Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

NHR-Nord@Göttingen

Holistic HPC I/O

Storage Architectures and Storage Tiering

Mohammad Hossein Biniaz, Kevin Lüdemann

12.06.2024

NHR Summer School 2024

Table of contents

Learning Objectives

- Understanding the storage stack
- Investigate IO access patterns
- Understand storage tiering
- Devellop and IO Workflows
- Learn about parallel file systems

High-Performance Computing (HPC)

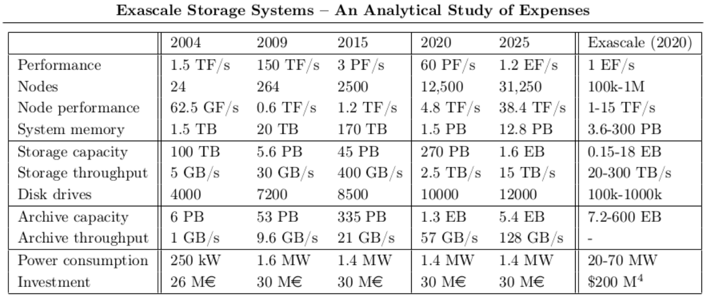
Definitions

- HPC: Field providing massive compute resources for a computational task
  - Task needs too much memory or time on a normal computer
  - ⇒ Enabler of complex scientific simulations, e.g., weather, astronomy
- Supercomputer: aggregates power of 10,000 compute devices
- Storage system: provides some kind of storage with some API
- File system: provides a hierarchical namespace and “file” interface
- Parallel I/O: multiple processes can access distributed data concurrently

Supercomputers Host Costly Storage

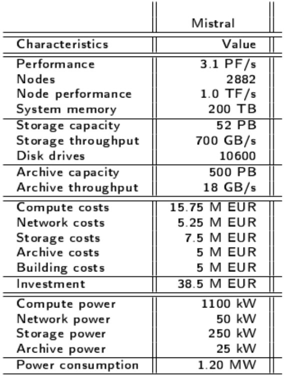
- Compute performance growth by 20x each generation (∼5 years).
- Storage throughput/capacity improves by just 6x.

Real Values – 2018



Exascale Storage Systems – An Analytical Study of Expenses

	2004	2009	2015	2020	2025	Exascale (2020)
Performance	1.5 TF/s	150 TF/s	3 PF/s	60 PF/s	1.2 EF/s	1 EF/s
Nodes	24	264	2500	12,500	31,250	100k-1M
Node performance	62.5 GF/s	0.6 TF/s	1.2 TF/s	4.8 TF/s	38.4 TF/s	1-15 TF/s
System memory	1.5 TB	20 TB	170 TB	1.5 PB	12.8 PB	3.6-300 PB
Storage capacity	100 TB	5.6 PB	45 PB	270 PB	1.6 EB	0.15-18 EB
Storage throughput	5 GB/s	30 GB/s	400 GB/s	2.5 TB/s	15 TB/s	20-300 TB/s
Disk drives	4000	7200	8500	10000	12000	100k-1000k
Archive capacity	6 PB	53 PB	335 PB	1.3 EB	5.4 EB	7.2-600 EB
Archive throughput	1 GB/s	9.6 GB/s	21 GB/s	57 GB/s	128 GB/s	-
Power consumption	250 kW	1.6 MW	1.4 MW	1.4 MW	1.4 MW	20-70 MW
Investment	26 M€	30 M€	30 M€	30 M€	30 M€	$200 M^4



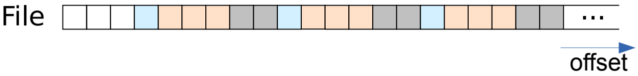
	Mistral
Characteristics	Value
Performance	3.1 PF/s
Nodes	2882
Node performance	1.0 TF/s
System memory	200 TB
Storage capacity	52 PB
Storage throughput	700 GB/s
Disk drives	10600
Archive capacity	500 PB
Archive throughput	18 GB/s
Compute costs	15.75 M EUR
Network costs	5.25 M EUR
Storage costs	7.5 M EUR
Archive costs	5 M EUR
Building costs	5 M EUR
Investment	38.5 M EUR
Compute power	1100 kW
Network power	50 kW
Storage power	250 kW
Archive power	25 kW
Power consumption	1.20 MW

Application Data vs. File

Applications work with (semi)structured data

- Vectors, matrices, n-Dimensional data

A file is just a sequence of bytes!

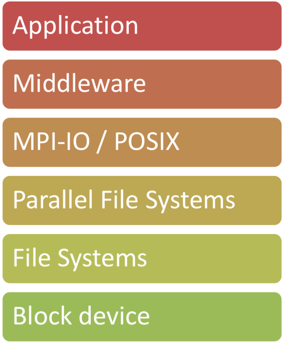


Applications/Programmers must serialize data into a flat namespace

- Uneasy handling of complex data types
- Mapping is performance-critical
- Vertical data access unpractical (e.g., to to pick a slice of multiple files)

The I/O Stack

- Parallel application
  - Is distributed across many nodes
  - Has a specific access pattern for I/O
  - May use several interfaces
    File (POSIX, ADIOS, HDF5), SQL, NoSQL
- Middleware provides high-level access
- POSIX: ultimately file system access
- Parallel file system: Lustre, GPFS, PVFS2
- File system: EXT4, XFS, NTFS
- Block device: utilizes storage media to export a block API
- Operating system: (orthogonal aspect)



Application

Middleware

MPI-IO / POSIX

Parallel File Systems

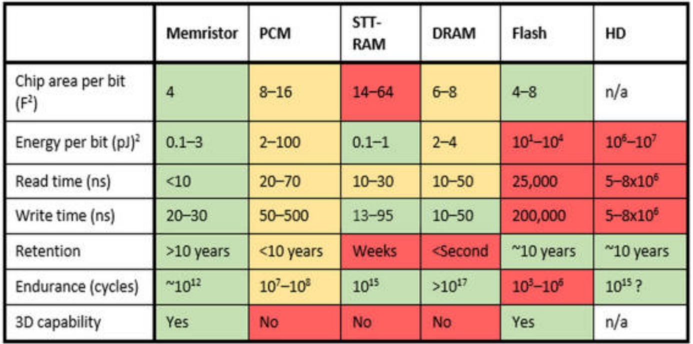
File Systems

Block device

Figure: Example I/O stack

Storage Media

- Many technologies are available with different characteristics
- Block-accessible or byte-addressable (NVRAM)



	Memristor	**PCM**	**STT-RAM**	**DRAM**	**Flash**	**HD**
Chip area per bit (F^2)	4	8–16	14–64	6–8	4–8	n/a
Energy per bit (pJ)^2	0.1–3	2–100	0.1–1	2–4	10^1–10^4	10^6–10^7
Read time (ns)	<10	20–70	10–30	10–50	25,000	5–8x10^6
Write time (ns)	20–30	50–500	13–95	10–50	200,000	5–8x10^6
Retention	>10 years	<10 years	Weeks	<Second	~10 years	~10 years
Endurance (cycles)	~10^{12}	10^7–10^8	10^{15}	>10^{17}	10^3–10^6	10^{15} ?
3D capability	Yes	No	No	No	Yes	n/a

Figure: Source: ZDNet [100]

Zoo of Interfaces

Multitude of data models

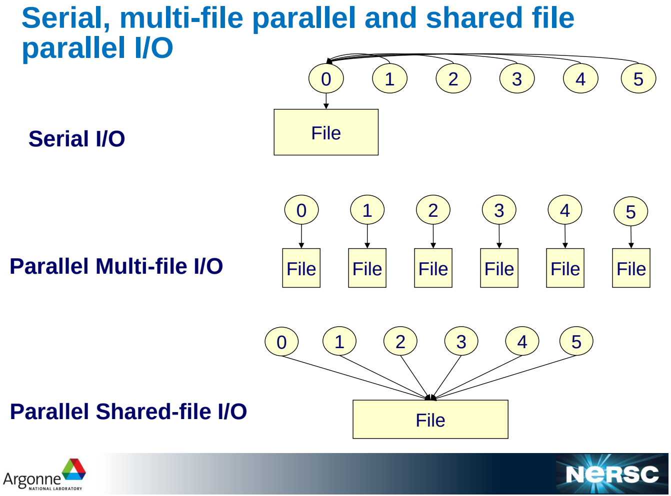
- POSIX File: Array of bytes
- HDF5: Container like a file system
  - Dataset: N-D array of a (derived) datatype
  - Rich metadata, different APIs (tables)
- Database: structured (+arrays)
- NoSQL: document, key-value, graph, tuple

Choosing the right interface is difficult – a workflow may involve several

Properties / qualities

- Namespace: Hierarchical, flat, relational
- Access: Imperative, declarative, implicit (`mmap()`)
- Concurrency: Blocking vs. non-blocking
- Consistency semantics: Visibility and durability of modifications

Application I/O Types



Serial, multi-file parallel and shared file parallel I/O

Serial I/O

Parallel Multi-file I/O

Parallel Shared-file I/O

Figure: Source: Lonnie Crosby [101]

Application I/O Access Patterns

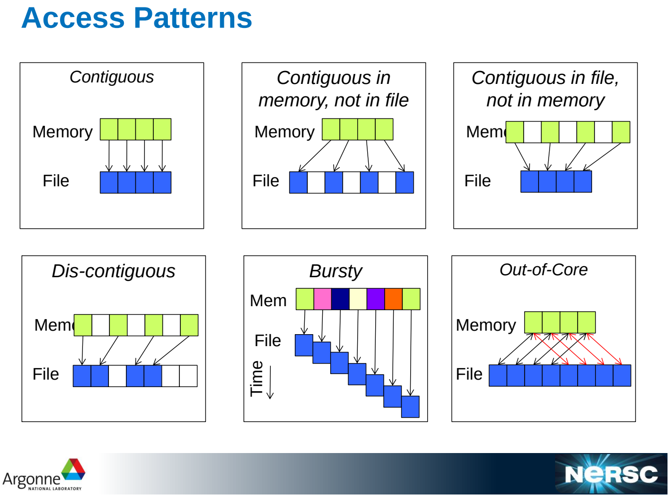
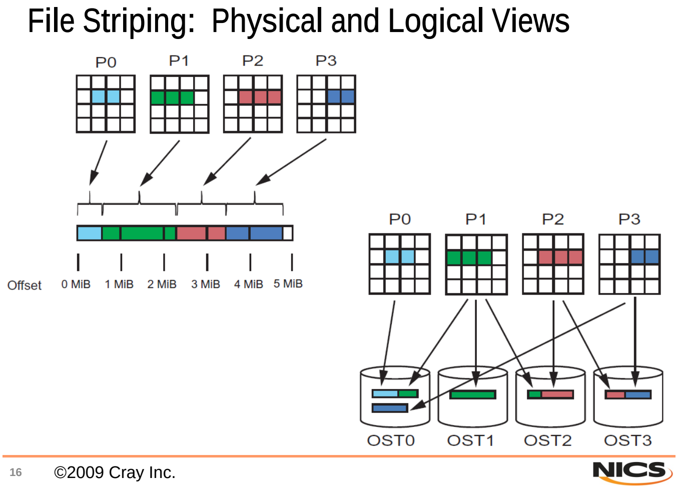


Figure: Source: Lonnie Crosby [101]

File Striping: Distributing Data Across Devices



File Striping: Physical and Logical Views

©2009 Cray Inc.

Figure: Source: Lonnie Crosby [101]

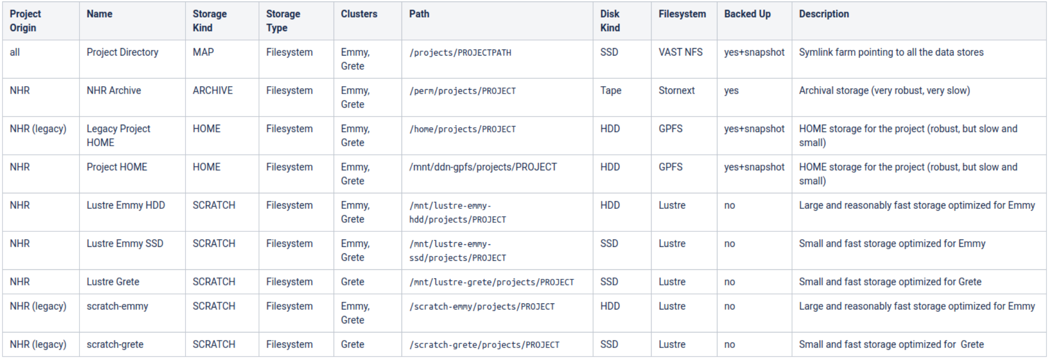
Parallel I/O Efficiency

- I/O intense science requires good I/O performance
- DKRZ file systems offer about 700 GiB/s throughput
  - I/O operations are typically inefficient: Achieving 10% of peak is good
  - Unfortunately, prediction of performance is barely possible
- Influences on I/O performance
  - Application's access pattern and usage of storage interfaces
  - Communication and slow storage media
  - Concurrent activity – shared nature of I/O
  - Tenable optimizations deal with characteristics of storage media
  - Complex interactions of these factors
- The I/O hardware/software stack is very complex – even for experts
- Requires tools and methods for
  - diagnosing causes
  - predicting performance, identification of slow performance
  - prescribing tunables/settings

Why Storage Tiering?

- Users have different requirements depending on the type of data
  - Think of Software as compared to hot data
- The different storage systems differ in many attributes, e.g.
  - Size
  - Speed
  - Data Durability
  - Backups
  - **Locality**
  - Lifetime, e.g. only available during job runtime, certain TTL, etc.

An Example at GWDG



Project Origin	Name	Storage Kind	Storage Type	Clusters	Path	Disk Kind	Filesystem	Backed Up	Description
all	Project Directory	MAP	Filesystem	Emmy, Grete	`/projects/PROJECTPATH`	SSD	VAST NFS	yes+snapshot	Symlink farm pointing to all the data stores
NHR	NHR Archive	ARCHIVE	Filesystem	Emmy, Grete	`/perm/projects/PROJECT`	Tape	Stornext	yes	Archival storage (very robust, very slow)
NHR (legacy)	Legacy Project HOME	HOME	Filesystem	Emmy, Grete	`/home/projects/PROJECT`	HDD	GPFS	yes+snapshot	HOME storage for the project (robust, but slow and small)
NHR	Project HOME	HOME	Filesystem	Emmy, Grete	/mnt/ddn-gpfs/projects/PROJECT	HDD	GPFS	yes+snapshot	HOME storage for the project (robust, but slow and small)
NHR	Lustre Emmy HDD	SCRATCH	Filesystem	Emmy, Grete	`/mnt/lustre-emmy-hdd/projects/PROJECT`	HDD	Lustre	no	Large and reasonably fast storage optimized for Emmy
NHR	Lustre Emmy SSD	SCRATCH	Filesystem	Emmy, Grete	`/mnt/lustre-emmy-ssd/projects/PROJECT`	SSD	Lustre	no	Small and fast storage optimized for Emmy
NHR	Lustre Grete	SCRATCH	Filesystem	Grete	`/mnt/lustre-grete/projects/PROJECT`	SSD	Lustre	no	Small and fast storage optimized for Grete
NHR (legacy)	scratch-emmy	SCRATCH	Filesystem	Emmy, Grete	`/scratch-emmy/projects/PROJECT`	HDD	Lustre	no	Large and reasonably fast storage optimized for Emmy
NHR (legacy)	scratch-grete	SCRATCH	Filesystem	Grete	`/scratch-grete/projects/PROJECT`	SSD	Lustre	no	Small and fast storage optimized for Grete

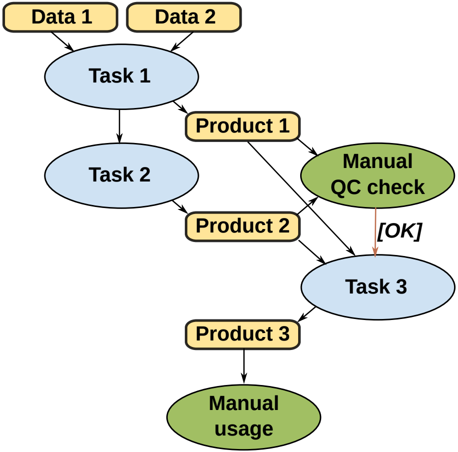
- Already 9 storge tiers, and the local `tmpfs` and SSD’s are even neglected

Resulting Problem

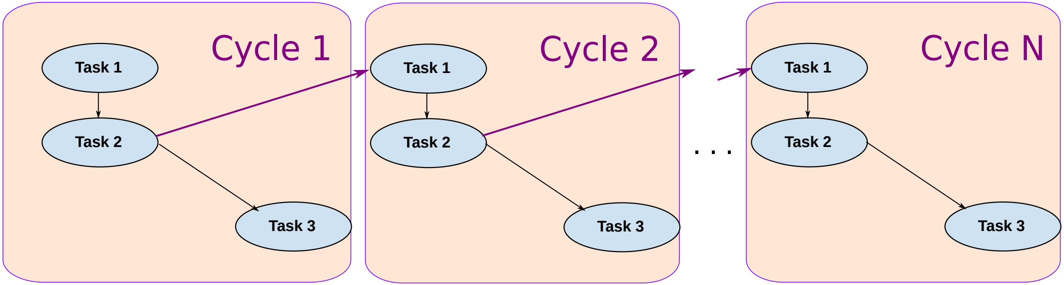
- People are overwhelmed and do wrong data placement
  - Hot data sits on cold storage
  - Standard datasets sit on expensive backed up storages
  - Important results are on fragile storage
  - The wrong storage system for the wrong cluster island is used
    - GWDG might be an edge case here, but also think of a Dragonfly Topology
- Many storage tiers quickly lead to a loss of oversight
  - Data is not cleaned up
  - Data is not reproducible, unclear where it belongs to
  - Data loss, hard to find
  - How can I select all data with a certain property?

Workflows

- Insight: What is of interest
- Consider workflow from 0 to insight
  - Needs input
  - Produces output data
  - Uses tasks
    - Parallel applications
    - Big data tools
    - Manual analysis / quality control
  - May need month to complete
  - Manual tasks are unpredictable

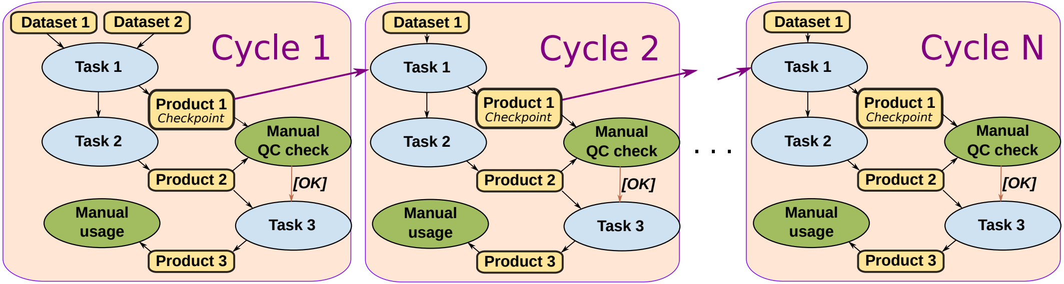


A (Science) Workflow Description



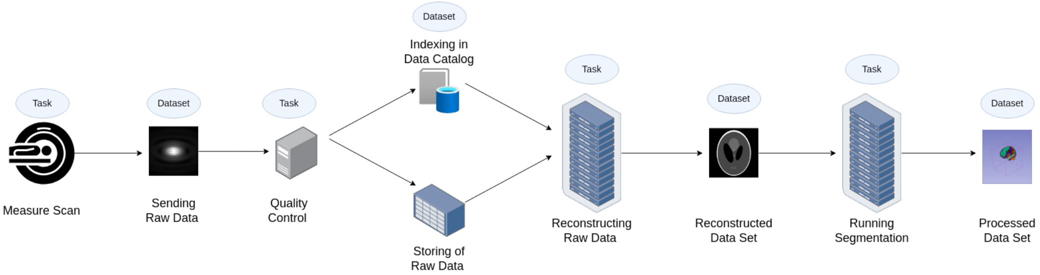
- Current practice (in climate/weather)
- Dependencies between tasks are described
- Assume a calculation that repeats for multiple cycles/iterations

Possible Extended (Science) Workflow Description



- Workflow description with IO characteristics
  - Input required
  - Needed input
  - Generated output and its characteristics
  - Information Lifecycle (data life)
  - ⇒ Explicit input/output definition (dependencies) instead of implicit

Experimental Planning Example: MRI Workflow Overview



Examples for HPC filesystems

- Opensource/free file systems
    - Lustre
    - BeeGFS (Enterprise is offered)
    - DAOS
- Exterprise file systems
    - IBM Spectrum Scale
    - Ceph
    - VAST Data
    - Weka.io

Lustre

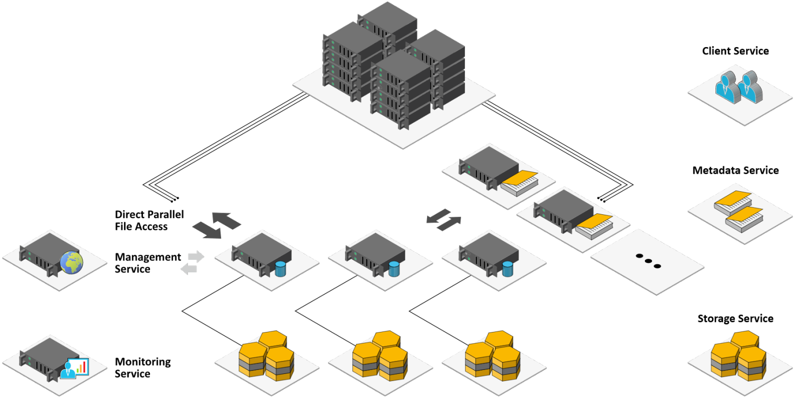
- Oldest parallel filesystem for HPC (first concepts in 1999 at CMU)
- Open Source
- Separate servers and targets for meta- and objectdata
- Every server has multiple storage targets
- Every client talks with every server in parallel
- Striping across targets/servers on client side
- User configure striping per directory/file (count, blocksize, storage pools)
- Integration in MPI-IO, HDF5, NetCDF, etc.
- Very low CPU requirements
- Backendstorage is patched Ext4 (ldiskfs) or ZFS

BeeGFS

- Developed as FhGFS at Fraunhofer ITWM (from 2005), spin-off ThinkParQ
- Open Source, but closed development
- Lustre IO performance with better metadata performance and easier usage
- Server concept like Lustre, multiple metadata server from the start
- Server processes in userspace
- Client out of tree kernel module,
- Very easy cconfiguration (text with a few lines of code per service)
- Every Linux filesystem can serve as backendstorage (recommendations for XFS, Ext4 and ZFS)

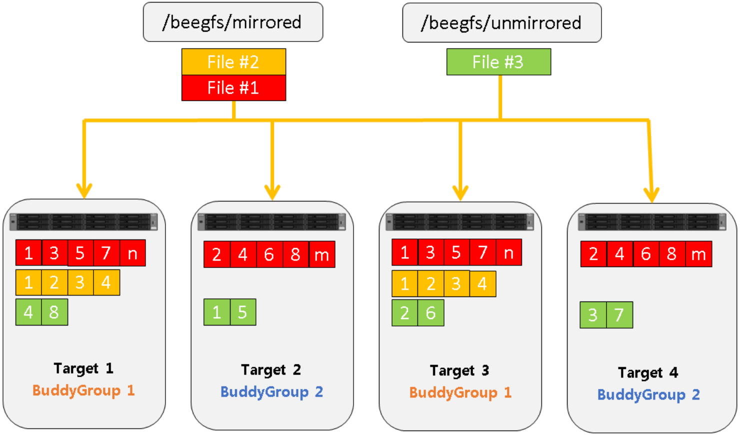
BeeGFS architecture

Source: BeeGFS documentation
`https://doc.beegfs.io/latest/architecture/overview.html`



BeeGFS filestriping

Source: BeeGFS documentation `https://doc.beegfs.io/latest/architecture/overview.html`



Performance comparison storage systems of HLRN-IV Emmy

IME: 10 DDN IME 140 systems with 90 4TB NVME SSDs

Lustre SSD: 2 DDN SFA200NV with 4 frontend servers, 46 4TB NVME SSDs

Lustre HDD: 2 DDN ES14KX, 1000 12TB HDDs

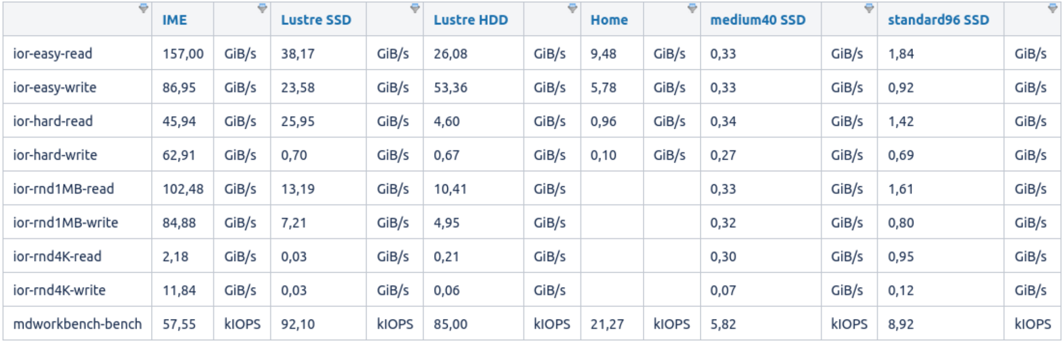
Home: GPFS via NFS from DDN GS7700X, 2 frontend servers, 120 4TB HDDs

medium40 SSD: local 480GB Intel S-ATA SSD in compute node

standard96 SSD: local 1TB Intel NVME SSD in compute node

Benchmarks with 32 processes per client, IME and Lustre with 64 clients, Home with 10 clients and 16 processes per client

Performance comparison storage systems of HLRN-IV Emmy



	IME		Lustre SSD		Lustre HDD		Home		medium40 SSD		standard96 SSD	
ior-easy-read	157,00	GiB/s	38,17	GiB/s	26,08	GiB/s	9,48	GiB/s	0,33	GiB/s	1,84	GiB/s
ior-easy-write	86,95	GiB/s	23,58	GiB/s	53,36	GiB/s	5,78	GiB/s	0,33	GiB/s	0,92	GiB/s
ior-hard-read	45,94	GiB/s	25,95	GiB/s	4,60	GiB/s	0,96	GiB/s	0,34	GiB/s	1,42	GiB/s
ior-hard-write	62,91	GiB/s	0,70	GiB/s	0,67	GiB/s	0,10	GiB/s	0,27	GiB/s	0,69	GiB/s
ior-rnd1MB-read	102,48	GiB/s	13,19	GiB/s	10,41	GiB/s			0,33	GiB/s	1,61	GiB/s
ior-rnd1MB-write	84,88	GiB/s	7,21	GiB/s	4,95	GiB/s			0,32	GiB/s	0,80	GiB/s
ior-rnd4K-read	2,18	GiB/s	0,03	GiB/s	0,21	GiB/s			0,30	GiB/s	0,95	GiB/s
ior-rnd4K-write	11,84	GiB/s	0,03	GiB/s	0,06	GiB/s			0,07	GiB/s	0,12	GiB/s
mdworkbench-bench	57,55	kIOPS	92,10	kIOPS	85,00	kIOPS	21,27	kIOPS	5,82	kIOPS	8,92	kIOPS

Storage stack

Perspective	Layer	Examples
User Perspective	**Application**	CODA, Python
Developer Perspective	**Middleware**	MPI-IO, (p)HDF5, Dask, Spark
	System-Level	POSIX, io_uring, VFS
System Perspective	**Storage-Level**	BeeGFS, Lustre, (p)NFS, DAOS, ext4
	Physical Medium	HDD, SSD, NVME, NVDIMM

Summary

- Understanding the full storage stack is difficult even for experts
- Accessing files in parallel requires specific applications
- Workflows required a lot of I/O
- Taking care of Storage Tiering is important
- Setting up and understanding parallel I/O is difficult