OPEN COMPOSABLE COMPATIBILITY LAB

Technical Brief

Implementing Intelligent Storage Tiering for AI and HPC Using IBM® Storage Scale and OpenFlex® Data24 4000 Series and Ultrastar® Data60

Problem Statement

AI and HPC environments are pushing storage in two directions at the same time. They need very high-performance for active work, and very large capacity for everything that accumulates around it. These include raw data, intermediate results, checkpoints, and long-term retention. As GPU clusters and compute nodes scale up, storage can become the bottleneck. Compute sits idle while data waits in line. This gap shows up as lower overall system efficiency and longer time to results, even when the compute side is fully provisioned.

At the same time, not all data is equally "hot." A small portion of data is accessed frequently and needs low latency and high IOPS, while a much larger portion is accessed occasionally and is better placed on cost-efficient capacity media. Without tiering, teams are forced into an inefficient choice. Either keep too much data on expensive performance storage, or accept performance drops when active workflows collide with slower tiers. The practical outcome is overspending, manual workarounds, or inconsistent performance across phases of the workflow.

In production deployments, moving data between tiers is also a problem. Manual copies and one-off scripts are common, but often brittle. They require constant attention, break when workflows change, and are hard to validate. The desired end state is simple to describe but hard to execute. Keep one shared namespace, place data on the right tier automatically, and move it as data "cools," without disrupting applications.

This solution addresses this by using a single file system across two pools (a high-performance NVMe™ tier and a capacity HDD tier), then applying clear placement and lifecycle policies (migration, compression, and cleanup) to steer data based on where it lives and how it is used.

This technical brief focuses on the specific challenge of building a tiered architecture that is both high-performance and operationally practical:

- A hot tier designed for active AI/HPC data using an NVMe-oF-based platform (OpenFlex Data24).
- A capacity tier designed for economical scale using dense HDD enclosures such as an Ultrastar Data60 (populated with SAS CMR drives).
- A policy-based approach that can be run on demand, triggered by capacity thresholds, or scheduled.

OpenFlex Data24 4000 Series Storage Platform

The OpenFlex Data24 4200 is a shared NVMe-oF storage platform built for workloads that need very high bandwidth and low latency. It takes NVMe SSD performance and makes it available to multiple servers over Ethernet. This lets teams treat flash as a pooled resource rather than something trapped inside individual compute nodes.

At a high level, the Data24 4000 series uses NVMe over Fabrics (NVMe-oF™) so hosts can access NVMe SSDs across the network with latency designed to mimic direct attached storage. The platform uses Western Digital RapidFlex™ A2000 NVMe-oF controllers, which provides up to 12 ports of 100Gb Ethernet and supports host connectivity using RDMA or TCP. It also supports a small, direct-attached design where up to six dual-pathed hosts can be connected without a switch.

The Data24 4200 is a 24 drive NVMe 2U chassis designed for high-density and parallel access. The platform includes dual I/O modules for parallel paths to storage. It supports multiple fabric connections, which deliver high aggregate bandwidth while keeping latency predictable. It also supports dual-ported, vendor-agnostic NVMe SSDs, which reinforces an open design and avoids lock-in to a single drive vendor.

IBM Storage Scale Overview

IBM Storage Scale is a high-performance software-defined parallel file system that provides shared access to large datasets across many servers. It is designed for AI, HPC, analytics, and other data-intensive workloads where many clients need to read and write at the same time. Data is presented through a single global POSIX-compliant namespace so applications can use standard file access methods while the system handles parallelism in the background.

A key concept in Storage Scale is that it can aggregate block devices into Network Shared Disks, also called NSDs. Storage Scale then distributes I/O across storage servers to increase concurrency and help maintain consistent performance under load. This design supports independent scaling of compute and storage. Teams can add storage capacity or storage bandwidth without rebuilding every compute node.

Storage Scale is also relevant in tiered designs because it can manage multiple storage pools under one file system. In this project, the environment was configured as one file system with two storage pools, a NVMe tier and an HDD tier. Storage Scale exposes this as one namespace while still allowing data to be placed on a specific pool based on configured policies. This sets up the policy model described later.

Ultrastar Data60 Overview

The Ultrastar Data60 is a high-density JBOD (Just a Bunch of Drives) platform used to add large amounts of HDD capacity in a small rack footprint. In this tiering architecture, it represents the capacity tier that complements the NVMe-based performance tier.

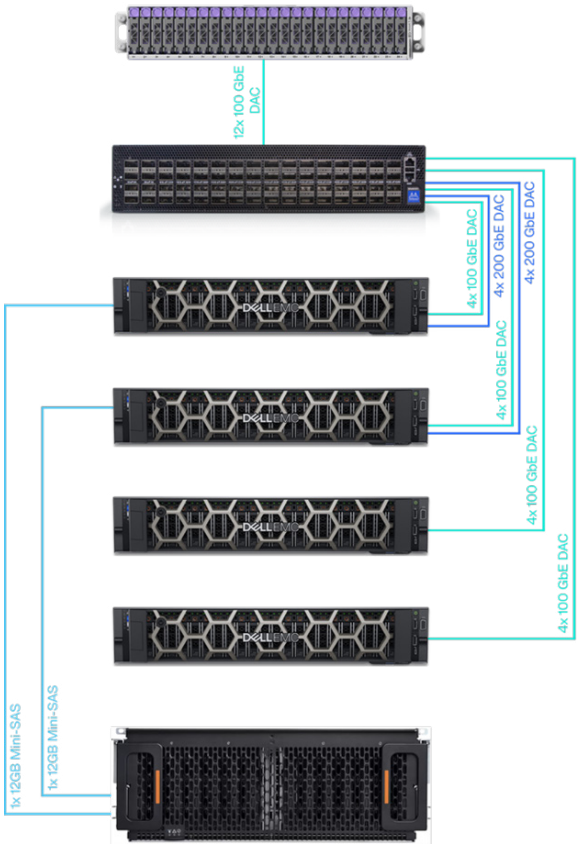
The Data60 is part of the Ultrastar JBOD family, with a 4U form factor that supports up to 60 HDDs. It is positioned for enterprise and big data storage use cases, including disaggregated storage for software-defined storage environments, large scale backup, and archive deployments. The platform supports both high availability connectivity and lower cost options, depending on how the enclosure is deployed and what the storage software expects.

AI and HPC workflows create a mix of data temperatures. Some data needs flash class performance, and much more data needs economical capacity for retention and reuse. Within a tiered design, the Data60 provides the warm and cold capacity layer. It is a good fit for datasets that must stay online and accessible, but do not need NVMe latency. This separation helps keep the NVMe tier focused on active work, while the HDD tier absorbs growth in project data, intermediate results, and long-lived retention datasets.

The Data60 is typically attached to storage servers using external SAS connectivity. Those storage servers then present the HDD capacity to the file system as a managed pool. This keeps the architecture modular. It also makes it easier to scale capacity by adding enclosures without changing the NVMe tier design.

Deployment Topology

Two Storage Scale storage servers provide the file system services. Each storage server connects to the shared NVMe tier over Ethernet and to the HDD enclosure over SAS. Two client servers access the file system over the client network. Management access is provided through the lab network. This minimum viable product design can be scaled up linearly with the demand for compute or storage.



Hardware

- Storage servers (Storage Scale)
  - 2x Dell® PowerEdge® R760
  - 256GiB memory per server
  - NICs: 2x NVIDIA® ConnectX®-6 and 2x NVIDIA ConnectX-7
  - SAS HBA: Broadcom 9600-16e
- Client servers
  - 2x Dell PowerEdge R750
  - 256GiB memory per server
  - NICs: 2x NVIDIA ConnectX-6
- NVMe tier (hot tier)
  - 1x OpenFlex Data24 4200
  - 24x 15.36TB[1] NVMe SSDs (Dapustor R5100D)
- HDD tier (capacity tier)
  - 1x Ultrastar Data60
  - 60x Ultrastar DC HC550 18TB HDDs (30 drives used for this configuration)
- Switching
  - 1x Mellanox® SN4600
  - 64 ports, up to 200GbE

Test Environment Overview

The test environment was configured as one Storage Scale file system built from 54 NSDs, split into two storage pools. Data could be placed on an NVMe tier (24 drives) or an HDD tier (30 drives) while remaining in a single namespace.

NSD and pool layout inside the file system:

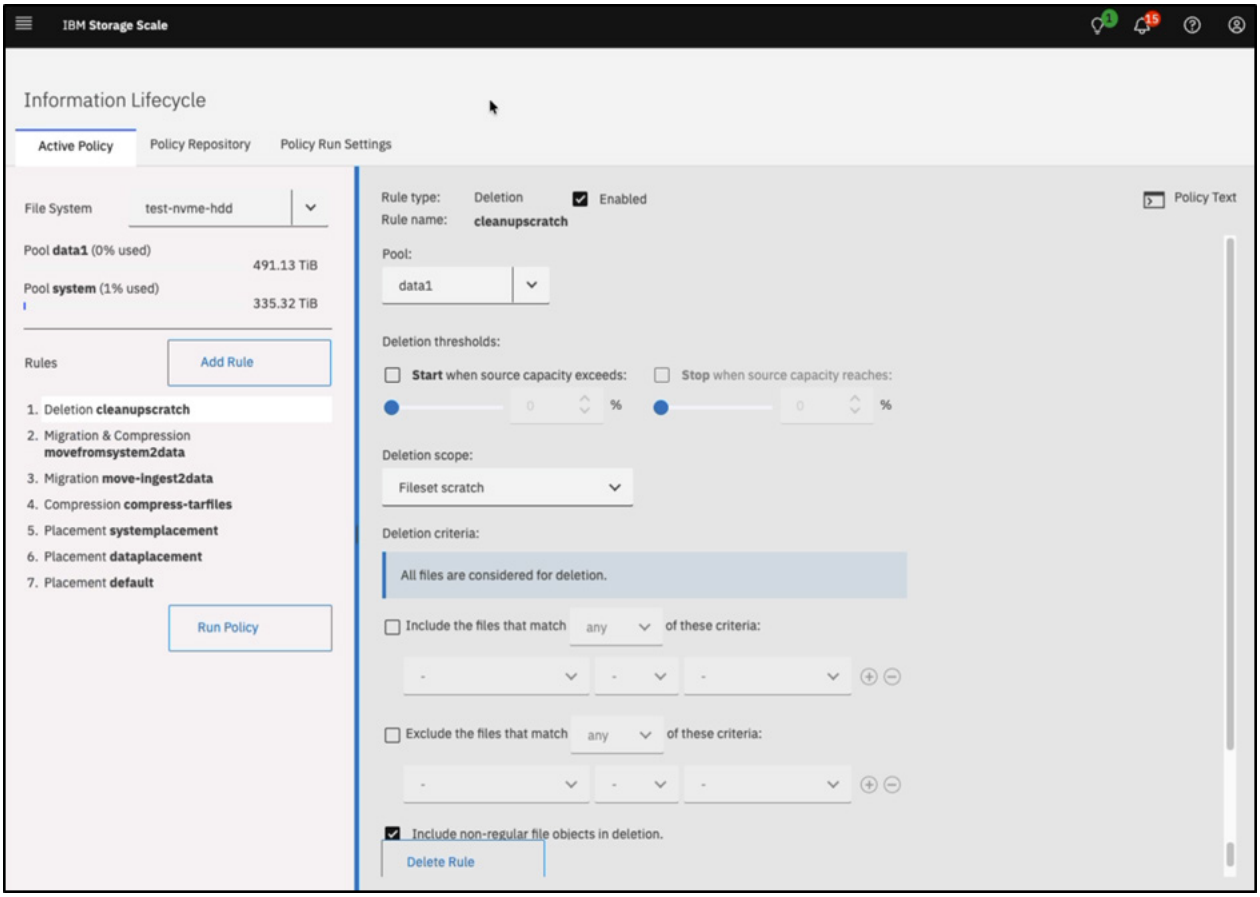
- The file system was created from 54 storage disks presented as NSDs.
- The NVMe tier was represented by NSD 1 through 24.
- The HDD tier was represented by a separate set of NSDs numbered 101 through 130. (separate numbering ranges were used to visually distinguish tiers)
- These NSDs were grouped into two storage pools, with the NVMe NSDs forming the “system” pool and the HDD NSDs forming the “data1” pool.

Filesets and default placement behavior:

- Sample filesets were created to represent common workflow areas, including Archive, Ingest, Projects, a default root fileset, and Scratch.
- Placement policies were configured so that data created in the Ingest and Scratch filesets landed on the NVMe pool, while data created in Archive, Projects, and the root fileset landed on the HDD pool.
- A catch-all default rule was also used so that data without a clear placement rule was placed on the HDD pool rather than consuming NVMe capacity.

[1] One terabyte (TB) is equal to one trillion bytes and one petabyte (PB) is equal to 1,000 TB. Actual user capacity may be less due to operating environment.

Tiering Overview



Storage Scale includes a policy-based framework that can automate data placement and lifecycle actions within a single file system. Policies were also created for lifecycle management tasks such as migration, compression, and cleanup actions.

In this environment, active workflow data is placed in the NVMe pool and then moved or compressed to the HDD pool as it cools over time. Policies can be configured through the GUI or edited directly in text form.

Operational Considerations

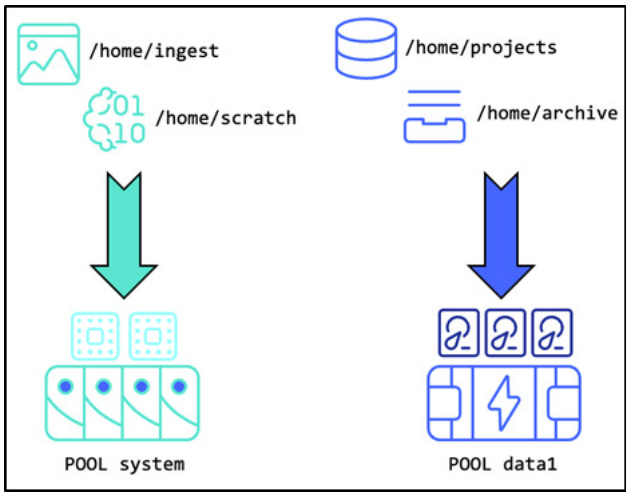
- Run policies safely by starting with a constrained scope (specific filesets or directories) and validating results before expanding. Policies can be executed and verified using both the GUI and command line.
- Rule ordering matters because policies are evaluated top down and the first matching rule is applied. Put the most specific rules first, then end with a catch-all to avoid unexpected placement or movement.
- Scheduling policies with cron is practical for operations today because it enables consistent, low-touch execution windows (for example, during off-hours) without requiring manual intervention.

Placement

Placement policies control where newly created files land by default. In this environment, data created in Archive, Projects, and the root fileset landed on the HDD pool, while data created in Ingest and Scratch landed on the NVMe pool. A catch-all rule ensured data without an explicit match defaulted to the HDD pool rather than consuming NVMe capacity.

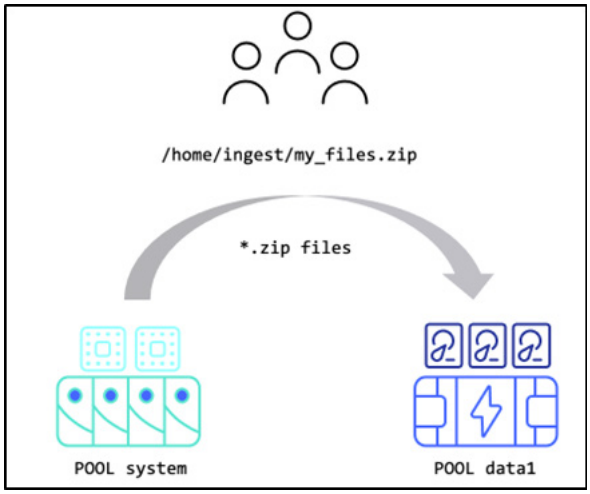
Policy Example (placement)

```
RULE 'systemplacement'
  SET POOL 'system'
    FOR FILESET ('ingest', 'newurgentproject', 'scratch')
RULE 'dataplacement'
  SET POOL 'data1'
    REPLICATE(2)
    FOR FILESET ('archive', 'projects', 'root')
RULE 'default'
  SET POOL 'data1'
```



Migration

Migration policies move existing data between pools based on defined criteria. Migration rules can target a specific fileset and then move data from the NVMe pool to the HDD pool when it is no longer considered hot, using conditions such as last-access time or file age.



Policy Example (migration)

```
RULE 'move-ingest2data'
  MIGRATE
    FROM POOL 'system'
  TO POOL 'data1'
    FOR FILESET ('ingest')
    WHERE
    ((LOWER(NAME) LIKE '%.zip') OR CURRENT_TIMESTAMP - MODIFICATION_TIME > INTERVAL '1' DAYS)
```

Compression

Compression policies are used to reduce capacity consumption for selected data on the HDD tier. Compression can be applied based on file characteristics and location, for example, targeting filesets intended for longer-term retention. Compression can be run independently or combined with migration so data can be moved to the capacity pool and compressed in the same policy run.

Policy Example (compression)

```
RULE 'compress-tarfiles'
  MIGRATE
    FROM POOL 'data1'
    WEIGHT(FILE_SIZE * (CURRENT_TIMESTAMP - ACCESS_TIME))
  COMPRESS('yes')
    FOR FILESET ('archive', 'newurgentproject', 'projects', 'root')
    WHERE
    ((LOWER(NAME) LIKE '%.tar')
```

Observations

In this lab environment, tiering actions (placement, migration, and compression) were straightforward to operationalize because they were policy-driven and repeatable, rather than script-based. Policies were run in controlled scopes (specific filesets) and could be scheduled during maintenance windows, which helps minimize disruption to active workflows. Rule ordering and a default catch-all behavior reduced the risk of unexpected placement or movement, making day-to-day operations predictable.

Conclusion

This project demonstrated a practical way to combine a shared NVMe tier and a high-capacity HDD tier under one IBM Storage Scale file system, while keeping a single namespace for users and applications. The environment used two storage pools and policy-based tiering so data could land on the right tier at creation time, then move later as it aged or requirements changed.

A key result is that tiering did not require application changes. It was implemented through Storage Scale constructs that are familiar to operations teams, including pools, NSDs, filesets, and policies. Filesets such as Archive, Ingest, Projects, and Scratch were used to represent workflow areas, and placement rules were used to steer new data to either the NVMe pool or the HDD pool.

The work also showed how lifecycle actions can be applied as policies, not as one-off scripts. Migration and compression were demonstrated as repeatable actions that can be run from the GUI, run from the command line, triggered by capacity thresholds, or scheduled using cron.

This technical brief builds on the Western Digital OpenFlex Data24-4200 and IBM Storage Scale High-Performance, Disaggregated File Storage for AI and HPC by focusing on the tiering behavior. It captures how the pools were laid out, how filesets mapped to tiers, and how policies were used to move and optimize data over time in a controlled and testable way.

Key Takeaways

- A two-tier design can be presented as one file system, so users work in one place while the platform handles placement and movement.
- Filesets plus placement rules provide a simple operational model for keeping active work on NVMe and longer-lived data on HDD.
- Migration and compression policies can be executed in several practical ways, including manual runs, capacity threshold triggers, and cron scheduling.

5601 Great Oaks Parkway
San Jose, CA 95119, USA
www.westerndigital.com