Redbooks
ibm.com/redbooks

Implementation Guide for IBM Storage Scale System 6000

Phillip Gerrard
Luis Bolinches
Bruce Brown
Puneet Chaudhary
Mika Heino
Nikhil Khandelwal
John Lewars
Chris Maestas
Roger E. Sanders
Lindsay Todd
Farida Yaragatti
Ricardo Zamora Ruvalcaba

Storage

IBM

Redbooks

IBM Redbooks

Storage Scale 6000 Implementation guide

August 2025

SG24-8566-00

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

First Edition (August 2025)

This edition applies to Version 5.2.3 of IBM Storage Scale and 6.2.3 of IBM Storage Scale System 6000.

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®
DB2®
developerWorks®
IBM®
IBM Cloud®
IBM FlashCore®
IBM Security®
IBM Spectrum®
IBM watsonx®
Interconnect®
POWER®
Power9®
Redbooks®
Redbooks (logo)  ®
watsonx®

The following terms are trademarks of other companies:

Intel, 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.

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

Ansible, OpenShift, Red Hat, 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. And unlike traditional applications that work with structured data 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 leverage AI to unlock the value of their data – wherever it might reside. And information technology (IT) leaders face many new challenges including:

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

Addressing these challenges requires specialized software and hardware, like:

- IBM Storage Scale: Software-defined file and object storage for both structured and unstructured data.
- IBM Storage Scale System 3500: A hardware implementation of IBM Storage Scale software, designed for organizations requiring enterprise-ready, entry-level or mid-level storage system.

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.

Authors

This book was produced by a team of specialists from around the world working for IBM Redbooks.

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.

Luis Bolinches brings over 20 years of enterprise infrastructure expertise from IBM®, where he was part of Storage Scale Server development, working with GPFS since version 3.4. Now at WorldQuant LLC, he applies his deep knowledge of high-performance computing, Linux, and networking to support quantitative finance infrastructure. His experience spans deployments across 55+ countries in diverse industries including telecommunications, government, oil & gas, financial services, defense, high-energy physics, manufacturing, and healthcare. He has architected and deployed mission-critical systems and supercomputers handling hundreds of petabytes, optimizing for multi-terabyte-per-second performance. Author of 14 IBM

Redbooks® publications and regular conference presenter, Luis excels at designing scalable solutions for demanding computational environments.

Bruce Brown is a Senior IT Engineer at Jeskell Systems, LLC with over 37 years of experience designing, delivering and supporting enterprise and embedded IT solutions. In his current role he is responsible for architecting, implementing and testing IBM Storage Scale solutions. Prior to joining Jeskell Bruce was an IT Architect at IBM focused on cyber security and HPC cluster technologies. He currently holds 24 certifications with IBM and was a Co-Author of Wiley's "Hadoop for Dummies" publication.

Puneet Chaudhary is a Technical Solutions Architect who works with the IBM ESS and IBM Spectrum® Scale solutions. He has worked with IBM GPFS, now IBM Spectrum Scale, for many years.

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

Mika Heino is a senior IT Management Consultant with IBM Lab Services working in Finland for local and international IBM accounts. He has a degree in Telecommunications and Computer Science from Turku University of Applied Sciences. Mika has 25 years experience with Linux, IBM AIX® and IBM i, and server virtualization for both Intel and IBM POWER®. He is Master Certified Technical Specialist for Storage Systems with more than 15 years of experience with storage area networks (SANs), IBM Storage Systems, and storage virtualization.

Nikhil Khandelwal Software Engineer IBM Storage Scale.

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.

Roger E. Sanders is a Principal, Learning Content Development – Data, AI, and Storage specialist in the United States. He is the author of 25 books on Db2®, one book on Open Database Connectivity (ODBC), 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 IBM's developerWorks® website, presented at a variety of 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/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 developerWorks Master Author, Level 2 (for his contributions to the IBM developerWorks community); and in 2021 he was recognized as an IBM Redbooks Platinum Author.

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.

Farida Yaragatti is a Senior Software Engineer at IBM India. She has a BE, Electronics and Communication from Karnataka University, India and has 12 years of experience in the Software Testing field. She is part of manual and automation testing for IBM Spectrum Scale and IBM Elastic Storage System (ESS) deployment as a Senior Tester. Farida worked at IBM for over five years and previously held roles within the IBM Platform Computing and IBM Smart analytics system (ISAS) testing teams. She has strong engineering professional skills in Software deployment testing, including automation using various scripting technologies, such as Python, shell scripting, Robot framework, Ansible, and Linux.

Ricardo D. Zamora Ruvalcaba is a Software Engineer at IBM Guadalajara, Mexico and holds a bachelor's degree in Mechatronics. He started his career in IBM Spectrum Scale as a Test Automation Engineer and eventually moved to the ESS deployment development team where he constantly implements new tools and technologies. As the focal point for Manufacturing and a Technical Advocate for IBM ESS customers around the globe, he has strong automation engineering skills in various scripting technologies, such as Python, Ansible, shell scripting, Jenkins, the Robot framework, and Linux

Thanks to the following people for their contributions to previous IBM Storage Scale Redbooks releases:

Monika Balichetty, Pidad D'Souza, Wesley Jones, Stieg Klein, Laxmi Rajmane, Van Smith, Ratan Swami, Jay Vaddi, and Olaf Weiser,

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 books to be as helpful as possible. Send us your comments about this book 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

This chapter provides an overview of IBM Storage Scale, describes the characteristics of IBM Storage Scale RAID (Redundant Array of Independent Disks), and introduces the IBM Storage Scale System 6000. It covers the following topics:

1.1 IBM Storage Scale

IBM Storage Scale abbreviated as ISS, formerly known as IBM Spectrum Scale and General Parallel File System (GPFS), is a file and object storage solution that enables interdependent clients to store and access large volumes of shared data concurrently, over a low-latency network. It utilizes a global namespace to facilitate data access. And it enables data (and metadata) to be read from or written to multiple storage devices, using multiple input/output (I/O) paths. Consequently, it enables organizations to build an enterprise-resilient, global data platform for big data, analytics, HPC (High Performance Computing), AI, and data-intensive technical computing.

First released in 1998 (as GPFS), Storage Scale is based on a massively parallel file system and can be deployed on multiple hardware platforms including x86, IBM Power, IBM zSystem mainframes, ARM-based Portable Operating System Interface (POSIX) clients, virtual machines, and Kubernetes. It also provides the Global Data Platform option for IBM Fusion (formerly known as IBM Storage Fusion) and the IBM Fusion Hyperconverged Infrastructure (HCI) appliance, both of which enable organizations to quickly and flexibly deploy Red Hat OpenShift applications and IBM watsonx®.

Tip: POSIX (Portable Operating System Interface) is a set of standard operating system interfaces, based on the UNIX operating system (OS), that are designed to maintain compatibility among different OSs. The most recent POSIX specification – Institute of Electrical and Electronics Engineers (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 be capable of running on, any operating system that adheres to POSIX standards.

IBM Storage Scale can be deployed as software-defined storage, as an integrated hardware and software solution (i.e., IBM Elastic Storage System 3500 or IBM Storage Scale System 6000), or as a cloud service. It can also 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 be connected to servers as if they were local disks), and Non-Volatile Memory express over Fabrics (NVMe-oF). Additionally it can be used to create a common global namespace with data protection that's provided through erasure coding or replication. (In this case, namespace is a term that is used to describe the tree structure of a file system; a directory is an example of namespace in an operating system.)

IBM Storage Scale is built to deliver four key capabilities:

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

1.1.1 Data Access

IBM Storage Scale's data access services enable organizations to maintain a "single source of truth" for all their data, even when multiple workloads are accessing the same datasets at the same time, across a variety of sites. It does this by providing the ability to work with

multiple data protocols, simultaneously. These include the POSIX and Network File System (NFS) protocols used by Unix and Linux, and Server Message Block (SMB) protocols used by Windows, and the Simple Storage Service (S3) object-based protocol used by cloud providers like Amazon Web Services. IBM Storage Scale also supports the Apache Hadoop Distributed File System (HDFS); IBM Storage Scale System 6000 also supports NVIDIA GPUDirect Storage (GDS) technology.

In addition to providing robust multiprotocol data access, Storage Scale's architecture is designed to provide linear scaling of both performance and capacity. As a result, it can deliver sub-millisecond latency, hundreds of gigabytes per second (GB/s) throughput, and tens of millions of input/output operations per second (IOPS) while addressing datasets in the exabyte range. This helps eliminate data silos since it enables multiple application programming interfaces (APIs) to access the same data, at the same time. It also enables AI and analytics workloads to use the output from one application to drive results to another.

1.1.2 Data Acceleration

Because IBM Storage Scale is a general parallel file system (GPFS) or “parallel clustered file system”, it can provide concurrent access to a single file system, or set of file systems, from multiple nodes and storage devices. Parallel file systems are the backbone of high-performance computing because they distribute data across multiple storage devices and servers (nodes), enabling lightning-fast input/output (I/O) operations for massive computational tasks.

Key benefits parallel file systems offer include:

- Parallel data access - Multiple clients can read from and write to a single file simultaneously, without creating performance bottlenecks.
- Distributed storage - Files are distributed across multiple storage nodes, allowing for faster data access.
- High throughput - Parallel file systems are optimized for high data transfer rates.
- Low latency - By design, parallel file systems minimize data access delays.
- High scalability - Storage capacity and bandwidth can be easily scaled by adding more nodes to a cluster.
- Data redundancy and replication - Data can be replicated across multiple nodes to ensure data availability and disaster recovery.
- Flexibility - Parallel file systems can be tailored to specific workload requirements; techniques like client-side caching and load balancing ensure efficient data access and processing.

In addition, IBM Storage Scale System 6000 supports the NVIDIA GPUDirect Storage (GDS) protocol. GDS enables a direct data path for direct memory access (DMA) transfers between graphics processing unit (GPU) memory and storage, avoiding the need for a "bounce buffer" through the central processing unit (CPU). This direct path ensures that the GPU, rather than the CPU, has the first and last touch of data that moves in or out of storage. 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.

1.1.3 Data Abstraction

Data abstraction refers to the process of simplifying complex data systems by hiding the details of their underlying implementation and presenting only what’s considered essential

information to applications and users. To this end, IBM Storage Scale provides transparent data abstraction with active file and object management by separating the details of how data is stored while providing interfaces that can be used to access and manipulate it. This means IBM Storage Scale can provide connectivity from multiple data sources and multiple locations to bring together data from both IBM and non-IBM storage environments.

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. Organizations can rapidly grow their storage infrastructure by adding storage capabilities where they can be most efficiently deployed, without regard to geography. Moreover, they can protect their existing information technology (IT) infrastructure investments and lower costs moving forward by using IBM Storage Scale to create an open ecosystem of storage options that leverage multi-vendor and multi-cloud resources.

1.1.4 Data assurance

When it comes to data Assurance, the primary focus of IBM Storage Scale is on security, data governance, and data protection. To enhance data resilience, a single 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 via a Wide Area Network (WAN) and file systems in the cluster are available and actively usable at both sites concurrently. These file systems can utilize “failure group” replication to ensure both sites have a current replica of all the data, so that if one site (or link to the site) fails, the other site always remains active and accessible. The result is a highly available file system with active-active synchronous replication.

IBM Storage Scale also includes IBM’s incredibly powerful protective Safeguarded Copy functionality, which can be used to create immutable snapshots of data on a regular schedule so that operational data can be rapidly recovered if a cyber-attack or other potential loss of data event occurs.

1.2 The basic structure of IBM Storage Scale

IBM Storage Scale presents itself as a clustered file system that is defined across multiple nodes (servers), each of which are running three basic components (that enable a node to participate in a cluster): administration commands, a kernel extension, and a multithreaded daemon. Interaction between nodes at the file system level is limited to the locks and control flows that are needed to maintain data and metadata integrity in the cluster environment.

The administration commands are programs and scripts that control the configuration and operation of IBM Storage Scale. They typically begin with the letters “mm.” (For example, the command used to display file system attributes is "mmlsfs.") The administration commands may be issued from appropriately configured nodes in a cluster - if the execution of a command requires tasks to be performed on other nodes, the requests 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 will then either fulfill the calls using resources that are already available, or it will send a message to the multithreaded daemon informing it to complete the request.

The multithreaded daemon performs all I/O operations and buffer management needed for IBM Storage Scale, including "read-ahead" for sequential and strided reads (reads of data not contiguously placed) and "write-behind" for non-synchronous writes. Internally, I/O operations are protected by IBM Storage Scale's distributed lock management using tokens to ensure single system consistency of data operations across all nodes in the cluster. Along with managing local I/O, the multithreaded daemon communicates with instances of the daemon on other nodes to coordinate configuration changes, data recovery operations, and parallel updates of internal data structures.

Figure 1-1 illustrates how a simple IBM Storage Scale cluster might look; it also identifies some of the key features and functionality IBM Storage Scale provides.

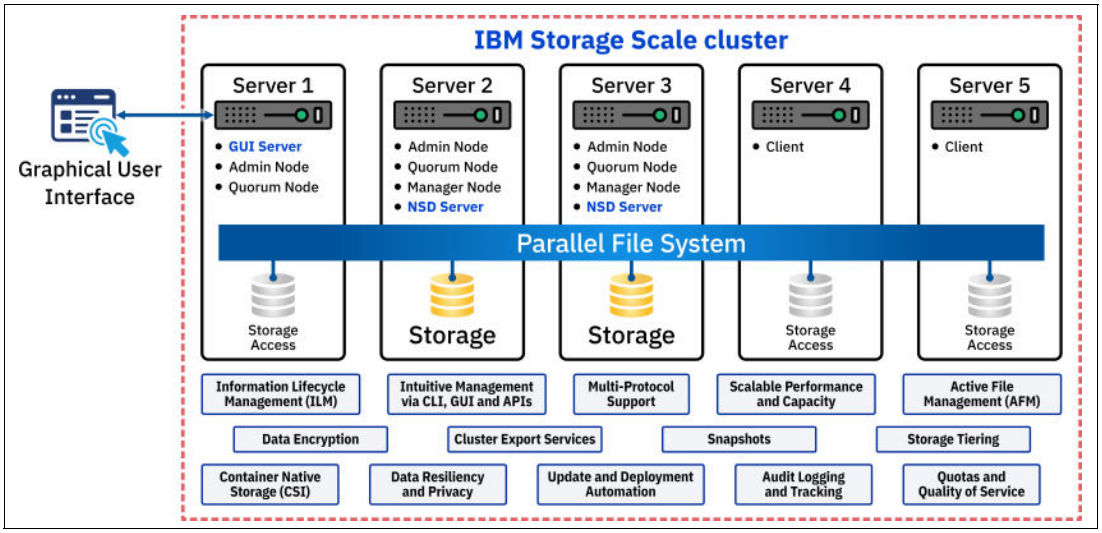


Figure 1-1 Example of a simple IBM Storage Scale cluster

Figure 1-1 depicts an IBM Storage Scale cluster that consists of five servers (nodes). The two nodes that have been assigned the "NSD Server" role are connected to Network Shared Disk (NSD) storage; the remaining nodes in the cluster have access to the storage via the IBM Storage Scale parallel file system.

As the illustration in Figure 1 shows, each node in an IBM Storage Scale cluster can be assigned one or more roles, depending on the service(s) it is expected to perform:

- GUI Server: A node in a cluster that the IBM Storage Scale Management Interface runs on. The IBM Storage Scale Management Interface provides both graphical user interface (GUI) and REpresentational State Transfer (RESTful) API access to an IBM Storage Scale cluster.
- Administrator Node: A node in a cluster that can be used to execute IBM Storage Scale administration commands (as well as execute administration commands on any other node in the cluster, without the need of a password).
- Quorum Node: A node in a cluster that participates in a quorum for the cluster. A quorum is a mechanism IBM Storage Scale uses to maintain data consistency even when a node in a cluster fails. (The quorum operates on a simple majority rule, meaning that a majority of quorum nodes in the cluster must be accessible before any node in the cluster can access a file system. One of the quorum nodes is elected by the voting quorum to serve as the "cluster manager", tracking the active members of the Scale cluster and controlling which nodes are allowed access to the cluster's storage devices.)

- 1.2, “The basic structure of IBM Storage Scale” on page 14 Node: A node in a cluster that is part of the node pool from which file system managers and token managers can be selected to perform various internal tasks.
- Client Node: A node in a cluster that accesses the cluster’s storage (typically to run applications or compute jobs) but is not running any of the cluster’s Scale services.
- CES Node: A node in a cluster that provides Cluster Export Services (CES), which is used to enable Server Message Block (SMB), Network File System (NFS), or Object access to data in an IBM Storage Scale file system. (These are often called “protocol nodes”.)
- NSD Server: A node in cluster that makes storage devices, internal or attached to an external storage system, using Fibre Channel, SAS, or a similar interconnect available to other cluster nodes that might not otherwise be able to access that storage device.
- AFM Gateway Node: A node in an Active File Management (AFM) cache cluster that provides a connection to a “home” cluster or storage service.

It should be noted that the 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 via one or more designated NSD servers (nodes). Up to eight NSD servers can be specified for each NSD, if one server fails, the next server on the list will automatically take control.

1.3 Key IBM Storage Scale components/functionality

In addition to the data access, data acceleration, data abstraction, and data assurance capabilities IBM Storage Scale provides, there are some key components and functionality IBM Storage Scale supplies that set it apart from the competition. These components/functionalities include:

- Cluster Export Services (CES)
- Container Storage Interface (CSI)
- Active File Management (AFM) and AFM-Disaster Recovery (AFM-DR)
- Information Lifecycle Management (ILM)

1.3.1 Cluster Export Services (CES)

Earlier, it was noted that IBM Storage Scale is capable of providing a global namespace because it supports a variety of data access protocols that can be used to bring different types of storage together. Cluster Export Services (CES) is the functionality that enables the use of NFS, S3, SMB, and HDFS protocols. Because CES has specific hardware and software requirements, it's code must be installed on nodes in an IBM Storage Scale cluster that are designated to run the CES software stack. (As noted earlier, such 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 to the node.

The CES infrastructure is responsible for:

- Managing the setup for high-availability (HA) clustering that is used by the NFS, S3, SMB, or HDFS protocols. Irrespective 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. (The addresses are considered "floating" because each address can move independently among all designated CES nodes.) In the event of a CES node failure, accessibility to all

remotely mounted storage is maintained as IP addresses are automatically moved from a failed CES node to a healthy one.

- Monitoring the health of the supported protocols on the CES nodes and raising events or alerts during failures. Monitoring ensures that CES IP addresses are assigned to the appropriate node and that processes that implement the services in the CES cluster are running correctly. Upon 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 the remotely mounted storage provided via the supported protocols by including failover and failback of these addresses in the event a CES node failure occurs. It should be noted that to avoid impacting the clients of other protocols during an HDFS failover, clients using NFS, S3, and SMB protocols should not share IP address with clients used by the HDFS service.

Common use cases for CES include:

- 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, etc.), 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 upon the application. For example, batch processing or grid computing.

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

1.3.2 Container Storage Interface (CSI)

Although containers were originally conceived of as having storage that was stateless like containers themselves, it soon became apparent that containerized applications needed to retain data for longer periods. So, a variety of ways of achieving persistent storage for containers – largely represented by Docker and the container orchestrator Kubernetes – were developed. The most popular method used was a Kubernetes volume plugin, which extended the Kubernetes volume interface to support a block and/or file storage system.

Initially, Kubernetes volume plugins were “in-tree,” meaning their code was part of the core Kubernetes code and shipped with the core Kubernetes binaries. As a result, vendors wishing 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 binaries. Furthermore, third-party code was often difficult (and in some cases impossible) for Kubernetes developers to test and maintain. A method for decoupling persistent storage development efforts from core cluster management tooling was sorely needed. This led to the creation of the Container Storage Interface (CSI).

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. Essentially, CSI is an API specification that enables developers to build custom drivers that handle the provisioning, attaching, and mounting of volumes in containerized workloads. As long as a driver correctly implements the CSI API specification, it

can be used in any supported COS. This gives Kubernetes users more options for storage and makes the system more secure and reliable.

The IBM Storage Scale CSI driver allows IBM Storage Scale to be used as persistent storage for stateful applications running in Kubernetes clusters. With this driver, Kubernetes persistent volumes (PVs) can be provisioned from IBM Storage Scale storage, making it possible for containerized applications and stateful microservices like containerized databases (MongoDB, PostgreSQL, etc.) to take advantage of everything IBM Storage Scale has to offer.

The IBM Storage Scale CSI driver provides the following functionality:

- Static provisioning: The ability to use existing directories and filesets as persistent volumes.
- Lightweight dynamic provisioning: The ability to create directory-based volumes, dynamically.
- Fileset-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 easier deployment, upgrade, and cleanup).
- IBM Storage Scale volume access mode support: RWX (ReadWriteMany) and RWO (ReadWriteOnce)

The IBM Storage Scale CSI driver makes REST API calls on an IBM Storage Scale storage system to perform provisioning requests.

1.3.3 Active File Management (AFM)

Active File Management (AFM) enables data sharing across storage clusters, even if the network used is unreliable or has high latency. Essentially, it’s 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's physically stored at a remote location in such a way that it appears to be local.

As has already been noted, 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 - those clusters, in turn can mount and access the file systems exported. And in many cases, IBM Storage Scale clients using POSIX have very high-performance access to the remote-mounted file systems. However, the greater the distance between clusters, the greater the latency. AFM reduces this latency by caching remote data, locally.

To illustrate, suppose an organization has an IBM Storage Scale compute cluster with a file system named “/scale/fs1” at its headquarters (main site) that is used to store company data. They also have a manufacturing facility (branch site) that has its own IBM Storage Scale compute cluster with a file system named ”/scale/fs2” that receives data being streamed from Internet of Things (IoT) sensors placed on equipment throughout the facility. Because the sites are several hundred kilometers apart, the compute clusters are connected to each other via a wide-area network (WAN). So, if the branch site needs data from the "/scale/fs1" file system, the latency between the sites can be problematic.

Using AFM, the organization can create cache filesets at the branch site to locally cache the parts of the "/scale/fs1" file system it needs, using NFS (version 3 or 4) or NSD/GPFS as the

transport protocol. (Filesets 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 appropriate cache filesets have been defined, everything needed in the "/scale/fs1" file system will be visible and accessible to the remote cluster at the branch site. However, only files that are actively being used will consume space in the branch site's cache. That's because file metadata is retrieved instantaneously, but file data is retrieved "on demand". This means that data is only requested and brought into the branch site's cache when an application running at the branch site attempts to work with a file that is stored on the main site.

The one limitation – which typically is not that important – is that AFM's semantics are designed to "eventually be consistent" across all sites (as opposed to "immediately being consistent"). This is a necessary concession to circumvent the latency a WAN would otherwise impose on lock management. AFM is often called asynchronous replication, but this is a bit misleading – AFM is really a data caching mechanism. In most cases, AFM checks periodically to ensure cached data is not "stale" and refreshes it, if appropriate. However, there are settings available to control when (or if) cache "revalidation" takes place.

While AFM can be used to create associations between two (or more) IBM Storage Scale clusters, it can also be used to create associations between IBM Storage Scale clusters and NFS data sources or S3 cloud object storage buckets. This allows users to implement a single namespace view across multiple sites around the world, as well as work with data stored on non-IBM storage and/or at non-IBM cloud providers.

Figure 1-2 illustrates how AFM can be used to create associations between two IBM Storage Scale clusters, a NFS data source, and a S3 cloud object bucket.

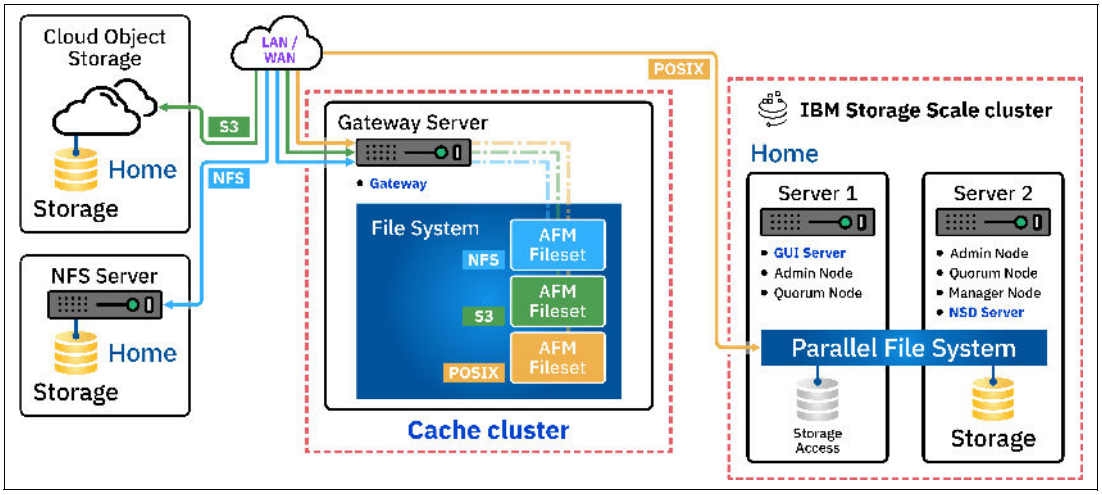


Figure 1-2 Example AFM cache cluster configuration

As Figure 1-2 shows, an IBM Storage Scale cluster that contains AFM filesets is called a "cache cluster." A cache cluster has an AFM relationship with a remote site called the "home," where either the cache or the home can access the data, with the home always being the ultimate destination. AFM constantly maintains an active relationship between the cache and the home; configuration is managed per fileset, resulting in a modular, scalable architecture that can support billions of files and petabytes of data. As Figure 2 illustrates, each AFM-enabled fileset is associated with a single home (path). However, multiple AFM filesets can reside in a single file system, each caching a different target. "Gateway" nodes (servers) are used to connect AFM cache clusters to the appropriate home; each AFM-enabled fileset uses a designated primary gateway node to establish this connection and will fail over to other gateway nodes (assuming other cluster gateway nodes have been defined) if necessary.

1.3.4 Active File Management - Disaster Recovery (AFM-DR)

One special use case for AFM is disaster recovery (DR). However, AFM alone does not contain some of the features needed to deliver a true (DR) solution. That's where Active File Management-Disaster Recovery (AFM-DR) comes into play.

AFM-DR provides IBM Storage Scale users with a fileset-level, asynchronous replication DR solution that is capable of:

- Providing business stability during a disaster
- Restoring business stability after a disaster
- Enduring multiple disasters
- Minimizing data loss due to a disaster

AFM-DR utilizes a one-to-one, active-passive model and is represented by two sites: the "primary" production site and the "secondary" DR site. The idea is that the primary site uses primary filesets as caches and relies on normal AFM transport mechanisms to update a corresponding secondary fileset at the secondary site. The primary fileset has read/write access to the data; the secondary fileset is kept read-only to prevent accidental or malicious changes. (Only updates made to the primary fileset should cause changes to the secondary fileset to be made.) Data written to the primary fileset is continuously replicated to the secondary fileset, asynchronously.

The normal AFM updating mechanism can be used to populate the "read-only" fileset on the secondary site. However, because this can take a significant amount of time to complete, an alternative is to “truck” data – using a non-AFM mechanism like rsync or tape – to pre-populate the secondary fileset and establish the AFM relationship afterwards. (Both the primary and the secondary must be an IBM Storage Scale cluster.) From that point on, AFM will ensure that the secondary fileset is always kept consistent with the primary.

In the event the primary fileset/cluster fails, replication ceases and the secondary fileset can be turned into a fully functional primary site by performing a failover operation. This causes the secondary fileset to be changed to the “acting primary fileset”, in which case it is given read/write access to the data at the secondary site. Then all applications must be reconfigured to point to the secondary cluster to ensure continuity. (Reconfiguring applications is outside the scope of IBM Storage Scale, or indeed of any storage solution alone. AFM-DR is only responsible for keeping the DR copy of the data consistent and making it available, when needed, as well as keeping track of all changes made.)

In a true DR scenario, an organization may run their business for days, or weeks, or even months on the secondary (failover) fileset. But eventually, the original primary fileset should come back on-line. When that happens, the organization has two options. One is to reverse the primary and secondary roles so the acting primary (original secondary) replicates to the new secondary (original primary). Often, this is the fastest way to restore business stability after a disaster.

For small clusters with small amounts of storage, a second option is for the original primary to ask the acting primary (original secondary) for any updates it missed. (This can happen repeatedly.) Then, when both filesets are fully synchronized, the acting primary can failback and the original replication relationship can be restored. (Applications will need to be reconfigured to use the original primary once more.)

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 fileset can be rolled back to the previous snapshot. This is useful when there is a risk that

out-of-order file updates could pose a problem to application. Peer snapshots are guaranteed to be “crash consistent”.

1.3.5 Information Lifecycle Management (ILM)

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. Using ILM, IBM Storage Scale administrators can manage sets of files and pools of storage, as well as automate the movement of data from one pool to another. A "storage pool" is a collection of NSDs (recall, an NSD is a disk or RAID volume managed by Scale). Storage pools enable users to create storage tiers by grouping storage devices together, based on common size, 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 while another might consist of disk controllers that host a large set of economical, lower performing Serial Advanced Technology Attachment (SATA) disks. With the help of auxiliary programs like Storage Archive and additional Scale services like DMAPI (Data Management Application Programming Interface), storage pools can even be outside the direct control of Scale itself, leveraging media such as tape – these so-called “external pools” are used for Hierarchical Storage Management (HSM).

Generally, applications aren't concerned about where data resides (or when it gets created), as long as it's available when its 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 required to make the data retrieval operation seamless, regardless of where the data physically resides. More importantly, because IBM Storage Scale knows exactly where data is located, it can easily move it from one location to another, without having to inform applications and users the data’s location has changed.

ILM provides a means to automate file management using a set of user-defined policies and rules. A policy is a set of rules that describes the life cycle 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 in which the data should be created. (A default pool named “system” is where data and metadata are stored by default, if no placement rules apply.) And, as data changes over its lifetime, migration rules can be applied to re-locate it or deletion rules can be applied to remove it entirely. For instance, older data that is not frequently accessed, but must be kept to comply with data retention policies, might be moved to tape if it has not been touched in the past 90 days. Regardless of where data is stored initially or re-located to, applications can continue to access it, seamlessly.

Thus, ILM helps ensure that premium storage resources are used efficiently, and that a proper balance of premium and less expensive storage resources is maintained.

1.4 IBM Storage Scale RAID

Redundant Array of Inexpensive Disks (RAID) technology is designed to protect against data loss in the event a disk failure occurs. Most RAID technologies employ some form of striping, mirroring, and/or parity to create large reliable data stores from multiple, general-purpose hard disk or solid-state drives. Several different levels of RAID are available; the levels and their associated data formats are standardized by the Storage Networking Industry Association (SNIA) in the Common RAID Disk Drive Format (DDF) standard. It should be noted that while most RAID levels offer protection against, and recovery from, hardware

defects or defective sectors/read errors, they do not provide protection against catastrophic failures (fire, flood, tornado, etc.) or "soft errors" like user error, software malfunction, or malware infection. Therefore, RAID should not be used as a replacement for a comprehensive data backup plan.

1.4.1 Understanding traditional RAID levels

To understand how IBM Storage Scale RAID works, it helps to understand how each standard level of RAID typically functions. RAID 0 splits (or "stripes") data evenly across two or more disks, without parity information, redundancy, or fault tolerance. Consequently, the failure of one drive will cause the entire RAID array to fail, due to data striping. RAID 0 is normally used to increase performance, although it can also be used to create a large logical volume out of two or more physical disks.

RAID 1 is a fault-tolerance configuration known as "disk mirroring" where data is duplicated on two separate disks. (A classic RAID 1 mirrored pair consists of two physical disk drives.) It is extremely safe, but it doubles the amount of disk space required. Therefore, it is normally used when read performance or reliability is more important than write performance and/or the amount of storage capacity available.

RAID 2, which is rarely used in practice, stripes data at the bit (rather than the block) level, and uses a Hamming code for error correction. (Hamming codes are a family of linear error-correcting codes that are capable of detect one-bit and two-bit errors.) The disks used are synchronized to spin at the same angular orientation (so they will reach the correct index at the same time). Therefore, RAID 2 generally cannot service multiple requests simultaneously. Since today's disk drives implement their own internal error correction, the complexity of using an external Hamming code instead of parity has made this level of RAID obsolete.

RAID 3, which is also rarely used in practice, consists of byte-level striping with a dedicated parity disk. Like RAID 2, RAID 3 generally cannot service multiple requests simultaneously because any single block of data will, by definition, be spread across all members of the disk set and will reside in the same physical location on each disk.

RAID 4 consists of block-level striping with a dedicated parity disk. Because data is written to data disks in stripes that are much larger than those used with RAID 2 or RAID 3, each data disk in a RAID 4 array can usually satisfy a separate user request at the same time. If one block on a disk goes bad, the parity disk in that disk's RAID group is used to recalculate the data, and the "restored" block can be mapped to a new location on disk. If an entire disk fails, the parity disk continues to prevent data from being lost - when the failed disk is replaced, the parity disk is used to recalculate its contents, automatically. An advantage of RAID 4 is that it provides good random read performance; a disadvantage is that random write performance is low due to the need to write all parity data to a single disk

RAID 5 consists of block-level striping with distributed parity - instead of keeping parity blocks on a single parity disk, RAID 5 cycles parity blocks among all the disks in the RAID array (parity for the first stripe is on the first disk, parity for the second stripe is on the second disk, and so on). The primary advantage of RAID 5 is that it eliminates the need for a single parity drive, which can become a bottleneck if large amounts of data are written to the RAID group in parallel. RAID 5's distributed parity also evens out the stress of a dedicated parity disk among all RAID members.

RAID 6 extends RAID 5 by adding a second parity block; thus, it uses block-level striping with two parity blocks distributed across all RAID group member disks instead of one. With the increase in drive capacities seen in the past few years, RAID 6 has become one of the most common RAID levels used because it protects data in the event a second disk in a RAID

group fails while another disk in the group is being rebuilt. RAID 6 does not have a performance penalty for read operations, but it does have a performance penalty on writes because of the overhead associated with parity calculations.

1.4.2 How IBM Storage Scale RAID differs from conventional RAID

IBM Storage Scale RAID is a software implementation of RAID technology that is provided by IBM Storage Scale, as opposed to an external RAID controller or other customized hardware. It automatically corrects for disk failures and other storage faults by reconstructing unreadable data using data redundancy information provided by a Reed-Solomon code or N-way replication. (Data redundancy information is created automatically, based on the RAID code used.) IBM Storage Scale RAID can automatically detect and correct up to two or three concurrent storage faults, depending on how it is configured. In the event of a total disk failure, IBM Storage Scale RAID will use the data redundancy information available to rebuild the drive onto spare space. The redundancy code layouts IBM Storage Scale RAID uses (which are referred to as "tracks") can be seen in Figure 1-3.

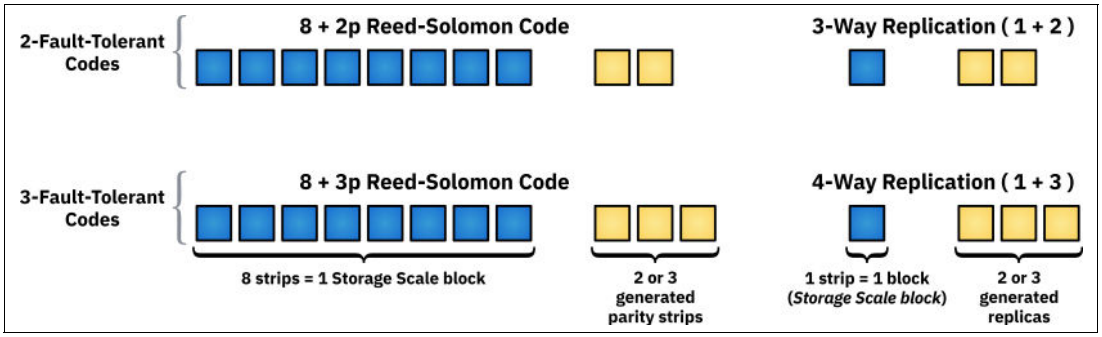


Figure 1-3 IBM Storage Scale RAID tracks (redundancy code layouts)

Using an appropriate Reed-Solomon code, IBM Storage Scale RAID divides a Storage Scale block of data equally into eight data strips and generates two or three redundant parity strips, based on the RAID code configured (i.e., 2-fault-tolerant or 3-fault-tolerant). This results in a track (stripe) width of 10 or 11 strips and a storage efficiency of 80% or 73%, respectively (excluding user-configurable spare space used for disk rebuild operations).

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

Declustered arrays

IBM Storage Scale RAID implements a sophisticated data and spare disk layout scheme that allows for arbitrarily sized disk arrays while reducing the overhead associated with recovering from disk failures.

It does this by uniformly spreading or "declustering" data, redundancy/parity information, and spare disk space across all the disks in the declustered array. Figure 1-4 on page 24 illustrates the difference between a conventional RAID layout and an equivalent declustered array.

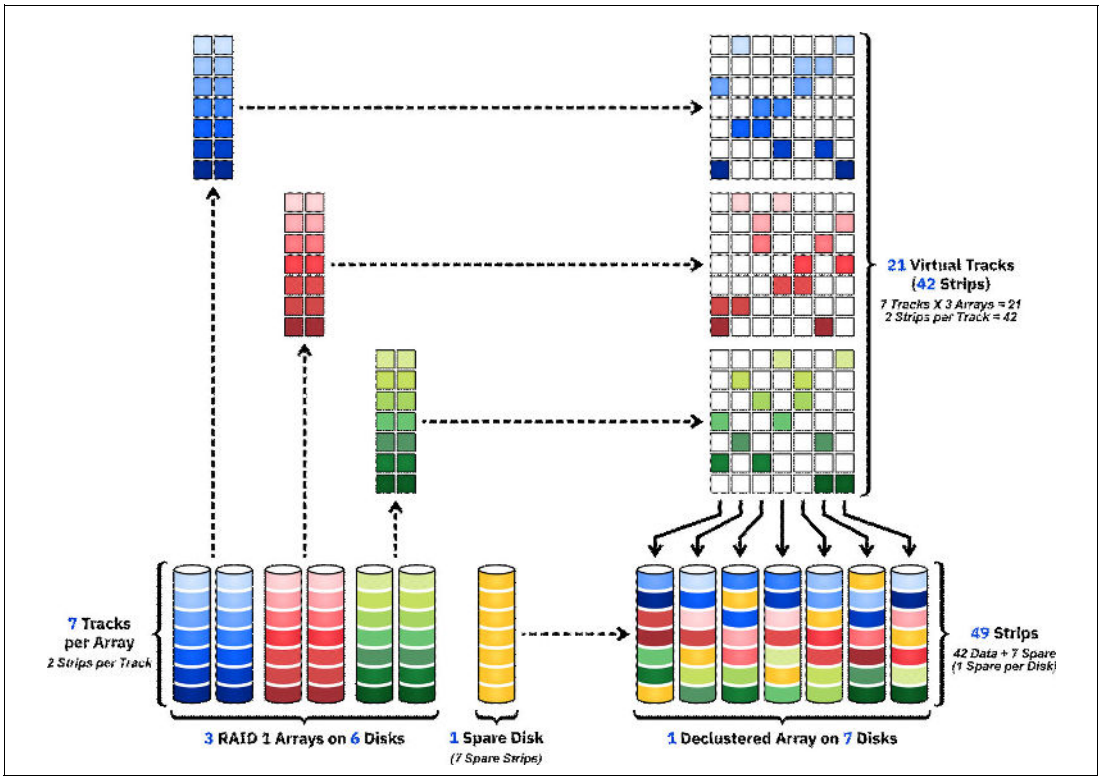


Figure 1-4 Conventional RAID layout verses a declustered array, using seven physical disks

The left-hand side of Figure 4 shows how data would typically be striped across three RAID 1 arrays consisting of two disks each (with data being stored on one disk and a replica being stored on the other). A single spare disk is also available for data rebuilding in the event a disk in one of the RAID arrays fails.

To decluster this array, the disks are divided into seven tracks (two strips per array plus the spare) as shown in the upper right-hand corner of Figure 4. The strips from each group are then spread – in no particular order – across all seven disk positions, for a total of 21 virtual tracks (3 x 7 = 21) as shown in Figure 4. The strips of each disk position for every track are then arbitrarily allocated onto the disks of the declustered array, as shown in the lower right-hand corner of Figure 4. (In this example, track distribution is performed by vertically sliding down and compacting the strips from the virtual tracks above.) The appropriate number of spare strips are also inserted randomly, one per disk, into the declustered array.

In the event a disk failure occurs, the data lost is rebuilt using all remaining operational disks in the declustered array. This results in a greater bandwidth than can be achieved with conventional RAID, which would typically be constrained by a smaller number of physical disks used. Furthermore, if an extra disk fault occurs during a rebuild operation, the number of impacted tracks that require repair is markedly less than with the previous failure. It is also less than the constant overhead of a conventional array rebuild operation. Figure 5 illustrates the difference between a disk rebuild for the conventional RAID configuration used earlier (shown at the top of the figure) and an equivalent rebuild of a declustered array (shown at the bottom).

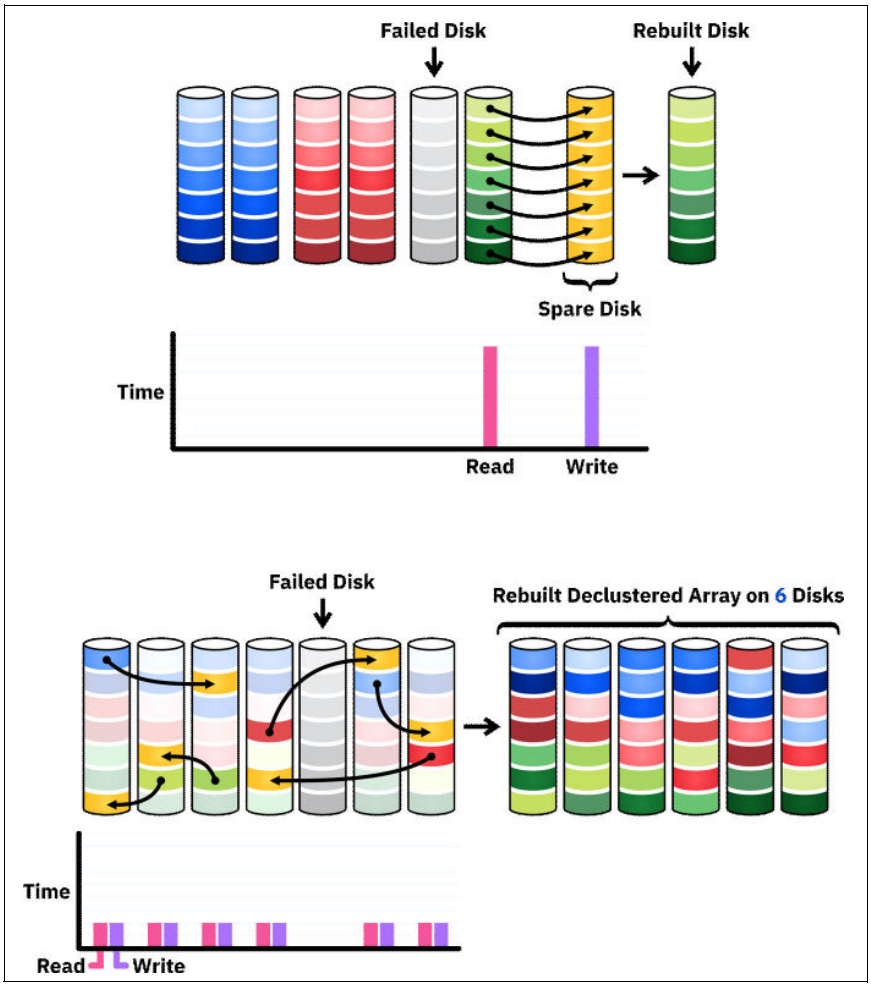


Figure 1-5 Conventional RAID disk rebuild verses a declustered array rebuild

When a single disk fails in the conventional RAID array shown at the top of Figure 5, the data on the replica (redundant) disk is read and copied onto the spare disk, which requires a throughput of seven strip I/O operations. When a disk fails in the declustered array shown at the bottom, all replica strips of the six impacted tracks are read from the surviving six disks and then written to the spare strips on these disks. Because replicas are read from all surviving disks and written to spare space on all surviving disks, rebuilding can be done in parallel resulting in a much faster rebuild time. The bar charts shown at the bottom of both arrays illustrate the disk read and write I/O throughput seen during both rebuild operations.

The decrease in declustered rebuild impact and overhead can be a factor of 3 to 4 times less than with conventional RAID. And because IBM Storage Scale stripes data across all the storage nodes of a cluster, file system performance becomes less dependent upon the speed of any single rebuilding storage array.

Another advantage of declustered array technology is that it minimizes the worst-case number of critical RAID tracks impacted by multiple disk failures. IBM Storage Scale RAID handles restoring protection to critical RAID tracks as a high priority and assigns a lower priority to RAID tracks that are not as critical. By prioritizing the rebuild of more critical tracks over less critical tracks, IBM Storage Scale RAID quickly gets out of a critical rebuild operation so it can tolerate another disk failure. Track priorities are dynamically set, so if a noncritical RAID track is used and more disk failures occur, the RAID track's rebuild priority can be escalated to "critical."

A third advantage of using declustered arrays is that they make it possible to support any number of disks in the array, as well as to dynamically add or remove disks to/from the array. With the exception of adding spares, the addition of disks to a traditional RAID configuration requires a significant amount of data reorganization and restriping. In contrast, only targeted data movement is needed to rebalance when a new disk is added to a declustered array.

End-to-end checksums

Most implementations of RAID implicitly assume that disks reliably detect and report faults, hard-read errors, and other integrity problems. However, studies show that disks do not always report read faults. And occasionally, they fail to write data event though they claim to have done so. To cover for these shortcomings, IBM Storage Scale RAID implements an end-to-end checksum that can detect silent data corruption that is caused by disks or other system components that transport or manipulate data.

Any time data is written to disk, a checksum of 8 bytes is calculated and appended to the data before it is sent to the IBM Storage Scale RAID server. When the data is received, IBM Storage Scale RAID calculates and verifies this checksum. Then, it stores the data, a checksum, and version number to disk, and then logs the version number for future verification.

When IBM Storage Scale RAID reads disks to satisfy a read operation, it compares the disk checksum against the disk data and validates the disk checksum version number. If the checksums and version numbers match, IBM Storage Scale RAID sends the data along with a checksum to the appropriate client. If the checksum or version numbers are invalid, IBM Storage Scale RAID reconstructs the data using parity or replication and returns the reconstructed data and a newly generated checksum to the client. Thus, both silent disk read errors and lost or missing disk writes are always detected and corrected.

1.5 IBM Storage Scale System

IBM Storage Scale System (formerly known as IBM Elastic Storage System) combines IBM Storage Scale software with a pre-tested, storage hardware platform. Essentially, it is a hardware implementation of Storage Scale software that is optimized for the most data-intensive workloads.

By design, IBM Storage Scale System uses a simple building-block approach with performance that scales linearly. For example, a cluster of 10 IBM Storage Scale System 3500 systems is capable of more than 1 terabyte per second (TB/s) throughput; a similar cluster of IBM Storage Scale System 6000 systems is capable of more than 3 TB/s throughput. Both systems support up to nine Serial Attached SCSI (SAS) hard disk drive expansion enclosures.

To optimize performance, IBM Storage Scale System is engineered to scale to thousands of nodes and yottabytes (YBs) of capacity. It runs IBM Storage Scale RAID, which 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 reliability demonstrated five 9’s (99.999%) of availability. Installations and updates are delivered by means of containerized software, which speeds and simplifies the maintenance process.

For data resilience, IBM Storage Scale System supports Safeguarded Copy, a data protection mechanism that provides the ability to create cyber-resilient, point-in-time copies of data volumes. Safeguarded Copy creates immutable (unchangeable) snapshots of data at

regularly scheduled intervals so operational data can be rapidly recovered in the event a cyber-attack or other potential data loss event occurs.

Further documentation for accessing an IBM Storage Scale file can be found in the IBM Storage Scale documentation and the IBM Elastic Storage System Introduction Guide, REDP-5253.

1.6 IBM Storage Scale System 6000

IBM Storage Scale System 6000 is a hardware platform that's designed to be the simplest and fastest way for organizations to build a global data platform for their file and object data. It leverages the power of IBM Storage Scale software combined with NVMe flash and hybrid flash/disk technology to deliver high-performance storage for AI, data analytics, HPC, and other file and object use cases. The IBM Storage Scale System 6000 is shown in Figure 6.

Figure 1-6 IBM Storage Scale 6000 front view

Storage Scale System 6000 is available in all-flash and hybrid configurations and can deliver:

- Up to 310 gigabytes per second (GB/S) throughput with low latency
- Up to 13 million input/output operations per second (IOPS) using NVMe over Fabrics (NVMe-oF)
- Up to 1.8 petabytes (PBe) of storage capacity in a standard 4U rack space.

Organizations often require access to two different types of storage systems – some that use high-performance media for quickly reading and writing active data, and others that provide more cost-effective storage for data that is accessed less frequently. Storage Scale System 6000 is designed to allow organizations to dial in the exact balance of performance and capacity they need. A system can be configured with standard NVMe flash drives for maximum performance or with IBM FlashCore® Modules when data density and compression are higher priority. For workloads where capacity is important, Storage Scale System 6000 supports up to nine IBM Storage Scale Expansion Enclosures.

The IBM Storage Scale Expansion Enclosure is an enterprise-class, fully redundant storage expansion enclosure that is optimized for the Storage Scale System 6000. Each enclosure can contain up to 91 20TB or 22TB self-encrypting SAS hard disk drives (HDDs). If 22TB SAS drives are used, a single enclosure is capable of delivering up to 18PB of usable storage capacity. With the ability to attach up to nine expansion enclosures to a Storage Scale System 6000, this expands the maximum additional storage capacity to 180PB of HDD storage per rack, enabling the Storage Scale System 6000 to address a wide range of workloads that operate on multi-petabyte data sets. The IBM Storage Scale Expansion Enclosure is shown in Figure 1-7 on page 28.

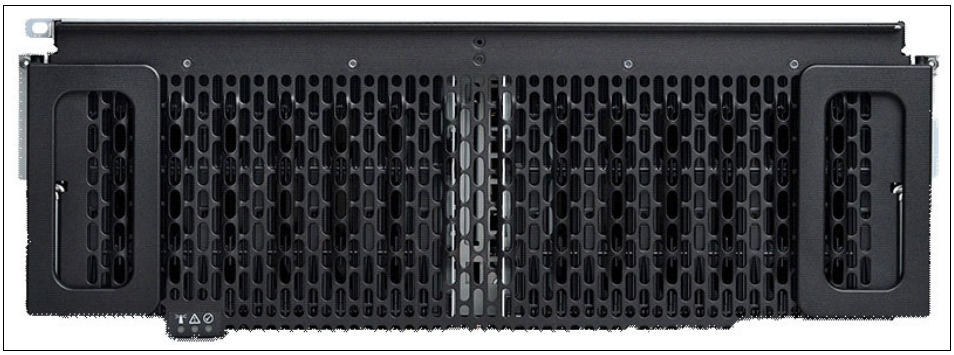


Figure 1-7 The IBM 5147-102 Storage Enclosure

In addition, IBM Storage Scale software enables the IBM Storage Scale System 6000 to scale linearly, so that throughput increases proportionally as more systems are added to a cluster.

1.6.1 IBM Storage Scale System 6000 technical overview

Each IBM Storage Scale System 6000 enclosure (MTM 5149-F48) contains two identical I/O server canisters that run the IBM Storage Scale software. Each canister uses two AMD EPYC 9454 SP5 processors and 24 Dual In-line Memory Modules (DIMMs) that can have a storage capacity of 32 or 64 Gibibytes (GiB). AMD EPYC 9454 SP5 processors have 48 cores, 96 threads, and 256 megabytes (MBs) of on-chip cache; SP5 refers to a specific CPU socket that has a large, high-pin count, Land Grid Array (LGA) with 6096 pins. Each I/O server canister also contains two 960 GB NVMe boot drives, as well as the following onboard ports:

- Two RJ45 (Registered Jack 45) 10 Gigabit Ethernet (GbE) ports, which are used for management operations
- One RJ45 1 GbE port, which is used for BMC (Baseboard Management Controller) inter-canister communication
- One mini-HDMI (High-Definition Multimedia Interface) port, which can be used for a display console
- One USB (Universal Serial Bus) 3.0 type A port, which can be used for a crash-cart keyboard
- One USB Mini-C port, which is used for service
- Additional I/O ports, based on the add-in cards used. Up to eight Gen 5 x16 PCIe (Peripheral Component Interconnect® Express) slots are available per canister and each slot can contain one NVIDIA ConnectX-7 network adapter card with "Virtual Protocol Interconnect" (VPI) capability and two 200 Gb ports or one NVIDIA ConnectX-7 InfiniBand network adapter card with one a 400 Gb port.

IBM Storage Scale System 6000 is an all-flash array platform that can house up to 48 Non-Volatile Memory express (NVMe)-attached drives. (These drives are accessible from the front of the enclosure.) Each NVMe-attached drive supports multiple queues so that each AMD EPYC CPU core in the enclosure can communicate directly with the drive. This results in better performance because it avoids the latency and overhead associated with core-to-core communication. In addition, the NVMe drive transport protocol used is a faster and less complicated storage drive transport protocol than Serial Attached SCSI (SAS).

Clients have the option of purchasing 24 drives in a half-populated configuration or 48 drives in a fully populated configuration. (A minimum number of 24 drives is required.) In a 24-drives

configuration, drives must only be plugged into slots with even numbers (2-24) in the top canister (canister 1) and slots with odd numbers (25-47) in the bottom canister (canister 2). Industry-standard, 2.5 inch NVMe drives with the following capacities are supported: 3.84 terabyte (TB), 7.68 TB, 15.36 TB, and 30.72 TB.

As an alternative to the current industry-standard NVMe flash drives, IBM Storage Scale System 6000 can now be configured with 19.2TB or 38.4TB IBM FlashCore Modules (FCMs) for increased data density and data resilience. FCMs are IBM's patented high-performance NVMe flash drives, designed for low latency, density, and reliability. They feature a PCIe Gen4 U.2 interface and high-speed NAND (NOT-AND) memory to provide high throughput and IOPS with consistent and predictable latency. IBM researchers have developed custom field programmable gate arrays (FPGAs) that free up computational overhead so that compression and encryption can be performed with zero effect on performance. FCMs also implement advanced flash management, which improves flash endurance over standard implementations without sacrificing latency.

1.6.2 Utility/EMS node

Each IBM Storage Scale System cluster requires a minimum of one Enterprise Management Server (EMS). The EMS, also referred to as the Utility Node, is the central management component in an IBM Storage Scale System. The EMS is implemented as a RHEL KVM VM and serves as the control hub for the Storage Scale System 6000 graphical user interface (GUI) and call home capabilities. The EMS/Utility VM provides management and health monitoring capabilities. It also runs a container with Ansible playbook that can provide the orchestration of complex tasks, such as cluster configuration, file system creation, and Storage Scale software code updates. The minimum release level on the IBM Storage Scale System 6000 is IBM Storage Scale System 6.1.9.

The EMS/Utility Node can be ordered as part of an IBM Storage Scale System 6000, but this typically applies to new clusters only. Conversion from a Power9® EMS to an IBM Storage Scale System 6000 Utility Node is not yet supported. The mixing of EMS types in the same environment is not supported either.

For more information about the IBM Storage Scale System EMS/Utility Node refer to the Model 5149-23E server specifications.

Advantages of Using a Utility Node Over a POWER9 EMS

In an IBM Storage Scale System 6000 environment, using a Utility Node instead of a traditional POWER9 EMS (Enterprise Management Server) offers several architectural and operational advantages. Here's a breakdown of why it's suggested and what benefits it brings:

- Hardware Flexibility and Modernization
  - Utility Nodes are based on x86 architecture, which is more widely available and often more cost-effective than POWER9 systems.
  - Easier to source, maintain, and upgrade due to broader industry support.
- Simplified Deployment and Management
  - IBM provides pre-integrated and validated configurations for Utility Nodes, reducing setup complexity.
  - The deployment guide includes streamlined instructions for using Utility Nodes, making them easier to manage in mixed or new environments.
- Unified EMS Functionality

  - Utility Nodes can consolidate multiple EMS roles into a single unified node, reducing hardware footprint and simplifying network architecture.
- Improved Support for Modern Features
  - Enhanced compatibility with features like RoCE, multi-rail TCP, and IPv6, which are increasingly important in high-performance and AI workloads.
- Better Integration with IBM Storage Scale Software
  - Utility Nodes are optimized for the latest versions of IBM Storage Scale, ensuring better performance and compatibility with new features and updates.

Note: Co-existence and Migration Support

Utility Nodes can co-exist with POWER9 EMS nodes, allowing for gradual migration or hybrid deployments. This is particularly useful in environments transitioning from older ESS models to the Storage Scale System 6000.

1.7 License considerations

The IBM Storage Scale System 6000 follows the same license model as the other IBM Elastic Storage System (ESS) products. The two currently available options for IBM Storage Scale System 6000 are IBM Storage Scale for ESS Data Access Edition and IBM Storage Scale for ESS Data Management Edition.

The IBM Storage Scale System 6000 uses capacity-based licensing, which means that a customer can connect as many clients as needed without extra license costs. For other types of configurations, contact IBM or your IBM Business Partner for license details.

For more information about licensing on IBM Storage Scale and IBM Storage Scale System 6000, refer to the Frequently Asked Questions (FAQ) page for IBM Storage Scale and IBM Storage Scale System Licensing details.

2

IBM Storage Scale System 6000 architecture and overview

This chapter describes the architecture and provides an overview of IBM Storage Scale System 6000 (ISS 6000). It covers the following topics:

2.1 Platform

This section provides an overview of the ISS 6000 platform. An ISS 6000 enclosure disk enclosures for up to 48 Non-Volatile Memory Express (NVMe)-attached SSD drives and a pair of server canisters. The ISS 6000 is available as a performance-model all-Flash array platform with NVMe-attached SSD drives (providing significant performance improvements compared to SAS-attached flash drives), a capacity-model with external enclosures of SAS HDD drives and a few internal NVMe drives for logtip, and a hybrid-model providing both internal NVMe and external enclosures with SAS HDDs

2.1.1 Canisters and servers

This section describes the CPU, memory, and networking components of the ISS 6000 (Model 5149-F48) system.

CPU

The ISS 6000 system uses dual-socket AMD EPYC Genoa 9454 48-core processor per I/O canister node for a total of four CPUs per enclosure. Figure 2-1 shows the CPU and DIMM slot locations within a canister.

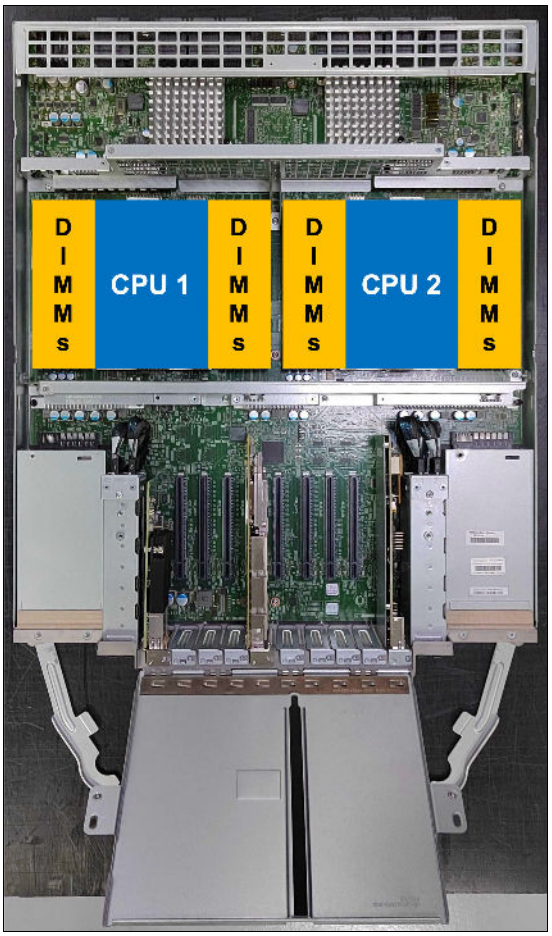


Figure 2-1 CPUs and DIMMs within a canister

Memory

Details on the memory available for each canister is depicted in Table 2-1 below.

Table 2-1 Memory configuration

Number of DIMMs per server canister	Total memory per server canister	Number of DIMMs per server enclosure	Total memory per server enclosure
24x 32 GiB	768 GiB	48x 32 GiB	1536 GiB
24x 64 GiB	1536 GiB	48x 64 GiB	3072 GiB

Networking

The ISS 6000 includes four Gen5 x16 Peripheral Component Interconnect express (PCIe) slots per canister. Adapter options are listed in Table 2-2.

Table 2-2 IBM Storage Scale 6000 adapter options

Feature Code	PCIe Form Factor	InfiniBand	Ethernet
AJQQ	CX-7 400 Gb NDR-IB single-port OSFP	NDR 400 Gb	N/A
AJRG	CX-7 VPI 200GbE/NDR200 dual-port Crypto Enabled	NDR 200 Gb (Network Direct RDMA)	200 Gigabit Ethernet (200GbE)
AJQS	*CX-7 VPI 200 GbE / NDR-IB 200 Gb dual-port*	*NDR 200 Gb, HDR200 200 Gb, HDR 100 100 Gb, EDR 100 Gb*	*200 GbE, 100 GbE*

Note: As of May 2025 the Feature Code AJQS has been withdrawn and is no longer available. This was replaced by Feature Code AJRG.

You can configure ISS 6000 in a mixed cluster that contains IBM ESS 3500, IBM ESS 5000, IBM ESS 3500, IBM ESS 3200, and IBM ESS 3000 systems

Figure 2-2 IBM Storage Scale 6000 Rear view

Peripheral Component Interconnect express (PCIe)

Each canister has 8 Gen5 x16 PCIe slots. Figure 2-3 on page 34 shows how the PCIe lanes are connected.

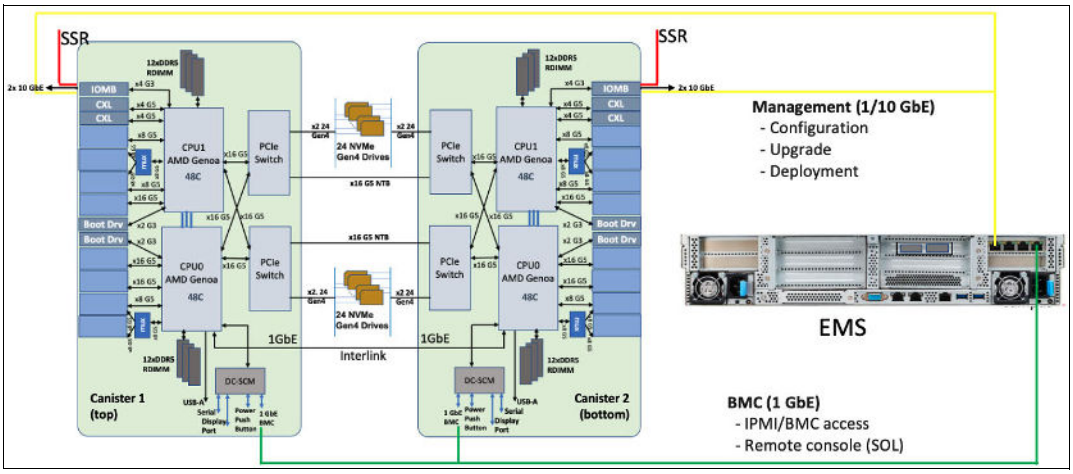


Figure 2-3 System topology for ISS 6000

2.2 IBM Storage Scale Utility Node

With the release of IBM Storage Scale 6.1.9.1 the ability to use an IBM Storage Scale Utility node is extended to the ISS 6000.

There are a number of use cases where additional nodes are needed in addition to the IBM Storage Scale System itself. These use cases include management of the Storage Scale System building block(s), serving protocols via cluster export services (CES), serving as Active File Management (AFM) gateways for caching use cases, serving as key management servers, and much more.

Figure 2-4 Scale System Utility Node

Key Features of the Scale System Utility Node:

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

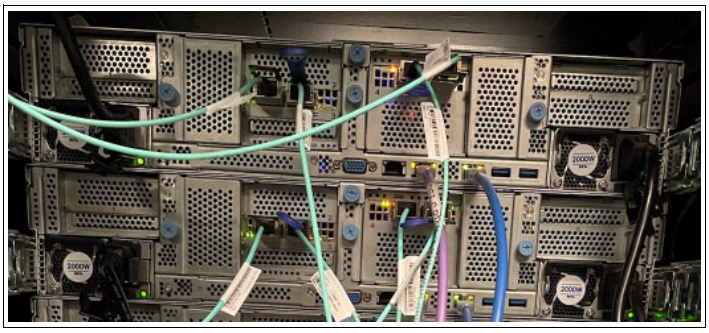


Figure 2-5 Utility Node Rear View

Scale System Utility Node Contains:

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

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

Note: POSIX (Portable Operating System Interface) is a set of standard operating system interfaces, based on the UNIX operating system (OS), that are designed to maintain compatibility among different OSs. The most recent POSIX specification – Institute of Electrical and Electronics Engineers (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 be capable of running on, any operating system that adheres to POSIX standards.

2.3 GUI overview

A graphical user interface (GUI) service runs on the ISS 6000 management server (EMS). It can be used to monitor the health of the ISS 6000 and to perform management tasks. This section provides an overview of the GUI but is not comprehensive.

To access the GUI, enter the IP address or hostname of the EMS in a web browser by using the secure https mode:

```
https://<IP or hostname of EMS>
```

The **systemctl** command can be used on the EMS to start or stop the GUI. Table 2-3 shows the **systemctl** command options.

Table 2-3 The systemctl command options

Description	Command
Start the GUI services	**systemctl start gpfsgui**

Description	Command
Check the status of the GUI service	`systemctl status gpfsgui`
Stop the GUI service	`systemctl stop gpfsgui`

2.3.1 GUI users

GUI users must be created before the GUI can be used. To grant special rights, roles are assigned to the users.

When the GUI is used for the first time, an initial user must be created:

`/usr/lpp/mmfs/gui/cli/mkuser` <username> **`-g SecurityAdmin`**

After the initial user is created, you can log in to the GUI with the newly created user. To create more users, select Services -> GUI -> Users and enter the user information. By default, users are stored in an internal user repository. Alternatively, an external user repository can also be used. An external user repository can be configured by selecting Services -> GUI -> Additional Authentication and entering the user information.

Administrators can also configure multi-factor authentication, which requires GUI users to be registered in an IBM Security Verify repository. After multi-factor authentication is enabled, additional login information will be sent to the registered email or phone of the user. The user must provide the additional login information for two factor authentication (2FA) before they are able to successfully log in to the GUI. For more information, see Configuring GUI details in IBM Security® Verify for multi-factor authentication.

2.3.2 System setup wizard

After you log in to the GUI for the first time, the system-setup wizard is started. The following steps are a high-level overview of what to expect when using the system-setup wizard:

1. After the welcome page, the Verify Storage page queries important system information and performs several checks to verify whether the system is ready for use.

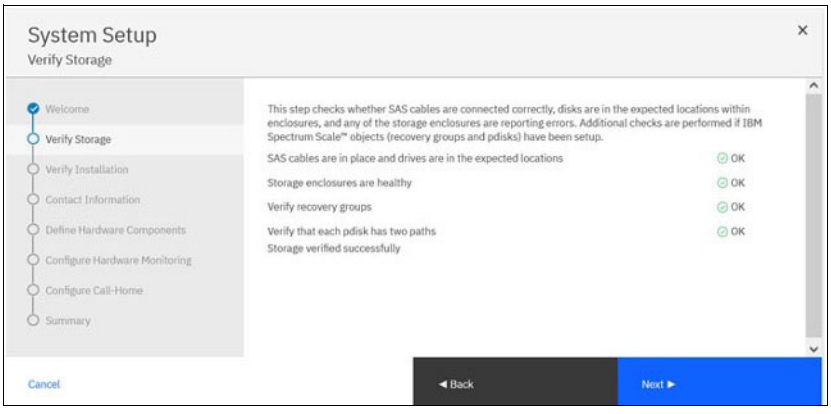


Figure 2-6 The system setup wizard GUI

2. After passing the storage verification check, the next Verify Installation page will check for the following other prerequisite conditions as shown in Figure 2-7 on page 37:

- Does the **`mmhealth`** utility report that the cluster file system is working correctly on the EMS and building block systems?
- Is there at least one recovery group configured?
- Is the system able to successfully look up the servers and enclosures, and add them to the component database?

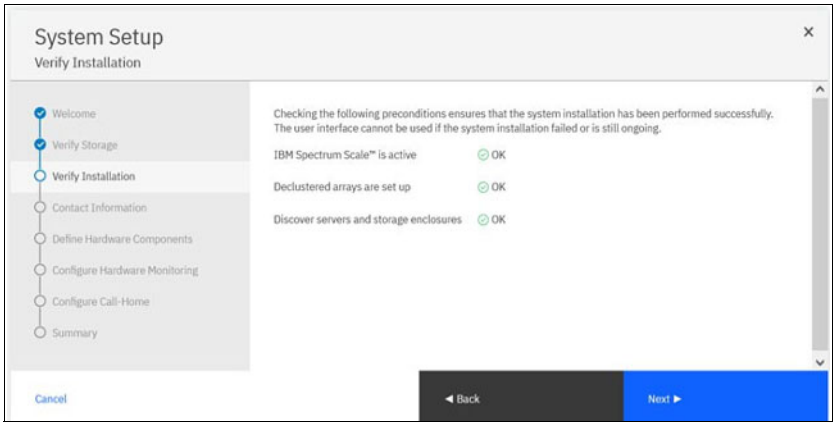


Figure 2-7 Verify installation GUI page

3. After the installation is verified, in the Contact Information page enter the information to enable the optional hardware **callhome support** feature. Provide the company information, system information, primary contact, and optional secondary contact information for the environment. See Figure 2-8, Figure 2-9 on page 38 and Figure 2-10 on page 38.

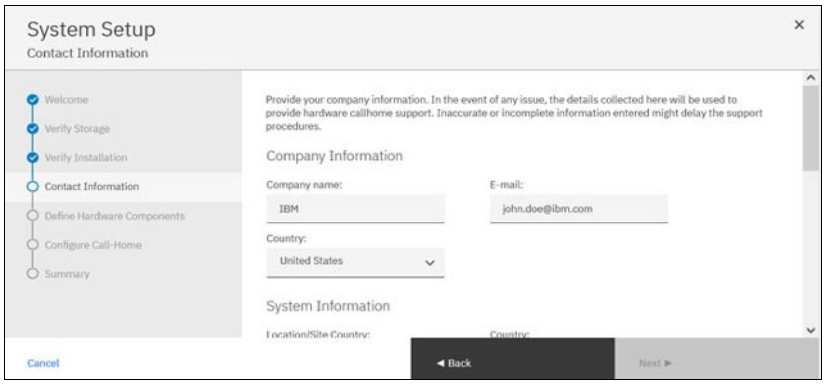


Figure 2-8 System setup, contact info part 1

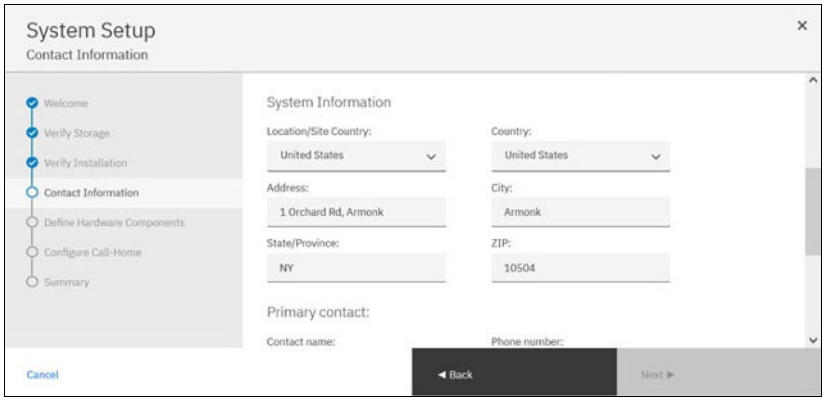


Figure 2-9 System setup, contact info part 2

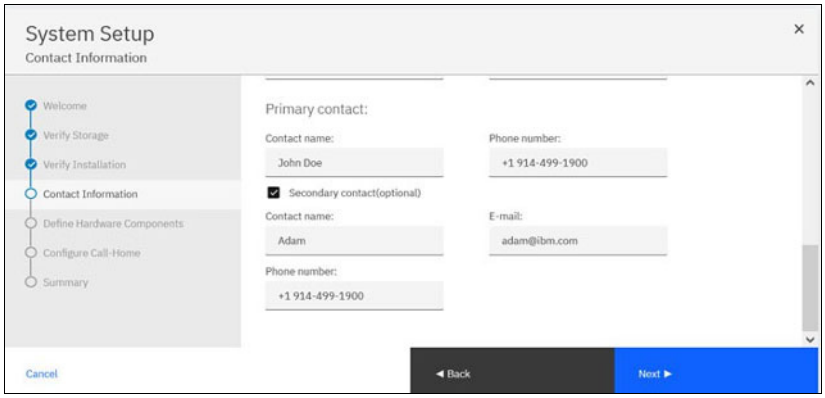


Figure 2-10 System setup, contact info part 3

4. After all checks pass, in the Racks page the user can define where the ISS 6000 systems are installed as shown in Figure 2-11. The user can either choose a predefined rack type or choose **Add new specification** if none of the rack types match the rack the ISS 6000 was installed in. It is important that the selected rack-type has the same number of height units. A meaningful name can then be associated with each of the racks that are defined to help differentiate them.

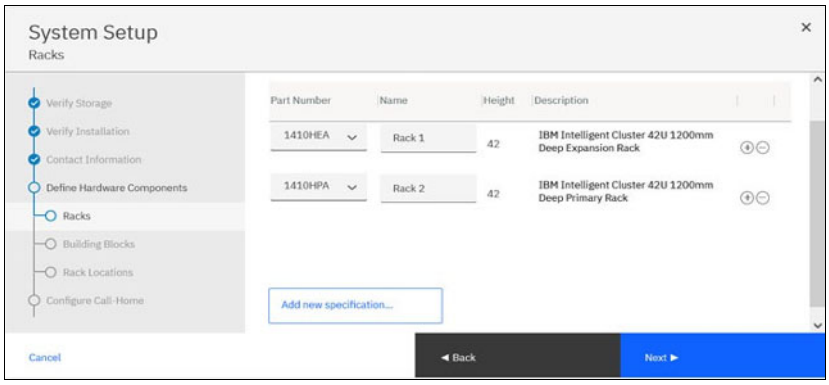


Figure 2-11 System setup, rack information

5. The Building Blocks page displays one row for each ISS 6000 or other IBM ESS models in the environment. The user can assign names to each building block or go with the default. See Figure 2-12 on page 39.

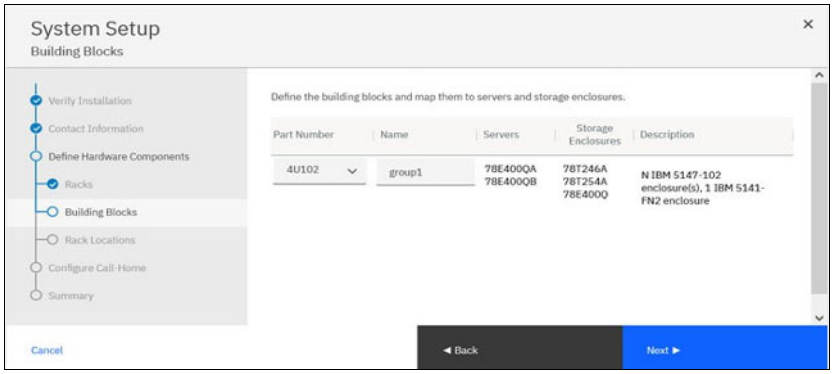


Figure 2-12 System setup, define building blocks

6. The ISS 6000 systems are assigned to the rack locations in which they are mounted. See Figure 2-13.

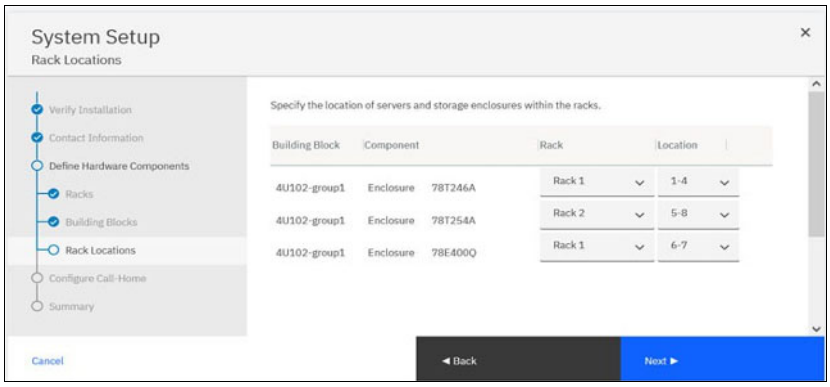


Figure 2-13 System setup, specify rack locations

7. After all component configuration steps are complete, provide the details to configure Call-Home:
   a. The Select Nodes page asks for the details that are needed to automatically create a support case with IBM Support when issues are reported as shown in Figure 2-14.

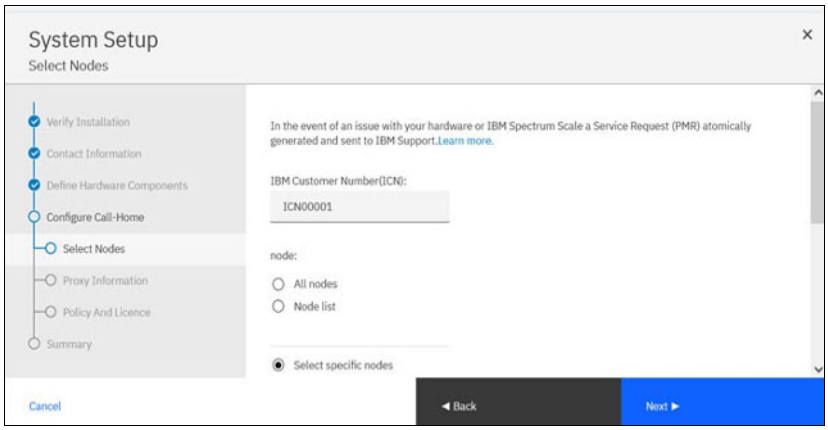


Figure 2-14 System setup, select nodes

b. The Proxy Information page is used to configure the network proxy settings to allow communication for Call-Home to contact IBM Support as shown in Figure 2-15 Configuring the proxy information is optional.

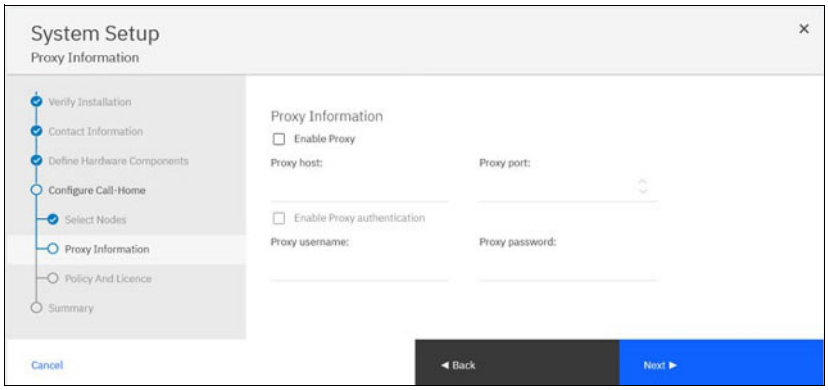


Figure 2-15 System setup, proxy information settings

c. The Policy and License section is the final page to enable call home. Review and accept the IBM privacy policy to enable call home as shown in Figure 2-16.

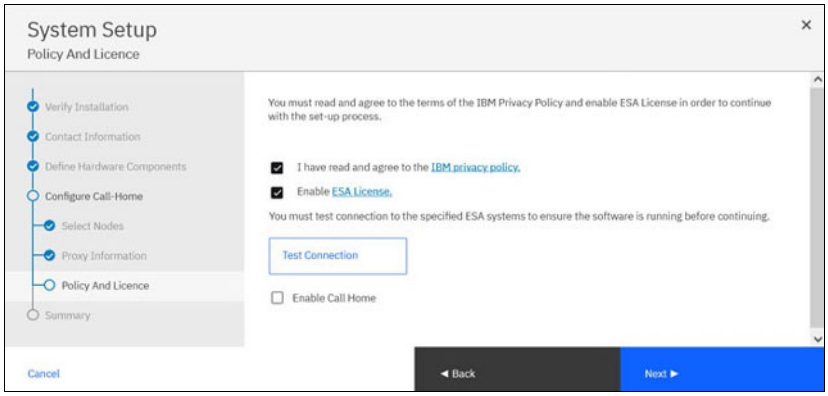


Figure 2-16 System setup, policy and license

2.3.3 Using the GUI

After logging in to the GUI, the Overview page will be displayed. This page provides a view of all components in the environment and their health states. Clicking the numbers or links displays a more detailed view see Figure 2-17 on page 41.

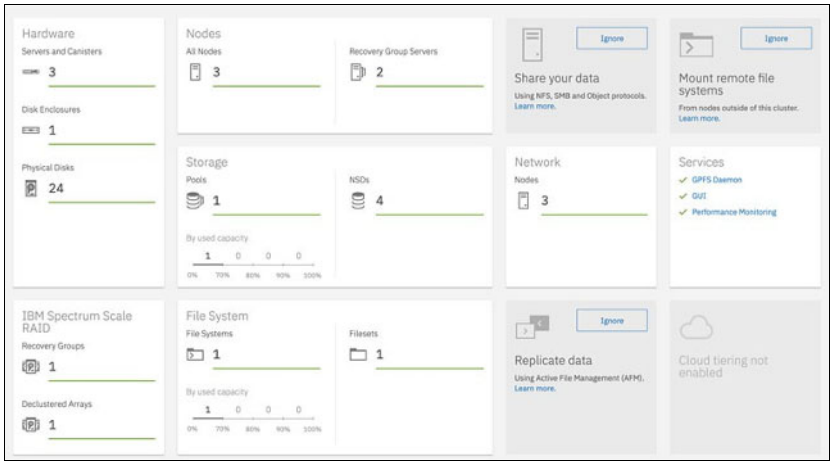


Figure 2-17 IBM Storage Scale GUI overview page

The header area of the GUI provides a quick view of the current health problems and tips for addressing them, if applicable. Also, some links to help resources are presented on the page.

Use the navigation menu, located in the panel on the left of the GUI, to select other GUI pages as shown in Figure 2-18 Each GUI page has a unique URL, which allows the user to bookmark and directly access specific pages and log in and load the GUI directly to a specific section.

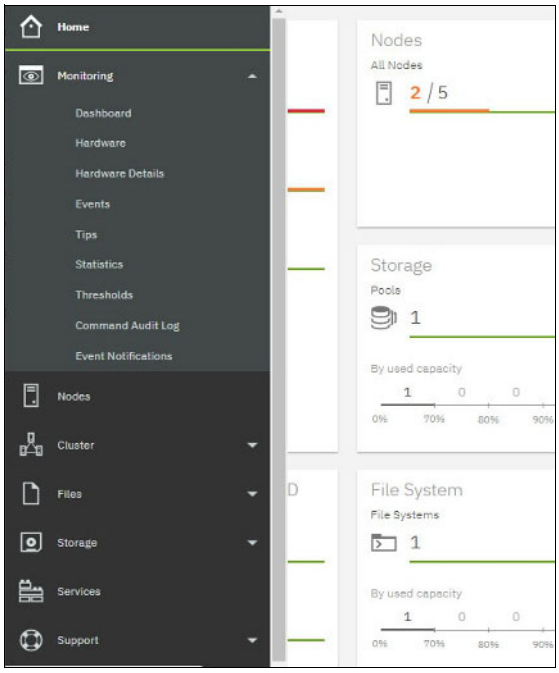


Figure 2-18 GUI navigation panel

Some menus, such as Protocols, are only displayed when the related features, such as NFS, SMB, or AFM are enabled.

Most tables that are shown in the GUI have columns that are hidden by default. Right-click the table header as shown in Figure 2-19 on page 42 and select the columns to display the columns.

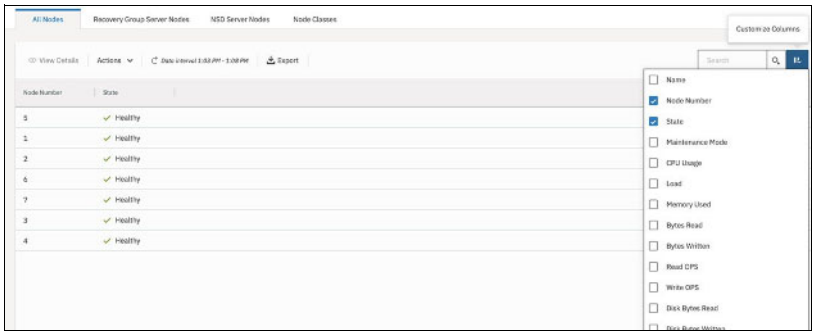


Figure 2-19 GUI show/hide column options

The table values can be sorted by clicking one of the column headers. The arrow in the table header indicates which column is being used for sorting.

Double-click a table row to open a more detailed view of the selected item.

2.3.4 Monitoring of ISS 6000 hardware

Select **Monitoring -> Hardware** to list the ISS 6000 enclosures within the racks. A table lists all enclosures and the related canisters.

Name	Serial Number	State	Building Block	Type	Daisy chained with
5149-F48-78L0032	78L0032	Healthy	sess6k1	F48 Enclosure	
sess6k1a-ib.pbm.ihos...	78L0032A	Healthy	sess6k1	Canister/Server	
sess6k1b-ib.pbm.ihos...	78L0032B	Healthy	sess6k1	Canister/Server	
sess6k1ems-ib.pbm.i...	78P002X	Unknown		Management Server	

Figure 2-20 Hardware page with ISS 6000 system

Select **Edit Rack Components** when ISS enclosures or servers are added or removed, or if their rack location changes.

Select **Replace Broken Disks** to start a guided procedure to replace any failed disks. Also, the ISS 6000 can be configured to replace broken disks by using commandless disk replacement.

Click the ISS 6000 in the virtual rack to see more information about the ISS 6000, including the physical disks and the two canisters as shown in Figure 2-20. Move the mouse over the components, such as drives and power supplies, to see more information. Clicking components opens a page with more detailed information on the specific component. Failed disks are indicated with the color red. Right-click the component to open a context menu that enables the user to replace the selected failed disk.

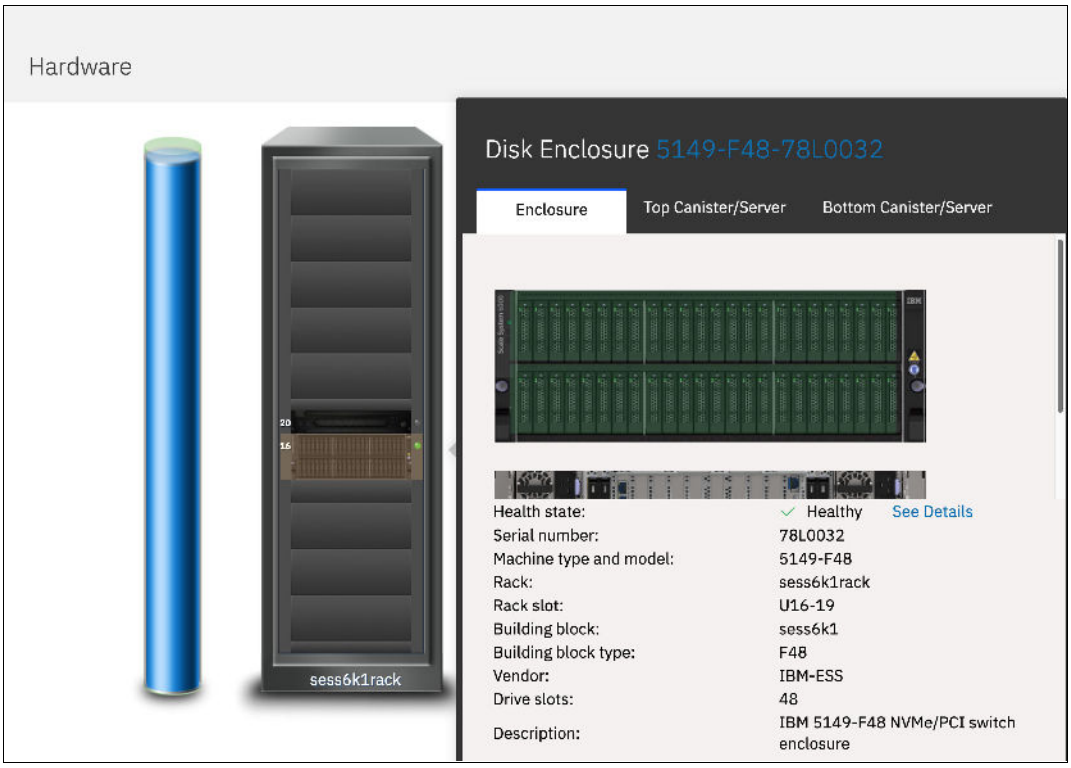


Figure 2-21 ISS 6000 details in the Hardware page

If there is more than one rack, click the arrows that are displayed on the left and the right side of the rack to switch to another rack.

Select **Monitoring -> Hardware Details** to display a more detailed information and the health states of the ISS 6000 and its internal components. See Figure 2-22.

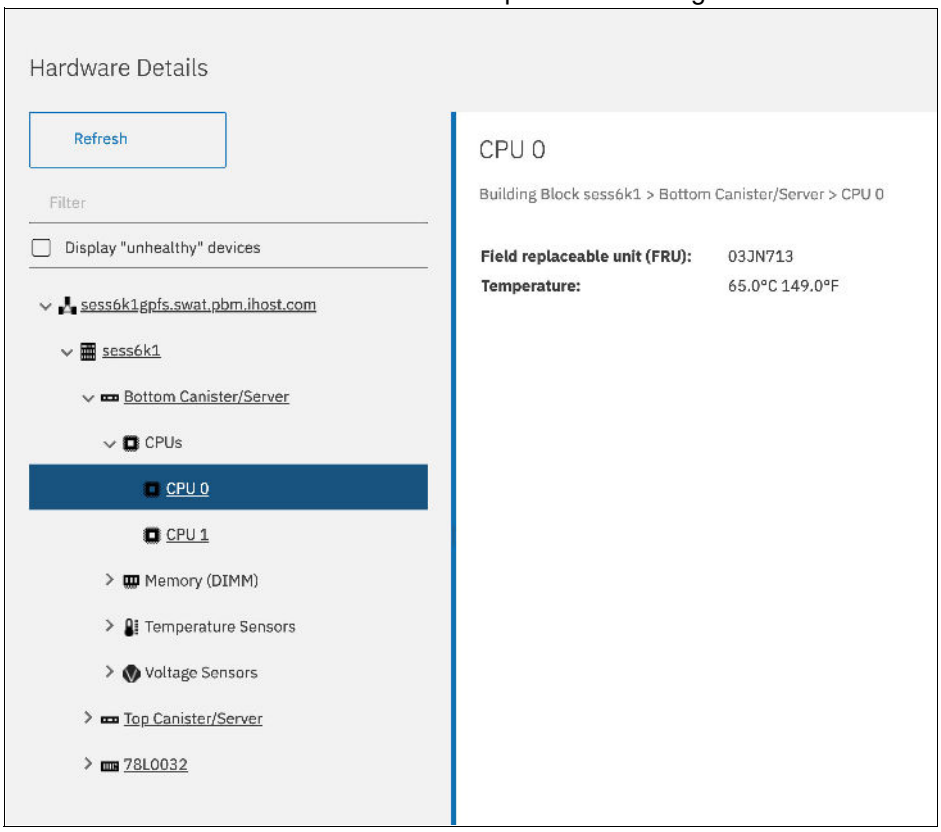


Figure 2-22 The hardware details page

The Hardware Details page allows the user to search for components by text and filter the results to display only unhealthy hardware.

Click the **>** icon on the tree nodes to display subsequent children. For example, to view all current sensors, the user can click **Current Sensors** of the canister in Figure 2-22.

2.3.5 Storage

The Storage menu provides views into the different storage components, such as the physical disks, declustered arrays, recovery groups, virtual disks, network shared disks (NSDs), and storage pools. The list of declustered arrays is shown in Figure 2-23.

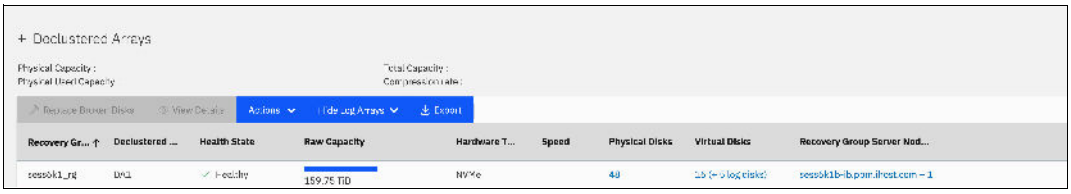


Figure 2-23 Declustered Arrays view

2.3.6 Replacing broken disks

The GUI provides a guided procedure that can be used to replace broken disks. Verify that the replacement disks have the same field-replaceable unit (FRU) numbers as the disks that are going to be replaced.

Note: If commandless disk replacement is enabled, the guided disk replacement is unavailable in the GUI.

This procedure can be started from multiple places within the GUI where disk notifications can be seen. For example, to check for broken disks, you can select

Storage -> Physical Disks as shown in Figure 2-24.

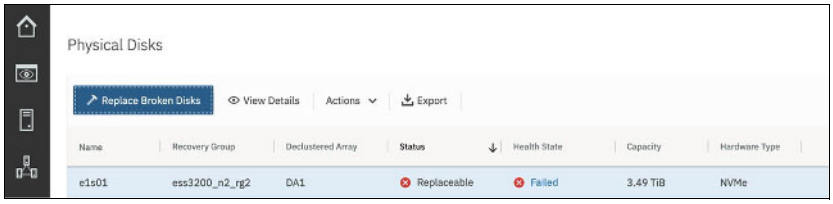


Figure 2-24 Physical disks page

Select **Replace Broken Disks** to see a list of all broken disks that can then be selected for replacement. Select an individual disk from the table and select **Replace Disk** to replace the selected disk. In both cases, a fix procedure guides you through replacing the disks. See Figure 2-25.

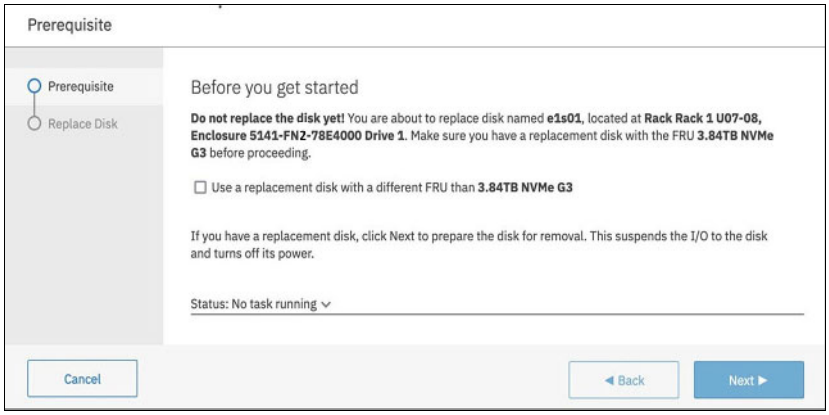


Figure 2-25 Disk replacement procedure dialog

2.3.7 Health events

Select **Monitoring -> Events** to review the entire set of events that are reported in the IBM ESS system. Under the Event Groups tab, all individual events with the same name are grouped into single rows, which can be useful when many events are reported. The Individual Events tab lists all the events, including multiple occurrences of the same event name. Events are assigned to a component, such as canister, enclosure, or file system. The user can click any of the components in the bar chart above the grid to filter for events of that selected component.

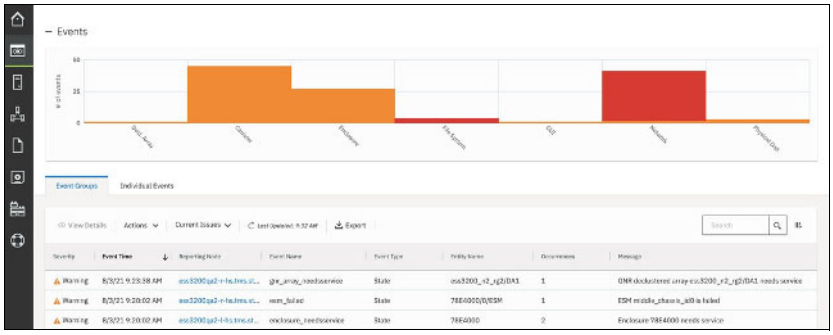


Figure 2-26 Health events overview page

The following filter options by event type are available as a drop-down list in the Events page shown in Figure 2-26:

- Current Issues displays all unfixed errors and warnings.
- Notices displays all transient messages of type “notice” that were not marked as read. While active state events disappear when the related problem is solved, the notices stay forever until they are marked as read.
- Current State displays all events that define the current state of the entities and excludes notices and historic events.
- All Events displays all messages, even historic messages, and messages that are marked as read. This filter is not available in the Event Groups view because of performance implications.

The user can mark events of type Notices as read to change the status of the event in the Events view. The status icons become gray if an error or warning is fixed, or if it is marked as read.

Some issues can be resolved by selecting **Run Fix Procedure**, which is available on select events. Right-click an event in the **Events** table to see this option.

2.3.8 Event notification

The system can send emails and Simple Network Management Protocol (SNMP) notifications when new health events appear. Any combination of these notification methods can be used simultaneously. Select **Monitoring -> Event Notifications** to configure event notifications.

Sending emails

Select **Monitoring -> Event Notifications -> Email Server** to configure the email server where the emails are sent. In addition to the email server, an email subject and the senders name can also be configured. Select **Test Email** to send a test-email to an email address. See Figure 2-27 on page 47.

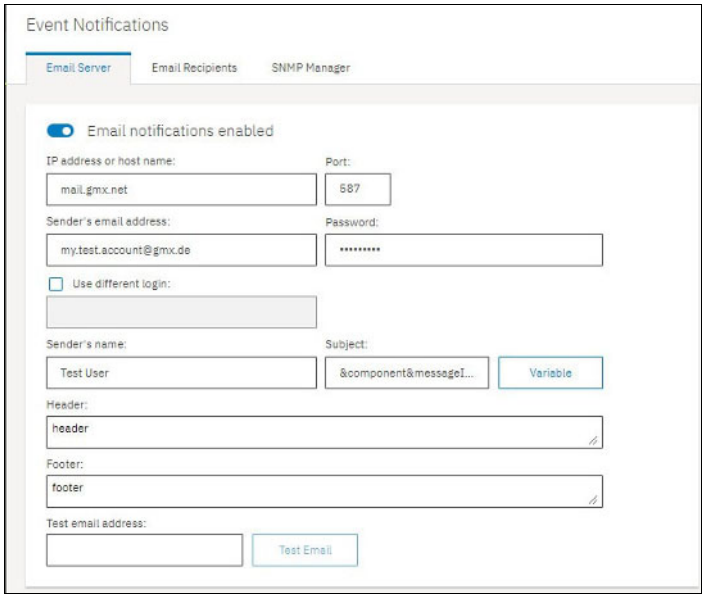


Figure 2-27 Event email server configuration dialog

You can define multiple email recipients, selecting **Monitoring -> Event Notifications -> Email Recipients**. For each recipient, the user can select the components for which to receive emails, and the **For minimum severity level** (Tip, Info, Warning, or Error). Optionally, instead of sending a separate email per event, a daily summary email can be sent. You can also elect to receive a **Daily Quota report**. See Figure 2-28.

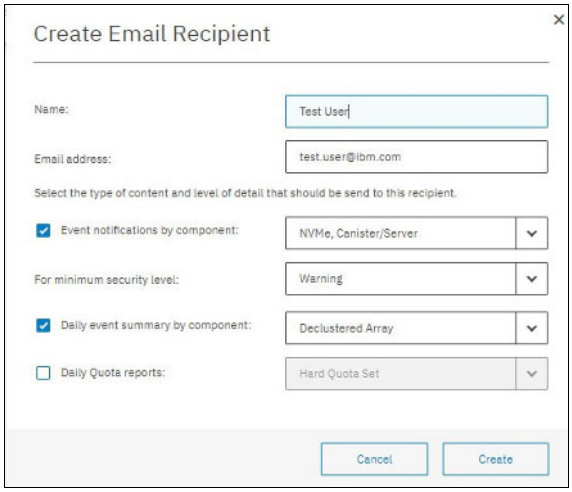


Figure 2-28 Event email recipient configuration dialog

2.3.9 Dashboards

Select **Monitoring -> Dashboard** to view an easy-to-read, single-page, real-time user interface that provides an overview of the system performance.

Some default dashboards are included with the product. Users can further modify, delete the default dashboards, and create new dashboards to suit their requirements. The same dashboards are available to all GUI users, so any modifications are visible to all users.

A dashboard consists of several dashboard widgets that can be displayed within a chosen layout.

Widgets are available to display the following items, as shown in Figure 2-29:

- Performance metrics
- System health events
- File system capacity by file set
- File sets with the largest growth rate in the last week
- Timelines that correlate performance charts with health events

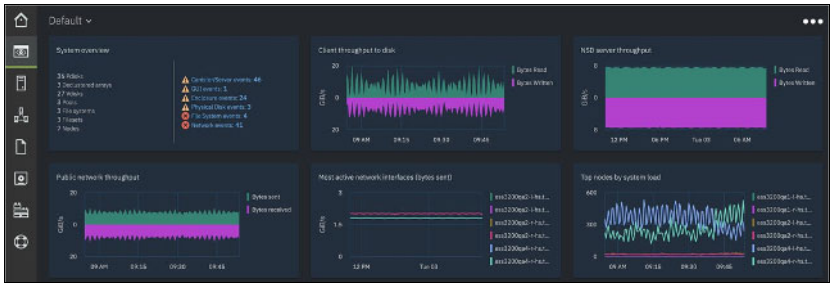


Figure 2-29 Monitoring dashboard gui

More information

The previous sections provided a rough overview of the GUI. For more detailed information on the GUI, read the Monitoring and Managing the IBM Elastic Storage Server Using the GUI, REDP-5471 and use the online help pages that are included within the GUI. Additional information is also available in IBM Storage Scale Version 5.2.2.

2.4 Software enhancements

In this section, details of the software enhancements for the ISS 6000 are described.

2.4.1 Containerized deployment

The IBM Scale System installation and management software includes but is not limited to the following items:

- ISS-specific documentation for installation and upgrade scripts
- A container-based deployment model that focuses on ease of use
- Other tools for the IBM SSR to use for installing IBM Scale System, such as essutils

Third- and fourth-generation IBM Scale Systems deploy a container-oriented management software stack in the IBM ESS Management Server that includes Ansible Playbooks for installation and orchestration.

IBM preinstalls this complete, integrated, and tested Scale System solution stack on the ISS servers during manufacturing.

Like other IBM software, the Scale System solution-stack levels are released as version, release, modification, and fix pack level.

For more information about Containerized deployment, see Storage Scale System Quick Deployment Guide.

The IBM Storage Scale System solution-stack components are periodically updated, tested, and released as a new level of ISS solution software. IBM recommends that clients plan to upgrade their ISS to the current level of ISS solution software stack at least once a year.

2.4.2 Red Hat Ansible

In the ISS 6000 container, the Ansible tool is included, which helps to orchestrate a set of commands (also called tasks). With this capability, you can automate the deployment process into a few commands. Figure 2-30 shows the tree of the ansible directory that is included in the container.

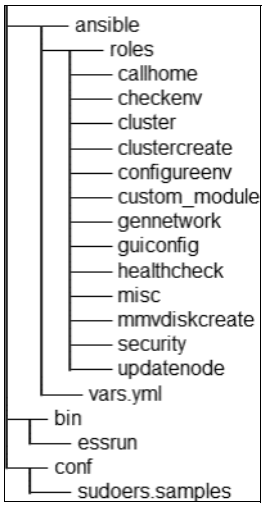


Figure 2-30 Ansible directory tree included in the container

The roles directory contains a set of folders that contain the various tasks that can be run, for example, configureenv, contains the essrun config load set of tasks.

If you want to import or use the roles within your own Ansible Playbook, you can import the roles and the vars.yml file because it contains several variables that are used within each role.

Example 2-1 shows how to import a Storage Scale System role into your own Ansible playbook.

Example 2-1 Import a Storage Scale System role into an Ansible playbook

```
---
- name: ESS config load
  hosts: all
  remote_user: root
  vars_files:
    - /opt/ibm/ess/deploy/ansible/vars.yml

  # importing roles tasks:
  - include_role:
      name: /opt/ibm/ess/deploy/ansible/roles/configureenv
```

2.4.3 The mmvdisk command

The **mmvdisk** command is an integrated command suite for IBM Storage Scale RAID. It can simplify IBM Storage Scale RAID administration and encourages and enforces consistent best practices for server, recovery group, VDisk NSD, and file system configuration.

The **mmvdisk** command is used to manage IBM Storage Scale RAID installations. If you are integrating ISS 6000 with an existing installation of ESS systems that are not **mmvdisk** recovery groups, those systems must be converted into mmvdisk recovery groups before you add the ISS 6000 into the same cluster.

For more information about the **mmvdisk** command, see Managing IBM Storage Scale RAID with the **mmvdisk** command.

2.4.4 The mmhealth command

The **mmhealth** command monitors and displays the health status of services that are hosted on nodes and the health status of an IBM Storage Scale cluster. Although a cluster might be made up of many different types of components, **mmhealth** provides a health status of the varied components in the system, including any important cluster issues.

For any type of cluster, the status includes the status of the IBM General Parallel File System (GPFS) daemons, the node software status, tracking of events that happened to the cluster, and the file system health status.

The depth of the details for one node depends on a few factors:

- If the node is a software-only node, where IBM Storage Scale formats only external block devices to the cluster
- When the ISS 6000, running with IBM Storage Scale RAID or running with IBM Fusion, is using **mmhealth** to monitor and report on the following, inexhaustive, list of items:
  - Hardware items which are the same as other ISS hardware solutions:
    - Temperature of different sensors of the enclosure
    - Power supply hardware status
    - Fan speeds and status
    - Voltage sensors data
    - Firmware levels reporting and monitoring
    - Boot drive status and monitoring
- IBM Storage Scale RAID specific items:
  - Recovery Groups status and monitoring
  - Declustered Array status and monitoring
  - Physical drives status and monitoring
  - VDisks status and monitoring
- IBM Storage Scale software-related items:
  - NSD status and monitoring
  - Network communication status and monitoring
  - GUI status and monitoring (of the GUI nodes)

- Status and monitoring protocol support
- File system status and monitoring
- Pool status and monitoring
- NSD protocol and statistics
- Statistics of other protocols when applicable

The **mmhealth** command provides IBM Storage Scale software-related checks across all node and device types present in the cluster. Software RAID checks are present across all IBM Storage Scale RAID offerings (such as IBM ESS 5000, IBM ESS 3500, and ECE). Devices (such as ISS 6000) that are integrated with IBM Storage Scale hardware, also get hardware checks and monitoring.

For more information about how to use the **mmhealth** command, see Monitoring system health by using the **mmhealth** command.

Monitoring IBM Storage Scale RAID

This section provides links to additional documentation:

- Commands to Monitor IBM Storage Scale RAID
-
- IBM Spectrum Scale Erasure Code Edition: Planning and Implementation Guide which describes the mmhealth command in the following sections:
  - Section 7.7 shows general command usage.
  - Section 7.8 shows example use case scenarios.
- The mmhealth command with the complete command usage, features, and examples

Recent updates to mmhealth command

Several significant changes were recently made to the **mmhealth** command for the release of the IBM ESS 3500. The **mmhealth** command was updated to support monitoring these new features. The following list includes some of these differences:

- Architecture change to x86_64 from IBM POWER
- Support for NVME drives
- External storage enclosures support
- Dual canister design within single building block

The **mmhealth** command includes several changes to support the IBM ESS 3000, the IBM ESS 3200, the IBM ESS 3500, and the ISS 6000.

- The “Canister events” category was included to support many of the differences between legacy IBM ESS systems and IBM ESS 3500 and later.
- The “Server” category was also adjusted.

Both of these adjustments and other changes to **mmhealth** are included in the following sections.

Canister events

These events are recent and were specifically added to support the newer canister-based building-block configuration of the IBM ESS 3000, IBM ESS 3200, IBM ESS 3500, and ISS 6000. For more information about, for example, events that are related to the boot drive, temperature, CPU, and memory, see Canister events in IBM Documentation.

A new command, **ess3kplt**, was created for Storage Scale RAID to provide CPU and memory health information to **mmhealth** with the following command path:

```
/opt/ibm/gss/tools/bin/ess3kplt
```

The **ess3kplt** command uses the following parameters:

```
usage: ess3kplt [-h] [-t SELECTION] [-Y] [-v] [--local]
Optional arguments:
-h, --helpShow this help message and exit.
-t SELECTIONProvide selection keyword [memory|cpu|all].
-Y Select report listing.
-v Enable additional output.
--localSelect localhost option.
```

The **ess3kplt** command can be used to inspect memory or CPU resources. Example 2-2 shows a sample output.

Example 2-2 Sample ess3kplt output

```
ESS3K Mem Inspection:
  InspectionPassed:True
  Total Available Slots:8(expected 8)
  Total Installed Slots:8(expected 0 or 8)
  DIMM Capacity Errors:0(Number of DIMMs with a size different from ['64 GB'])
  DIMM Speed Errors:0(Number of DIMMs with a speed of neither 3200 MT/s nor 3200
MT/s MT/s)
  Inspection DateTime:2021-08-31 14:02:28.338486

ESS3K Cpu Inspection:
  InspectionPassed:True
  Total CPU Sockets:1(expected 1)
  Total Populated Sockets:1(expected 1) Total Enabled   CPU Sockets: 1(expected 1)
Total Cores:48 (expected [48])
  Total Enabled Cores:48 (expected [48]) Online CPUs:---
  Total Threads:96 (expected 96)
  CPU Speed Errors :0(Number of CPUs with a speed different from ['3300 MHz'] MHz)
Inspection DateTime:2021-08-31 14:02:28.562756
```

Example 2-3 shows a sample verbose output.

Example 2-3 Sample ess3kplt verbose output

```
ess3kplt:memory:HEADER:version:reserved:reserved:location:size:speedMTs:
ess3kplt:memorySummary:HEADER:version:reserved:reserved:availableSlots:installedSl
ots:capacityEr ror:speedError:inspectionPassed:
ess3kplt:memory:0:1:::PO_CHANNEL_D:passed:passed:
ess3kplt:memory:0:1:::PO_CHANNEL_C:passed:passed:
ess3kplt:memory:0:1:::PO_CHANNEL_B:passed:passed:
ess3kplt:memory:0:1:::PO_CHANNEL_A:passed:passed:
ess3kplt:memory:0:1:::PO_CHANNEL_E:passed:passed:
ess3kplt:memory:0:1:::PO_CHANNEL_F:passed:passed:
ess3kplt:memory:0:1:::PO_CHANNEL_G:passed:passed:
ess3kplt:memory:0:1:::PO_CHANNEL_H:passed:passed:
ess3kplt:memorySummary:0:1:::8:8:0:0:true:
ess3kplt:cpu:HEADER:version:reserved:reserved:location:speedMHz:status:status2:num
Cores:numCores Enabled:numThreads:
ess3kplt:cpuSummary:HEADER:version:reserved:reserved:totalSockets:populatedSockets
```

```
:enabledSocket
s:totalCores:enabledCores:totalThreads:speedErrorrs:inspectionPassed:
ess3kplt:cpu:0:1:::CPU0:passed:ok:ok:48:48:96:
ess3kplt:cpuSummary:0:1:::1:1:1:48:48:96:0:true:
```

CPU and DIMM-related events reported by the **mmhealth** command rely on the **ess3kplt** command in the IBM ESS 3000, IBM ESS 3200, IBM ESS 3500, and ISS 6000 environments.

2.5 RAS enhancements

ISS 6000 is the next generation of the IBM Storage Scale System product family that is built on a high availability and performance storage server platform.

The ISS 6000 system consists of the server portion, MTM 5149-F48, and the storage enclosure, MTM 5149-91 (1 – 9 JBOD drawers are supported). For the purposes of this document, the focus is on the 4U form factor server portion.

ISS 6000 is designed to provide an improved customer experience compared to previous ESS releases. The IBM ESS 3500 has improvements in the following areas:

- Ordering
- Installing
- Upgrading
- Using
- Servicing

The following list includes the key components of the server portion, 5149-F48:

- IBM storage enclosure with commercial NVMe drives
- Red Hat Enterprise Linux (RHEL) 9.2 with NVMe support
- IBM Storage Scale 5.1.9.x software features and functions
- IBM Storage Scale Software RAID

ISS 6000 is an IBM installed product with a combination of customer-replaceable units (CRUs) and FRUs.

2.5.1 RAS features

ISS 6000 was designed with improvements to reduce the frequency of failures, minimize workload interruptions, and easily detect, identify, and report problems to decrease service-repair time. It also incorporates redundancy into its design so that component replacements do not interfere or affect system operations.

ISS 6000 is designed to offer high system and data availability with the following features:

- Dual-active, intelligent node canisters with mirrored cache
- Erasure coding for data durability that uses the RAID component of IBM Storage Scale for ISS, supporting a combination of 3-way, 4-way, 8+2P, and 8+3P erasure codes
- Checksums and versions of data blocks with always on checks for data validity
- Rapid rebuild of failed drives or data blocks with detected errors

- Dual-port flash drives with automatic drive failure detection and RAID rebuild
- Redundant hardware, including power supplies and fans
- Hot-swappable and client replaceable components
- Automated path failover support for the data path between the server and the drives
- Embedded BMC and remote-control capability
- Monitoring
  - Hardware components
  - Firmware levels
  - GNR and IBM Storage Scale components
- Event notification
- Call home
- IBM Storage Scale Health checker
- First-time data capture (FTDC)

To maintain high levels of system availability, multiple methods and services are used to monitor the various system components. System status changes are reported by notifications that provide detailed event information, which also includes user-action information for next-steps or required actions to address the issue. If callhome is enabled, ISS 6000 can generate a call home for inventory updates and for various failures that can occur on the system. This automated service processes and sends the proper diagnostic information to IBM Support servers to assist with debug and troubleshooting.

ISS 6000 RAS features consist of the following items:

- Monitoring
  - Hardware components
  - Firmware levels
  - GNR and IBM Storage Scale components
- Event notification
- Call home
- IBM Storage Scale Health checker
- First-time data capture (FTDC)

2.5.2 Enclosure overview

ISS 6000 includes node-to-node communication through an internal Ethernet private network and nontransparent bridge (NTB) for peer node diagnostic and control. Remote console through serial over LAN (SOL) using baseboard management controller (BMC) intelligent platform management interface (IPMI) is available for monitoring and controlling the ISS 6000 enclosure and to assist with deployment and installation.

Several methods of power control for the canister and drive slots are available on the system to assist with system recovery and troubleshooting, which helps to decrease component downtime. LED power indicators are in place on the front of the enclosure, drive carrier, fan, power module, and the canister as a visible sign that the hardware is receiving power. LED status indicators are used for the drives, fans, canisters, power modules, and enclosures that help point out if any components might be experiencing issues.

ISS 6000 offers an NVMe drive with the following capacity with either 24-drive or 48-drive installation options:

- 3.84 TB 2.5 inch PCIe Gen4 NVMe
- 3.84 TB U.3 PCIe Gen5 NVMe
- 7.68 TB 2.5 inch PCIe Gen4 NVMe
- 7.68 TB U.3 PCIe Gen5 NVMe
- 15.36 TB 2.5 inch PCIe Gen4 NVMe
- 30.72 TB 2.5 inch PCIe Gen4 NVMe
- 19.2 TB FlashCore Module (FCM4)
- 38.4 TB FlashCore Module (FCM4)

The drives are accessible from the front for service without having to extend the drawer to a service position. Figure 2-31 shows the front view of the ISS 6000 enclosure.

Figure 2-31 IBM Storage Scale 6000 enclosure front view

The canister FRUs and the power modules are accessible from the rear without extending the drawer into a service position. Figure 2-32 shows the rear view of the ISS 6000 enclosure (the 5149-F48 portion) and Table 2-4 outlines the rear port mapping.

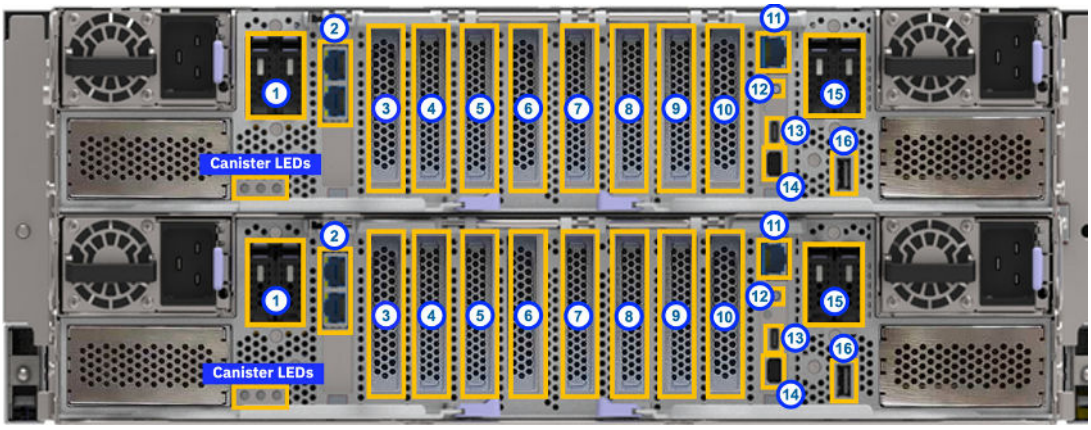


Figure 2-32 IBM Storage Scale 6000 enclosure rear view

Table 2-4 IBM Storage Scale 6000 rear port mapping

Item Number	Description/Usage
1	Dual Boot Drives (mirrored)
2	2x10 GbE Management Network
3 - 10	Nvidia CX7 networks x16 G5
11	1 GbE for BMC or host
12	Push Button for BMC reset
13	USB C-mini port
14	Mini HDMI
15	Dual CXL (not used)
16	USB A port

2.5.3 Memory configuration details

The IBM Storage Scale 6000 canisters each contain 24 DIMM slots and are available in two memory configuration options:

- 768 GiB with 24 × 32 GB DDR5 4800 MT/s Memory RDIMMs (#AK0N)
- 1536 GiB with 24 × 64 GB DDR5 Memory RDIMMs (#AK0P)

Note: Memory DIMMs cannot be mixed.

2.5.4 Disk details

IBM Storage Scale System 6000 uses a mirrored set of 960 GB M.2 NVMe SSD drives as the boot disks. The M.2 SSD includes the Power Loss Protection (PLP) feature. Given the IBM Storage Scale RAID design and given the M.2 PLP feature to ensure that the data is persistent for IBM Storage Scale RAID log files that are maintained in the boot disks, ISS 6000 does not require a Battery Backup Unit (BBU).

2.5.5 Networking details

When planning to install a 100G adapter, the following adapters are available:

- ConnectX-7 InfiniBand or VPI dual port HPC adapter (#AJQS)
  - InfiniBand - HDR200 200 Gb / HDR100 100 Gb / EDR 100 Gb / NDR 200 Gb
  - Ethernet - 100 GbE / 200 GbE
- ConnectX-7 InfiniBand, single port, 400 Gb (#AJQQ)

2.6 Machine type model and warranty

The IBM Storage Scale System 6000 server has a single MTM value: 5149-F48. It includes a 3-year warranty. ISS 6000 also offers same-day service upgrade options, and optional priced services that include lab-based services (LBS) installation.

2.7 Components: FRU and CRU

The components within the IBM Storage Scale 6000 enclosure are similar to the previous IBM ESS 3500 and 3200:

- Dual-canister architecture with two hot swappable I/O canister nodes
- Designed for easier access to the system by using rear access for replaceable parts and a simple pull-down lever mechanism.
- Six Fan units (5+1 redundant) at the front of the enclosure from the top, which allows for removal and replacement without interrupting system functionality.

The ISS 6000 contains some notable improvements compared to previous ESS versions:

- The canisters are modified with a focus on increased serviceability over IBM ESS 3200.
- Each canister has two additional adapter slots, allowing for added connectivity and support for attaching additional storage enclosures.
- No removable screws when accessing FRU parts inside a canister.
- A redesigned single canister lid, with no attached riser, improves serviceability and accessibility. All adapters now attach to the riser on the canister system board.
- Service Port (SSR port) added. See Figure 2-33.

Figure 2-33 SSR port location on back panel

The fans can be accessed for service by extending the drawer into a service position as shown in Figure 2-34.

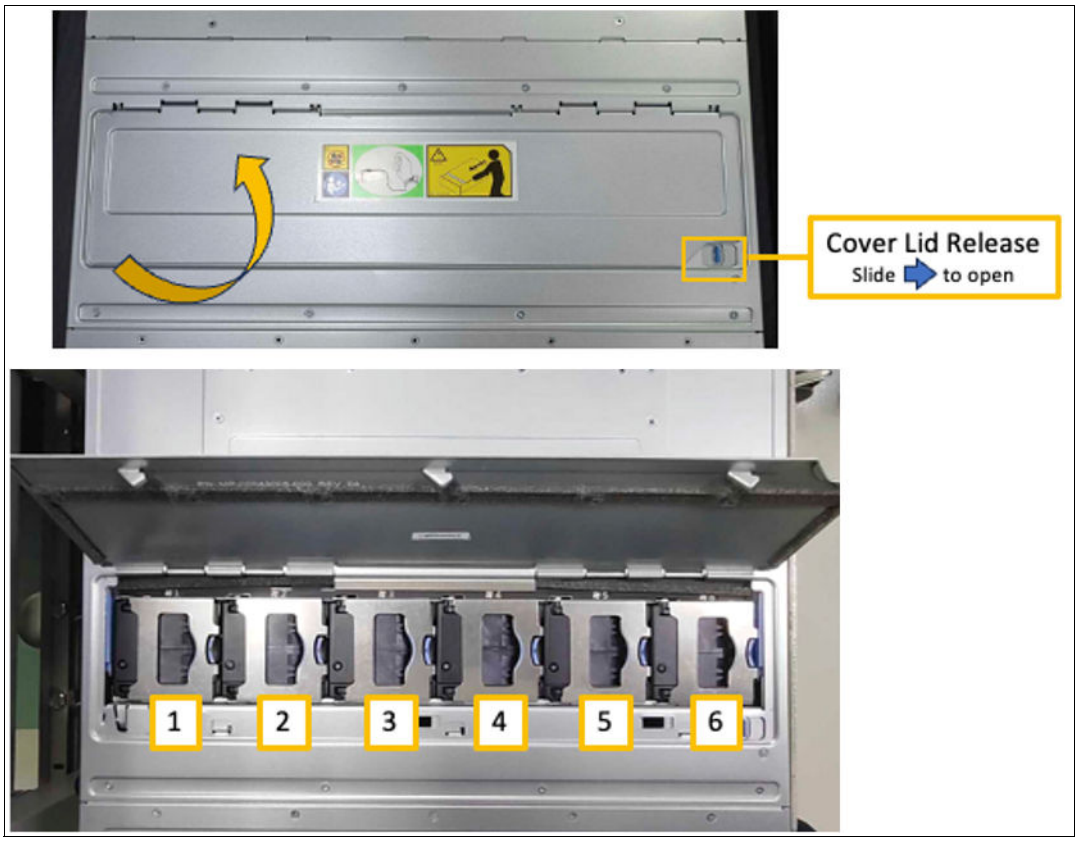


Figure 2-34 Fan cover access

Replacing the enclosure and the rails also requires extending the drawer to the service position.

In general, ISS 6000 service strategy takes the following approach:

Any hardware component that can be accessed only by opening the cover of the canister is considered to be a FRU. A FRU requires IBM service personnel to perform any needed service or repair to those components that are inside the canister cover.

- FRUs
  - Fan module (from top)
  - Server Module Canister FRU kit (from rear) – FRU
  - Canister top lid FRU kit - FRU
  - Additional FRUs (inside canister)
  - Boot Drive
  - DIMM Memory
  - Coin Cell Battery
  - TPM
  - PCIe network adapter
  - Enclosure Chassis, including midplanes, which cannot be replaced separately
  - Cable Management Arm (CMA)

  - Any hardware component that can be accessed for maintenance without removing the canister is typically a CRU. You can perform the repair by using assistance and information from the ESS GUI or related IBM Documentation.
- CRUs
  - NVMe drives at the front
  - Power Supply Module at the rear
  - Enclosure bezel kit
  - Drives: 3.84 TB, 7.68 TB, 15.36 TB, 30.72 TB, Drive Filler

2.8 Maintenance and service procedures

The IBM Storage Scale 6000 platform can identify a failed component for repair based on the monitoring and failure isolation capability of the RAS software. Guided maintenance and service procedures are available in the IBM ESS 3500 Service Guide.

Because IBM Documentation is a single point of reference that provides information about IBM systems hardware, operating systems, and server software, it is recommended that users search for IBM Storage Scale 6000 to get the most recent updates. The online IBM documentation is actively maintained and updated with the latest information.

When an issue is identified, a Service Guide option is listed that brings up the appropriate service procedure. The ISS Service Guide can also be downloaded by using the link in the navigation.

Concurrent maintenance repair and updates are available on the ISS 6000 for servicing and replacing of FRUs and CRUs in the system. This allows for maintenance to be performed on the system while it is used in normal operations. In addition, CRUs are hot-swappable components in the ISS 6000 that allow for replacement without powering down the server. Components such as fans and power modules are supported as concurrently removable and replaceable parts of the ISS 6000.

2.9 Software-related RAS enhancements

Managing and monitoring software components of the ISS 600 is critical to obtain the necessary detail from the command-line interface to properly debug a failing component. To increase availability to physical disks, there is physical disk redundancy in the ISS 6000, where each server has a path to the data physical disks, but the paths from both servers are always active. In addition, to achieve optimal performance on the ESS, a shared recovery group layout is available that allows both servers in an ISS 6000 building-block to concurrently access all the available drives and their bandwidth. Additional information on recovery groups is provided in the next section.

2.9.1 FRU and location

The mmlsenclosure command displays the environmental status of IBM Storage Scale RAID disk enclosures. The enclosure status reported by the mmlsenclosure command (and consequently, the GUI) displays the FRU number and the location of that FRU within the enclosure to help diagnose failures. Information about multiple components, including fans

and power modules, is reported with an indication for service, if needed. For more information, see the mmlsenclosure command.

2.9.2 Enclosure components

Several component statuses are reported by the enclosure as shown in Table 2-5.

Table 2-5 Component status reported by the mmlsenclosure command

Component	Command output
Canister	Status of left and right node canister
CPU	Status of each CPU in each canisters
DIMM	Status of each memory module associated with a canister
Fan	Status of each fan in canisters and power supplies

2.10 Call home functions

Concurrent with the release of the ISS 6000 is a new type of call home. In this document, the term “legacy call home” refers to what has been in place for previous ESS solutions. ESS legacy call home applies to IBM ESS 5000, IBM ESS 3000, and IBM ESS 3200. It is also referred to as hardware and software call home. ESS legacy call home uses the Electronic Service Agent (ESA) on the EMS. It uses legacy IBM infrastructure.

The ISS 6000 uses a new version of the call home software that is called Call Home Connect Cloud. Call Home Connect Cloud applies to the EMS plus ISS 6000 for all clusters that contain at least one ISS 6000.

Call Home Connect Cloud is a more holistic or unified approach to addressing service needs for the hardware. The new code in the command mmsysmon runs on the EMS and IO nodes and is part of the IBM Spectrum Scale code base. Now, as part of the IBM Spectrum Scale code, software-based call home channels all of the needed information to IBM by using cloud-based infrastructure to process the data.

Daily call home still uploads files directly to ECuRep.

The main advantage of the new call home is that it addresses fragmentation of hardware and software data. No longer are there two different systems or sets of data, “software call home” and “hardware call home”.

The following list provides some other advantages of using Call Home Connect Cloud:

- Enables easier integration of internal tools inside IBM (Health Checker, quality databases, customer portals).
- Call home messages are not triggered solely for support cases. Call home events can trigger additional events for monitoring.
- Allows two ways communication with customer systems (IBM Spectrum Scale back channel).

For more information about how call home works, see the IBM Storage System Deployment Guide and the IBM Storage Scale System 6.2.3 guide.

For a more information on the ESA, see IBM Electronic Service Agent.

3

Planning considerations

This chapter provides planning information specific to deploying the IBM Storage Scale System 6000 hardware, software, networking, and IBM Storage Scale System Utility Node (EMS). Guidance is also provided on required skills and recommendations for services that you might want to consider. It covers the following topics:

3.1 Planning

This section provides planning information specific to deploying the IBM Storage Scale System 6000 hardware, software, networking, and IBM Storage Scale System Utility Node. Guidance is also provided on required skills and recommendations for services that you might want to consider.

3.1.1 Technical and delivery assessment

When you order an ISS 6000, certain functional and non-functional requirements need to be fulfilled before the configuration can be created and the order can be entered.

A technical and delivery assessment (TDA) is an internal IBM process that includes a technical inspection of a completed solution design. This process assures customer satisfaction and ensures a smooth and timely installation. Technical subject matter experts (SMEs) who were not involved in the solution design participate to answer the following questions:

- Will the ISS 6000 solution work?
- Is the implementation and plan sound?
- Will it meet customer requirements and expectations?

TDAs are necessary for every ESS order. For the systems service representative (SSR) to begin installation, the TDA procedure must be performed to complete the installation worksheet. The worksheet outlines the items that must be implemented by the SSR during setup.

Include the following information on the TDA worksheet:

- Management IP address and netmask
- Baseboard Management Control (BMC) IP address and netmask
- Root password

The two TDA processes are described in the following list:

1. The pre-sales TDA. This is done by IBMers or IBM Business Partners by using the file and object solution design engine (FOSde) tool.
2. The pre-install TDA. SMEs also evaluate the customer's readiness to install, implement, and support the proposed solution. This can be done with the IMPACT tool.

The two TDA processes have assessment questions, but also baseline benchmarks that need to be performed before the order can be fulfilled. Those tools are run by IBM sales or resellers, so they can help and direct you regarding this process.

3.1.2 Hardware planning

These requirements include the hardware solution components that are mandatory but are not included in the ISS 6000 building block (2U). Two types of these requirements are described in the following list:

1. The requirements that must be IBM-provided, such as the management switch
2. The requirements that can be either customer-provided or IBM-provided, such as the HS network or rack

Rack solution

The ISS 6000 comes with at least one rack from IBM; it can come with more if multiple building blocks (BB) are ordered. The rack also holds the IBM Storage Scale System Utility Node (EMS) and the management switch. If the HS switches are ordered from IBM, those are also included in the rack.

Although the preferred option is the rack version of the ISS 6000 solution, it is possible to order the ISS 6000 without the rack. If you choose to follow this path, you must verify that the rack can hold the weight of the solution, and that the power distribution units (PDUs) on the rack are the right ones for the ISS 6000 solution. In addition, you must contact IBM to configure the management switch that comes with the solution.

Management switch

The ISS 6000 first building block on a site includes a 1GbE management switch from IBM (8831-S52). This switch is part of the ISS 6000 solution, and it is not an independent part that can be replaced with equivalent hardware by the customer.

Figure 3-1 is an example of the 1GbE Management Switch deployment. Deployment specifics can change from release to release. Always check the ESS Quick Deployment Guide for the information for your ESS implementation. At the time of this writing, you can consult the IBM ESS Version 6.2.2 ISS Deployment Guide.

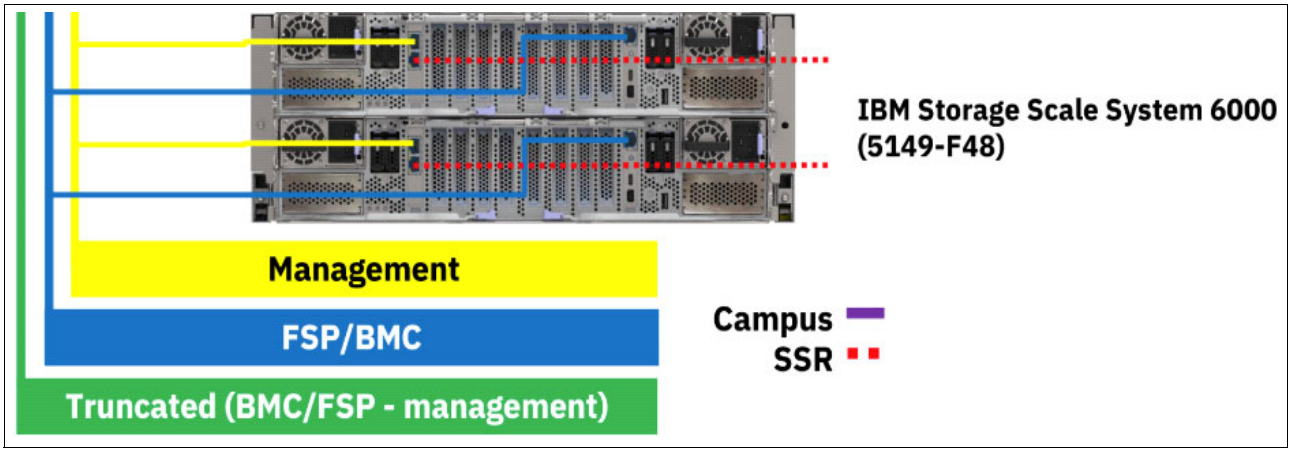


Figure 3-1 Overview of the management network and ports

All the ISS 6000 server canisters must be connected to ports 1 - 12 only. In case you have already a management switch that was ordered before you ordered the ISS 6000 and does not have the new management switch configuration (v2), you can ask IBM TSS to convert the switch to be usable by ISS 6000 systems. You can also convert the switch yourself with the instructions in Appendix A, “Configuring the 48 ports top of the rack management network switch” on page 119.

Dual 24 port (48 ports) management switch

When the management TOR is included in an order, you will receive two of the 24 port management switches. The reason to deliver two instead of one is to keep similar number of ports available as with the 48 ports switch option. The switches appear as seen in Figure 3-2.

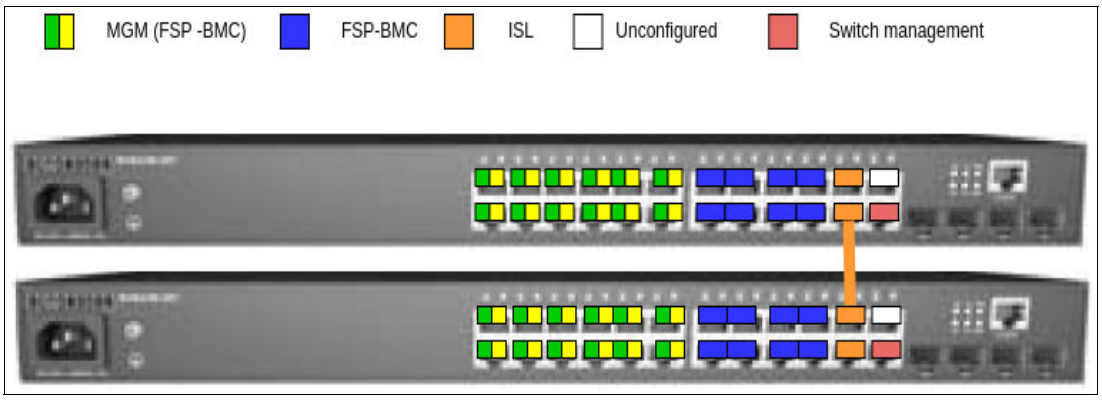


Figure 3-2 Dual 24 port management switch

The orange-colored cable shown in Figure 3-2 must be connected between port 22 of the upper switch and 21 of the lower switch as part of the configuration. That cable works as inter-switch link (ISL) between the two switches.

High-speed network

As with any other IBM Spectrum Scale and IBM ESS configurations, the ISS 6000 requires a high-speed (HS) network to be used as the data-storage cluster network. In some product documentation, this network is referred to as a clustering network. The hardware for the HS network can be provided by IBM or by the customer. If the hardware is provided by the customer, it must be compatible with the network interfaces that the ISS 6000 supports. See 2.1.1, “Canisters and servers” to see the available network options on ISS 6000.

ESS Management Server

The ISS 6000 requires an IBM Storage Scale System Utility Node (EMS), IBM machine 5149-23E. The EMS is required on standalone installs. If the ISS 6000 is added to an existing IBM ESS IBM Spectrum Scale cluster that already has a IBM Storage Scale System Utility Node (EMS), then the existing EMS can support the ISS 6000. The minimum release level on the
ISS 6000 canisters is ESS 6.1.9.0.

For more details about the EMS server, see IBM Documentation for 5102-22E.

Notes: If your previously-installed IBM Spectrum Scale or ESS configuration uses a previous-generation POWER9 EMS, you must add a IBM Storage Scale System Utility Node (EMS) to support the ISS 6000.

The IBM or IBM Business Partner team uses the FOSde tool and IBM eConfig for Cloud to configure the EMS. IBM eConfig for Cloud configures the EMS with the appropriate network cards such that the EMS can connect to the same HS networks that are configured on the ISS 6000.

The default IBM ESS Management Server memory size is enough for most IBM ESS installations. If many IBM ESSs are used in your IBM Spectrum Scale IBM ESS configuration, check with your IBM representative to see whether larger IBM ESS Management Server

memory sizes might be required for your installation. More EMS memory can be specified at order time or added later as a field Miscellaneous Equipment Specification (MES).

Best practices for deploying and optimizing a stand-alone ISS 6000

A standalone ISS 6000 unit, which is known as a building block, must minimally consist of the following components:

- One EMS node in a 2U form factor
- One ISS 6000 node in 4U form factor
- 1 GbE Network switch for management network (1U)
- 100 or 200 Gb high-speed IB or Ethernet network for internode communication (1U)

The EMS node acts as the administrative end point for your ISS 6000 environment. It performs the following functions:

- Hosts the IBM Spectrum Scale GUI
- Hosts Call Home services
- Hosts system health and monitoring tools.
- Manages cluster configuration, file system creation, and software updates
- Acts as a cluster quorum node

The ISS 6000 features a new container-based deployment model that focuses on ease-of-use. The container runs on the EMS node. All of the configurations tasks that were performed by the **essutils** utility in legacy ESS are now implemented as Ansible Playbooks that are run inside of the container. These playbooks are accessed using the **essrun** command.

The **essrun** tool handles almost the entire deployment process, and is used to install software, apply updates, and deploy the cluster and file system. Only minimum initial user input is required, and most of that is provided during the TDA process before setting up the system. The **essrun** tool automatically defines system parameters to optimize performance of the single ISS 6000 system. File system parameters and IBM Spectrum Scale RAID Erasure code selection can be customized from their defaults before file system creation.

For more information about deployment customization, see the ESS Deployment Guide.

These are some of the best practices:

- Refrain from running admin commands directly on the ISS 6000 I/O canisters. Use the EMS node instead.
- Do not mount the file system on the ISS 6000 I/O canisters because this consumes additional resources. The file system must be mounted on the EMS node for the GUI to function properly.
- To access the file system managed by the ISS 6000 building block, you must use external GPFS client nodes or protocol nodes.
- On a single building block deployment, the I/O canister nodes are specified as GPFS cluster or file system manager nodes while the EMS node is not. Although the EMS node is considered the building block's primary management server, avoid specifying the EMS node as a manager node. The GPFS management role is an internal designation that was previously the manager of the cluster and the file system and does not directly affect the function of the EMS node.

3.1.3 Software planning

The ISS 6000 provides an integrated, tested ESS solution software stack that includes the following features:

- Embedded Red Hat Enterprise Linux license
- Firmware drivers for the Mellanox network cards
- IBM Spectrum Scale
- All necessary supporting software

IBM supports the ESS software stack as a solution.

When installing ESS software updates, each installation cannot be done in the same way. Each installation has different operational and non-operational requirements that can impact what is possible to achieve and when and how often is possible to do software updates.

Ideally, you would update systems at least once a year, but for some installations, annual updates are not possible because of legal certification reasons, operational, or other reasons. Consequently, updates might occur once every three or more years.

IBM strongly recommends that the following key points are followed when doing software currency on ESS-related environments:

- Never do more than N-3 jump of an ESS-software update. Do intermediate jumps, if needed, to maintain this rule.
- Always update the EMS first.
- Perform offline updates when possible. If online update is a requirement, explore the **`-serial`** option to limit the risk exposure in case some nodes experience problems during the update.
- If you encounter a problem, contact IBM Support. Resolving the problem without the help of support might fix the problem in the near term, but could create future issues due to the automation expecting the configuration to be a certain way. So, we recommend that you stabilize the environment and then contact IBM Support.
- Always keep the ESS cluster in the same level. You can update different systems separately, but all should be running the same version after the updates complete. If that is not possible, consider partitioning your backend cluster to achieve this rule.
- Use defaults, unless you have a specific reason to not do so.

3.1.4 Network planning

The ISS system includes certain network names. To avoid confusion, those networks are defined in this section.

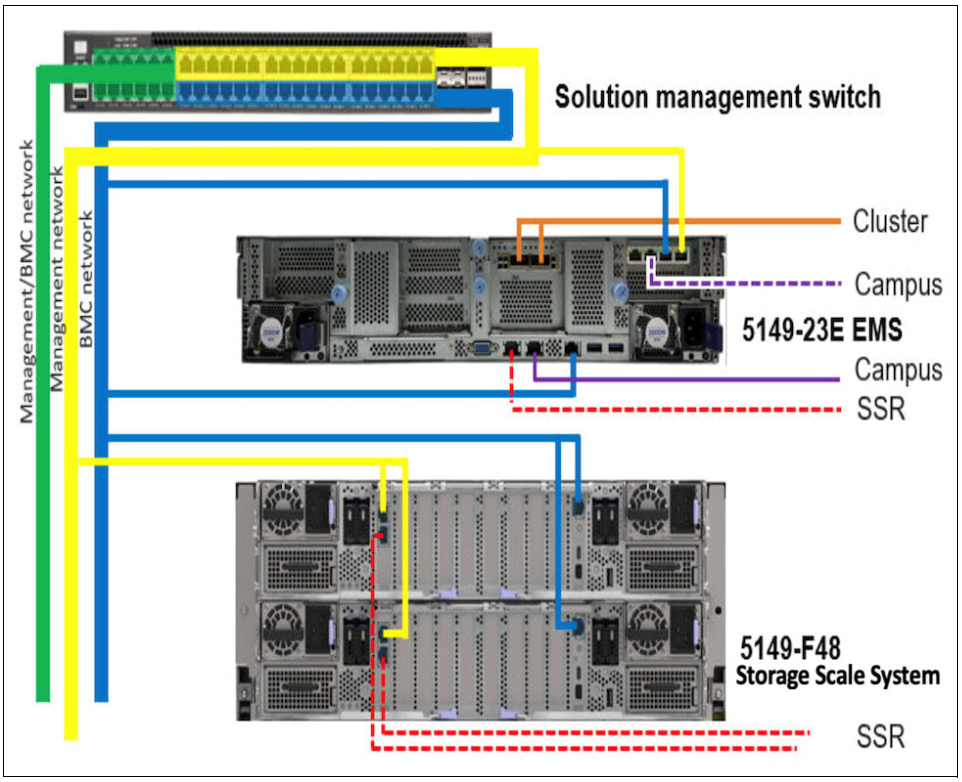


Figure 3-3 ISS networking overview

Management network

The *management network* is a non-routable private network. It connects the EMS PCI card slot 5 (C5) port 1 (T1).

Through this network, EMS, and containers on the EMS, manage the OS of the I/O nodes. This network cannot use VLAN tagging of any kind, so it must be configured as an access VLAN on the switch. You can choose any netblock that fits your needs, but as a best practice use a /24 block. If you have no preference, use the 192.168.45.0/24 block because it is the one that is used in most of the documentation examples.

Flexible service processor network

The *flexible service processor (FSP) network* is a non-routable private network. It connects the EMS C5 T2 with each out-of-band management ports of the I/O nodes that are labeled as “HMC 1" or “BMC”. That includes the EMS that has a connection to the BMC from this network and any other ESS node running on a BMC supported platform in the cluster managed by the same EMS.

The EMS and the containers running on the EMS use this network to do FSP operations and BMC operations on the physical servers, which include powering-on and powering-off the servers.

If you do not have a preference, use 172.16.0.0/24 for the same reasons as described for the management network.

Campus network

The campus network is also termed as public or external network that connects to C5-T3 on the EMS node and bottom T3. This connection serves as a way to access the GUI or the ESA agent (call home) from outside of the management network. The container creates a bridge to the management network, thus having a campus connection is highly advised.

IBM Storage Scale System Utility Node (EMS) campus connection must be set prior to deployment (Bottom-T3). This allows remote access to the EMS and ensures you will not lose a connection when starting the container. This will allow remote access to the BMC which aids the recovery of the node (console/power control) in case of an outage.

High-speed network

The HS data network is where the IBM Spectrum Scale daemon and admin networks should be configured. It is a customer-provided and customer-managed network.

Network design for a parallel file system can be complex. The HS network design and implementation is usually the deciding factor on what the overall performance your system delivers. Unfortunately, there is no single design that fits every use case.

Here are some design ideas that must be considered:

- The IBM Spectrum Scale admin network has the following characteristics:
  - Used for the running of administrative commands
  - Requires TCP/IP
  - Can be the same network as the IBM Spectrum Scale daemon network or a different one
  - Establishes the reliability of IBM Spectrum Scale
- The IBM Spectrum Scale daemon network has the following characteristics:
  - Used for communication between the `mmfsd` daemon of all nodes.
  - Requires TCP/IP.
  - In addition to TCP/IP, IBM Spectrum Scale can be optionally configured to use Remote Direct Memory Access (RDMA) for daemon communication. TCP/IP is still required if RDMA is enabled for daemon communication.
  - Establishes the performance of IBM Spectrum Scale, as determined by its bandwidth, latency, and reliability of the IBM Spectrum Scale daemon network.

In cases where the HS data network is Ethernet based, it is a best practice to place the daemon and admin network on the HS Ethernet network.

If you have InfiniBand networks, you can use Ethernet adapters on the HS network if they are available, or you can use an IP over InfiniBand encapsulation.

The Management or FSP network should not be part of the IBM Spectrum Scale cluster as a management or daemon network.

Sizing the network

With the networking information described in this section, perform the following sizing exercise by considering two parameters:

1. The expected client-required throughput performance and number of client-ports with their aggregated performance
2. The number of ISS 6000 or other ESS I/O nodes or canister aggregated performance.

Include any inter switch links (ISLs) that are in place as well as PCIe speeds and feeds for each system. As an example, consider a simple two HS IB 200 Gbit ports scenario, where each ISS 6000 includes eight ports connected. Assuming there are PCI3 Gen 4 x16 lines on the clients and 200 Gbit IB ports connected (high dynamic range (HDR)), it is an ideal scenario to have up to eight of those clients and four ISLs between switches. If you increase the demand on the network beyond that, then some network oversubscriptions might occur, which could have a negative impact on your workload.

IBM SSR network port

The IBM System Services Representative (SSR) network port on the IBM EMS is on Bottom port T1. This port should never be cabled or connected to any switch as it is only for IBM field engineers to use. This port is configured on the 10.111.222.100/30 block.

The ESS canisters do not have direct Ethernet connectivity through the SSR port. The IBM SSR accesses the device through a serial cable on each canister.

Note: The management network, flexible processor network and high-speed network are required to be reachable by all nodes in the cluster including the EMS servers. If the systems are located in different racks, rooms or even data centers (called a stretch cluster). All nodes in the stretch cluster must be able to reach these networks from all racks, rooms and data centers to be a supported configuration. It is not supported for a node or EMS in one site to only have access to the nodes of its own site and vice versa.

3.1.5 ESS Management Server considerations:

The ISS 6000 requires a IBM Storage Scale System Utility Node (EMS), IBM machine type 5149-23E. The EMS is required on standalone installations. If the ISS 6000 is added to an existing IBM Spectrum Scale or ESS cluster that already has a POWER9 EMS, the existing EMS can support the ISS 6000.

Note: If a previously-installed IBM Spectrum Scale or ESS configuration uses a previous-generation IBM POWER8 EMS, you must add an IBM POWER9 EMS to support the ISS 6000. IBM does not support re-purposing an existing server, VM or LPAR to be used as the EMS.

For more details about the EMS server, see the Documentation for 5149-23E.

IBM or the IBM Business Partner team uses the FOSde tool and IBM eConfig for Cloud to configure the EMS. eConfig configures the EMS with the appropriate network cards such that the EMS can participate in the same high-speed networks that are configured on the ISS 6000.

The default IBM EMS memory size is sufficient for most IBM ESS installations. If a large number of IBM ESS enclosures are used in your IBM Spectrum Scale IBM ESS configuration, then check with your IBM representative to see whether an IBM EMS with more memory might be required for your environment. More EMS memory can be specified at order time or added later, as a field miscellaneous equipment specification (MES).

3.1.6 Skills and services

Installation and support of the ISS 6000 requires several skills:

- Red Hat Enterprise Linux

- TCP/IP and high-speed networking
- IBM Spectrum Scale

IBM and IBM Business Partners can provide education courses and services to teach or improve these skills.

Customers and IBM Business Partners also have the ability to engage IBM Systems Lab Services, which are available and recommended, to provide customized help in integrating ISS 6000 into a new or existing client environment.

3.2 Standalone environment

This section describes best practices for deploying and optimizing a standalone ISS 6000.

A standalone ISS 6000 unit, which is known as a *building block*, must minimally consist of the following components:

- One EMS node in a 2U form factor
- One ISS 6000 node in 4U form factor
- 1 GbE Network switch for management network (1U)
- 100 or 200 Gb high-speed IB or Ethernet network for internode communication (1U)

The EMS node acts as the administrative end point for the ISS 6000 environment. It performs the following functions:

- Hosts the IBM Spectrum Scale GUI
- Hosts Call-Home services
- Hosts system health and monitoring tools.
- Manages cluster configuration, file system creation, and software updates
- Acts as a cluster quorum node

The ISS 6000 features a container-based deployment model that focuses on ease-of-use. The container runs on the EMS node. All of the configurations tasks that were performed by the **`gssutils`** utility in legacy ESS are now implemented as Ansible Playbooks that are run inside of the container. These playbooks are accessed using the **`essrun`** command.

The **`essrun`** tool handles the majority of the deployment process, and is used to install software, apply updates, and deploy the cluster and file system. Only minimum initial user input is required, most of which is information covered by the TDA process which is required to be completed before setting up the system. The **`essrun`** tool automatically configures performance-related parameters to get the most out of a single ISS 6000 system. File system parameters and IBM Spectrum Scale RAID Erasure code selection can be customized from their defaults before file system creation.

For more information about deployment customization, see the Quick Deployment Guide.

The following items are considered best practices:

- Refrain from running admin commands directly on the ISS 6000 I/O canisters. Use the EMS node instead.
- The file system must be mounted on the EMS node for the GUI to function properly. Do not mount the file system on the ISS 6000 I/O canisters as this consumes additional resources.

- To access the file system managed by the ISS 6000 building block, you must use external GPFS client nodes or protocol nodes.
- On a single building block deployment, the I/O canister nodes are specified as GPFS cluster and file system manager nodes while the EMS node is not. Although the EMS node is considered the building block's primary management server, avoid specifying the EMS node as a manager node. The GPFS-management role is an internal designation that, previously, was the manager of the cluster and the file system and does not directly affect the function of the EMS node.

3.3 Mixed environment

This section provides information about integrating the ISS 6000 into an existing IBM Storage Scale environment. This includes additional considerations when integrating into a mixed-vendor environment for migration purposes, or for using IBM ESS 3200 as the HS storage tier.

3.3.1 Adding the ISS 6000 to an existing Storage Scale cluster

The following guidance is for adding a Standalone ISS 6000 building block into an existing ESS cluster or into an existing IBM ESS 3500, IBM ESS 3200, IBM 3000 or IBM ESS 5000.

Prerequisites and assumptions

There are prerequisites and assumptions for adding an ISS 6000 to an existing Storage Scale cluster:

- Existing IBM ESS 3500 cluster is connected or reachable to the same HS network block.
- Existing IBM ESS 3000, IBM ESS 5000, IBM ESS 3200 or IBM ESS 3500 is connected or reachable to the same management low-speed network block.
- IBM ESS 3000 contains a IBM Storage Scale System Utility Node (EMS) node and it is running Podman container.
- IBM ESS 5000 contains a IBM Storage Scale System Utility Node (EMS) and it is running Podman container.
- IBM ESS 3200 contains a IBM Storage Scale System Utility Node (EMS) and it is running Podman container.
- IBM ESS 3500 contains a IBM Storage Scale System Utility Node (EMS) and it is running Podman container.
- ISS 6000 nodes were added to `/etc/hosts` which contains the same information on POWER9 EMS and/or an IBM Storage Scale System Utility Node (EMS):
  - Low-speed names: fully qualified domain names (FQDNs), short names, and IP addresses
  - High-speed names: FQDNs, short names, and IP addresses (add suffix of low-speed names)
- Host name and domain is set in an IBM Storage Scale System Utility Node (VM).
- Latest code for IBM ESS 3000 and IBM ESS 5000 stored in `/home/deploy` on POWER9 EMS and/or an IBM Storage Scale System Utility Node (VM).
- Linux root password is common across all of the nodes (Legacy, IBM ESS 3000, IBM ESS 5000, IBM ESS 3200, IBM ESS 3500 and ISS 6000).

Adding ISS 6000 to an IBM ESS 5000 cluster

Run the **config load** command within the ESS container that is running in the IBM Storage Scale System Utility Node (VM) to fix the SSH keys across all of the nodes.

Example 3-1 Run config load with ISS 6000

```
ESS UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N s6kNode1,s6kNode2,
,EMSVM,ESS5000Node1,ESS5000Node2 config load -p
```

Create bonds in ISS 6000 building block within ESS container that is running in the IBM Storage Scale System Utility Node (VM).

Example 3-2 Create network bonds

```
ESS UNIFIED v6.2.0.0 [root@cems0 /]# essrun -N s6kNode1,s6kNode2 network
--suffix=Suffix
```

Add ISS 6000 I/O nodes to the existing IBM ESS 5000 cluster from within the ESS container that is running in the IBM Storage Scale System Utility Node (VM).

Example 3-3 Add ISS 6000 nodes to ESS 5000 cluster

```
ESS UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N ESS5000Node1 cluster
--add-nodes s6kNode1,s6kNode2 --suffix=Suffix
```

Adding ISS 6000 to an ESS 3000 cluster

Run the **config load** command within ESS Unified container that is running in the IBM Storage Scale System Utility Node (VM) to fix the SSH keys across all of the nodes.

Example 3-4 Run config load with ESS 3000

```
ESS UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N
s6kNode1,s6kNode2,EMSVM,ESS3000Node1,ESS3000Node2
config load -p
```

Create bonds in ISS 6000 building block within ESS unified container that is running in the IBM Storage Scale System Utility Node (VM). See Example 3-9.

Example 3-5 Create network bonds

```
UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N S6kNode1,s6kNode2 network
--suffix=Suffix
```

Add ISS 6000 I/O nodes to existing the ESS 3000 cluster from within ESS 3000 container that is running in the IBM Storage Scale System Utility Node (VM).

Example 3-6 Add ISS 6000 nodes to IBM ESS 3000 cluster

```
UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N ESS3000Node1 cluster
--add-nodes s6kNode1,s6kNode2--suffix=Suffix
```

Adding ISS 6000 to a mixed ESS cluster (IBM ESS 3000, IBM ESS 3500 and IBM ESS 5000)

Run the **config load** command within ISS 6000 container that is running in the IBM Storage Scale System Utility Node (VM) to fix the SSH keys across all of the nodes.

Example 3-7 Running config load with ESS 3500 + ESS 3000 + ESS 5000

```
ESS UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N
ESS3500Node1,ESS3500Node2, ESS5000Node1,ESS5000Node2,ESS3000Node1,
ESS3000Node2,s6kNode1,s6kNode2,EMSVM
config load -p
```

Create bonds in ISS 6000 building block within the container that is running in the IBM Storage Scale System Utility Node (VM). See Example 3-13.

Example 3-8 Creating network bonds

```
ESS UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N s6kNode1,s6kNode2 network
--suffix=Suffix
```

Add ISS 6000 I/O nodes to existing IBM ESS 3000 cluster, IBM ESS 5000 and IBM ESS 3500 from within container that is running in the IBM Storage Scale System Utility Node (VM). See Example 3-14.

Example 3-9 Adding ISS 6000 nodes to IBM ESS 5000 cluster

```
ESS UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N ESS3500Node1 cluster
--add-nodes s6kNode1,s6kNode2 —suffix=Suffix
```

Adding ISS 6000 to an IBM ESS 3500 cluster

Run the **`config load`** command within the ISS 6000 container that is running in the IBM Storage Scale System Utility Node (VM) to fix the SSH keys across all of the nodes.

Example 3-10 Defining SSH keys across the nodes

```
ESS UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N
ESS3500Node1,ESS3500Node2,EMSVM, s6kNode1,s6kNode2 config load -p
```

Create bonds in ISS 6000 building block within ISS 6000 container that is running in the IBM Storage Scale System Utility Node (VM).

Example 3-11 Creating network bonds

```
ESS UNIFIED v6.2.0.0 [root@cems0 /]# essrun -N s6kNode1,s6kNode2 network
--suffix=Suffix
```

Add ISS 6000 I/O nodes to the existing IBM ESS 3200 cluster from within an ESS container that is running in the IBM Storage Scale System Utility Node (VM).

Example 3-12 Adding ISS 6000 I/O nodes to an existing IBM ESS 3200 Cluster

```
ESS UNIFIED v6.2.0.0 CONTAINER [root@cems0 /]# essrun -N ESS3500Node1 cluster
--add-nodes s6kNode1,s6kNode2--suffix=Suffix
```

3.3.2 Scenario 1: Using ISS 6000 for metadata network shared disks

This section describes how to use ISS 6000 for metadata network shared disks (NSDs) with the existing file system.

This scenario contains an existing IBM ESS 3500 cluster and file system deployed from a IBM Storage Scale System Utility Node (VM).

The steps for the high-level plan are as follows:

1. Deploy the ISS 6000 container into the IBM Storage Scale System Utility Node (VM).
2. Add the ISS 6000 Building Block to the cluster.
3. Create the ISS 6000 VDisk set as metadataOnly.
4. Add the ISS 6000 VDisk set to the existing IBM ESS 3500 file system.

The following steps provide guidance to set up the ISS 6000 for metadata NSDs for the existing file system:

1. Deploy the ISS 6000 container into the IBM Storage Scale System Utility Node (VM): Log in to the IBM Storage Scale System Utility Node (VM) and completing the ESS common installation instructions from the Quick Deployment Guide.
2. Add the ISS 6000 Building Block to the cluster: From step 1, within the container, run the Ansible **essrun** command to add the new ISS 6000 to the IBM ESS 3500 cluster. The parameter “essio1” is an example name of an existing ESS I/O node in the cluster:

   Example 3-13

   ```
   root@cems0:/ # essrun -N essio1 cluster --add-nodes CommaSeparatedNodesList
   --suffix=-hs
   ```

3. Create the ISS 6000 VDisk set as metadataOnly:

   Within the container, run the Ansible **essrun** command to create the new ESS VDisk set (using 16M as block size, because IBM ESS 3500 file system is the default one):

   Example 3-14

   ```
   root@cems0:/ # essrun -N ess32001a,ess32001b vdisk --name newVdisk --bs 16M
   --suffix=-hs --extra-vars "--nsd-usage metadataOnly --storage-pool system"
   ```

4. Add the ISS 6000 VDisk set to the existing IBM ESS 3500 file system.

   From the IBM Storage Scale System Utility Node (VM) node, use **mmvdisk** command to add the new VDisk to the existing file system:

   Example 3-15

   ```
   [root@emsvm ~]# mmvdisk filesystem add --file-system filesystemName --vdisk-set
   vs_newVdisk
   ```

3.3.3 Scenario 2: Using ISS 6000 to create a file system

To create a file system with ISS 6000, the IBM Storage Scale System Utility Node (VM) container must be running.

After the container is running and the cluster and recovery groups are created, create the file system by running the **essrun** command in which “essio1” and “essio1” are example names of existing IO nodes:

```
$ essrun -N essio1,essio2 filesystem --suffix=-hs
```

Note: This command creates vdisk sets, NSDs, and file systems by using **mmvdisk**. The defaults are 4M blocksize, 80% set size, and 8+2p RAID code. These values can be customized by using additional flags.

For CES deployment, the ISS 6000 system should have a CES file system. To create the CES file system, run the following **essrun** command:

```
$ essrun -N essio1,essio2 filesystem --suffix=-hs --name cesSharedRoot --ces
```

Note: A CES and other file systems can coexist on the same IBM ESS cluster.

4

IBM Storage Scale 6000 Configuration

This chapter provides an introduction to the different IBM Storage Scale 6000 configurations available and covers the following topics:

4.1 Overview

In this chapter, we use a hands-on and planning approach, to outline the different configurations supported with the IBM Storage Scale (ISS) 6000. At the time of writing there are 4 supported configuration types: Performance, FCM, Capacity and Hybrid.

When comparing the ISS 6000 to IBM's previous system, the ESS 3500, the ISS 6000 doubles the maximum number of NVMe SSD drives from 24 to 48 while increases its size from 2U to 4U.

The 6000 also introduces the Flash Core Module 4.1 (FCM) into the ISS family. Developed by IBM, FCM is a thin provisioned, in-flight compression device that can significantly expand the usable capacity and density in the ISS family when implemented with proper planning considerations. The most important of which will be discussed later in this chapter.

For details on FCM, we recommend the IBM Redbook publication "IBM FlashCore Module (FCM) Product Guide" which can be found at the following link:
https://www.redbooks.ibm.com/redpieces/pdfs/redp5725.pdf

4.2 Performance configuration

In this section we cover the ISS 6000 performance configuration. This was the first configuration available with the ISS 6000 and consists of NVMe SSD Gen4 U2.2 interface drives, focused on providing consistent and high throughput to feed high demand data requirements.

4.2.1 Performance and sizing

At the time of writing, the supported performance configurations are 24 or 48 NVMe SSD drives in one ISS 6000 building block. When choosing the 24 drive versions, the other 24 slots contain fillers to provide for optimal cooling. It can be online extended from 24 to 48, but the size of the new 24 must be identical to the already installed 24.

The synthetic expected performance of an ISS 6000 would depend on multiple factors and must take into account many variables including the number of drives (24 or 48), the network connectivity (Number of ports and TCP or RDMA), the number and type of clients and the workload. The sales material performance numbers as measured on the clients, not the backend, are shown in the following table:

Table 4-1 ISS 6000 base performance per number of drives

Number of Drives	100% Read	100 Write
24	90 GB/s	51.4 GB/s
48	171 GB/s	112 GB/s

Note: These results are overall simplified and represent the expected or commonly achievable results for each configuration.

These numbers might vary depending on your environment. The ISS 6000 performance configuration has been certified at the NVIDIA SuperPOD "Best" level. During certification testing, a single ISS 6000 provided (as measured by the GPU clients) 310 GB/sec read performance and 13 million IOPS. This was then able to scale up to ten units which provided a nice round number of 3 TB/sec read performance as measured at the clients. For more information, see the following link for additional details:

https://community.ibm.com/community/user/storage/blogs/mike-kieran/2024/08/29/ibm-storage-scale-system-6000-receives-nvidia-dgx

In terms of capacity, the drive size options for ISS 6000 are: 3.89 TB, 7.68 TB, 15.36 TB and 30.72 TB. It is not possible to mix sizes within a single ISS 6000 and it would not be recommended to mix across different ISS 6000 if they are serving in the same storage pool.

The following table provides the usable capacity for a single filesystem and 8MB block size and 8+2p RAID for each disk size and capacity option.

Table 4-2 ISS 6000 usable capacity 8+2p single filesystem 8MB block size

SSD Size	Usable Capacity 24 SSDs	Usable Capacity 48 SSDs
3.89 TB	48 TB	98 TB
7.68 TB	98 TB	197 TB
15.36 TB	197 TB	395 TB
30.72 TB	395 TB	793 TB

If you need more details or different capacities, contact an IBM technical sales representative or use the IBM public ECE capacity tool. This tool works with ISS, as explained in the README. Use half the drives per server (which is 2 in ISS). You can find the tool on the following link:

https://github.com/IBM/SpectrumScale_ECE_CAPACITY_ESTIMATOR

Note: This tool is provided as is and not supported by IBM.

4.3 IBM Flash Core Module 4 configuration

In this section we describe and detail the FCM option for ISS 6000, its peculiarities and its differences versus the other storage type options.

4.3.1 Introduction to IBM Flash Core Module (FCM)

IBM Flash Core Module (FCM) are a family of high-performance flash drives made available in a standard 2.5", 15 mm form factor.

FCM uses the NVMe protocol, a PCIe Gen4 U.2 interface, and high-speed NAND memory to provide high throughput and IOPS with consistent and predictable latency. These devices have been a member of the IBM family of block storage devices for over ten years and are now also available for use with GNR when included in an ISS 6000.

Within the ISS 6000 family, current FCM 4 usable sizes are configurable as 19.2 TB, and 38.4 TB, giving an effective storage size of 57.6 TB and 115.2 TB relative to the selected drive sizes.

For more details on FCM technology check the link to the IBM Redbook mentioned in the overview subsection, 4.1, “Overview” on page 80.

4.3.2 FCM and GPFS Native RAID (GNR)

As with any other type of drive in GNR, all the sizes of the drives must be of the same size. When planning for an implementation of GNR ensure you carefully choose the size of the drive before purchase. Also at this time, it is not supported to have 24 FCM and 24 NVMe SSD in the same ISS 6000 building block.

Currently, the supported configurations are 24 or 48 FCM in one ISS 6000 building block. When choosing 24, the other 24 slots are empty and filled with fillers to accommodate for optimal cooling.

Due to the inline compression of the FCM devices, determining appropriate sizing becomes a more thoughtful exercise than with other ISS storage configuration options. Sizing this configuration must also treat the FCM devices as thin provisioning devices, while the rest of the options are all thick provisioning.

Note: At the time of writing with ISS version 6.2.2.1 or higher, for best results it is suggested to use all FCM ISS systems for a data safe run.

4.3.3 Performance and sizing with FCM

While there is a long running joke that "it depends" when asked to determine what the right amount of storage for an environment, this has never been more true than with FCM. Sizing the environment is the most complex part of using FCM in GNR, as it directly impacts performance.

The performance with FCM depends directly on the compressibility of the data that is being moved. FCM devices compress data at physical block level, not filesystem level. Even data already compressed at the software level, can get some additional compression at this lower level. The better the compressibility of the data, the better the overall performance. With a 2:1 or better compression ratio, it’s possible to reach over 145 GB/sec read and 55 GB/sec write using FCM, while on the corner case of no compressibility at all results in around 90 GB/sec read and 37 GB/sec write.

Note: The performance numbers stated are for Interleaved or Random (IOR) synthetic workloads using RDMA and enough client nodes to drive the performance.

TRIM (or UNMAP for SCSI) is also very important for FCM as it is for thick SSD devices.

It is important to understand that several elements use very little or no physical storage space: spare space, unallocated metadata, empty filesystem space, and any areas where no real data is written to the device. GNR must account for all of these factors when using FCM.

At the time of the writing, FCM on ISS 6000 will hold some internal reservations of space and tiers of performance based on the usable (physical) utilization of the FCM. When a drive is

added to a recovery group, 13% percent of the usable (physical) size of FCM is reserved by GNR for critical recoveries, leaving the remaining 87% for vdisk use.

An additional 17% of disk space is not allowed to be used and all IO writes will be blocked when reaching that threshold. This leaves 70% of usable (physical) capacity for actual usable storage space in user vdisks.

When reaching 55% of the usable capacity (physical space) you should get a message of warning about the utilization threshold, no IO throttling will take place at this time. At 60% of the usable capacity (physical space) all IO writes will be throttled to 20GB/sec by the ISS 6000. When 65% of capacity (physical space) is reached, all IO writes will be further throttled to 2GB/sec. If the 70% threshold is reached, it would then no loner be possible to perform any writes to the devices.

All of the above information can be summarized visually in the following figure:

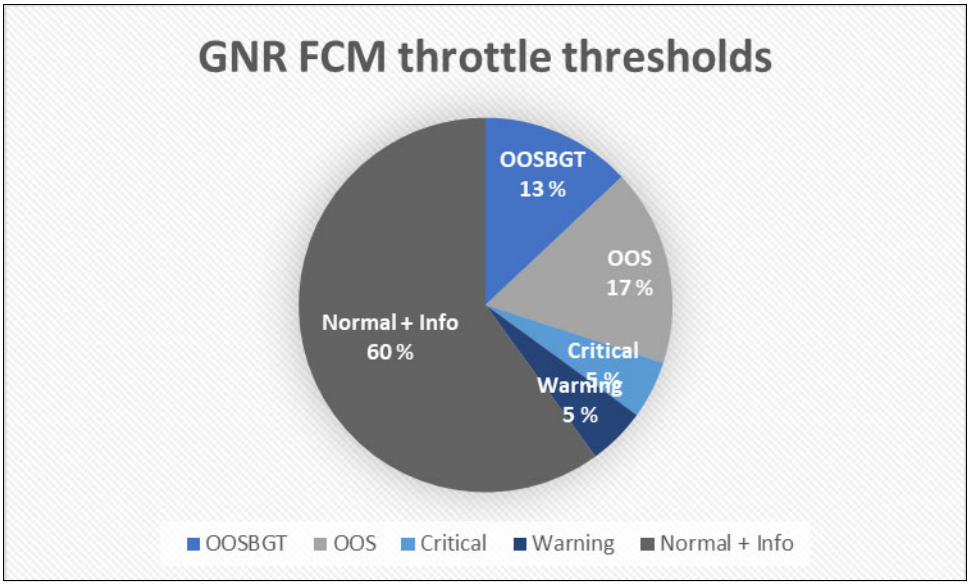


Figure 4-1 FCM GNR storage space and throttle thresholds

How this translates into real sizing is a bit more complex, but the following table can be used as general guidelines related to the physical sizes that FCM has versus what is presented as usable:

Table 4-3 Usable (physical) capacity at different thresholds

Usable size (Physical)	70%	65%	60%
19.2 TB	13.44 TB	12.48 TB	11.52 TB
38.4 TB	26.88 TB	24.96 TB	23.04 TB

To this point, this publication has not yet mentioned effective capacity at the FCM level nor at GNR level. The main reason for this is, that it is less relevant to FCM than on thick devices, the most important metric for FCM sizing is the usable (physical) utilization.

The metric with the second most impact is the compression ratio of the data in the FCM, which translates data usage into usable (physical) utilization. When the compression ratio is better than 3:1 (which we will explain further), additional benefits are realized.

The effective size reported from FCM is 21.99 TB, 28.8 TB, 57.6 TB, or 115.2 TB, depending on the specific FCM model installed in the system.

For this example sizing exercise we are going to use a single ISS 6000 with 48 FCM of usable (physical) size of 38.4 TB, and usable reported size of 115.2 TB of each FCM. After creating a RG and DA, and some vdisks it reports as follows on mmvdisk rg show command:

```
      capacity        phy capacity
total raw free raw    total     free
--------- --------  --------- --------
 4803 TiB 1934 TiB   1676 TiB 1312 TiB
```

It is important not to jump to the conclusion that because the effective capacity, reported just as "capacity" has already used 2869 TiB, and the usable (physical) capacity (reported as "phy capacity" of 364 TiB) that we are getting a compression of near 8:1. This is not what is happening.

The free effective capacity (reported as "capacity") gets allocated and shown as used when the vdisksets are created, regardless the compression you might get from the FCM engines. It is static and never changes over the life of those vdisksets.

The free usable (physical) capacity (reported as "phy capacity") is dynamic and gets updated as data gets written into the FCM. Which results in this being the metric that must be monitored to ensure the system does not reach the throttle thresholds mentioned previously.

The reader might have observed that the ratio between usable (physical) and effective sizes on the FCM device is ~3:1, which means that if your data gets better compression than 3:1 it is not going to be usable as you can never get more vdiskset size (vdisks) into the FS than the effective capacity that GNR is presenting you and that is hardcoded to a max of 3:1. This is for an aggregate of all data, if some data has better than 3:1 and other data less, what matters is the average of the compression ratio lands at a near 3:1 ratio.

As you can imagine the compressibility of the data is a critical factor for both the performance and the sizing of ISS 6000 system using FCM. Unfortunately, at the time of writing, IBM does not provide a tool to provide the compressibility ratio of data. Making the sizing exercise on ISS 6000 FCM rather complex without access to actual hardware or ability to share some data with IBM to get the real expected compression ratios.

Having a way to get the real compression ratio as reported from the FCM device is also critical. While you might be able to get this information by using the compression ratio from the mmvdisk rg show command, there are certain factors like TRIM, that would cause this output to deviate from the real ratio. At the time of publishing, the only way to get an exact ratio is from FCM drive dumps, which can be created by IBM to tell you the actual ratio.

Last but not least, on thick NVMe SSD devices there is the rule of never go over 80% of vdiskset size, which on FCM is the effective capacity, to leave physical space for TRIM and other SSD garbage collecting operations of the SSD.

Important: To use ISS 6000 with FCM and TRIM. It is extremely important, specially if adding it to already running clusters, to check with IBM support and confirm that all nodes which mount the filesystem are at the Storage Scale software version that has TRIM fixes to avoid having data corruption in your filesystem. At the time of the publication, 5.1.9.8 or 5.2.2.1 or higher releases should be safe for all clients.

On ISS 6000 with FCM this may result in 20% of free usable (physical) capacity while still use more than 80% of the effective capacity on GNR. This is dependent on the actual compression ratio that your data is getting inside the FCM devices.

If you cannot estimate your expected compression when planning your system sizing, it is suggested to be conservative and size for and expected 1.5:1 to 2:1 compression ratio for binary data, and then, over time get real data from your devices.

The following tables can be used along with the information in this chapter to perform a sizing exercise for ISS 6000 FCM system. In the tables below, we continue with the single ISS 6000 48 FCM with a usable (physical) size of 38.4 TB, and usable reported space of 115.2 TB. these examples numbers can be adapted to numbers for the different sizes of FCM that are available for ISS 6000.

Remember that the maximum we can allocate on vidiskset is what GNR presents of 4803 TiB, or 80% of that if the 20% physical free is not reached, 3842.4 TiB.

For the case of 60% percent of usable (physical) space with no throttle applied, the Table 4-4 showing different compression ratios applies:

Table 4-4 Drive capacity with compression and 60% usable capacity

Usable to Effective Ratio	Usable Space Used (TiB)	Effective Space Used (TiB)
1:1	1005.6	1005.6
1.5:1	1005.6	1508.4
2:1	1005.6	2011.2
2.5:1	1005.6	2514.0
3:1	1005.6	3016.8
3.5:1	1005.6	3520.4
4:1	1005.6	4022.4 (3842.4)
4.5:1	1005.6	4525.2 (3842.4)
5:1	1005.6	5028.0 (3842.4)

In the worst case of 1:1 only 1005.6 TiB can be allocated as vdiskset, any more it would result in IO writes being throttled. For a conservative 1.5:1 we could go up to 1508.4 TiB, 2:1 2011.2 TiB and so on.

Using any further storage space would enter into the "warning" threshold and result in throttling the IO writes to 20GB/sec until reaching the 65% threshold.

Table 4-5 65% usable capacity and compression ratios

Usable to Effective Ratio	Usable Space Used (TiB)	Effective Space Used (TiB)
1:1	1089.4	1089.4
1.5:1	1089.4	1634.1
2:1	1089.4	2178.8
2.5:1	1089.4	2722.5

Usable to Effective Ratio	Usable Space Used (TiB)	Effective Space Used (TiB)
3:1	1089.4	3267.6
3.5:1	1089.4	3812.9
4:1	1089.4	4357.6
4.5:1	1089.4	4902.3 (4803)
5:1	1089.4	5447.0 (4803)

If your system passes the capacities listed in the table above, the IO write will be heavily throttled to 2GB/sec until you reach the 70% threshold shown in the following table below. This is the "OOS" threshold and at this point all IO write will be halted.

Table 4-6 70% usable capacity and compression ratios

Usable to Effective Ratio	Usable Space Used (TiB)	Effective Space Used (TiB)
1:1	1173.2	1173.2
1.5:1	1173.2	1759.8
2:1	1173.2	2346.4
2.5:1	1173.2	2931.0
3:1	1173.2	3520.8
3.5:1	1173.2	4105.6
4:1	1173.2	4688.8
4.5:1	1173.2	5272.8 (4803)
5:1	1173.2	5860.0 (4803)

Depending on the configuration of your filesystem, for example if your metadata is stored here, the halting of IO writes will cause your access to the filesystem to fail and require a lengthy recovery process to be completed.

The following tables are provided as a reference for this planning exercise for cross vdiskset size and compression ratios and the thresholds at which throttling occurs:

Set Size	Example Per Drive Logical Space (XL, TB)	DA Physical Space Utilization	Compression N:1	Write Throttle Per BB	DA Space Condition	Generated Event
50%	57.5	55 %	2.7	No Throttle	Information	INFO
50%	57.5	60 %	2.5	20 GB/s	Warning	WARNING
50%	57.5	65 %	2.3	2 GB/s	Critical	WARNING
50%	57.5	70 %	2.1	Write rejected	OOS	ERROR

Set Size	Example Per Drive Logical Space (XL, TB)	DA Physical Space Utilization	Compression N:1	Write Throttle Per BB	DA Space Condition	Generated Event
55%	63.25	55 %	3.0	No Throttle	Information	INFO
55%	63.25	60 %	2.7	20 GB/s	Warning	WARNING
55%	63.25	65 %	2.5	2 GB/s	Critical	WARNING
55%	63.25	70 %	2.4	Write rejected	OOS	ERROR

Set Size	Example Per Drive Logical Space (XL, TB)	DA Physical Space Utilization	Compression N:1	Write Throttle Per BB	DA Space Condition	Generated Event
60%	69	55 %	3.3	No Throttle	Information	INFO
60%	69	60 %	3.0	20 GB/s	Warning	WARNING
60%	69	65 %	2.8	2 GB/s	Critical	WARNING
60%	69	70 %	2.6	Write rejected	OOS	ERROR

Set Size	Example Per Drive Logical Space (XL, TB)	DA Physical Space Utilization	Compression N:1	Write Throttle Per BB	DA Space Condition	Generated Event
70%	80.5	55 %	38	No Throttle	Information	INFO
70%	80.5	60 %	3.5	20 GB/s	Warning	WARNING
70%	80.5	65 %	3.2	2 GB/s	Critical	WARNING
70%	80.5	70 %	3.0	Write rejected	OOS	ERROR

Set Size	Example Per Drive Logical Space (XL, TB)	DA Physical Space Utilization	Compression N:1	Write Throttle Per BB	DA Space Condition	Generated Event
80%	92	55 %	4.4	No Throttle	Information	INFO
80%	92	60 %	4.0	20 GB/s	Warning	WARNING
80%	92	65 %	3.7	2 GB/s	Critical	WARNING
80%	92	70 %	3.4	Write rejected	OOS	ERROR

It is recommended not to reach anything above 60% if performance consistency is the goal and never to reach 70% if data accessibility is required for the environment.

4.4 Capacity Configuration

Depending on the planned use case for the IBM ISS 6000, the system can be configured as either a performance or capacity model. In addition, a third "hybrid model" option is also available, providing a balanced combination of both, performance and capacity. This hybrid model is covered in more detail in section 4.5, “Hybrid Configuration” of this publication.

The following table provides the possible capacity configurations that are available for order from IBM at the time of publishing for this document. The table shows an overview of the following information: number of storage enclosures, number of drives, size of the drives, capacity provided, read/write performance.

Capacity and performance numbers are provided as a guideline with the assumption of 8+2P RAID and 8MB block size is being used and system is configured with non-blocking I/O adapter configuration.

Table 4-7 Capacity model options

Enclosures	Drives	System Capacity (16 / 20 / 22) TB	Performance (100% Read / 100% Write) GB
1	91	982 / 1228 / 1351	10 / 7
2	182	1965 / 2456 / 2702	20 / 14
3	273	2947 / 3684 / 4053	30 / 21
4	364	3930 / 4913 / 5404	40 / 28
5	455	4913 / 6141 / 6755	50 / 35
6	546	5895 / 7369 / 8106	60 / 42
7	637	6878 / 8598 / 9457	70 / 49
8	728	7861 / 9826 / 10808	80 / 56
9	819	8843 / 11054 / 12160	90 / 63

4.5 Hybrid Configuration

The hybrid capacity option is likely the most straightforward to comprehend after reviewing the previous contents of this chapter as it provides a good mix of both read/write performance and storage capacity.

A hybrid configuration, refers to any combination of supported Performance or Flash Cache Module (FCM) and Capacity. Specifically, this configuration entails an ISS 6000 with SSD and 1 to 9 enclosures of NL-SAS.

It is important to note that, at the time of publishing, mixing Performance and FCM within the same ISS 6000 is not a supported option.

It must be understood that while the capacity scales linearly by aggregating the different components of the combined tiers (SSD or FCM and NL-SAS), the performance may not scale linearly. This is contingent upon the usage of the tiers at any given time. For instance, if the SSDs are operating at maximum performance, it may affect the peak performance of the NL-SAS at that moment, and vice versa.

5

Performance

This chapter provides an overview of tuning and configuration considerations for an IBM Storage Scale 6000 deployment. This is an initial starting point and set of best practices for the IBM Storage Scale 6000 storage servers to optimize the performance and stability of the deployment.

This chapter describes the following topics

5.1 Introducing performance storage use cases

Across industries and locations around the world, high-performance storage use cases can appear to vary significantly. However, most of today's storage data and AI applications, have general common requirements for high-performance storage that fit into today's storage data and AI infrastructure. See Figure 5-1.

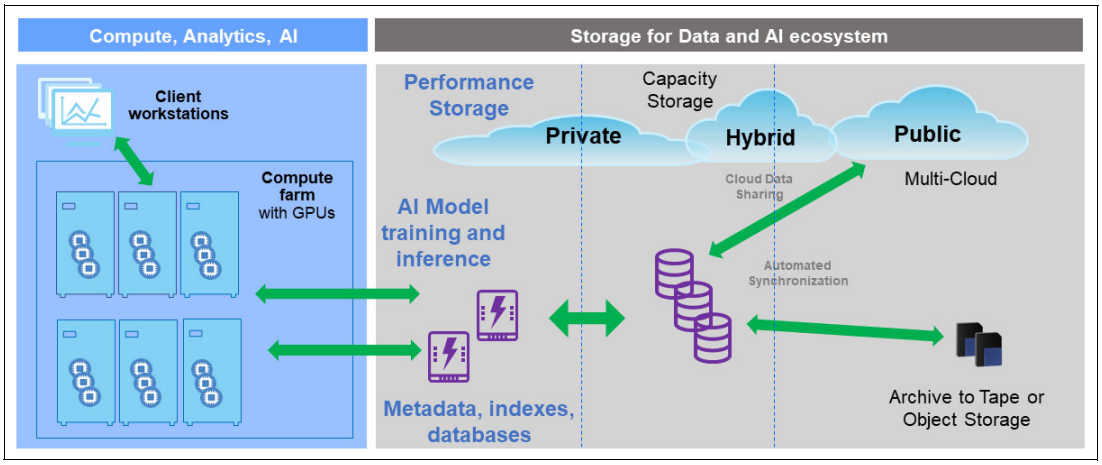


Figure 5-1 Performance storage use case positioning

Storage data and AI use cases generally require these common features:

- An infrastructure of dynamic, scalable, reliable, high-performance storage
- A storage tier that delivers GBps to TBps of throughput to provide storage for graphics processing units (GPUs) and modern computers.
- Performance storage that must also seamlessly integrate with an enterprise data fabric that also has capacity tiers for other uses of the storage such as:
  - Enterprise data repositories
  - Scalable flexible hybrid cloud tiers
  - Cost-effective archive tape and object tiers

The following sections describe how the IBM Storage Scale 6000 (ISS 6000), as performance storage, addresses essential storage data and AI application requirements for AI model training, inference, metadata, indexes, and databases. The ISS 6000 is also described as a seamless, integrated data component, in a larger data and AI infrastructure based on IBM Storage Scale.

5.1.1 IBM ISS 6000 as part of a larger storage for data and AI infrastructure

The IBM ISS 6000 is part of a larger family of IBM Storage solutions that comprehensively covers all aspects of the storage for data and AI configuration, as shown in Figure 5-2.

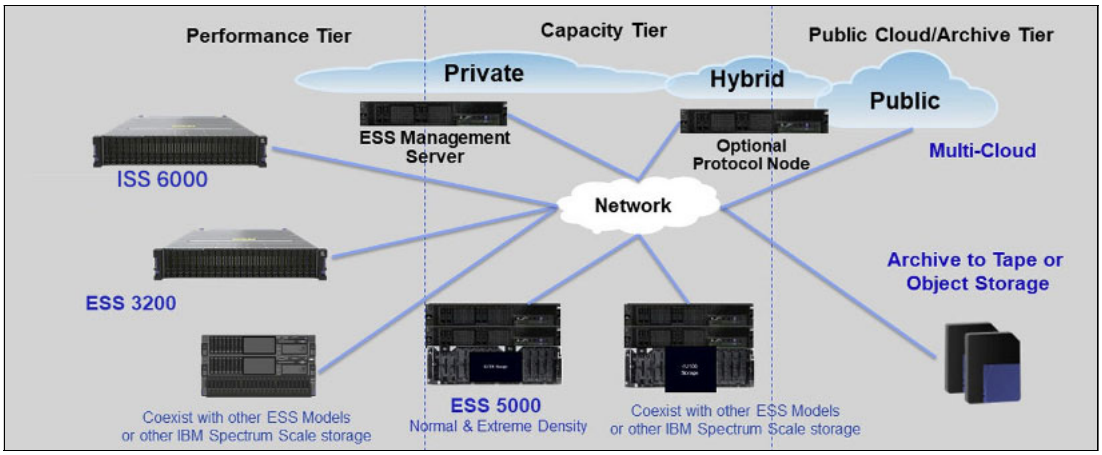


Figure 5-2 ISS positioning within the data and AI storage infrastructure

The ISS 6000 integrates will into large storage infrastructures. Within the larger storage infrastructure, you can use IBM Storage Scale to nondisruptively add, expand, and modify the storage data and AI infrastructure, as needed. The ISS 6000 serves as storage for multiple types of configurations:

- You can install the ISS 6000 as a stand-alone, single, high-performance system.
- You can add additional ISS 6000 building blocks to expand the performance tier, which may also include the user of IBM ESS 3500 building blocks as well.
- You can add one or multiple IBM ESS 5000 enclosures as HDD high-capacity tier.
- Storage Scale environments also have great flexibility in adding other IBM and third-party storage components to the storage infrastructure, including hybrid cloud capacity, and archive capacity to tape or object storage.

The ISS 6000 offers exceptional diversity and flexibility by integrating storage and AI resources under a single global namespace, powered by IBM Storage Scale. This enables a unified data fabric and comprehensive enterprise data management across all storage tiers.

5.1.2 Typical ISS 6000 performance storage use cases

The ISS 6000 is designed to provide scalable, reliable, dense, fast storage for the high performance storage tier, and optionally can provide colder tier HDD storage via expansion drawers. Typical use cases for the ISS 6000 include specific performance tier high-performance computing (HPC), AI, analytics, or other high-performance workloads:

- AI applications that require high-performance data effectively using GPU technology at high resource usage
- Acceleration of scale-out applications with dense NVMe Flash technology
- Information Lifecyle Management and data-tiering management of data in new or existing IBM Storage Scale environments
- Metadata acceleration, indexes, database acceleration
- High-performance storage at the edge

The following sections of this chapter explore some of these use cases.

5.2 Metadata and high-speed data tier

In an IBM Storage Scale cluster, performance of the entire cluster can be accelerated by placing IBM Storage Scale and other metadata on the ISS 6000. Thus, in a high-performance computing environment, a predominant use for the ISS 6000 is to provide the high-performance metadata storage for IBM Storage Scale by using the high throughput of NVMe flash storage.

Metadata generally refers to *data about data*, and in the context of IBM Storage Scale *metadata* refers to various on-disk data structures that are necessary to manage user data. Directory entries and inodes are defined as metadata, but at times the distinction between data and metadata might not be obvious.

For example, examine a configuration with a 4 KB inode. Although the inode might contain user data, the inode is still classified as IBM Storage Scale metadata. The inode is placed in a metadata pool if data and metadata are separated. Another example is the case of directory blocks, which are classified as metadata but also contain user file and directory names.

In many high-performance use cases, performance improvements might be obtained by assigning IBM Storage Scale metadata to a fast tier, which can be accomplished by placing faster ISS 6000 storage in its own storage pool. For details about how to implement this technique, see 3.3.2, “Scenario 1: Using ISS 6000 for metadata network shared disks” on page 75.

This approach to metadata tiering can be adopted when optimizing the performance of metadata operations, such as listing directories and making **`stat()`** calls on files. For more information, see IBM Documentation for IBM Storage Scale on User Storage Pools.

Alternatively, instead of tiering data based on a data or metadata classification, you can use the IBM Storage Scale File Heat function to migrate data between storage pools based on how frequently data is accessed. For more information about this approach, see IBM Documentation for IBM Storage Scale File Heat: Tracking File Access Temperature.

5.3 Data feed to GPUs for massive AI data acceleration

Another predominant use for the ISS 6000 is to provide the high-performance storage throughput that is necessary to feed modern GPU data acceleration systems for real-time AI and machine learning (ML). NVIDIA and IBM created a reference architecture for NVIDIA DGX and ISS 6000 working together on AI and ML workloads. Together, NVIDIA and IBM provide an integrated, individually scalable compute and storage solution with end-to-end parallel throughput from flash to GPU for accelerated DL training, and inference. For more information, see IBM Storage and SDI solutions Reference Architecture and IBM Storage Reference Architecture with NVIDIA DGX A100 Systems.

These reference architectures provide a blueprint for enterprise leaders, solution architects, and others who want to learn how the IBM Storage for AI with NVIDIA DGX systems simplifies and accelerates AI. The scalable infrastructure solution integrates the NVIDIA DGX systems with IBM Storage Scale GPU Direct file storage software, which powers the IBM Storage Scale family of storage systems that includes the ISS 6000.

The reference architecture document describes how the read throughput of the DGX A100 system increases linearly with the addition of additional IBM Storage Scale Systems. For example, in the reference document, an older single IBM ESS system is able to provide a read throughput of 48 GBps, while adding a second older IBM ESS system increases the read throughput to 94 GBps, which is near linear scaling.

AI and ML workloads can benefit from the outstanding performance capabilities of the ISS 6000 system. For more information about using the ISS 6000 with high-performance GPU, see IBM, NVIDIA Team on Supercomputing Scalability for AI.

For more information about the IBM Storage Scale GPUDirect Storage (GDS), see GPUDirect Storage support for IBM Storage Scale.

5.4 Other use cases

The ISS 6000 runs IBM Storage Scale as its file system, so some use cases and planning that apply to other members of the IBM Storage Scale family also apply to the ISS 6000.

5.4.1 Genomics medicine workloads in IBM Storage Scale

IT administrators, physicians, data scientists, researchers, bioinformaticians, and other professionals who are involved in the genomics workflow need the right foundation to achieve their research objectives efficiently. At the same time, they want to improve patient care and outcomes. Thus, it is important to understand the different stages of the genomics workload and the key characteristics of it.

Advanced genomics medicine customers are outgrowing network-attached storage (NAS). The move from a traditional NAS system or a modern scale-out NAS system to a parallel file system like IBM Storage Scale requires a new set of skills. For basic background information and for information about optional professional services, see *IBM Storage Scale Best Practices for Genomics Medicine Workloads*, REDP-5479.

You can combine the ISS 6000 in an IBM Storage Scale cluster, with Cloud Object Storage through the IBM Storage Scale Active File Management (AFM)-to-Cloud Object Storage feature. This function enables copies of files or objects in an ISS 6000 or IBM Storage Scale cluster to be written to, or retrieved from, an external Cloud Object Storage. The same functions can also be used to read or write data to or from other external NFS data sources.

This integration provides the ISS 6000 with the ability to seamlessly integrate and accelerate IBM Storage Scale data access to and from external NFS storage and to object storage such as Amazon S3 and IBM Cloud® Object Storage.

The ISS 6000 integrates external NFS storage and object storage into a common data repository with enterprise-high scalability, data availability, security, and performance. The AFM-to-cloud object storage associates an IBM Storage Scale file set with a Cloud Object Storage bucket. Figure 5-3 shows an example of this configuration.

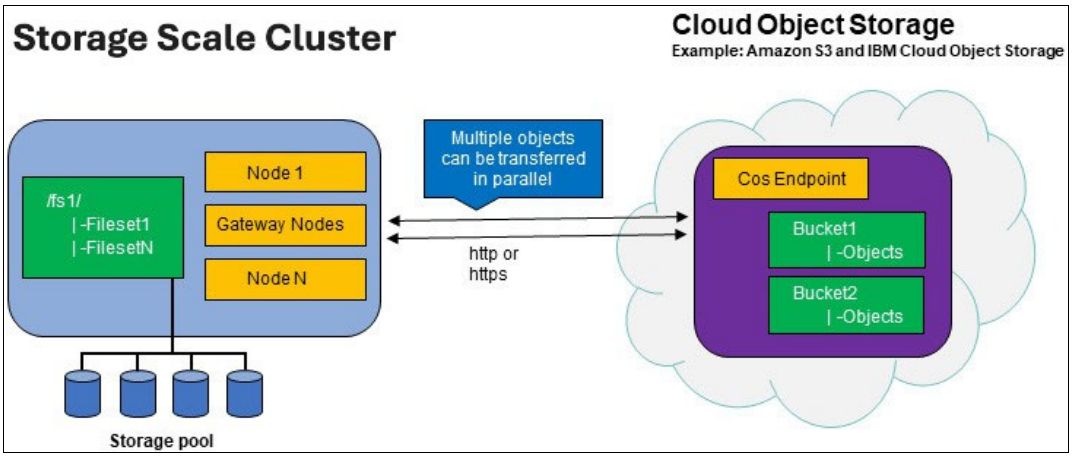


Figure 5-3 IBM Storage Scale Active File Management connected to Cloud Object Storage

Using this function, IBM Storage Scale file sets and Cloud Object Storage buckets become extensions of each other. Files and objects that are required for applications such as AI and big data analytics can be shared, downloaded, worked upon, and uploaded between the ISS 6000, IBM Storage Scale, and the Cloud Object Storage. These use cases are shown in Figure 5-4.

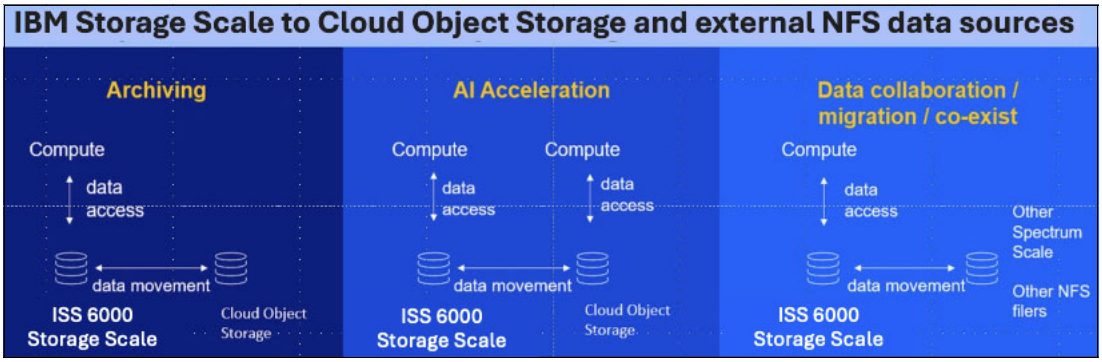


Figure 5-4 IBM Storage Scale to Cloud Object Storage and external NFS data sources

The following use cases are a subset of possible use cases:

- Archiving data to and from external object storage data sources
- Optimizing data movement and data access, speeding time to data value
- Connecting and migrating data to consolidate, or co-exist with, external NFS NAS data sources

The workloads and workflows that might benefit from these use cases include (but are not limited to) mobile applications, backup and restore, enterprise applications, big data analytics, and file servers.

The AFM-to-Cloud Object Storage feature also allows data center administrators to free ISS 6000 and IBM Storage Scale storage capacity by policy management. Data is moved to lower-cost or off-premise cloud storage, which reduces capital and operational expenditures. The data movement can be done automatically through the AFM-based cache eviction feature or through policy. The data movement can be used to automate and optimize data placement between ISS 6000 and other storage within the IBM Storage Scale storage infrastructure.

5.5 Server Tuning

This chapter provides an overview of tuning and configuration considerations for an ISS 6000 deployment. An initial starting point set of best practices for ISS 6000 storage servers is provided to optimize the performance and stability of the deployment.

5.5.1 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, at the time of writing, that supported Storage Scale code levels are being run. For more information about supported code levels, see IBM Storage Scale Frequently Asked Questions and Answers.

5.5.2 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 deployment, client nodes that use the file system are called NSD clients, while IBM Storage Scale 6000 servers are called NSD servers. For more information about NSD clients and servers, “IBM Storage Scale cluster configuration”.

5.5.3 Starting point for tuning mmchconfig parameters

This section outlines the initial IBM Storage Scale tuning recommendations for client nodes in an ISS 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 recommendation that apply to the earlier 5.1.X code, which is supported at the time of writing. Some tuning defaults have been changed in later code levels (e.g., in the 5.2.2 code) but it’s OK to set the recommendations below, even in cases in which the default has already been changed to match the recommended value.

Example 5-1 shows the starting point for the system settings.

Example 5-1 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 5-2.

Example 5-2 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 5-3.

Example 5-3 The verbsRdmaCm option

```
verbsRdmaCm=enable
```

To enable 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. Additionally, proper setup of the RDMA Connection Manager (RDMA CM) is typically required to establish and manage connections in RoCE environments. 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 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 5.5.5, “Details about best practices for the mmchconfig tuning options” on page 99.

5.5.4 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 5.7, “Storage network configuration and validation” on page 104). 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 5-4.

Example 5-4 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, you can typically omit the `verbsRdmaCm=enable` option because Infiniband systems rarely enable the connection manager, while it's almost always enabled for Ethernet (RoCE) deployments.

5.5.5 Details about best practices for the mmchconfig tuning options

This section provides more information about best practices for the `mmchconfig` tuning options for ISS 6000 deployments. For more information, see the mmchconfig command.

Here are details on 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, turning this setting to no 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 MB/s instead of MiB/s to calculate the correct value for the `maxMBpS` setting, 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 ISS 6000 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 mounting a remote GPFS file system.
- `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 an ISS 6000 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 as described in 1.2, “The basic structure of IBM Storage Scale” on page 14).
- `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. 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.

Note: The new default setting is not enabled automatically until 5.2.3-1. For all Scale systems 5.2.0 to 5.2.3 manually update the `numaMemoryInterleave` setting to YES.

- `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 an ISS 6000 deployment), 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 IBM Storage Scale always uses an amount of real memory equal to the `pagepool` size. When configuring `pagepool`, consider both the IBM Storage Scale 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 if sufficient memory is available, set `pagepool` no lower than 4 GiB on 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 ISS 6000 deployments, it's suggested to set this tuning option to the value of the smallest block size, out of all file systems being used. For file systems with a block size equal to or larger than the specified value, a more gradual prefetching mode will be enabled, improving prefetch 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 ISS 6000 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 ISS 6000 deployment, see 5.7, “Storage network configuration and validation” on page 104.

  The definition of `verbsPorts` consists of a series of elements, as shown in Example 5-5 on page 102.

Example 5-5 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 typically 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 such Ethernet networks, set `verbsRdmaCm` to `enable` along with `verbsRdma` and define a valid `verbsPorts` string. As noted below, 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. It is best practice to enable `verbsRdmaSend` to receive optimal RDMA performance.

- `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 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.

5.6 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 5.5.5, “Details about best practices for the mmchconfig tuning options” on page 99, use the configuration options in Example 5-6 to enable RDMA.

Example 5-6 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 5-7.

Example 5-7 Enabling verbsRdmaCm for RoCE

```
verbsRdmaCm=enable
```

When you define the list of `verbsPorts` in a ISS 6000 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, see 5.7, “Storage network configuration and validation” on page 104.

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 5.5.2, “Starting point for IBM Storage Scale tuning best practice for clients” on page 97. 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 enable `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 5-8.

Example 5-8 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.

5.6.1 IBM Storage Scale 6000 file system block size considerations

As a best practice, start with an 8 MB block size, but you may also consider using a 16 MB block size with an ISS 6000 deployment. 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.

5.7 Storage network configuration and validation

When deploying an IBM Storage Scale 6000 storage, not all network interfaces on the clients are used for communication with the IBM Storage Scale server nodes. 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.

It is considered best practice to configure and use RDMA on the IBM Storage Scale clients and servers. This section describes how to pick the interfaces that are used for the `mmchconfig verbsPorts` option, and validating the network by using `nsdperf`.

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.

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

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 5-9 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 5-9 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)
```

In Example 5-9, there are two physical networks (each adapter alternates networks) used by 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
```

It 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, we 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 5.5.5, “Details about best practices for the mmchconfig tuning options” on page 99, the verbsPorts definition has the format that is shown in Example 5-10.

Example 5-10 Format of a verbsPorts definition

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

To create a verbsPorts string with the format shown in Example 5-10, enter the appropriate hca_ids and port numbers, and assign a virtual network ID to both of the physical networks. To create the virtual network IDs, define the first switch as fabric_number=0, and the second switch as fabric_number=1, as shown in Example 5-11.

Example 5-11 Defining the verbsPorts (including switch fabrics) for the ISS 6000 servers

```
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>
```

We’ll use the same string definitions when running nsdperf to validate network performance (for the ‘-r’ option passed to nsdperf) as well as when defining the verbsPorts configuration option for the IBM Storage Scale 6000 servers.

On the client nodes, repeat the same steps, but include only the hca_ids used for storage network communication. Exclude interfaces dedicated to other functions, such as MPI-based compute communication. Generally the selected hca_ids for Storage Scale communication on the clients are connected to the same physical network(s) as the IBM Storage Scale 6000 servers.

For this example, we assume that the clients have two adapters that are connected to two separate physical storage networks:

```
mlx5_0 port 1 connected to the first fabric
mlx5_1 port 1 connected to the second fabric
```

We used mlx5_0 and mlx5_1 as the clients’ storage fabric connections for the example, but the mapping of adapters will vary for different systems – e.g. the first eight adapters might be used for compute workloads and the first adapter used by Storage Scale might be mlx5_8.

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 5-12.

Example 5-12 Defining the verbsPorts definition for the client nodes

```
mmchconfig verbsPorts="mlx5_0/1/0 mlx5_1/1/1"
```

We used mlx5_0 and mlx5_1 as the clients’ storage fabric connections for the example, but the mapping of adapters will vary for different systems – e.g. the first eight adapters might be used for compute workloads and the first adapter used by Storage Scale might be mlx5_8.

5.7.2 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. For the case of a single physical network, 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 a given storage fabric.

Another option, instead of using bonding/teaming, 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.

5.7.3 Network verification with nsdperf

An important part of installing a new ISS 6000 deployment is to 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, and then the `nsdperf` tool should be run to validate 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 to be used by 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 is intended to work using 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, at the time of writing 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 both RDMA and TCP/IP only configurations.

The following steps show an example of running `nsdperf` with RDMA by using the `hca_ids` and mappings that are described in 5.7.1, “Selecting the network interface list for RDMA communication for the IBM Storage Scale 6000 server node network” on page 105. Although it is a best practice to enable RDMA for IBM Storage Scale communication, in addition to the TCP/IP communication that is always required, we also provide instructions about how to modify the RDMA measurement example to instead test only TCP/IP communication without RDMA.

When testing a single physical network using RDMA, or a non-RDMA configuration that uses bonding, teaming, or routing protocols to aggregate multiple TCP/IP interfaces, 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 5-13.

Example 5-13 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 5-14, and save it in a file that is named `server.list`.

Example 5-14 Server list (one server per line)

```
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 5-15.

Example 5-15 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 5-16 to build `nsdperf` for only TCP/IP measurements.

Example 5-16 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 5-17.

Example 5-17 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 when running `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 5-18. 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 is to avoid killing the wrong process).

Example 5-18 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 5-19.

Example 5-19 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 5-9 on page 105. 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 5-20.

Example 5-20 Starting the ISS 6000 nsdperf server processes with RDMA on port 6665 (first physical network)

```
mmdsh -N server.list “numactl -i all /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 to ndsperf. Use the command that is shown in Example 5-21 instead of the one that is shown in Example 5-20.

Example 5-21 Starting the ISS 6000 nsdperf server processes without RDMA

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

   When you test a single physical network configuration, or any TCP/IP network in which all the interfaces used by storage are aggregated to a single interface (e.g. via routing protocols) then you may use a single instance of nsdperf to test network performance. If you are testing such a 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 5-22.

Example 5-22 Starting the ISS 6000 nsdperf server processes on port 6666 (second physical server)

```
mmdsh -N server.list “numactl -i all /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 client nodes.

   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 5-9 on page 105. The clients connect to the first storage fabric by using port 1 on hca_id mlx5_0, so start the nsdperf server processes on port 6665 by running the command in Example 5-23.

Example 5-23 Starting the client nsdperf server processes on port 6665 (first physical network)

```
mmdsh -N client.list “numactl -i all /usr/lpp/mmfs/samples/net/nsdperf -p 6665 -s
-w 192 -r mlx5_0/1/0" > /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 5-24 instead of the one that is shown in Example 5-23 on page 110.

Example 5-24 Starting the nsdperf server processes on port 6665 without option -r

```
mmdsh -N client.list "numactl -i all /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 you test a single physical network configuration, or any TCP/IP network in which all the interfaces used by storage are aggregated to a single interface (e.g. via routing protocols) then you may use a single instance of nsdperf to test network performance. If you are testing such a 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_1`, so start the `nsdperf` server processes on port 6666, as shown in Example 5-25.

Example 5-25 Starting the nsdperf server processes without RDMA

```
mmdsh -N client.list "numactl -i all /usr/lpp/mmfs/samples/net/nsdperf -p 6666 -s
-w 192 -r mlx5_1/1/1" > /tmp/nsdperf.client.log2 2>&1 &
```

Check that every node has a `nsdperf` server process running by running the command in Example 5-26.

Example 5-26 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
```

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 5-27 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 should match the IP interface name(s) that you are trying to test.

   If you're defining a single interface for storage scale traffic (e.g. using bonding or routing protocols), the client and server names should be the interface that aggregate the physical links used for storage network.

   When using an MROT (Multi Rail Over TCP) configuration in IBM Storage Scale to leverage multiple IP interfaces—rather than relying on bonding, teaming, or routing protocols for traffic aggregation—you can validate performance by running multiple instances of `nsdperf`, each on a different TCP/IP port. For details, 5.7.4, "Validating networks without RDMA enabled" on page 113.

Example 5-27 Sample nsdperf.cmd.file file

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

```
client c91f93h2O0-2
client c91f93h2O0-3
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 120
buffsize 16777216
threads 32
parallel 2
usecm on # this is only needed when RDMA connection manager is enabled (typically
         # Ethernet RoCE runs)
rdma on
test nwrite
test read
quit
```

When testing multiple physical networks with RDMA, you should 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.

For example, run the `nsdperf.cmd.file` file on both fabrics concurrently by running the commands that are shown in Example 5-28.

Example 5-28 Running the nsdperf.cmd.file file on both fabrics concurrently

```
/usr/lpp/mmfs/samples/net/nsdperf -i nsdperf.cmd.file -p 6665 >
nsdperf.fabric0.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 c91f93h2O0-2: 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 5-29 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 5-29 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 5-30 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 5-30 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 5.7.5, “Diagnosing network performance issues” on page 114.

5.7.4 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 5-31 on page 114 and Example 5-32 on page 114, all the -eth0 and -eth1 interfaces on the clients and servers have connectivity to each other.

Example 5-31 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 when the RDMA connection manager is
                  # enabled (typically Ethernet RoCE runs)
rdma on
test nwrite
test read
quit
```

Example 5-32 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 when the RDMA connection manager is
                  # enabled (typically 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 5-33.

Example 5-33 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
```

5.7.5 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.

5.8 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 in a storage cluster handle token management, which is a performance-sensitive operation that coordinates access to storage resources. Note that token handling can be memory-intensive depending on how the system is tuned, and the token handling function applies only to manager nodes in clusters that provide storage. Manager nodes in clusters composed solely of client nodes do not act as token servers when those clients join remote clusters to access their file systems.

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.

6

Use cases

This chapter provides a high level overview of use cases where the IBM Storage Scale System 6000 is able to excel and provide value beyond the previous ISS offerings. It includes the following topics:

6.1 Use Case overview

As a high-performance, scalable, and software-defined storage solution IBM Storage Scale (formerly known as IBM Spectrum Scale) is designed for managing large amounts of data across various environments. Able to provide extremely high speed data access, or high speed access to large amounts of storage depending on the configurations, here are some common use cases suitable for the ISS 6000:

High-Performance Computing (HPC)

Use Case: Scientific simulations, weather modeling, genomics, and engineering workloads.

Why: Delivers low-latency, high-throughput access to massive datasets across thousands of nodes.

AI and Machine Learning Workloads

Use Case: Training and inference pipelines for AI/ML models.

Why: Provides fast, parallel access to training data and supports GPU-accelerated environments.

Hybrid Cloud and Multi-Cloud Storage

Use Case: Seamless data movement between on-premises and cloud environments.

Why: Supports cloud bursting, backup, and disaster recovery with integration into IBM Cloud, AWS, Azure, etc.

Enterprise File Services

Use Case: Centralized file storage for enterprise applications and users.

Why: Offers POSIX-compliant file access, multi-protocol support (NFS, SMB), and high availability.

Big Data and Analytics

Use Case: Data lakes, Apache Spark, Hadoop, and other analytics platforms.

Why: Enables concurrent access to large datasets with high throughput and scalability.

Media and Entertainment

Use Case: Video editing, rendering, and broadcasting.

Why: Handles large media files efficiently with fast read/write speeds and parallel access.

Backup and Archiving

Use Case: Long-term data retention and compliance.

Why: Integrates with IBM Spectrum Protect and supports tiered storage to reduce costs.

Containerized Workloads

Use Case: Kubernetes and OpenShift environments.

Why: Offers CSI drivers and native integration for persistent storage in containerized apps.

A

Configuring the 48 ports top of the rack management network switch

This appendix also provides you with the information to consider when not using the switches that IBM provides and supports for the management network. It describes how to configure the management network switch to achieve the same functionality with your own network devices.

Configuring the switches

As shown in 3.1.2, “Hardware planning” on page 64, the management switch includes ports 1 - 12 as “ISS 6000" ports. Those ports are different from Version 1 because both management flexible service processor (FSP) networks are configured in the same port.

The process to platform the switch is not changed from Version 1. The configuration-content of the file is used to platform the switch. The same two VLANs that were used on Version 1 are used in Version 2. New VLANs are not added from Version 1.

Follow these steps to configure a Version 2 switch:

1. Connect through *serial*. You lose access to the switch if you apply it through IP access because the default configuration does not have any IP address configured.

   You must have the cumulus user ID and password. It is likely the user ID of *cumulus* and a password of *CumulusLinux!*, but it might be the serial number. The serial configuration is:

   – Baud rate: 115200
   – Parity: None
   – Stop bits: 1
   – Data bits: 8
   – Flow control: None

You need the configuration file that is detailed at the end of this appendix in Example A-1 on page 121.

2. Log in to the switch by using the *cumulus* user ID.
3. Enter the command `sudo su -`.
4. As `root`, put the contents of the configuration file into `/etc/network/interfaces`. (You can copy and paste.) Previous contents of the configuration file must be discarded.

Important: Connect through serial, or you lose access in step 5.

5. Apply the configuration with the command **`ifreload -a`**.

   At this point, the new configuration is applied.

6. Confirm that the configuration change was applied by using **`ifquery -a.`**
7. A best practice is to set a static IP for remote log in on the switch, for example, given these parameters: network is 192.168.44.0/24; IP switch 192.168.44.20; gateway 192.168.44.1:

   – **`net add interface eth0 IP address 192.168.44.20/24`**
   – **`net add interface eth0 IP gateway 192.168.44.1`**
   – **`net pending`**
   – **`net commit`**

Note: When you convert a switch that was not previously configured for an ISS 6000, if ports 1 - 12 are not already being used, then go directly to the steps shown in the following example. If any port from 1 - 12 is being used on the switch, then these ports must be reconfigured.

Move the cables on the upper ports 1 -12, to any free upper port that is outside the 1-12 port range. Any lower cable plugged to a port in the range 1-12 also needs to be moved to any lower port not in the 1-12 port range.

Move one cable at a time and wait until the link LED on the destination port becomes lit. After all the ports in the range 1-12 are no longer cabled, continue and apply the steps that are listed in Example A-1.

The file with the configuration must contain the data that is shown in Example A-1.

Example A-1 Data that is required for the configuration file

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).
source /etc/network/interfaces.d/*.intf
# The loopback network interface auto
lo
iface lo inet loopback
# The primary network interface auto
eth0
iface eth0 inet dhcp
# EVEN Ports/Lower ports PVID 101 for FSP network auto
swp14
iface swp14
bridge-access 101
auto swp16
iface swp16
bridge-access 101
auto swp18
iface swp18
bridge-access 101
auto swp20
iface swp20
bridge-access 101
auto swp22
iface swp22
bridge-access 101
auto swp24
iface swp24
bridge-access 101
auto swp26
iface swp26
bridge-access 101
auto swp28
iface swp28
bridge-access 101
auto swp30
iface swp30
bridge-access 101
auto swp32
```

```
iface swp32
bridge-access 101
auto swp34
iface swp34
bridge-access 101
auto swp36
iface swp36
bridge-access 101
auto swp38
iface swp38
bridge-access 101
auto swp40
iface swp40
bridge-access 101
auto swp42
iface swp42
bridge-access 101
auto swp44
iface swp44
bridge-access 101
auto swp46
iface swp46
bridge-access 101
auto swp48
iface swp48
bridge-access 101


# ODD Ports/Upper ports PVID 102 for xCAT network auto
swp13
iface swp13
bridge-access 102
auto swp15
iface swp15
bridge-access 102
auto swp17
iface swp17
bridge-access 102
auto swp19
iface swp19
bridge-access 102
auto swp21
iface swp21
bridge-access 102
auto swp23
iface swp23
bridge-access 102
auto swp25
iface swp25
bridge-access 102
auto swp27
iface swp27
bridge-access 102
auto swp29
iface swp29
```

```
bridge-access 102
auto swp31
iface swp31
bridge-access 102
auto swp33
iface swp33
bridge-access 102
auto swp35
iface swp35
bridge-access 102
auto swp37
iface swp37
bridge-access 102
auto swp39
iface swp39
bridge-access 102
auto swp41
iface swp41
bridge-access 102
auto swp43
iface swp43
bridge-access 102
auto swp45
iface swp45
bridge-access 102
auto swp47
iface swp47
bridge-access 102


# ISS 6000 ports (1 to 12) FSP + OS on single physical port
auto swp1
iface swp1
bridge-pvid 102
bridge-vids 101
auto swp2
iface swp2
bridge-pvid 102
bridge-vids 101
auto swp3
iface swp3
bridge-pvid 102
bridge-vids 101
auto swp4
iface swp4
bridge-pvid 102
bridge-vids 101
auto swp5
iface swp5
bridge-pvid 102
bridge-vids 101
auto swp6
iface swp6
bridge-pvid 102
bridge-vids 101
```

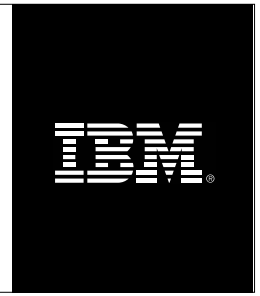
```
auto swp7
iface swp7
bridge-pvid 102
bridge-vids 101
auto swp8
iface swp8
bridge-pvid 102
bridge-vids 101
auto swp9
iface swp9
bridge-pvid 102
bridge-vids 101
auto swp10
iface swp10
bridge-pvid 102
bridge-vids 101
auto swp11
iface swp11
bridge-pvid 102
bridge-vids 101
auto swp12
iface swp12
bridge-pvid 102
bridge-vids 101


# Bridge setup
auto bridge iface
bridge
bridge-vlan-aware yes
bridge-ports glob swp1-48
bridge-pvid 101
bridge-pvid 102
bridge-stp
off
```

Implementation Guide for IBM Storage Scale System 6000

SG24-8566-00

ISBN

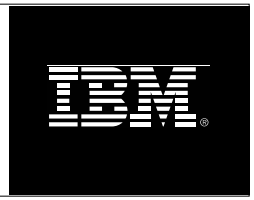


(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages

Implementation Guide for IBM Storage Scale System 6000

SG24-8566-00

ISBN



(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Storage Scale 6000 Implementation guide

SG24-8566-00

ISBN

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Storage Scale 6000 Implementation guide

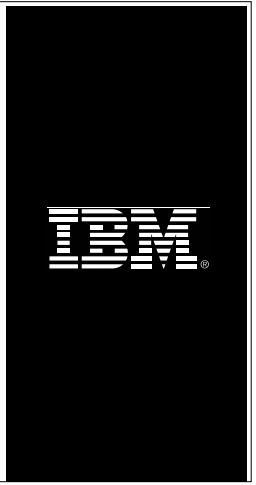
(0.2”spine)
0.17”<->0.473”
90<->249 pages

(0.1”spine)
0.1”<->0.169”
53<->89 pages

Implementation Guide for IBM Storage Scale System

SG24-8566-00

ISBN

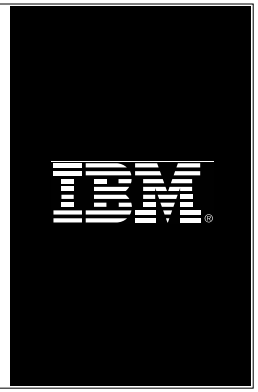


(2.5” spine)
2.5”<->nnn.n”
1315<-> nnnn pages

Implementation Guide for IBM Storage Scale System 6000

SG24-8566-00

ISBN



(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

SG24-8566-00

ISBN

Printed in U.S.A.