High Performance
Computing &
Big Data Services

hpc.uni.lu

hpc@uni.lu

@ULHPC

LUXEMBOURG
LET'S MAKE IT HAPPEN

Uni.lu HPC School 2020

Keynote: User environment and data management

Uni.lu High Performance Computing (HPC) Team

S. Peter

University of Luxembourg (UL), Luxembourg

http://hpc.uni.lu

Latest versions available on Github:

UL HPC tutorials: https://github.com/ULHPC/tutorials

UL HPC School: http://hpc.uni.lu/hpc-school/

Keynote tutorial sources: ulhpc-tutorials.rtfd.io/en/latest/

Summary

Motivation

- HPC = high performance **computing**

Motivation

- HPC = high performance **computing**
- But computing needs **input** and produces **output**
- Both input and output might be very valuable

Motivation

- HPC = high performance **computing**
- But computing needs **input** and produces **output**
- Both input and output might be very valuable
- There is also **code** that describes how to get from input to output

Motivation

- HPC = high performance **computing**
- But computing needs **input** and produces **output**
- Both input and output might be very valuable
- There is also **code** that describes how to get from input to output
- Data and code need to be kept save and protected from unauthorized access

Motivation

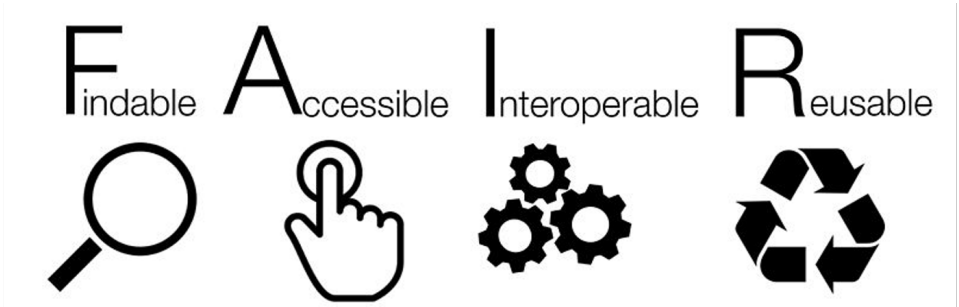
- HPC = high performance **computing**
- But computing needs **input** and produces **output**
- Both input and output might be very valuable
- There is also **code** that describes how to get from input to output
- Data and code need to be kept save and protected from unauthorized access
- Additional requirement: **reproducibility**

Data management definition

"Research data management (RDM) concerns the organisation of data, from its entry to the research cycle through to the dissemination and archiving of valuable results. It aims to ensure reliable verification of results, and permits new and innovative research built on existing information."

Whyte, A., Tedds, J. (2011). 'Making the Case for Research Data Management'. DCC Briefing Papers.

FAIR data



Wilkinson M, Dumontier M et al. Nature Scientific Data 2016. "The FAIR Guiding Principles for scientific data management and stewardship"

Questions

- Data storage:
  - ↪ Where can I store my data on the HPC?
  - ↪ What are the differences between the storage options?
  - ↪ How safe is my data?
  - ↪ What restrictions do I have in terms of storage?

Questions

- Data storage:
  - ↪ Where can I store my data on the HPC?
  - ↪ What are the differences between the storage options?
  - ↪ How safe is my data?
  - ↪ What restrictions do I have in terms of storage?
- Data organisation/management:
  - ↪ How can I keep my files organised?
  - ↪ How can I keep my code organised?

Questions

- Data storage:
  - ↪ Where can I store my data on the HPC?
  - ↪ What are the differences between the storage options?
  - ↪ How safe is my data?
  - ↪ What restrictions do I have in terms of storage?
- Data organisation/management:
  - ↪ How can I keep my files organised?
  - ↪ How can I keep my code organised?
- Data security:
  - ↪ How should I deal with sensitive/personal data on the HPC?
  - ↪ How can I protect my data from unauthorized access?
  - ↪ How can I encrypt data on the HPC?
  - ↪ What passwords should I use and how should I store them?

Session objectives

- Provide an overview of the **storage** options at UL HPC.
- Explain **quotas**, how to check them and what the default settings are.
- Provide information on what data is backed up, where and how long **backups** are retained.
- Give guidance on **data management**, file organisation and naming.
- Briefly introduce **version control** with git and it's benefits.
- Provide information on where **personal or sensitive data** is handled on the UL HPC and give recommendations how it can be secured.
- Introduce different file **encryption** options available on the UL HPC clusters.
- Explain which **passwords** are secure and how they can be stored.

Summary

[Big]Data Management: FS Summary

- **File System (FS)**: Logical manner to *store, organize & access* data
  - ↪ (local) **Disk FS** : `FAT32`, `NTFS`, `HFS+`, `ext4`, `{x,z,btr}fs...`
  - ↪ **Networked FS**: `NFS`, `CIFS/SMB`, `AFP`
  - ↪ **Parallel/Distributed FS**: `SpectrumScale/GPFS`, `Lustre`
    - ✓ typical FS for HPC / HTC (High Throughput Computing)

[Big]Data Management: FS Summary

- **File System (FS)**: Logical manner to *store, organize & access* data
  - ↪ (local) **Disk FS** : `FAT32`, `NTFS`, `HFS+`, `ext4`, `{x,z,btr}fs...`
  - ↪ **Networked FS**: `NFS`, `CIFS/SMB`, `AFP`
  - ↪ **Parallel/Distributed FS**: `SpectrumScale/GPFS`, `Lustre`
    - ✓ typical FS for HPC / HTC (High Throughput Computing)

Main Characteristic of Parallel/Distributed File Systems

Capacity and Performance increase with #servers

[Big]Data Management: FS Summary

- **File System (FS)**: Logical manner to *store, organize & access* data
  - ↪ (local) **Disk FS** : `FAT32`, `NTFS`, `HFS+`, `ext4`, `{x,z,btr}fs...`
  - ↪ **Networked FS**: `NFS`, `CIFS/SMB`, `AFP`
  - ↪ **Parallel/Distributed FS**: `SpectrumScale/GPFS`, `Lustre`
    - ✓ typical FS for HPC / HTC (High Throughput Computing)

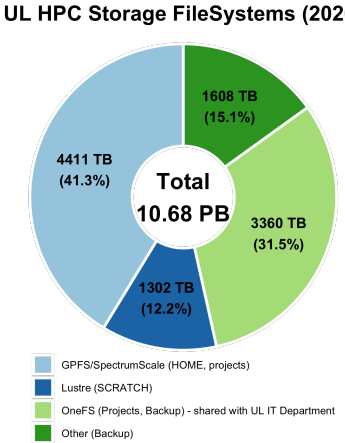
Main Characteristic of Parallel/Distributed File Systems

Capacity and Performance increase with #servers

Name	Type	Read* [GB/s]	Write* [GB/s]
`ext4`	Disk FS	0.426	0.212
`nfs`	Networked FS	0.381	0.090
`gpfs` (iris/aion)	Parallel/Distributed FS	**11.25**	**9,46**
`lustre` (iris/aion)	Parallel/Distributed FS	**12.88**	**10,07**

* maximum **random** read/write, per IOZone or IOR measures, using concurrent nodes for networked FS.

UL HPC Storage Systems



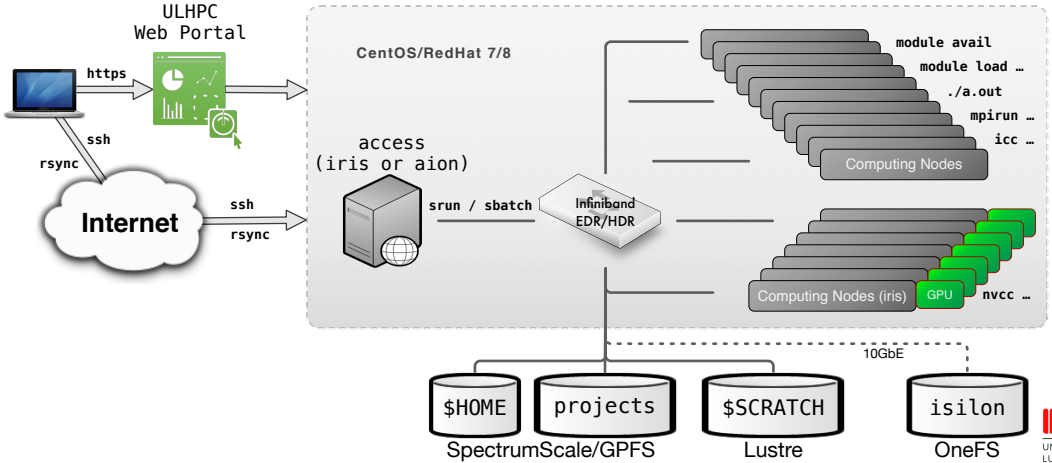
Understanding Your Storage Options

Where can I store and manipulate my data?

- **Shared** storage
  - ↪ `NFS` - **not scalable** $\sim\simeq$ 1.5 GB/s (R) — $\mathcal{O}$(100 TB)
  - ↪ `GPFS` - **scalable** $\sim\sim\simeq$ 10 GB/s (R) — $\mathcal{O}$(1 PB)
  - ↪ `Lustre` - **scalable** $\sim\sim\simeq$ 5 GB/s (R) — $\mathcal{O}$(0.5 PB)
- **Local** storage
  - ↪ local file system (`/tmp`) — $\mathcal{O}$(200 GB)
    - ✓ over HDD $\simeq$ 100 MB/s, over SDD $\simeq$ 400 MB/s
  - ↪ RAM (`/dev/shm`) $\simeq$ 30 GB/s (R) — $\mathcal{O}$(20 GB)
- **Distributed** storage
  - ↪ HDFS, Ceph, GlusterFS - **scalable** $\sim\sim\simeq$ 1 GB/s

$\Rightarrow$ **In all cases:** small I/Os really **kill** storage performances

Compute Nodes Environment



Where is what

Directory	Env variable	Filesystem
`/home/users`	`$HOME`	SpectrumScale
`/work/projects`	-	SpectrumScale
`/scratch/users`	`$SCRATCH`	Lustre
`/mnt/isilon/projects`	-	OneFS

How to use

Directory	Usage
`/home/users`	personal space, software & packages
`/work/projects`	shared project storage
`/scratch/users`	intermediate fast storage, work here
`/mnt/isilon/projects`	archival storage, do **not** use for processing

Checking quota

Check file size quota with

```
df-ulhpc
```

Check inode quota with

```
df-ulhpc -i
```

Checking disk usage

Check free space on all file systems with

```
df -h
```

Check free space on current file system with

```
df -h .
```

To see what directories are using your disk space:

```
ncdu
```

Soft quota, hard quota and grace period

- Once you reach the **soft quota** you can still write data until the **grace period** expires (7 days) or you reach the **hard quota**.
- After you reach the end of the grace period or the hard quota, you have to reduce your usage to below the soft quota to be able to write data again.

Default quotas

Directory	size quota	inode quota
`$HOME`	500 GB	1,000,000
`$SCRATCH`	10 TB	1,000,000
`/work/projects/...`	16 MB	-
`/mnt/isilon/projects/...`	1.14 PB globally	-

Remarks

Isilon quota almost reached

- Global quota for all HPC users.
- Try to clean-up `/mnt/isilon/projects/...`

Quota for `clusterusers` group in project directories is 0

- Make sure the *setgid* bit is set on all folders in the project directories `chmod g+s`
- Transfer files without preserving group `rsync --no-p --no-g`

Backup: Iris

- `$HOME`
  - ↪ daily backup to another server in the same data center
  - ↪ retention: last 7 daily backups, at least one per month for the last 2 months
- `/work/projects`
  - ↪ daily backup to another server in the same data center
  - ↪ retention: one backup per week of the backup directory (`$PROJECT/backup/`) for at least 1 week

Backup: Isilon (HPC share)

- `/mnt/isilon/projects`
- weekly snapshot
- retention: only one snapshot kept
- no true backup, because it's on the same system

Backup: Warning

- **NO** backup in `$SCRATCH` (`/scratch` or `/tmp`) directories
- **Clean-up:** files in `$SCRATCH` older than 60 days are removed every month
- **Clean-up:** files in `/tmp` on compute nodes are removed at the end of the job

Summary

Guidelines for raw data

- Collect and store **metadata**, e.g. in `README` files that accompany data
  - ↪ Minimal metadata describing what is the data and how it got there:
    - ✓ Title
    - ✓ Date of creation/receipt
    - ✓ Data origin
    - ✓ Version of the data
    - ✓ Data owner/responsible
    - ✓ Data structure
    - ✓ How was the data downloaded
- Generate checksums for raw/source data, e.g. with `md5sum`
  - ↪ Before and after data transfer
  - ↪ Before archival

Guidelines for raw data

Make raw data read-only!

File naming

3 main principles

- Machine readable
- Human readable
- Plays well with default ordering

Sources and further reading:

- Jenny Brian's talk on "Naming things" from Reproducible Science Workshop, Duke, 2015
- Kristin Briney's file naming convention worksheet. Caltech Library, 2020

Version control

- "backup" for your code
- keep track of your processing / analysis history
- benefits (from Atlassian):
  - ↪ complete long-term change history of every file
  - ↪ branching and merging
  - ↪ traceability
- important for reproducibility and GDPR compliance

Gitlab.uni.lu

- local GitLab instance hosted by HPC
- data stays within UL
- as many private repositories as you want
- access for external collaborators with Github account

Getting started with git

- Many IDEs have git integration
- **Learning git:**
  - ↪ tutorial: Software Carpentry: Version Control with Git
  - ↪ `https://git-scm.com/`
- **Additional resources**
  - ↪ tutorial: IT/Dev[op]s Army Knives Tools for the Researcher
  - ↪ tutorial: Reproducible Research at the Cloud Era

Summary

GDPR and UL HPC

- EU General Data Protection Regulation (**GDPR**)
  - ↪ replaces the Data Protection Directive 95/46/EC
  - ↪ legislation came into effect May 25th 2018

- **The UL HPC facility handles both**:
  - ↪ **data about people** (facility users identification details)
    - ✓ ULHPC Identity Management (IdM) system
    - ✓ Account request form results
  - ↪ **large scale data** that may contain Personally Identifiable Info
    - ✓ stored by facility users in networked, parallel & distributed filesystems used across the HPC infrastructure
    - ✓ can be considered as falling under GDPR regulations.

GDPR and UL HPC

- **Personal data is/may be visible, accessible or handled**:
  - ↪ directly on the HPC clusters
  - ↪ through *Resource and Job Management System* (RJMS) tools
    - ✓ glue for a parallel computer to execute parallel jobs
    - ✓ **Goal**: satisfy users demands for computation
    - ✓ comes with web interfaces
  - ↪ through service portals
  - ↪ on code management portals
  - ↪ on secondary storage systems

Monika, Ganttchart
Open OnDemand
GitLab, GitHub
DropIT, OwnCloud

Best practices for you

General

- data (pseudo-) anonymisation
- data minimisation
- data partitioning

Best practices for you

General (continued)

- secure laptop
  - ↪ enable FileVault / disk encryption
  - ↪ lock your screen when you leave your place
  - ↪ apply security updates
  - ↪ anti-virus / anti-malware software
  - ↪ (encrypted) backup of your laptop
- secure access credentials
  - ↪ consider using a password manager
  - ↪ use 2FA when possible (authenticator better than SMS)
- **Follow policies of your institute and the UL!**

Best practices for you

On ULHPC

- double-check permissions on your `$HOME` and `$SCRATCH` folders
- secure your SSH key with a passphrase
- empty `/tmp` at the end of the job
- reserve a full node
- store your data on iris (SED)
- mind backups
- encrypt your files

Data encryption

- Basic approach: **GPG**
  - ↪ Encrypt single files
  - ↪ Files need to be completely decrypted for processing
- More convenient: **gocryptfs**
  - ↪ Encrypt all files within a folder
  - ↪ Can be mounted in a *view* folder where you can read and write the unencrypted files
  - ↪ Automatically unmounted upon job termination
- For LCSB: **PetaSuite Protect**
  - ↪ Encrypt and compress genomic data
  - ↪ Encryption keys and access managed centrally
  - ↪ Decryption and decompression on-the-fly using a library that intercepts all FS access
  - ↪ Commercial software
  - ↪ Contact lcsb.software@uni.lu if you would like to use it
- In git repositories: **git-crypt**

GPG encryption

Encrypt a single file (will ask for a passphrase):

```
gpg -c file_to_be_encrypted
```

Decrypt the file:

```
gpg encrypted_file
```

Instead of using a passphrase, you can also encrypt files using an encryption key.

Gocryptfs

```
module load tools/gocryptfs
mkdir dir.crypt dir
gocryptfs -init dir.crypt
gocryptfs dir.crypt dir
echo "Happy secure computing!" > dir/message.txt
fusermount -u dir
```

Details: https://hpc.uni.lu/blog/2018/sensitive-data-encryption-using-gocryptfs/

Warnings

- Encryption keys and passphrases need to be kept safe and protected from unauthorised access.
- Loosing your encryption key means loosing your data.
- Ensure you have an off-site backup of critical data stored on the platform under encryption.
- (Disaster) recovery of encrypted data is not guaranteed to be viable, depending on internal consistency when the recovery snapshot is taken.

Password Management

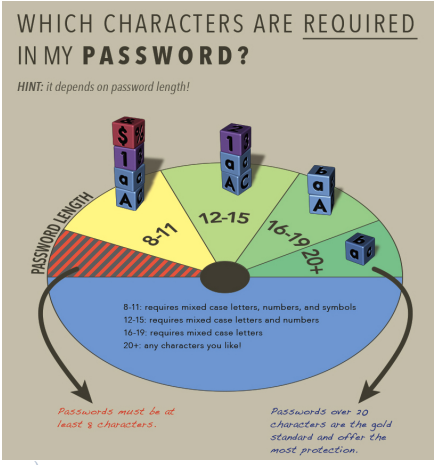
Traditional [Strong] Password policy

- $\geq$ **15** characters, including digits, special chars (#,&,@,$ etc.)
  - ↪ mix upper/lower case
- **avoid** matching dictionary/personal/company/dates info
- renew **periodically**, typically after 180 days.

- Build by selecting words / sentence easy to remember
  - ↪ combine them to respect the above rules

Stanford Password Policy

https://itservices.stanford.edu/service/accounts/passwords/quickguide



Password Manager

Password Manager

- Ensure a safe and **secure** way to store/organize passwords
  - ↪ privilege **random & unique** passwords **everywhere**
  - ↪ ideally: cross-platform applications, with browser integration
- encrypted back-end/vault, eventually shared over Cloud storage
  - ↪ Dropbox, iCLoud, S3, OneDrive...

Password Manager

Password Manager

- Ensure a safe and **secure** way to store/organize passwords
  - ↪ privilege **random & unique** passwords **everywhere**
  - ↪ ideally: cross-platform applications, with browser integration
- encrypted back-end/vault, eventually shared over Cloud storage
  - ↪ Dropbox, iCLoud, S3, OneDrive...

Open-Source / **Cloud-based**

Password Manager

Password Manager

- Ensure a safe and **secure** way to store/organize passwords
  - ↪ privilege **random & unique** passwords **everywhere**
  - ↪ ideally: cross-platform applications, with browser integration
- encrypted back-end/vault, eventually shared over Cloud storage
  - ↪ Dropbox, iCLoud, S3, OneDrive...

Open-Source / Cloud-based

Commercial

1Password

dashlane

GPG+Git Password Management: pass

- pass: the standard Unix password manager
  - ↪ stores passwords as encrypted files – default: ~/.password-store/
  - ↪ cross-platform GUI clients, incl. iOS/Android / Pass4Win
  - ↪ multiple recipient can share a sub-directory
- Installation: { brew | yum | apt-get } install pass

```
$> pass init <ID> && pass git init                    # Create the store over git
```

GPG+Git Password Management: pass

- pass: the standard Unix password manager
  - ↪ stores passwords as encrypted files – default: ~/.password-store/
  - ↪ cross-platform GUI clients, incl. iOS/Android / Pass4Win
  - ↪ multiple recipient can share a sub-directory
- Installation: { brew | yum | apt-get } install pass

```
$> pass init <ID> && pass git init                    # Create the store over git
```

```
$> pass insert <domain>/<name>                        # store <domain>/<name>.gpg
```

GPG+Git Password Management: pass

- pass: the standard Unix password manager
  - ↪ stores passwords as encrypted files – default: ~/.password-store/
  - ↪ cross-platform GUI clients, incl. iOS/Android / Pass4Win
  - ↪ multiple recipient can share a sub-directory
- Installation: { brew | yum | apt-get } install pass

```
$> pass init <ID> && pass git init                    # Create the store over git
```

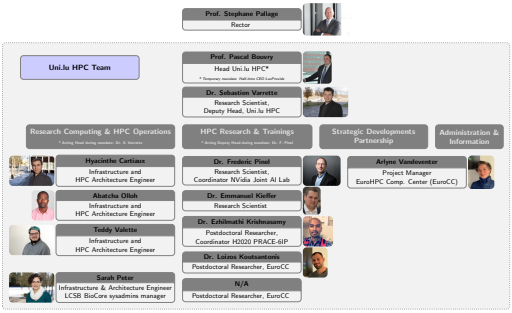
```
$> pass insert <domain>/<name>                        # store <domain>/<name>.gpg
```

```
$> pass [<domain>/<name>]                             # list / retrieve password <name>
```

Thank you for your attention...

Questions?

High Performance Computing @ Uni.lu



University of Luxembourg, Belval Campus
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu

1. **Session Objectives**
2. **Data storage**
   - [Big] Data components in HPC
   - Shared Storage on UL HPC
   - User environment
   - Quotas
   - Backup
3. **Data organisation and management**
   - File organisation and naming
   - Version control with Git
4. **Data security**
   - GDPR and data security
   - Encryption
   - Passwords

https://hpc.uni.lu