

Back to the fundamental:

Architecture and concept of Scale and ESS

Mirko Bertaccini

Storage Technical Specialist

mirko.bertaccini@ibm.com



Storage Scale

Agenda

- 1** Introduction
- 2** Architecture of Storage Scale
- 3** Architecture of Storage Scale System
- 4** Advanced Features

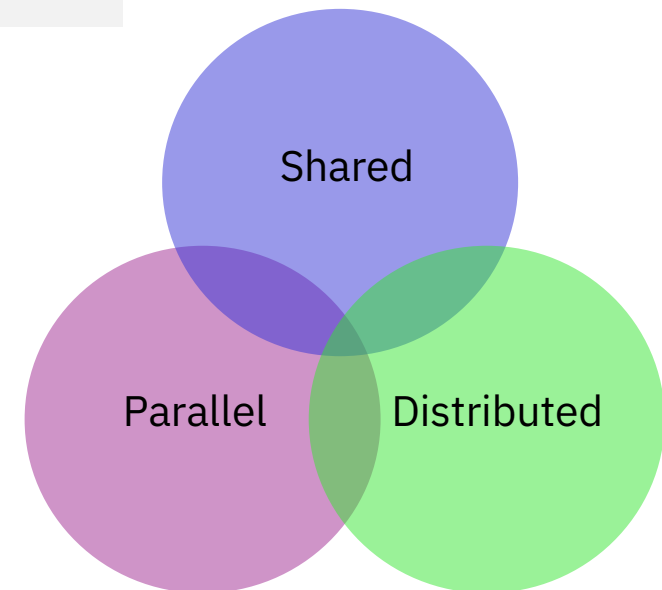


1- Introduction: A General Parallel File System

Scale is a global file system

Storage Scale is a **software:**
highly scalable distributed parallel POSIX filesystem

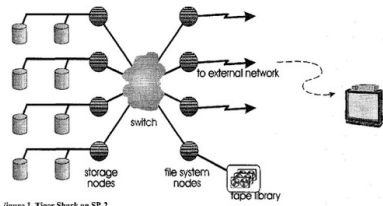
- Able to run on systems under **Linux, AIX, and Windows** on **x86, ARM, IBM Power, and IBM Z**
- Building upon **any** block storage (from local disk to SAN-attached, not necessarily from IBM)
- With advanced data management features built into it that go well beyond the capabilities of traditional file systems, able to tier and cache from many storage types
- Features layered on top of it to make that storage accessible through NFS, SMB, and S3 protocols.



The Many Lives of IBM Storage Scale

1993

Began as the **Tiger Shark File System**. Designed to support multi-media applications (where the CLI prefix “mm...” comes from)



1998

Leveraged **Vesta** filesystem’s ability to partition files and enable parallel access to create **GPFS** (General Parallel File System)

IBM
GPFS

2007

Leveraged some aspect of GPFS in an appliance-based implementation of the **Scale out File Services** (SoFS) for NAS (SONAS)

2015

Rebranded as
“Spectrum Scale”

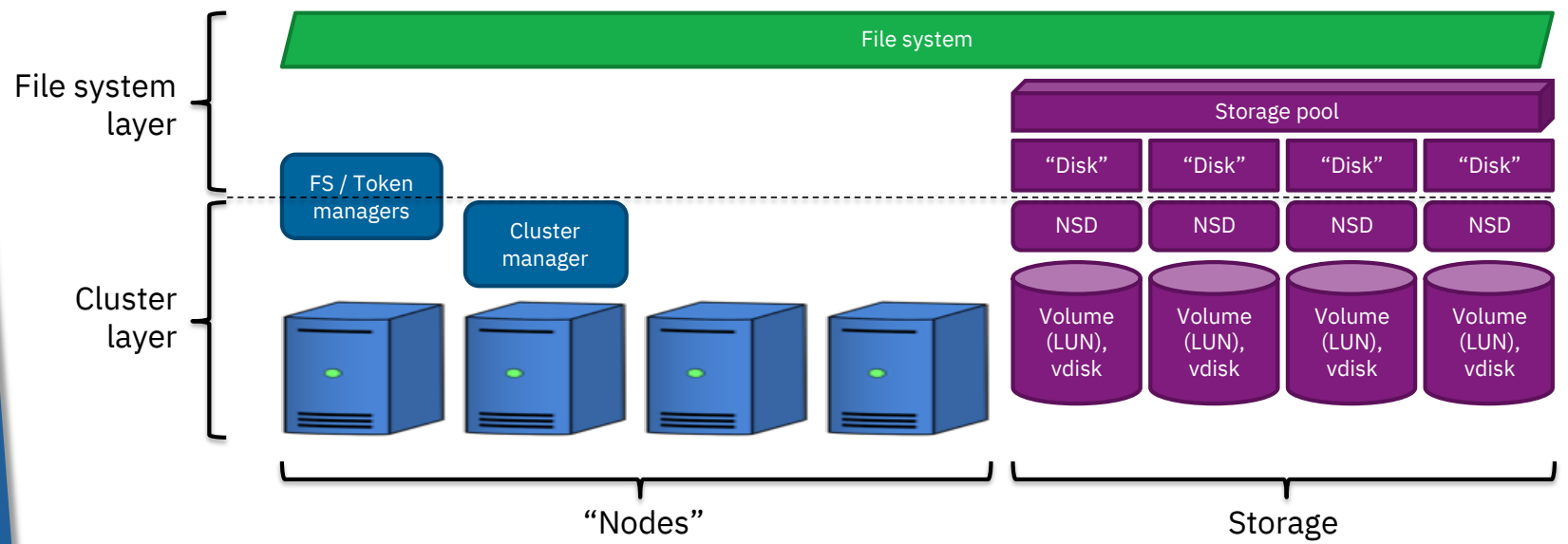


2022

Rebranded as
“IBM Storage Scale”

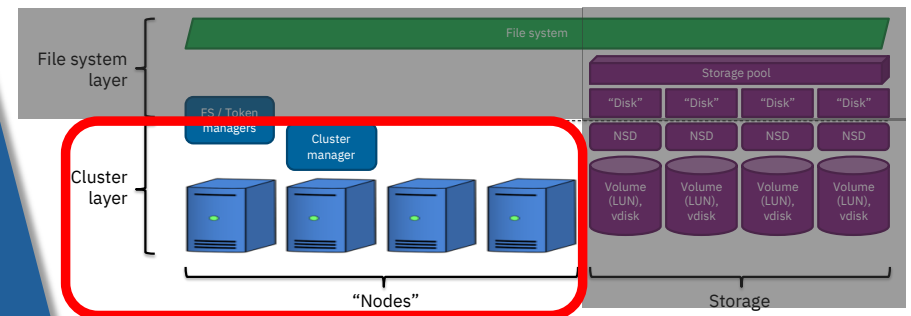


2- Architecture of Storage Scale



Architecture of Storage Scale

↳ **Cluster and nodes**



Storage Scale Node Architecture

Every system (or **node**) running Scale software has these components:

- Administrative commands :

on Linux or AIX, these are found in /usr/lpp/mmfs/bin, so you probably want to put this directory on your PATH)

- A multi-threaded daemon: *mmfsd*

Most local node state is stored in the directory tree rooted at /var/mmfs.

- Several kernel extensions:

Linux nodes will have the Scale open-source portability layer, with several kernel modules

Scale leverages the capabilities of the operating system for networking (e.g., bonding) and accessing block storage (e.g., multipath).

Think of it as a systems-level software application.

```
# ls /usr/lpp/mmfs/bin
auditConsumerManager      mmdelnsd                mmunmount
auth.snap.py              mmdelsnapshot           mmuserauth
callhome.snap.py          mmddevdiscover          mmvaryonvg
ces.snap.py               mmdf                    mmwatch
cleanupkeystonedb         mmddiag                 mmxargs
core.snap.py              mmdsh                   mmyfields
daemonize                 mmdsm
msgqueueDaemonManager     mmdspmsg                msgqueue.snap.py
gpfsrunlevel              mmdump                  nfs.snap.py
gpfs.snap                 mmdumpfilocks           object.snap.py
gui.snap.py               mmdumpkthreads          objmetric_count.py
hadoop.snap.py            mmdumpperfdata          perfmon.snap.py
kdump                     mmdynamicmap            perf.snap.py
kdump-3.10.0-693.el7.x86_64 mmeditac1               protect.snap.py
lxtrace
...

# ps -fC mmfsd
UID      PID   PPID  C  STIME TTY      TIME CMD
root    27473 27153  0  Oct09 ?        00:16:10 /usr/lpp/mmfs/bin/mmfsd

# lsmod | grep mmfs
mmfs26                2618329  2
mmfslinux              800204  25 mmfs26
tracedev              48579   3 mmfs26,mmfslinux

# ls /var/mmfs
ccr ces etc gen mmbackup mmpmon mmsysmon ssl tmp
```

Scale administrative commands

Commands and binaries are stored in
/usr/lpp/mmfs/bin

All commands intended for customer use have the prefix **mm-**

- Generally, the form is:

mm{verb}{object}

{verb} is an action like create, add, delete, change, list, ...

{object} is class of object of interest, like cluster, node, nsd, disk, file system, ...

Examples:

- **mmlscluster**
- **mmlsconfig**

- Note the new Native REST API and its `scalectl` command, an emerging role-based replacement for many **mm-** commands.

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

GPFS cluster information

=====

```
GPFS cluster name:      slsr.local
GPFS cluster id:        14559237835526968155
GPFS UID domain:        slsr.local
Remote shell command:   /usr/bin/ssh
Remote file copy command: /usr/bin/scp
Repository type:        CCR
```

Node	Daemon node name	IP address	Admin node name	Designation
1	a-scale01	10.0.200.11	a-scale01	quorum-manager
2	b-scale01	10.0.200.21	b-scale01	quorum-manager
3	tiebreak	10.0.200.31	tiebreak	quorum
4	a-scale02	10.0.200.12	a-scale02	
5	a-scale03	10.0.200.13	a-scale03	
6	b-scale02	10.0.200.22	b-scale02	
7	b-scale03	10.0.200.23	b-scale03	

Did you know?

- The **mm-** prefix hearkens to Scale's origin as the "Multi-Media File System".
- Many internal commands start with the prefix **ts-**, for an earlier Tigershark File System
- Don't use the **ts-** commands unless directed by support!

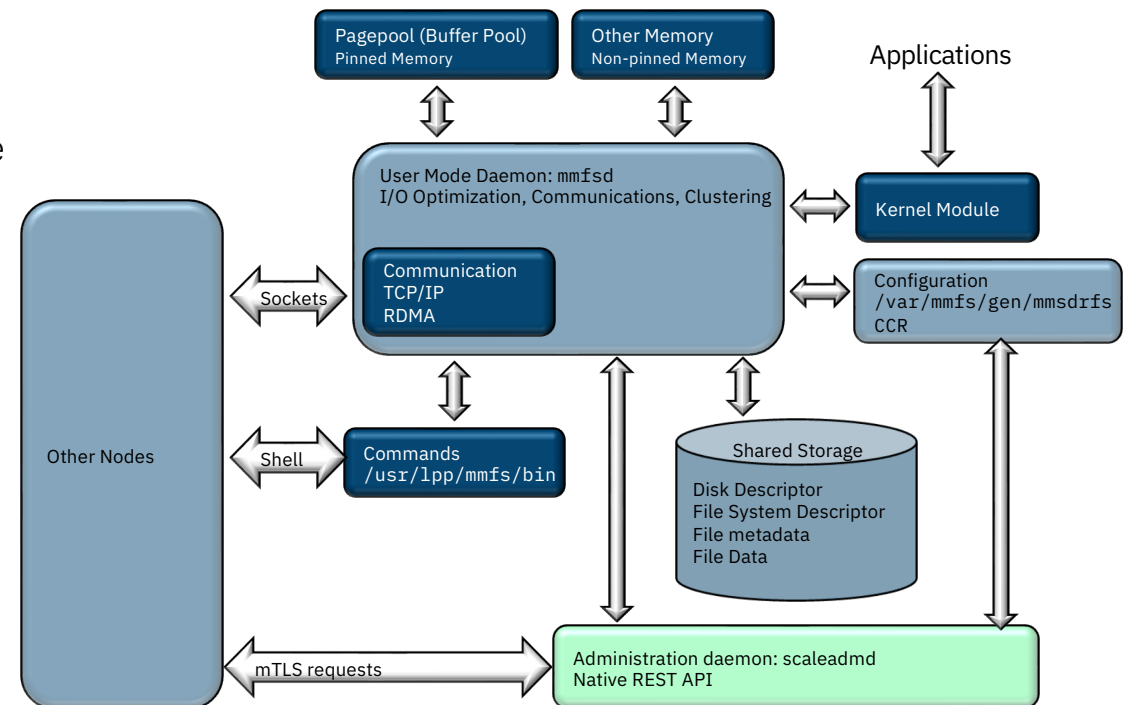
Scale linux kernel extensions

The Scale kernel extensions link to the operating system “vnode” and “VFS” (virtual file system) internal interfaces.

- Scale appears to the application as just another file system
- Due to that, Scale satisfies a system call by the resources available in the kernel extension or sends a request to the daemon to complete the request. Moreover, the VFS provides facilities, such as dcache, that can be used by file systems

I/O Flow

- Applications make file system calls to the kernel
- VFS presents calls to the Scale kernel extension
- Scale kernel extensions will either satisfy requests using information already available or send a message to the Scale daemon mmfsd to complete the request.



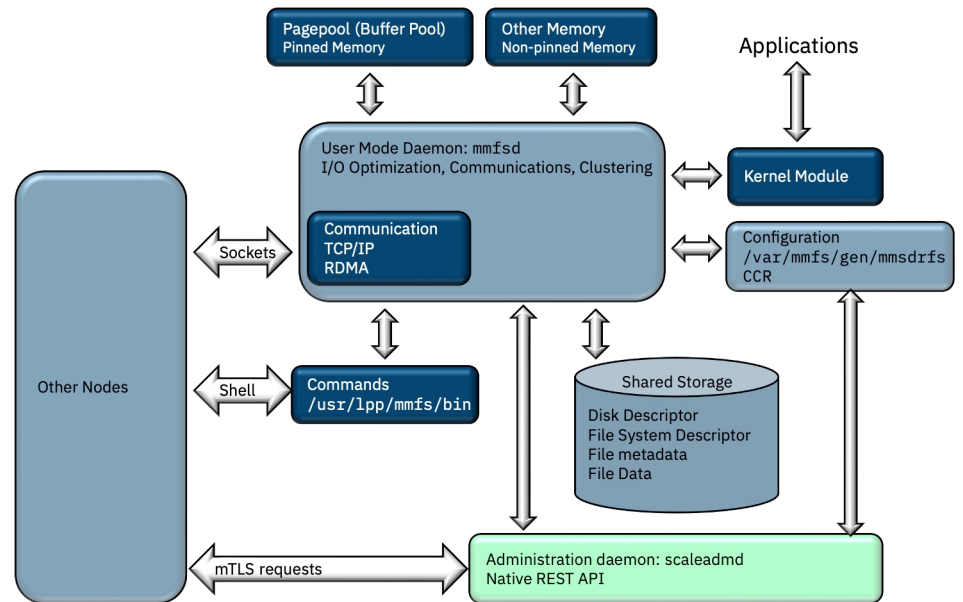
Storage Scale Daemon

The Scale daemon, mmfsd, performs:

- all I/O and buffer management
- reading ahead for sequential reads
- write behind for all writes not specified as synchronous.

All I/O are protected by a “distributed lock manager” to ensure file system consistency

The mmfsd is highly multi-threaded, with some threads dedicated to specific functions.



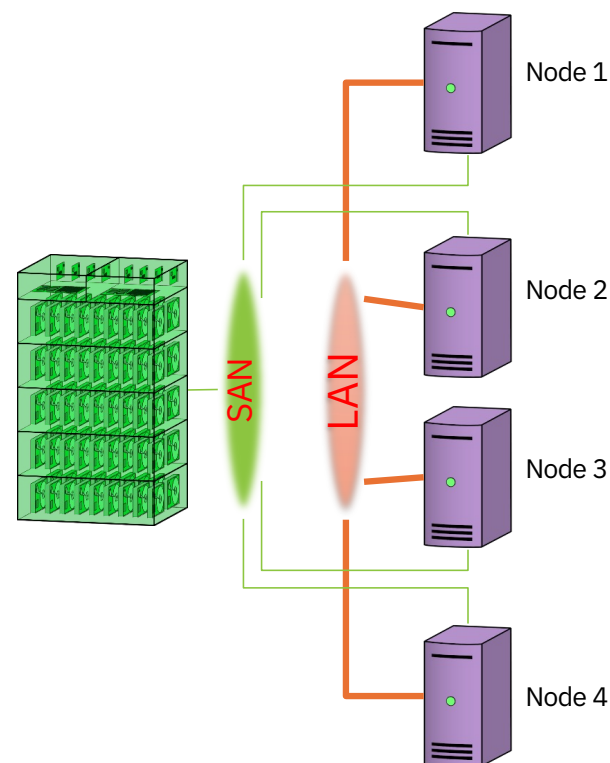
```
# ps -Lf -C mmfsd | head
UID      PID  PPID  LWP  C  NLWP  STIME  TTY      TIME  CMD
root    27473 27153 27473  0  504  Oct09  ?        00:00:00 /usr/lpp/mmfs/bin/mmfsd
root    27473 27153 27491  0  504  Oct09  ?        00:00:01 /usr/lpp/mmfs/bin/mmfsd
root    27473 27153 27493  0  504  Oct09  ?        00:00:00 /usr/lpp/mmfs/bin/mmfsd
root    27473 27153 27494  0  504  Oct09  ?        00:00:00 /usr/lpp/mmfs/bin/mmfsd
root    27473 27153 27495  0  504  Oct09  ?        00:00:21 /usr/lpp/mmfs/bin/mmfsd
root    27473 27153 27496  0  504  Oct09  ?        00:00:00 /usr/lpp/mmfs/bin/mmfsd
root    27473 27153 27497  0  504  Oct09  ?        00:00:00 /usr/lpp/mmfs/bin/mmfsd
root    27473 27153 27498  0  504  Oct09  ?        00:00:00 /usr/lpp/mmfs/bin/mmfsd
root    27473 27153 27499  0  504  Oct09  ?        00:00:00 /usr/lpp/mmfs/bin/mmfsd
```

Scale cluster

A Scale **cluster** is a group of systems, the **nodes**, with Scale software installed and configured into a single administrative grouping:

- All nodes have a common view of the data.
- The nodes are tightly coupled, trusting each others' authentication of users.
- A cluster can have several Scale file systems.
- A cluster may share a file system with an authenticated remote cluster.

Clustering ensures all nodes in the cluster have a *consistent* view of the Scale file systems, even when more than one node is actively accessing a file system (or even the same file).



Cluster role

Many nodes may be **configured** to be part of the cluster.

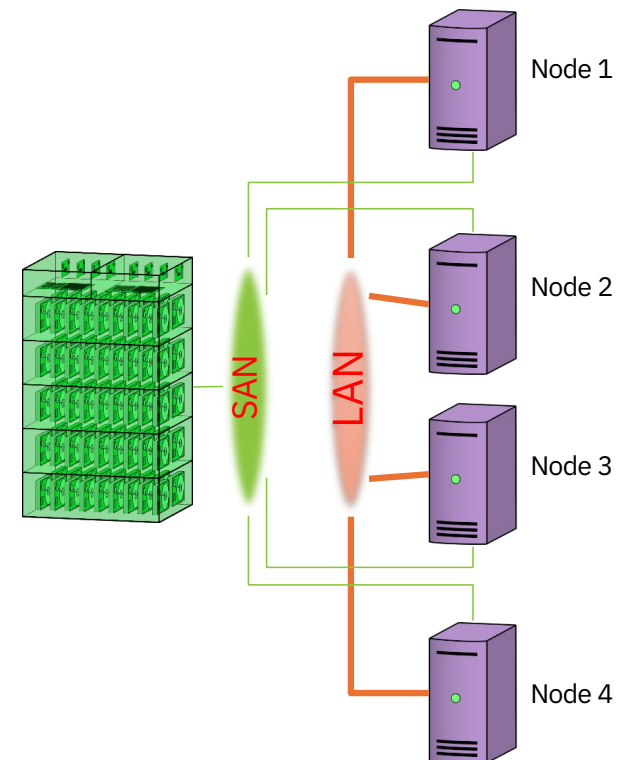
The **active cluster member** is the set of nodes currently communicating with each other and sharing resources.

A **cluster manager** keeps track of which nodes are currently part of the active cluster, controls access to managed cluster resources and maintains the configuration of the cluster.

To avoid the cluster manager from being a single point of failure, this is a **role** that may run on one of several cluster nodes.

Because failures happen, the cluster manager may need to **expel** “dead” nodes from the active cluster and **fence** them from resources like shared disks:

- Perhaps a node has failed
- Perhaps a node's network has failed



Cluster role

NSD server

- Makes storage available over the LAN to nodes
- As we'll see, a node is an NSD server simply if it is assigned NSDs to serve.

Manager

- Runs token managers, and a candidate for the file system manager

Graphical User Interface (GUI)

- Visually monitor performance
- GUI control over many cluster and file system functions
- RESTful API for managing the cluster

Performance monitoring

- Monitor performance of nodes, storage, network through CLI or GUI

AFM gateway

- Used to move data when using AFM

Cluster Export Services ("protocol node")

- Exports data from file systems through SMB, NFS, Swift

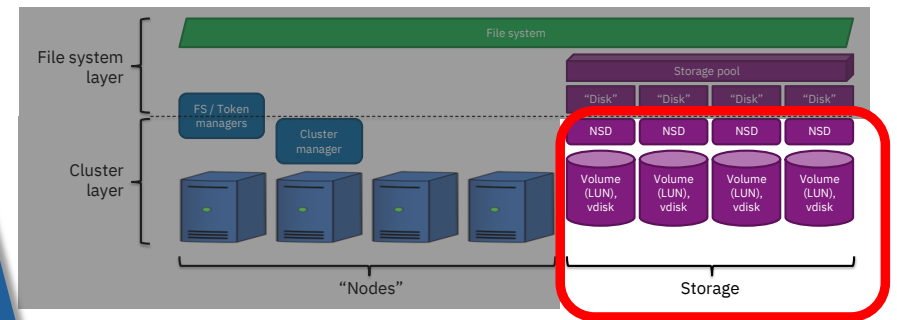
```
# mmlcluster
GPFS cluster information
=====
GPFS cluster name:      Plato.
GPFS cluster id:       8052019643637662953
GPFS UID domain:       Plato.
Remote shell command:  /usr/bin/ssh
Remote file copy command: /usr/bin/scp
Repository type:       CCR

Node  Daemon node name  IP address  Admin node name  Designation
-----
  1  plato01-dmz        192.168.62.30  plato01          quorum-manager-perfmon
  2  plato02-dmz        192.168.62.39  plato02          quorum-manager-perfmon
 12  newton05-dmz       192.168.62.44  newton05         perfmon
 15  newton04-dmz       192.168.62.43  newton04         perfmon
 16  gauss01-dmz        192.168.62.99  gauss01          perfmon
 17  gauss02-dmz        192.168.62.98  gauss02          perfmon
 19  plato08-dmz        192.168.62.42  plato08          perfmon
 22  gauss03-dmz        192.168.62.97  gauss03          perfmon
 23  gauss04-dmz        192.168.62.96  gauss04          perfmon
 24  plato04-dmz        192.168.62.37  plato04          perfmon
 25  plato05-dmz        192.168.62.36  plato05          perfmon
 26  plato03-dmz        192.168.62.38  plato03          quorum-manager-perfmon
 30  ashe               192.168.62.14  ashe
 31  billie             192.168.62.15  billie

# mmchnode --gateway --perfmon --snmp-agent -N ashe,billie
```

Architecture of Storage Scale

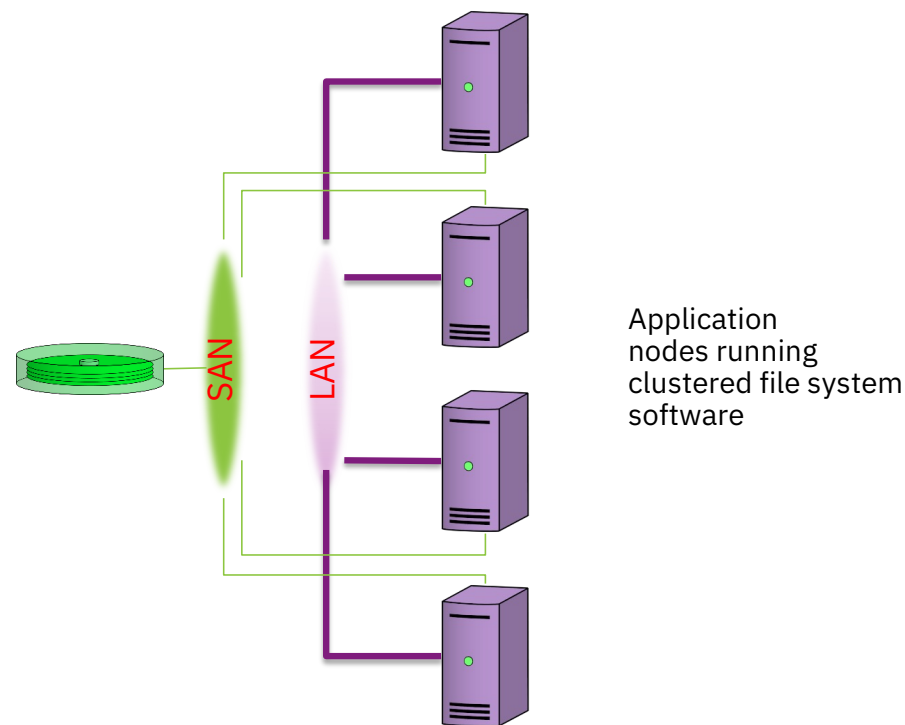
↳ **Clustering and storage**



Principle of *Clustered file systems*

A **clustered file system** may be mounted on multiple file system nodes simultaneously, sharing access to the same disk or volume through the **SAN**.

- Software implementing the clustered file system must be running on **each node**.
- A **logical volume manager** might allow multiple disk volumes to be federated into a single logical volume.
- This clustering software **coordinates access** to the actual (logical) disk volume to maintain disk integrity.
- This coordination is done **through messages on the LAN**.
- There are no “clients” and “servers” – **everything participates**.
- All nodes in the cluster are able to be actively accessing files (even multiple nodes accessing the same file).



Storage Scale: a parallel clustered file system

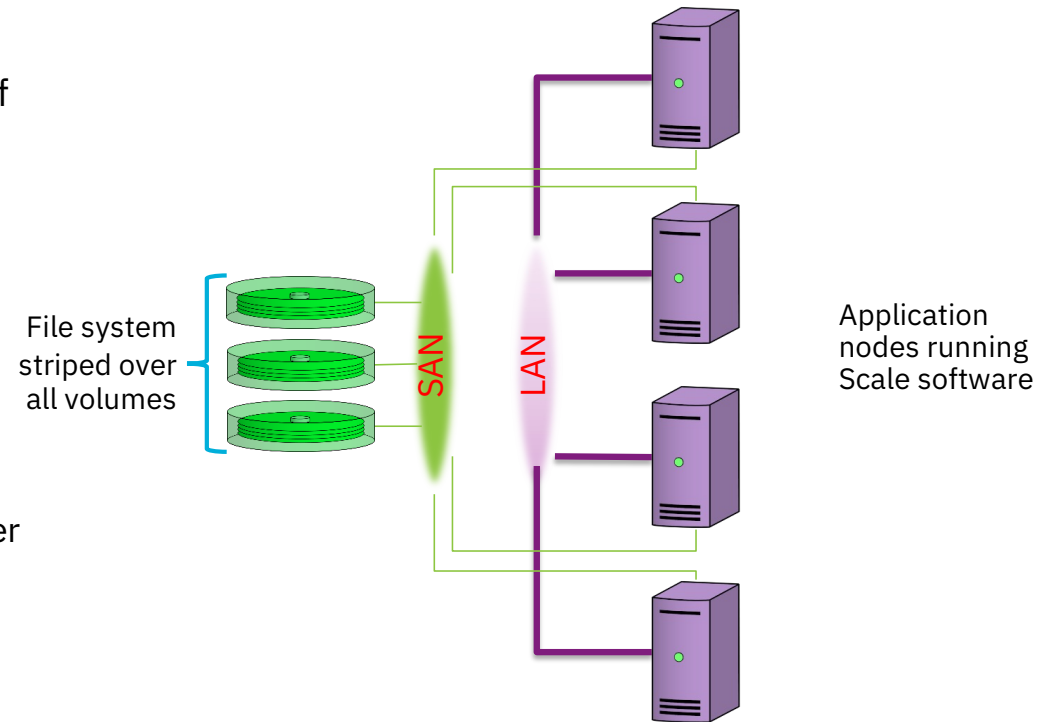
Scale is fundamentally a clustered file system:

- All nodes in the cluster share the responsibility of accessing the disk volumes to manage the file system and the data stored in it.

Scale is a **parallel** file system:

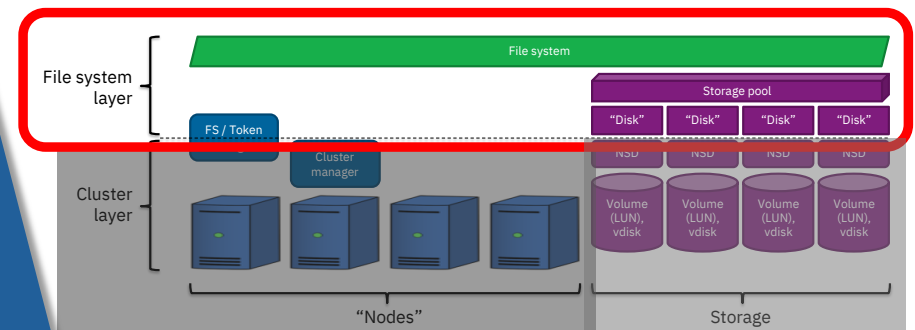
- The file system is striped over potentially many disk volumes.
- Even individual files (once large enough) are striped over multiple volumes.
- Uniquely, Scale even stripes file system metadata over many volumes.

All disk volumes controlled by a Scale cluster are called **NSDs**.

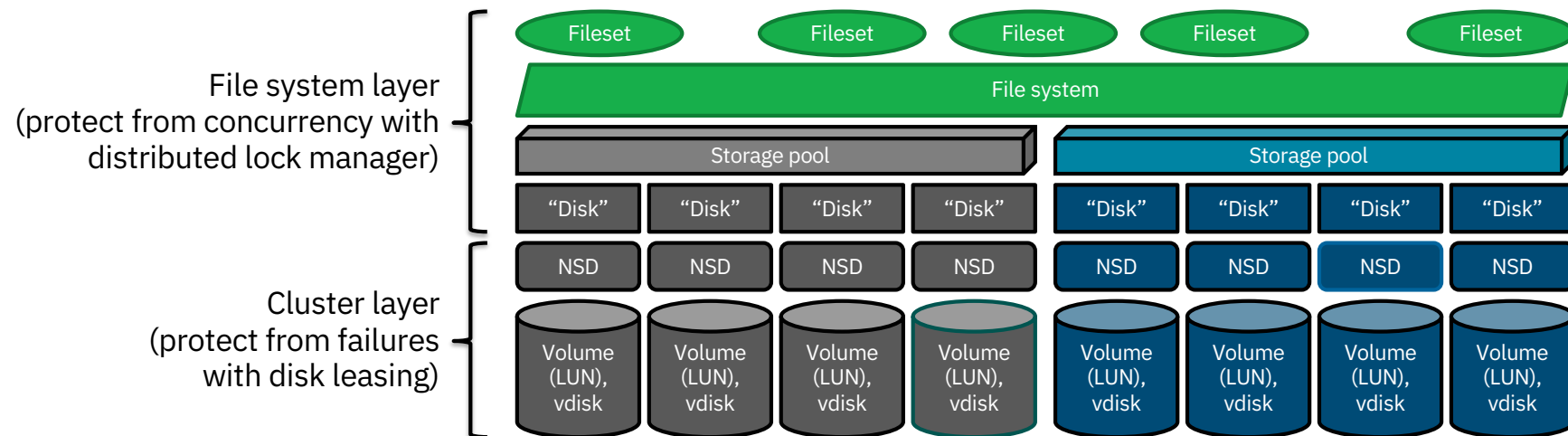


Architecture of Storage Scale

↳ **File system layer**



A logical overview of Storage Scale



Storage Scale – a POSIX file system

- POSIX file systems are organized as *directories* (folders) containing *files* (documents).
- POSIX defines programming APIs for manipulating files, such as:
 - mount, umount the file system
 - open, close file descriptor
 - write, read to/from file descriptor
 - mkdir, rmdir, creat, unlink, link, symlink
 - fcntl (byte range locks, etc.)
 - stat, utimes, chmod, chown, chgrp
- Path names are case sensitive; components are separated with forward slash (/).
- Essentially, this is the API that evolved for Unix and adopted by Linux.
- An Access Control Lists mechanism
- Filesets: partitioning a file system into pieces, without losing performance but gaining manageability.

```
$ cd /home/lindsay/Documents/Projects/Software
$ ls
SPEC_SCALE_STD_4.2_WINDOWS_X86.zip
Spectrum_Scale_Standard-4.2.0.1-x86_64-Windows-install.zip
$
```

File system manager

Several of the nodes in a cluster (frequently quorum nodes) are designated as **manager nodes**.

For each file system, one manager node is chosen by the cluster manager to be the **file system manager**.

The remaining manager nodes are additional **token managers** for that file system.

The file system manager is responsible for:

- Managing the internal file system configuration
 - Adding and removing disks
 - Changes to disk status
 - Repairs to the file system
- Managing disk space allocation
- Quota controls and management
- **Token management** (along with the other token managers)

```
# mmlsmgr fs1
file system      manager node      [from 10.0.200.11 (a-scale01)]
-----
fs1              10.0.200.11 (a-scale01)

# mmchmgr fs1 b-scale01
Sending migrate request to current manager node 10.0.200.11 (a-scale01).
Node 10.0.200.11 (a-scale01) resigned as manager for fs1.
Node 10.0.200.21 (b-scale01) appointed as manager for fs1.
```

To see the file system manager, use: `mmlsmgr FSDEV`

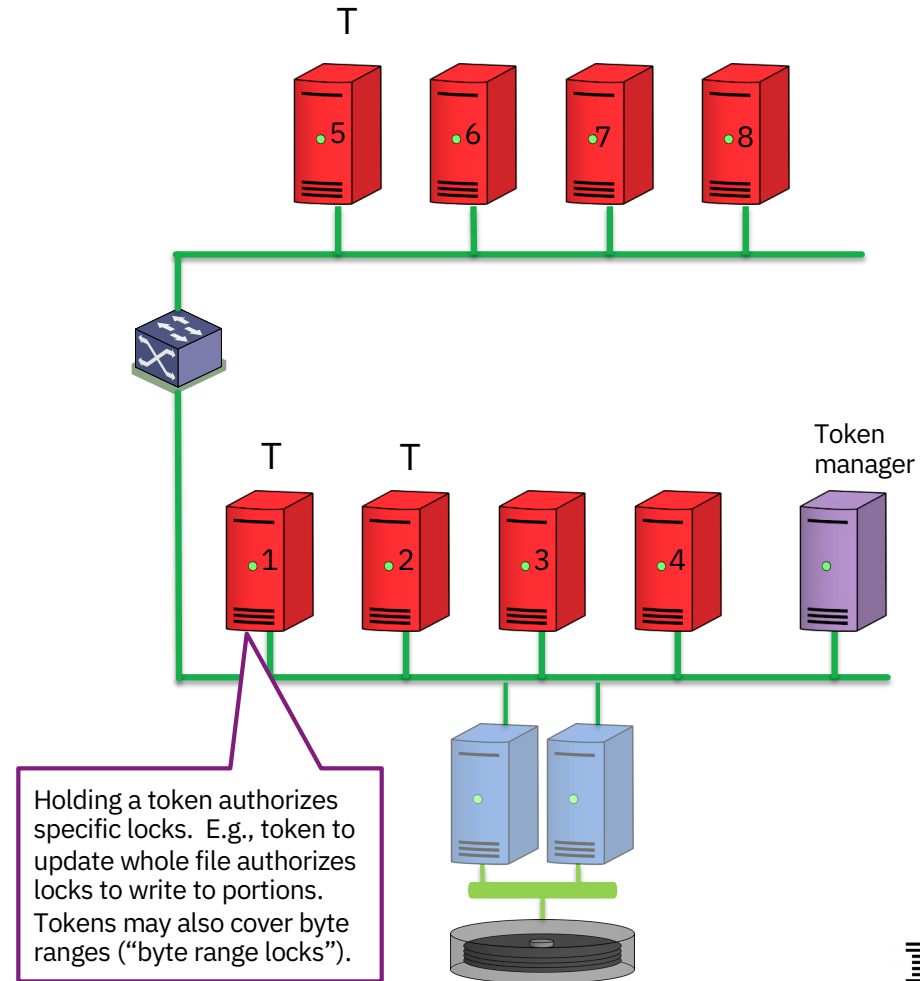
To move the file system manager, use: `mmchmgr FSDEV NODE`

Distributed lock management: Token manager roles

Using block-level byte-range locks, **tokens** authorize local caching: this allows parallel writes to distinct file areas, forcing serialization only for overlapping operations.

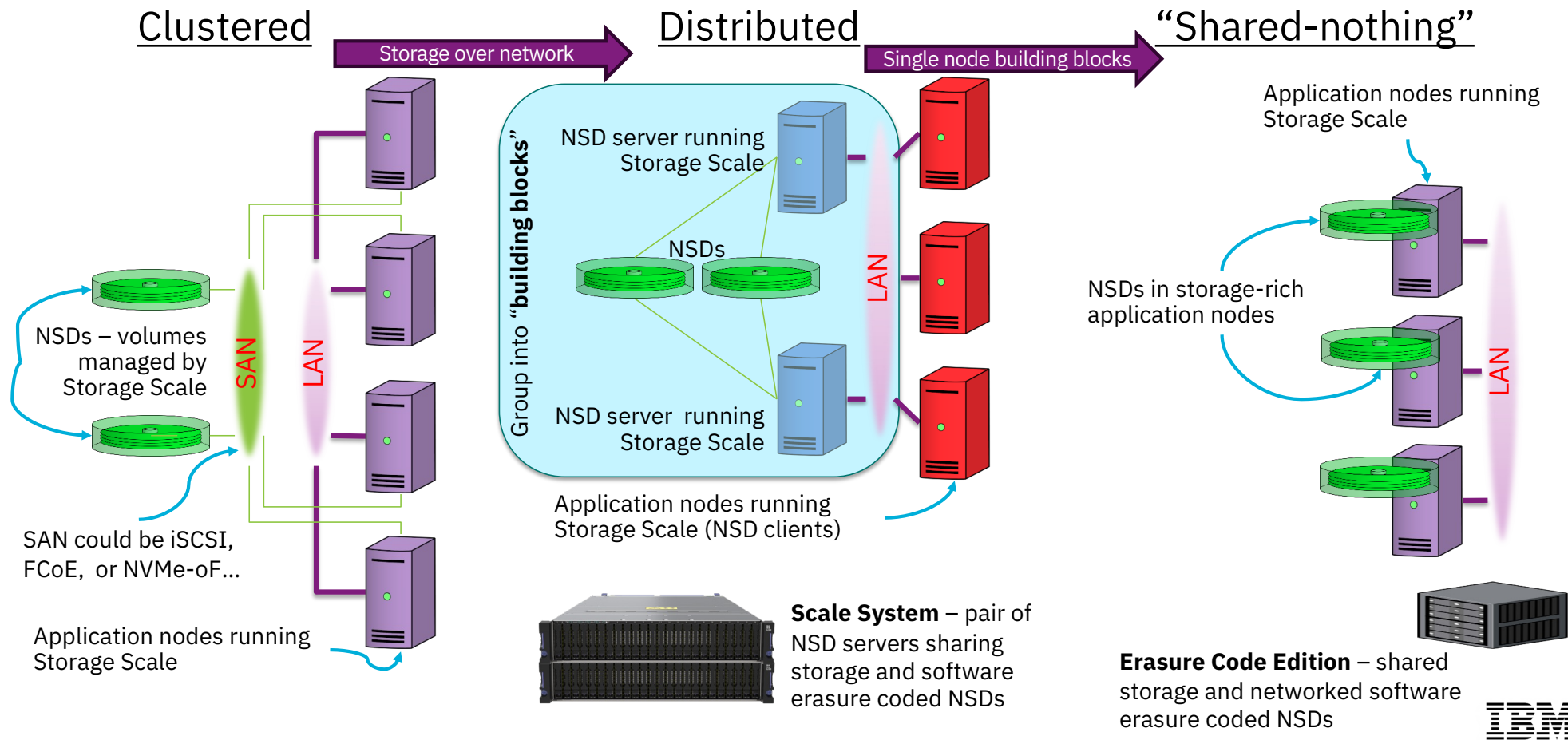
Nodes request tokens from the **token managers** (including the file system manager).

- If granted, a node may operate on the object until the token is revoked.
- If denied:
 1. The token manager tells the requesting node which other nodes have conflicting tokens.
 2. The node requesting the token communicates with those other nodes to revoke their tokens, collecting their responses.
 3. The requesting node updates the token manager with the current status.

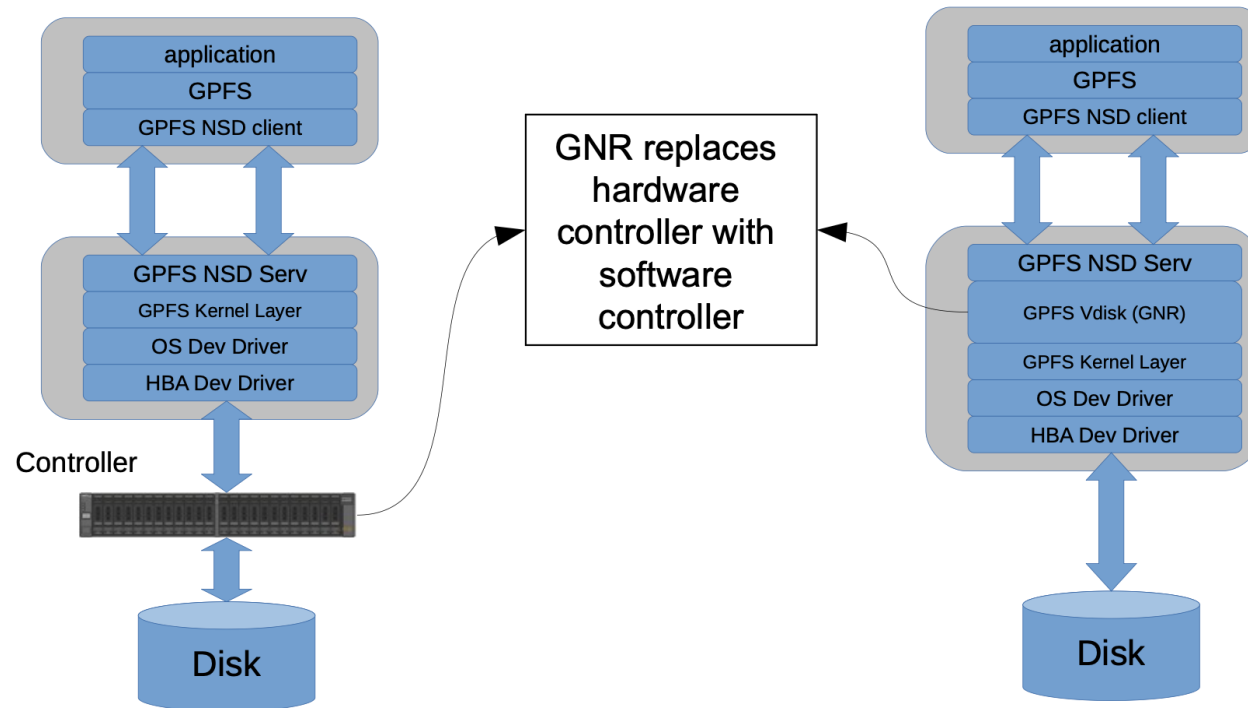


3- Storage Scale System- Architecture

High-level deployment strategies for NSDs



GPFS Native Raid (GNR): getting closer to the data

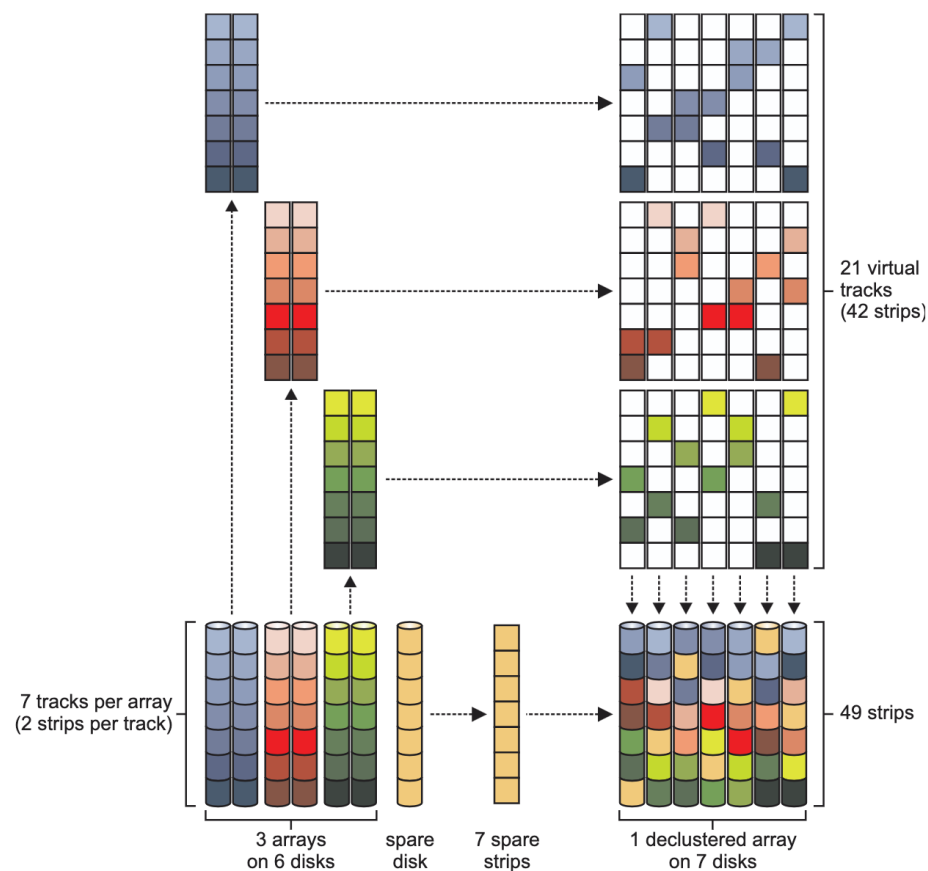


Declassified software RAID

IBM **Storage Scale RAID** is a *software* implementation of “declassified” or “**distributed RAID**”:

- Extremely fast rebuild after a disk failure, with minimal impact on performance
- Very strong data integrity checks
 - Additional erasure codes, such as 8+3p
 - Error detection codes enable detecting track errors and dropped writes
- Consistent performance from 0 – 99% utilization or 1 to many jobs in parallel
- All NSDs are wide-striped over many internal disks.

Storage Scale RAID is currently available only with Storage Scale System (IBM’s reference architecture) and Erasure Code Edition.



Declustered software RAID

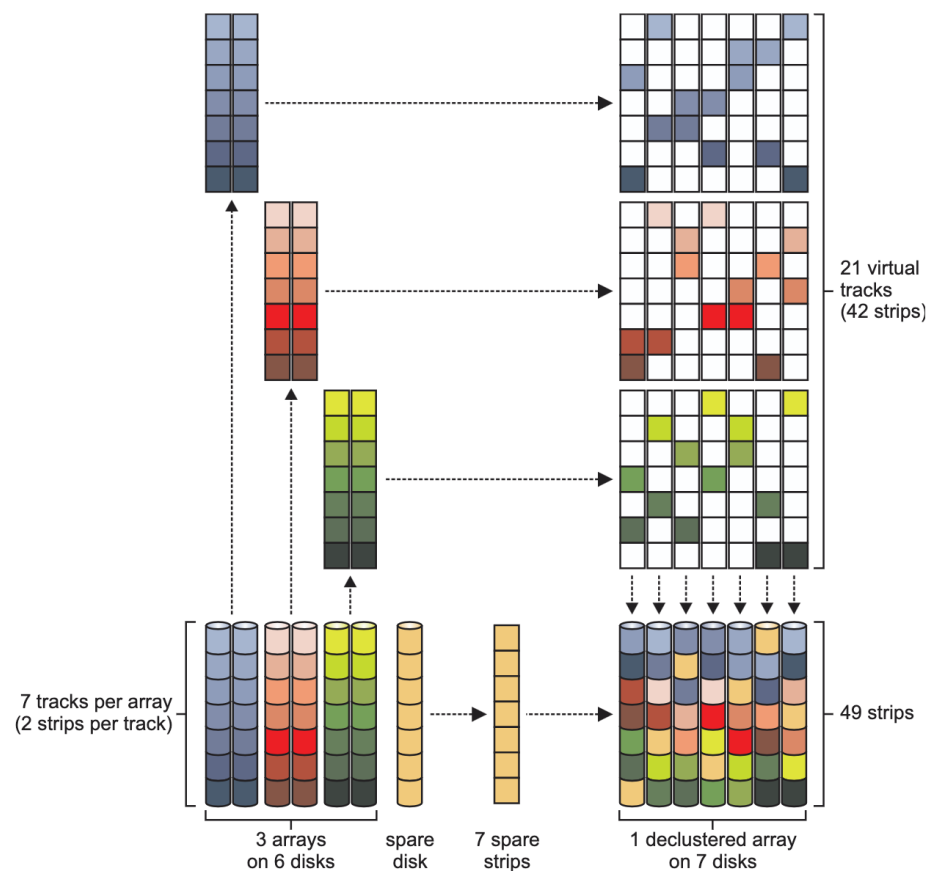
IBM Storage Scale RAID term to know:

Physical Disk (pdisk): a software abstraction of a single physical storage drive that belongs to exactly one declustered array.

Declustered Array: A declustered array is a disjoint subset of physical disks within a recovery group across which user data, redundancy information, and spare space are uniformly distributed to improve rebuild times.

Recovery Group: the fundamental organizing structure of GNR, acting as the internal software equivalent of a hardware disk controller to manage a specific set of physical disks.

Virtual Disk (vdisk): a type of Network Shared Disk (NSD) implemented across all physical disks of a declustered array, functioning as the equivalent of a logical unit number (LUN)



IBM Storage Scale System 6000

The IBM Scale System 6000 and ESS 3500 are the latest generation of integrated and tested IBM-provided **NSD server building block** solutions for Scale

- Fully validated IBM hardware and software stack
- Pre-assembled, pre-configured and installed
- Scale + Storage Scale RAID + Scale System GUI
- Scale System-aware performance, monitoring, installation, and upgrade
- Containerized Scale deployment based on Ansible playbooks, simplifying install and updates.

From sw release 7.0.0:

- Up to **47PB** per rack
- All-Flash Expansion Enclosure (JBoF)
- 3PB of raw flash capacity in a 2U Expansion Enclosure
- 30TB, 60, **122TB** SSD QLC



SSS6000



ESS500



4- Advanced functions

Storage Scale can be ... s t r e t c h e d ...

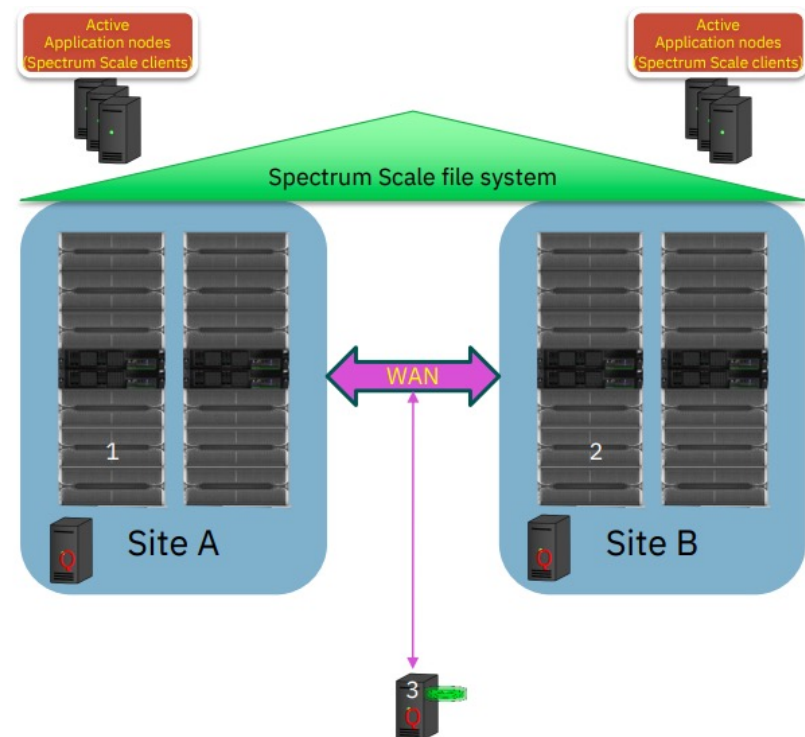
A *single* Scale cluster may be configured using nodes and storage from two data centers.

It is “***stretched***” between two sites, connected by a WAN.

The file systems use “**failure group**” replication to ensure both sites have a current replica of all the data.

- Careful design can ensure one site remains active, even if the other site (or the link between sites) fails.
- Both sites may **concurrently** access the file system for both reading and writing.

A stretched cluster provides an **active/active highly available** synchronously-replicated Scale file system across two data centers.



ILM Policies and storage pools

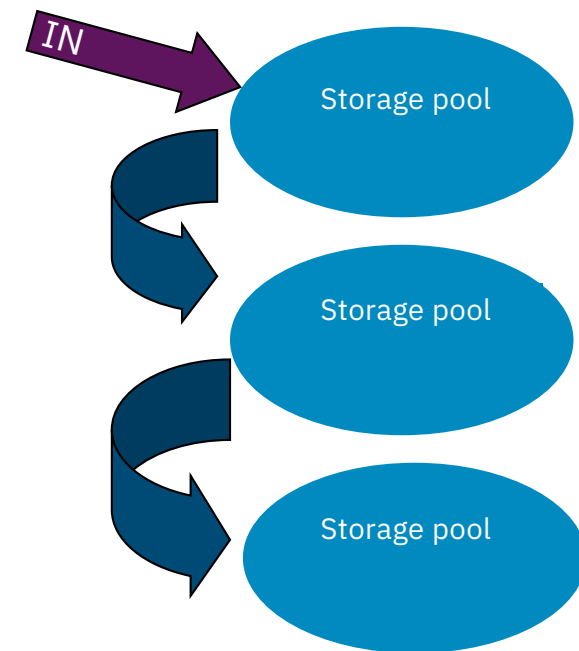
Policies for different purposes:

- **File placement policies** – Rules determining where and how file data is placed when a file is created
- **File management policies** – Rules for migration, deletion, encryption, etc. of files based on chosen criteria

Policies select files on which to operate from a diverse set of criteria, such as:

- File name, file type, path name, fileset, pool, name of the file, or snapshot
- Owner or group of the file
- Age, size, “file heat” of the file
- Extended attributes on the file
- file deletion, moving to “external pools”, triggering actions, encryption, compression.

The attribute must be known when the policy is run (so, e.g., a placement policy can’t know the ultimate length of a file, but it does know the owner)



Overview of Active File Management (AFM)

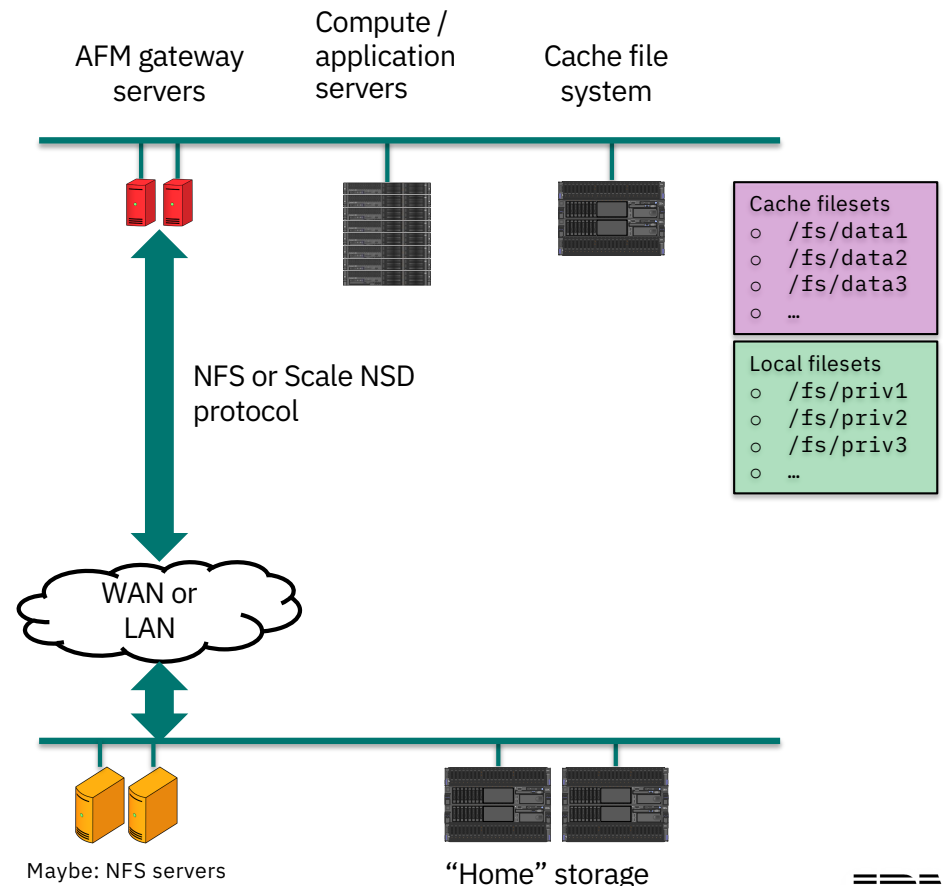
AFM **cache filesets** cache data from a target or **home**, using several modes:

- **Read-only** – A read-only cache of the home
- **Local update** – Read-only cache, but local changes allowed
- **Single writer** – Only the single cache; it alone may update home
- **Independent writers** – multiple caches may update same home

Gateway servers maintain the freshness of caches and send updates to the home. They queue updates even if the link to home is down.

A **home** is a directory tree being cached by AFM somewhere else. The home is unaware of the cache and need not even be in Scale.

The transport is either the native Scale NSD transport or NFS.



Cluster Export Services (CES)

Expose Storage Scale file system to different protocols

Supported Protocols:

- NFS
- SMB
- HDFS
- S3

A set of Storage Scale nodes are assigned CES service

CES IP:

- Shared between all CES protocols
- Auto failover
- Assigned virtual IP address

Address distribution policy:

- Even-coverage (default)
- Balanced-load (based on network and CPU)
- Node-affinity

```
# mmces service enable {NFS | SMB | HDFS | S3 }
```

```
# mmces address add --ces-ip Address[,Address...] --ces-node Node
```

```
# mmces address policy [even-coverage | balanced-load | node-affinity | none]
```

```
# mmces address move --rebalance
```

IBM Storage Scale – global data platform for unstructured data

