`	
`scp -p /var/mmfs/ssl/id_rsa.pub root@100.40.10.1:/tmp/IBM/DEMO-dgx.id_rsa.pub`	
`scp -p root@100.40.10.1: /var/mmfs/ssl/id_rsa.pub /tmp/IBM/DEMO-ISS.id_rsa.pub`	
	`mmauth add DEMO-dgx.superpod.company.com -k /tmp/IBM/DEMO-dgx.id_rsa.pub`
	`mmauth grant DEMO-dgx.superpod.company.com -f gpfs0`
`mmremotecluster add DEMO-ISS.superpod.company.com -n ems01-ib, n01a-ib, n02a-ib -k /tmp/IBM/DEMO-ISS.id_rsa.pub`	
`mmremotefs add gpfs0 -f gpfs0 -C DEMO-ISS.superpod.company.com -T /ess/gpfs0`	

Complete the following steps:

1. Confirm the connectivity between the IBM Storage Scale System cluster and the available file system, as shown in Example 3-38.

Example 3-38 Confirming the connectivity between the cluster and file system

```
root@slogin2:~# mmremotecluster show
Cluster name:    DEMO-ISS.superpod.company.com
Cluster ID:      16436673894563987403
Contact nodes:   ems01-ib.superpod.company.com, n01a-ib.superpod.company.com,
n02a-ib.superpod.company.com
SHA digest:      a2e6f0c2875669a0c81f076c7971e20d62e63b337a8c6a31c1389de9e55515cb
File systems:    gpfs0 (gpfs0)

root@slogin2:~# mmremotefs show
Local Name  Remote Name  Cluster name       Mount Point        Mount Options
Automount  Drive  Priority
```

```
gpfs0      gpfs0      DEMO-ISS.superpod.company.com /ess/gpfs0       rw
no           -        0

root@slogin2:~# mmmount /ess/gpfs0 -a
```

2. Use the `mmlsmount` command to confirm that the IBM Storage Scale file system from the IBM Storage Scale storage cluster is remotely mounted on the IBM Storage Scale client DGX nodes before proceeding with I/O to the file system.

   Example 3-39 shows an example response from running `mmlsmount`.

Example 3-39 Example response from running mmlsmount

```
root@slogin2:~# mmlsmount gpfs0 -L
File system gpfs0 (DEMO-ISS.superpod.company.com:gpfs0) is mounted on 45 nodes:
10.40.100.11  ces02-ib      DEMO-ISS.superpod.company.com
10.40.100.10  ces01-ib      DEMO-ISS.superpod.company.com
10.40.100.15  afm04-ib      DEMO-ISS.superpod.company.com
10.40.100.14  afm03-ib      DEMO-ISS.superpod.company.com
10.40.100.13  afm02-ib      DEMO-ISS.superpod.company.com
10.40.100.12  afm01-ib      DEMO-ISS.superpod.company.com
10.40.100.5   n02b-ib       DEMO-ISS.superpod.company.com (internal mount)
10.40.100.4   n02a-ib       DEMO-ISS.superpod.company.com (internal mount)
10.40.100.3   n01b-ib       DEMO-ISS.superpod.company.com (internal mount)
10.40.100.2   n01a-ib       DEMO-ISS.superpod.company.com (internal mount)
10.40.100.1   ems01-ib      DEMO-ISS.superpod.company.com
10.40.1.81    slogin1-ib      DEMO-dgx.superpod.company.com
10.40.1.82    slogin2-ib      DEMO-dgx.superpod.company.com
10.40.1.83    slogin3-ib      DEMO-dgx.superpod.company.com
10.40.1.6     dgx06-ib      DEMO-dgx.superpod.company.com
10.40.1.28     dgx28-ib      DEMO-dgx.superpod.company.com
10.40.1.22     dgx22-ib      DEMO-dgx.superpod.company.com
10.40.1.24     dgx24-ib      DEMO-dgx.superpod.company.com
10.40.1.1     dgx01-ib      DEMO-dgx.superpod.company.com
10.40.1.20     dgx20-ib      DEMO-dgx.superpod.company.com
10.40.1.11    dgx11-ib      DEMO-dgx.superpod.company.com
10.40.1.12    dgx12-ib      DEMO-dgx.superpod.company.com
10.40.1.5     dgx05-ib      DEMO-dgx.superpod.company.com
10.40.1.16    dgx16-ib      DEMO-dgx.superpod.company.com
10.40.1.25     dgx25-ib      DEMO-dgx.superpod.company.com
10.40.1.18     dgx18-ib      DEMO-dgx.superpod.company.com
10.40.1.27     dgx27-ib      DEMO-dgx.superpod.company.com
10.40.1.23     dgx23-ib      DEMO-dgx.superpod.company.com
10.40.1.30     dgx30-ib      DEMO-dgx.superpod.company.com
10.40.1.3     dgx03-ib      DEMO-dgx.superpod.company.com
10.40.1.4     dgx04-ib      DEMO-dgx.superpod.company.com
10.40.1.21     dgx21-ib      DEMO-dgx.superpod.company.com
10.40.1.31     dgx031-ib      DEMO-dgx.superpod.company.com
10.40.1.26     dgx16-ib      DEMO-dgx.superpod.company.com
10.40.1.2     dgx02-ib      DEMO-dgx.superpod.company.com
10.40.1.15    dgx15-ib      DEMO-dgx.superpod.company.com
10.40.1.13    dgx13-ib      DEMO-dgx.superpod.company.com
10.40.1.9     dgx09-ib      DEMO-dgx.superpod.company.com
10.40.1.7     dgx07-ib      DEMO-dgx.superpod.company.com
10.40.1.10    dgx10-ib      DEMO-dgx.superpod.company.com
10.40.1.32     dgx32-ib      DEMO-dgx.superpod.company.com
```

```
10.40.1.14    dgx14-ib      DEMO-dgx.superpod.company.com
10.40.1.8     dgx08-ib      DEMO-dgx.superpod.company.com
10.40.1.29     dgx029-ib      DEMO-dgx.superpod.company.com
10.40.1.19     dgx19-ib      DEMO-dgx.superpod.company.com
10.40.1.17     dgx17-ib      DEMO-dgx.superpod.company.com
```

Reference: For more information, see Accessing a remote GPFS file system.

Note: If the deployment schedule requires the DGX cluster deployment to start before the NFS shares are exported from the IBM Storage Scale cluster CES/Protocol nodes, you can set up a temporary NFS server as the login node (slogin1) to provide the NFS shares for BCM. First, create the DGX IBM Storage Scale client cluster while excluding `slogin1`, with one of the DGX nodes (`dgx01`) as the quorum nodes. After you change the NFS shares to use the NFS shares from the IBM Storage Scale cluster CES/Protocol node, add `slogin1` to the DGX IBM Storage Scale client cluster. You can change the node types for `slogin1` and `dgx01` by running the `mmchnode` command.

3. After you migrate the NFS shares for the BCM on the temporary NFS shares on `slogin1` to use the NFS shares from the IBM Storage Scale file systems that are exported from the IBM Storage Scale CES/protocol nodes, you can add the DGX login node `slogin1-Ib` to the DGX IBM Storage Scale client cluster by running the commands that are shown in Example 3-40. Run the commands from one of the DGX nodes with passwordless SSH for root to other nodes in the DGX cluster.

Example 3-40 Adding the DGX login node slogin1-Ib to the DGX IBM Storage Scale client cluster

```
#/usr/lpp/mmfs/bin/mmaddnode -N slogin1-ib.superepod.company.com
#/usr/lpp/mmfs/bin/mmchlicense server --accept -N slogin1-ib.superepod.company.com
#/usr/lpp/mmfs/bin/mmchnode --nonquorum --nomanager -N
dgx01-ib.superpod.company.com
#/usr/lpp/mmfs/bin/mmchnode --quorum --manager -N slogin01-ib.superpod.company.com
#/usr/lpp/mmfs/bin/mmchnodeclass login add -N slogin1-ib.superepod.company.com
#/usr/lpp/mmfs/bin/mmstartup -N slogin1-ib.superepod.company.com
#/usr/lpp/mmfs/bin/mmmount /ess/gpfs0 -N slogin1-ib.superepod.company.com
```

The DGX cluster is now ready to run non-containerized applications from any of the DGX H100 nodes with I/O to the IBM Storage Scale file system `/ess/gpfs0`, which is remotely mounted from the IBM Storage Scale storage cluster.

Alternatively, the DGX cluster can support containerized applications by using Kubernetes with run:ai, with the IBM Storage Scale CSI deployed and configured.

3.4.5 DGX testing and confirmation

Running I/O sequential throughput tests from the IBM Storage Scale client DGX H100 nodes to the IBM Storage Scale file system that is mounted from the IBM Storage Scale cluster is useful to establish the baseline throughput performance and validate the installed configuration for future reference.

The published Interleaved or Random (IOR) benchmark for one IBM Storage Scale 6000 performance building block is up to 310GBps sequential read, and up to 155GBps sequential write.

Example 3-41 on page 75 shows an example of running the IOR benchmark.

Example 3-41 Example of running the IOR benchmark

```
#Commands from the shell script:
================================================================
WORKDIR=/ess/gpfs0/ibm/rundir

OUTDIR=/ess/gpfs0/ibm/rundir/output

IOR=/ess/gpfs0/ibm/ior/bin/ior

NODELIST=/ess/gpfs0/ibm/rundir/dgx_nodes.txt
IORDIR=/ess/gpfs0/ibm/iordir/dir$$ ; mkdir -p $IORDIR
i=256

time -p mpirun -np $i --map-by node --machinefile $NODELIST $IOR -a POSIX -b 20G
-t 8M -o ${IORDIR}/iorfile  -w -r -C -F -g -i 3 -d 30 -e >>
$OUTDIR/ior_x8M_32n_${i}p_$(date +%Y%m%d%H%M%S).txt

================================================================
#The mpirun host machinefile /ess/gpfs0/ibm/rundir/dgx_nodes.txt
================================================================
dgx01
dgx02
dgx03
dgx04
dgx05
dgx06
dgx07
dgx08
dgx09
dgx10
dgx11
dgx12
dgx13
dgx14
dgx15
dgx16
dgx17
dgx18
dgx19
dgx20
dgx21
dgx22
dgx23
dgx24
dgx25
dgx26
dgx27
dgx28
dgx29
dgx30
dgx31
dgx32
================================================================
```

3.5 IBM Storage Scale Container Interface Driver

The CSI is a standard for exposing block and file storage systems to containerized workloads on Container Orchestration Systems like Kubernetes. The IBM Storage Scale Container Interface Driver enables an IBM Storage Scale file system to be used by applications running in a Kubernetes (k8S) cluster and as the input data source for a newly configured DGX configuration.

The following sections describe the deployment of the IBM Storage Scale CSI on the NVIDIA DGX SuperPOD cluster running Kubernetes (k8S) and run:ai, which is the software platform that optimizes artificial intelligence (AI) workloads.

3.5.1 CSI driver deployment on a Kubernetes cluster

The IBM Storage Scale CSI driver requires the IBM Storage Scale Management REST API to enable communication between the IBM Storage Scale client cluster on the NVIDIA DGX SuperPOD and the IBM Storage Scale storage cluster on the IBM Storage Scale storage solution. The IBM Storage Scale management REST API (one of the services that are provided by the IBM Storage Scale GUI) must be always running and in a healthy state for the IBM Storage Scale CSI driver to function.

Configuring high availability for the IBM Storage Scale REST API

Multiple GUI services (REST API) must be set up and running on the IBM Storage Scale cluster to help ensure high availability (HA) of the GUI service for the IBM Storage Scale CSI driver. Up to three GUI nodes can be configured in an active/active configuration. The performance collector running on each of the GUI servers is configured as the multi-collector federated configuration so that each of the GUI servers has the updated performance data in the IBM Storage Scale cluster.

Figure 3-3 shows a HA configuration of the IBM Storage Scale GUI with two GUI nodes.

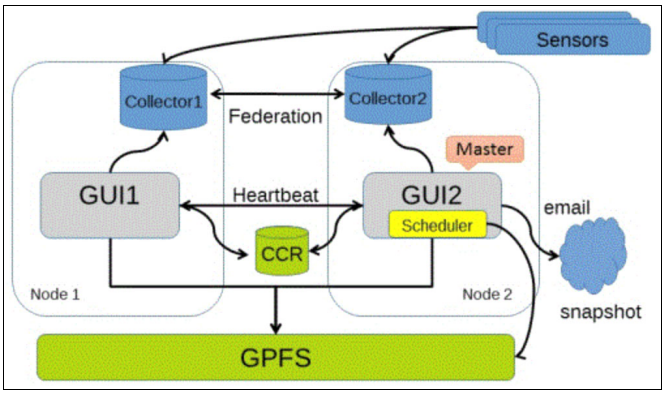


Figure 3-3 High availability configuration of IBM Storage Scale GUI with two GUI nodes

The IBM Storage Scale CSI driver should be installed and deployed after the Kubernetes (k8S) and run:ai installations are complete on the NVIDIA DGX SuperPOD cluster. The IBM Storage Scale client cluster on the NVIDIA DGX SuperPOD cluster initial deployment is designed to run non-containerized workloads with the SLURM scheduler. It is modified to support containerized workloads on Kubernetes with the IBM Storage Scale CSI driver.

To configure the IBM Storage Scale cluster to support containers, complete the following steps:

1. Add the two run:ai nodes to the DGX IBM Storage Scale client cluster as quorum and manager nodes.
2. Install and configure the performance collectors on the two run:ai nodes in a multi-collector federated configuration.
3. Remove the DGX login nodes (the k8S masters) from the DGX IBM Storage Scale client cluster. Change the node type of at least one DGX nodes to quorum to maintain at least three quorum nodes for the DGX IBM Storage Scale client cluster.
4. Create the primaryFS file system on the IBM Storage Scale storage cluster that will be remotely mounted on the DGX IBM Storage Scale client cluster. The primaryFS file system will be configured with the file system tuning that is required by IBM Storage Scale CSI.
5. Install the external snapshot utility's customer resource definitions (CRDs) and snapshot controller on to the k8S and DGX cluster if they are not already installed.
6. Proceed with the IBM Storage Scale CSI installation and configuration on the DGX H100 nodes (the k8S workers) by following the instructions at Air gap installation.

3.5.2 Adding the run:ai nodes to the DGX IBM Storage Scale client cluster

The NVIDIA DGX k8S cluster is deployed with the login nodes as the k8S control nodes. The DGX nodes and the run:ai nodes are k8S workers.

This scenario outlines the steps to install and configure the IBM Storage Scale CSI 2.11 for the NVIDIA DGX SuperPOD + IBM Storage Scale solution. The same steps apply to more recent releases. For more information about installation planning, see Table 1. CSI Features, OCP, Kubernetes, and IBM Storage Scale Compatibility Matrix.

The BCM image on the NVIDIA DGX SuperPOD for the run:ai nodes can be cloned from the image of the DGX login node and then modified to install the IBM Storage Scale performance monitoring and GUI software.

To add IBM Storage Scale software packages from the extracted tree to the BCM images, complete the following steps:

1. For IBM Storage Scale performance monitoring, add the following packages:

   ```
   /usr/lpp/mmfs/5.2.1.0/zimon_debs/ubuntu/ubuntu22/gpfs.gss.pmcollector_5.2.1-0.U
   22.04_a
   md64.deb
   /usr/lpp/mmfs/5.2.1.0/zimon_debs/ubuntu/ubuntu22/gpfs.gss.pmsensors_5.2.1-0.U22
   .04_am d64.deb
   /usr/lpp/mmfs/5.2.1.0/zimon_debs/ubuntu  /gpfs.pm-ganesha_10.0.0-2_amd64.deb
   ```

2. For the IBM Storage Scale GUI, add the following packages:

   ```
   /usr/lpp/mmfs/5.2.1.0/gpfs_debs/gpfs.gui_5.2.1-0_all.deb/usr/lpp/mmfs/5.2.1.0/gp
   fs_debs/gpfs.java_5.2.1-0_amd64.deb
   ```

3. To add the prerequisite operating system packages for the IBM Storage Scale GUI, use the following command:

   ```
   postgresql
   ```

The k8S cluster is deployed with the login nodes as the k8S control nodes. The DGX nodes and the run:ai nodes are the k8S workers. The run:ai nodes are added to the DGX IBM Storage Scale client cluster as the quorum nodes and to the IBM Storage Scale GUI server.

3.5.3 Preparing the run:ai nodes for IBM Storage Scale installation

Update the `/etc/hosts` files on the DGX IBM Storage Scale Client clusters and the IBM Storage Scale storage clusters with the entries for the run:ai nodes to ensure that the IPoIB address (which is the GPFS demon network) can be resolved.

Ensure that the IPoIB interface of the run:ai nodes are active and set up as passwordless SSH for root among the run:ai nodes and DGX H100 nodes.

On one of the DGX H100 nodes (the k8S worker) with passwordless SSH for root where GPFS is active, complete the following steps:

1. Change the quorum nodes on the DGX IBM Storage Scale client cluster from the DGX login nodes (the k8S masters) to the DGX H100 nodes (the k8S workers), as shown in Example 3-42.

Example 3-42 Changing the quorum nodes on the DGX IBM Storage Scale client cluster

```
mmchnode —-quorum —-manager -N dgx01-ib.superpod.company.com,
dgx06-ib.superpod.company.com, dgx12-ib.superpod.company.com
mmchnode —-nonquorum —-nomanager -N slogin01-ib.superpod.company.com,
slogin02-ib.superpod.company.com, slogin03-ib.superpod.company.com
```

2. Add the run:ai nodes as quorum nodes to the DGX IBM Storage Scale client cluster, as shown in Example 3-43.

Example 3-43 Adding the run:ai nodes as quorum nodes to the DGX IBM Storage Scale client cluster

```
mmaddnode -N runai01-ib,runai02-ib
mmchnode —-quorum --manager -N runai01-ib, runai02-ib
mmchlicense server --accept -N runai01-ib, runai02-ib
mmstartup -N runai01-ib,runai02-ib
mmcrnodeclass runai -N runai01-ib,runai02-ib
```

3. Remove the DGX login nodes (the k8S master) from the DGX IBM Storage Scale client cluster and configure the multi-collector federated performance monitoring, as shown in Example 3-44.

Example 3-44 Removing the DGX login nodes from the DGX IBM Storage Scale client cluster

```
mmdelnode -N slogin01-ib,slogin02-ib, slogin03-ib
mmchnode —-noperfmon -N all
mmperfmon config generate —collectors runai01-ib, runai02-ib
mmchnode --perfmon -N all
mmch -N all systemctl enable pmsensors
mmdsh -N all systemctl restart pmsensors
mmdsh -N runai systemctl enable pmcollector
mmdsh -N runai systemctl start pmcollector
```

4. Start the GUI services on the nodeclass of the run:ai node to enable the IBM Storage Scale REST API services, as shown in Example 3-45.

Example 3-45 Starting the GUI services on the nodeclass of the run:ai node

```
mmdsh -N runai systemctl enable gpfsgui
mmdsh -N runai systemctl start gpfsgui
```

5. Initialize the GUI database on each of the run:ai nodes by logging in to the GUI console or running the command `/usr/lpp/mmfs/gui/cli/initgui`. The run:ai nodes are added to the `GUI_MGMT_SERVERS` node class.
6. Set and activate the IBM Storage Scale specific tuning for the IBM Storage Scale GUI service on the run:ainodes, as shown in Example 3-46.

Example 3-46 Setting and activating the IBM Storage Scale specific tuning

```
mmchconfig verbsPorts=" mlx5_0/1 mlx5_8/1" -N runai
mmchconfig pagePool=16G -N runai
maxStatCache=128K -N runai
mmshutdown -N runai
mmstartup -N runai
```

3.5.4 Deploying the IBM Storage Scale CSI 2.11.1

The operator manifest and the customer resource file for the IBM Storage Scale CSI must be downloaded and customized with site-specific details. For CSI 2.11.1, download the operator manifest yaml file and the custom resource yaml file.

When deploying an air-gapped installation for the IBM Storage Scale CSI installation, see Air gap installation, which outlines the files and changes that must be made.

For the IBM CSI driver 2.11.1, you can download IBM Storage Scale CSI installation images that are listed in Table 1. Image Links for IBM Storage Scale Container Storage Interface driver 2.11.1.

Download the IBM Storage Scale CSI installation images and upload them into the private registry, as shown in Table 3-2.

Table 3-2 CSI image private registry details

Name	Version	Private registry
ibm-spectrum-scale-csi-operator	2.11.1	internal.registry.company.com/demo-csi211/ibm-spectrum-scale-csi-operator:2.11.1
ibm-spectrum-scale-csi-driver	2.11.1	internal.registry.company.com/demo-csi211/ibm-spectrum-scale-csi-driver:2.11.1
csi-node-driver-registrar	2.10.0	internal.registry.company.com/demo-csi211/csi-node-driver-registrar:2.10.0
livenessprobe	2.12.9	internal.registry.company.com/demo-csi211/livenessprobe:2.12.0
csi-attacher	4.5.0	internal.registry.company.com/demo-csi211/csi-attacher:4.5.0
csi-provisioner	4.0.0	internal.registry.company.com/demo-csi211/csi-provisioner:4.0.0
csi-snapshotter	7.0.1	internal.registry.company.com/demo-csi211/csi-shapshotter:7.0.1
csi-resizer	1.10.0	prodoce.registry.company.com/demo-csi211/csi-resizer:1.10.0

In this example, you must update the following site-specific details in the operator manifest and the custom resource files:

- The primaryFS from the IBM Storage Scale cluster is `csi-ess-gpfs0`.
- The IBM Storage Scale GUI admin secrets file for the DGX IBM Storage Scale client cluster is `csi-dgx-secret`.
- The IBM Storage Scale GUI admin secrets file for the IBM Storage Scale storage cluster is `csi-ess-secret`.
- The IBM Storage Scale GUI port was configured on the non-default port of 22443 to avoid conflict with the 443 port of the run:ai web console.

Note: For the IP address or hostname of the GUI node of the primary cluster (the DGX IBM Storage Scale client cluster) and the remote cluster (the IBM Storage Scale cluster), the Ethernet or OS management interface is used instead of the IPoIB (the GPFS daemon network) because of incompatibility of the IPoIB network with the k8S network.

Example 3-47 shows the IP addresses or hostnames for the DGX IBM Storage Scale client cluster.

Example 3-47 IP addresses or hostnames for the DGX IBM Storage Scale client cluster

```
restApi:
        - guiHost: "10.70.10.84"
          guiPort: 22443
        - guiHost: "10.70.10.85"
          guiPort: 22443
```

Example 3-48 shows the IP addresses or hostnames for the IBM Storage Scale storage cluster.

Example 3-48 IP addresses or hostnames for the IBM Storage Scale storage cluster

```
restApi:
        - guiHost: "10.70.10.153"
          guiPort: 443
```

For the `nodeMapping` between the k8SNode and the `spectrumscaleNode` (the GPFS daemon interface) for this example, we included a few DGX nodes, as shown in Example 3-49.

Example 3-49 nodeMapping between spectrumscaleNode

```
nodeMapping:
k8SNode: "dgx01"
spectrumscaleNode: "dgx01-ib.superpod.company.com"
k8SNode: "dgx02"
spectrumscaleNode: "dgx02-ib.superpod.company.com"
k8SNode: "dgx03"
spectrumscaleNode: "dgx03-ib.superpod.company.com"
```

Example 3-50 shows the key changes for an air-gapped installation for the `ibm-spectrum-scale-csi-operator.yaml` file.

Example 3-50 The ibm-spectrum-scale-csi-operator.yaml file

```
- name: CSI_DRIVER_IMAGE
          value:
prodoce.registry.company.com/demo-csi211/ibm-spectrum-scale-csi-driver:2.11.1
        - name: CSI_SNAPSHOTTER_IMAGE
          value: prodoce.registry.company.com/demo-csi211/csi-shapshotter:7.0.1
        - name: CSI_ATTACHER_IMAGE
          value: prodoce.registry.company.com/demo-csi211/csi-attacher:4.5.0
        - name: CSI_PROVISIONER_IMAGE
          value: prodoce.registry.company.com/demo-csi211/csi-provisioner:4.0.0
        - name: CSI_LIVENESSPROBE_IMAGE
          value: prodoce.registry.company.com/demo-csi211/livenessprobe:2.12.0
        - name: CSI_NODE_REGISTRAR_IMAGE
          value:
prodoce.registry.company.com/demo-csi211/csi-node-driver-registrar:2.10.0
        - name: CSI_RESIZER_IMAGE
          value: prodoce.registry.company.com/demo-csi211/csi-resizer:1.10.0
```

Example 3-51 shows the `csiscaleopertors.csi.ibm.com.cr.yaml` file with site-specific updates.

Example 3-51 The csiscaleopertors.csi.ibm.com.cr.yaml file

```
---
apiVersion: csi.ibm.com/v1
kind: "CSIIBM Storage ScaleOperator"
metadata:
  name: "ibm-spectrum-scale-csi"
  namespace: "ibm-spectrum-scale-csi-driver"
  labels:
    app.kubernetes.io/name: ibm-spectrum-scale-csi-operator
    app.kubernetes.io/instance: ibm-spectrum-scale-csi-operator
    app.kubernetes.io/managed-by: ibm-spectrum-scale-csi-operator
    release: ibm-spectrum-scale-csi-operator
status: {}
spec:

# A pass-through option that distributes an imagePullSecrets array to the
containers
# generated by the CSI scale operator. Refer to official k8S documentation for
# your environment for more details.
https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-regist
ry/
#
#################################################################################
#  imagePullSecrets:
#    - APullSecret
#    - AnotherOptional

# Below specifies the details of an IBM Storage Scale cluster configuration that
is used by the
# plug-in. It can have multiple values.
```

```
#
#################################################################################
#
  clusters:
    - id: "11691967945767875033"
      secrets: "csi-dgx-secret"
      secureSslMode: false
      primary:
        primaryFs: "csi-ess-gpfs0"
#        primaryFset: "< Fileset in Primary Filesystem >"   # Optional -
default:spectrum-scale-csi-volume-store
#        inodeLimit: "< inode limit for Primary Fileset >" # Optional
        remoteCluster: "5588634210597745484"
#        remoteCluster: "< Remote ClusterID >"            # Optional - This is
only required if primaryFs is a remote cluster's file system and this ID should
have a separate entry in Clusters map too.
#      cacert:  "< Name of CA cert configmap for GUI >"   # Optional
      restApi:
        - guiHost: "10.70.10.84"
          guiPort: 22443
        - guiHost: "10.70.10.85"
          guiPort: 22443
#       - guiHost: "< IP/Hostname of another GUI node of primary cluster >"
#Optional - Multiple GUI IPs/Hostnames can be provided like this, if the storage
cluster has GUI installed on multiple nodes.
#
# In the case we have multiple clusters, specify their configurations.
#
#################################################################################
#    - id: "< Cluster ID >"
#      secrets: "< Secret for Cluster >"
#      secureSslMode: false
#      restApi:
#      - guiHost: "< IP/Hostname of a GUI node of the cluster >"
#      - guiHost: "< IP/Hostname of another GUI node of the cluster >" #Optinal -
Multiple GUI IPs/Hostnames can be provided like this, if the storage cluster has
GUI installed on multiple nodes.
#      cacert:  "< Name of CA cert configmap for GUI >"   # Optional
    - id: "5588634210597745484"
      secrets: "csi-ess-secret"
      secureSslMode: false
      restApi:
        - guiHost: "10.70.10.153"
          guiPort: 443

# attacherNodeSelector specifies on which nodes we want to run attacher sidecar
# In the below example attacher will run on nodes that have label as "scale=true"
# and "infranode=2". Can have multiple entries.
#
#################################################################################
  attacherNodeSelector:
    - key: "scale"
      value: "true"
#    - key: "infranode"
#      value: "2"
```

```
# provisionerNodeSelector specifies on which nodes we want to run provisioner
# sidecar. In the below example provisioner will run on nodes that have label as
# "scale=true" and "infranode=1". Can have multiple entries.
#
================================================================================
  provisionerNodeSelector:
    - key: "scale"
      value: "true"
#    - key: "infranode"
#      value: "1"

# snapshotterNodeSelector specifies on which nodes we want to run snapshotter
# sidecar. In the below example snapshotter pod will run on nodes that have label
as
# "scale=true" and "infranode=1". Can have multiple entries.
#
================================================================================
  snapshotterNodeSelector:
    - key: "scale"
      value: "true"
#    - key: "infranode"
#      value: "1"

# resizerNodeSelector specifies on which nodes we want to run resizer
# sidecar. In the below example resizer pod will run on nodes that have label as
# "scale=true" and "infranode=1". Can have multiple entries.
#
================================================================================
  resizerNodeSelector:
    - key: "scale"
      value: "true"
#    - key: "infranode"
#      value: "1"

# pluginNodeSelector specifies the nodes on which we want to run plug-in daemoset
# In the below example plug-in daemonset will run on nodes that have label as
# "scale=true". Can have multiple entries.
#
================================================================================
  pluginNodeSelector:
    - key: "scale"
      value: "true"

# In case k8S nodes name differs from IBM Storage Scale nodes name, we can provide
# node mapping by using nodeMapping attribute. Can have multiple entries.
#
================================================================================
#  nodeMapping:
#    - k8SNode: "< k8S Node Name >"
#      spectrumscaleNode: "< IBM Storage Scale Node Name >"
# In case k8S node name start with number then use following node mapping format.
#    - k8SNode: "k8SNodePrefix_< k8S Node Name >"
#      spectrumscaleNode: "< IBM Storage Scale Node Name >"
  nodeMapping:
```

```
- k8SNode: "dgx01"
  spectrumscaleNode: "dgx01-ib.superpod.company.com"
- k8SNode: "dgx02"
  spectrumscaleNode: "dgx02-ib.superpod.company.com"
- k8SNode: "dgx03"
  spectrumscaleNode: "dgx03-ib.superpod.company.com"
- k8SNode: "dgx04"
  spectrumscaleNode: "dgx04-ib.superpod.company.com"
- k8SNode: "dgx05"
  spectrumscaleNode: "dgx05-ib.superpod.company.com"
- k8SNode: "dgx06"
  spectrumscaleNode: "dgx06-ib.superpod.company.com"
- k8SNode: "dgx07"
  spectrumscaleNode: "dgx07-ib.superpod.company.com"
- k8SNode: "dgx08"
  spectrumscaleNode: "dgx08-ib.superpod.company.com"
- k8SNode: "dgx09"
  spectrumscaleNode: "dgx09-ib.superpod.company.com"
- k8SNode: "dgx10"
  spectrumscaleNode: "dgx10-ib.superpod.company.com"
- k8SNode: "dgx11"
  spectrumscaleNode: "dgx11-ib.superpod.company.com"
- k8SNode: "dgx12"
  spectrumscaleNode: "dgx12-ib.superpod.company.com"
- k8SNode: "dgx13"
  spectrumscaleNode: "dgx13-ib.superpod.company.com"
- k8SNode: "dgx14"
  spectrumscaleNode: "dgx14-ib.superpod.company.com"
- k8SNode: "dgx15"
  spectrumscaleNode: "dgx15-ib.superpod.company.com"
- k8SNode: "dgx16"
  spectrumscaleNode: "dgx16-ib.superpod.company.com"
- k8SNode: "dgx17"
  spectrumscaleNode: "dgx17-ib.superpod.company.com"
- k8SNode: "dgx18"
  spectrumscaleNode: "dgx18-ib.superpod.company.com"
- k8SNode: "dgx19"
  spectrumscaleNode: "dgx19-ib.superpod.company.com"
- k8SNode: "dgx20"
  spectrumscaleNode: "dgx20-ib.superpod.company.com"
- k8SNode: "dgx21"
  spectrumscaleNode: "dgx21-ib.superpod.company.com"
- k8SNode: "dgx22"
  spectrumscaleNode: "dgx22-ib.superpod.company.com"
- k8SNode: "dgx23"
  spectrumscaleNode: "dgx23-ib.superpod.company.com"
- k8SNode: "dgx24"
  spectrumscaleNode: "dgx24-ib.superpod.company.com"
- k8SNode: "dgx25"
  spectrumscaleNode: "dgx25-ib.superpod.company.com"
- k8SNode: "dgx26"
  spectrumscaleNode: "dgx26-ib.superpod.company.com"
- k8SNode: "dgx27"
  spectrumscaleNode: "dgx27-ib.superpod.company.com"
- k8SNode: "dgx28"
```

```
      spectrumscaleNode: "dgx28-ib.superpod.company.com"
    - k8SNode: "dgx29"
      spectrumscaleNode: "dgx29-ib.superpod.company.com"
    - k8SNode: "dgx30"
      spectrumscaleNode: "dgx30-ib.superpod.company.com"
    - k8SNode: "dgx31"
      spectrumscaleNode: "dgx31-ib.superpod.company.com"
    - k8SNode: "dgx32"
      spectrumscaleNode: "dgx32-ib.superpod.company.com"

# Array of tolerations that will be distributed to CSI pods. Refer to official
# k8S documentation for your environment for more details.
# https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/
#
#============================================================================
#  tolerations:
#    - key: "key1"
#      operator: "Equal"
#      value: "value1"
#      effect: "NoExecute"
#      tolerationSeconds: 3600

# Kubelet root directory path, in case we don't want to use the default kubelet
# root directory path.
#
#============================================================================
#  kubeletRootDirPath: "/var/lib/kubelet"
```

A storage class with the IBM Storage Scale CSI driver on the remotely mounted file system `csi-ess-spod` from the IBM Storage Scale storage cluster is defined, and the file sets on the `csi-ess-spod` are created to define dynamic persistent volumes (PVs) and persistent volume claims (PVCs) from the storage class.

To use existing data in file sets on the `csi-ess-spod` file systems for the containerized workload on k8S, you must create static PVs and PVCs. Use the run:ai web console to create the dynamic PVs and PVCs from the storage class. You can also use this console to consume the static PVs and PVCs in the run:ai projects.

Example 3-52 is an example YAML file that creates the independent file set storage class on the file system `csi-ess-spod`.

Example 3-52 Example YAML file

```
apiVersion: storage.k8S.io/v1
kind: StorageClass
metadata:
  name: ibm-spectrum-scale-csi-fileset
provisioner: spectrumscale.csi.ibm.com
parameters:
  volBackendFs: csi-ess-gpfs0
reclaimPolicy: Delete
```

References: For more information about the topics in this section, see the following resources:

- IBM Storage Scale Container Storage Driver introduction
- Ensuring high availability of the IBM Storage Scale GUI service
- Configuring the IBM Storage Scale performance collector
- Configuring the IBM Storage Scale multi-collector federated configuration
- Configuring multiple collectors
- Using the IBM Storage Scale Container Storage Interface driver

3.6 Active File Management

The AFM nodes are the gateway for caching data from data sources that are external to the IBM Storage Scale cluster and intended to be used by the applications running on the NVIDIA DGX SuperPOD. Therefore, the AFM nodes may be in a different Ethernet subnet for access to the external data sources and provisioned with a different OS than the CES nodes in the IBM Storage Scale storage cluster.

To add the AFM nodes to an IBM Storage Scale storage cluster that is defined on the IBM Storage Scale 6000 with the admin network and the daemon network on the InfiniBand network, the InfiniBand network on the AFM nodes must have routes to the corresponding network on the IBM Storage Scale storage cluster.

In the following example, the AMF nodes use an RHEL 9.4 kernel that supports IBM Storage Scale 5.2.1-0.

Install the following software:

- Supported OS: RHEL 9.4
- IBM Storage Scale 5.2.1-0
- OFED for the NVIDIA ConnectX-6 InfiniBand adapters: `MLNX_OFED_LINUX-23.10-3.2.2.0-rhel9.4-x86_64.iso`

The AFM nodes of the IBM Storage Scale storage cluster can be deployed by using manual installation methods that are similar to the steps in 3.3, “Configuring Cluster Export Services” on page 53.

3.6.1 Configuring the AFM gateway node

To configure the AFM gateway node, complete the following steps:

- On the AFM node:
  a. Install the operating system prerequisites for IBM Storage Scale, including the additional software `nfs-utils` that is required for AFM.
  b. Install the IBM Storage Scale software package.
  c. Build the GPFS portability layer.
  d. Ensure that the `/etc/hosts` file in all the nodes in the IBM Storage Scale storage cluster can resolve the hostnames for the AFM nodes (both the Ethernet OS management and AFM outbound and the IPoIB IBM Storage Scale daemon network interfaces).

e. Set up passwordless SSH for root across all the nodes in the IBM Storage Scale storage cluster and the AFM nodes.

- On the EMS server:

  a. Add the AFM node to the IBM Storage Scale storage cluster, create the nodeclass `afm`, and start GPFS, as shown in Example 3-53.

Example 3-53 Adding the AFM node

```
mmaddnode -N afm01-ib, afm02-ib, afm03-ib, afm04-ib
mmchlicense server --accept -N afm01-ib, afm02-ib, afm03-ib, afm04-ib
mmcrnodeclass afm -N afm01-ib, afm02-ib, afm03-ib, afm04-ib
mmstartup -N afm
mmchnode –gateway –perfmon -N afm
```

  b. Configure performance monitoring on the AFM gateway nodes (the nodeclass `afm`), as shown in Example 3-54.

Example 3-54 Configuring performance monitoring on the AFM gateway nodes

```
mmperfmon config update GPFSAFM.period=10
mmperfmon config update GPFSAFM.restrict='afm'
mmperfmon config update GPFSAFMFS.period=10
mmperfmon config update GPFSAFMFS.restrict='afm'
mmperfmon config update GPFSAFMFSET.period=10
mmperfmon config update
GPFSAFMFSET.restrict='afm' mmdsh -N afm systemctl
restart pmsensors
mmchconfig maxFilesToCache=10000 -N afm
mmchconfig maxStatCache=20000 -N afm
mmshutdown -N afm
mmstartup -N afm
```

If the AFM cluster is the gateway for NFS 4.1 for exporting file systems, set the configuration parameters that are shown in Example 3-55.

Example 3-55 Setting the configuration parameters

```
mmchconfig afmNFSVersion=4.1 -i
mmchconfig afmSyncNFSv4ACL=1 -i
```

The AFM gateway is now ready.

3.6.2 Creating the AFM relationship

In this example, the read-only AFM target is the file set `inbound` on the IBM Storage Scale file system `ess-gpfs0`. The file set is created from the NFS data source, which is the file system `/projects/inbound` that is exported from the NFS server at 10.20.201.74. The AFM gateway Ethernet IP addresses (10.20.100.*/24) must be routable to the NFS data source.

Example 3-56 shows the creation of the `inbound` file set.

Example 3-56 Creating the inbound file set

```
mmcrfileset ess-gpfs0 inbound -p afmtarget=10.20.201.74:/projects/data -p
afmmode=read-only –inode-space=new
mmlinkfileset ess-gpfs0 -J /gpfs/ess-gpfs0/inbound
```

Check the state of the cache file set to confirm the configuration, as shown in Example 3-57.

Example 3-57 Checking the state of the cache file set

```
[root@afm03 ~]# /usr/lpp/mmfs/bin/mmafmctl ess-gpfs0 getstate

Fileset Name   Fileset Target
Cache State       Gateway Node   Queue Length   Queue numExec

inbound        nfs://10.20.201.74//projects/inbound
Dirty             afm03-ib       4              58724429
```

Additional AFM tuning, which includes parallel data transfers that use the mapping of the NFS target to AFM gateway nodes and use multiple remount mounts, can be configured to improve AFM performance.

Reference: For more information about the topics that were described in this section, see the following resources:

- AFM operating system prerequisites
- List to access Active File Management (AFM) documentation
- Example of creating AFM relationship by using the NFS protocol

3.7 Monitoring

Monitoring the ESS system includes system health, performance, and capacity monitoring. You can monitor the system either through the ESS GUI or by using a CLI.

Monitoring system health

The ESS system provides background monitoring capabilities to check the health of a cluster and each node of the cluster, including all the services that are hosted on a node. You can view the system health states or corresponding events for the selected health state on the individual pages, widgets, or windows of the ESS GUI. You can also view system health details by issuing the `mmhealth` command with options like `mmhealth cluster show`, `mmhealth node show`, or similar options.

Monitoring capacity

You can monitor the capacity of the file system, pools, file sets, NSDs, users, and user groups in the IBM Storage Scale system.

The capacity details in the GUI are obtained from the following sources:

- GPFS quota database: The system collects the quota details and stores it in the PostgreSQL database.
- The performance monitoring tool collects the capacity data. The GUI queries the performance monitoring tool and displays capacity data in the GUI.

For both GPFS quota database and performance monitoring tool-based capacity and quota collection, use Files Quotas to enable quota data collection per file system and enforce quota limit checking. If this tool is not enabled for a file system, then the following limitations exist:

- No capacity and inode data is collected for users, groups, and file sets.
- Quota limits for users, groups, and file.sets cannot be defined.
- No alerts are sent and the data writes are not restricted.

Performance monitoring

The performance monitoring tool helps you to view the metrics that are associated with GPFS and the associated protocols; get a graphical representation of the status and trends of the key performance indicators; and analyze ESS performance problems. You can use both the CLI and the GUI to monitor the system performance based on various aspects.

The following options are available to monitor system performance:

- The `mmperfmon` command helps fetch system performance details that are collected through the performance monitoring tools.
- The `mmpmon` command helps monitor the GPFS performance on the node in which it is run, and other specified nodes.
- The ESS GUI.
- Grafana is an open-source tool that you can use to monitor performance details that are collected through the performance monitoring tools.

3.7.1 Prometheus or Grafana integration

Grafana is an open-source tool for visualizing time series and application metrics. It provides a powerful platform to create, explore, and share dashboards and data.

The IBM Storage Scale bridge for Grafana can be used for exploring IBM Storage Scale performance data on Grafana dashboards. Grafana Bridge is a stand-alone Python application. It converts the IBM Storage Scale meta-data and performance data that is collected by the IBM Storage Scale performance monitoring tool (ZiMon) into the query requests that are acceptable by the Grafana integrated openTSDB plug-in.

Figure 3-4 shows an example of Grafana bridges in an IBM Storage Scale cluster.

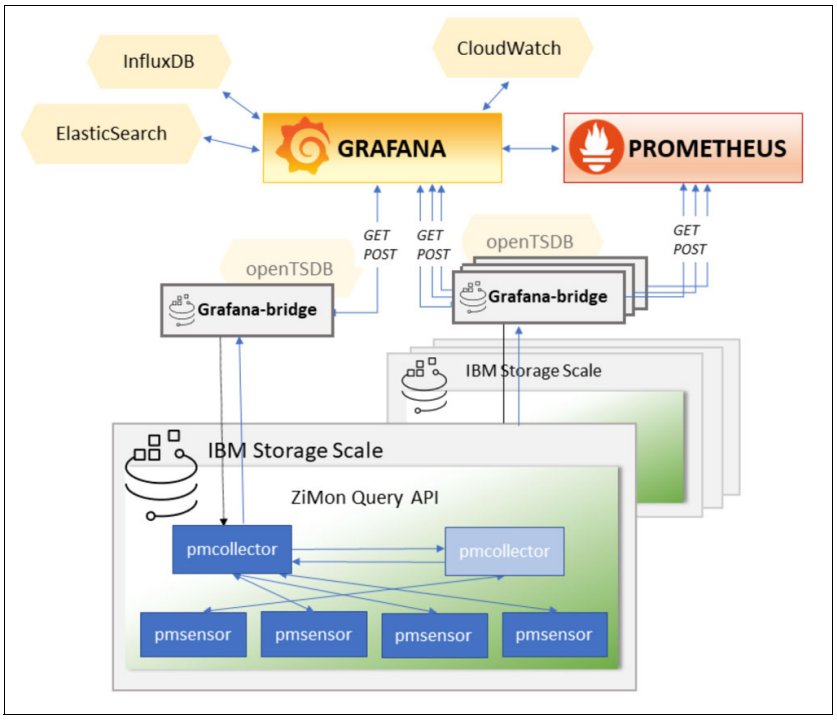


Figure 3-4 Example of Grafana bridges in an IBM Storage Scale cluster

Prometheus is an open-source tool that you can use to monitor performance details that are collected through the performance monitoring tools. Prometheus exporter for IBM Storage Scale is a software component that exposes IBM Storage Scale performance metrics in a format that the Prometheus timeseries database can scrape.

The IBM Storage Scale bridge for Grafana 8.x enables multiple APIs. The communication port must be specified in the `config.ini` file for each plug-in to be activated. The Prometheus Exporter plug-in is integrated in the IBM Storage Scale bridge for Grafana 8.0.0, as shown in Figure 3-4. The Prometheus Exporter API is supported only in classic IBM Storage Scale.

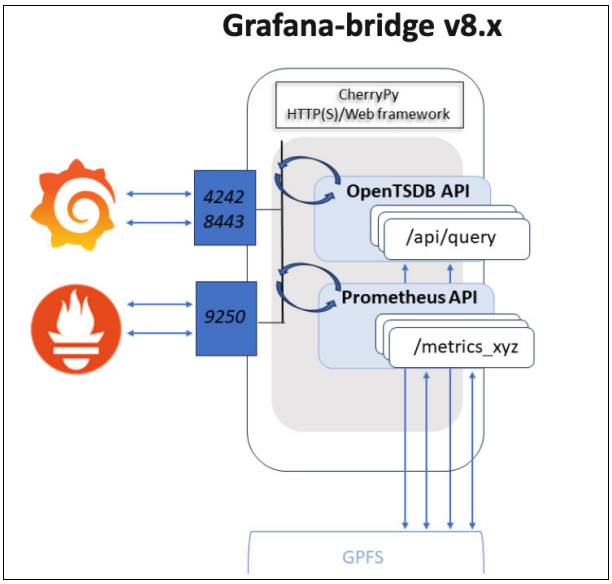


Figure 3-5

For more information, see IBM Storage Scale Bridge for Grafana.

4

Server tuning

This chapter provides an overview of tuning and configuration considerations for an IBM Storage Scale 6000 and NVIDIA DGX SuperPOD deployment. An initial starting point set of best practices for NVIDIA DGX SuperPOD clients and IBM Storage Scale 6000 storage servers are provided to optimize the performance and stability of the deployment. Section 3.3, “Configuring Cluster Export Services” on page 53 has more tuning recommendations regarding tuning Cluster Export Service nodes.

This chapter describes the following topics:

- 4.1, “General software level considerations around tuning” on page 94
- 4.2, “Starting point for IBM Storage Scale tuning best practice for clients” on page 94
- 4.3, “IBM Storage Scale tuning best practices for IBM Storage Scale 6000 servers” on page 100
- 4.4, “Storage network configuration and validation” on page 101
- 4.5, “IBM Storage Scale 6000 and client cluster configuration performance considerations” on page 112

4.1 General software level considerations around tuning

As a best practice, apply regular code updates to the IBM Storage Scale clients and servers to help ensure that they are at relatively recent code levels. This document does not assume that the latest code levels are applied, so there are references to tuning options that are not required on the latest code. All such references assume that the supported Storage Scale code levels are run at the time of writing. For more information about supported code levels, see IBM Storage Scale Frequently Asked Questions and Answers.

4.2 Starting point for IBM Storage Scale tuning best practice for clients

This section focuses on Network Shared Disk (NSD) client tuning by providing best practices and detailed tuning considerations, including guidance on further tuning. In an IBM Storage Scale 6000 SuperPOD deployment, client nodes that use the file system are called NSD clients, while IBM Storage Scale 6000 servers are NSD servers. For more information about NSD clients and servers, see 1.1.2, “IBM Storage Scale fundamental components” on page 4.

4.2.1 Starting point for tuning mmchconfig parameters

This section outlines the initial IBM Storage Scale tuning recommendations for client nodes in an IBM Storage Scale 6000 deployment. It begins with a list of `mmchconfig` options, followed by a description of key considerations for each setting. Although it is a best practice that the clients run the latest code, there are references to tuning parameters in the earlier 5.1.X code, which is supported at the time of writing.

Example 4-1 shows the starting point for the system settings.

Example 4-1 Starting point recommendations for tuning the mmchconfig parameters

```
idleSocketTimeout=0    # this is the default starting with the Scale core
(gpfs.base) 5.1.9 release
ignorePrefetchLUNCount=yes   # this is the default for clusters created with the
Scale core (gpfs.base) 5.2.2 release
maxblocksize=16M
maxMBpS=100000      # assumes client has two 25 GB/s links connected to the IBM
Storage Scale 6000 servers
maxFilesToCache=128K
maxStatCache=128K
numaMemoryInterleave=yes        # make sure numactl RPM is installed - this is the
default for clusters created with the Scale core (gpfs.base) 5.2.2 release
pagepool= <set to 25% of clients memory, up to 64GiB, even if the clients have
more than 256 GiB of memory>
prefetchLargeBlockThreshold=4M
proactiveReconnect=no        # this is the default for clusters created with the
Scale core (gpfs.base) 5.2.2 release
workerThreads=1024        # this is the default for clusters created with the
Scale core (gpfs.base) 5.2.2 release
```

Note: The symbol K is interpreted by the **mmchconfig** command as KiB, and M is interpreted as MiB.

Enabling RDMA is a best practice for optimal IBM Storage Scale performance. When RDMA is enabled, set the `verbsPorts`, `verbsRdma`, and `verbsRdmaSend` options should be set, as shown in Example 4-2.

Example 4-2 The verbsPorts, verbsRdma, and verbsRdmaSend options

```
verbsPorts=<list of RDMA-capable interfaces used to access the IBM Storage Scale
servers>
verbsRdma=enable
verbsRdmaSend=yes
```

If RDMA over Ethernet (RoCE) is enabled, you must also use the option that is shown in Example 4-3.

Example 4-3 The verbsRdmaCm option

```
verbsRdmaCm=enable
```

To enabe RDMA communication at scale, the network must be configured for RDMA. For InfiniBand systems, a minimal configuration is required beyond ensuring that a subnet manager (such as OpenSM or a hardware-based subnet manager) is running. For Ethernet-based RoCE networks, configuring Priority Flow Control (PFC), Explicit Congestion Notification (ECN), and lossless network settings is essential to achieving optimal RDMA performance. For more information, see *Highly Efficient Data Access with RoCE on IBM Elastic Storage Systems and IBM Spectrum Scale*, REDP-5658.

Configure the `verbsGPUDirectStorage` option if you plan to use GPUDirect Storage, as described in 1.3.3, “GPUDirect Storage” on page 22 and Configuring GPUDirect Storage for IBM Storage Scale. To enable this feature, set the option to `verbsGPUDirectStorage=yes`.

To set `verbsGPUDirectStorage`, you must shut down IBM Storage Scale on all nodes in the cluster. For more information, see 4.2.3, “Details about best practices for the mmchconfig tuning options” on page 96.

4.2.2 An example of client tuning by using mmchconfig

This example demonstrates how to use `mmchconfig` to configure a set of identically configured clients. If the clients have different configurations, they can be tuned with different settings by using the `-N` option of `mmchconfig` as described in the mmchconfig command.

In this example, each client node has at least 256 GB of memory that is installed, so the starting `pagepool` size should be 64 GB (25% of real memory). If each client has two RoCE cable links with `hca_id=mlx5_8` and `hca_id=mlx5_9`, and each adapter has a connection to a separate switch fabric that is used to access the IBM Storage Scale servers, then the correct `verbsPorts` setting is `mlx5_8/1/0 mlx5_9/1/1` (for more information about defining the `verbsPort` string, see 4.4, “Storage network configuration and validation” on page 101). If the clients are in a dedicated client-only cluster of only the compute clients, you do not need the `mmchconfig -N` option.

You can set all the `mmchconfig` parameters as shown in Example 4-4.

Example 4-4 Setting the mmchconfig parameters

```
mmshutdown -N All # If you're changing maxblocksize this requires that all nodes
be shutdown before running

mmchconfig idleSocketTimeout=0,ignorePrefetchLUNCount=yes,maxblocksize=16M,\
maxMBpS=60000,maxFilesToCache=128K,maxStatCache=128K,\
numaMemoryInterleave=yes,pagepool=64G,prefetchLargeBlockThreshold=4M,\
proactiveReconnect=no, verbsGPUDirectStorage=yes,\
verbsPorts="mlx5_8/1/0 mlx5_9/1/1",verbsRdmaCm=enable,\
verbsRdma=enable,verbsRdmaSend=yes,workerThreads=1024
```

These example settings should provide good performance for IBM Storage Scale clients in a RoCE-based deployment. For a deployment that uses InfiniBand RDMA, omit the `verbsRdmaCm=enable` option.

4.2.3 Details about best practices for the mmchconfig tuning options

This section provides more information about best practices for the `mmchconfig` tuning options for NVIDIA DGX SuperPOD deployment. For more information, see the mmchconfig command.

Here are the options and best practices for them:

- `idleSocketTimeout`: The `idleSocketTimeout` option controls how long IBM Storage Scale keeps idle TCP connections open before disconnecting them. This option is not officially documented in the IBM Storage Scale documentation. Before IBM Storage Scale 5.1.9.0, the default `idleSocketTimeout` was `3600`, which is defined in seconds, meaning that idle TCP connections were automatically closed after an hour of inactivity. However, starting in Version 5.1.9.0, the default value was changed to `0`, which disables automatic disconnection of idle TCP connections. Disabling idle socket disconnections avoids the potential impacts and problems that are associated with reconnecting sockets, leading to improved performance and stability. Keeping idle connections open is a best practice in most cases. However, in large-scale environments with thousands of clients, consider setting `idleSocketTimeout` to `3600` if there is a concern over the resources that are used by open TCP connections
- `ignorePrefetchLUNCount`: When `ignorePrefetchLUNCount` is set to `yes`, the number of prefetch and write-behind threads running concurrently on an IBM Storage Scale client depends on the number of LUNs that are assigned to the file system and the configured `maxMBpS` setting. When `ignorePrefetchLUNCount` is set to `no`, the system ignores the LUN count and dynamically determines the number of prefetch threads based solely on `maxMBpS`. For clusters that were created with IBM Storage Scale 5.2.0 or later, the default value of this setting is `yes`. Older configurations default to `no`. In IBM Storage Scale Server systems, where LUNs consist of multiple physical disks, this setting might underestimate the available I/O concurrency. Therefore, for IBM Storage Scale 6000 storage configurations, setting `ignorePrefetchLUNCount` to yes is a best practice.
- `maxblocksize`: This parameter defines the maximum allowable file system block size that can be used in a cluster. File systems with a block size larger than the specified value cannot be created or mounted unless `maxblocksize` is increased. For more information, see the documentation for the mmcrfs command.

  Starting with IBM Storage Scale 5.0.0.0, new clusters have a default `maxblocksize` of 4 MiB. Changing `maxblocksize` requires shutting down IBM Storage Scale on all nodes in

the cluster. It is a best practice to set `maxblocksize` to `16M` because this setting provides the greatest flexibility for potential future workloads, enabling file systems of any block size to be used.

- `maxMBpS`: Estimates the maximum bidirectional data transfer rate that a single client can achieve to its IBM Storage Scale servers. This value helps determine the I/O bandwidth that is available for data prefetching (for readers) and write-behind operations (for writers). The default value is `2048`. As a best practice, use MBps instead of MiBps to calculate `maxMBpS`, which is a tested approach that simplifies calculations. The optimal setting for each client should be twice its unidirectional network bandwidth to the IBM Storage Scale servers, up to a maximum of 100000. For example, if a client has a single 200 Gbps link to IBM Storage Scale servers, the `maxMBpS` setting should be `50000`. This setting is calculated by converting 200 Gbps to 25,000 MBps (unidirectional) and multiplying by 2 to account for bidirectional transfers.
- `maxFilesToCache`: This parameter defines the maximum number of unique files that can be cached on an IBM Storage Scale client at a time. It does not limit the number of files that can remain concurrently open on the node, and files can still be cached in memory after they are closed. The default `maxFilesToCache` value is `4000`, and the supported range 1 - 100 million. For NVIDIA DGX SuperPOD deployments, the best practice starting value is `128K`. Some workloads. particularly non-Direct I/O workloads that frequently access the same file data, might benefit from higher values, but be careful when tuning this option because increasing it results in more pageable IBM Storage Scale memory allocations that are used for caching on the clients, in addition to increased token memory allocations on the manager nodes in the storage cluster (also called the owning cluster, as described in 1.1.3, “Storage and client clusters” on page 7).
- `maxStatCache`: Specifies the number of inodes to keep in the stat cache. The stat cache maintains only enough inode information to resolve the `stat()` calls that are required by the `ls -l` command. The calculation for the default value of the `maxStatCache` parameter is described in the mmchconfig command. The valid ranges for `maxStatCache` are 0 - 100 million. The starting point best practice for this option in a SuperPOD deployment is `128K`. Some workloads, particularly ones that frequently perform `stat()` operations on the same inodes, might benefit from higher values, but be careful when tuning this option because increasing it results in more pageable IBM Storage Scale memory allocations that are used for caching on the clients, in addition to increased token memory allocations on the manager nodes in the storage cluster (also called the owning cluster, as described in 1.1.3, “Storage and client clusters” on page 7).
- `numaMemoryInterleave`: For IBM Storage Scale to allocate `pagepool` memory across multiple NUMA domains on a Linux system, this option must be set to `yes` and the `numactl` RPM must be installed. If this option is set to `no`, or the `numactl` RPM is not installed, then all `pagepool` memory will be allocated from a single NUMA domain, which can result in an unbalanced memory configuration. (You can check the number of NUMA domains in a machine has by running `numactl -H`.) The default for `numaMemoryInterleave` is `yes` for new clusters that are created with IBM Storage Scale 5.2.0 or later, but older cluster levels default to `no`. For performance reasons and to avoid memory imbalances that might lead to an out-of-memory condition, set this option to `yes` on systems with more than one NUMA domain.

- `pagepool`: This parameter defines the amount of `pagepool` memory that is used for caching data. For more information, see Pinned memory. When the `mmchconfig` option `dynamicPagepoolEnabled` is set to `no` (the best practice default for SuperPOD deployments), this parameter defines a fixed amount of pinned `pagepool` memory that is allocated for caching data on IBM Storage Scale clients. For clusters created with IBM Storage Scale 5.2.0 or later, the default fixed size of the client `pagepool` is one-third of the physical memory on the node or 4 GiB, whichever is smaller. In earlier versions, the default follows the same formula but substitutes 1 GiB instead of 4 GiB.

  A best practice is to start with `pagepool` set to 25% of the total installed memory per client, up to a maximum of 64 GiB, even for clients with more than 256 GiB of memory. Workloads that frequently reread the same file data without direct I/O might benefit from a larger `pagepool`, but do not overcommit memory on client nodes. Pinned memory cannot be swapped to disk, meaning GPFS always uses an amount of real memory equal to the `pagepool` size. When configuring `pagepool`, consider both the GPFS requirements and the memory needs of other applications running on the clients. Even workloads that do not benefit from caching still require some `pagepool` memory for optimal performance, so set `pagepool` no lower than 4 GiB on NVIDIA DGX SuperPOD client nodes, even if there is little expectation of rereading the same file data.

- `prefetchLargeBlockThreshold`: Introduced in IBM Storage Scale 5.1.3, this option sets a block size threshold for a gradual prefetching mode that increases request sizes progressively rather than always reading full file system block-sized I/O requests. Gradual prefetching is enabled for file systems with a block size equal to or greater than the configured `prefetchLargeBlockThreshold` value. Without this mode, larger file system block sizes result in higher prefetch rates, which might be inefficient due to read amplification, where excessive data is prefetched beyond what applications use. By default, `prefetchLargeBlockThreshold` is set to `32M`. Because IBM Storage Scale supports a maximum block size of `16M`, the default setting prevents the gradual prefetching mode from being used unless `prefetchLargeBlockThreshold` is adjusted. For NVIDIA DGX SuperPOD deployments, enabling gradual prefetching improves efficiency, particularly for larger block sizes, by reducing unnecessary prefetching that might overload IBM Storage Scale 6000 nodes. The downside to this tuning is that `mmap` workloads with sequential reads might see reduced performance in this alternative prefetching mode, although this situation is often outweighed by the benefits of reducing read amplification

- `proactiveReconnect`: When set to `yes`, this option causes IBM Storage Scale to close and reestablish TCP connections that appear to be in a problematic state. Clusters that are on a minimum release level (`minReleaseLevel`) from 5.0.4 to 5.2.1 have a default value of `yes`. Clusters that are created with IBM Storage Scale 5.2.2 or later have a default value of `no`. It is a best practice to set this setting to `no`, which disables it, for NVIDIA DGX SuperPOD deployments for performance and stability reasons. Setting `proactiveReconnect` to `yes` might impact the stability of clusters with network connectivity issues.

- `verbsPorts`: Specifies the interfaces that are used for all RDMA communication. Enable `verbsRdma` to enable RDMA to use the list of defined `verbsPorts`. The default value is a NULL string. The parameter must be set to enable RDMA. It is a best practice to enable RDMA for IBM Storage Scale. For more information about how to pick the *verbsPorts* setting in an NVIDIA DGX SuperPOD deployment, see 4.4, “Storage network configuration and validation” on page 101.

  The definition of `verbsPorts` consists of a series of elements, as shown in Example 4-5 on page 99.

Example 4-5 Defining the verbsPorts option

```
verbsPorts="RDMA_adapter_name[/Port_Number][/Fabric_Number/][/Desired_SL_Level]"
RDMA_adapter_name: The name of the adapter as returned from the ibv_devinfo
command
```

- `Port Number`: An optional field that defines the port number to use on the adapter, with a default of `1`. Sometimes a 2-port adapter might be described as two separate single port adapters in the output of `ibv_devinfo`. Use the output of `ibv_devinfo` to determine the appropriate port number to specify.
- `Fabric_Number`: An optional field that contains the virtual fabric ID number that the adapter is part of, with a default of `0`. This option is needed only if there are multiple distinct physical networks that do not have RDMA connectivity between each other. When such multiple networks are used, each distinct physical network must be assigned a different fabric number so that adapters on different physical fabrics do not attempt to communicate with each other.
- `Desired_SL_Level`: This optional field is used only by InfiniBand networks to define the Service Levels (SL) that the adapter should use when different quality of service (QoS) levels are defined for traffic. These SLs determine the path selection through the fabric based on virtual lanes (VLs) and network fabric settings. Configuring different InfiniBand SLs is typically not required and is out of scope for this document.
- `verbsGPUDirectStorage`: Enables or disables the GPUDirect Storage (GDS) feature. When GDS is enabled, file data can be transferred directly between an NSD server's page pool into the GPU buffer of an IBM Storage Scale client through RDMA. Using RDMA to transfer the data directly to the client's GPU offers a performance benefit due to skipping the copy of the data from the client host to the client's GPU. The default for this option is `no`. It is a best practice to set this option to `yes` only if the SuperPOD deployment intends to use the GPUDirect Storage (GDS) feature because enabling this feature can have a small performance impact on non-GDS workloads. For more information about GDS support software and hardware prerequisites, see GPUDirect Storage support for IBM Storage Scale. Changing this tuning option means that IBM Storage Scale must be stopped on all nodes in the cluster.
- `verbsRdma`**:** Enables or disables RDMA over InfiniBand or RoCE by using the Verbs API for data transfers between an NSD client and NSD server. Valid values are `enable` or `disable`. The default is `disable`. It is a best practice that this option is set to `enable` for network configurations that support RDMA.
- `verbsRdmaCm`: This option is required for RoCE because Ethernet lacks the subnet manager that is used in InfiniBand to establish dynamic connections. Enabling this option activates the RDMA Connection Manager, which uses the `RDMA_CM` API to establish connections. Valid values for `verbsRdmaCm` are `enable` or `disable`, with `disable` as the default. To enable RDMA on Ethernet networks, set `verbsRdmaCm` to `enable` along with `verbsRdma` and define a valid `verbsPorts` string. It is a best practice to set `verbsRdmaSend` to `enable` for optimal RDMA performance.
- `verbsRdmaSend`: Enables or disables the usage of RDMA for more IBM Storage Scale communication than only data transfers between NSD clients and NSD servers. Valid values are `yes` or `no`, and the default is `no`. It is a best practice to enable this option on networks that support RDMA. When set to `yes`, the `verbsRdma` option should be enabled and a valid `verbsPorts` setting must be defined. When the attribute is set to `no`, only data transfers between an NSD server and an NSD client are eligible for RDMA if RDMA is enabled by using the `verbsRdma` setting. When this option is set to `yes`, IBM Storage Scale uses RDMA connections for most daemon-to-daemon communication.

- `workerThreads`: This option controls an integrated group of variables that tune the level of concurrency of various aspects of file system performance. The default value is `256`. For new clusters that are installed with IBM Storage Scale 5.2.0 or later, the default value is changed to `256` from the previous default value of `48`. The valid range of tuning options is 1 – 8192. For IBM Storage Scale 6000 SuperPOD deployments, set this tuning option set to 1024. Setting this value too low results in less than optimal performance due to insufficient concurrency, and setting this option too high can result in a performance loss that comes from locking-related overheads.

4.3 IBM Storage Scale tuning best practices for IBM Storage Scale 6000 servers

Tuning for the IBM Storage Scale System 6000 is largely automated by the IBM Storage Scale code, but you check the RDMA tuning recommendations to help ensure that RDMA is correctly setup. As described in 4.2.3, “Details about best practices for the mmchconfig tuning options” on page 96, use the configuration options in Example 4-6 to enable RDMA.

Example 4-6 Enabling RDMA

```
verbsRdma=enable
verbsRdmaSend=yes
verbsPorts=<list of RDMA-capable interfaces used to access the IBM Storage Scale
clients>
```

If RDMA over Ethernet (RoCE) is enabled, you must use the setting that is shown in Example 4-7.

Example 4-7 Enabling verbsRdmaCm for RoCE

```
verbsRdmaCm=enable
```

When you define the list of `verbsPorts` in a NVIDIA DGX SuperPOD deployment, select only the network interfaces that the clients use to communicate with the IBM Storage Scale servers on the storage network. For more information about how to pick the appropriate **verbsPorts** setting in a SuperPOD deployment, see 4.4, “Storage network configuration and validation” on page 101.

There are two other `mmchconfig` options that you can consider setting on the IBM Storage Scale 6000 servers:

- If the IBM Storage Cluster was created with code levels earlier than 5.2.2, disable `proactiveReconnect` for stability and performance reasons, as described in 4.2, “Starting point for IBM Storage Scale tuning best practice for clients” on page 94. Set the option as `proactiveReconnect=yes`.
- If you plan to use GPUDirect Storage, set the `verbsGPUDirectStorage=yes` option, as described in Configuring GPUDirect Storage for IBM Storage Scale.

Note: If you set `verbsGPUDirectStorage`, you must shut down IBM Storage Scale on all nodes in the cluster.

Verifying the IBM Storage Scale 6000 tuned configuration

IBM Storage Server (previously referred to as the IBM Elastic Storage System (IBM ESS) tuning uses a tuned Red Hat subsystem. To validate the IBM Storage Scale 6000 tuning configuration, run the `tuned-adm active` and `tuned-adm verify` commands, as shown in Example 4-8.

Example 4-8 The tuned-adm active and tuned-adm verify commands

```
# tuned-adm active
Current active profile: scale
# tuned-adm verify
Verification succeeded, current system settings match the preset profile.
See the tuned log file ('/var/log/tuned/tuned.log') for details.
```

Further ESS configuration validation

For more verification of an IBM Storage Scale 6000 configuration, run the `essinstallcheck` command on the management server that is defined for the IBM Storage Scale 6000. The `essinstallcheck` command performs a tuned verification and checks other aspects of the installation, such as firmware and code levels. For more information, see the essinstallcheck command.

4.3.1 IBM Storage Scale 6000 file system block size considerations

As a best practice, start with an 8 MB block size, but you can consider using a 16 MB block size with an NVIDIA DGX SuperPOD or similar configuration. In general, workloads emphasizing large streaming I/Os perform better with 8 MiB and 16 MiB block sizes. An 8 MiB block size provides excellent large I/O performance while offering some of the benefits of a smaller block size.

Here are the advantages of a smaller file system block size:

- Because GPFS locking is done at the file system block size level of granularity, greater small I/O parallelism may be achieved with smaller block sizes when using buffered I/O.
- More consistent and less aggressive prefetching behavior may be achieved with smaller block sizes by default, but this behavior can be addressed by using the tuning option `prefetchLargeBlockThreshold`.

Note: Smaller block sizes no longer have the potential space savings benefits that they had in releases before IBM Storage Scale 5. Version 5 changed the default block size to 4 MiB and introduced variable sub block sizes, making space allocations for smaller files more efficient with larger block sizes and improving file creation and block allocation times.

4.4 Storage network configuration and validation

When deploying an NVIDIA DGX SuperPOD deployment with IBM Storage Scale 6000 storage, not all network interfaces on the clients are used for communication with the IBM Storage Scale server nodes, as described in 2.4.1, “AFM concepts” on page 44. This section describes how to determine the appropriate network interfaces to configure when defining the `mmchconfig` options, and how to configure the `nsdperf` tool to validate the performance of the storage network.

As described in 1.4.5, “Storage fabric” on page 24, it is a best practice to configure RDMA on the IBM Storage Scale clients and servers. Section 4.4.1, “Selecting the network interface list for RDMA communication for the IBM Storage Scale 6000 server node network” on page 102 describes how to pick the interfaces that are used for the `mmchconfig verbsPorts` option, and validating the network by using `nsdperf`.

4.4.1 Selecting the network interface list for RDMA communication for the IBM Storage Scale 6000 server node network

The IBM Storage Scale 6000 storage servers in a SuperPOD deployment use all their available links to communicate with IBM Storage Scale clients.

For RDMA configurations, one way to determine the list of interfaces to configure for the `mmchconfig verbsPorts` option is to run `ibv_devinfo` and include all links that are cabled and timed.

Example 4-9 shows an example of running `ibv_devinfo` on an IBM Storage Scale 6000 server that has eight HDR200 links that are cabled for a total of 400 Gbps network bandwidth.

Example 4-9 Running ibv_devinfo

```
ibv_devinfo  | grep -w -e hca_id -e state: -e port:
hca_id: mlx5_0
                port:   1
                        state:                  PORT_ACTIVE (4)
hca_id: mlx5_1
                port:   1
                        state:                  PORT_ACTIVE (4)
hca_id: mlx5_2
                port:   1
                        state:                  PORT_ACTIVE (4)
hca_id: mlx5_3
                port:   1
                        state:                  PORT_ACTIVE (4)
hca_id: mlx5_4
                port:   1
                        state:                  PORT_ACTIVE (4)
hca_id: mlx5_5
                port:   1
                        state:                  PORT_ACTIVE (4)
hca_id: mlx5_6
                port:   1
                        state:                  PORT_ACTIVE (4)
hca_id: mlx5_7
                port:   1
                        state:                  PORT_ACTIVE (4)
```

Note: This example configuration uses eight 200 Gbps connections, but the IBM Storage Scale 6000 also supports four 400 Gbps connections per IBM Storage Scale 6000 server on InfiniBand networks, and similar support is planned for Ethernet at the time of writing.

In Example 4-9, there are two physical networks (each adapter alternates networks) in the list of `hca_ids` (`mlx5_0`, `mlx5_1`, `mlx5_2`, `mlx5_3`, `mlx5_4`, `mlx5_5`, `mlx5_6`, and `mlx5_7`).

This example configuration assumes that the following adapters are connected to the first fabric on the IBM Storage Scale 6000 servers:

```
mlx5_0,mlx5_2,mlx5_4,mlx5_6
```

This example also assumes that the following adapters are connected to the second fabric on the IBM Storage Scale 6000 servers:

```
mlx5_1,mlx5_3,mlx5_5,mlx5_7
```

For this example, refer to these example lists of `hca_ids` for running `nsdperf` to validate network performance and defining the `verbsPorts` configuration option for the IBM Storage Scale 6000 servers. As described in 4.2.3, “Details about best practices for the mmchconfig tuning options” on page 96, the `verbsPorts` definition has the format that is shown in Example 4-10.

Example 4-10 The verbsPorts definition

```
verbsPorts="RDMA_adapter_name[/Port_Number][/Fabric_Number/][/Desired_SL_Level]"
```

For Example 4-10, enter the `hca_ids` and port numbers, and assign a virtual network to both of the physical networks. Define the first switch as `fabric_number=0`, and the second switch as `fabric_number=1`, as shown in Example 4-11.

Example 4-11 Defining the switch fabrics

```
mmchconfig verbsPorts="mlx5_0/1/0 mlx5_1/1/1 mlx5_2/1/0 mlx5_3/1/1 mlx5_4/1/0
mlx5_5/1/1 mlx5_6/1/0 mlx5_7/1/1" -N <insert_name_of_ISS6000_node_class>
```

Use this list of `hca_ids` when running `nsdperf` to validate network performance and defining the `verbsPorts` configuration option for the IBM Storage Scale 6000 servers.

Perform the same set of steps on the client nodes, but exclude the `hca_ids` that are used for communication between the client nodes. Instead, include only the `hca_ids` that are dedicated to communication with the storage server nodes.

When configuring fabric numbers, check how the adapters are connected to physical switches in your environment. As a best practice, have two separate physical fabrics, but configurations involving only a single fabric are also described in this section. For this example, assume that you have the following fabric mapping for the client adapters that are connected to the storage network:

```
mlx5_8 port 1 connected to the first fabric
mlx5_9 port 1 connected to the second fabric
```

Use `mlx5_8` and `mlx5_9` as the clients’ storage fabric connections for the example, but check which adapters on the compute nodes are connected to the storage network on your system. The example here is from an H100 system with eight dual-ported adapters, in which each port might show up in `ibv_devinfo` as a separate ‘port 1’ adapter, such that the first dual-ported connections appear as separate adapters (for example, port 1 on `mlx5_0` and `mlx5_1` instead of port 1 and port 2 on `mlx5_0`).

For this example, use the first four `hca_ids` that are connected to the storage fabric, and you fill in the port numbers and `fabric_ids` in the same manner as you did on the storage servers. Define the first switch as `fabric_number=0` and the second switch as `fabric_number=1`, which results in the verbsPorts definition that is shown in Example 4-12.

Example 4-12 The verbsPorts definition

```
mmchconfig verbsPorts="mlx5_8/1/0 mlx5_9/1/1"
```

The same list of `verbsPorts` interfaces that you define on the clients is used in configuring `nsdperf` to validate network performance with RDMA.

4.4.2 Configuring TCP/IP communication

Using RDMA for IBM Storage Scale communication is a best practice to achieve optimal performance, but all IBM Storage Scale deployments require configuring TCP/IP.

If RDMA is not configured, then ensure that all the network interfaces on the IBM Storage Scale 6000 are configured to communicate with the network interfaces that are connected to the storage network on the compute nodes, and that all network interfaces on the compute nodes are used for communication to the IBM Storage Scale servers. This task can be accomplished by assigning the IBM Storage Scale daemon node name to a virtual bonded interface or a single interface that uses Layer 3 routing to use all physical network interfaces that are connected to the storage fabric.

Another option when not using RDMA is configuring the IBM Storage Scale Multi-Rail over TCP (MROT) feature, as described in Configuring multi-rail over TCP (MROT). For more information, see GPFS and network communication and Creating an IBM Storage Scale cluster. Relying on MROT without bonding, teaming, or using L3 routing does not provide a fully resilient solution against link failures because some aspects of IBM Storage Scale, such the IBM Storage Scale daemon startup and the Cluster Configuration Repository (CCR), rely on the daemon node name address being reachable.

Network verification with nsdperf

An important part of installing a new SuperPOD deployment is to help ensure that the network is functional and performing well. Initial network verification can be done by using the `mmnetverify` tool, as described in Verifying network operation with the mmnetverify command, but run the `nsdperf` tool to validate the performance.

Use the `nsdperf` tool to measure network throughput in a manner that is representative of IBM Storage Scale network usage. All network communication is done by using TCP socket connections or RDMA verbs, and the mode of communication that you choose for `nsdperf` should match the configuration that is used for IBM Storage Scale. This section describes the steps that are involved in validating network performance with `nsdperf`.

For RDMA communication over two separate physical networks, you may run performance tests with a single instance of `nsdperf`, defining different `fabric_numbers` in the RDMA interface strings that are passed to the `nsdperf` server processes, as you did for the `mmchconfig verbsPorts` strings. However, you might encounter a potential problem with the `fabric_ids` that `nsdperf` uses.

To avoid this problem, this section shows how two physical networks can be tested with RDMA running two concurrent instances of `nsdperf`, each instance using a separate TCP port number. You can use this same approach of running multiple instances of `nsdperf` concurrently to test a TCP/IP configuration that uses multiple IP interfaces instead of a single bonded or teaming daemon node name, as described in Configuring multi-rail over TCP (MROT). Step 2 on page 106 also provides more details.

The following steps show an example of running `nsdperf` with RDMA by using the `hca_ids` and mappings that are described in 4.4.1, “Selecting the network interface list for RDMA communication for the IBM Storage Scale 6000 server node network” on page 102. It is a best practice to enable RDMA for IBM Storage Scale communication, but this section provides instructions about how to modify the RDMA measurement example to test TCP/IP communication without RDMA.

When testing a non-RDMA configuration that uses a single bonded (or teaming) interface for IBM Storage Scale daemon node name without configuring MROT, then you can use a single instance of `nsdperf` to test network performance.

Complete the following steps:

1. Create the `host.list` files.

   Create files that contain the client and server hostnames that you will use for `nsdperf` testing. These files will be used with the `mmdsh` command to manage `nsdperf` processes (see the mmdsh command). Create these files on a node that is not part of the client and server list that has `mmdsh` access to all the clients and servers, such as a management server node.

   In this example of network verification, use the client list that is saved in a file that is called `client.list`, as shown in Example 4-13.

Example 4-13 The client.list file

```
c91f93h200-1
c91f93h200-2
c91f93h200-3
c91f93h200-4
c91f93h200-5
c91f93h200-6
c91f93h200-7
c91f93h200-8
c91f93h200-9
c91f93h200-10
c91f93h200-11
c91f93h200-12
c91f93h200-13
c91f93h200-14
c91f93h200-15
c91f93h200-16
```

   Use the server list (one server name per line) that is shown in Example 4-14, and save it in a file that is named `server.list`.

Example 4-14 Server list

```
c91ess6000-1-a
c91ess6000-1-b
```

2. Build nsdperf.

   The `nsdperf` utility is a sample utility with IBM Storage Scale. It is in the `/usr/lpp/mmfs/samples/net` directory. Build the utility before you use it:

   – For systems that use RDMA, build the utility as shown in Example 4-15.

Example 4-15 Building nsdperf for RDMA

```
cd /usr/lpp/mmfs/samples/net
g++ -O2 -DRDMA -o nsdperf nsdperf.C -lpthread -lrt -libverbs -lrdmacm
```

   – For systems that do not have RDMA configured, use the command that is shown in Example 4-16 to build `nsdperf` for only TCP/IP measurements.

Example 4-16 Building nsdperf for only TCP/IP measurements

```
cd /usr/lpp/mmfs/samples/net
g++ -O2 -o nsdperf nsdperf.C -lpthread -lrt
```

   For example, you can build `nsdperf` with RDMA support on all the client and server nodes by running the command that is shown in Example 4-17.

Example 4-17 Building nsdperf with RDMA support

```
mdsh -N client.list,server.list "cd /usr/lpp/mmfs/samples/net; g++ -O2 -DRDMA -o
nsdperf nsdperf.C -lpthread -lrt -libverbs -lrdmacm; cksum
/usr/lpp/mmfs/samples/net/nsdperf"
```

3. Kill any previous copies of `nsdperf` that might be running.

   Before each network test, kill any copies of `nsdperf` that might be running on both the clients and servers. In this example, use the `pkill` command to kill processes that match the specific arguments that you use in `nsdperf`.

   In this example, you use port 6665 for the test that is running on the first network fabric and port 6666 for the test that is running on the second network fabric, as shown in Example 4-18. You can include these port numbers to narrow the scope of the string that is passed to the `pkill` command to kill `nsdperf` processes (this action limits the chance of killing the wrong process).

Example 4-18 Killing the nsdperf processes

```
mmdsh -N client.list,server.list "pkill -f \"/usr/lpp/mmfs/samples/net/nsdperf -p
6665\""
mmdsh -N client.list,server.list "pkill -f \"/usr/lpp/mmfs/samples/net/nsdperf -p
6666\""
```

   To check whether any `nsdperf` processes are still running, run the command that is shown Example 4-19.

Example 4-19 Checking for nsdperf processes that are running

```
mmdsh -N client.list,server.list "ps -ef | grep nsdperf | grep -v grep"
```

4. Start the nsdperf server processes on the IBM Storage Scale 6000 server nodes.

   To run nsdperf, start the nsdperf server process on the list of server nodes. With RDMA enabled, use port 6665 in this example to control the copies of nsdperf that test the first fabric, and port 6666 to control the copies of nsdperf that test the second fabric. For RDMA configurations, include the -r option with nsdperf to define the list of hca_ids that are used for RDMA communication. Use the list of hca_ids and fabric mappings that are listed in Example 4-9 on page 102. In this example, the servers connect to the first storage fabric by using port 1 on hca_ids mlx5_0, mlx5_2, mlx5_4, and mlx5_6, and you start the nsdperf server processes on port 6665 by using the command that is shown in Example 4-20.

Example 4-20 Starting the nsdperf server processes on port 6665

```
mmdsh -N server.list “/usr/lpp/mmfs/samples/net/nsdperf -p 6665 -s -w 192 \
-r mlx5_0/1/0,mlx5_2/1/0,mlx5_4/1/0,mlx5_6/1/0" > /tmp/nsdperf.server.log1 2>&1 &
```

   When RDMA is not enabled, do not use the -r option, so use the command that is shown in Example 4-21 instead of the one that is shown in Example 4-20.

Example 4-21 Starting the nsdperf servers processes without RDMA

```
mmdsh -N server.list “/usr/lpp/mmfs/samples/net/nsdperf -p 6665 -s -w 192" >
/tmp/nsdperf.server.log1 2>&1 &
```

   When you test a non-RDMA configuration that uses a single bonded (or teaming) interface for the IBM Storage Scale daemon node name without configuring MORT, then you may use a single instance of nsdperf to test network performance. If you are testing a non-RDMA configuration, go to step 5.

   In this example with RDMA enabled, the servers connect to the second storage fabric by using hca_ids mlx5_1, mlx5_3, mlx5_5, and mlx5_7. Start the nsdperf server processes on port 6666, as shown in Example 4-22.

Example 4-22 Starting the nsdperf server processes on port 6666

```
mmdsh -N server.list “/usr/lpp/mmfs/samples/net/nsdperf -p 6666 -s -w 192 \
-r mlx5_1/1/1,mlx5_3/1/1,mlx5_5/1/1,mlx5_7/1/1" > /tmp/nsdperf.server.log2 2>&1 &
```

5. Start the nsdperf server processes on the SuperPOD client nodes.

   Start the nsdperf server processes on the clients. With RDMA enabled, use port 6665 to control the copies of nsdperf that test the first fabric and port 6666 to control the copies of nsdperf that test the second fabric. For RDMA configurations, use the -r option with nsdperf to define the list of hca_ids that are used for RDMA communication. Use the list of hca_ids and fabric mappings from Example 4-9 on page 102. The clients connect to the first storage fabric by using port 1 on hca_id mlx5_8, so start the nsdperf server processes on port 6665 by running the command in Example 4-23.

Example 4-23 Starting the nsdperf server processes on port 6665

```
mmdsh -N client.list “/usr/lpp/mmfs/samples/net/nsdperf -p 6665 -s -w 192 -r
mlx5_8/1/0" > /tmp/nsdperf.client.log1 2>&1 &
```

When RDMA is not enabled, do not use the `-r` option, so use the command that is shown in Example 4-24 instead of the one that is shown in Example 4-23 on page 107.

Example 4-24 Starting the nsdperf server processes on port 6665 without option -r

```
mmdsh -N client.list "/usr/lpp/mmfs/samples/net/nsdperf
-p 6665 -s -w 192" > /tmp/nsdperf.client.log1 2>&1 &
```

As noted in step 4 on page 107, when testing a non-RDMA configuration that uses a single bonded (or teaming) interface for IBM Storage Scale daemon node name without configuring MROT, you may use a single instance of `nsdperf` to test network performance. If you are testing a non-RDMA configuration, go to step 6.

In this example with RDMA enabled, the clients connect to the second storage fabric by using port 1 on `hca_id mlx5_9`, so start the `nsdperf` server processes on port 6666, as shown in Example 4-25.

Example 4-25 Starting the nsdperf server processes on port 6666

```
mmdsh -N client.list "/usr/lpp/mmfs/samples/net/nsdperf -p 6666 -s -w 192 -r
mlx5_9/1/1" > /tmp/nsdperf.client.log2 2>&1 &
```

Check that every node has a `nsdperf` server process running by running the command in Example 4-26.

Example 4-26 Checking that every node has a nsdperf server process running

```
mmdsh -N server.list,client.list "ps -ef | grep nsdperf | grep -v grep" | wc -l
18
```

Note: The value that is returned from `wc -l` should be equal to the total number of clients and servers that are involved in the test.

6. Create a script file of the `nsdperf` command and run it.

   Create a `nsdperf` command file (`nsdperf.cmd.file` in this example) with the full list of `nsdperf` commands, along with the names of the client and server network interfaces to use.

   Example 4-27 shows a sample file with a list of clients and server names, followed by a list of `nsdperf` commands. When you create your own `nsdperf.cmd.file` file, substitute the list of hosts that are configured in your environment for the ones that are shown in the example file. When using TCP/IP for the IBM Storage Scale data transfers instead of RDMA, the names that are assigned to each client and server keywords must use the physical networks that are assigned to the storage network. For example, the client and server names should be (virtual) bonded or teaming interfaces that aggregate all the physical links that are connected to the storage network.

   If an MROT configuration is used to enable the usage of multiple IP interfaces by IBM Storage Scale instead of using bonding or teaming, then you can run multiple copies of `nsdperf` by using different TCP/IP ports, as described in 4.4.3, “Validating networks without RDMA enabled” on page 110.

Example 4-27 Sample nsdperf.cmd.file file

```
server c91ess6000-1-a
server c91ess6000-1-b
client c91f93h200-1
client c91f93h200-2
client c91f93h200-3
```

```
client c91f93h2O0-4
client c91f93h2O0-5
client c91f93h2O0-6
client c91f93h2O0-7
client c91f93h2O0-8
client c91f93h2O0-9
client c91f93h2O0-10
client c91f93h2O0-11
client c91f93h2O0-12
client c91f93h2O0-13
client c91f93h2O0-14
client c91f93h2O0-15
client c91f93h2O0-16
ttime 12O
buffsize 16777216
threads 32
parallel 2
usecm on # this is only needed for Ethernet RoCE runs
rdma on
test nwrite
test read
quit
```

You want to run the same `nsdperf.cmd.file` file on both fabrics concurrently to measure the aggregate network performance that can be achieved. If you must do isolated debugging, run one fabric at a time or reduce the list of nodes that you test by modifying the command file to exclude nodes.

Run the `nsdperf.cmd.file` file on both fabrics concurrently by running the commands that are shown in Example 4-28.

Example 4-28 Running the nsdperf.cmd.file file on both fabrics concurrently

```
/usr/lpp/mmfs/samples/net/nsdperf -i nsdperf.cmd.file -p 6665 >
nsdperf.fabricO.out 2>&1 &
/usr/lpp/mmfs/samples/net/nsdperf -i nsdperf.cmd.file -p 6666 >
nsdperf.fabric1.out 2>&1 &
wait
```

7. Review the `nsdperf` output and address any network connectivity problems.

   Make sure that all `nsdperf` server processes started on all the client and server hosts that are defined in the `nsdperf.cmd.file` files. For any node that the `nsdperf` server process is not started on correctly, you see a `Connection refused` error in the output file:

   ```
   Cannot connect to c91f93h2OO-2: Connection refused
   ```

   If such errors are in the output, check for nodes that are not running `nsdperf` server processes and review the output from attempts to start the `nsdperf` server processes (for example, `/tmp/nsdperf.client.log`). Make sure that both the `nsdperf` and `numactl` processes exist and have `execute` permission.

   Also, look for errors about establishing connections for other reasons, which might indicate network problems or misconfigurations. If all the nodes are connected, you should see the correct client/server counts in the `nsdperf` output lines that precede the nwrite and read performance results.

Example 4-29 shows an example ‘nwrite test, which simulates the clients writing to an IBM Storage Scale file system on the server nodes. This example ran with 16 clients connecting to two servers (one IBM Storage Scale 6000 building block), and the expected client/server count in the 16-2 nwrite output is correct.

Example 4-29 Example nwrite test

```
grep nwrite nsdperf.out.1 nsdperf.out.2
nsdperf.out.1:16-2 nwrite 174000 MB/sec (10400 msg/sec), cli 1% srv 8%, time 120,
buff 16777216, th 32, parallel 2, RDMA
nsdperf.out.2:16-2 nwrite 174000 MB/sec (10400 msg/sec), cli 1% srv 8%, time 120,
buff 16777216, th 32, parallel 2, RDMA
```

In this nwrite example, both concurrent tests achieved 174000 MBps for an aggregate network bandwidth of 348 GBps across both tests. Write performance on a single IBM Storage Scale 6000 building block is limited by disk performance, and any result over 250 GBps indicates that there is enough network bandwidth to achieve optimal performance.

Example 4-30 shows an example read test, which simulates the clients that are read from an IBM Storage Scale file system on the server nodes. This example ran with 16 clients connecting to two servers (one IBM Storage Scale 6000 building block), and the expected client/server count in the “16-2 read” output is correct.

Example 4-30 Example read test

```
grep -w read nsdperf.out.1 nsdperf.out.2
nsdperf.out.1:16-2 read 192000 MB/sec (11500 msg/sec), cli 1% srv 9%, time 120,
buff 16777216, th 32, parallel 2, RDMA
nsdperf.out.2:16-2 read 192000 MB/sec (11500 msg/sec), cli 1% srv 9%, time 120,
buff 16777216, th 32, parallel 2, RDMA
```

In this read example, both concurrent tests achieved 192000 MBps for an aggregate network bandwidth of 384 GBps across both tests. Read performance on a single IBM Storage Scale 6000 building block is limited by the servers’ system fabric performance, and any result over 350 GBps indicates that there is enough network bandwidth to achieve optimal performance.

If you are getting less than the expected performance, see 4.4.4, “Diagnosing network performance issues” on page 111.

4.4.3 Validating networks without RDMA enabled

If you must test multiple IP interfaces concurrently (for example, if you use an MROT configuration without RDMA), create different `nsdperf` command files that define all the IP interfaces that you want to test. When using this approach, ensure that all the hostname interfaces that specified in the same file have connectivity to each other. In Example 4-31 on page 111 and Example 4-32 on page 111, all the -eth0 and -eth1 interfaces on the clients and servers have connectivity to each other.

Example 4-31 Sample nsdperf.cmd.file.1 file for MROT tests

```
server c91ess6000-1-a-eth0
server c91ess6000-1-b-eth0
client c91f93h200-1-eth0
client c91f93h200-2-eth0
[…]
ttime 120
buffsize 16777216
threads 32
parallel usecm on # this is only needed for Ethernet RoCE runs
rdma on
test nwrite
test read
quit
```

Example 4-32 Sample nsdperf.cmd.file.2 file for MROT tests

```
server c91ess6000-1-a-eth1
server c91ess6000-1-b-eth1
client c91f93h200-1-eth1
client c91f93h200-2-eth1
[…]
ttime 120
buffsize 16777216
threads 32
parallel usecm on # this is only needed for Ethernet RoCE runs
rdma on
test nwrite
test read
quit
```

Note: Create as many files that you need for your environment.

For this non-RMDA configuration, concurrently run the `nsdperf` command files that define the different IP interfaces that you want to test, as shown in Example 4-33.

Example 4-33 The nsdperf command files for different IP interfaces

```
/usr/lpp/mmfs/samples/net/nsdperf -i nsdperf.cmd.file.1 -p 6665 >
nsdperf.mrot0.out 2>&1 &
/usr/lpp/mmfs/samples/net/nsdperf -i nsdperf.cmd.file.2 -p 6666 >
nsdperf.mrot1.out 2>&1 &
#[… more than 2 copies of nsdperf can be run in this manner . . .]
wait
```

4.4.4 Diagnosing network performance issues

Running the `nsdperf` command is an effective way to validate storage network performance. If network performance is below the expected throughput for your infrastructure, isolate which part of the network is experiencing issues. For example, you can determine whether the problem is limited to a single network adapter or a specific switch link.

Here are some options for debugging:

- Test specific links by modifying the `-r` option in the `nsdperf` server process to change the RDMA interfaces under test.
- Remove nodes from the hostlist to isolate network performance at the node level.
- Check for errors and dropped packets on network adapters and switches by using tools such as the following ones:
  - `ethtool -S <interface>` for Ethernet interfaces and InfiniBand links. It provides more detailed information about packet loss.
  - `ibstat` or `ibportstate` for InfiniBand links.
  - `netstat -in` provides high-level statistics about dropped packets for both Ethernet interfaces and InfiniBand links.
  - For pure RDMA flows, commands like `rdma statistic` show RDMA-specific statistics.

Network verification can be done with other tools, such as `ib_send_bw` and `ib_read_bw`, but `nsdperf` is intended to be representative of IBM Storage Scale usage of the network, so it is the best choice for initial analysis.

If the issue appears isolated to specific network interfaces, further investigation of switch configurations, link errors, or congestion is recommended.

Some performance issues that appear network-related might originate from other system bottlenecks. CPU jitter, memory over-commitment, or misconfigured NUMA settings can impact performance, so check them alongside network metrics.

This document focuses on the methodology for verifying the network by using `nsdperf` and basic error checks. For advanced debugging, or if network issues are identified during validation, the appropriate support teams should be engaged. For IBM Storage Scale related issues, contact IBM Storage Scale support. For NVIDIA Spectrum-X switch issues, contact NVIDIA support.

4.5 IBM Storage Scale 6000 and client cluster configuration performance considerations

When defining node designation roles in an IBM Storage Scale cluster, one of the most important considerations for performance is manager node selection. Manager nodes handle token management, which is a performance-sensitive operation that coordinates access to storage resources, which can be memory-intensive depending on how the system is tuned.

Increasing the number of manager nodes can improve parallelism, enabling more efficient handling of token-related requests and providing a larger memory pool for token management. However, the total available memory for managing tokens is limited by the manager node with the smallest amount of available memory. If one or more nodes in the cluster have less memory than the others, exclude them from the manager node list to prevent performance bottlenecks.

Also, if multiple file systems are defined within the cluster, distribute the file system manager role across multiple manager nodes to help ensure that no single node is overwhelmed with file system management tasks, which improves scalability and resilience (for more information, see the mmchmgr documentation).

Beyond memory, CPU and network bandwidth should be considered when designating manager nodes. High token request traffic can increase both CPU load and network impact, making it essential to select nodes that have sufficient processing power and connectivity in addition to memory capacity. Careful selection of manager nodes based on memory, CPU, network capacity, and workload distribution ensures optimal IBM Storage Scale performance and scalability. For more information, see File system manager.

A

IBM Storage Scale System 6000 hosts file for NVIDIA DGX SuperPOD

This appendix is an example `/etc/hosts` file that includes the fully qualified hostnames for all servers in the IBM Storage Scale System 6000 storage cluster solution and the NVIDIA DGX IBM Storage Scale client cluster.

This chapter describes the following topic:

- Contents of the /etc/hosts file for the IBM Storage Scale NVIDIA DGX SuperPOD solution

Contents of the /etc/hosts file for the IBM Storage Scale NVIDIA DGX SuperPOD solution

Here are the contents of the `/etc/hosts` file for the IBM Storage Scale NVIDIA DGX SuperPOD solution:

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.l.ocaldomain6

#IBM Storage Scale EMS Campus Network
10.153.10.1  emsvm01.superpod.company.com emsvm01
10.153.10.101  ems01.superpod.company.com ems01

#1GbE IBM Storage Scale Management Network
192.168.10.1   ems01.superpod.company.com   ems01
192.168.10.2   n01a.superpod.company.com    n01a
192.168.10.3   n01b.superpod.company.com    n01b
192.168.10.4   n02a.superpod.company.com    n02a
192.168.10.5   n02b.superpod.company.com    n02b
192.168.10.6   n03a.superpod.company.com    n03a
192.168.10.7   n03b.superpod.company.com    n03b
192.168.10.8   n04a.superpod.company.com    n04a
192.168.10.9   n04b.superpod.company.com    n04b

#IBM Storage Scale Baseboard Management Controller (BMC) Network
10.0.0.101 ems01-bmc.superpod.company.com   ems01
10.0.0.1   ems01vm-bmc.superpod.company.com   emsvm01
10.0.0.2   n01a-bmc.superpod.company.com    n01a
10.0.0.3   n01b-bmc.superpod.company.com    n01b
10.0.0.4   n02a-bmc.superpod.company.com    n02a
10.0.0.5   n02b-bmc.superpod.company.com    n02b
10.0.0.6   n03a-bmc.superpod.company.com    n03a
10.0.0.7   n03b-bmc.superpod.company.com    n03b
10.0.0.8   n04a-bmc.superpod.company.com    n04a
10.0.0.9   n04b-bmc.superpod.company.com    n04b

#IBM Storage Scale Highspeed InfiniBand Network
10.40.10.1   ems01-ib.superpod.company.com   ems01-ib
10.40.10.2   n01a-ib.superpod.company.com    n01a-ib
10.40.10.3   n01b-ib.superpod.company.com    n01b-ib
10.40.10.4   n02a-ib.superpod.company.com    n02a-ib
10.40.10.5   n02b-ib.superpod.company.com    n02b-ib
10.40.10.6   n03a-ib.superpod.company.com    n03a-ib
10.40.10.7   n03b-ib.superpod.company.com    n03b-ib
10.40.10.8   n04a-ib.superpod.company.com    n04a-ib
10.40.10.9   n04b-ib.superpod.company.com    n04b-ib

#IBM Storage Scale Protocol/CES nodes -- Management/CES IPs
10.153.10.92   ces01.superpod.company.com   ces01
10.153.10.93   ces02.superpod.company.com   ces02

#IBM Storage Scale Protocol/CES nodes -- BMC Network
10.0.10.1  ces01-bmc.superpod.company.com   ces01-bmc
10.0.10.2  ces02-bmc.superpod.company.com   ces02-bmc
```

```
#IBM Storage Scale Protocol/CES nodes - CES VIPs
10.153.10.94  cesvip01.superpod.company.com ces
10.153.10.95  cesvip02.superpod.company.com ces

#IBM Storage Scale Protocol/CES nodes -- Highspeed InfiniBand Network
10.40.10.10 ces01-ib.superpod.company.com ces01-ib
10.40.10.11 ces02-ib.superpod.company.com ces02-ib

#IBM Storage Scale Active File Management (AFM) nodes -- Highspeed InfiniBand
Network
10.40.10.12 afm01-ib.superpod.company.com afm01-ib
10.40.10.13 afm02-ib.superpod.company.com afm02-ib
10.40.10.14 afm03-ib.superpod.company.com afm03-ib
10.40.10.15 afm02-ib.superpod.company.com afm02-ib

#NVIDIA DGX SuperPOD storage InfiniBand network
#
## NVIDIA DGX SuperPOD Login Node Highspeed InfiniBand Storage Network
#
10.40.1.81    slogin01-ib.superpod.company.com  slogin01-ib
10.40.1.82    slogin02-ib.superpod.company.com  slogin02-ib
10.40.1.83    slogin03-ib.superpod.company.com  slogin03-ib
#
# NVIDIA DGX SuperPOD Highspeed InfiniBand Storage Network
#
10.40.1.1     dgx01-ib.superpod.company.com  dgx01-ib
10.40.1.2     dgx02-ib.superpod.company.com  dgx02-ib
10.40.1.3     dgx03-ib.superpod.company.com  dgx03-ib
10.40.1.4     dgx04-ib.superpod.company.com  dgx04-ib
10.40.1.5     dgx05-ib.superpod.company.com  dgx05-ib
10.40.1.6     dgx06-ib.superpod.company.com  dgx06-ib
10.40.1.7     dgx07-ib.superpod.company.com  dgx07-ib
10.40.1.8     dgx08-ib.superpod.company.com  dgx08-ib
10.40.1.9     dgx09-ib.superpod.company.com  dgx09-ib
10.40.1.10    dgx10-ib.superpod.company.com  dgx10-ib
10.40.1.11    dgx11-ib.superpod.company.com  dgx11-ib
10.40.1.12    dgx12-ib.superpod.company.com  dgx12-ib
10.40.1.13    dgx13-ib.superpod.company.com  dgx13-ib
10.40.1.14    dgx14-ib.superpod.company.com  dgx14-ib
10.40.1.15    dgx15-ib.superpod.company.com  dgx15-ib
10.40.1.16    dgx16-ib.superpod.company.com  dgx16-ib
10.40.1.17    dgx17-ib.superpod.company.com  dgx17-ib
10.40.1.18    dgx18-ib.superpod.company.com  dgx18-ib
10.40.1.19    dgx19-ib.superpod.company.com  dgx19-ib
10.40.1.20    dgx20-ib.superpod.company.com  dgx20-ib
10.40.1.21    dgx21-ib.superpod.company.com  dgx21-ib
10.40.1.22    dgx22-ib.superpod.company.com  dgx22-ib
10.40.1.23    dgx23-ib.superpod.company.com  dgx23-ib
10.40.1.24    dgx24-ib.superpod.company.com  dgx24-ib
10.40.1.25    dgx25-ib.superpod.company.com  dgx25-ib
10.40.1.26    dgx26-ib.superpod.company.com  dgx26-ib
10.40.1.27    dgx27-ib.superpod.company.com  dgx27-ib
10.40.1.28    dgx28-ib.superpod.company.com  dgx28-ib
10.40.1.29    dgx29-ib.superpod.company.com  dgx29-ib
10.40.1.30    dgx30-ib.superpod.company.com  dgx30-ib
```

```
10.40.1.31    dgx31-ib.superpod.company.com  dgx31-ib
10.40.1.32    dgx32-ib.superpod.company.com  dgx32-ib
#
# NVIDIA DGX SuperPOD run:ai Highspeed InfiniBand storge network
#
10.40.1.84    runai01-ib.superpod.company.com  runai01-ib
10.40.1.85    runai02-ib.superpod.company.com  runai02-ib
```

B

IBM Storage Scale System NVIDIA DGX SuperPOD Solution Network Installation Worksheet

This appendix is an example of the Network Installation Worksheet that you must complete for the installation planning for the IBM Storage Scale System NVIDIA DGX SuperPOD solution.

This chapter describes the following topic:

- x86 Utility Node and ESS Management Server

x86 Utility Node and ESS Management Server

Figure B-1 shows the node ports for the 5149-23E ESS Management Server (EMS).

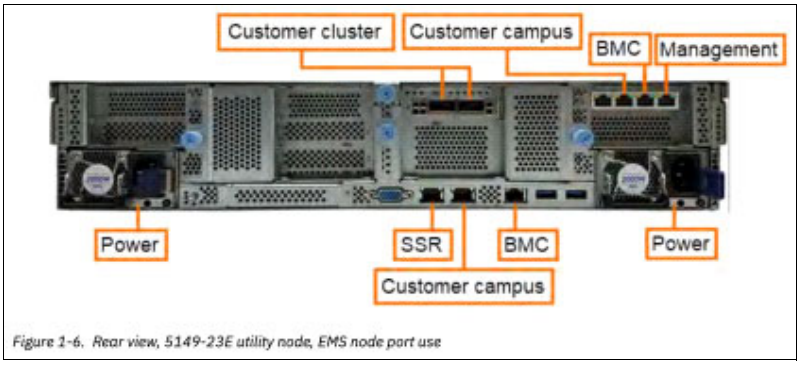


Figure B-1 5149-23E EMS node ports

In a single-CPU EMS node deployment, the PCIe adapter ports are used as follows:

- PCIe adapters:
  - PCIe Riser C: NVIDIA CX7 two-port Virtual Protocol Interconnect (VPI) InfiniBand/Ethernet high-speed adapter

    A high-speed adapter that is connected to the customer high-speed cluster network.
  - PCIe Riser D: Intel X710 4-port Ethernet adapter
- Three ports on this four-port adapter are used as follows (from right-to-left):
  - Solution management network connection to the solution management switch
  - Solution Baseboard Management Controller (BMC) service network connection to the solution management switch
  - Customer campus connection to a customer network switch
- In EMS node deployments, the onboard Ethernet ports are used as follows:
  - Onboard 10 Gb RJ45 Ethernet ports:
    - Left port: IBM Service Support Representative (IBM SSR) service laptop connection
    - Right port: Customer campus connection
  - Onboard 1 Gb RJ45 Ethernet port

    BMC service network connection to the solution management switch

Figure B-2 on page 121 shows the network connectivity of the example IBM Storage Scale System 6000.

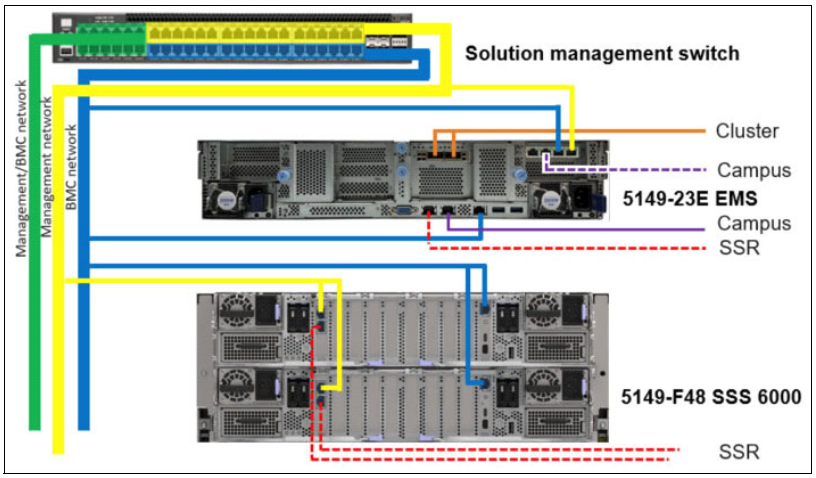


Figure B-2 Network connectivity of the IBM Storage Scale System 6000

The IBM Storage Scale Solution management switch is configured for compatibility with the IBM Storage Scale 3500. The VLAN tag is required only on IBM Storage Scale 3500 to route traffic on the BMC interface to the service network. This interface is connected to the “green” trunk ports on the switch. The default is 101.

IBM Storage Scale 3200 and 3500 VLAN Tag: VLAN TAG: 101

Best practices:

- Keep all management interfaces on 192.168.x.x/24 (netmask 255.255.255.0).
- Keep all BMC (HMC1) interfaces on the 10.0.0.x/24 (netmask 255.255.255.0).
- All IP addresses should be on the same subnet.

Example:

- All management interfaces are on 192.168.x.x/24.
- All BMC interfaces are on 10.0.0.x/24.

Domain name: `superpod.company.com`

GPFS cluster name

Here are the entries for the GPFS cluster name for our example:

- Storage cluster: `ISS.superpod.company.com`
- DGX client cluster: `dgx.superpod.company.com`

Table B-1 shows the serial numbers.

Table B-1 Serial numbers

Device	Serial number
EMS	5149-23E S/N 78P###X
IBM Storage Scale 6000-1	5149-F48 S/N 78##XXA
	5149-F48 S/N 78##XXB

- Default gateway:
- EMS campus network: 10.70.10.254

x86 Utility Node/EMS Ethernet IP addresses

Table B-2 shows the x86 Utility Node / EMS Ethernet IP addresses.

Table B-2 x86 Utility Node / EMS Ethernet IP addresses

Ethernet	Hostname	IP address	Netmask
EMS MGMT (Yellow; adapter)	ems01vm-mgmt	192.168.10.1	255.255.255.0 (/24)
EMS (virtual machine) BMC (Blue; adapter)	ems01vm-bmc	10.0.0.1	255.255.255.0 (/24)
EMS BMC (Blue; onboard)	ems01-bmc	10.0.0.101	255.255.255.0 (/24)
EMS External (virtual machine) (Dashed; Campus)	ems01vm	10.70.10.153	255.255.255.192 (/26)
EMS External (Solid; Campus)	ems01	10.7010.152	255.255.255.192 (/26)

IBM Storage Scale System 6000 Building Block 1 Ethernet IP addresses

Table B-3 shows the IBM Storage Scale System 6000 Building Block 1 Ethernet IP addresses.

Table B-3 IBM Storage Scale System 6000 Building Block 1 Ethernet IP addresses

Ethernet	Hostname	IP address	Netmask
IBM Storage Scale 6000 Canister 1 management Interface (left)	n01a	192.168.10.2	255.255.255.0 (/24)
IBM Storage Scale 6000 Canister 1 BMC Interface (left)	n01a-bmc	10.0.0.2	255.255.255.0 (/24)
IBM Storage Scale 6000 Canister 2 management Interface (right)	n01b	192.168.10.3	255.255.255.0 (/24)
IBM Storage Scale 6000 Canister 2 BMC Interface (right)	n01b-bmc	10.0.0.3	255.255.255.0 (/24)

IBM Storage Scale System 6000 Building Block 2 Ethernet IP addresses

Table B-4 shows the IBM Storage Scale System 6000 Building Block 2 Ethernet IP addresses.

Table B-4 IBM Storage Scale System 6000 Building Block 2 Ethernet IP addresses

Ethernet	Hostname	IP address	Netmask
IBM Storage Scale 6000 Canister 1 management Interface (left)	n02a	192.168.10.4	255.255.255.0 (/24)
IBM Storage Scale 6000 Canister 1 BMC Interface (left)	n02a-bmc	10.0.0.4	255.255.255.0 (/24)
IBM Storage Scale 6000 Canister 2 management Interface (right)	n02b	192.168.10.5	255.255.255.0 (/24)
IBM Storage Scale 6000 Canister 2 BMC Interface (right)	n02-bmc	10.0.0.5	255.255.255.0 (/24)

CES/Protocol nodes Ethernet IP addresses

Table B-5 shows the CES/Protocol nodes Ethernet IP addresses.

Table B-5 CES/Protocol nodes Ethernet IP addresses

Ethernet	Hostname	IP address	Netmask
CES/Protocol node 1 (mgmt/ces)	ces01	10.153.10.92	255.255.255.192 (/26)
CES/Protocol node 2 (mgmt/ces)	ces02	10.153.10.93	255.255.255.192 (/26)
CES/Protocol node 1 (bmc)	ces01-bmc	10.0.10.1	255.255.255.192 (/26)
CES/Protocol node 2 (bmc)	ces02-bmc	10.0.10.2	255.255.255.192 (/26)
CES Interface IP 1	cesvip01	10.153.10.94	255.255.255.192 (/26)
CES Interface IP 2	cesvip02	10.153.10.95	255.255.255.192 (/26)

Active File Management gateway nodes Ethernet IP addresses

Table B-6 shows the Active File Management gateway nodes Ethernet IP addresses.

Table B-6 Active File Management gateway nodes Ethernet IP addresses

Ethernet	Hostname	IP address	Netmask
Active File Management (AFM) gateway 1 (mgmt/afm)	afm01	10.5.100.201	255.255.255.0 (/24)
AFM gateway 2 (mgmt/afm)	afm02	10.5.100.199	255.255.255.0 (/24)
AFM gateway 3 (mgmt/afm)	afm03	10.5.100.200	255.255.255.0 (/24)
AFM gateway 4 (mgmt/afm)	afm04	10.5.100.198	255.255.255.0 (/24)
AFM gateway 1 (bmc)	afm01-bmc	10.0.20.1	255.255.255.0 (/24)
AFM gateway 1 (bmc)	afm02-bmc	10.0.20.2	255.255.255.0 (/24)
AFM gateway 1 (bmc)	afm03-bmc	10.0.20.3	255.255.255.0 (/24)
AFM gateway 1 (bmc)	afm04-bmc	10.0.20.4	255.255.255.0 (/24)

IBM Storage Scale System NVIDIA DGX SuperPOD High-Speed InfiniBand IP addresses

Table B-7 shows the IBM Storage Scale System NVIDIA DGX SuperPOD High-Speed InfiniBand IP addresses.

Table B-7 IBM Storage Scale System NVIDIA DGX SuperPOD High-Speed InfiniBand IP addresses

InfiniBand	Hostname	IP address	Netmask
EMS	ems01-ib	10.40.10.1	255.255.0.0 (/16)
IBM Storage Scale 6000 Building-Block 1 Canister 1 (left)	n01a-ib	10.40.10.2	255.255.0.0 (/16)
IBM Storage Scale 6-00 Building-Block 1 Canister 2 (right)	n01b-ib	10.40.10.3	255.255.0.0 (/16)
IBM Storage Scale 6000 Building-Block 2 Canister 1 (left)	n02a-ib	10.40.10.4	255.255.0.0 (/16)
IBM Storage Scale 6000 Building-Block 2 Canister 2 (right)	n02b-ib	10.40.10. 5	255.255.0.0 (/16)
CES/Protocol node 1	ces01-ib	10.40.10.10	255.255.0.0 (/16)
CES/Protocol node 2	ces02-ib	10.40.10.11	255.255.0.0 (/16)
AMF gateway 1	afm01-ib	10.40.10.12	255.255.0.0 (/16)
AFM gateway 2	afm02-ib	10.40.10.13	255.255.0.0 (/16)

InfiniBand	Hostname	IP address	Netmask
AFM gateway 3	afm03-ib	10.40.10.14	255.255.0.0 (/16)
AFM gateway 4	afm04-ib	10.40.10.15	255.255.0.0 (/16)

NVIDIA DGX SuperPOD Inband Ethernet IP addresses

Table B-8 shows the NVIDIA DGX SuperPOD Inband Ethernet IP addresses.

Table B-8 NVIDIA DGX SuperPOD Inband Ethernet IP addresses

Ethernet	Hostname	IP address	Netmask
slogin-01	slogin01	10.153.10.81	255.255.255.192(/26)
slogin-02	slogin02	10.153.10.82	255.255.255.192(/26)
slogin-03	slogin03	10.153.10.83	255.255.255.192(/26)
DGX-01	dgx01	10.153.10.1	255.255.255.192(/26)
DGX-02	dgx02	10.153.10.2	255.255.255.192(/26)
DGX-03	dgx03	10.153.10.3	255.255.255.192(/26)
DGX-04	dgx04	10.153.10.4	255.255.255.192(/26)
DGX-05	dgx05	10.153.10.5	255.255.255.192(/26)
DGX-06	dgx06	10.153.10.6	255.255.255.192(/26)
DGX-07	dgx07	10.153.10.7	255.255.255.192(/26)
DGX-08	dgx08	10.153.10.8	255.255.255.192(/26)
DGX-09	dgx09	10.153.10.9	255.255.255.192(/26)
DGX-10	dgx10	10.153.10.10	255.255.255.192(/26)
DGX-11	dgx11	10.153.10.11	255.255.255.192(/26)
DGX-12	dgx12	10.153.10.12	255.255.255.192(/26)
DGX-13	dgx13	10.153.10.13	255.255.255.192(/26)
DGX-14	dgx14	10.153.10.14	255.255.255.192(/26)
DGX-15	dgx15	10.153.10.15	255.255.255.192(/26)
DGX-16	dgx16	10.153.10.16	255.255.255.192(/26)
DGX-17	dgx17	10.153.10.17	255.255.255.192(/26)
DGX-18	dgx18	10.153.10.18	255.255.255.192(/26)
DGX-19	dgx19	10.153.10.19	255.255.255.192(/26)
DGX-20	dgx20	10.153.10.20	255.255.255.192(/26)
DGX-21	dgx21	10.153.10.21	255.255.255.192(/26)
DGX-22	dgx22	10.153.10.22	255.255.255.192(/26)
DGX-23	dgx23	10.153.10.23	255.255.255.192(/26)
DGX-24	dgx24	10.153.10.24	255.255.255.192(/26)

Ethernet	Hostname	IP address	Netmask
DGX-25	dgx25	10.153.10.25	255.255.255.192(/26)
DGX-26	dgx26	10.153.10.26	255.255.255.192(/26)
DGX-27	dgx27	10.153.10.27	255.255.255.192(/26)
DGX-28	dgx28	10.153.10.28	255.255.255.192(/26)
DGX-29	dgx29	10.153.10.29	255.255.255.192(/26)
DGX-30	dgx30	10.153.10.30	255.255.255.192(/26)
DGX-31	dgx31	10.153.10.31	255.255.255.192(/26)
DGX-32	dgx32	10.153.10.32	255.255.255.192(/26)
run-ai-01	runai01	10.153.10.84	255.255.255.192(/26)
run-ai-02	runai02	10.153.10.85	255.255.255.192(/26)

NVIDIA DGX SuperPOD High-Speed Storage InfiniBand IP addresses

Table B-9 shows the NVIDIA DGX SuperPOD High-Speed Storage InfiniBand IP addresses.

Table B-9 NVIDIA DGX SuperPOD High-Speed Storage InfiniBand IP addresses

InfiniBand (HCA-10: mlx5_7) slot-4 (port-1)	Hostname	IP address	Netmask
slogin-01	slogin01-ib	10.40.1.81	255.255.0.0 (/16)
slogin-02	slogin02-ib	10.40.1.82	255.255.0.0 (/16)
slogin-03	slogin03-ib	10.40.1.83	255.255.0.0 (/16)
DGX-01	dgx01-ib	10.40.1.1	255.255.0.0 (/16)
DGX-02	dgx02-ib	10.40.1.2	255.255.0.0 (/16)
DGX-03	dgx03-ib	10.40.1.3	255.255.0.0 (/16)
DGX-04	dgx04-ib	10.40.1.4	255.255.0.0 (/16)
DGX-05	dgx05-ib	10.40.1.5	255.255.0.0 (/16)
DGX-06	dgx06-ib	10.40.1.6	255.255.0.0 (/16)
DGX-07	dgx07-ib	10.40.1.7	255.255.0.0 (/16)
DGX-08	dgx08-ib	10.40.1.8	255.255.0.0 (/16)
DGX-09	dgx09-ib	10.40.1.9	255.255.0.0 (/16)
DGX-10	dgx10-ib	10.40.1.10	255.255.0.0 (/16)

InfiniBand (HCA-10: mlx5_7) slot-4 (port-1)	Hostname	IP address	Netmask
DGX-11	dgx11-ib	10.40.1.11	255.255.0.0 (/16)
DGX-12	dgx12-ib	10.40.1.12	255.255.0.0 (/16)
DGX-13	dgx13-ib	10.40.1.13	255.255.0.0 (/16)
DGX-14	dgx14-ib	10.40.1.14	255.255.0.0 (/16)
DGX-15	dgx15-ib	10.40.1.15	255.255.0.0 (/16)
DGX-16	dgx16-ib	10.40.1.16	255.255.0.0 (/16)
DGX-17	dgx17-ib	10.40.1.17	255.255.0.0 (/16)
DGX-18	dgx18-ib	10.40.1.18	255.255.0.0 (/16)
DGX-19	dgx19-ib	10.40.1.19	255.255.0.0 (/16)
DGX-20	dgx20ib	10.40.1.20	255.255.0.0 (/16)
DGX-21	dgx21-ib	10.40.1.21	255.255.0.0 (/16)
DGX-22	dgx22-ib	10.40.1.22	255.255.0.0 (/16)
DGX-23	dgx23-ib	10.40.1.23	255.255.0.0 (/16)
DGX-24	dgx24ib	10.40.1.24	255.255.0.0 (/16)
DGX-25	dgx25-ib	10.40.1.25	255.255.0.0 (/16)
DGX-26	dgx26-ib	10.40.1.26	255.255.0.0 (/16)
DGX-27	dgx27-ib	10.40.1.27	255.255.0.0 (/16)
DGX-28	dgx28-ib	10.40.1.28	255.255.0.0 (/16)
DGX-29	dgx29-ib	10.40.1.29	255.255.0.0 (/16)
DGX-30	dgx30-ib	10.40.1.30	255.255.0.0 (/16)
DGX-31	dgx31-ib	10.40.1.31	255.255.0.0 (/16)
DGX-31	dgx32-ib	10.40.1.32	255.255.0.0 (/16)
run-ai-01	runai01-ib	10.40.1.84	255.255.0.0 (/16)
run-ai-02	runai02-ib	10.40.1.85	255.255.0.0 (/16)

Abbreviations and acronyms

AFM	Active File Management
AI	artificial intelligence
API	application programming interface
BCM	Base Command Manager
BMC	Baseboard Management Controller
CCR	Cluster Configuration Repository
CES	Cluster Export Services
CIFS	Common Internet File System
CNSA	Container Native Storage Access
COS	Container Orchestration Systems
CPU	central processing unit
CSI	Container Storage Interface
DAS	Direct Attached Storage
DAT	Data Acceleration Tier
DLM	distributed lock manager
DMA	direct memory access
DR	disaster recovery
ECN	Explicit Congestion Notification
EMS	ESS Management Server
ESS	Elastic Storage System
ETL	Extract, Transform, and Load
FCM4	FlashCore Module 4
FQDN	fully qualified domain name
GBps	gigabytes per second
GDS	GPUDirect Storage
GPFS	General Parallel File System
GPU	graphics processing unit
HA	high availability
HBA	Host Bus Adapter
HDD	hard disk drive
HDFS	Hadoop Distributed File System
HPC	high-performance computing
IBM	International Business Machines Corporation
IBM SSR	IBM Service Support Representative
IDUG	International Db2 User's Group
IEEE	Institute of Electrical and Electronics Engineers
ILM	Information Lifecycle Management
IOPS	input/output operations per second
IOR	Interleaved or Random
IoT	Internet of Things
IP	internet protocol
IPoIB	IP over InfiniBand
iSCSI	Internet Small Computer System Interface
IT	information technology
JBOD	Just a Bunch of Disks
k8S	Kubernetes
ML	machine learning
MPI	Message Passing Interface
MROT	Multi-Rail over TCP
MTS	Mellanox Tools Services
NAS	Network Attached Storage
NFS	Network File System
NIC	network interface card
NSD	Network Shared Disk
NVMe	Non-Volatile Memory Express
NVMe-oF	Non-Volatile Memory Express over Fabrics
OFED	OpenFabrics Enterprise Distribution
OS	operating system
OSFP	octal small form factor pluggable
PBe	petabytes-effective
PDU	power distribution unit
PFC	Priority Flow Control
POSIX	Portable Operating System Interface
PV	persistent volume
PVC	persistent volume claims
QoS	quality of service
RA	Reference Architecture
RAID	Redundant Array of Independent Disks
RDMA	Remote Direct Memory Access
REST	Representational State Transfer
RHEL	Red Hat Enterprise Linux
RoCE	RDMA over Converged Ethernet
S3	Simple Storage Service
SAN	Storage Area Network

SATA	Serial Advanced Technology Attachment
SDS	software-defined storage
SED	self-encrypted drive
SIEM	security information and event management
SL	Service Level
SLURM	Simple Linux Utility for Resource Management
SMB	Server Message Block
SQL	Structured Query Language
SU	Scalable Unit
TCP	Transmission Control Protocol
UFM	NVIDIA Unified Fabric Manager
VFS	Virtual File System
VL	virtual lane
VPI	Virtual Protocol Interconnect
WAN	wide area network
YB	yottabyte

REDP-5746-00

ISBN 0738462039

Printed in U.S.A.