PARALLEL I/O AND PORTABLE DATA FORMATS

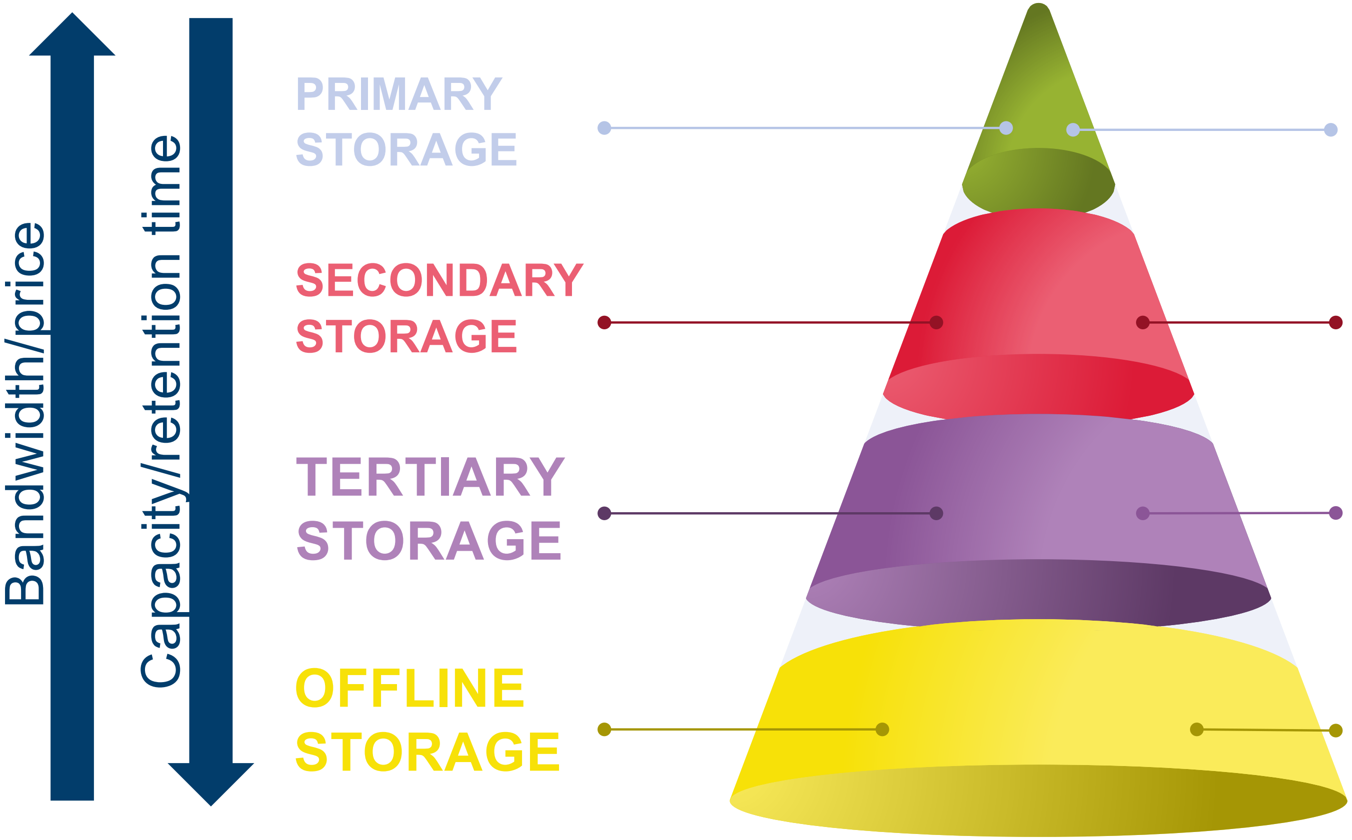
INTRODUCTION AND PARALLEL I/O STRATEGIES

04.11.2024 I ILYA ZHUKOV

JÜLICH Forschungszentrum

Mass storage hierarchy

Four levels of storage hierarchy



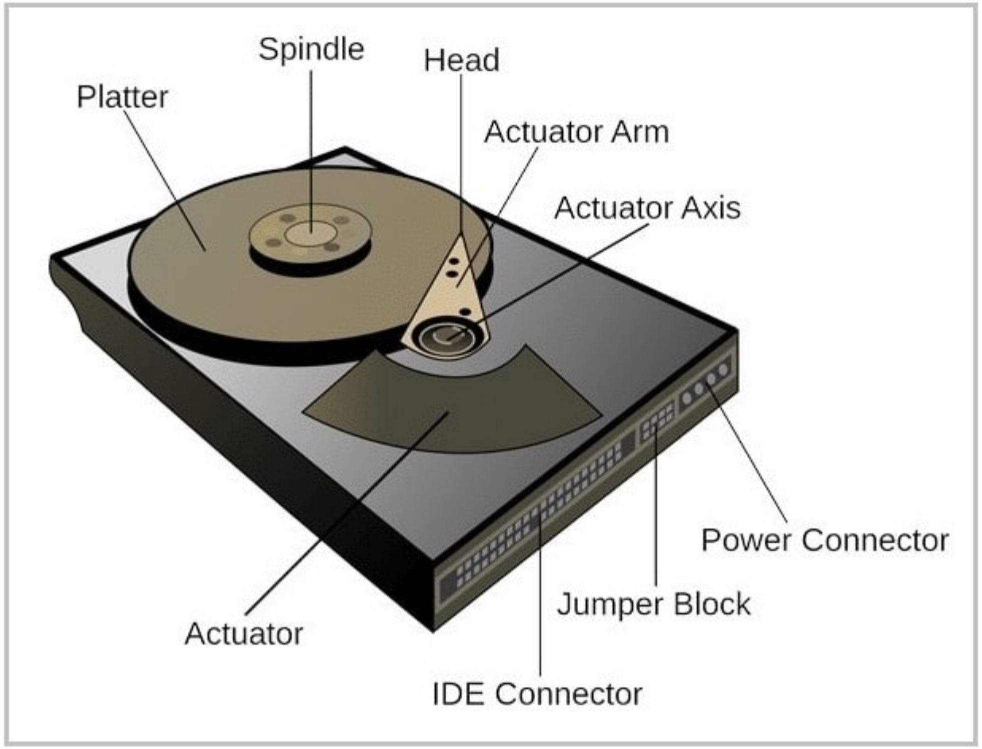
- Volatile memory
- Directly accessed by CPU
- Holds active data & programs
- Examples: CPU register set, caches, system memory

- Non-volatile memory
- Not accessible by CPU directly, mediated by OS
- Granularity limited to fixed-size blocks
- Examples: HDD, SSD

- Not immediately accessible or powered-off, can be quickly enabled for online usage
- Examples: tapes

- Requires human intervention
- Provides additional “air gap“ security level
- Examples: DVD, external disks

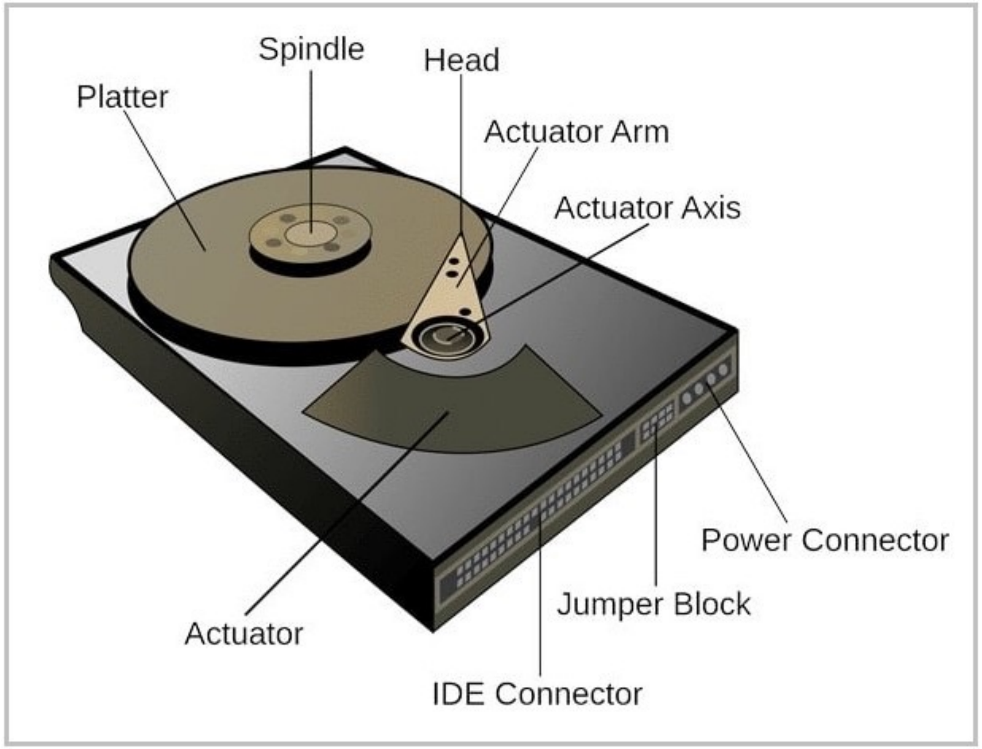
How Hard Disk Drive (HDD) works



- Uses magnetic disks (**platters**) to store data.
- R**ead/Write head** moves over the platters to read and write data.
- **Spindle motor** spins the platters at high speeds for quick access.
- **Actuator arm** positions the read/write head accurately over the desired track on the platter.
- Information is encoded magnetically on the surface of the platters.
- Information on HDD is stored in **blocks**, which are the smallest units of data that can be read or written.
- Sizes: from 500GB to 20TB
- Bandwidth: from 80MB/s to 250MB/s
- For more details watch YouTube video
  - https://youtu.be/wtdnatmVdIg

* Picture is taken from https://www.partitionwizard.com/partitionmagic/how-does-a-hard-drive-work.html

HDD: advantages and disadvantages

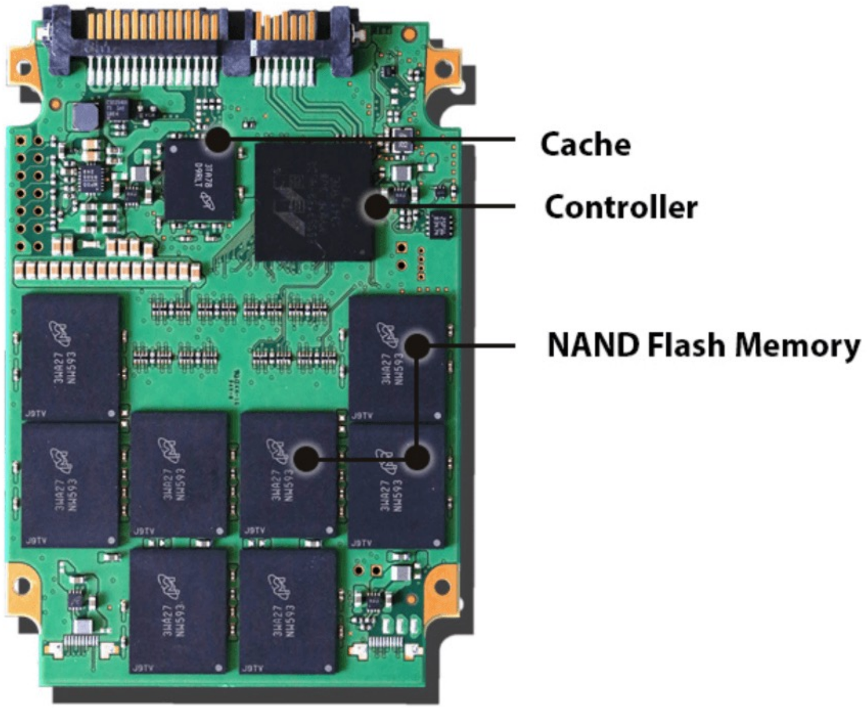


+ Cost-effective: cheaper per GB compared to SSDs.
+ High capacity: available in larger storage sizes, suitable for bulk data storage.
+ Long lifespan: can last for many years with proper use and care.

- Slower speed: slower data access and transfer rates compared to SSDs.
- Fragility: moving parts can be susceptible to damage from shocks and drops.
- Power consumption: typically consumes more power than SSDs.

* Picture is taken from https://www.partitionwizard.com/partitionmagic/how-does-a-hard-drive-work.html

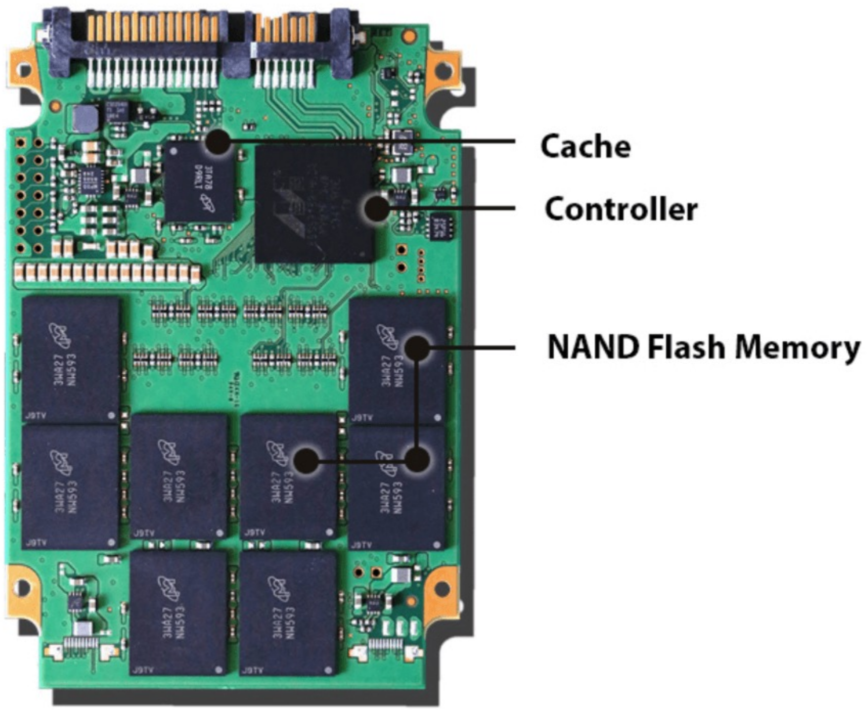
How Solid State Drive (SSD) works



* Picture is taken from https://www.backblaze.com/blog/how-reliable-are-ssds/

- Solid State Drives (SSDs) operate using flash memory chips and a controller to store and manage data.
  - **NAND Flash Memory:** the primary storage medium in SSDs, consisting of interconnected flash memory chips that store data using electrical charges.
  - **Controller:** the brain of the SSD, responsible for managing data processing, error correction, and storage management. It coordinates read/write activities and interfaces with the host system.
  - **Cache:** a high-speed RAM area used to temporarily store frequently accessed data, boosting read and write speeds for small files and random access
  - Common interfaces include SATA, PCIe, and NVMe.

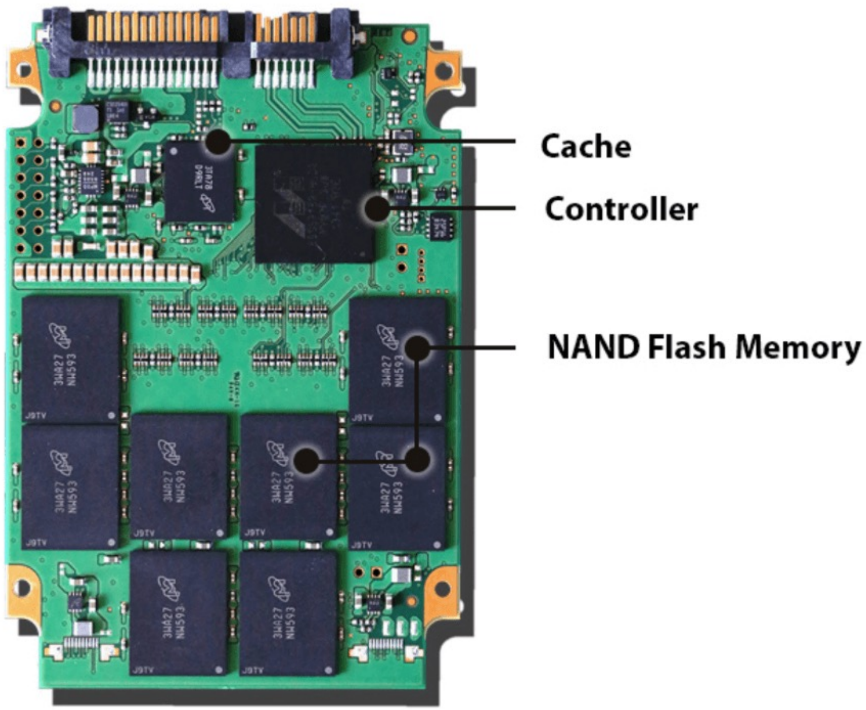
How Solid State Drive (SSD) works II



- SSDs store data using floating gate transistors in memory cells.
- Data is stored as electrical charges in the memory cells, with each cell capable of holding one to four bits of data (SLC, MLC, TLC, or QLC).
- Sizes: from 128GB to 8TB
- Bandwidth: from 500MB/s to 12,000MB/s
- For more details watch YouTube video
  - https://youtu.be/5Mh3o886qpg

* Picture is taken from https://www.backblaze.com/blog/how-reliable-are-ssds/

SSD: advantages and disadvantages



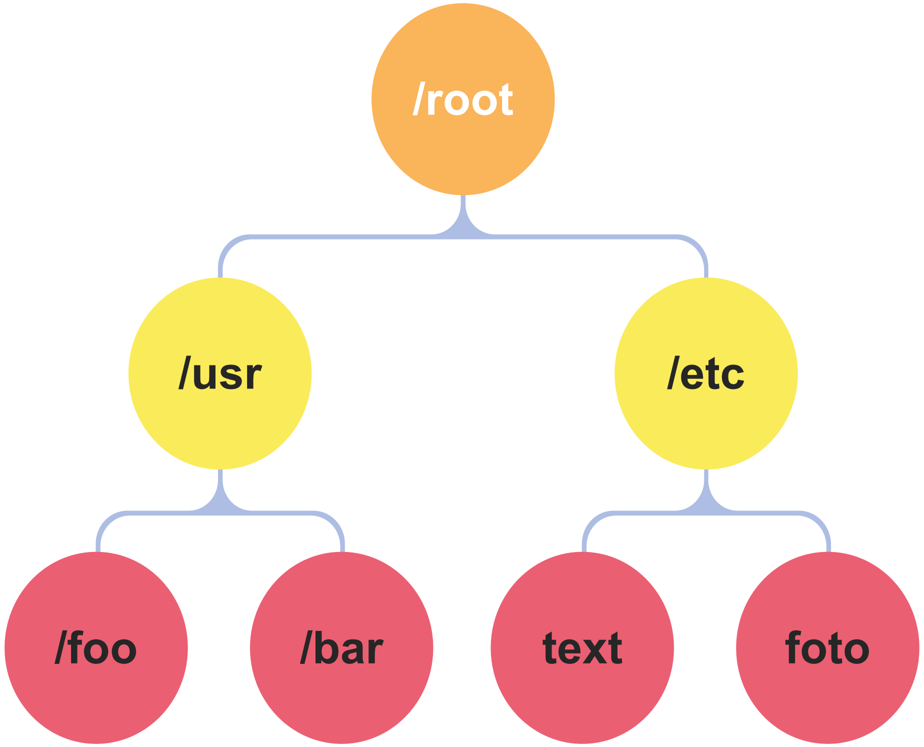
+ **Speed:** SSDs are much faster than HDDs, resulting in quicker boot times and file transfers.
+ **Durability:** With no moving parts, SSDs are more resistant to physical shock and vibration.
+ **Energy efficiency:** SSDs consume less power and generate less heat compared to HDDs.
+ **Silent operation:** The lack of moving parts makes SSDs quieter than HDDs.
+ **Compact size:** SSDs come in various form factors.

- **Limited lifespan:** SSDs have a finite number of write cycles.
- **Higher cost** compared to HDDs.
- **Less storage space** than HDD at comparable price
- **Slower write speeds:** While SSDs excel at reading data, they may take longer to save data compared to HDDs

* Picture is taken from https://www.backblaze.com/blog/how-reliable-are-ssds/

What is a filesystem?

- In Linux and most of OS data are stored in files.
- **File** is a collection of data stored as a single object on a disk.
- **Directory** contains files and stores on a disk.
- Hierachy of directories and files comprises a **filesystem.**
- Path is the unique location of a file or directory within a filesystem.
  - Notation /root/etc/foto
  - There are two types of paths, i.e. absolute and relative.
- Common operations: create, delete, read, write, copy, move.



What is metadata?

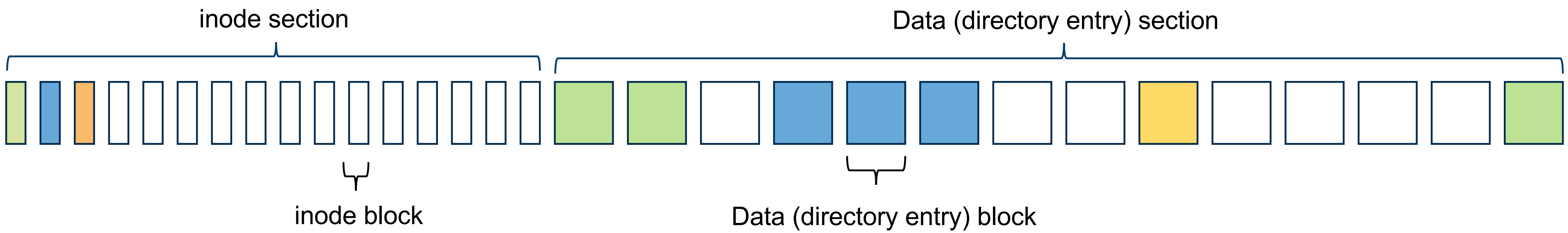
- **Metadata** is essentially “data about data”. It provides essential information about each file and directory without holding the file contents.
- Key metadata elements
  - **File size**: size in bytes.
  - **File type:** specifies if it’s a regular file, directory, symbolic link, etc.
  - **Ownership**: user ID (UID), group ID (GID) of the file owner.
  - **Permissions**: read, write, and execute permissions for the owner, group, and others.
  - **Timestamps**: important dates like creation (ctime), last modification (mtime), and last access (atime).

What is a inode?

- An **inode** (index node) is a data structure in Unix-like filesystems that stores metadata information about each file and directory, excluding the filename and data itself.
- Each inode represents a file or directory and contains pointers to the file data blocks, enabling the filesystem to locate the actual file contents on the disk.
- The **inode number** uniquely identifies each inode within a filesystem.
- A **directory entry** is a fundamental element in Linux filesystems that associates a **filename** with an **inode number**.

Combine software and hardware

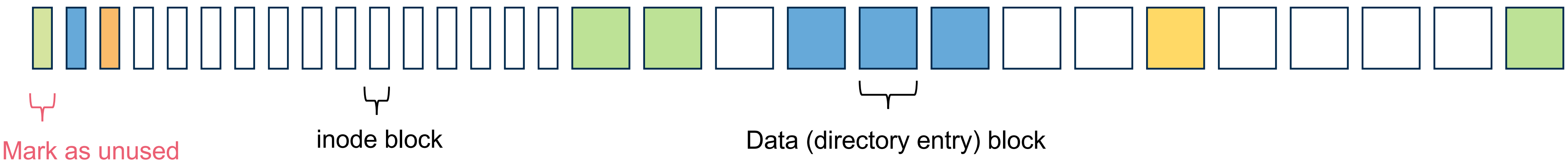
Global view



inode	Type	Permissions	Created	Data blocks	Name
2488	file	rwx	10/10/2024	0,1,13	Matrix.exe
2489	file	r--	01/01/2021	3, 4, 5	42.txt
2490	directory	rw-	02/07/2023	8	Notebook.py

Combine software and hardware

Remove Matrix.exe



inode	Type	Permissions	Created	Data blocks	Name
~~2488~~	~~file~~	~~rwx~~	~~10/10/2024~~	~~0,1,13~~	~~Matrix.exe~~
2489	file	r--	01/01/2021	3, 4, 5	42.txt
2490	directory	rw-	02/07/2023	8	Notebook.py

Understanding permissions

- **Permissions overview**
  - **User**, **Group**, **Others** can each have:
    - **Read** (r): View contents
    - **Write** (w): Modify contents
    - **Execute** (x): Run as program (or access directory)

- **Checking permissions**

```
$ ls -l hello_world
-rwxr-xr-x 1 zhukov1 jusers 188552 Sep  1  2022 hello_world
```

- **Checking directory permissions**

```
$ ls -d test_dir
drwxr-xr-x 5 zhukov1 jusers 8192 Jul 31  2023 test_dir
```

- **Changing permissions: chmod**
  - **Symbolic**
    - chmod [who][+/-][permission] <filename>
    - Example: chmod u+x file.txt
  - **Octal**
    - chmod <permissions> <filename>
    - Example: chmod 755 file.txt
- **Permissions calculator**
  - https://chmodcommand.com

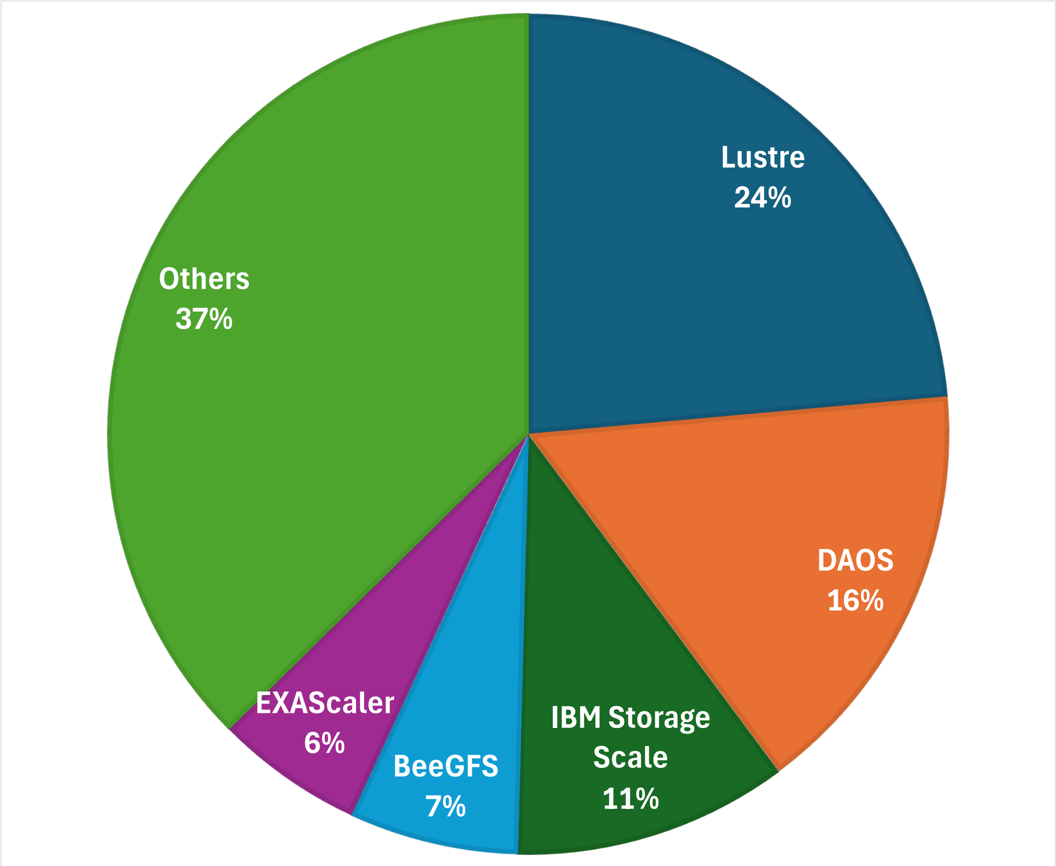
More utilities to test

- **Display detailed metadata information**
  - $ stat <filename>
- **Update file timestamps or create an empty file**
  - $ touch <filename>
- **To create hard and soft links**
  - $ ln
- **Identify associated inode**
  - $ ls -l <filename>
- **Amount of inodes in directory (sorted)**
  - $ du -s --inodes * | sort –rn
- **Global statistics about inode usage**
  - $ df -ih
- **To find out which sub-directories consume how much disk size**
  - $ du -h --max-depth=1 | sort –hr
- **Set and view Access Control Lists (ACLs) for fine-grained permissions**
  - $ setfacl / $ getfacl

Exercise 1

- Go to project directory **/p/project1/training2403**
- Create your own directory with your login name, e.g. **zhukov1**
- Inside this directory create two subdirectories, i.e. **private** and **shared**, and in each subdirectory create file **notes**
  - Modify permissions such as **shared** and **notes** are only accessible for reading for everyone
  - **Private** is only accessible to you.
  - Find a partner next to you and allow only them to modify **notes** in your **shared** directory
  - Modify **notes** in **shared** directory of your partner by writing your name and organisation there
  - Try to open and read information from your partner's private directory, and let them know you did!
- Play with other commands, e.g. identify inode of any file or directory, create hard/soft link to any file, etc.

IO500.org statistics (status from June 2024)



Lustre

- High-performance, distributed parallel filesystem designed for large-scale HPC environments, uses Object Storage Targets (OSTs) and Metadata Servers (MDSs) to manage data and metadata independently.
- **Advantages**
  - High scalability with support for petabytes of data and thousands of clients.
  - Open-source and widely supported in HPC.
  - Robust data management features, including snapshots and replication.
- **Disadvantages**
  - Complex to install, configure, and maintain.
  - Requires dedicated storage and network resources.
  - Some performance degradation under heavy metadata workloads.

DAOS (Distributed Asynchronous Object Storage)

- DAOS focuses on distributing metadata across all servers to eliminate bottlenecks typical in traditional file systems.
- **Advantages**
  - Provides fine-grained data and metadata operations, suitable for HPC and AI workloads.
  - Fully distributed architecture enhances scalability.
- **Disadvantages**
  - Still maturing; limited adoption compared to Lustre or IBM Spectrum Scale.
  - Requires modern storage and networking infrastructure.
  - Can be complex to integrate into existing HPC setups.

IBM Spectrum Scale

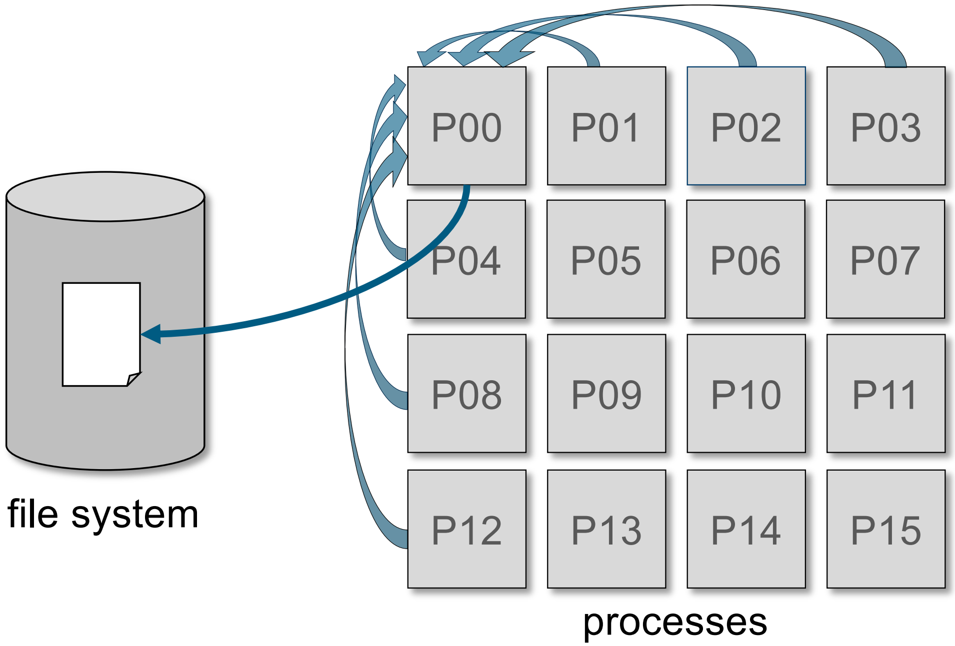
- IBM Spectrum Scale (formerly GPFS) is designed for data-intensive applications, providing a scalable, high-performance file system that supports both traditional and cloud environments.
- **Advantages**
  - High availability and resilience through data replication and tiering.
  - Supports a wide range of storage types.
  - Strong integration with IBM analytics tools enhances data processing capabilities.
- **Disadvantages**
  - Licensing costs can be high compared to open-source alternatives.
  - Setup and tuning are complex, especially for heterogeneous environments.
  - Performance can degrade without sufficient tuning and optimization.

BeeGFS

- BeeGFS is a parallel file system designed for performance and ease of use in clustered environments. It employs a modular architecture that separates metadata from data storage to improve efficiency.
- **Advantages**
  - User-friendly setup and management interface.
  - High performance for both small and large files due to its parallel architecture.
  - Flexible deployment options, suitable for various hardware configurations.
- **Disadvantages**
  - Less mature than Lustre or IBM Spectrum Scale.
  - Limited community support compared to more established systems.
  - May not scale as effectively in extremely large environments as competitors.

Parallel I/O Strategies

One process performs I/O



Parallel I/O Strategies

One process performs I/O

+ Simple to implement

- I/O bandwidth is limited to the rate of this single process
- Additional communication might be necessary
- Other processes may idle and waste computing resources during I/O time

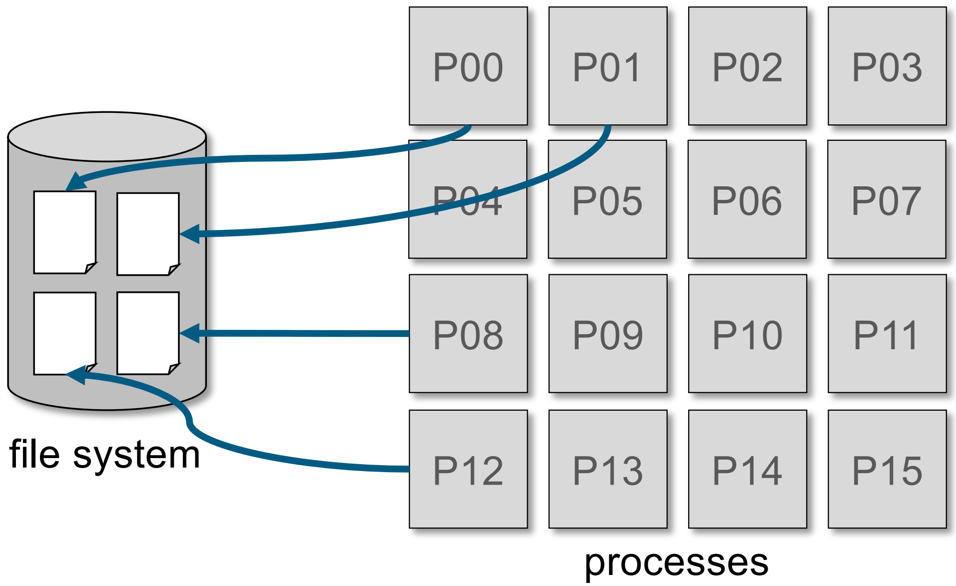
Parallel I/O Pitfalls

Frequent flushing on small blocks

- Modern file systems in HPC have **large file system blocks** (e.g. 16MB)
- A flush on a file handle forces the file system to perform all pending write operations
- If application writes in small data blocks, the same file system block it has to be **read and written multiple times**
- Performance degradation due to the inability to combine several write calls

Parallel I/O Strategies

Task-local files



Parallel I/O Strategies

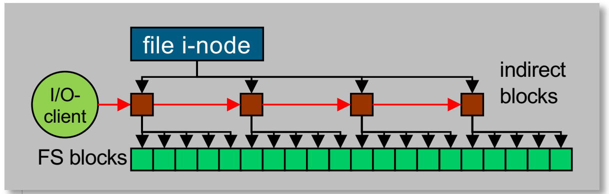
Task-local files

- \+ Simple to implement
- \+ No coordination between processes needed
- \+ No false sharing of file system blocks

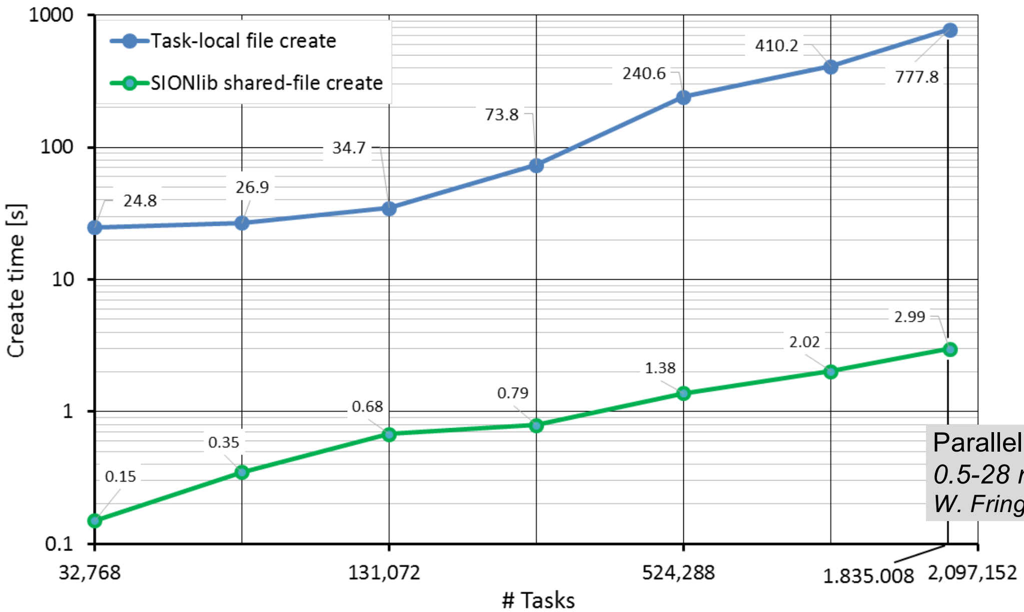
- \- Number of files quickly becomes unmanageable
- \- Files often need to be merged to create a canonical dataset
- \- File system might serialize meta data modification

Parallel I/O Pitfalls

Serialization of meta data modification



Example: Creating files in parallel in the same directory



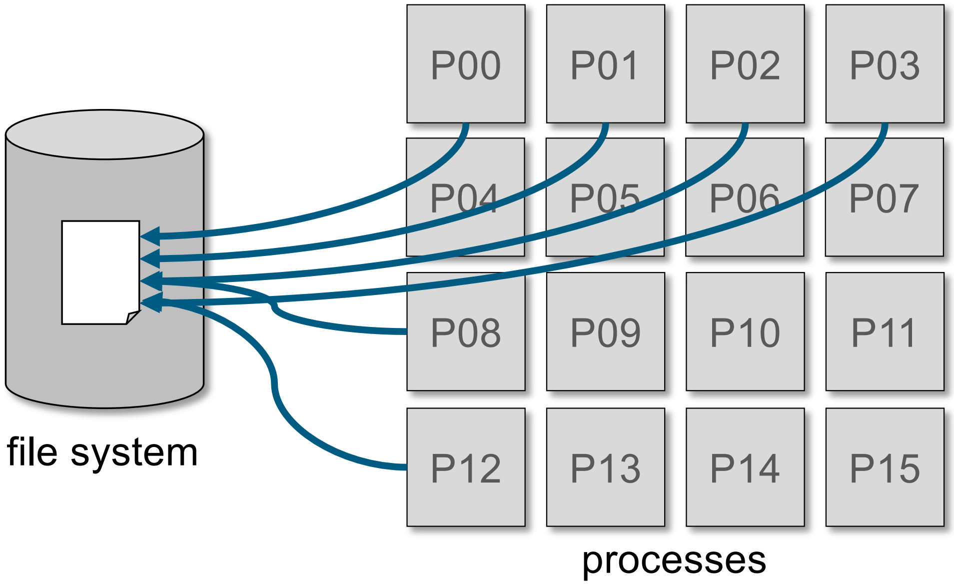
Parallel file creation on *JUQUEEN*
0.5-28 racks, 64 tasks/node
W. Frings

- Meta-data wall on file level
  - File changes by multiple processes can cause serialization
  - File meta-data management
  - Locking

The creation of 2.097.152 files costs 113.595 core hours on JUQUEEN!

Parallel I/O Strategies

Shared files



Parallel I/O Strategies

Shared files

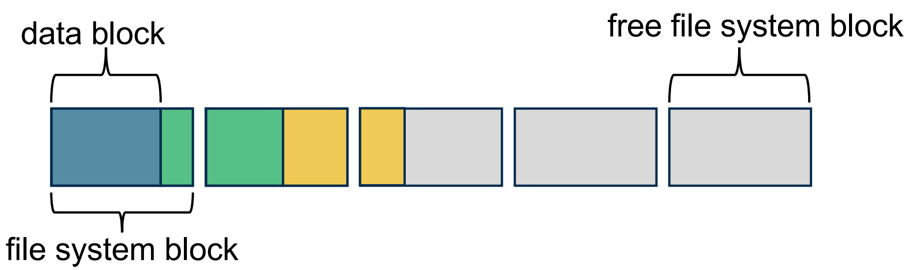
- \+ Number of files is independent of number of processes
- \+ File can be in canonical representation (no post-processing)

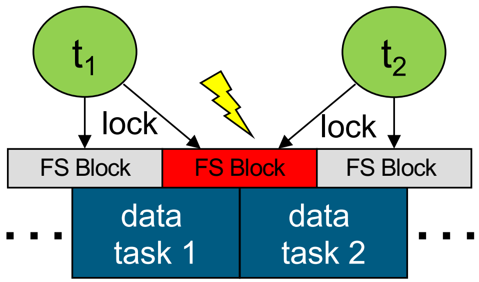
- \- Uncoordinated client requests might induce time penalties
- \- File layout may induce false sharing of file system blocks

Parallel I/O Pitfalls

False sharing of file system blocks

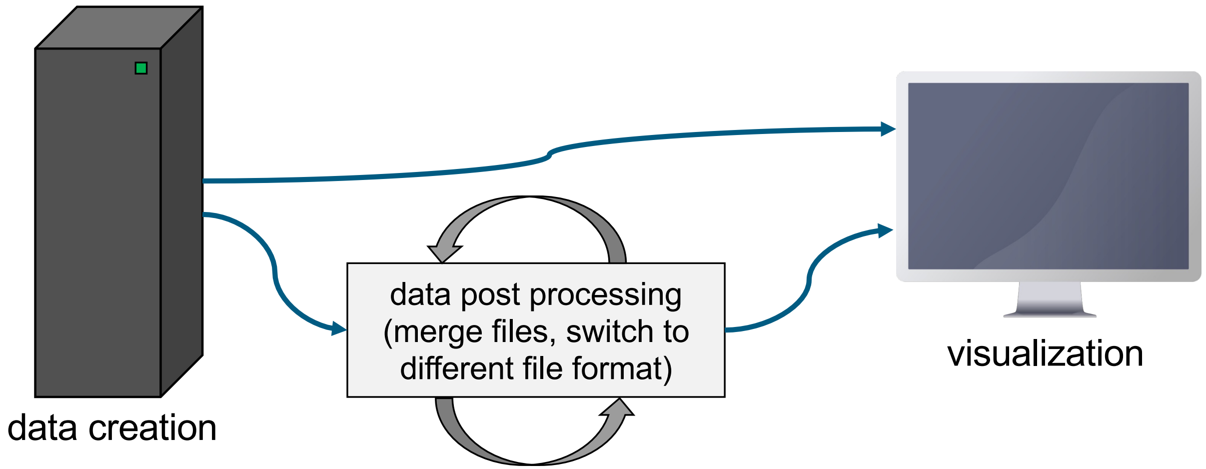
- Data blocks of individual processes do not fill up a complete file system block
- Several processes share a file system block
- Exclusive access (e.g. write) must be serialized
- The more processes have to synchronize the more waiting time will propagate





I/O Workflow

- Post processing can be very time-consuming (> data creation)
  - Widely used portable data formats avoid post processing
- Data transportation time can be long:
  - Use shared file system for file access, avoid raw data transport
  - Avoid renaming/moving of big files (can block backup)



Parallel I/O Pitfalls

Portability

- Endianness (byte order) of binary data
- Conversion of files might be necessary and expensive

2,712,847,316

=

10100001 10110010 11000011 11010100

Address	Little Endian	Big Endian
1000	**11010100**	**10100001**
1001	**11000011**	**10110010**
1002	**10110010**	**11000011**
1003	**10100001**	**11010100**

Parallel I/O Pitfalls

Portability

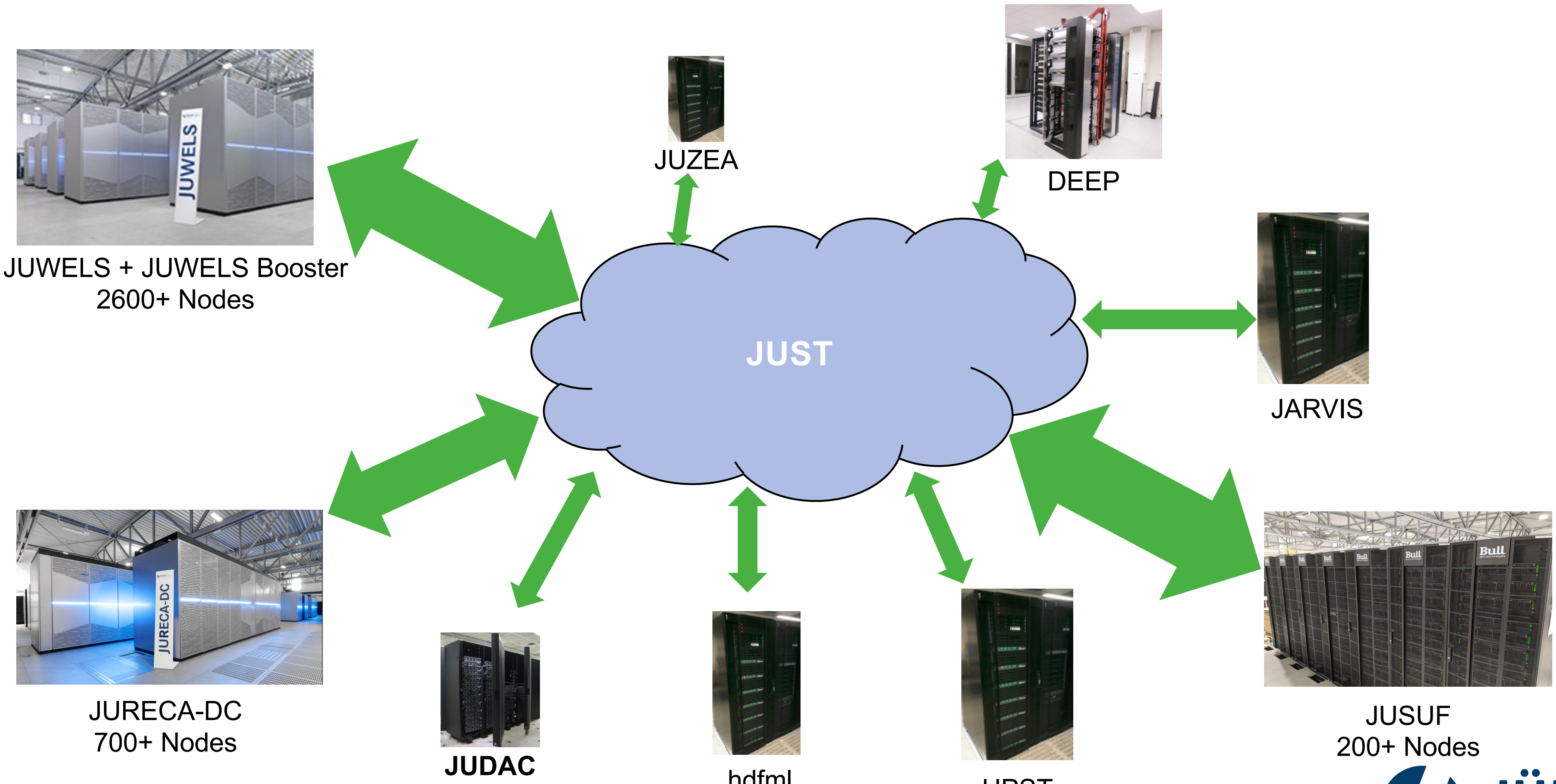
- Memory order depends on programming language
- Transpose of array might be necessary when using different programming languages in the same workflow
- Solution: Choosing a portable data format (HDF5, NetCDF)

1	2	3
4	5	6
7	8	9

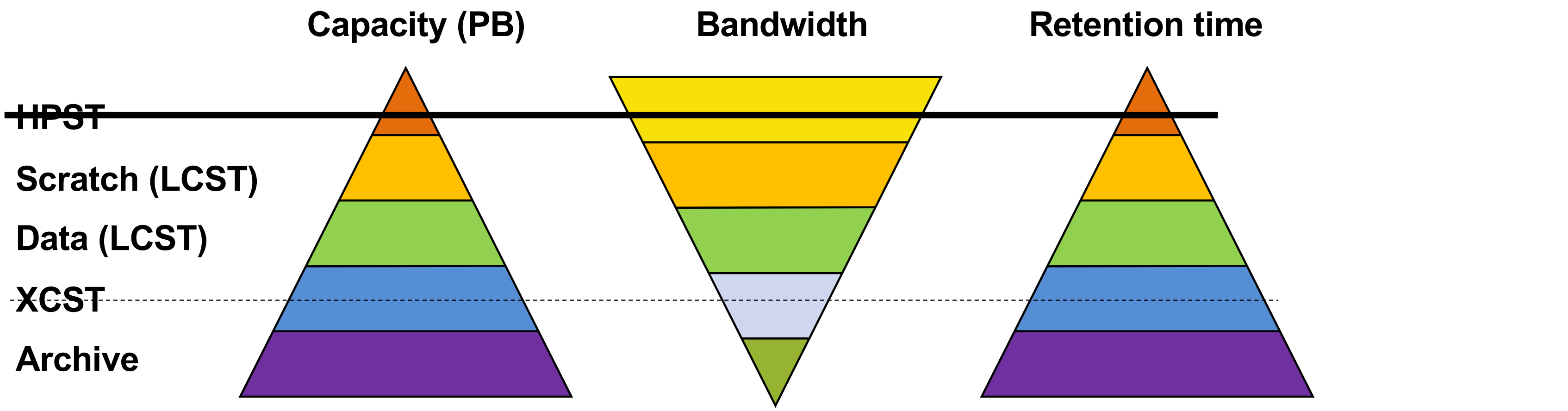
→

Address	row-major order (e.g. C/C++)	column-major order (e.g. Fortran)
1000	1	1
1001	2	4
1002	3	7
1003	4	2
1004	5	5
…	…	…

Juelich STORAGE (JUST)



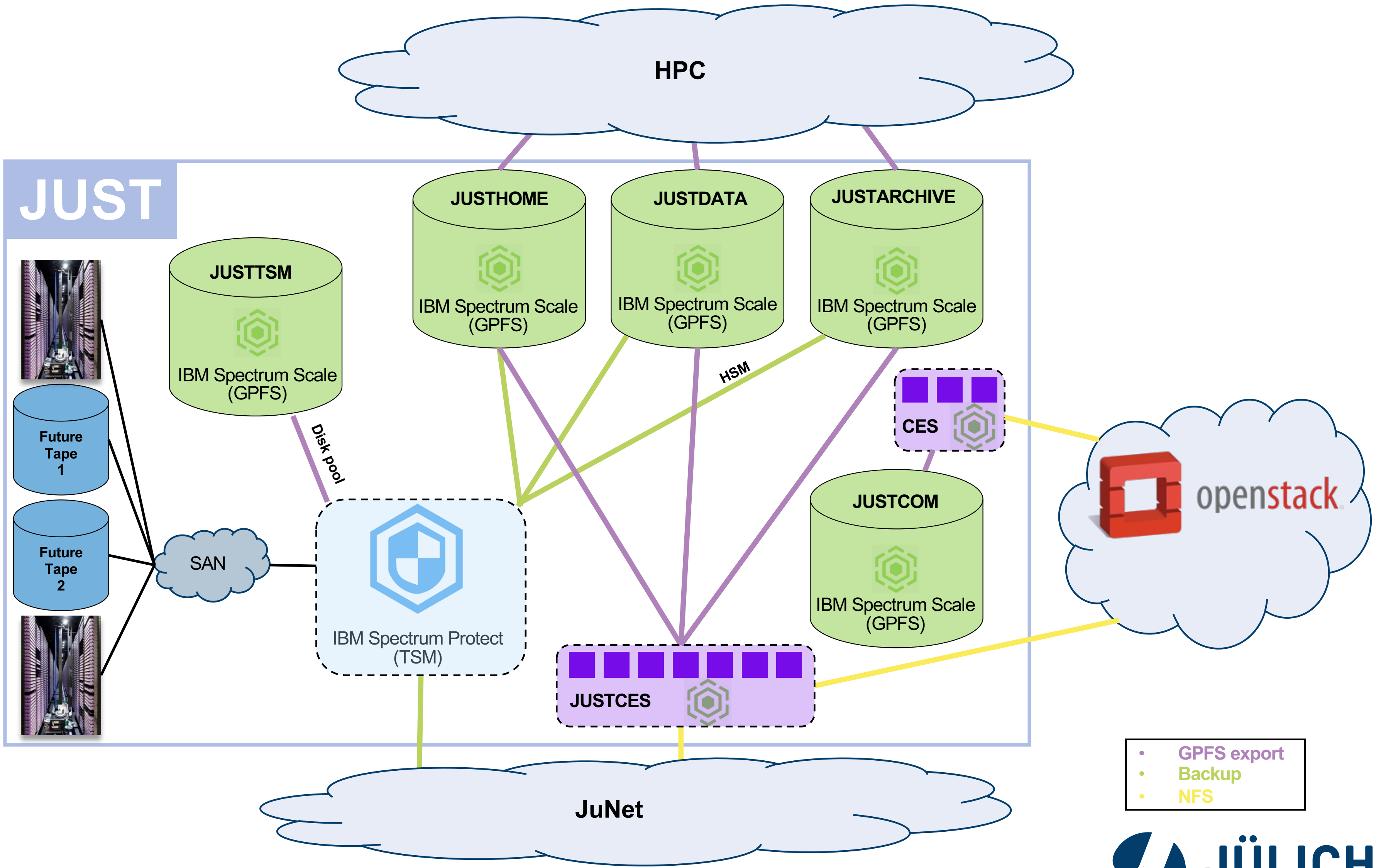
Jülich STORAGE "JUST"



- ~~High Performance Storage Tier (HPST): NVMe based Storage (low latency+high bandwidth)~~
- Large Capacity Storage Tier (LCST): IBM Storage Scale Cluster (GNR, **6th** Gen. of JUST, bandwidth optimized)
- ~~Extended Capacity Storage Tier (XCST): GPFS Building Blocks (target: capacity)~~
- Archive: Tape Storage Tier (Backup + GPFS & TSM-HSM)

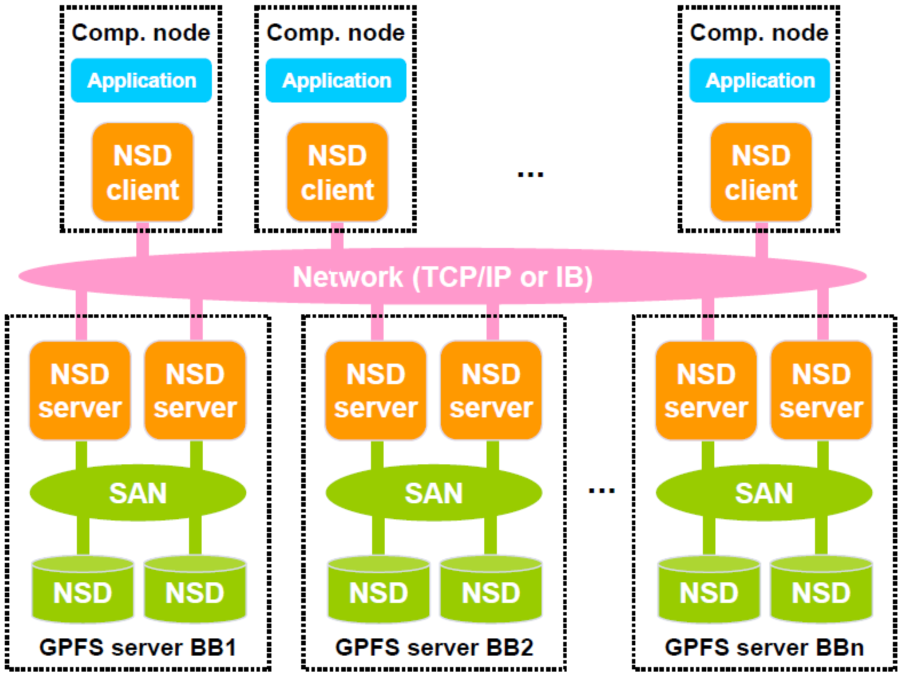
JUST6

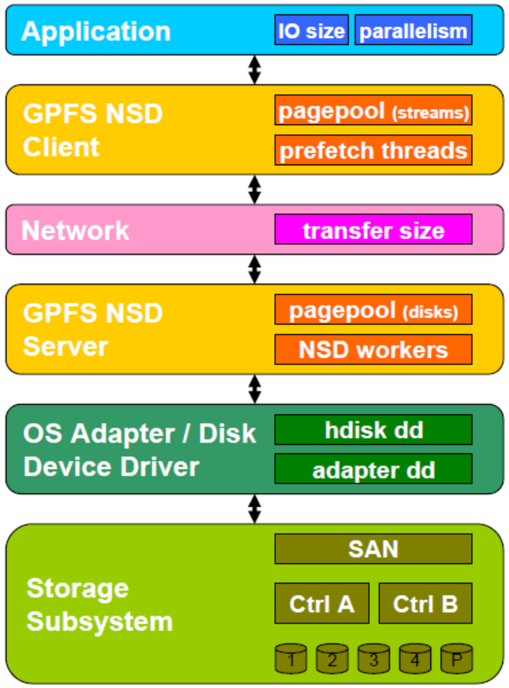
Logical view



System overview

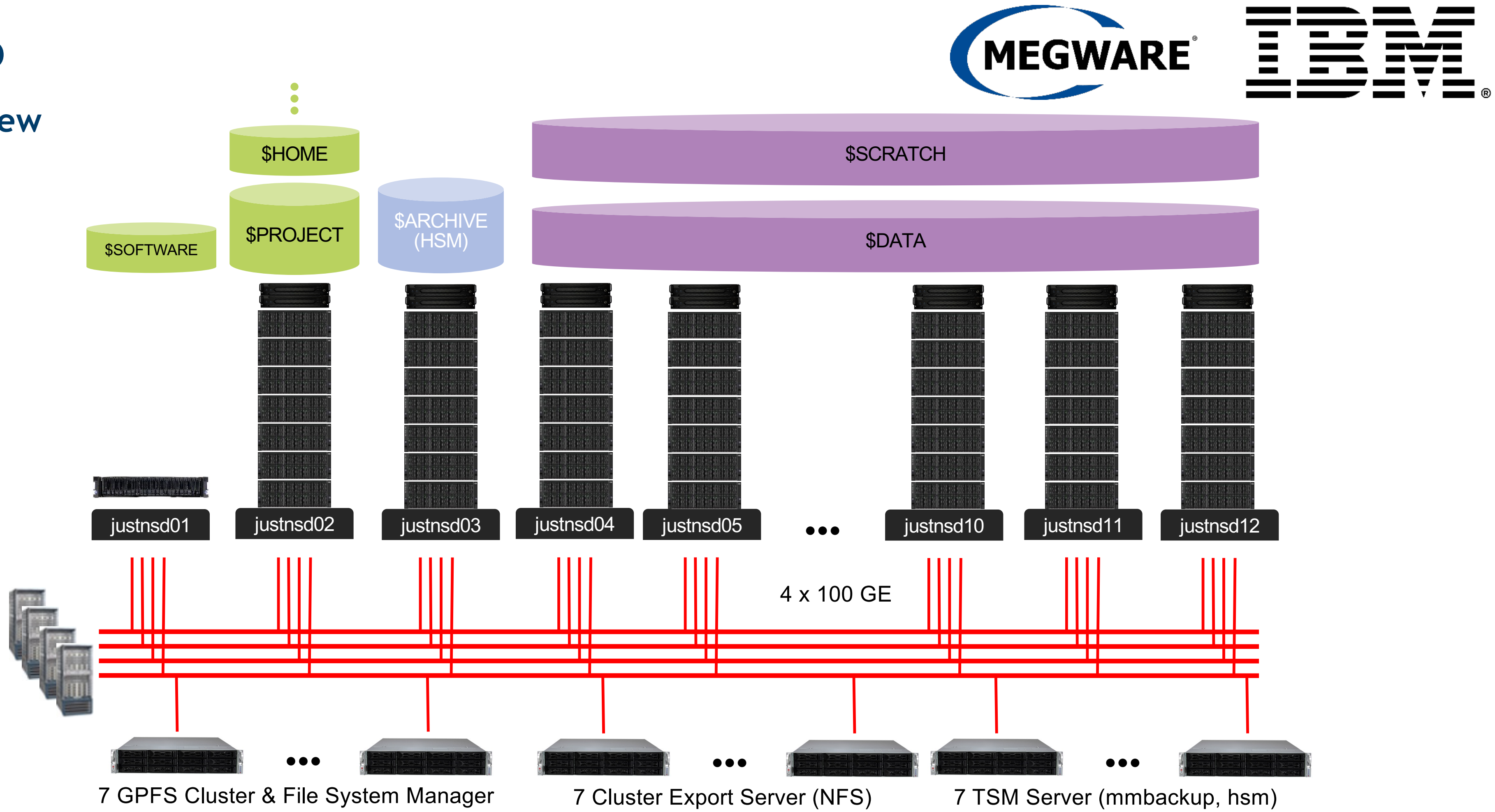
File I/O to GPFS





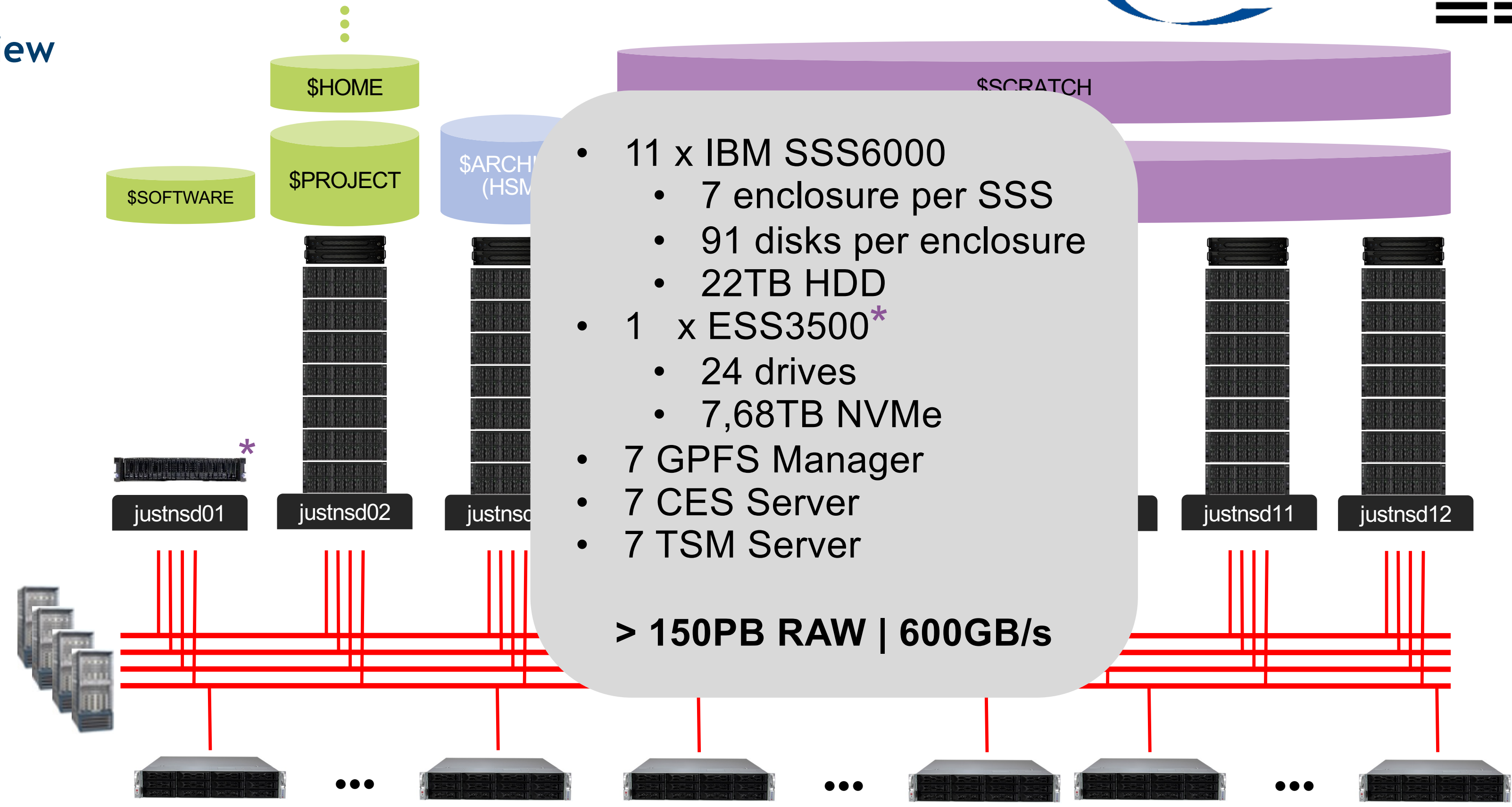
JUST6

Physical view



JUST6

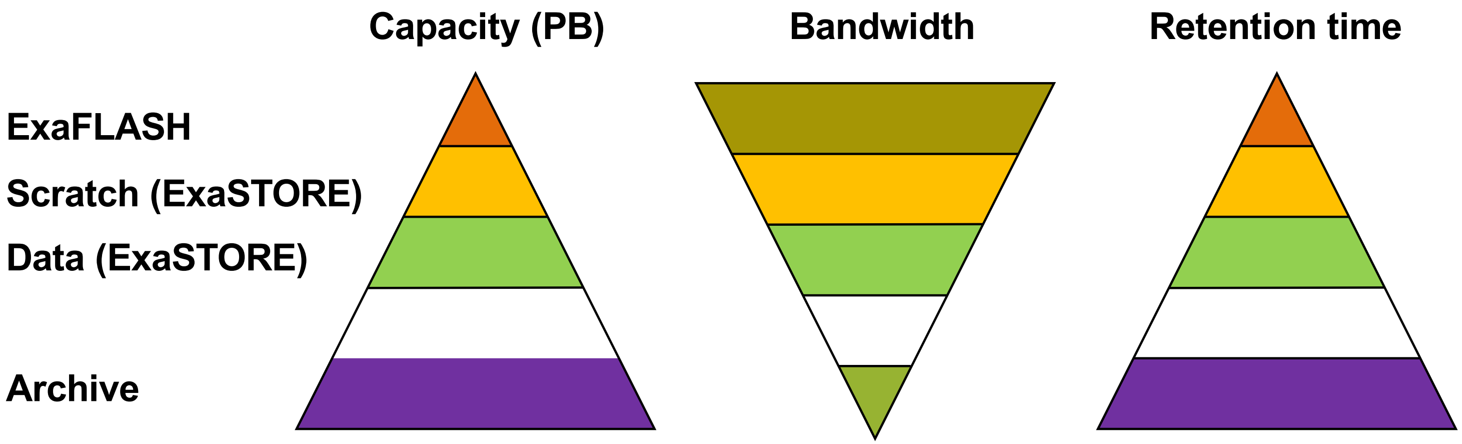
Physical view



- 11 x IBM SSS6000
  - 7 enclosure per SSS
  - 91 disks per enclosure
  - 22TB HDD
- 1 x ESS3500*
  - 24 drives
  - 7,68TB NVMe
- 7 GPFS Manager
- 7 CES Server
- 7 TSM Server

> 150PB RAW | 600GB/s

Jupiter – storage



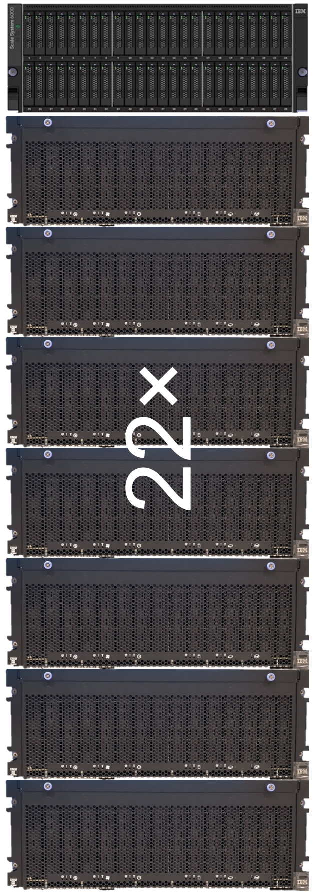
- ExaFLASH: NVMe IBM Storage Scale (low latency+high bandwidth)
- ExaSTORE: Large Capacity Storage Tier, IBM Storage Scale Cluster (GNR, bandwidth optimized)
- Archive: Tape Storage Tier (Backup + GPFS & TSM-HSM) - Same as for JUST6

JUPITER – Storage (Exastore)

In kind contribution from JSC, not part of the JUPITER procurement

MEGWARE IBM

- **Gross Capacity: 308 PB; Net Capacity: 210 PB**
- **Bandwidth: 1.1 TB/s Write, 1.4 TB/s Read**
- 22× IBM SSS6000 Building Blocks (44 servers)
  - 2× NDR200 per server
  - 7× JBOD enclosures, each with 91x 22 TB Spinning Disks per block
  - IBM Storage Scale (aka Spectrum Scale/GPFS)
- 1 x IBM Storage Scale System 3500: 24 x 7.68 TB NVMe
- Manager and Datamover Nodes
- Exclusive for JUPITER: Integrated into InfiniBand fabric
- Same HW as JUST6 for flexibility to move HW



Preliminary numbers, might change during installation

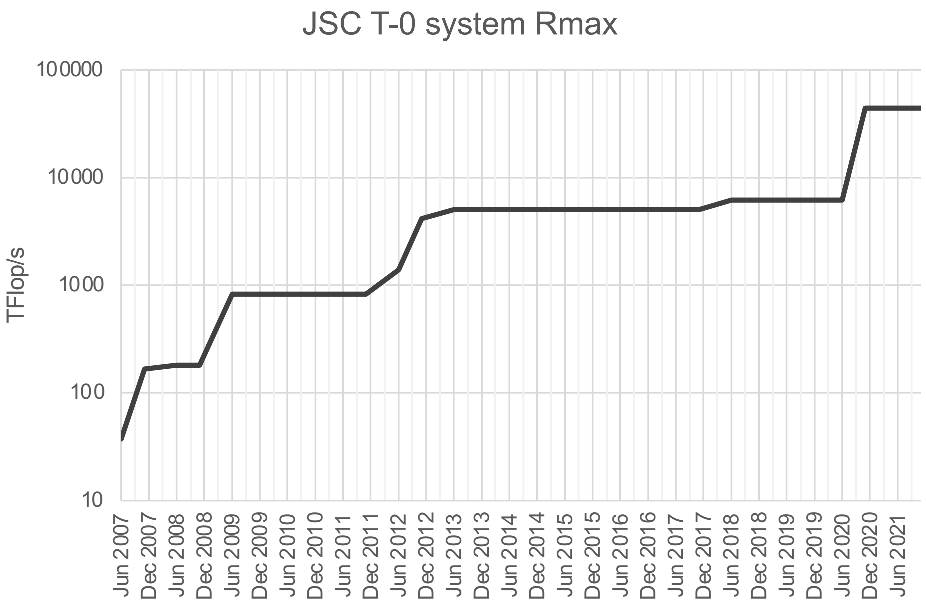
JUPITER – Storage (ExaFlash)

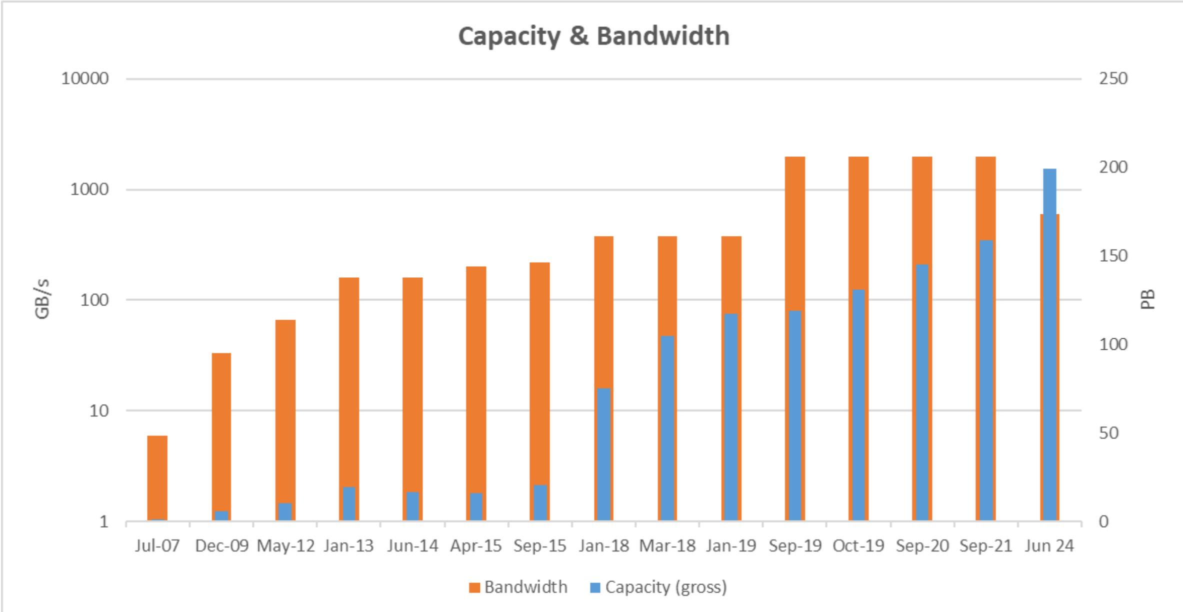
- **Gross Capacity: 29 PB; Net Capacity: 21 PB**
- **Bandwidth: 2.1 TB/s Write, 3.1 TB/s Read**
- 20× IBM SSS6000 Building Blocks (40 servers)
  - 2× NDR400 per server
  - 48× 30 TB NVMe drives per block
  - IBM Storage Scale (aka Spectrum Scale/GPFS)
- Manager and Datamover Nodes
- Exclusive for JUPITER: Integrated into InfiniBand fabric

Preliminary numbers, might change during installation

System overview

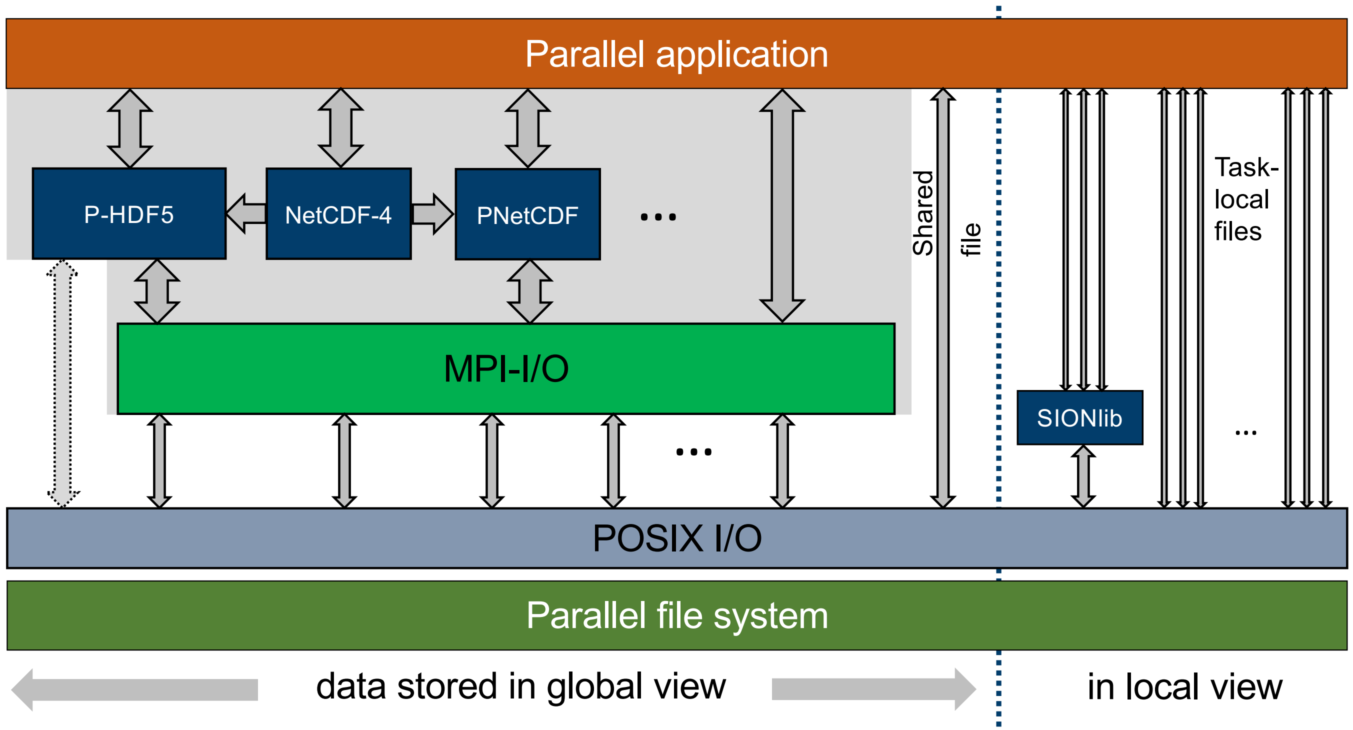
Computational vs I/O performance





JÜLICH Forschungszentrum

Parallel I/O Software Stack



GREETINGS FROM THE STORAGE TEAM

Questions?