IBM Storage Scale:
Konzepte

Lars Lauber
Storage Technical Specialist
lauberla@de.ibm.com

Disclaimer

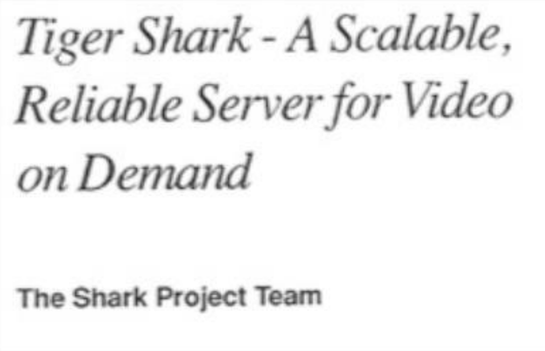
Storage Scale

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

IBM reserves the right to change product specifications and offerings at any time without notice. This publication could include technical inaccuracies or typographical errors. References herein to IBM products and services do not imply that IBM intends to make them available in all countries.

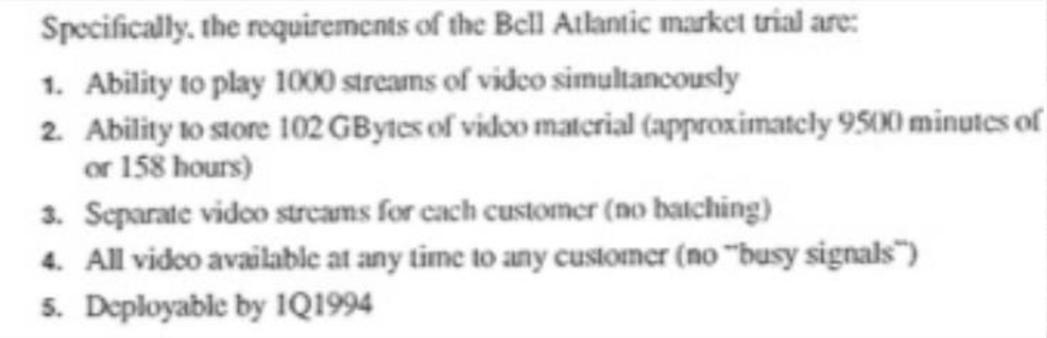
Historie

The IBM Shark Project Team, 1994



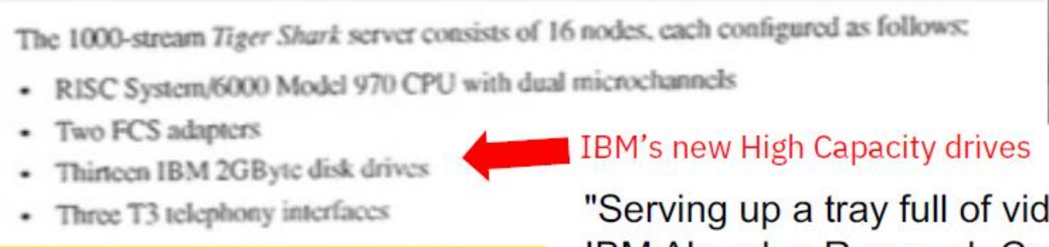
Tiger Shark - A Scalable, Reliable Server for Video on Demand

The Shark Project Team



Specifically, the requirements of the Bell Atlantic market trial are:

1. Ability to play 1000 streams of video simultaneously
2. Ability to store 102 GBytes of video material (approximately 9500 minutes of or 158 hours)
3. Separate video streams for each customer (no batching)
4. All video available at any time to any customer (no "busy signals")
5. Deployable by 1Q1994



The 1000-stream *Tiger Shark* server consists of 16 nodes, each configured as follows:

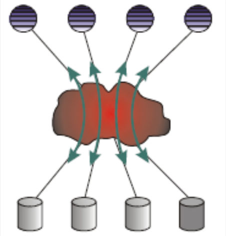
- RISC System/6000 Model 970 CPU with dual microchannels
- Two FCS adapters
- Thirteen IBM 2GByte disk drives
- Three T3 telephony interfaces

IBM's new High Capacity drives

13 years before Netflix offered video download

"Serving up a tray full of videotapes is the Almaden video-on-demand team."
IBM Almaden Research Center, Almaden Views, Sept/Oct, 1994.
Front, left to right: Roger Haskin, Jim Wyllie, Frank Schmuck, Robin Williams, and Carol Hartman.
Back: Mike Roberts, Dan McNabb, and Sam Drake.

GPFS (Spectrum Scale) started in 1998

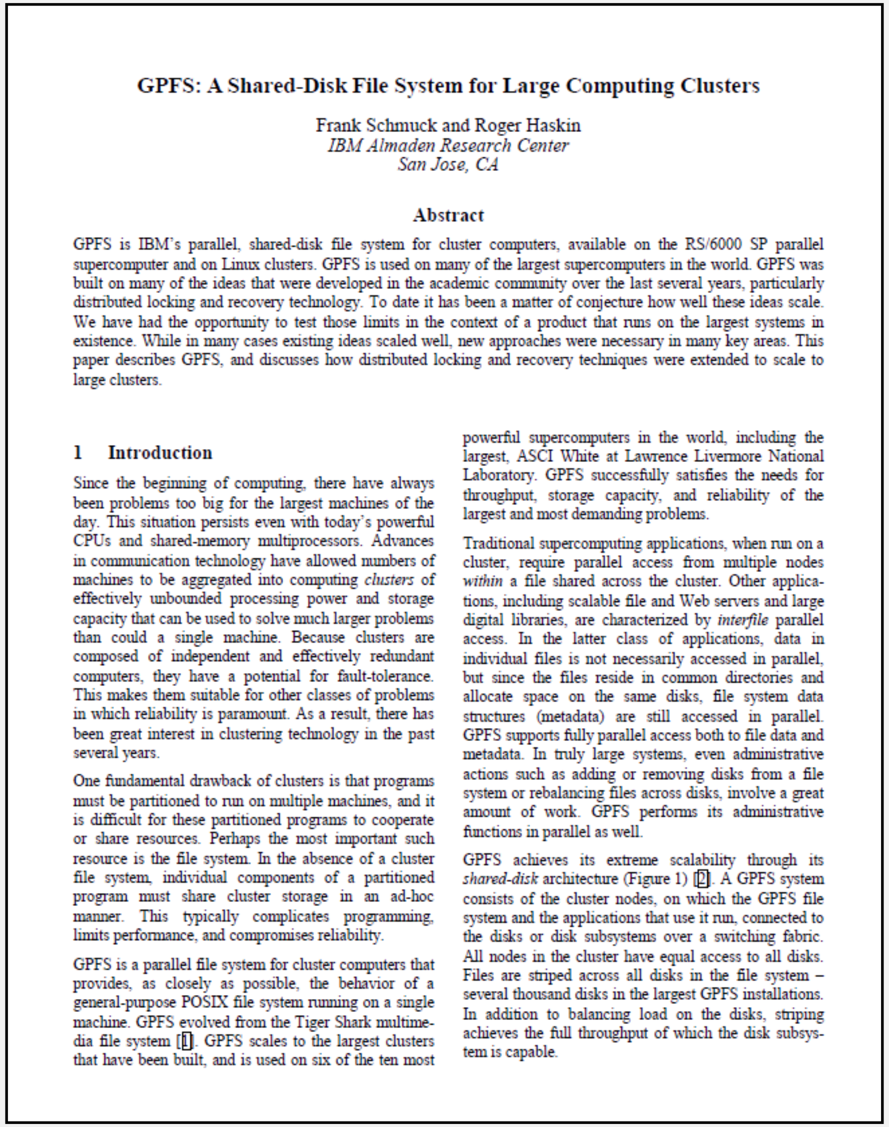


GPFS: A Shared-Disk File System for Large Computing Clusters

Frank Schmuck and **Roger Haskin**
IBM Almaden Research Center
San Jose, CA

Frank

Roger



GPFS: A Shared-Disk File System for Large Computing Clusters

Frank Schmuck and Roger Haskin
IBM Almaden Research Center
San Jose, CA

Abstract

GPFS is IBM's parallel, shared-disk file system for cluster computers, available on the RS/6000 SP parallel supercomputer and on Linux clusters. GPFS is used on many of the largest supercomputers in the world. GPFS was built on many of the ideas that were developed in the academic community over the last several years, particularly distributed locking and recovery technology. To date it has been a matter of conjecture how well these ideas scale. We have had the opportunity to test those limits in the context of a product that runs on the largest systems in existence. While in many cases existing ideas scaled well, new approaches were necessary in many key areas. This paper describes GPFS, and discusses how distributed locking and recovery techniques were extended to scale to large clusters.

1 Introduction

Since the beginning of computing, there have always been problems too big for the largest machines of the day. This situation persists even with today's powerful CPUs and shared-memory multiprocessors. Advances in communication technology have allowed numbers of machines to be aggregated into computing *clusters* of effectively unbounded processing power and storage capacity that can be used to solve much larger problems than could a single machine. Because clusters are composed of independent and effectively redundant computers, they have a potential for fault-tolerance. This makes them suitable for other classes of problems in which reliability is paramount. As a result, there has been great interest in clustering technology in the past several years.

One fundamental drawback of clusters is that programs must be partitioned to run on multiple machines, and it is difficult for these partitioned programs to cooperate or share resources. Perhaps the most important such resource is the file system. In the absence of a cluster file system, individual components of a partitioned program must share cluster storage in an ad-hoc manner. This typically complicates programming, limits performance, and compromises reliability.

GPFS is a parallel file system for cluster computers that provides, as closely as possible, the behavior of a general-purpose POSIX file system running on a single machine. GPFS evolved from the Tiger Shark multimedia file system [1]. GPFS scales to the largest clusters that have been built, and is used on six of the ten most powerful supercomputers in the world, including the largest, ASCI White at Lawrence Livermore National Laboratory. GPFS successfully satisfies the needs for throughput, storage capacity, and reliability of the largest and most demanding problems.

Traditional supercomputing applications, when run on a cluster, require parallel access from multiple nodes *within* a file shared across the cluster. Other applications, including scalable file and Web servers and large digital libraries, are characterized by *interfile* parallel access. In the latter class of applications, data in individual files is not necessarily accessed in parallel, but since the files reside in common directories and allocate space on the same disks, file system data structures (metadata) are still accessed in parallel. GPFS supports fully parallel access both to file data and metadata. In truly large systems, even administrative actions such as adding or removing disks from a file system or rebalancing files across disks, involve a great amount of work. GPFS performs its administrative functions in parallel as well.

GPFS achieves its extreme scalability through its *shared-disk* architecture (Figure 1) [2]. A GPFS system consists of the cluster nodes, on which the GPFS file system and the applications that use it run, connected to the disks or disk subsystems over a switching fabric. All nodes in the cluster have equal access to all disks. Files are striped across all disks in the file system – several thousand disks in the largest GPFS installations. In addition to balancing load on the disks, striping achieves the full throughput of which the disk subsystem is capable.

What is Storage Scale (aka GPFS, aka Spectrum Scale)?

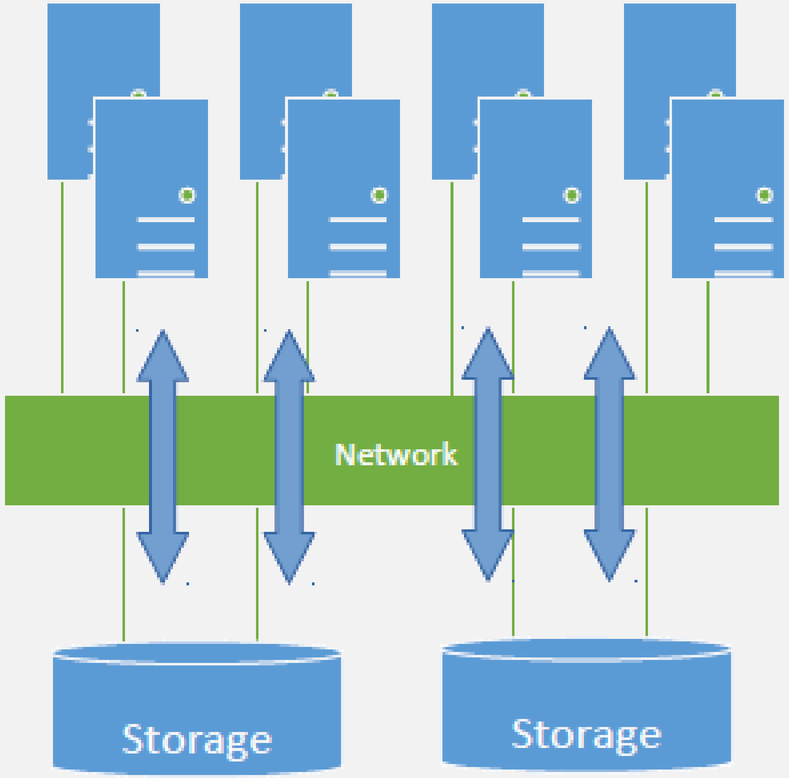
"General Parallel File System": IBM's shared disk, parallel cluster file system. Runs under AIX, Linux and Windows OS on IBM Power and Intel/AMD x86 architecture. Designed for high performance commercial and scientific applications. Used on many of the largest supercomputers in the world.

Cluster: 2-10,000 nodes, fast reliable communication, common admin domain.

Shared disk: all data and metadata on storage devices accessible from any node through block I/O interface

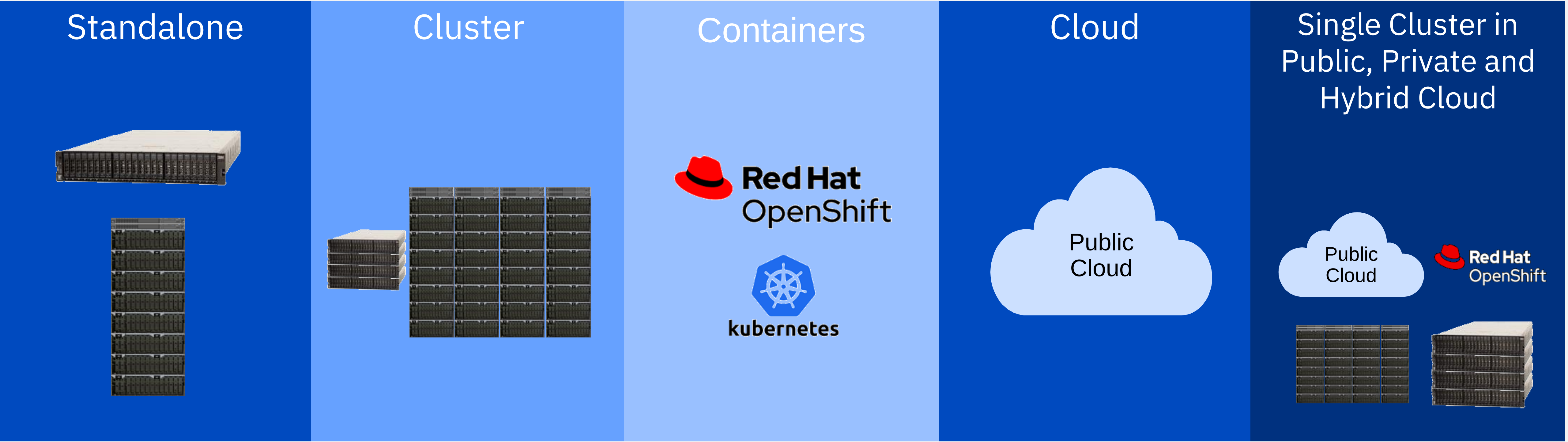
("disk": any kind of block storage device)

– *Parallel:* data and metadata flow from all of the nodes to all of the disks in parallel.



One YB scalable namespace. Anywhere.
Easier access to data. Everywhere
With one view and management of data.

Data Catalog and Policy Engine



Standalone

Cluster

Containers

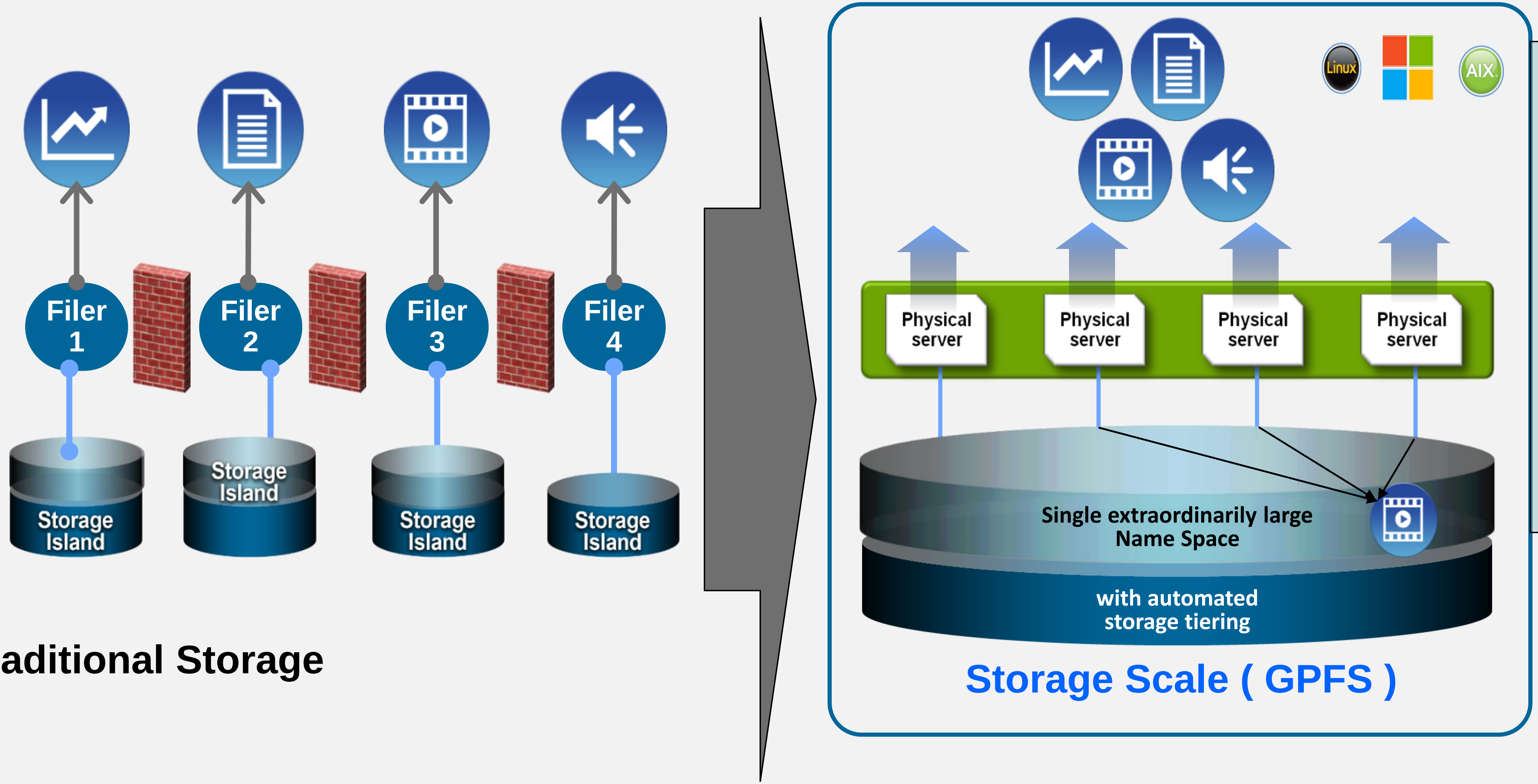
Cloud

Single Cluster in Public, Private and Hybrid Cloud

File and Object Access

Simplicity with a single namespace and cluster to manage

one big platform



traditional Storage

Storage Scale (GPFS)

General

Parallel

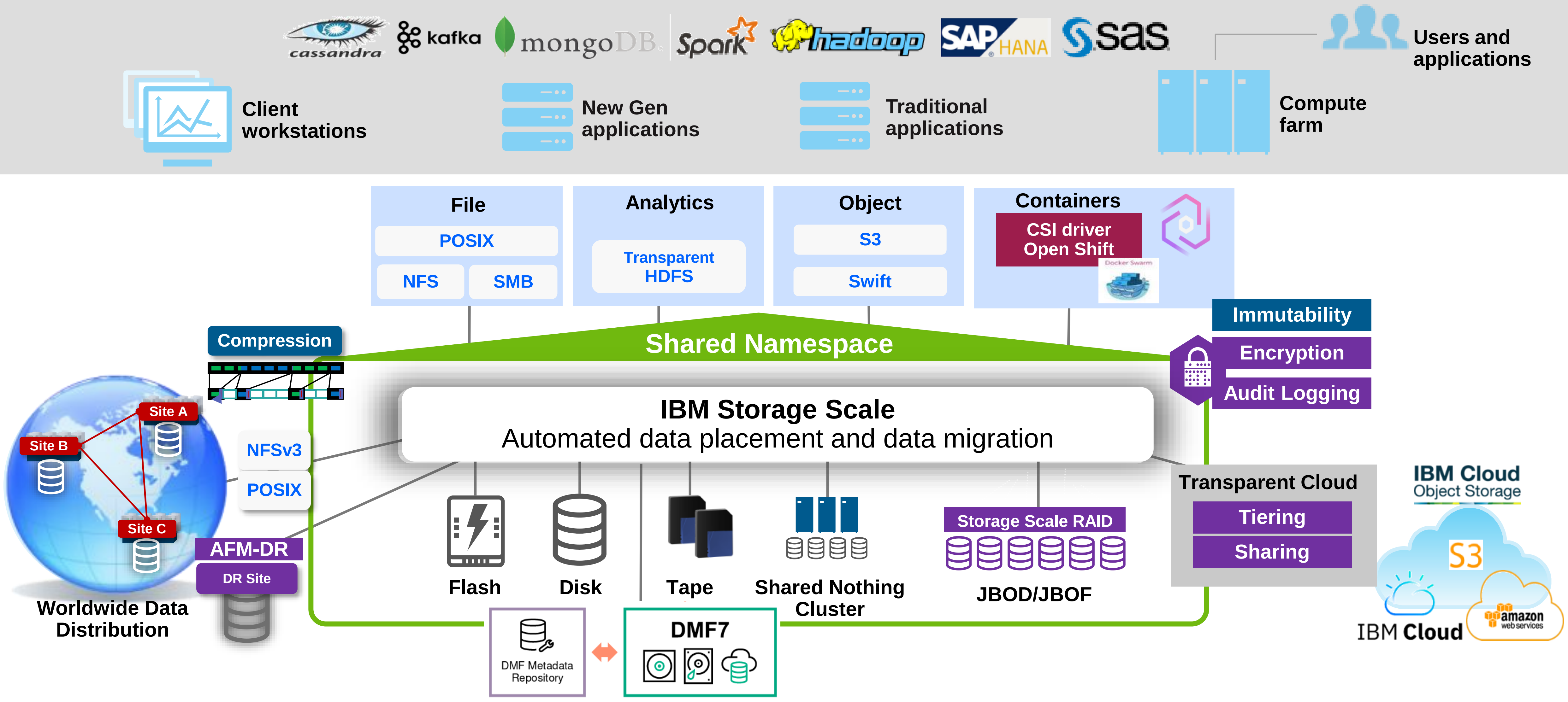
File

Shared

System

Global Data Platform

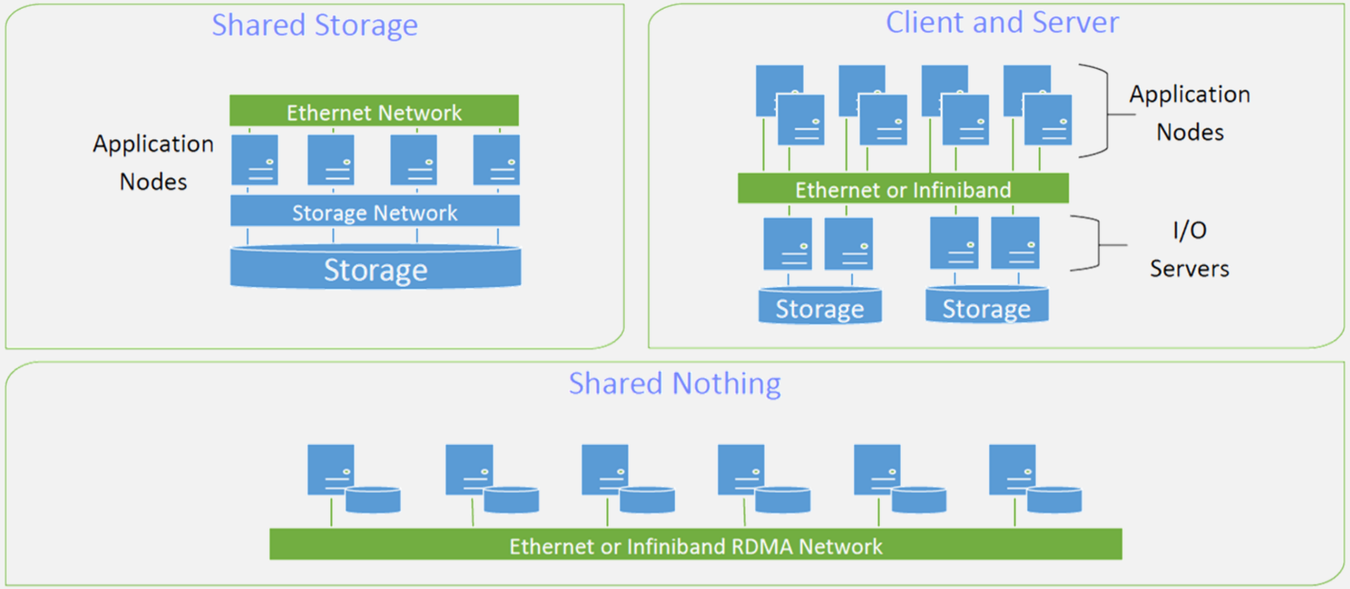
Unleash new storage economics on a global scale



Basic architecture

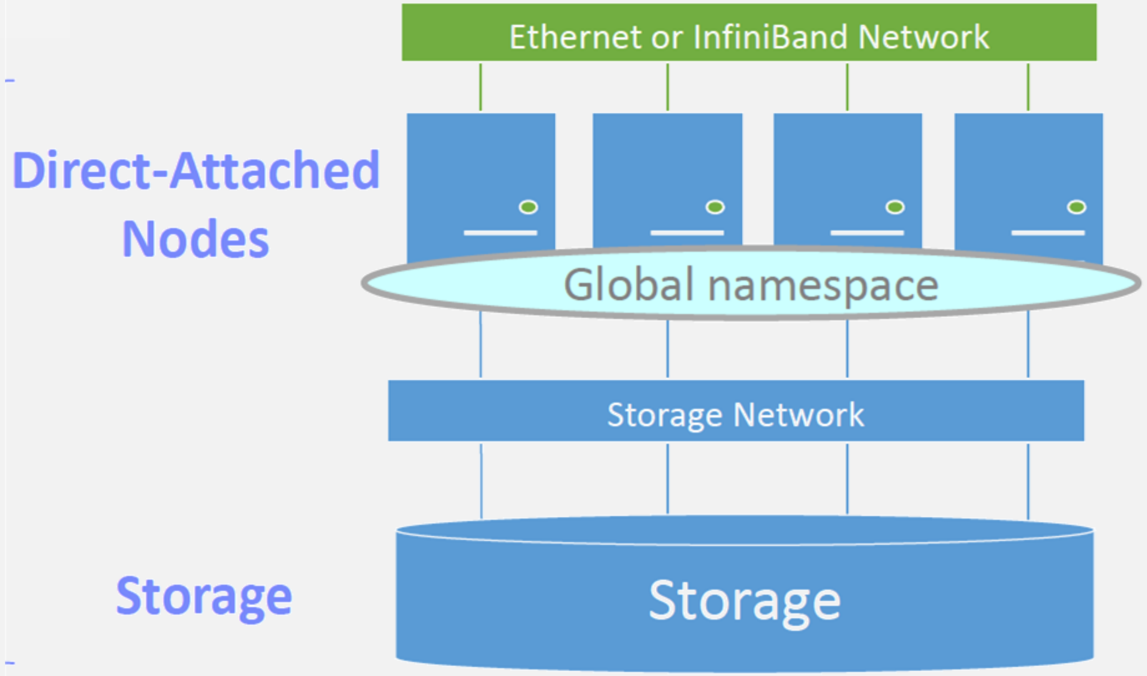
What is a Scale Cluster?

- A IBM Storage Scale cluster is represented by a set of nodes running GPFS
- Scale nodes share a set of file systems and resources Nodes are considered “trusted”
- All or a subset of nodes can administrate the cluster
- Nodes have storage attached and represent cluster topology



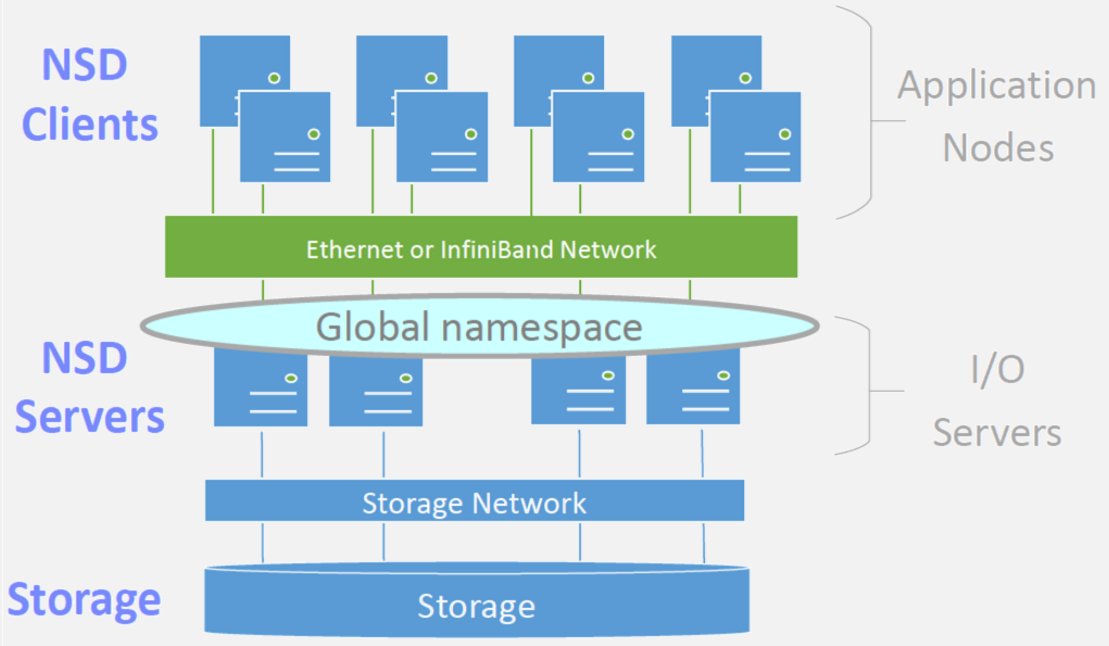
Storage Scale shared storage architecture

- Storage Scale nodes are clustered and represent global namespace
- File system is configured on Network Shared Disk(NSD) devices
- All servers have access to all storage LUNs (same platform)



Storage Scale client server architecture

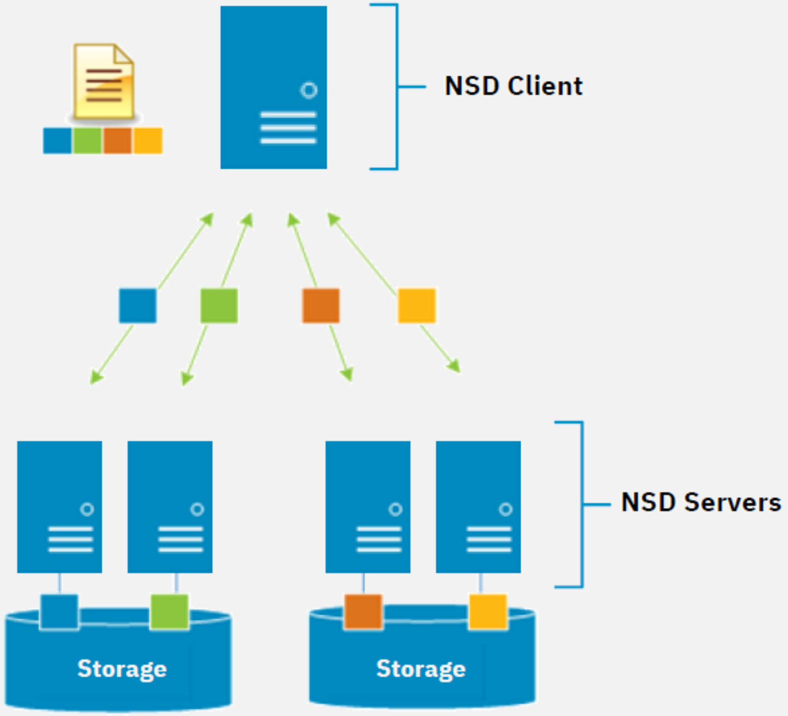
- Storage Scale NSD servers provide access to NSDs for clients
- NSD clients are cluster members and access file systems through NSD protocol



Storage Scale Parallel Architecture

No bottleneck – maximum performance

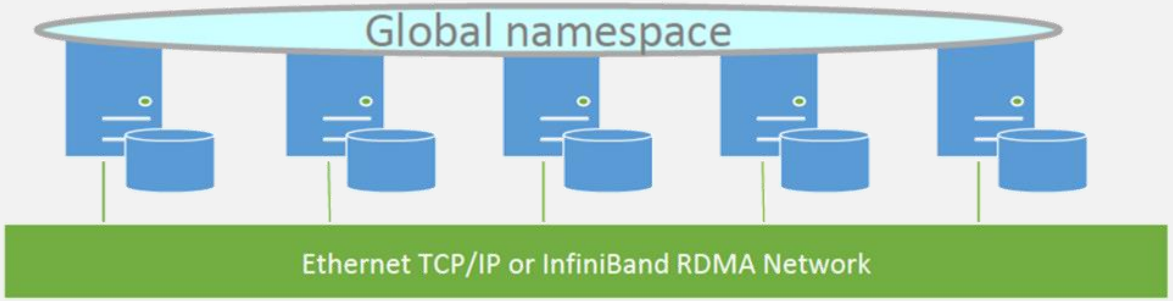
- All NSD servers export to all clients in active-active mode
- Scale stripes files across NSD servers and NSDs in units of file-system block-size
- File-system load spread evenly
- Easy to scale file-system capacity and performance while keeping the architecture balanced



IBM Storage Scale NSD Client does real-time parallel I/O to all the Scale NSD servers and storage volumes/NSDs

Shared Nothing Cluster architecture

- NSD servers have internal disk which are not shared
- Data is striped and replicated over all NSD servers / NSDs
- Also referred to as File Placement Optimizer (FPO)

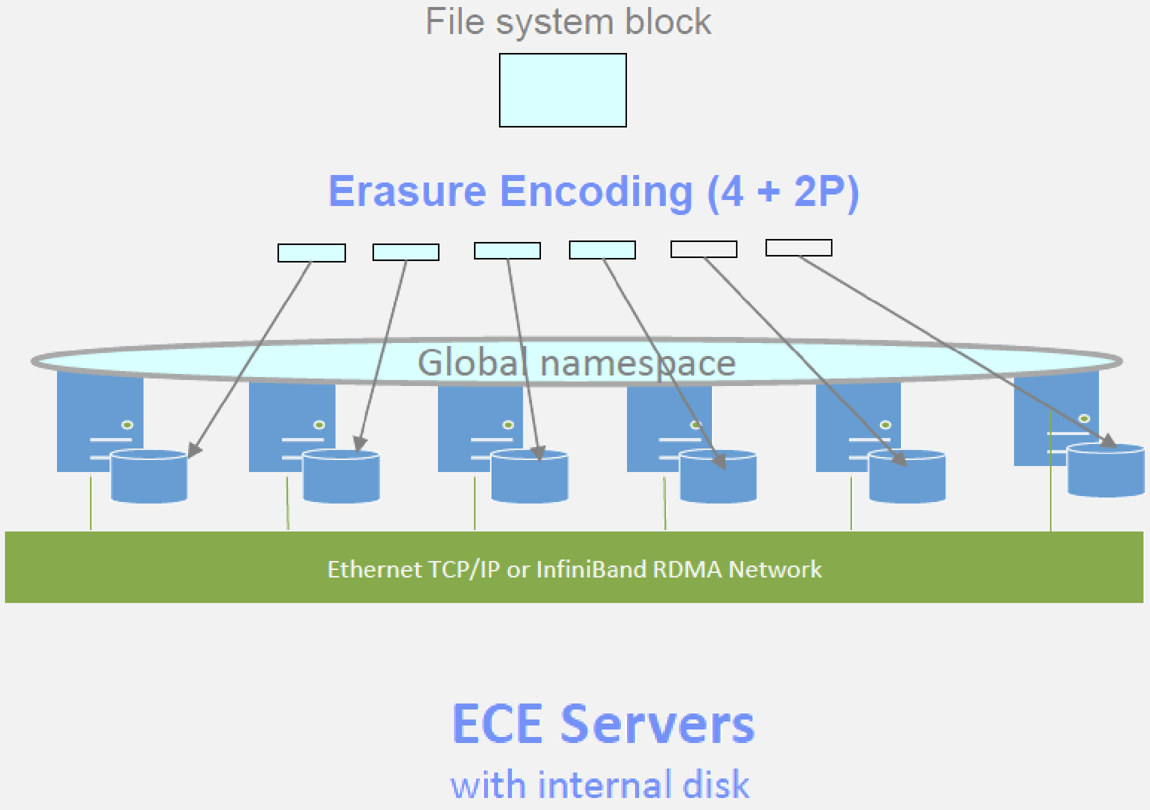


NSD Servers

with internal disk

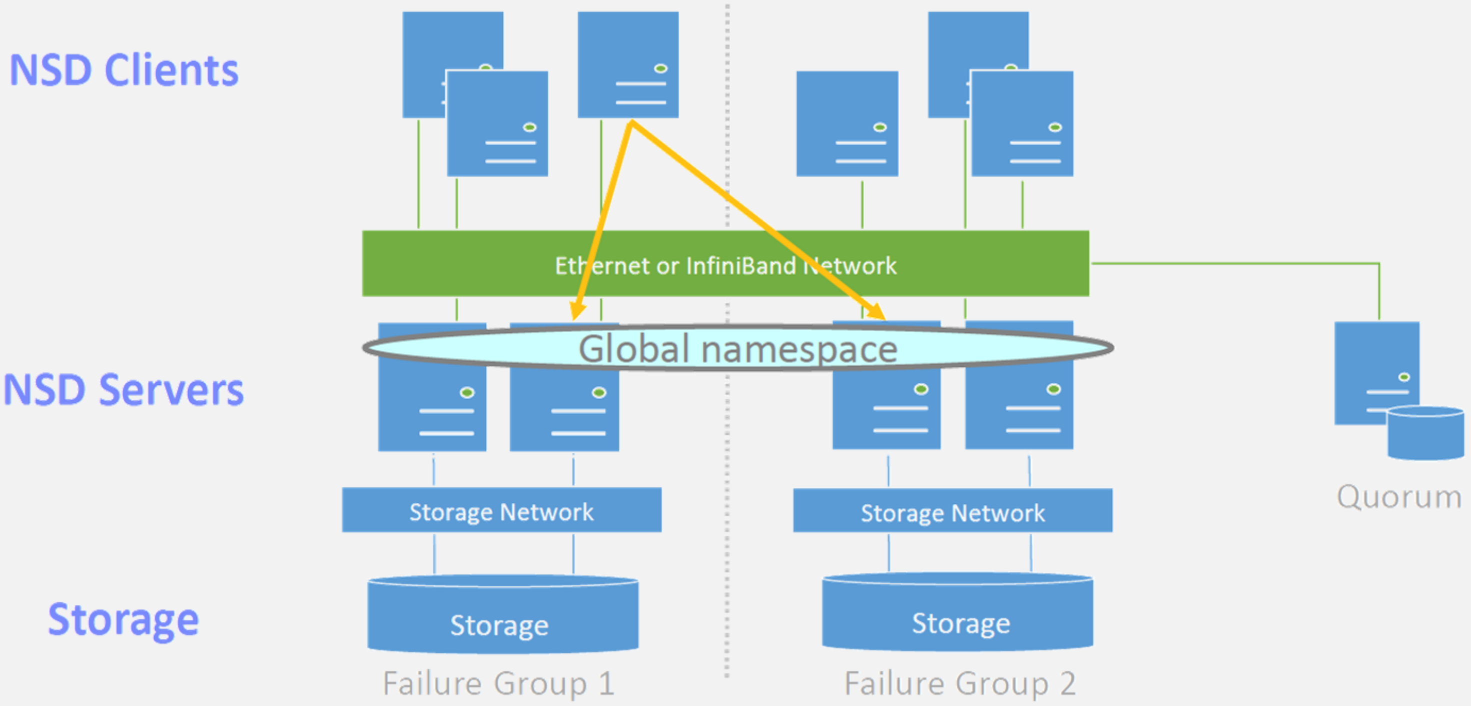
Erasure Code Edition (ECE)

- ECE servers have internal disk where data is stored
- Datablocks are sliced and ersaure encoded (4+2p, 4+3p, 8+2p, 8+3p)
- p servers can fail without losing access to data
- Provides redundancy with less storage


File system block
Erasure Encoding (4 + 2P)
Global namespace
Ethernet TCP/IP or InfiniBand RDMA Network
ECE Servers
with internal disk

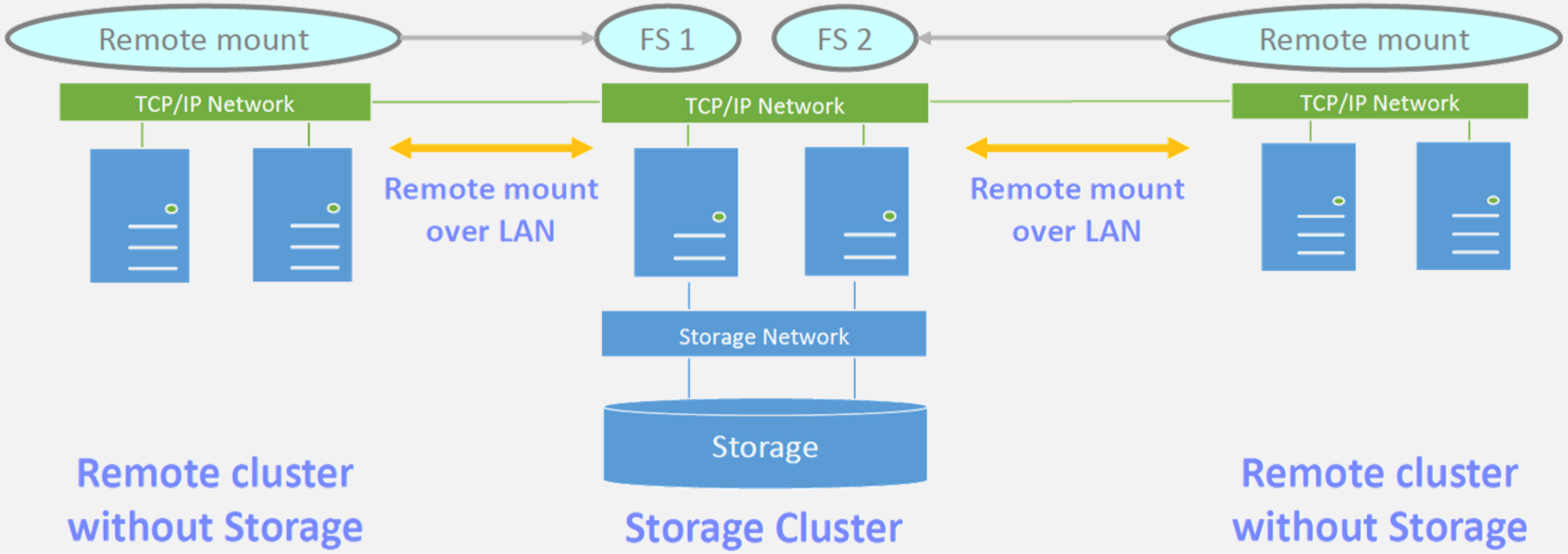
Synchronous replication

- NSD servers are active –active and configured for replication in two failure groups
- NSD client directly writes data to NSD servers in parallel via cluster network
- Scale synchronous replication is optimized for client throughput



Multi-Cluster Architecture

- Independent Scale clusters are active
- Each cluster is an independent administrative domain
- Cluster can or can not have storage
- Clusters can mount file systems of other cluster remotely



Pause

IBM Storage Scale Konzepte Teil 2

Lars Lauber
Storage Technical Specialist
lauberla@de.ibm.com

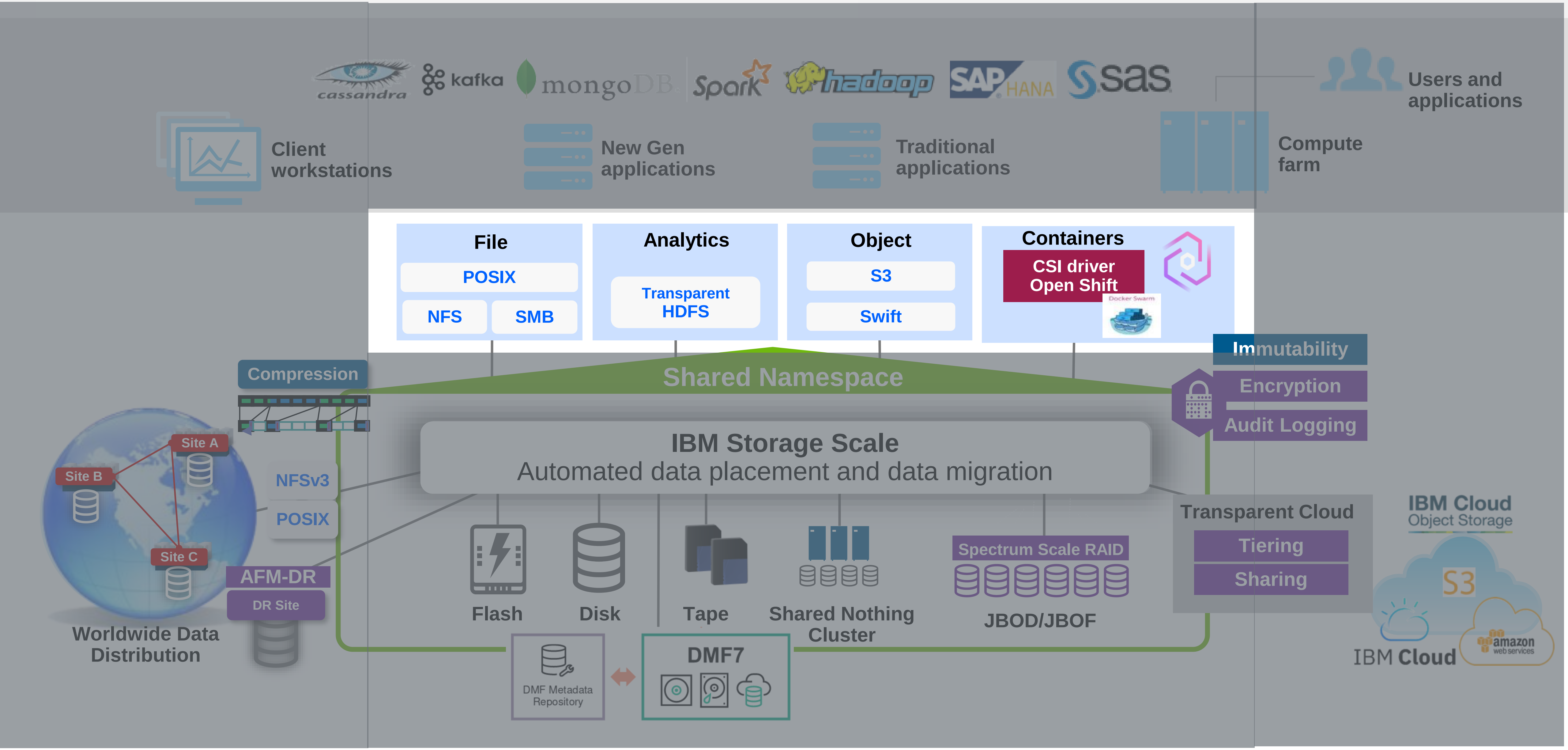
Tiering and ILM Policies

-> “Data Tiering mit Storage Scale” von Max Huber um 15:30 Uhr

Protocols

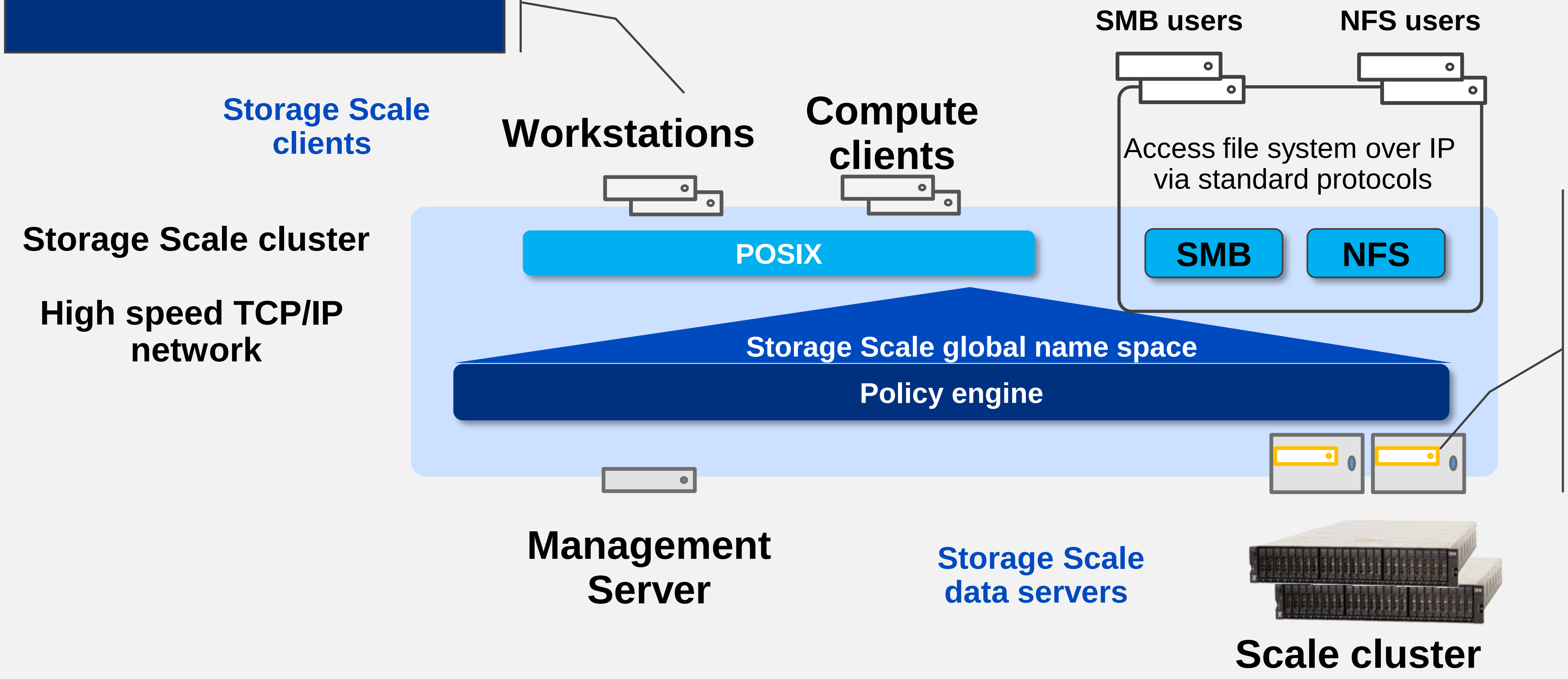
Global Data Platform

Unleash new storage economics on a global scale



Access Data trough client or protocols

Storage Scale clients installed on every workstation that will access data

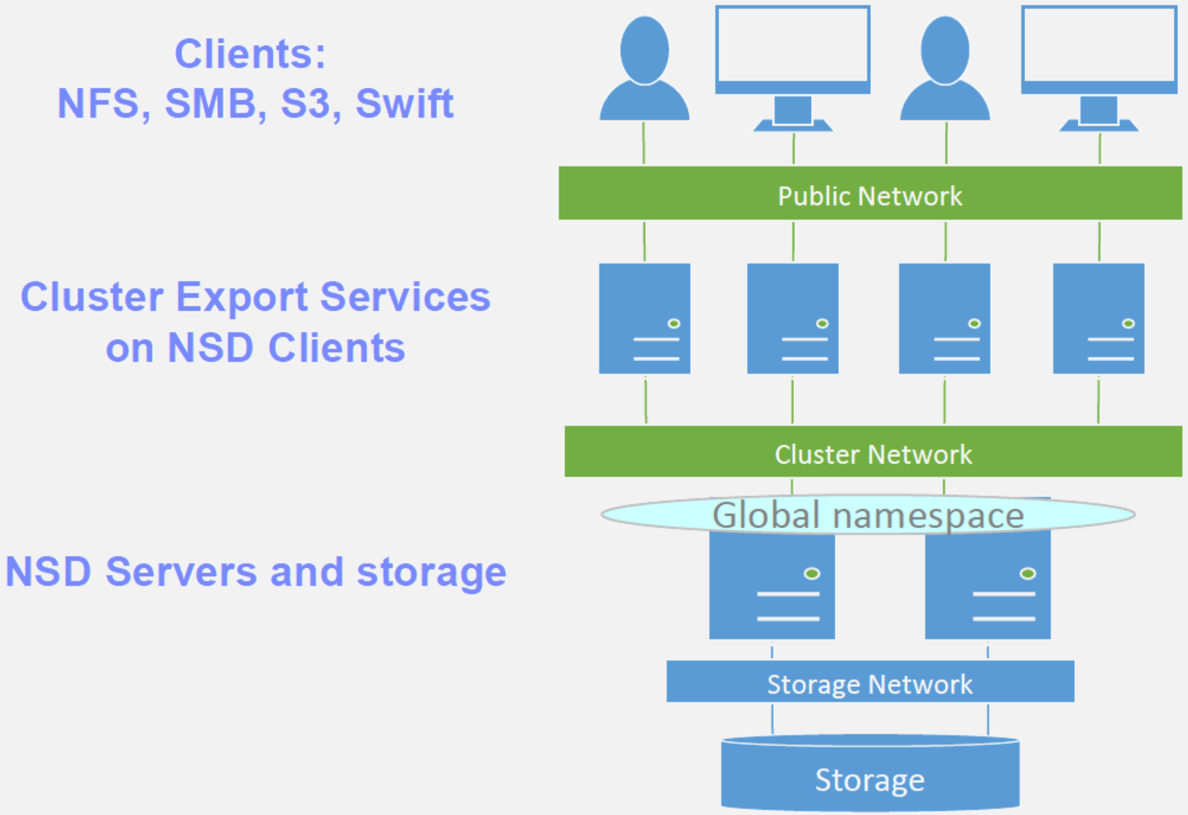


Scale Systems can enable VM node NFS/SMB node for moderate use
(1024 NFS connections
512 SMB connections)

Storage Scale multi protocol support

Scale provides data access through standard protocols

- Scale “protocol” nodes provide NFS, SMB, and HDFS (Swift/S3 up to v5.1.8)
- High availability - if one node fails another one takes over
- Files and objects are stored in IBM Storage Scale file systems
- Also known as Cluster Export Services (CES)



Clients:
NFS, SMB, S3, Swift

Public Network

Cluster Export Services
on NSD Clients

Cluster Network

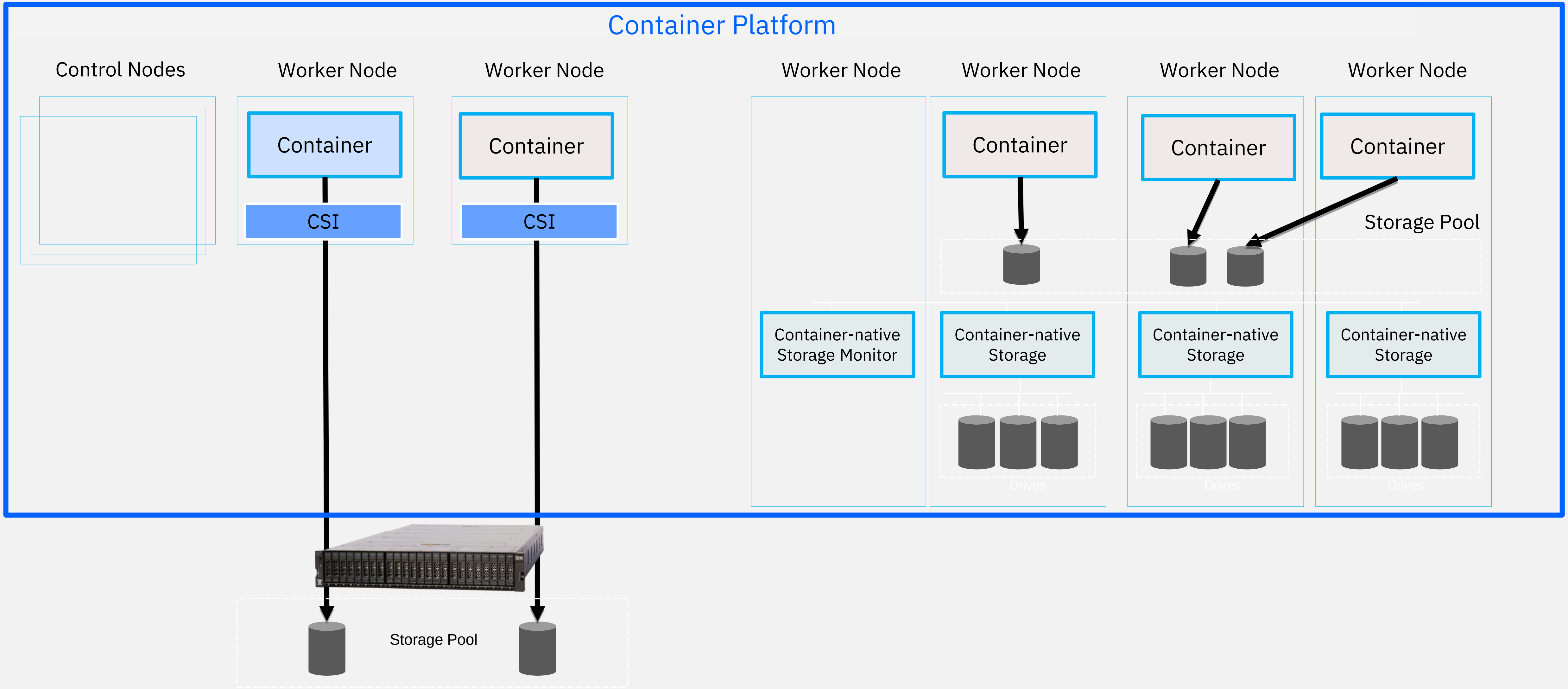
Global namespace

NSD Servers and storage

Storage Network

Storage

OpenShift and Kubernetes container support through CSO or Container Native Access



Container-ready Storage

Container-native Storage

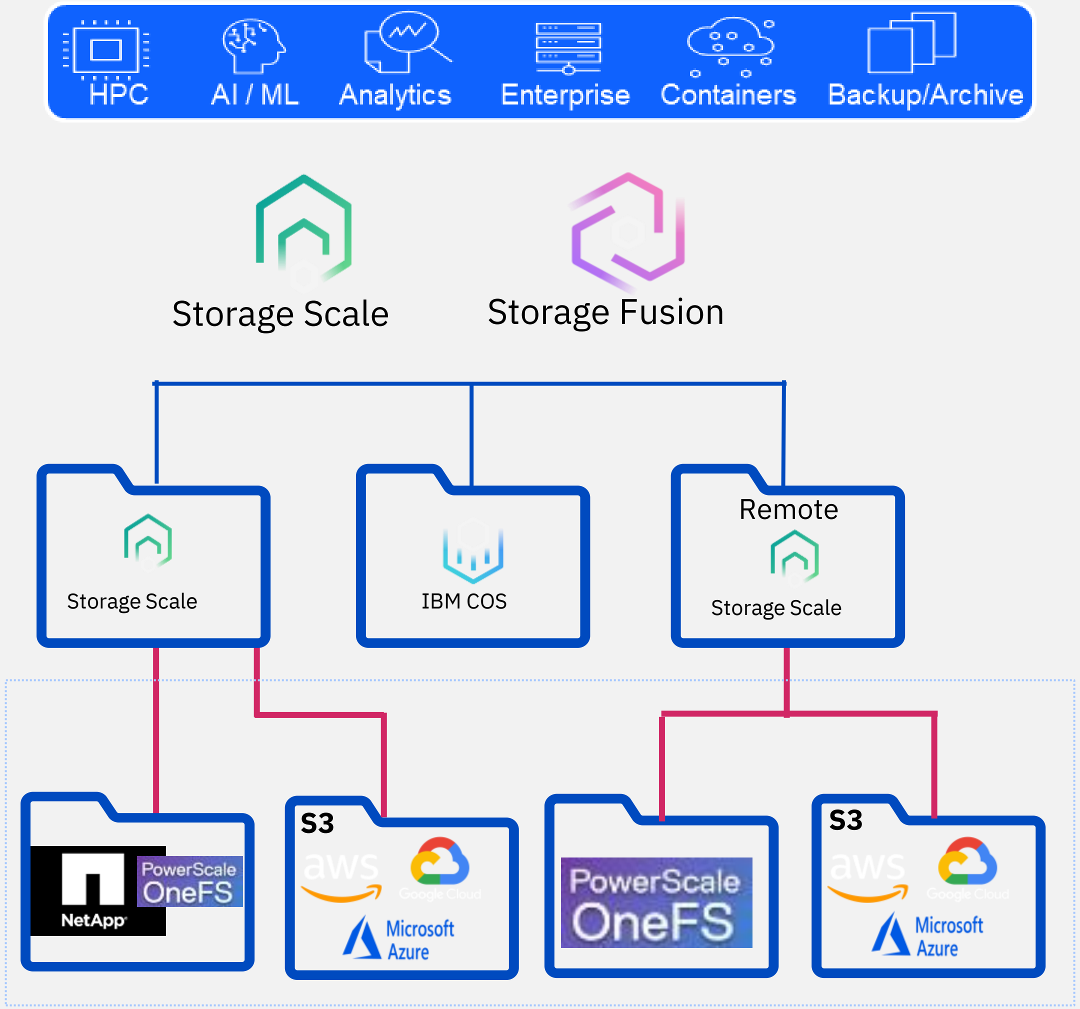
Active File Management (AFM)

Global access to data with active file management (AFM)

Access to multiple sources provides investment protection, global access to data and faster results

- **Investment protection** with an open ecosystem of storage options leveraging multi-vendor and multi-cloud resources
- **Increase application agility** accessing data from edge to core to cloud by bringing more data to applications wherever they are deployed
- **Quickly scale your data** from resources you choose with performance you require
- **Faster access to remote data** by transparently caching remote data locally when needed

- Turns remote file and object data into active capacity (open ecosystem)
- Masks wide-area network latencies and outages by transparently caching data locally
- Individual files in the file set can be cached ?



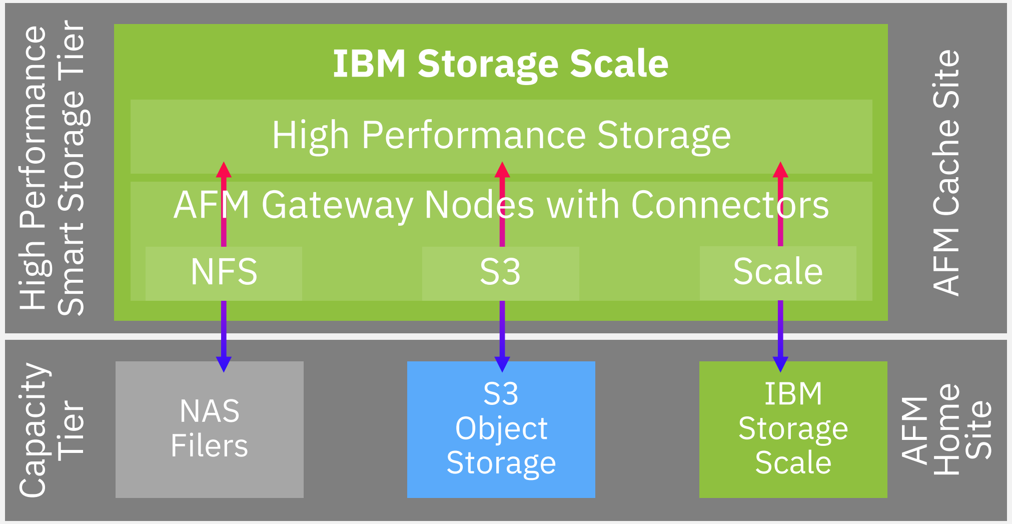
Access any File/Object storage

Data Caching Services (Active File Management) Use cases

Data Virtualization	**Data Collaboration**	**Data Resilience**	**Hybrid cloud / Bursting**
• **Integrate legacy file and object data** stores into single file system to breakdown legacy data silos • **Migrate data to new storage** or continue to use legacy stores • **Create a High-Performance Tier** for analytics for legacy data with transparent data access	• **Geo-distributed collaboration** on data transparently shared between data centers, the cloud and edge sites • **Coalesce data to a home site** from the edge and redistribute it to all sites	• Provide an **asynchronous Disaster Recovery solution** for business continuity over WAN distances • Supports analytics and archival access to passive data	• Dynamically increase computation resources in the cloud and **optimally make required data available for Cloud bursting** • Process data consolidated on S3 Cloud Storage on with **high performance tier in the Cloud** Compute Cluster • **Archive data to S3 Object** storage
Public Cloud Services Use case: Enables end user service to upload large amount of data via Object interface that can be analysed on high performance file system	Research / University Use case: Generate 100's of TB per day across multiple silos, leveraged AFM to provide common namespace with transparent multiprotocol data access	Multinational financial services Use case: Disaster Recovery, retention and compliance data with FileNet and ESS	Research Biopharmaceutical Use case: Multi site / public cloud bursting for collaboration

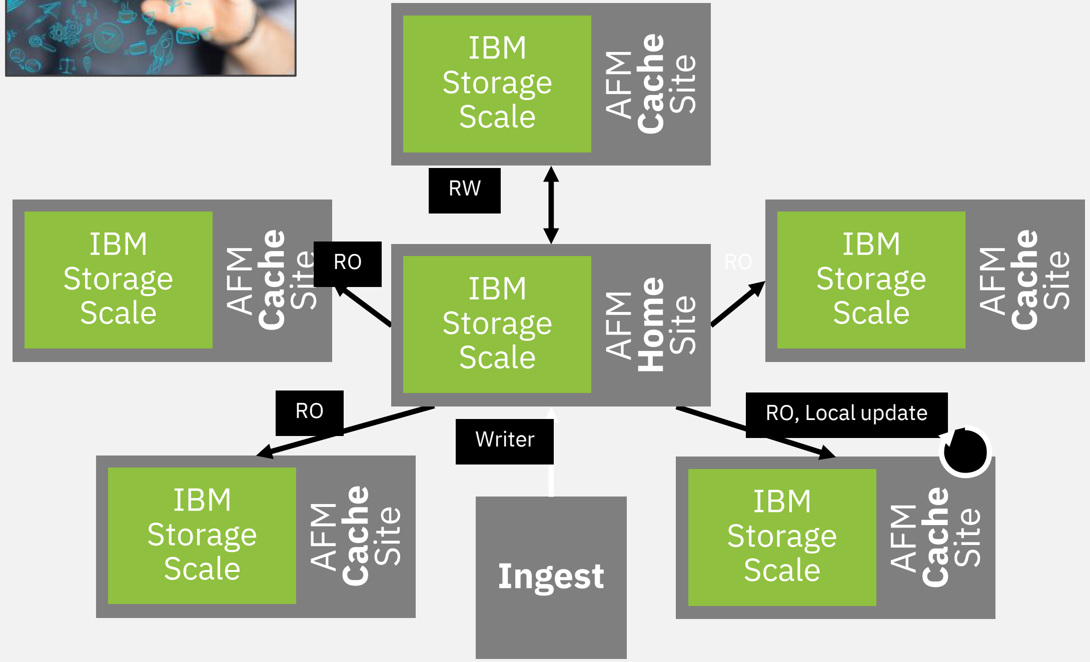
AFM Use Case Details

Data Virtualization



- Vertical caching
- Common namespace across isolated data silos in legacy 3rd party data stores
- Transparent access to all data regardless of silos
- Scale-out Posix performance
- Data export via NFS, SMB, HDFS, Object
- Can be used to seamlessly migrate data to new storage

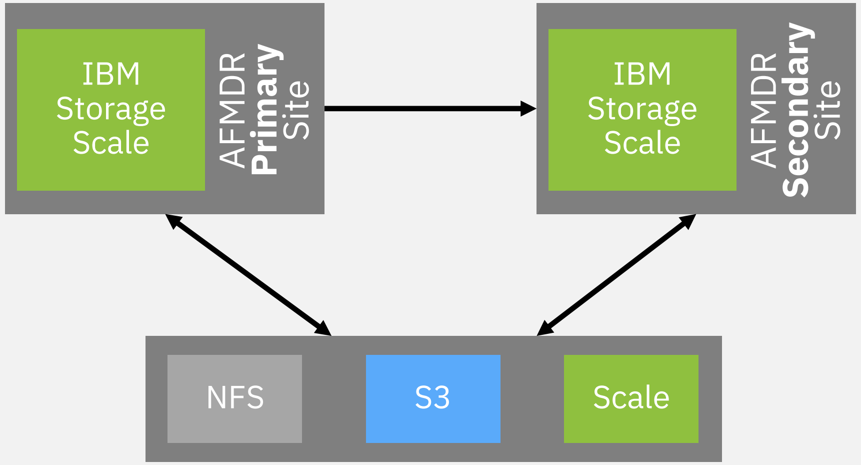
Data Collaboration



- Consistent cache provides a single source of truth with no stale data copies
- Horizontal caching
- Bi-direction traffic from Edge to Center
- Eventually Consistent data cache
- Transparent on-demand data access and transfer
- Policy driven data prefetch and eviction

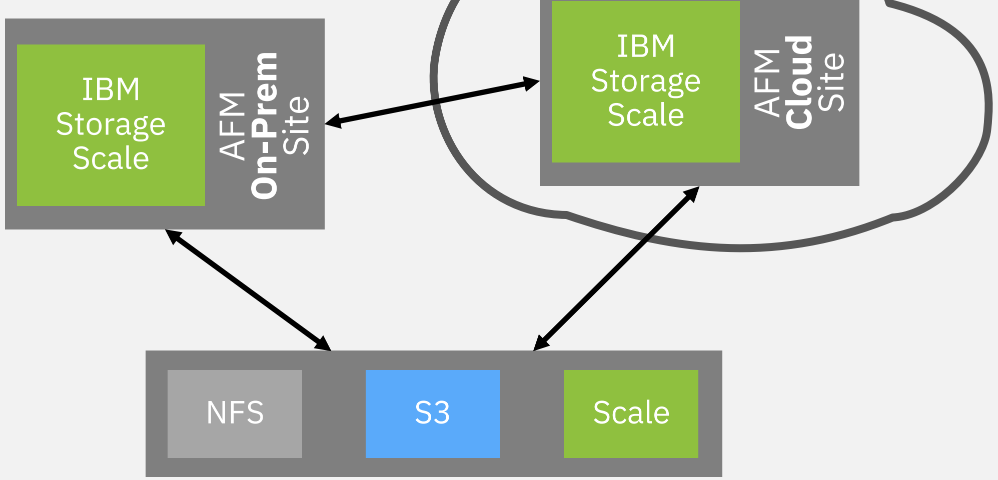
AFM Use Case Details

Data Resilience



- Active-Passive DR over WAN or Cloud
- Designed for high latency and asynchronous DR
- Hot standby failover to DR site
- Automatic fallback data reconciliation
- Read-only access / analytics to all data at passive site

Hybrid cloud / Bursting



- Rapidly expand compute resources to cloud or data centers
- Common file system creates a single namespace across all locations
- Transparent access to data
- Cost effective way to increase compute on existing data
- Analytic results automatically pushed to home site

Additions to S3 Object Storage support

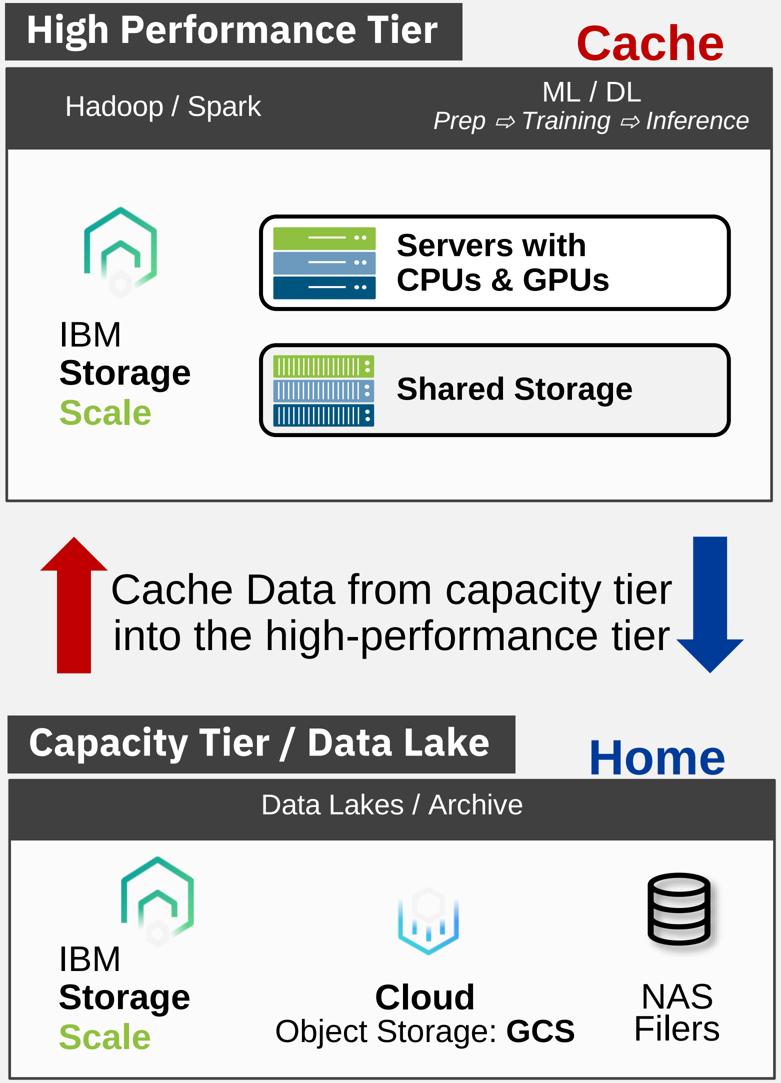
Continued additions of Cloud Object Storages environments

- IBM Cloud Object Storage 5.1.0
- Amazon S3 5.1.0

- Microsoft Azure Blob storage using S3 Gateway 5.1.3
  - AFM doesn't support Microsoft Azure native API
  - Requires intermediate S3 gateway such as Minio
  - Two ways to deploy S3 Gateway (Minio)
    - Deploy as fully managed service in Microsoft Azure market place
    - Deploy Minio on-prem and configure to communicate to Azure Blob

- Google Cloud Platform 5.1.4
  - Use –gcs option while configuring AFM to Google Cloud Storage relationship
  - Creation of bucket not supported via AFM. Ensure bucket exists on GCS

- Seagate Lyve Cloud Object Storage 5.1.5
  - Lyve cloud APIs are almost similar with S3 API



AFM Object Storage Operation modes

ObjectFS mode

- Behaves like normal AFM modes fileset.
- Objects are downloaded on read or on access
- IW and SW modes push files to cloud object storage RO, LU, IW automatically pulls objects from the cloud object storage and stores as files.

ObjectOnly mode

- Default for object operation mode
- No on-demand refresh on read
- Need to manually download metadata/data from COS.
- Objects are uploaded automatically (IW and SW)
- Avoids frequent trips and reduce network contention by selective download/uploads.

	ObjectFS	ObjectOnly
Read Only (RO)	Upload - NA Download – On access (Auto)	Upload - NA Download - On demand
Local Update (LU)	Upload – NA (only On demand) Download – On access (Auto)	Upload – NA (only On demand) Download - On demand
Single Writer (SW)	Upload - Auto Download - On access / On demand	Upload - Auto Download – On demand
Independent Writer (IW)	Upload - Auto Download – On access (Auto)	Upload - Auto Download - On demand
Manual Update (MU)	Upload - On demand Download - On demand	

Let’s talk about Metadata

“Metadata” means different things to different people

- Filesystem metadata
  - Directory structure, access permissions, creation time, owner ID, last modified etc.
- Scientist’s metadata
  - EPIC persistent identifier, Grant ID, data description, data source, publication ID, etc.
- Medical patient’s metadata
  - National Health ID, Scan location, Scan technician, etc.
- Object metadata
  - MD5 checksum, Account, Container, etc.

Filesystem Metadata (MD)

Used to find, access, and manage data (in files)

- Hierarchical directory structure
- POSIX standard for information in the filesystem metadata (Linux, UNIX)
  - POSIX specifies what not how
    - – Filesystem handles how it stores and works with its Metadata
  - Can add other information or functions...
    as long as the POSIX functions work

Why focus on Filesystem Metadata (MD)?

Can be become a performance bottleneck

- Examples:
  - Scan for files changed since last backup
  - Delete files owned by specific user (who just left the company)
  - Migrate least used files from the SSD tier to the disk tier
  - Delete snapshot #5, out of a total of 10 snapshots

Why focus on Filesystem Metadata (MD)?

Can be a significant cost

– For performance, MD may need to be on Flash or SSD or NVMe

- Let’s try to get the capacity and performance right
- Performance on NVMe is not infinite

Scale Filesystem Metadata (MD)

- POSIX compatible filesystem, including multi-user locking (to 10,000+ users!)
- Designed to support **extra IBM Storage Scale functions**:
  - Small amounts of file **data inside Metadata inode**
  - **Multi-site "stretch cluste**r"
    - Via replication of Metadata and Data
  - **HSM / ILM / Tiering** of files to Object storage or Tape tier
    - Via MD Extended Attributes
  - **Fast directory scans** using many servers in parallel ("Policy Engine")
    - Bypasses "normal" POSIX directory functions using special Scale MD design
  - Other types of metadata using **Extended Attributes** (EAs)
    - EAs can "tag" the file with user defined information
  - **Snapshots**

Filesystem MD capacity

Filesystem MetaData capacity is used up- *mostly* by

- File inodes = 1 per file

  + Indirect Blocks as needed (might take up a lot of capacity)

- Directory inodes = 1 per directory

  + Directory Blocks as needed

- Extended Attributes: in Data inode

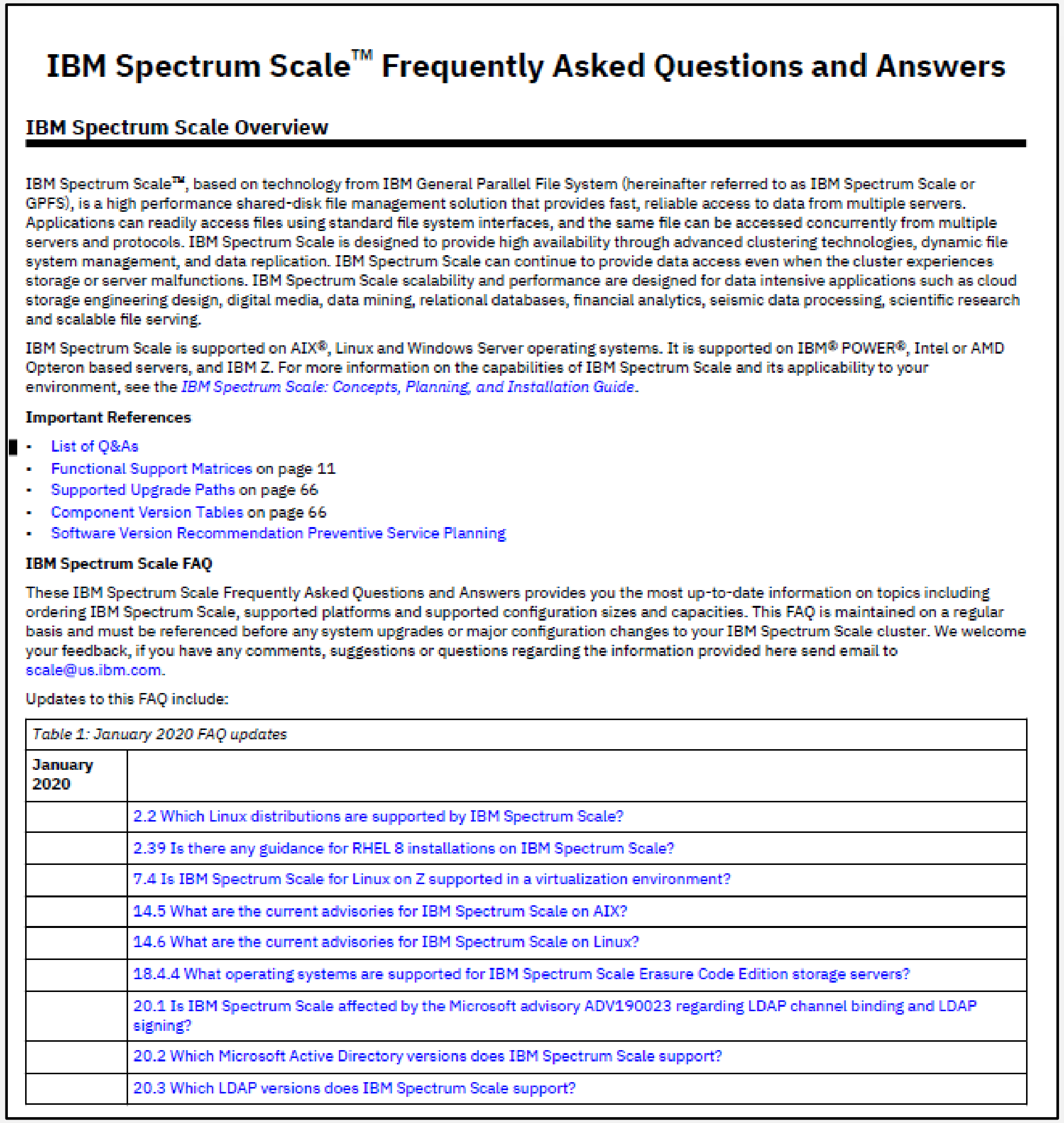
  + EA blocks as needed

- MD also used up by other things, such as snapshots, etc.

Workloads which might need Flash/SSD for MD

- Intensive use of the Scale policy engine
  - Storage Protect Incremental backups (mmbackup),
  - ILM/tiering: disk⇔Flash/SSD, disk⇔tape, etc.
- Snapshots- deletes of a “middle” snapshot in a series
  - Reconciliation to later snapshots is MD intensive
- Lots of “find”, or “create file”, or “delete file” tasks, especially from OS
- Work on small files with data in inodes

IBM Storage Scale FAQ & Redbooks



IBM Spectrum Scale™ Frequently Asked Questions and Answers

IBM Spectrum Scale Overview

IBM Spectrum Scale™, based on technology from IBM General Parallel File System (hereinafter referred to as IBM Spectrum Scale or GPFS), is a high performance shared-disk file management solution that provides fast, reliable access to data from multiple servers. Applications can readily access files using standard file system interfaces, and the same file can be accessed concurrently from multiple servers and protocols. IBM Spectrum Scale is designed to provide high availability through advanced clustering technologies, dynamic file system management, and data replication. IBM Spectrum Scale can continue to provide data access even when the cluster experiences storage or server malfunctions. IBM Spectrum Scale scalability and performance are designed for data intensive applications such as cloud storage engineering design, digital media, data mining, relational databases, financial analytics, seismic data processing, scientific research and scalable file serving.

IBM Spectrum Scale is supported on AIX®, Linux and Windows Server operating systems. It is supported on IBM® POWER®, Intel or AMD Opteron based servers, and IBM Z. For more information on the capabilities of IBM Spectrum Scale and its applicability to your environment, see the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Important References

- List of Q&As
- Functional Support Matrices on page 11
- Supported Upgrade Paths on page 66
- Component Version Tables on page 66
- Software Version Recommendation Preventive Service Planning

IBM Spectrum Scale FAQ

These IBM Spectrum Scale Frequently Asked Questions and Answers provides you the most up-to-date information on topics including ordering IBM Spectrum Scale, supported platforms and supported configuration sizes and capacities. This FAQ is maintained on a regular basis and must be referenced before any system upgrades or major configuration changes to your IBM Spectrum Scale cluster. We welcome your feedback, if you have any comments, suggestions or questions regarding the information provided here send email to scale@us.ibm.com.

Updates to this FAQ include:

Table 1: January 2020 FAQ updates	
January 2020	
	2.2 Which Linux distributions are supported by IBM Spectrum Scale?
	2.39 Is there any guidance for RHEL 8 installations on IBM Spectrum Scale?
	7.4 Is IBM Spectrum Scale for Linux on Z supported in a virtualization environment?
	14.5 What are the current advisories for IBM Spectrum Scale on AIX?
	14.6 What are the current advisories for IBM Spectrum Scale on Linux?
	18.4.4 What operating systems are supported for IBM Spectrum Scale Erasure Code Edition storage servers?
	20.1 Is IBM Spectrum Scale affected by the Microsoft advisory ADV190023 regarding LDAP channel binding and LDAP signing?
	20.2 Which Microsoft Active Directory versions does IBM Spectrum Scale support?
	20.3 Which LDAP versions does IBM Spectrum Scale support?

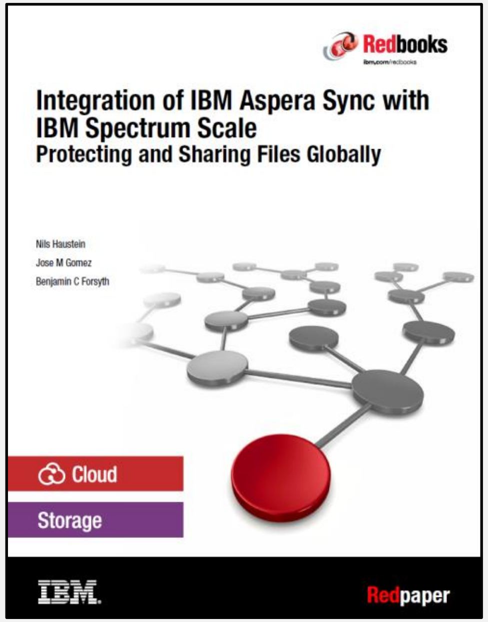
Contact
mailto:scale@us.ibm.com
if you need more info.

Knowledge Center:
https://www.ibm.com/docs/en/spectrum-scale/

IBM Spectrum Scale Security

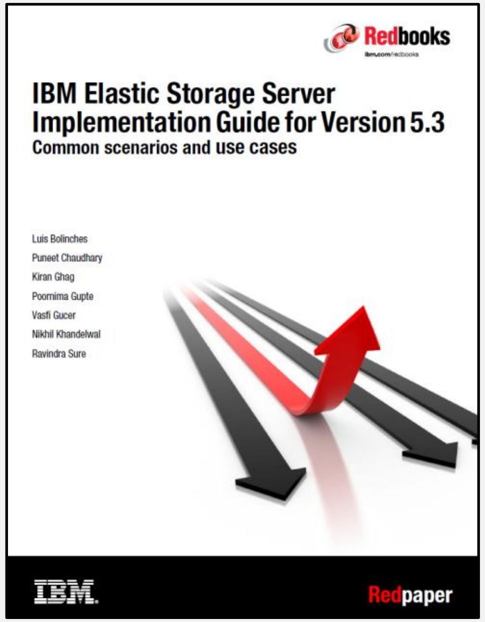
Felipe Knop
Sandeep R. Patil
Larry Coyne

In partnership with
IBM Academy of Technology



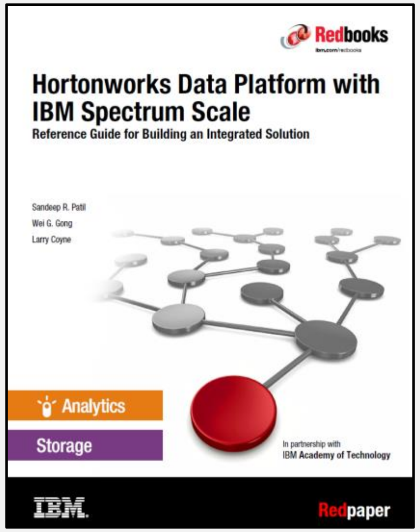
Integration of IBM Aspera Sync with IBM Spectrum Scale
Protecting and Sharing Files Globally

Nils Haustein
Jose M Gomez
Benjamin C Forsyth



IBM Elastic Storage Server Implementation Guide for Version 5.3
Common scenarios and use cases

Luis Bolinches
Puneet Chaudhary
Kiran Ghag
Poornima Gupte
Vasfi Gucer
Nikhil Khandelwal
Ravindra Sure



Hortonworks Data Platform with IBM Spectrum Scale
Reference Guide for Building an Integrated Solution

Sandeep R. Patil
Wei G. Gong
Larry Coyne

In partnership with
IBM Academy of Technology