DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

STORAGE@TGCC & LUSTRE FILESYSTEMS

WORKING & BEST PRACTICES

Thomas LEIBOVICI | CEA/DAM/DIF

PATC PARALLEL I/O
6-7 MARS 2017

AGENDA

TGCC storage architecture

TGCC storage workspaces

Lustre parallel file system

Hierarchical Storage

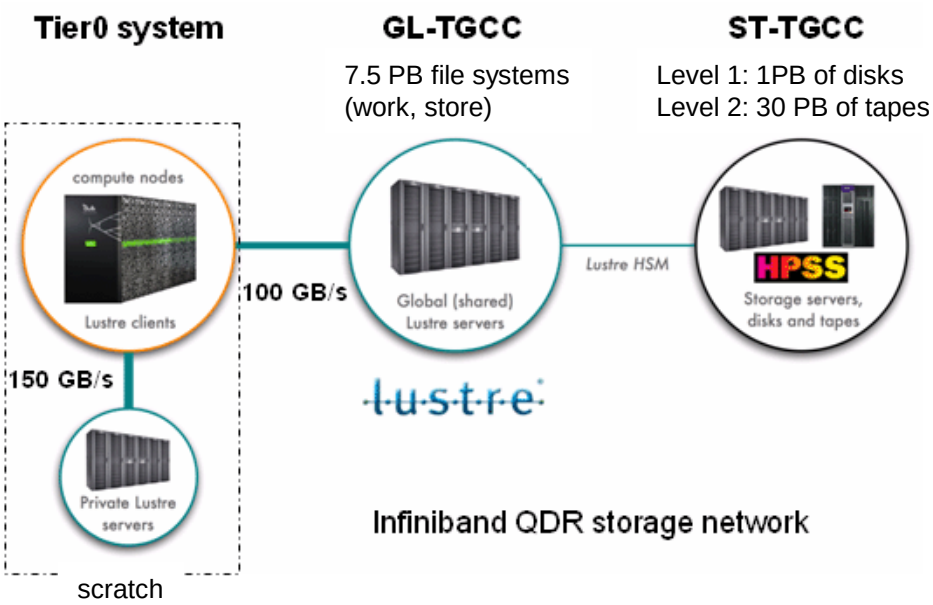
TGCC ARCHITECTURE

GL-TGCC

- Fast access to data,
- Gateway to files
- post-processing

ST-TGCC

- Long term storage
- Keeps simulation results



LUSTRE FILE SYSTEMS @ TGCC (1/2)

scratch

- Workspace for temporary data
- Mount point: /ccc/scratch ($CCCSCRATCHDIR)
- Unused files deleted after 40 days
- **Designed for throughput and performance**

store

- Long term storage: should be used to store final results
- Connected to a HSM (see later slides) for bigger capacity
- Recommended file size : 1GB-100GB
- Quotas : 100k inodes per user, no quota on volume
- Automated migration and staging with the HSM (see later slides)
- Mount point: /ccc/store ($CCCSTOREDIR)
- **Designed for data capacity**

LUSTRE FILE SYSTEMS @ TGCC (2/2)

work

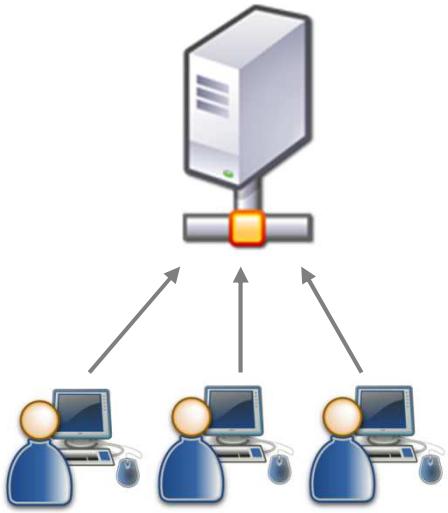
- Permanent workspace (no purge)
- Accessible from all compute clusters
- Quotas : 1TB, 500k inodes per user
- Mount point: /ccc/work ($CCCWORKDIR)

LUSTRE PARALLEL FILE SYSTEM

FROM LOCAL FILESYSTEMS … TO PARALLEL FILESYSTEMS

Local file system
1 disk, 1 client

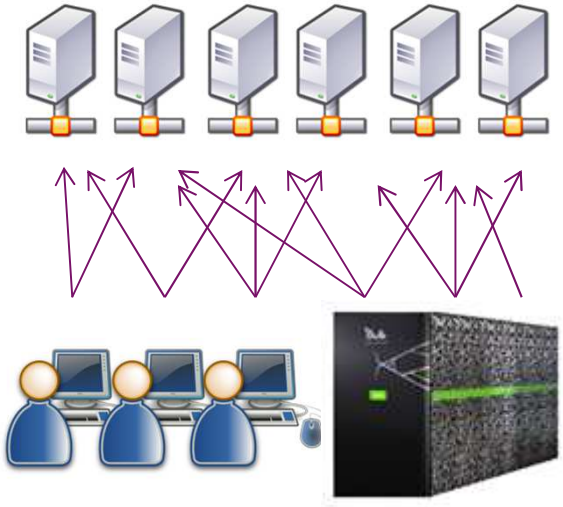
Examples:
- Personnal computer
- /tmp of compute nodes



File server
1 server, N clients

Example:
Login home

Interests:
sharing, access from any workstation

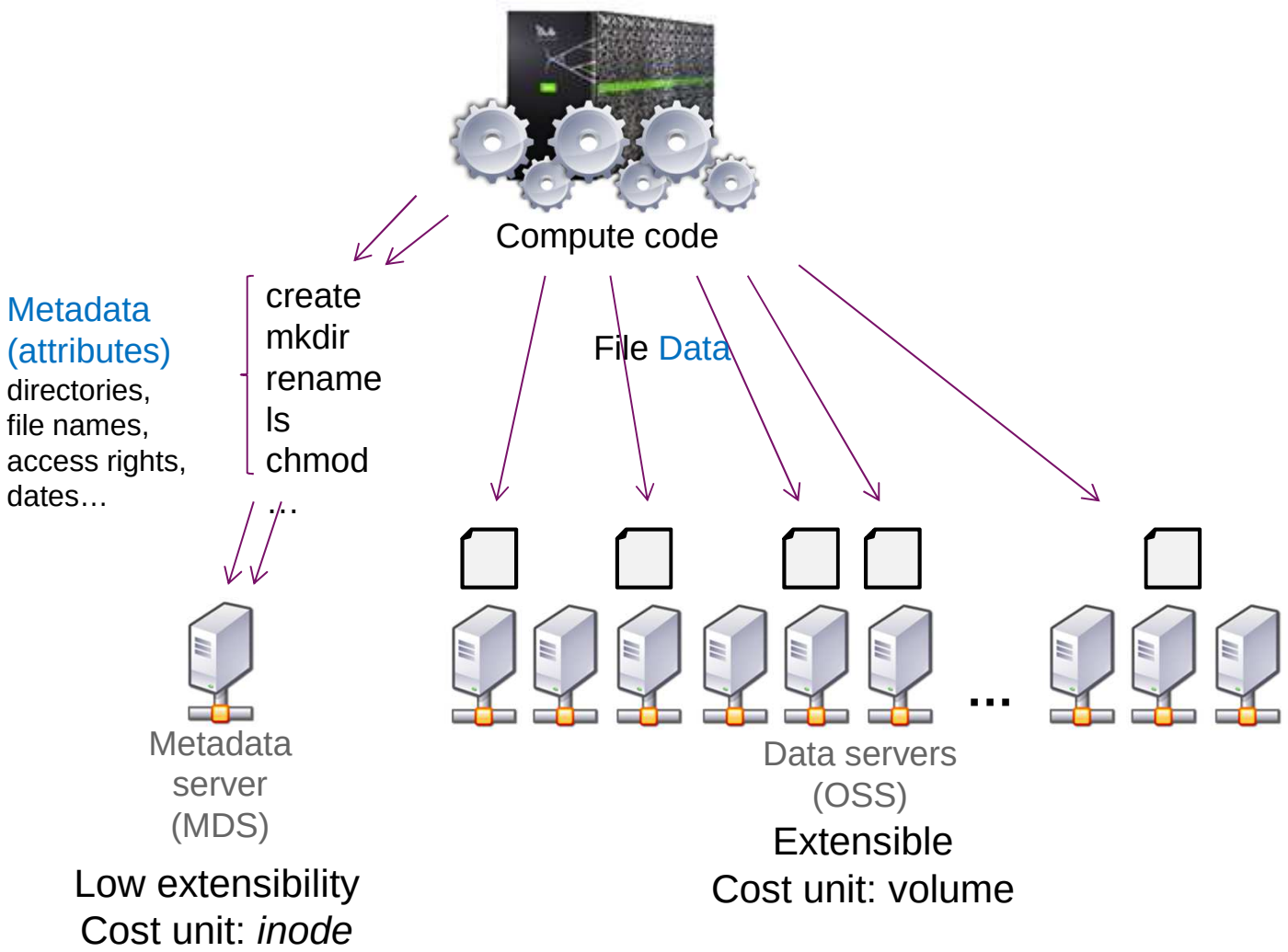


Parallel file system
M servers, N clients

Example:
scratch of supercomputer

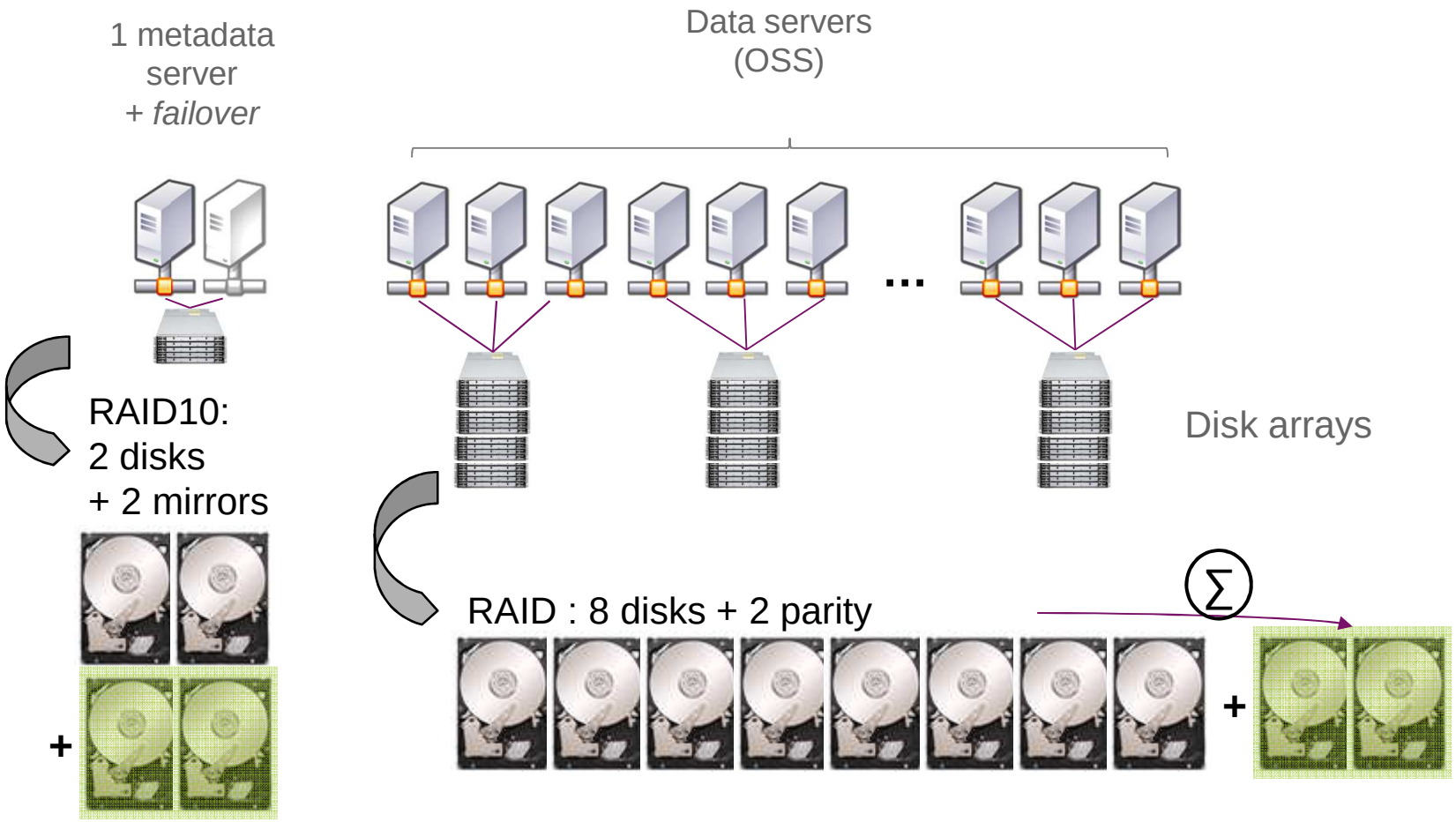
Interests:
scalability, performance, fault tolerance

LUSTRE: A PARALLEL FILE SYSTEM



LUSTRE: HARDWARE REDUNDANCY

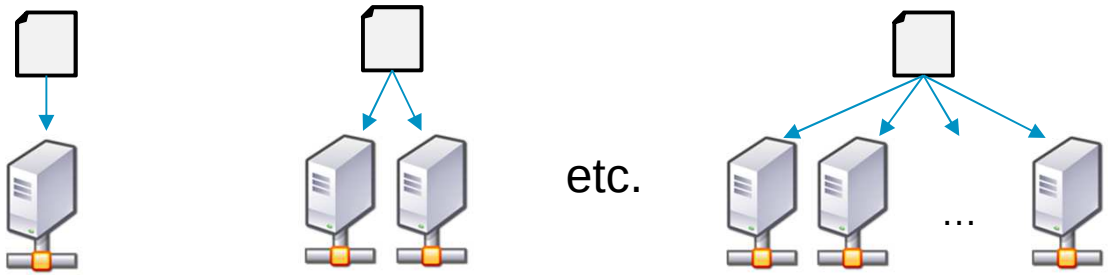
Hardware redundancy of Lustre filesystems



LUSTRE STRIPING

What is striping?

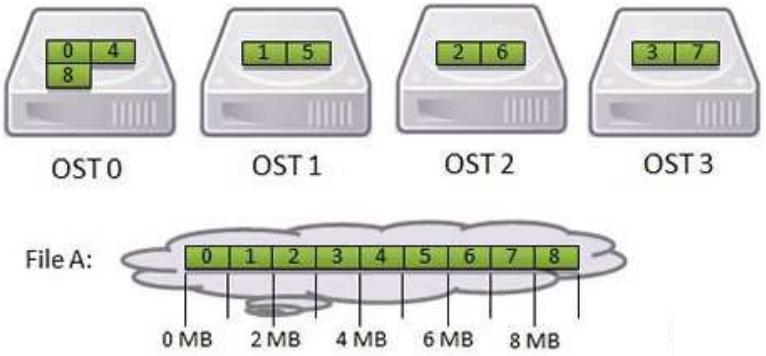
- To increase data throughput Lustre can paralellize file storage on several servers



Stripe count = 1 **Stripe count = 2** **Stripe count = N**

- Data is distributed across servers as blocks of « **stripe size** »

Example:
stripe_count=4
stripe_size=1MB



STRIPING: RECOMMENDATIONS

What striping should be set?

- Striping > 1 induces extra costs (N servers to communicate with) but results in an increased bandwidth
- **Useless** for small files (< a few MB)
- **Worthwhile** for bigger files (~ Gigabyte-sized)
  - If accessed from a single client: stripe_count = 2 is enough to get the max throughput
  - Increase stripe count if many clients write large volumes of data to the same file

    - As much as possible, align writes with stripe_size
- **Mandatory** for huge files (x100 GB): avoid having more than 500GB / server

SETTING STRIPE

How to set stripe?

- Per directory
  - File and sub-directories inherit when they are created
- Only affects new file creation (not previously created files)
- Command:

```
lfs setstripe -c <stripe_count>  <directory>
```

Default stripe @ TGCC

- 1 on *scratch* and *work*
- 4 on *store*
- 4 with MPI-IO

LUSTRE: BEST PRACTICES

Best practices

- Avoid using « ls –l » when « ls » is enough
- Avoid having a huge number of files in a single directory (<1000)
- Avoid small files on Lustre filesystems
- Use a stripe count of 1 for directories with many small files
- Lustre filesystems are not backed up: keep critical data (e.g. source code) in your home
- Limit the number of processes writing to the same file (locking contentions)
- Avoid starting executables on Lustre (they run slower)
- Avoid repetitive open/close operations
  - Example of wrong script:

```
while … do
        echo ‘bla’ >> my_file.out
done
```

- Open « read-only » when only reading a file to reduce locking contentions
  - In Fortran, use ACTION='read' instead of the default ACTION='readwrite‘

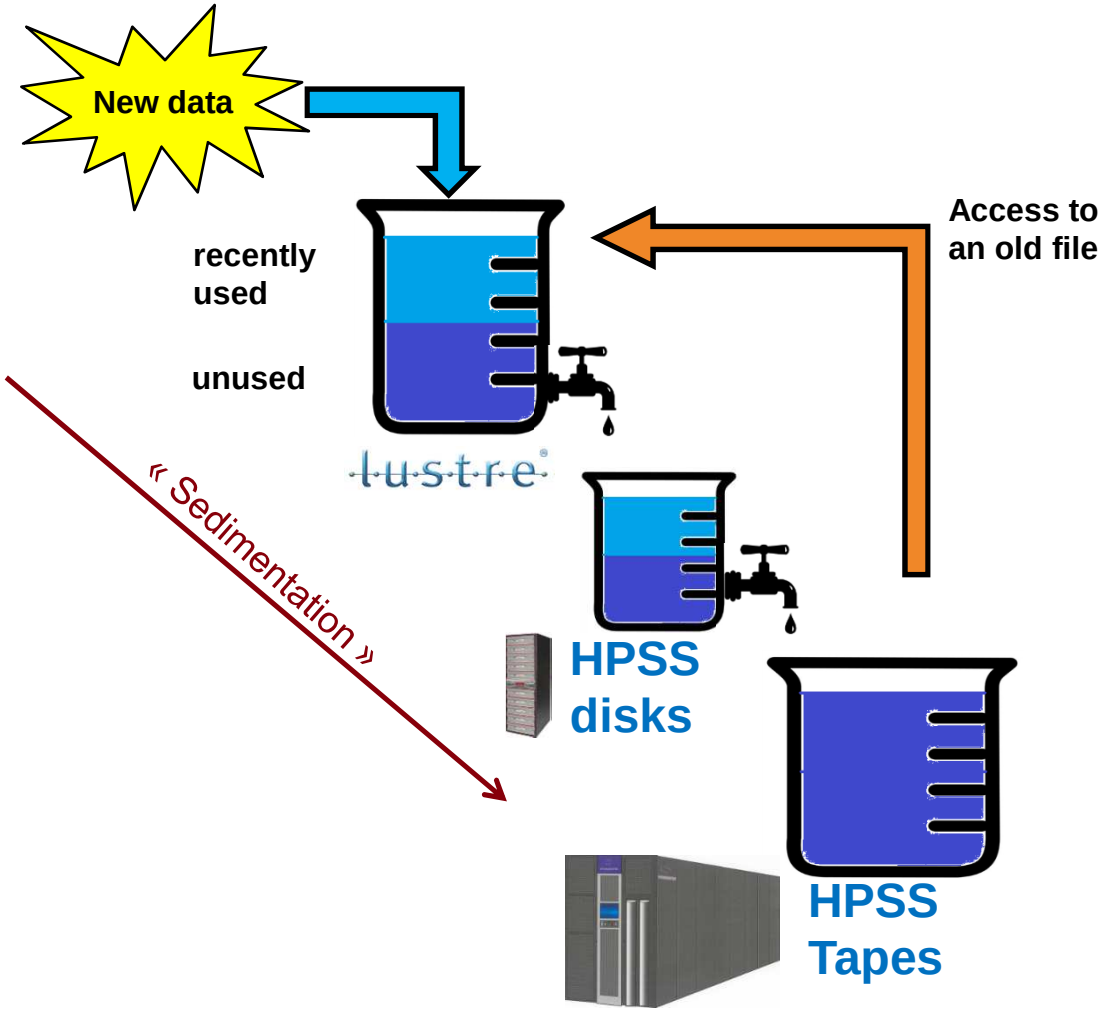
More details

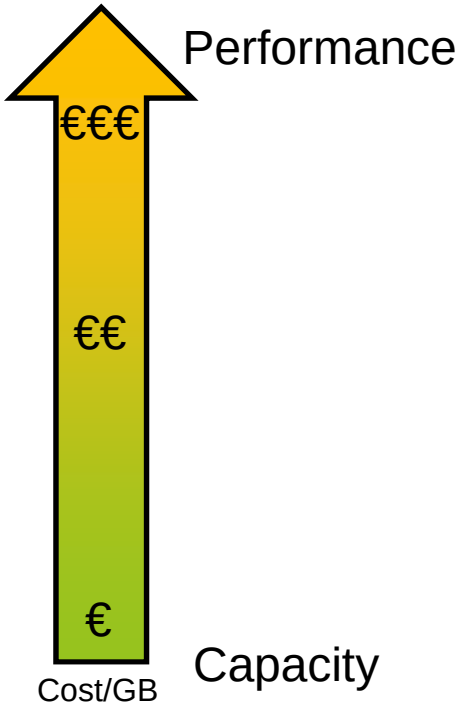
- Google « Lustre Best Practices »: some sites have good doc available online (NASA, NICS…)

Store: Hierarchical Storage Management

BASEMENT OF HSM

Data « sedimentation »

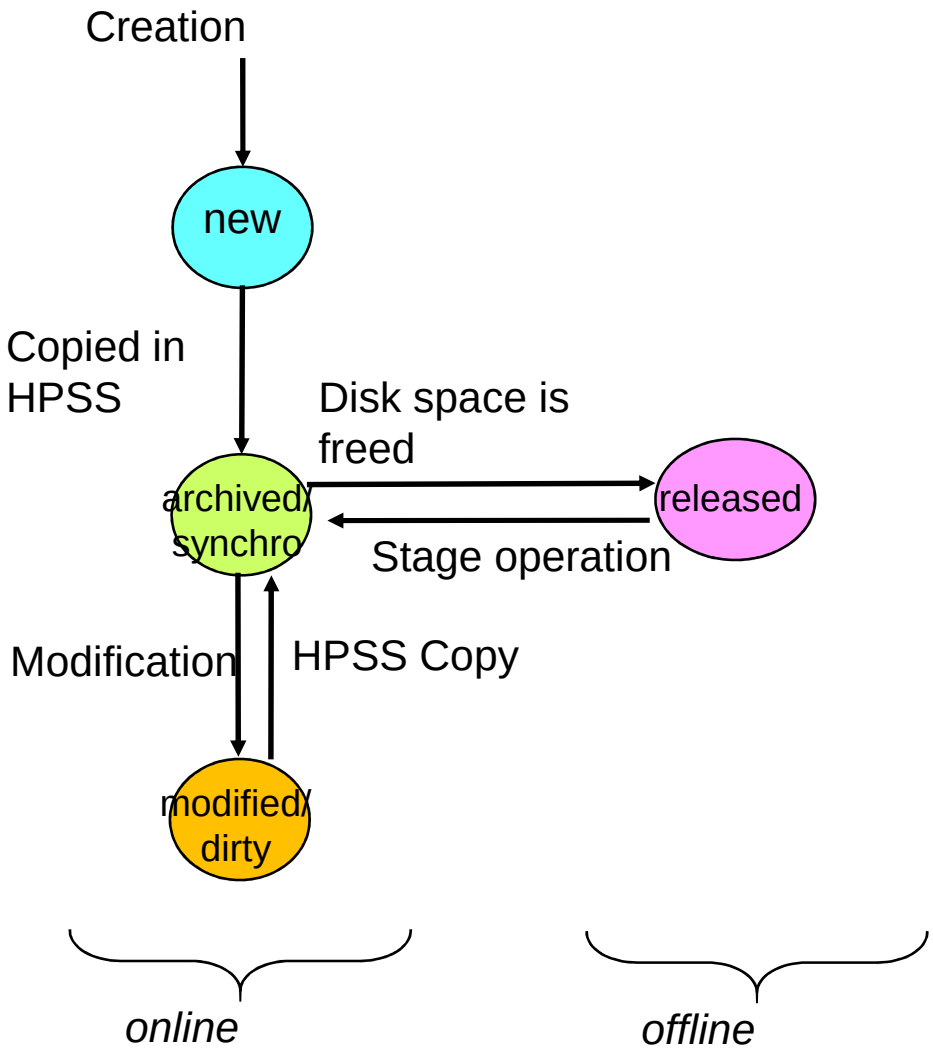




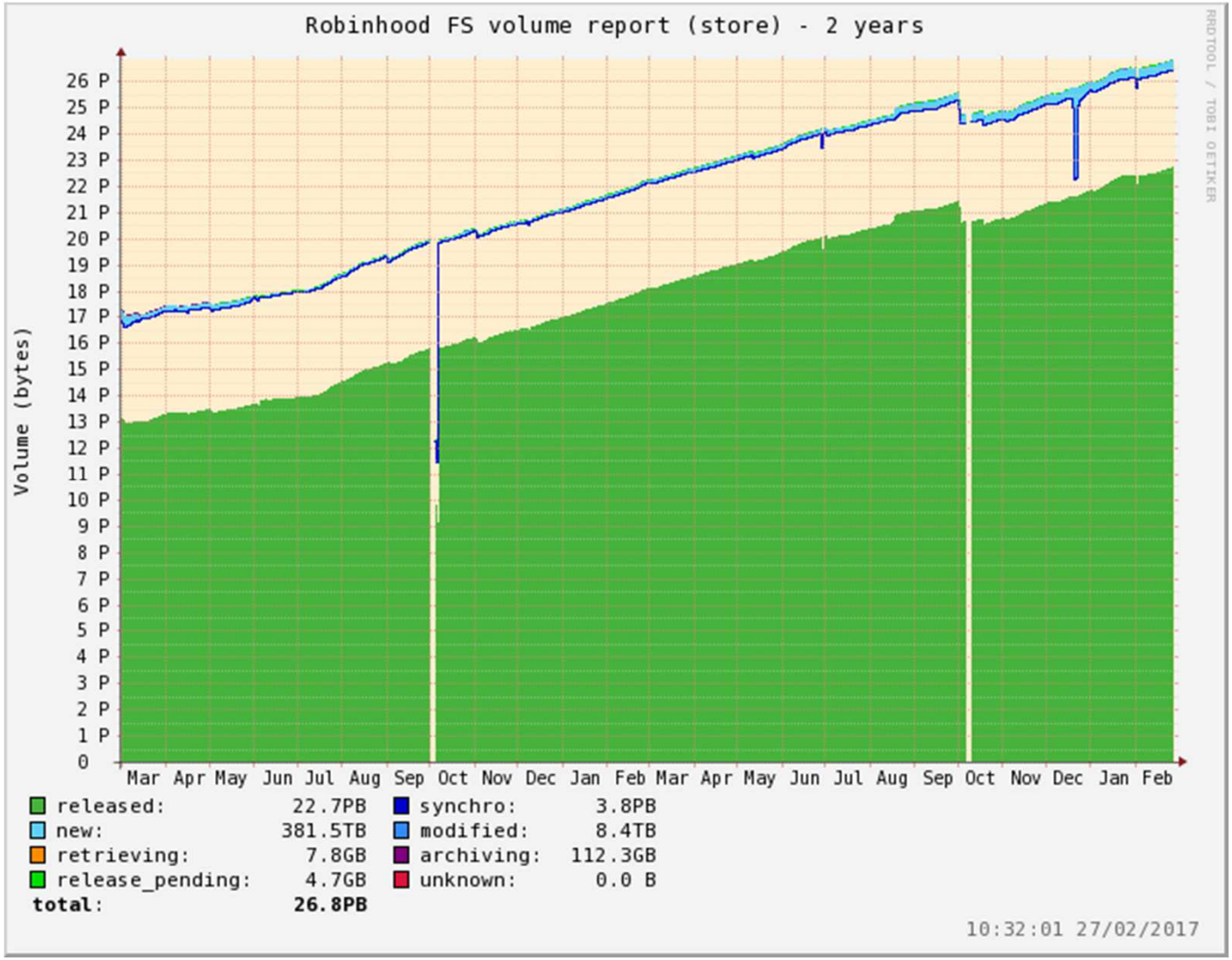
How HSM works

- *store* is permanently watched by a *Policy Engine* (*Robinhood*)
- Eligible files for migration are automatically stored in HPSS
  - The filesystem is saved in the HSM
  - *Possible recovery in case of crash, major hardware failure, FS been reformatted*
- Older files are
  - Still visible in *store* with their original size
  - Their contents are out of store and kept in HPSS
  - This is fully transparent to the end-user
  - Space freed in store is available for new files
- Freed files are staged back at first access
  - Transparent to the end-user
  - The first IO call is blocked until the stage operation is completed

A FILE'S LIFE


Creation
new
Copied in HPSS
Disk space is freed
archived/ synchro
released
Stage operation
Modification
HPSS Copy
modified/ dirty
online
offline

FILES STATUS



USER INTERFACE

Users' view:

- User has access to data via a standardized path:

  /ccc/store/contxxx/grp/usr ($STOREDIR)
- No direct access to HPSS, it's « hidden » behind store
- Regular commands apply to store
- Accessing a released file stages it back to LUSTRE.

  Data access is blocked until the transfer is completed.

ccc_hsm command:

ccc_hsm status : query file status `(online, released, …)`

ccc_hsm get : prefetch files

ccc_hsm ls : does « ls » but show hsm status (online, offline) too

CCC_HSM GET

Preloading data

- Retrieving data from tapes can be long: mounting and reading magnetic tapes

- It is **advised** to preload data **before submitting** a job (to reuse or post-process an **old computation**)
  - Preloading data **avoid wasting compute time**

- « `ccc_hsm get` » to preload ‘released’ files

‘DU’ ON STORE

What ‘du’ displays on /ccc/store ?

- By default, ‘du’ displays space used on **disk**, i.e. only on the Lustre level:

```
du -sh $CCSTOREDIR
2T (?!)
```

- If you want to get the total usage for both Lustre and HPSS **use ‘–b’ option**:

```
du -bsh $CCCSTOREDIR
224T ☺
```

BEST PRACTICE (STORE): PACKING DATA

Pack your data into big files

- The time to reload each file from tape is significant:
  - Time to move & load the tape in a tape drive, time to rewind the tape…

➔ **Packing data into bigger files makes it possible to reduce the time to read-back data from tapes**

Example: Reading the same amount of data (100GB) from tapes

	10 files x 10GB	1000 files x 100MB
Time to read-back from tapes	A few minutes	Several hours
Overall system usage (tape drives)	Partial	Full

Recommended file size: 1GB to 500GB

TAR IS YOUR FRIEND

TAR is dangerous only in cigarettes

- Using “tar” command is an easy way of packing files
  - `tar cf output.tar source_directory`
- Tools exist to access tarballs from software
  - Tarfiles follow a well known standard
  - See libarchive for example
- TAR preserves metadata
  - Permissions
  - Owners/groupes
- TAR preserves symlinks
- TAR can be appended

Alternative: you can use cpio if you prefer ;-)

Thinking on a framework to perform IO in simulation code is never a bad idea.

KEEP IN MIND WHAT THE RESOURCES ARE MADE FOR

- STOREDIR = LONG-TERM & CAPACITY
- WORKDIR = WORKING & SHARING
- SCRATCH = TEMPORARY & PERFORMANCES

DE LA RECHERCHE À L'INDUSTRIE

cea

Thanks for your attention

Questions?

Commissariat à l'énergie atomique et aux énergies alternatives
Centre de Saclay | 91191 Gif-sur-Yvette Cedex

CEA/DAM/DIF

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019