Data Storage and Filesystems

COMPLECS Team

https://bit.ly/COMPLECS
https://github.com/sdsc-complecs

SDSC
SAN DIEGO SUPERCOMPUTER CENTER
UC San Diego

COMPLECS NSF award #2320934

About COMPLECS

COMPLECS (COMPrehensive Learning for end-users to Effectively utilize CyberinfraStructure) is a new training program offered by SDSC that covers the most important ***non-programming*** concepts and skills that you need to effectively use supercomputers. Topics include parallel computing concepts, Linux tools and bash scripting, security, batch and interactive computing, how to get help, and data management.

COMPLECS is supported by NSF CISE/OAC-2320934

COMPLECS: Data Management Series

Data Transfer

How to get the data you need for your research to and from high-performance computing systems.

Data Storage and File Systems

How to use the data storage and file systems you'll find mounted on high-performance computing systems.

Parallel I/O

How to make effective use of parallel filesystems.

Data Storage and Filesystems

Learning Goals:

- What is a filesystem?
- What are the most common file systems you'll find on HPC systems today?
- How do you choose which file system to use for your work?
- Why is understanding metadata so important to effectively utilizing HPC storage systems?

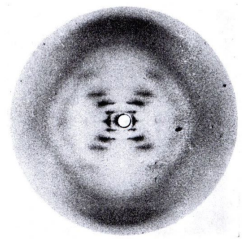
Prerequisite Knowledge:

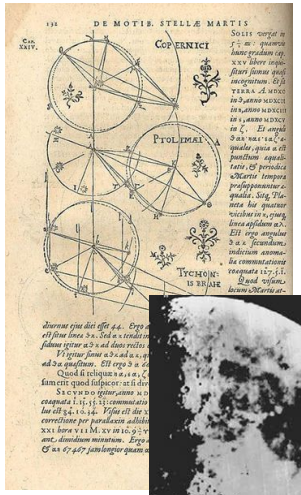
- Intermediate Linux and Shell Scripting

Table of Contents

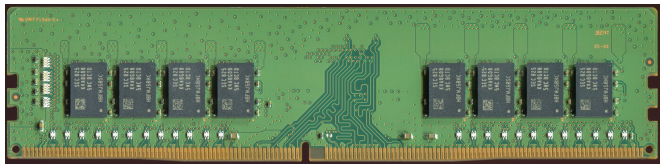
Data Storage

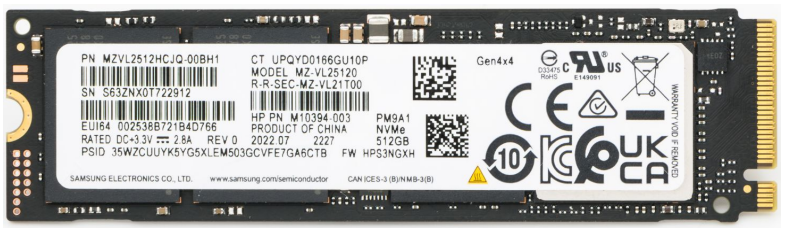
Data storage is the process of recording information (or data) to a ***storage medium*** such that it is both preserved and retrievable.



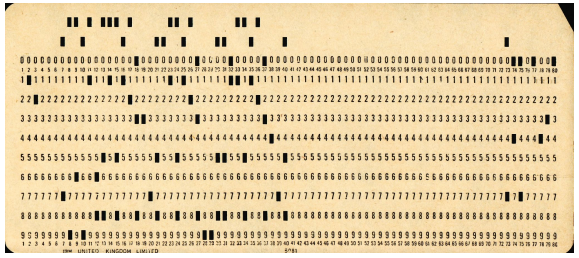






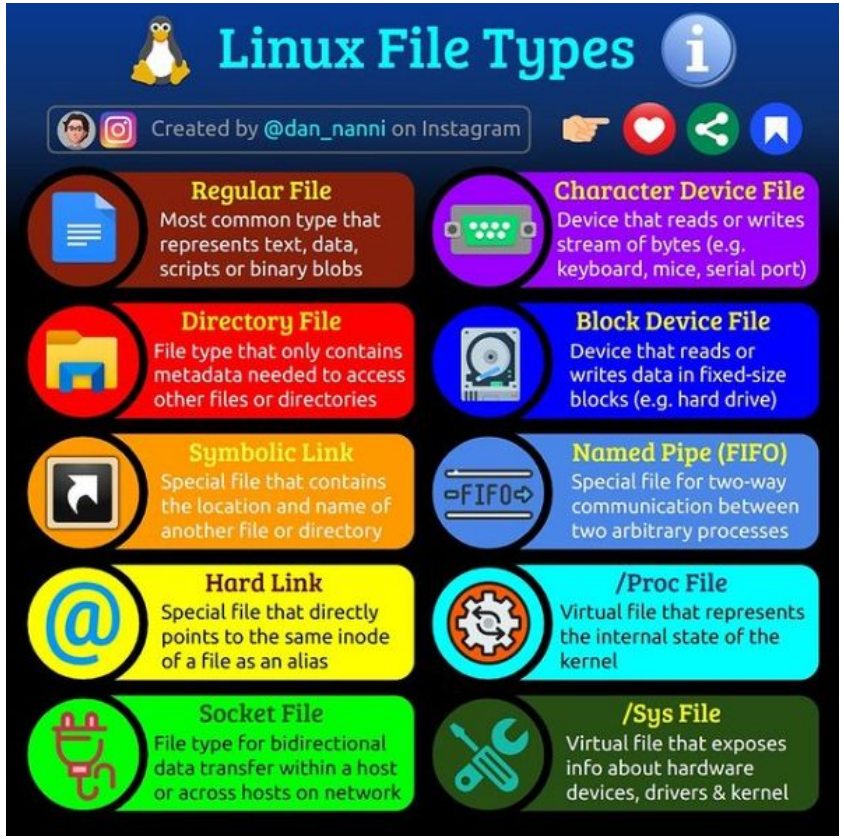



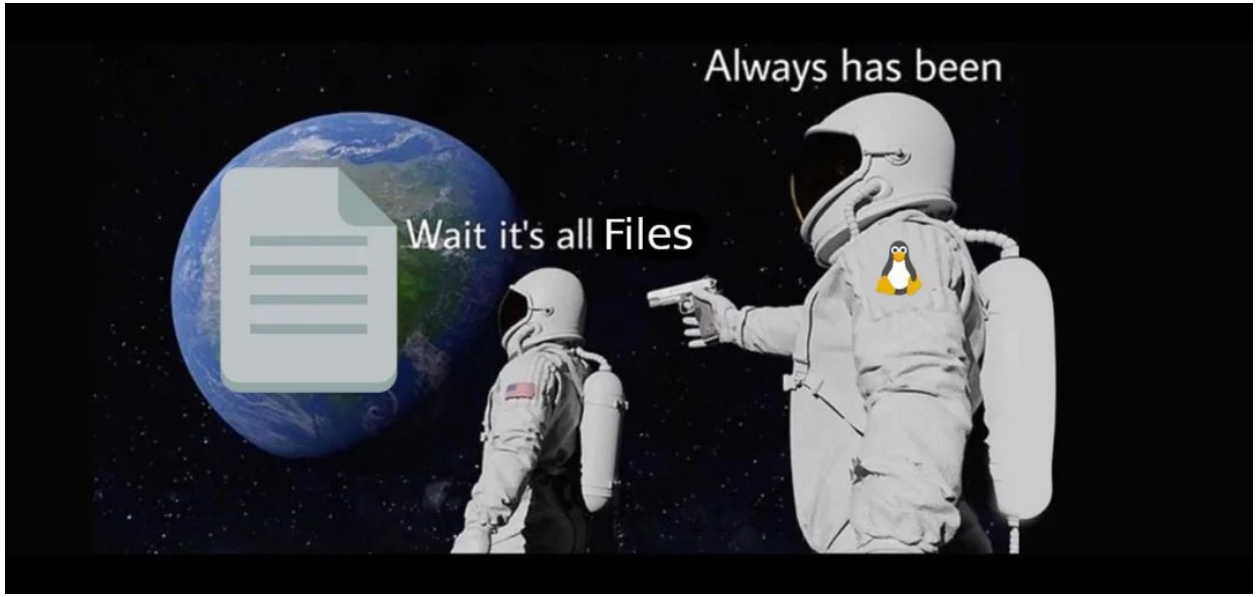


Computer File

A **computer file** is a system resource for recording data on a storage device. Files have different specialized formats to provide encodings for the particular type of data or information that they are designed to store.





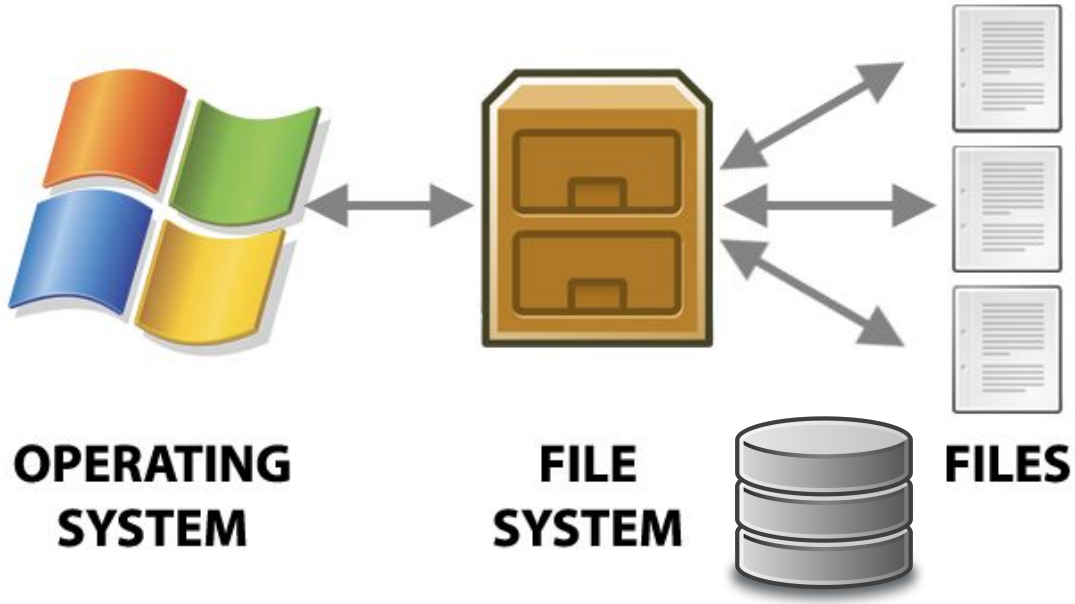
Everything is a file!

File System

A **file system** (also often written as filesystem) provides an operating system with the ability to create, organize, and manage access to files on storage devices.

All filesystems support a set of standard operations that can be performed on a file:

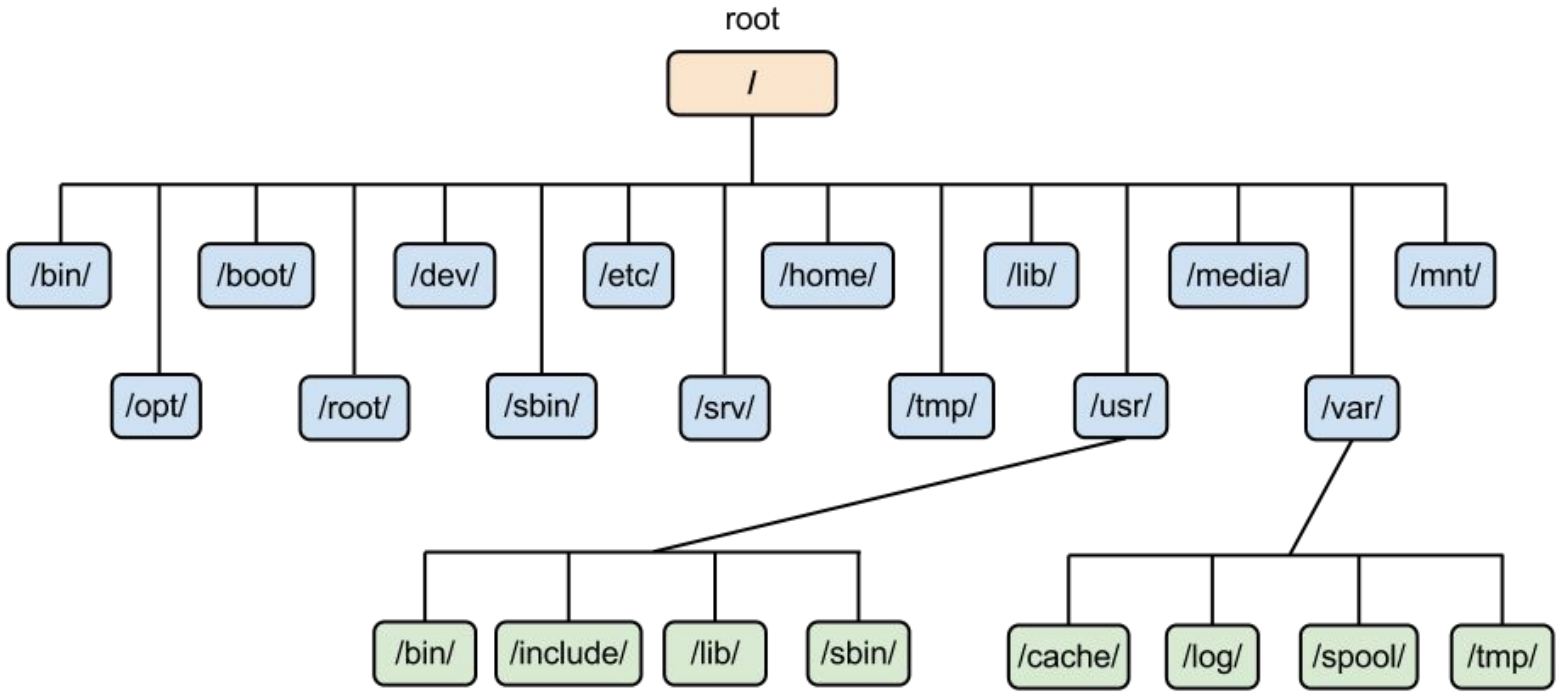
- Create a new file
- Change access permissions and/or attributes of a file
- Open a file
- Read data from a file
- Write data to a file
- Delete a file
- Close a file



Nearly all of the filesystems you will likely interact with on an HPC system are POSIX-based.

Filesystem Hierarchy Standard

The Filesystem Hierarchy Standard (FHS) describes the conventions used to organize the layout of standard directories and files in Unix-like operating systems such as Linux.




```
mkandes@login02:~
[mkandes@login02 ~]$ tree -L 1 -d /
/
├── bin -> usr/bin
├── boot
├── cm
├── cvmfs
├── dev
├── etc
├── expanse
├── home
├── lib -> usr/lib
├── lib64 -> usr/lib64
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── scratch
├── srv
├── sys
├── tftpboot
├── tmp
├── usr
└── var

25 directories
[mkandes@login02 ~]$
```

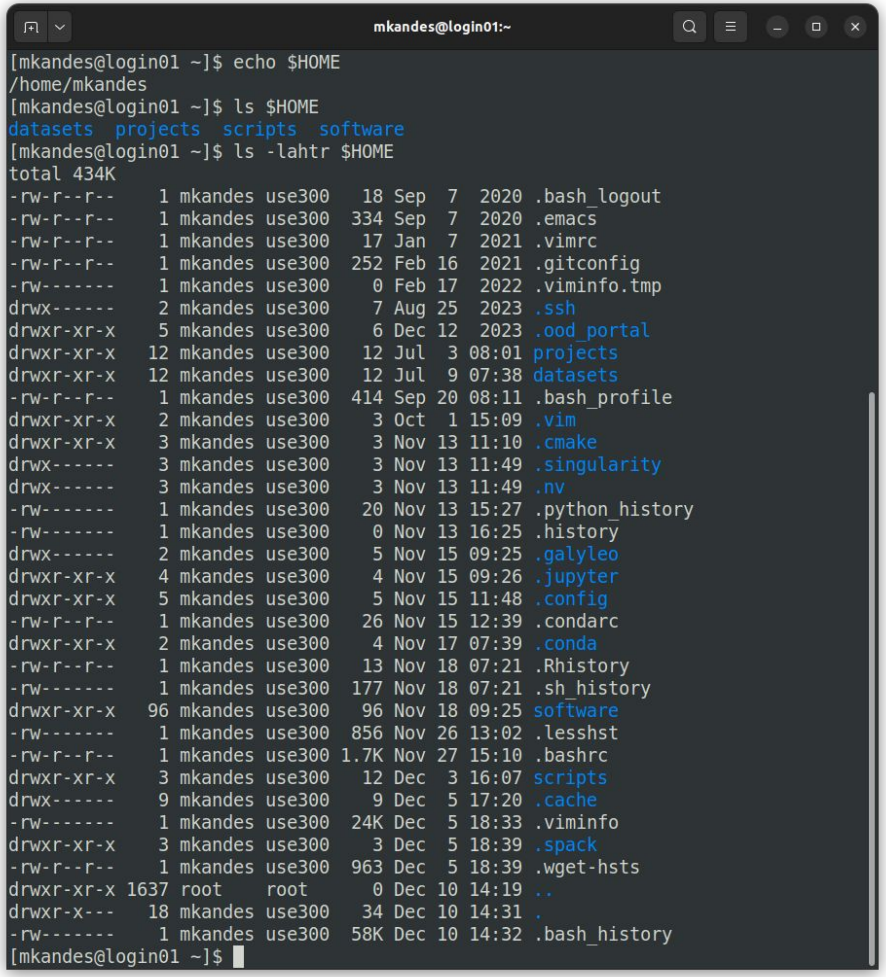
The most important directory for you is **`/home`**. This is where your **`$HOME`** directory resides.

$HOME

A $HOME directory on a shared multi-user system contains the directories and files owned by a given user of the system.

On an HPC system, your $HOME directory:

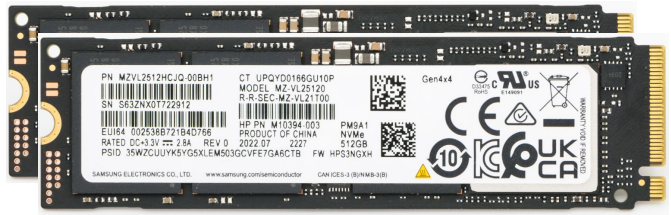
- is your working directory on login.
- is where you will install software, create and submit batch job scripts, and store input and output files for your jobs.
- has a limited amount of storage available. i.e., you will need to store large amounts of data on other types of filesystems.
- can support a limited number of read/write operations. i.e., you may need to limit the number of jobs interacting with its filesystem.



Local vs. Distributed Filesystems

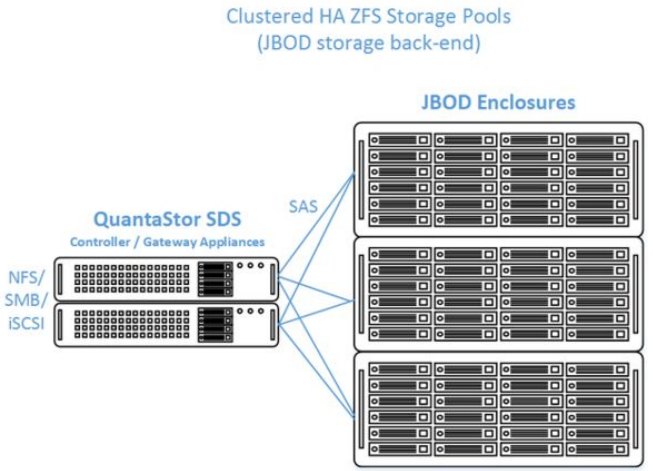
There are many different types of (POSIX) filesystems. And every type of filesystem has its own advantages and limitations, which will determine ***how*** and ***where*** you will want to store different ***types*** of data at any given ***time***. However, there are two general classes of filesystems you will interact with on any HPC system.

A **local file system** provides the ability to interact with files on a storage device ***attached to the same computer.*** In other words, files are *isolated* and *not accessible from other computers*.



RAID

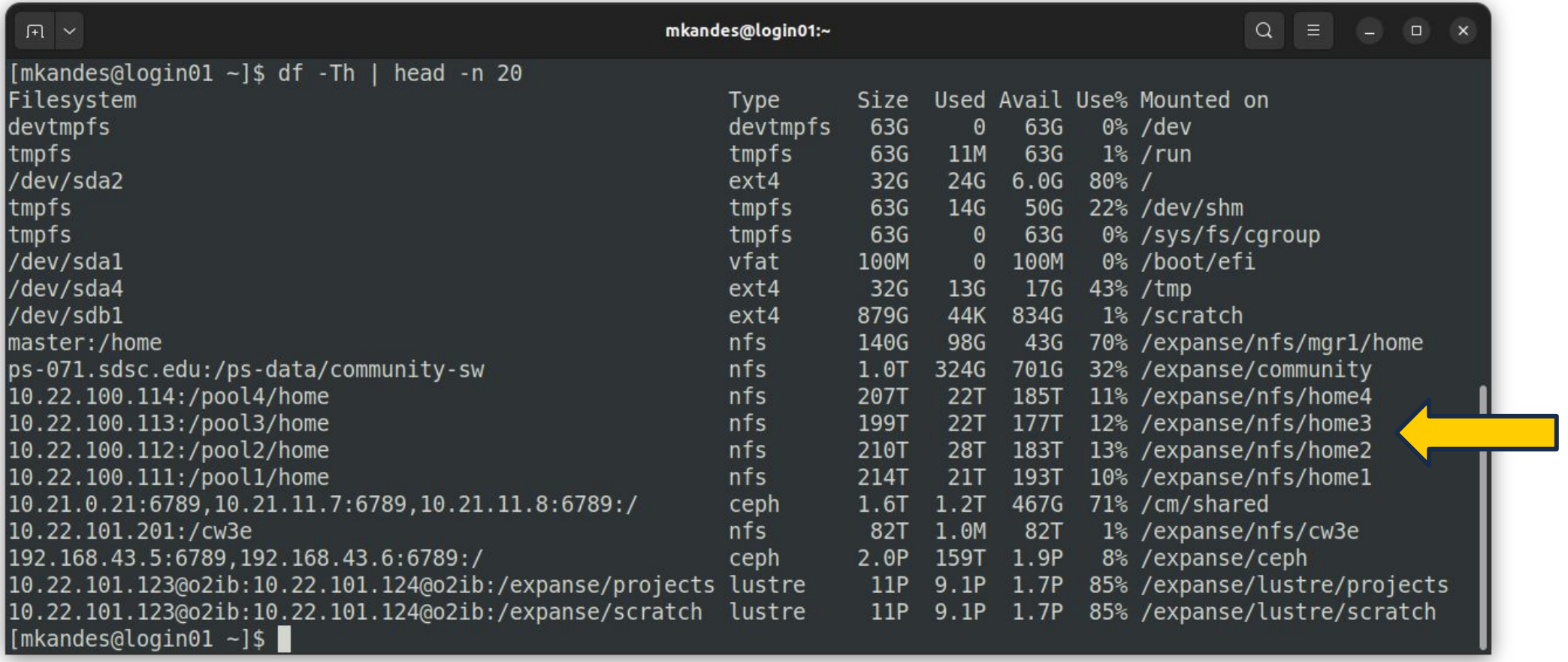
A **distributed file system** provides a service to store and ***share*** files across multiple storage devices ***over a network*** protocol. In general, however, they allow files to be accessed by many nodes in a cluster using the same interfaces and semantics as a local file system.



HA

df – report file system disk space usage

`df` (an abbreviation for *disk free*) is a standard Unix command used to display the amount of available disk space for file systems.



Disk Quota

A disk quota (or simply, quota) is a limit set by a system administrator that restricts certain aspects of file system usage on modern operating systems. In general, quotas are used to *allocate limited disk space* in a reasonable way.

So, how do you find out what your quota is?

Unfortunately, this will vary by filesystem. Some filesystems have standard command-line tools, while some HPC administrators have their own custom command-line tools and/or storage usage reporting for users.

The best place to start is to review the storage section of your HPC system's user guide.
e.g., https://www.sdsc.edu/support/user_guides/expanse.html#storage

Note, however, there is one standard command to check how much space you are currently consuming on a given filesystem …

du - estimate file space usage

du (an abbreviation for *disk usage)* is a standard Unix program used to estimate file space usage — space used under a particular directory or files on a file system.

```
mkandes@login02:~
[mkandes@login02 ~]$ du -h $HOME
2.8M    /home/mkandes/software/mpi4py/releases
2.8M    /home/mkandes/software/mpi4py
14M     /home/mkandes/software/iqtree2/releases
14M     /home/mkandes/software/iqtree2
217M    /home/mkandes/software/gcc/releases
217M    /home/mkandes/software/gcc
12M     /home/mkandes/software/subversion/releases
12M     /home/mkandes/software/subversion
1.6M    /home/mkandes/software/pkgconf/releases
1.6M    /home/mkandes/software/pkgconf
19M     /home/mkandes/software/hdf5/releases
19M     /home/mkandes/software/hdf5
3.2M    /home/mkandes/software/bamutil/releases
3.2M    /home/mkandes/software/bamutil
1.6M    /home/mkandes/software/git-lfs/releases
1.6M    /home/mkandes/software/git-lfs
4.5G    /home/mkandes/software/spark
24M     /home/mkandes/software/lapack/releases
24M     /home/mkandes/software/lapack
4.7M    /home/mkandes/software/mpmath/releases
4.7M    /home/mkandes/software/mpmath
263M    /home/mkandes/software/boost/releases
263M    /home/mkandes/software/boost
```

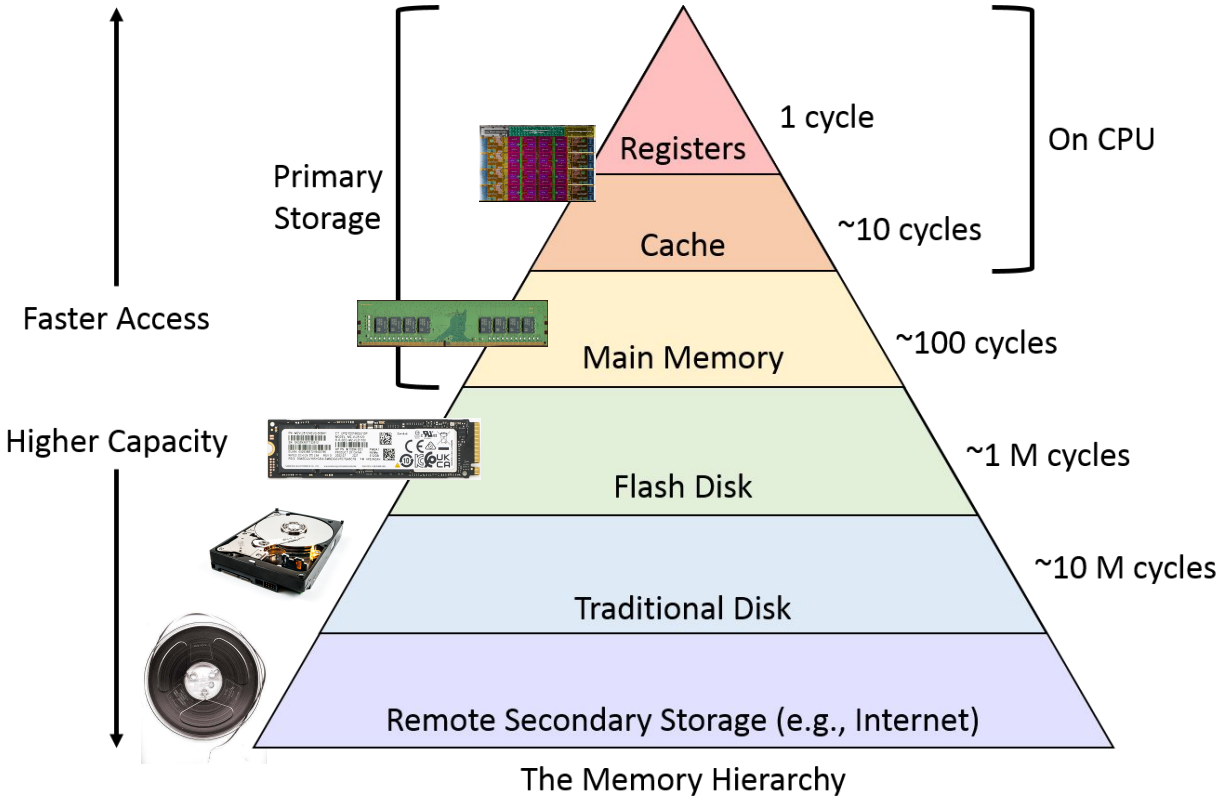
This command can take quite some time to run depending on the total number of directories and files AND on the type of filesystem the top-level directory is located on.

Table of Contents

- What is a filesystem?
- Data storage devices and systems
- Files, Directories, and Inodes
- Filesystems in High-Performance Computing
- Filesystem performance: An AI example
- Summary and Conclusion: Best practices
- Questions & Answers (30 min)

Computer Data Storage Systems (or Devices)

A **computer data storage system** (or device) consists of the physical hardware components, recording media, and other technologies that are used to retain digital information (or data).



The Memory Hierarchy

The memory hierarchy separates computer storage based on response time, which is typically referred to as latency.

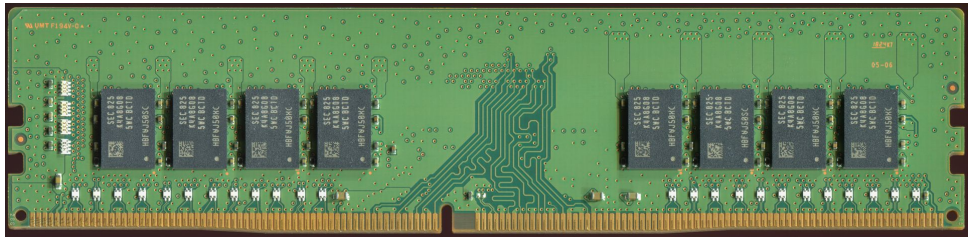
Four key measures of a storage system are:

- **Capacity** - How much data can it hold?
- **Performance** - How quickly and efficiently a can it read and write data?
- **Reliability** - How long can it maintain data integrity and availability over time?
- **Scalability** - How well can it accommodate growth in data volumes and compute workloads?

Memory

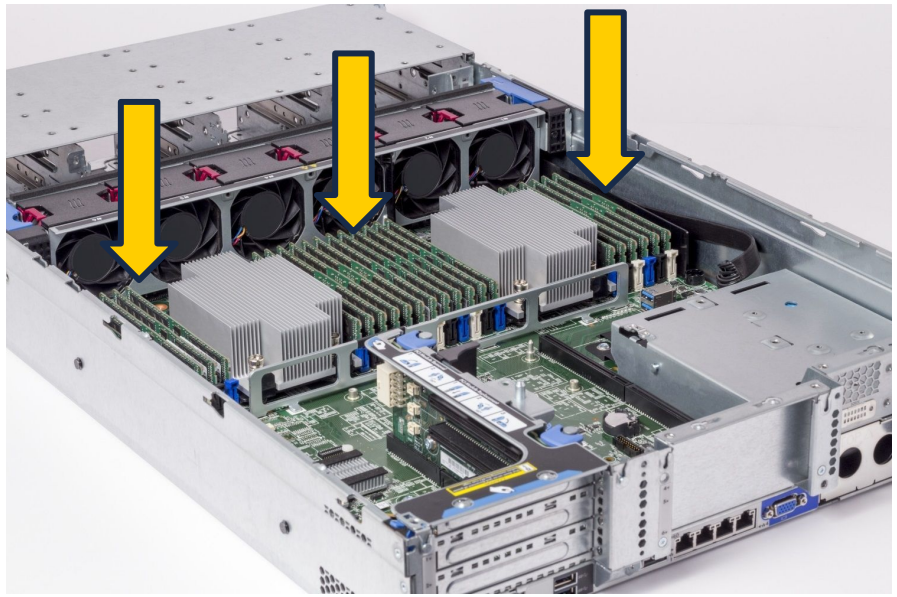
Memory stores information, such as data and programs, for immediate use in a computer's central processing unit(s) (or some other computation device — e.g. a GPU). Memory is often synonymous with the terms RAM, main memory, or primary storage.

Memory modules on HPC systems are almost always error-correcting, which increases their reliability while in use.



DDR4

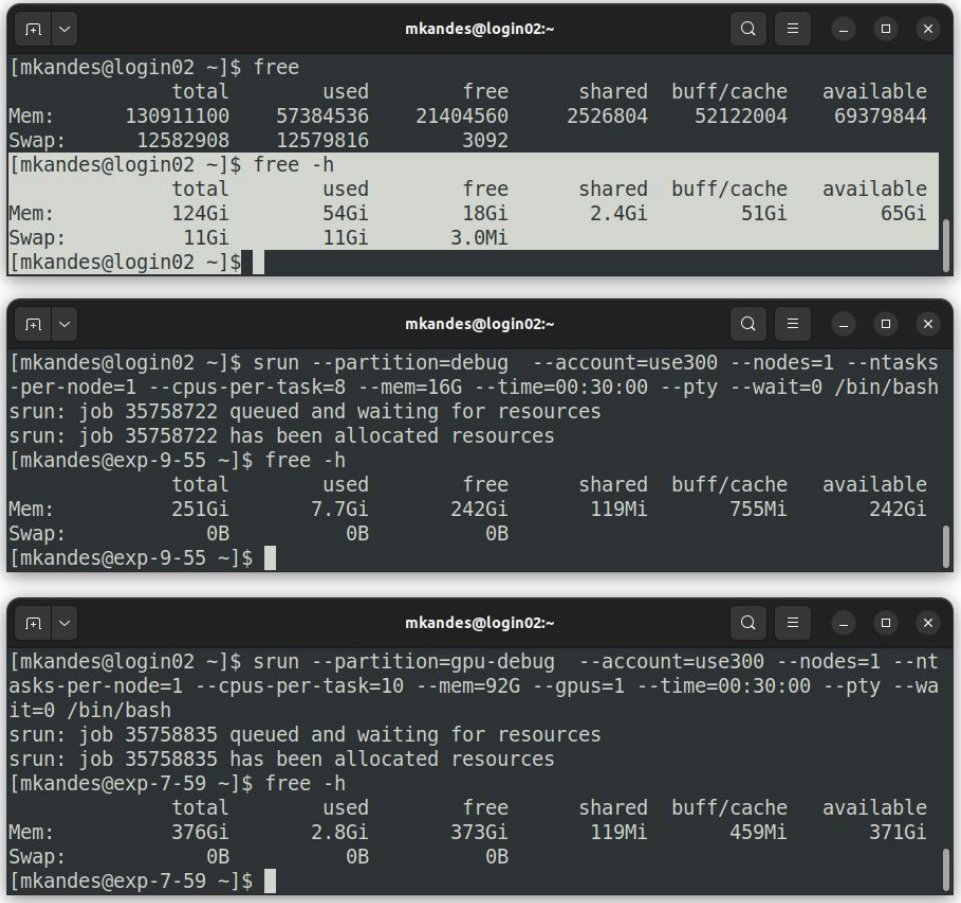
Many applications can control how much data they store in memory. However, this is often because the data is larger than the memory available on a compute node — e.g., see out-of-core algorithms.



Memory is also where your *fastest filesystem* resides – e.g., see RAM disk.

free - display amount of memory in the system

free is a UNIX command which displays the total amount of free and used physical and swap memory in the system, as well as the shared memory and buffers used by the kernel.



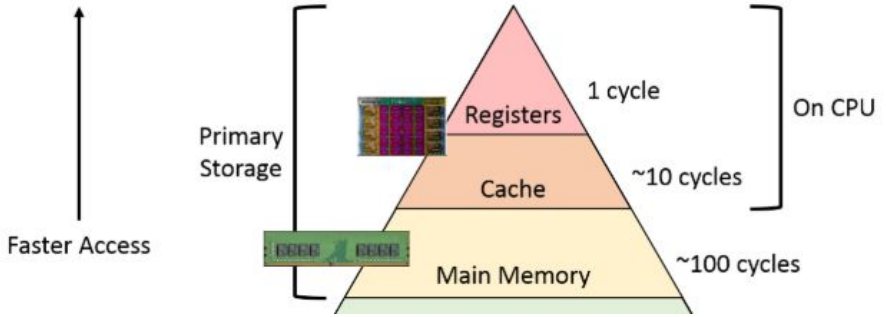
A few things to remember on HPC systems:

- Different node types will have different amounts of memory.
- The **free** memory shown is likely not the amount available for your batch job, unless it has exclusive access to the compute node.
- Only CPU memory availability and usage is reported. Other compute devices like GPUs have their own on-board memory hierarchy, and their own command-line tools to query this type of information. e.g., `nvidia-smi`

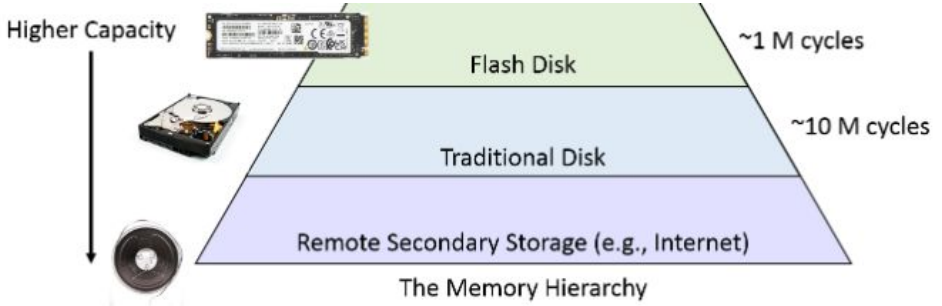
Volatile vs. Non-Volatile Memory (or Storage)

There are many different types of memory and storage devices. However, there are two general classes of storage devices you will interact with on any computing system.

Volatile memory **requires power** to maintain stored data; it retains its contents while powered on *but when the power is interrupted, the stored data is quickly lost*. It is used as primary storage.

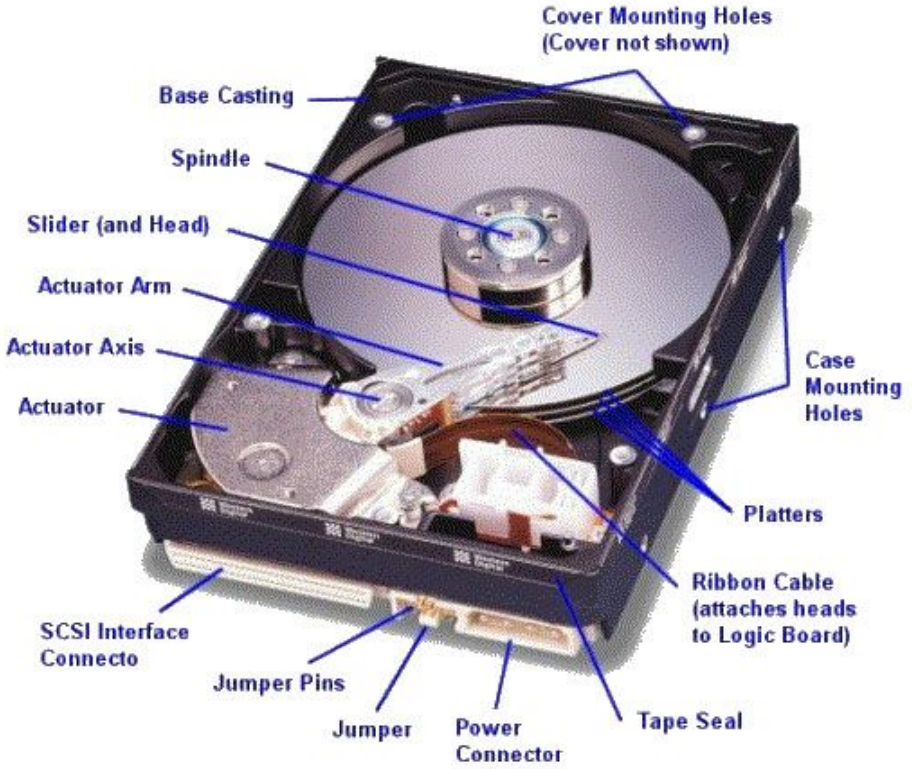


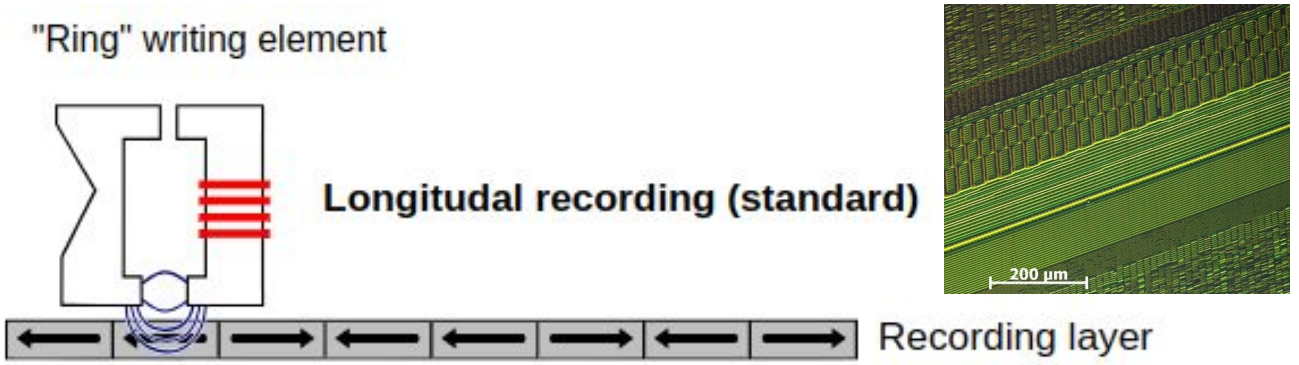
Non-volatile memory (or storage) can retain stored data even **after power is removed**. It is used for secondary (or long-term persistent) storage.



Hard Disk Drives (HDDs)

A hard disk drive (HDD) is an electro-mechanical data storage device that stores and retrieves digital data using magnetic storage with one or more rigid rapidly rotating platters coated with magnetic material.





While HDDs are not used as node-local storage on most HPC compute nodes these days, they are still widely deployed as high-capacity storage devices as part of a distributed filesystem. i.e., the performance of a distributed filesystem may also be limited due to the mechanical properties of the type of disks in use.

Solid State Drives (SSDs)

A solid-state drive (SSD) is a type of solid-state storage device that uses only integrated circuits to store data persistently. *No moving parts!*

Like your personal computer, SSDs are used for node-local storage on most HPC compute nodes these days. e.g., at least one SSD in every node will host the files required to run its operating system.

The next most important use case is to serve as a node-local **`/scratch`** disk, where you can temporarily read in and/or write files to while your batch job is running.

Also used to create a fast tier of storage in a multi-tier distributed file system.

Magnetic Tape

Magnetic tape is a medium for magnetic storage made of a thin, magnetizable coating on a long, narrow strip of plastic film. It is still widely used as a part of modern data storage systems. However, they are used mostly for archival and backup use cases.

See Ranch at TACC!

Table of Contents

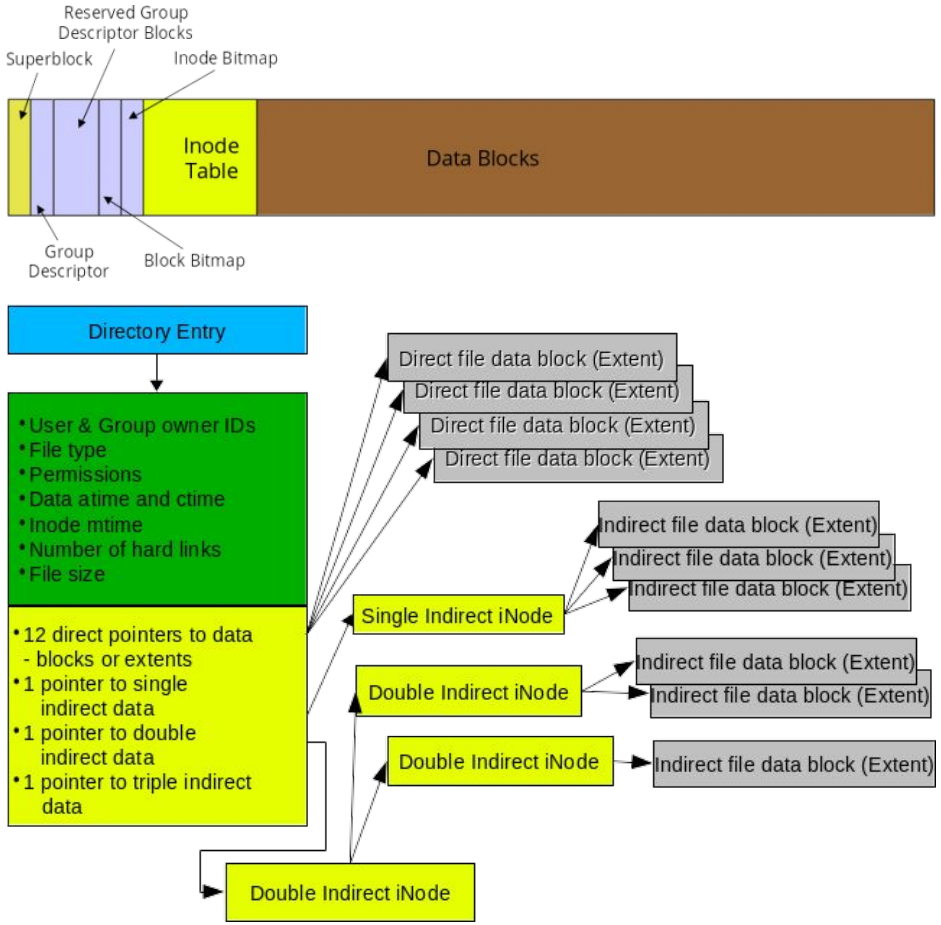
Elements of a POSIX filesystem

POSIX filesystems are organized as directories containing files. However, there are also more fundamental elements in POSIX compliant filesystems like inodes and data blocks.

All filesystems support a set of standard operations that can be performed on a file:

- Create a new file
- Change the access permissions and/or attributes of a file
- Open a file
- Read data from a file
- Write data to a file
- Delete a file
- Close a file

All of the filesystems you will likely interact with on an HPC system are POSIX-based.



Block

A block, sometimes also called a *physical record,* is a sequence of bytes or bits, usually containing some whole number of records, having a maximum length; a *block size*.

POSIX filesystems are treated as a sequence of bytes. As such, internally the data stored within a file is a logical sequence of filesystem blocks.

- Each block is a fixed number of bytes (usually 4 KiB). Last block(s) need not be filled.
- All the bytes are sequential within a block. However, the blocks might not reside sequentially on the disk within a file.
- Underlying storage systems are usually organized as blocks, and ideally filesystem blocks are aligned with the storage system's blocks.

Inode

An inode (or index node) is a data structure in a POSIX filesystem that describes a filesystem object such as a file or a directory. Each inode stores the attributes and disk block *locations* of the object's data. It provides the metadata about a filesystem object.

Inodes provide metadata such as:

- a timestamp of when the *inode* was last modified (ctime)
- a timestamp of when the *file's contents* were last modified (mtime)
- a timestamp of when file was last accessed (atime)
- the numeric user id (UID) of the user that is the owner of the data
- the numeric group id (GID) of the group that is the owner of the backed data
- the POSIX mode bits (often called permissions or permission bits)

Directory

A directory is a special filesystem object that which contains references to files, and other directories. Each POSIX filesystem may have its own implementation of a directory.

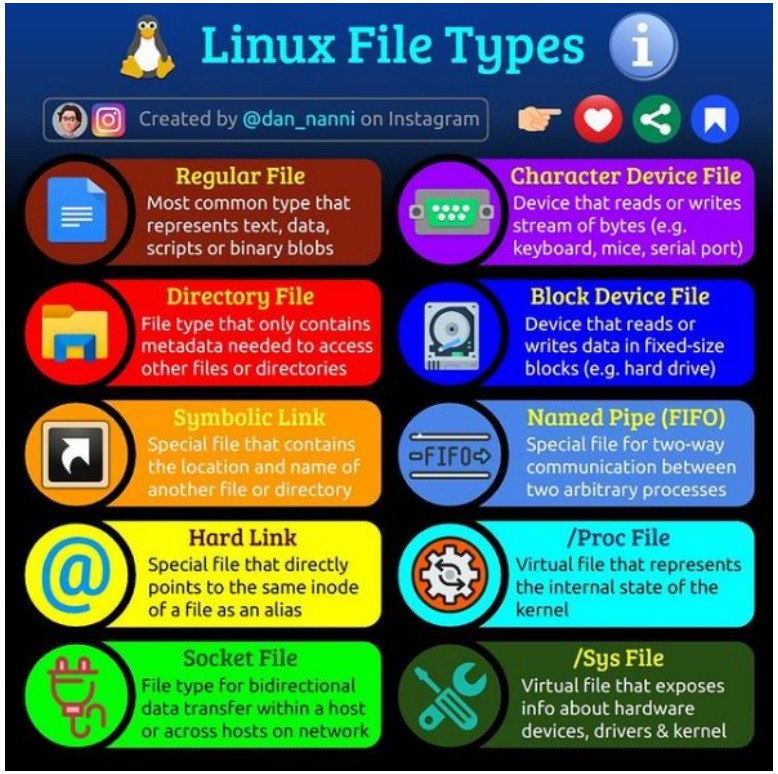
However, common among all filesystems are:

- each directory should at least have a logical name along with the number of the inodes that contains the metadata backing up the entry
- the name can be any string made of any characters with the exception of "/" character and of the "null" character; the length of the string is limited, but the actual length depends the filesystem implementation
- the name is the primary key – no two entries can have the same path

File

A file is a set of blocks used to store data.

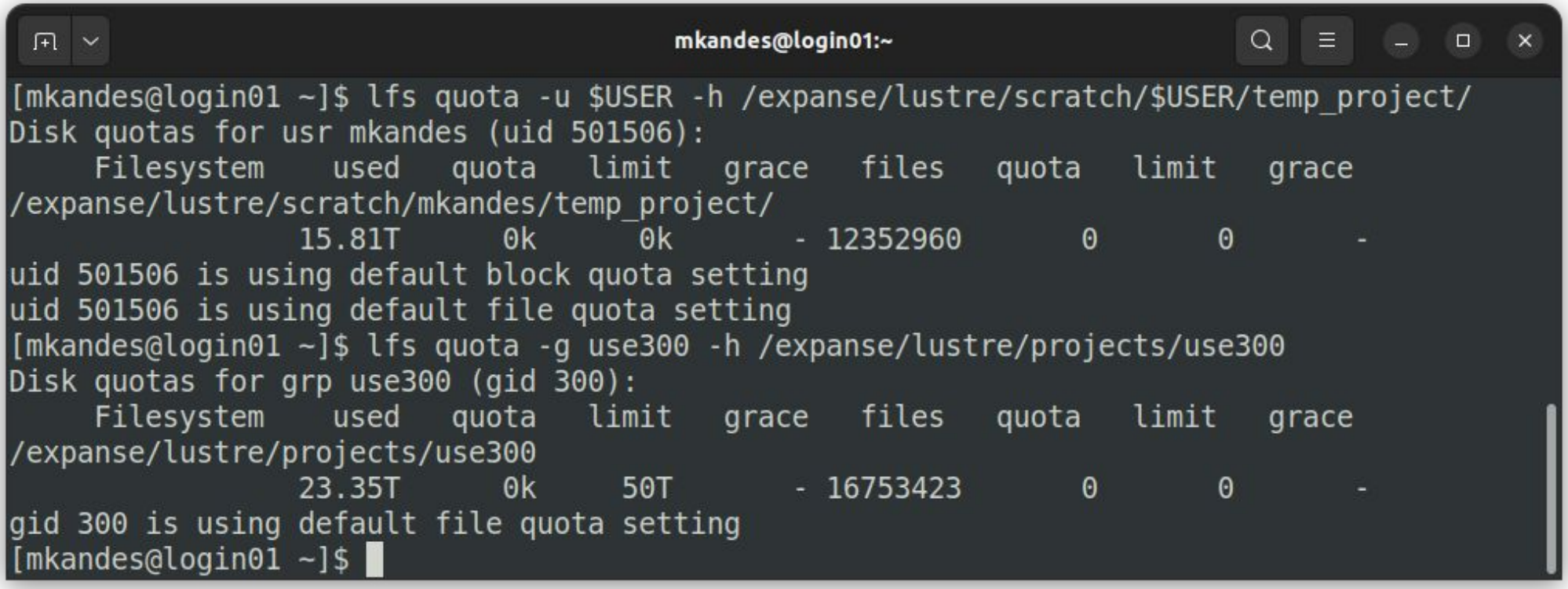
- Files do not have a name.
- They are a set of blocks that can be either directly referenced from the inode or indirectly referenced from another data block.
- The name of the file is instead stored inside a directory as a directory entry.



Everything is a file!

lfs – client utility for Lustre filesystems

`lfs` is a command-line utility for Lustre filesystems. It can be used to create a new file with a specific striping pattern, determine the default striping pattern, gather the extended attributes (object numbers and location) for a specific file. You can also use it to check your quota(s).



lfs – client utility for Lustre filesystems

`lfs` is a command-line utility for Lustre filesystems. It can be used to create a new file with a specific striping pattern, determine the default striping pattern, gather the extended attributes (object numbers and location) for a specific file. You can also use it to check your quota(s).

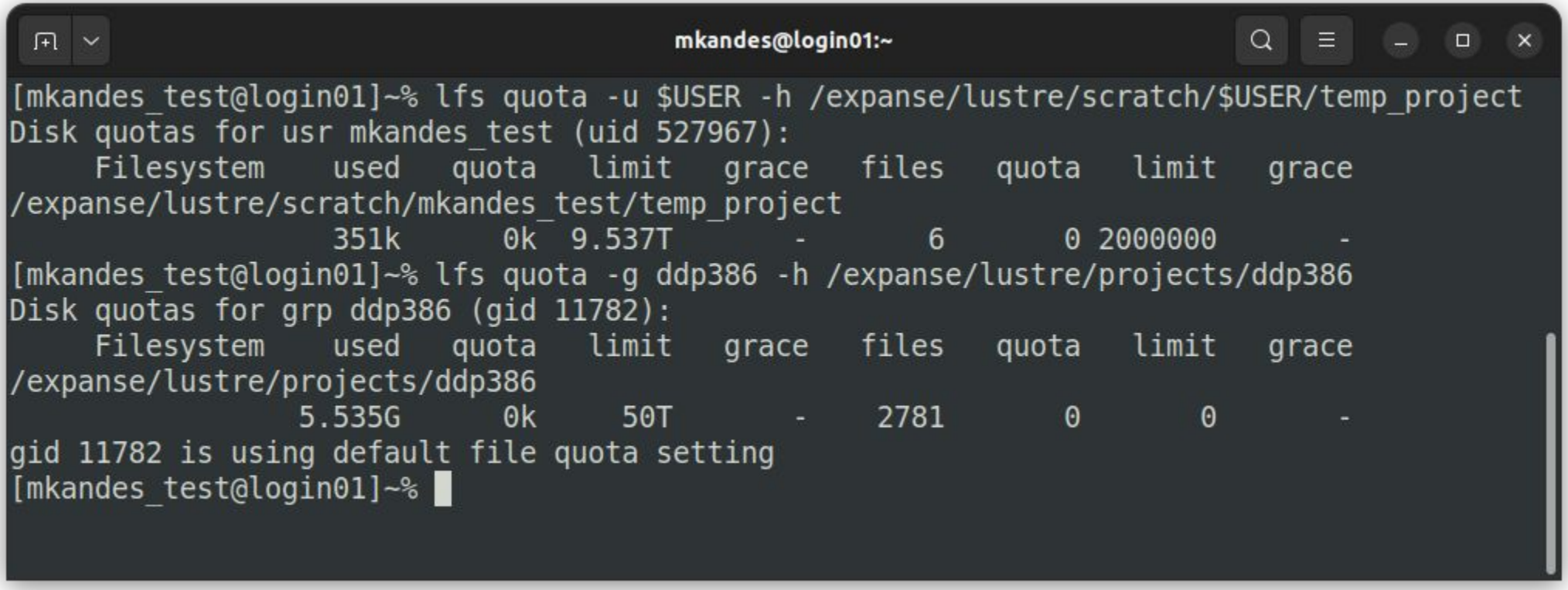
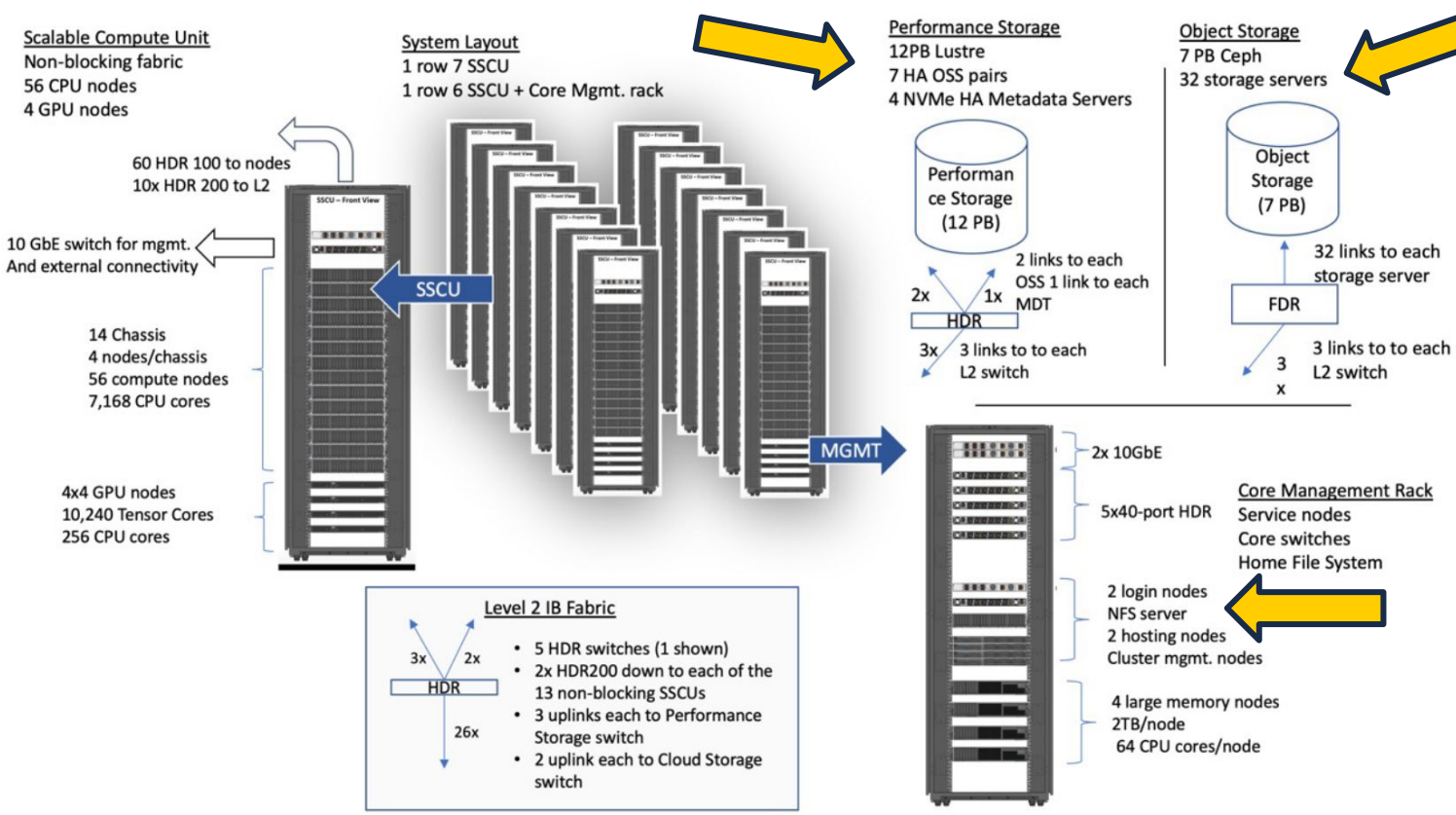
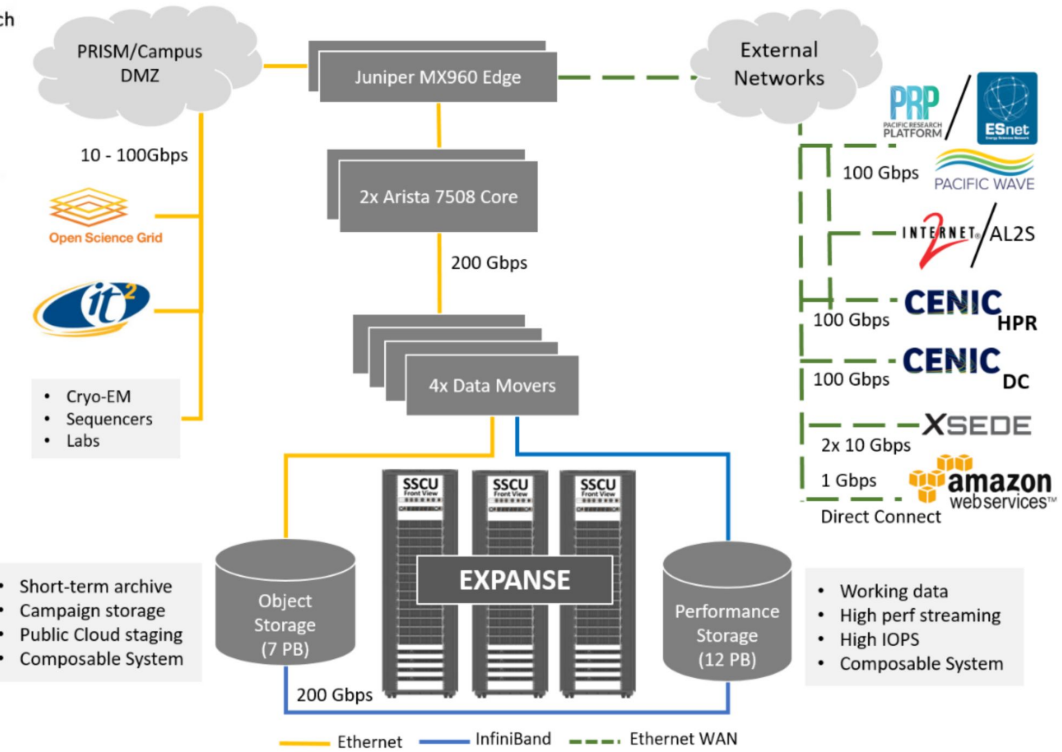


Table of Contents

HPC System Architecture: Expanse @ SDSC





https://access-ci.org

https://expanse.sdsc.edu

https://www.youtube.com/watch?v=uNZyg6X_t3s

HPC System Architecture: Conceptual Model

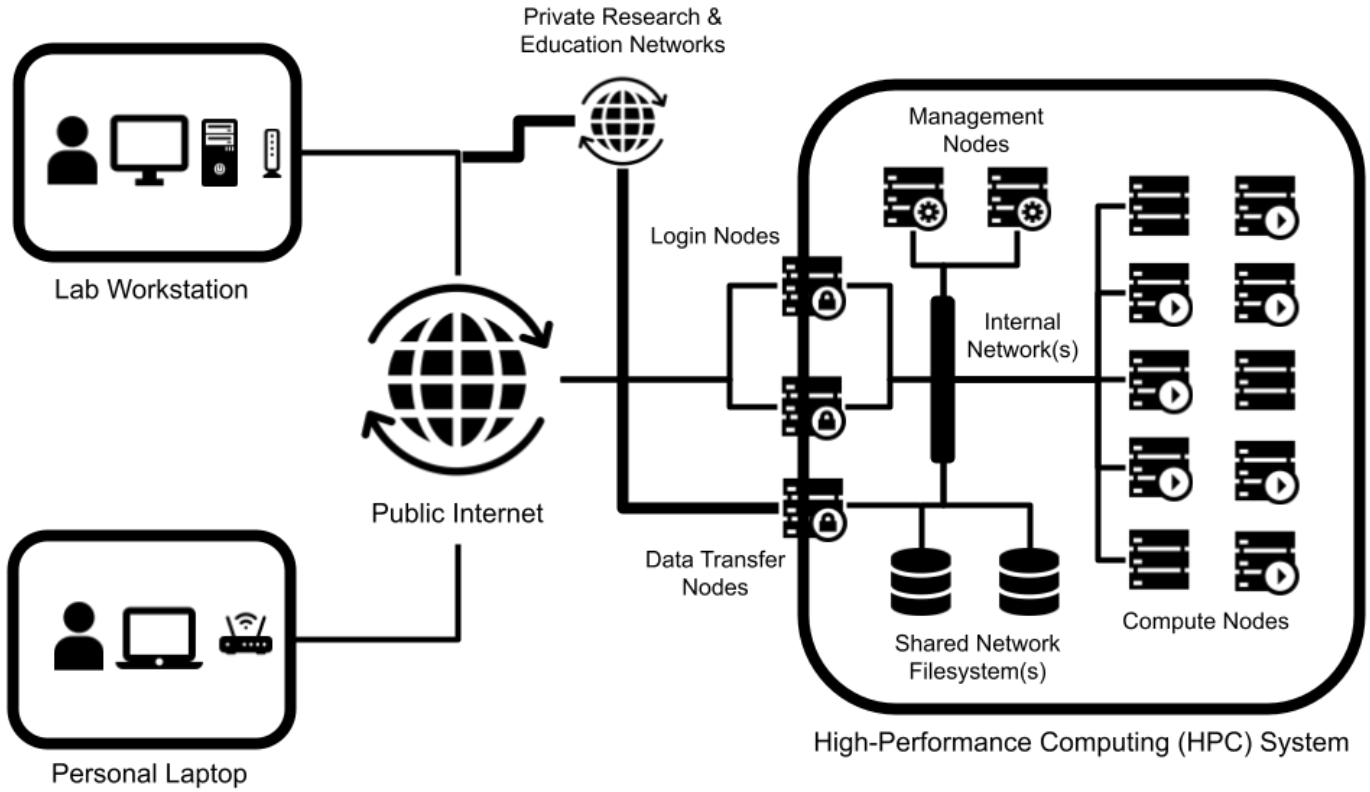
Login node(s): Provide remote access to an HPC system; use only for simple tasks such as editing files, limited data transfers to and from the system, and batch job submission

Compute nodes: Run computational workloads: simulations, data analysis and visualization

Internal Network(s): Provide high-bandwidth, low-latency communication between computes nodes ; access to shared (parallel) filesystems; system management

Shared Network Filesystem(s): Provide input/output (I/O) access to data storage systems from any compute node

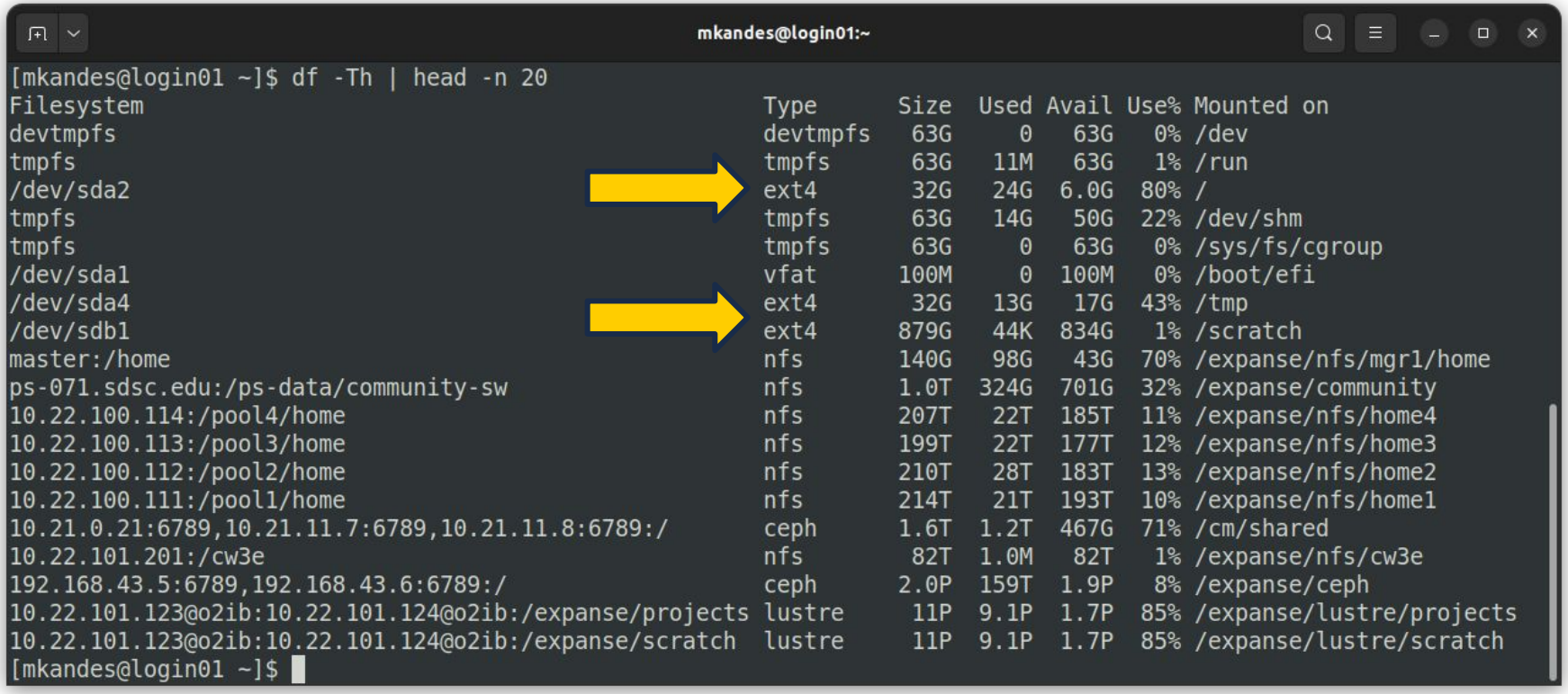
Data Transfer Node(s) : Deployed and configured specifically for transferring data over networks, usually between HPC data storage systems



Management node(s): Run core system services such as cluster management software, system monitoring software, *batch job scheduler*, etc

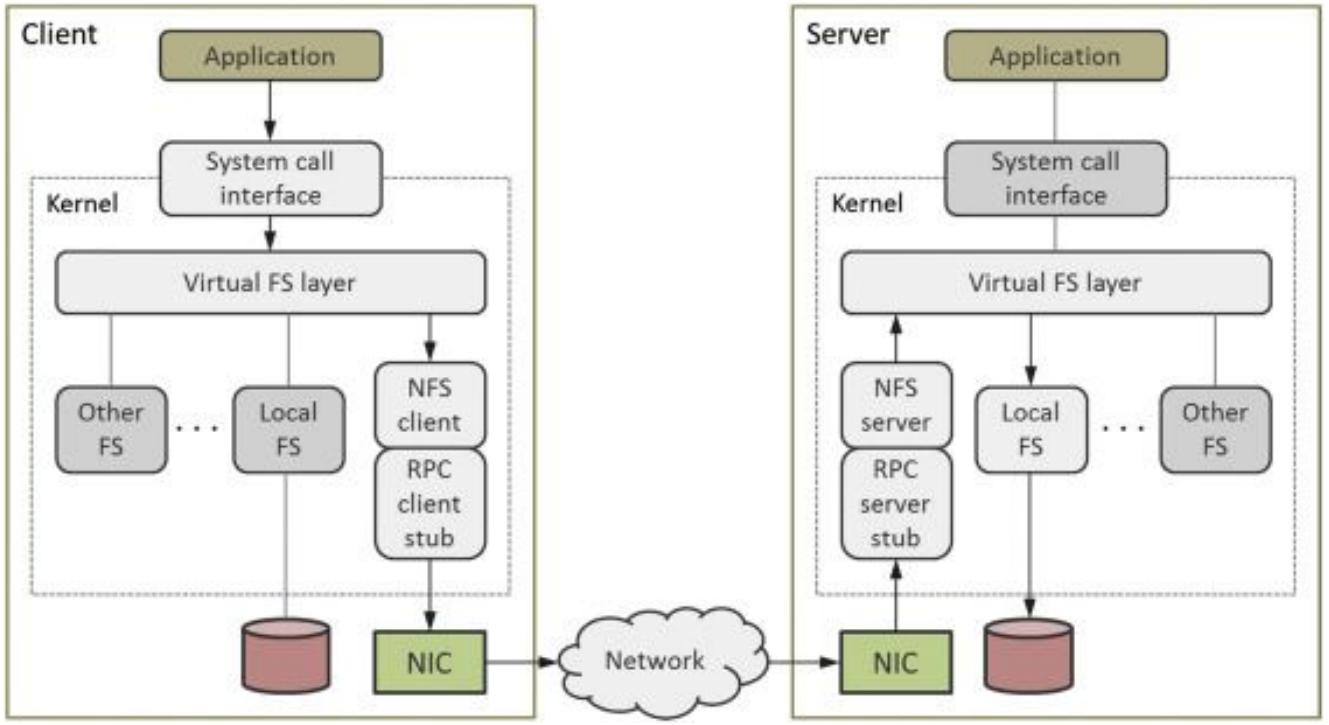
ext4

ext4 is the default *local filesystem* for most Linux operating system distributions. It is a journaling file system that can support large storage volumes — up to 1 exbibyte (EiB) — and large single file sizes — up to 16 tebibytes (TiB).



Network File System (NFS)

The **Network File System (NFS)** is one of the most popular *distributed filesystem* protocols in use on HPC systems. It allows a user on a client computer to access files stored on a remote server over a network.

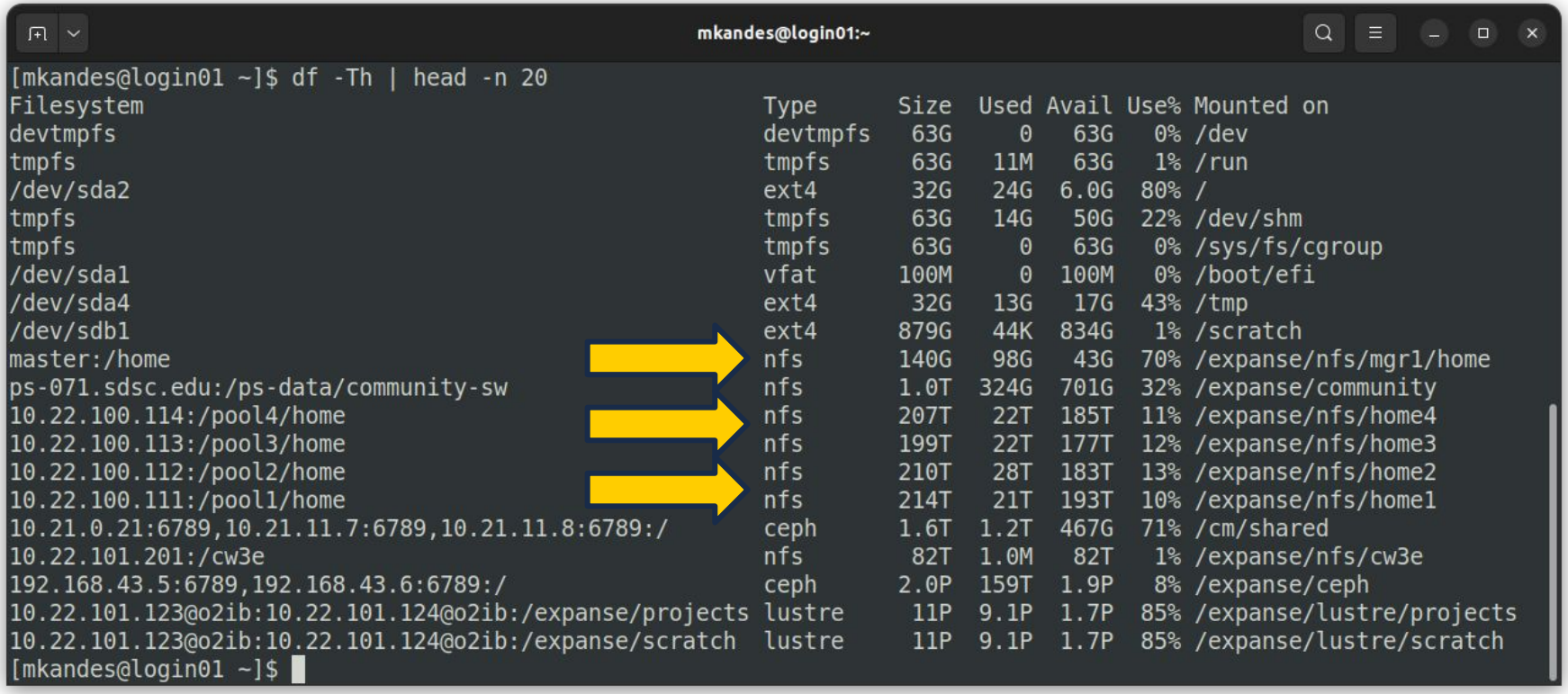


NFS is often used to mount your **`$HOME`** directory system-wide across all nodes of an HPC system.

In this use case, the total amount of storage is typically quite limited and intended only for software installation, batch job scripts, and essential input/output files.

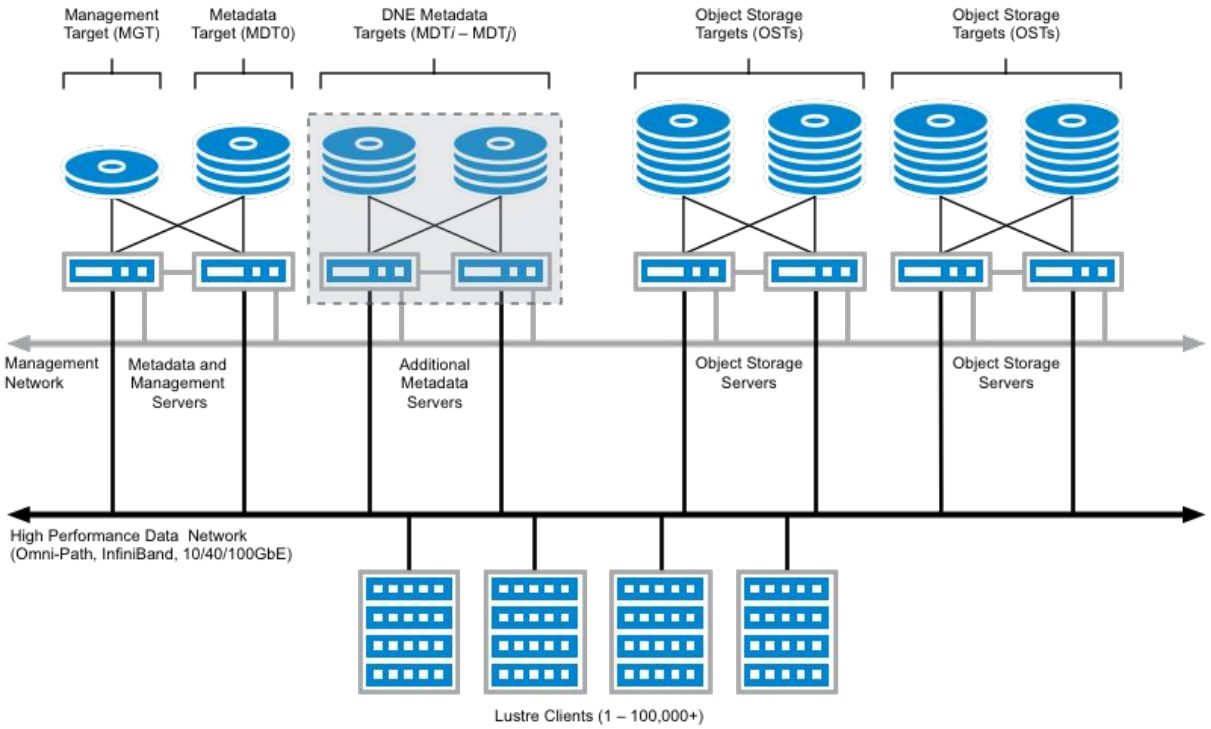
Network File System (NFS)

The **Network File System (NFS)** is one of the most widely used distributed file system protocols in use on HPC systems. It allows a user on a client computer to access files stored on a remote server over a network.



Lustre

Lustre is a *parallel distributed filesystem* commonly used in HPC. It is highly-scalable and can support up to hundreds of thousands of clients, hundreds of petabytes (PB) of storage across hundreds of storage servers, and tens of terabytes per second (TB/s) of aggregate I/O throughput.



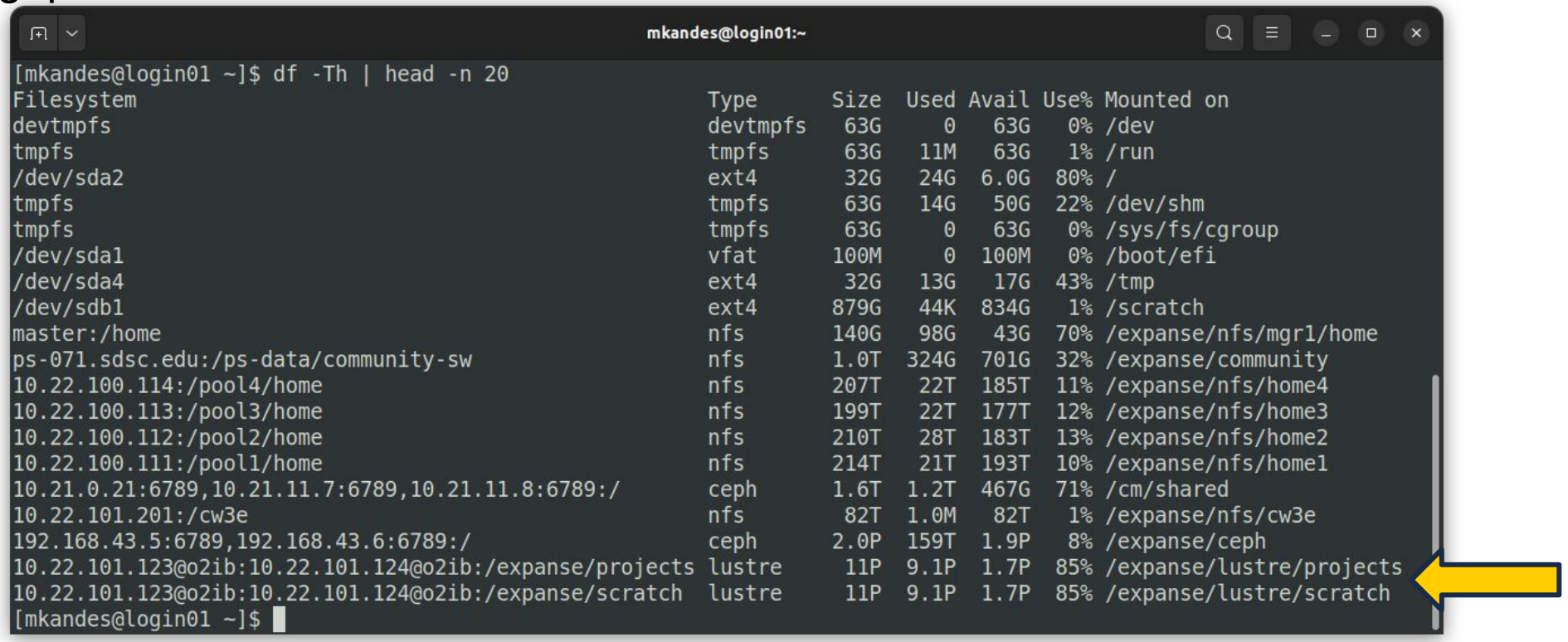
Lustre is typically used as a high-performance scratch filesystem, where you can read and write large amounts of data (in parallel; e.g., MPI I/O).

Lustre was not designed for frequent, large bursts of metadata operations. **e.g., reading or writing too many small files too quickly or too often can cause problems filesystem-wide**

Lustre may be a common target source and/or destination file system for your data transfers.

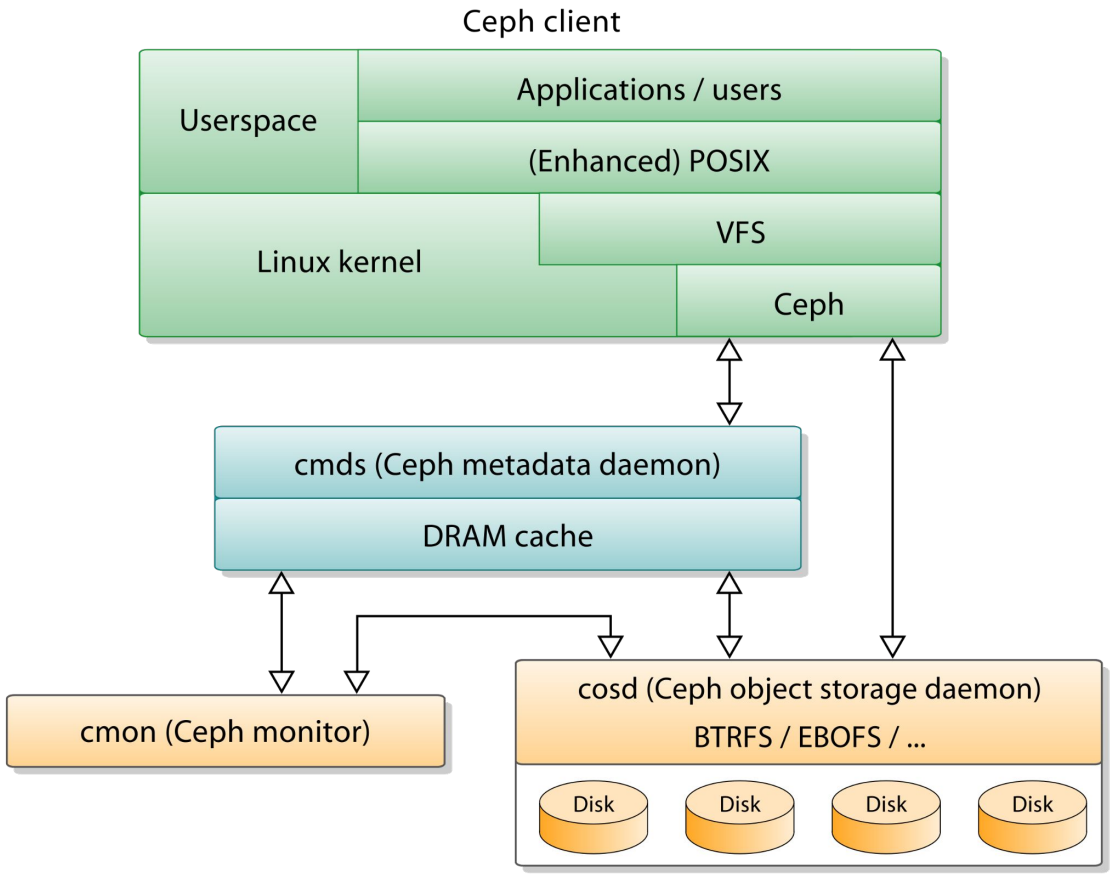
Lustre

Lustre is a *parallel distributed filesystem* commonly used in HPC. It is highly-scalable and can support up to hundreds of thousands of clients, hundreds of petabytes (PB) of storage across hundreds of storage servers, and tens of terabytes per second (TB/s) of aggregate I/O throughput.



Ceph

Ceph is a distributed storage system that provides block, file, and object-based storage.



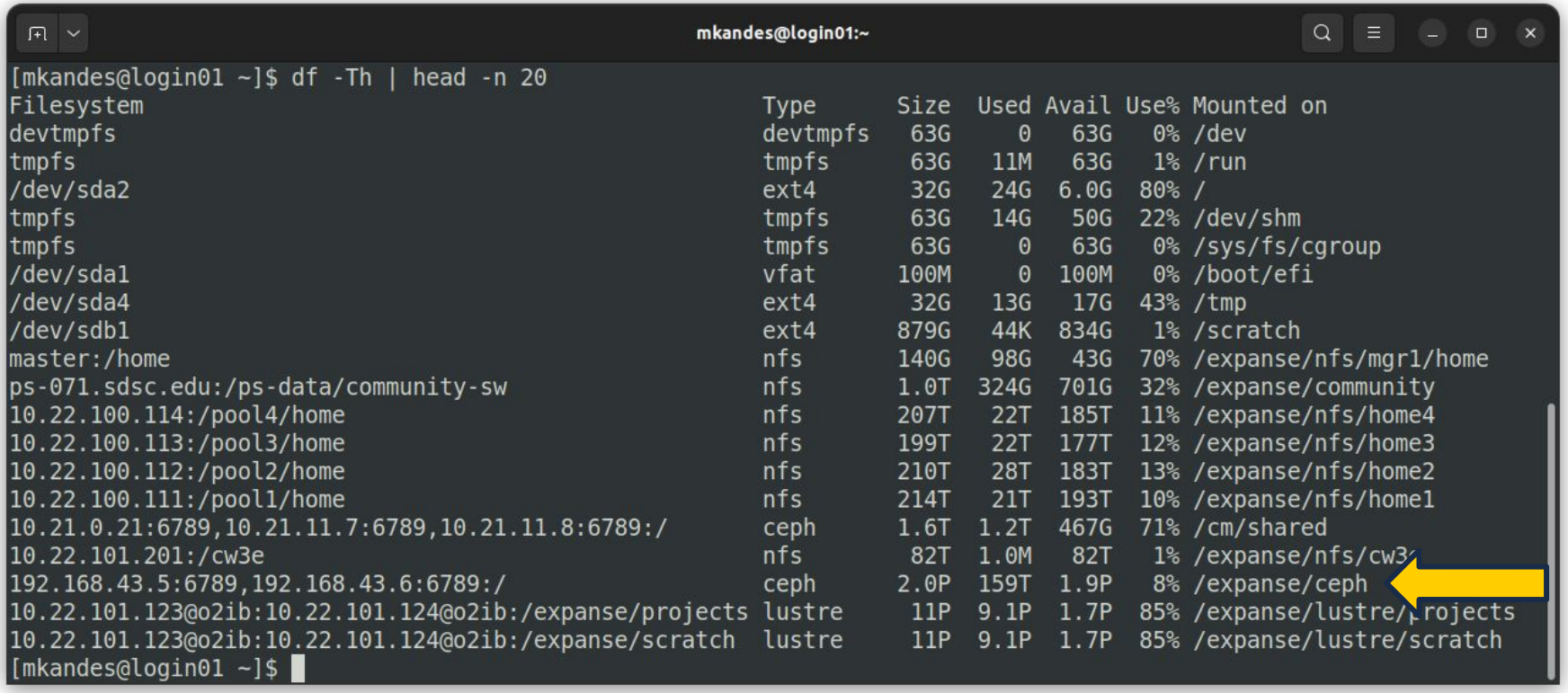
Ceph is typically used as longer-term, large-scale project storage on an HPC system.

Ceph excels at data replication with fault tolerance and high-availability, which in turn helps provide strong data durability.

Ceph may be a more common target source and/or destination storage system for your data transfers in the future.

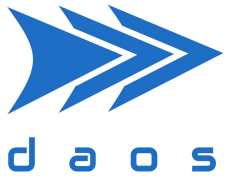
Ceph

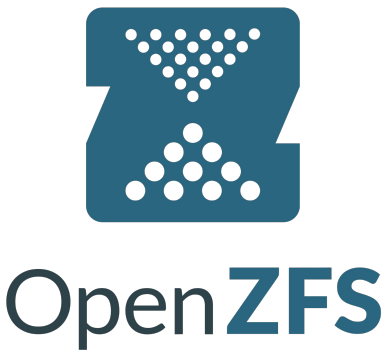
Ceph is a distributed storage system that provides block, file, and object-based storage.



Other Data Storage and File Systems

- BeeGFS
- DAOS
- IBM Spectrum Scale (GPFS)
- OrangeFS
- Qumulo
- VAST
- WEKA
- ZFS

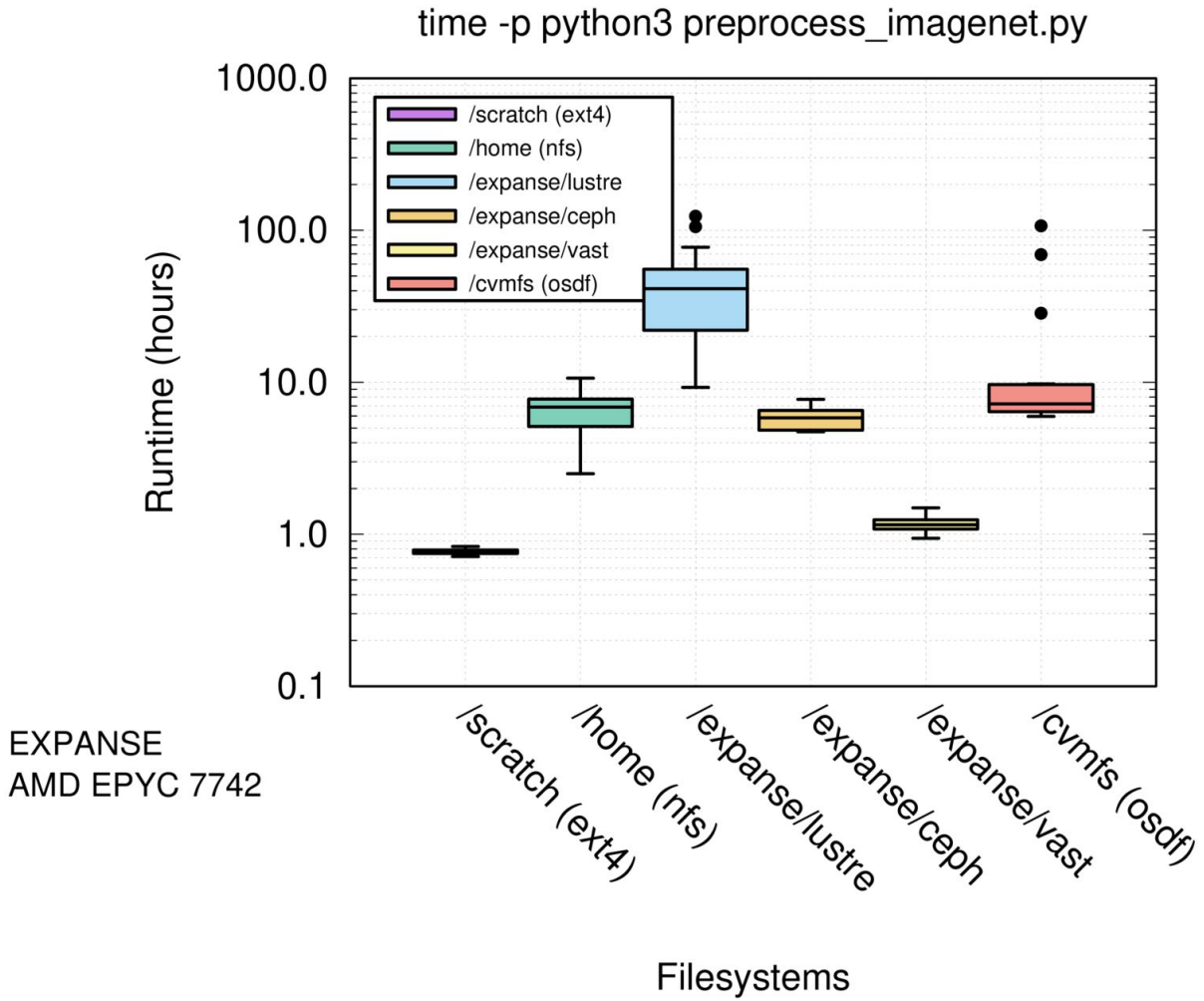




https://io500.org

Table of Contents

Transforming ImageNet to TFRecords



https://github.com/sdsc-complecs/data-management/blob/main/run-preprocess-imagenet-tensorflow-2.8.3-ubuntu-20.04-cuda-11.2-mlnx-ofed-4.9-4.1.7.0-openmpi-4.1.3-20221008-lustre-scratch.sh

Table of Contents

Summary and Conclusion: Best practices

- Understand what types of filesystems are available to you on an HPC system
- Think about how reading and writing (and packaging) your data may affect overall performance of your workload when using any given filesystem
- Avoid small file reads and writes, if possible.
- Keep reads and writes as infrequent as possible
- Always try and use node-local /scratch disks when available.
- Don’t have too many batch jobs accessing the same files simultaneously; make local copies on node-local /scratch instead
- Keep the number of files (and directories) in each directory reasonable (< 10K)
- **Do not colorize your SHELL!** Do you want more metadata lookups!?
- **Always backup important data to your own storage; HPC file systems don’t typically have a backup of your data**

Table of Contents

- What is a filesystem?
- Data storage devices and systems
- Files, Directories, and Inodes
- Filesystems in High-Performance Computing
- Filesystem performance: An AI example
- Summary and Conclusion: Best practices
- Questions & Answers (30 min)

Additional Information

- Data Managment and File Systems on Expanse
- Using the NIH HPC Storage Systems Effectively

Questions & Answers