Consortium des Équipements de Calcul Intensif

Funded by F.R.S.-FNRS

www.ceci-hpc.be

The new CÉCI common storage

brought to you by Thomas Keutgen, David Colignon, Juan Cabrera, Frédéric Wautelet, Raphael Leplae,Bernard Van Renterghem, Sébastien Skozlowskij and Damien François

2017 CECI scientific day - Louvain-la-Neuve

damien.francois@uclouvain.be
www.uclouvain.be/cism

The new CÉCI common storage

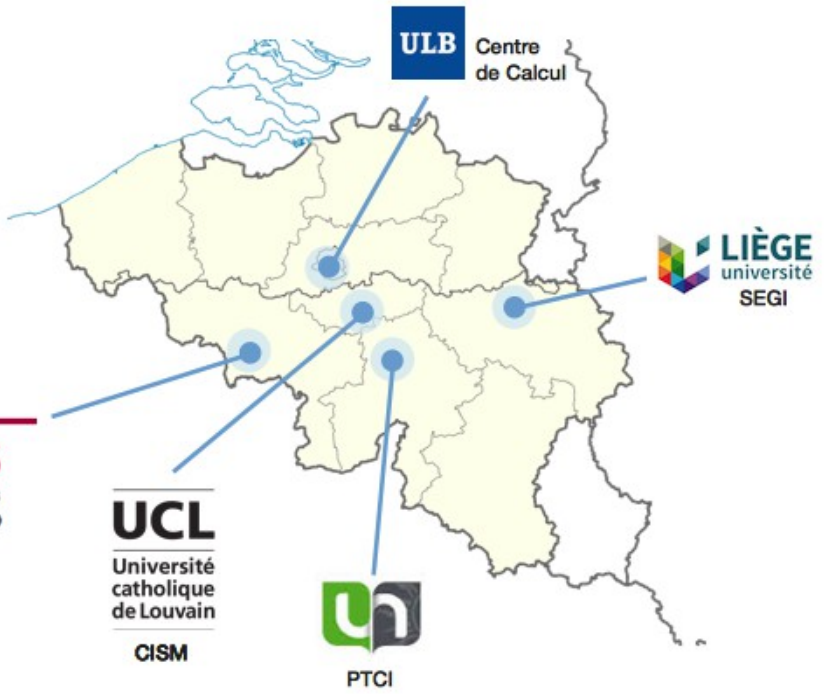
Features

C.E.C.I

6 clusters
5 sites

Vega

Dragon1



NIC4

Lemaitre2

Hmem

Hercules

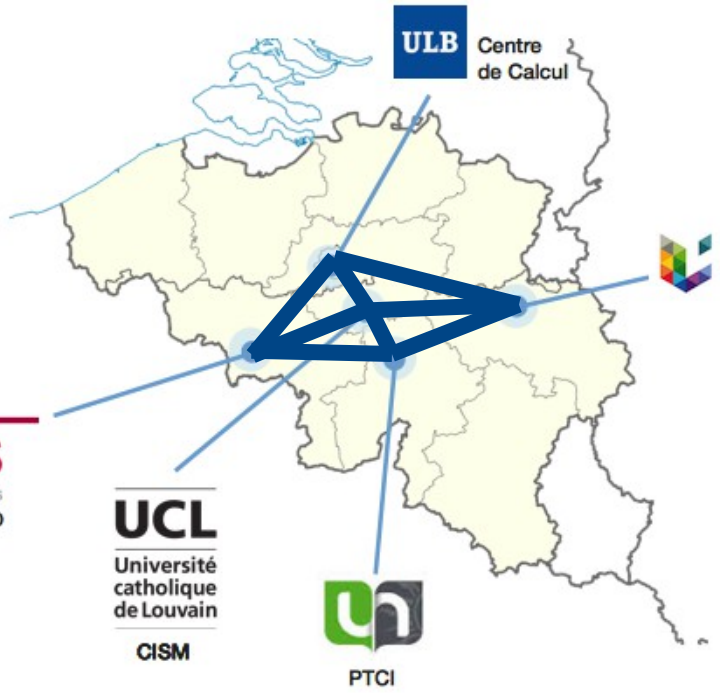
1. New network

new

Belnet
dedicated connectivity

10Gbps, dedicated, optical network direct links between the sites

Vega



NIC4

Dragon1

Lemaitre2

Hmem

Hercules

New hardware

2. New hardware

Two large storage solutions:

- Liège
- Louvain-la-Neuve (DCIII)

One smaller storage system on each site

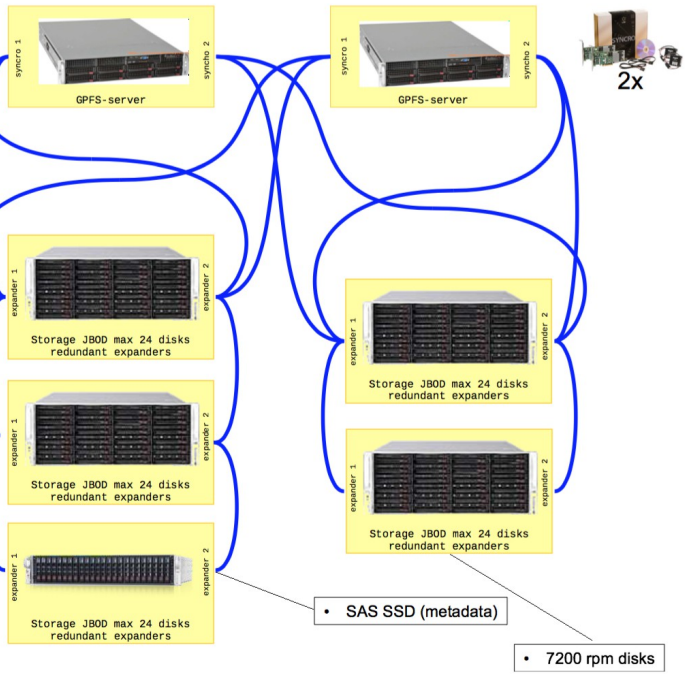
new

C.E.C.I

New hardware

2x

450TB netto
(formatted)



96 x 6TB SAS + 4 x 800GB SSD
8 x (10+2) RAID6

Each system is robust to:
- loss of one server
- loss of one pathway
- loss of two disks

Solution is robust to:
- loss of an entire system
- loss of connectivity

5x

55TB netto
(formatted)

12 x 6TB SAS + 2 x 480GB SSD
1 x (10+2) RAID6

Each system is robust to:
- loss of two disks

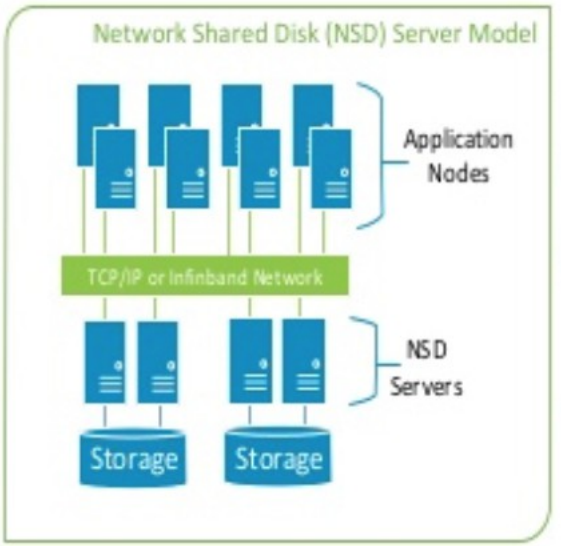
Solution is robust* to:
- loss of entire disk array
- loss of network access

C.E.C.I

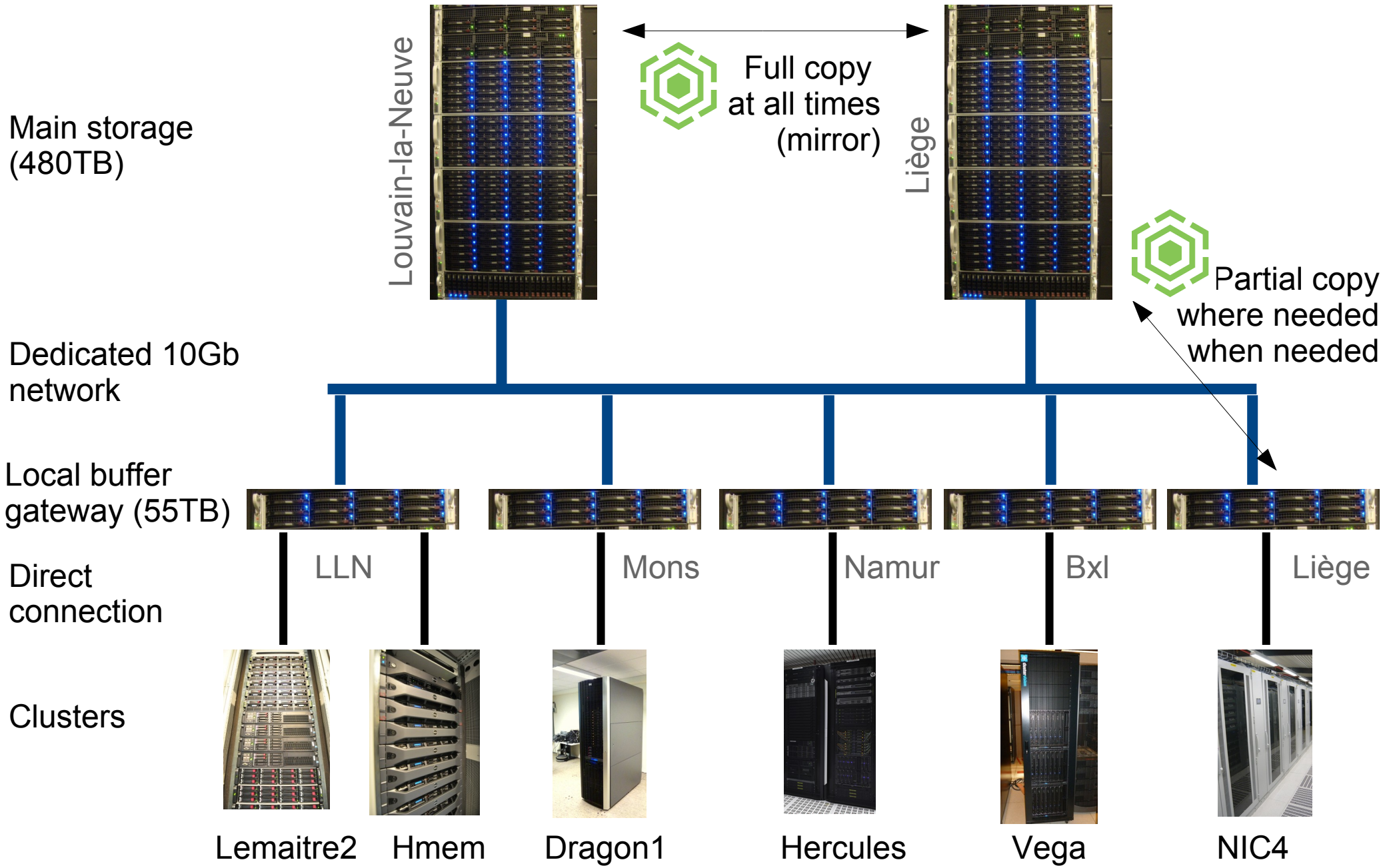
New software

3. New software

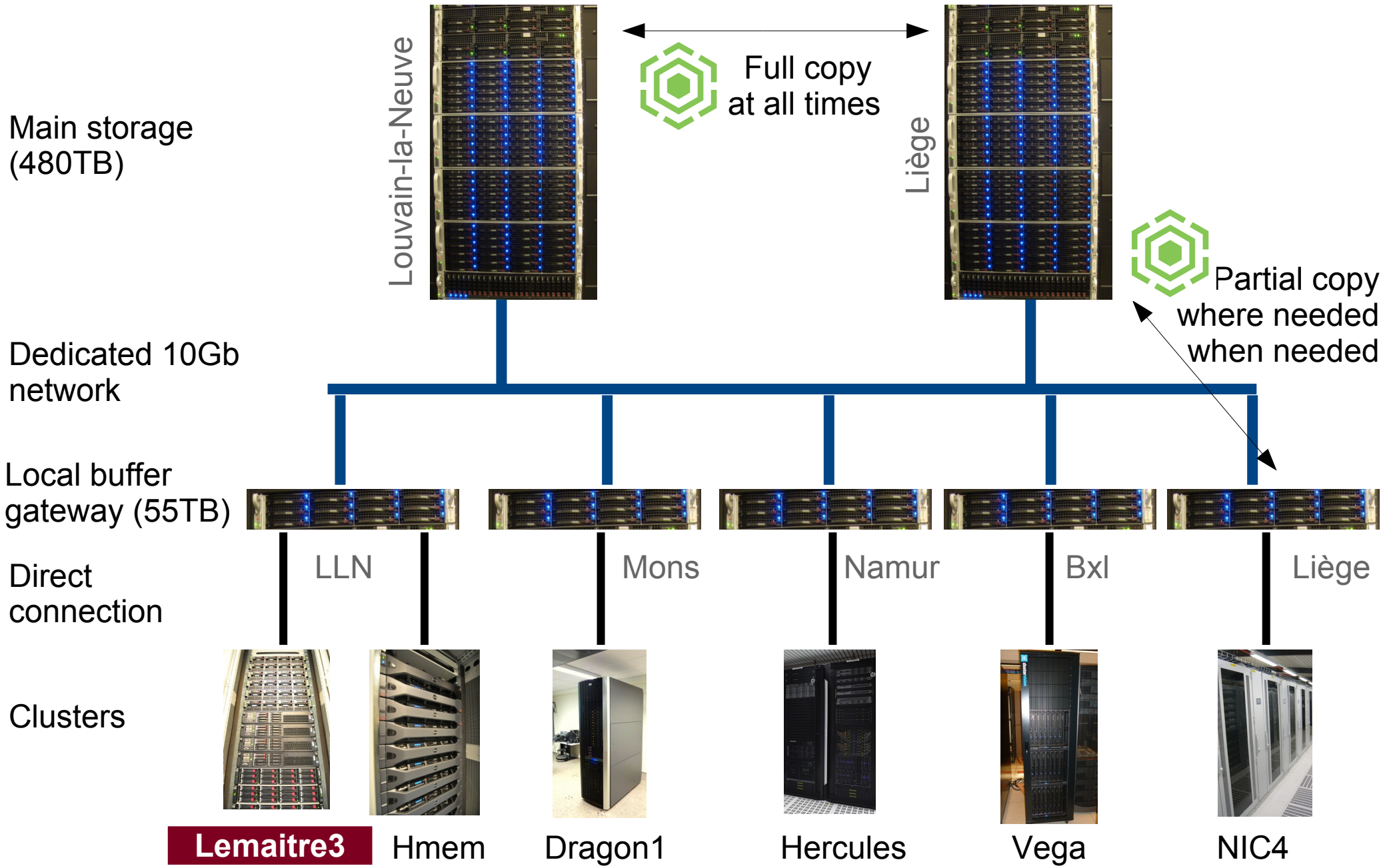
IBM Spectrum Scale



Setup overview



Setup overview



The new CÉCI common storage

Short-term benefits

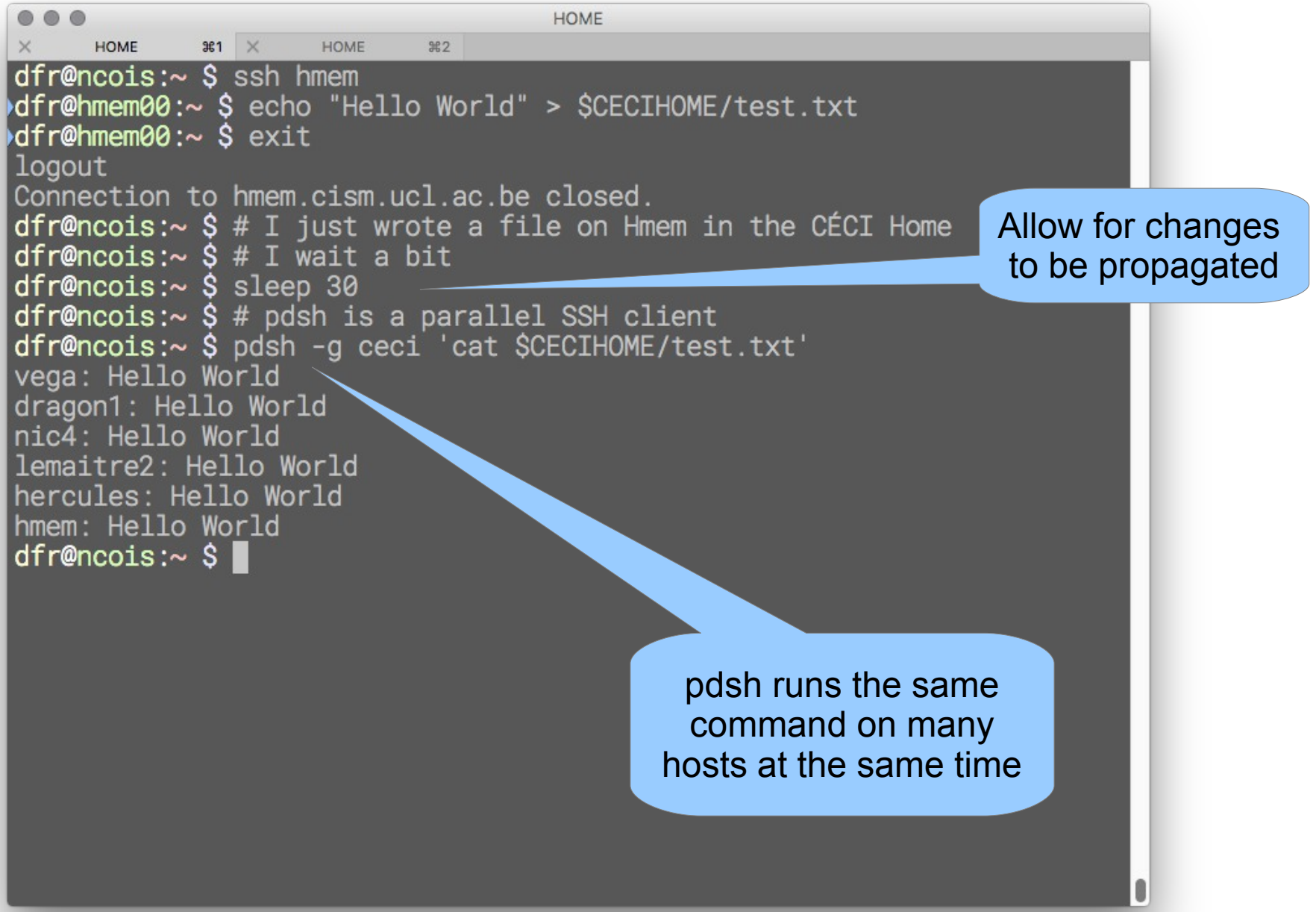
Four spaces

- /CECI/home
  - Quota 100GB/User
  - Daily snapshots
- /CECI/proj
  - Upon request
  - Quota and duration based on request
- /CECI/trsf
  - Quota per user 100GB soft 10TB hard
  - Automatic purge of files older than 6 months
- /CECI/soft
  - Common software + modules

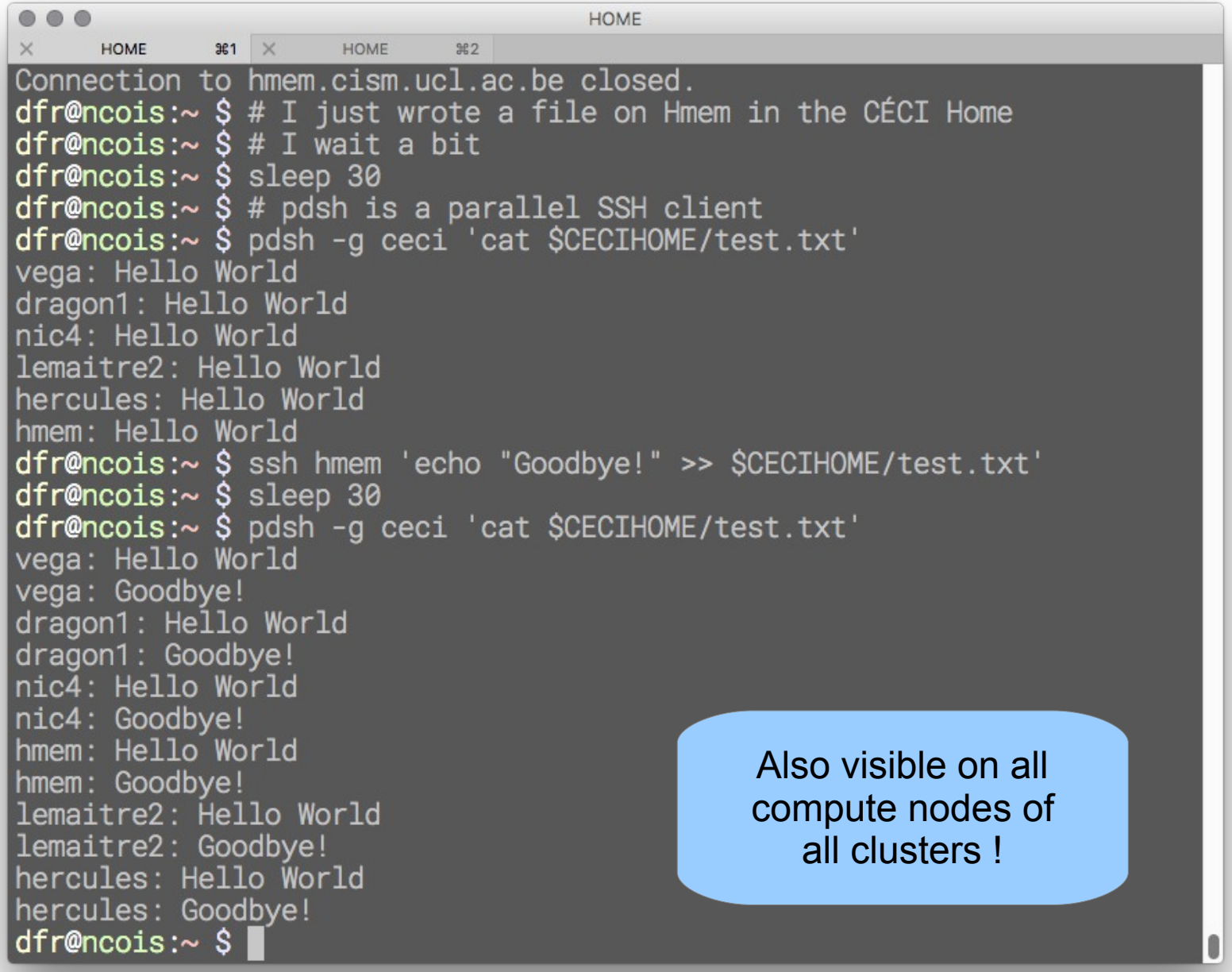
Four spaces

- /CECI/home
  - Quota 100GB/User
  - Daily snapshots
- /CECI/proj
  - Upon request
  - Quota and duration based on request
- /CECI/trsf
  - Quota per user 100GB soft 10TB hard
  - Automatic purge of files older than 6 months
- /CECI/soft
  - Common software + modules

Files written to $CECIHOME are visible on all clusters

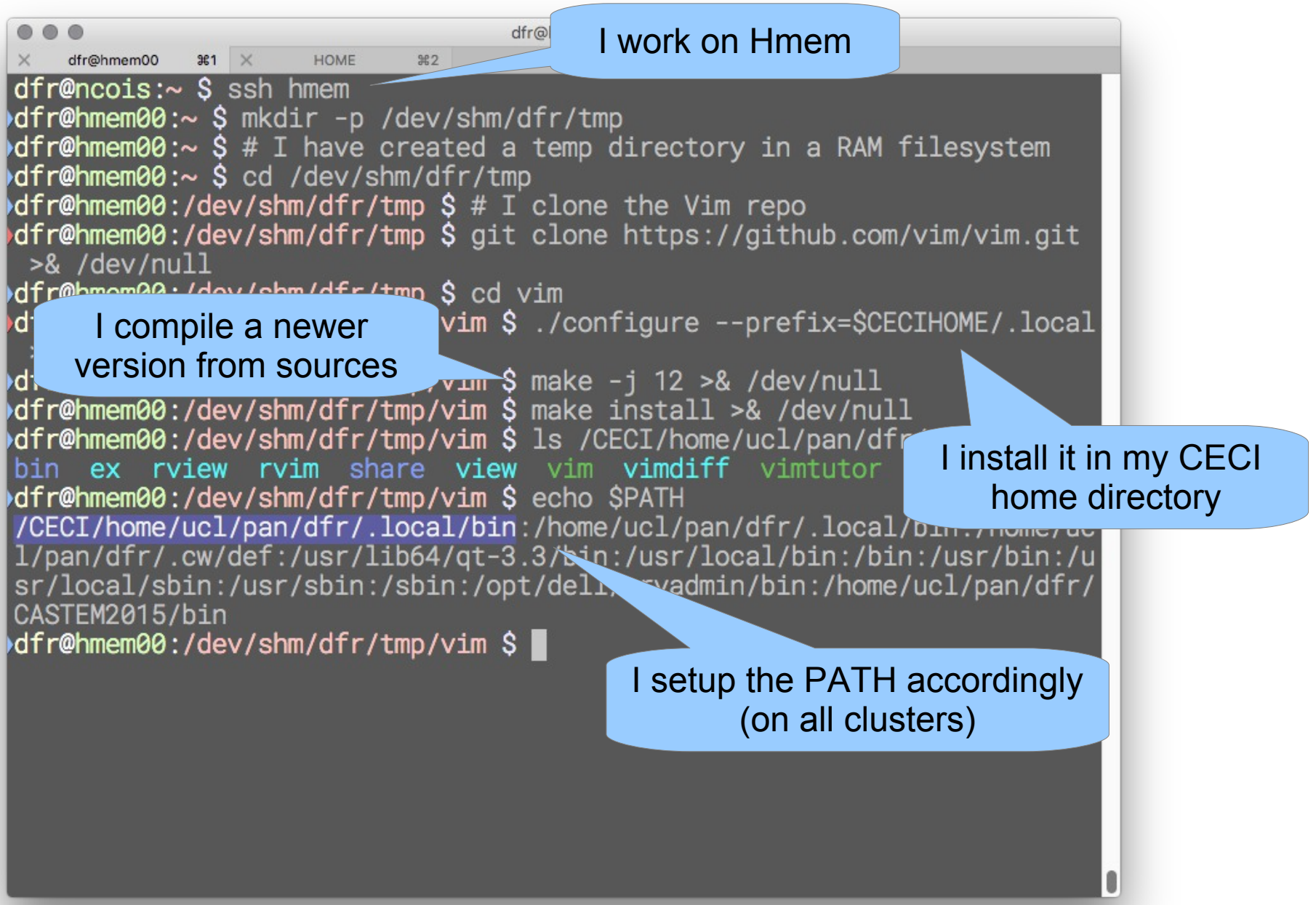


Files written to $CECIHOME are visible on all clusters

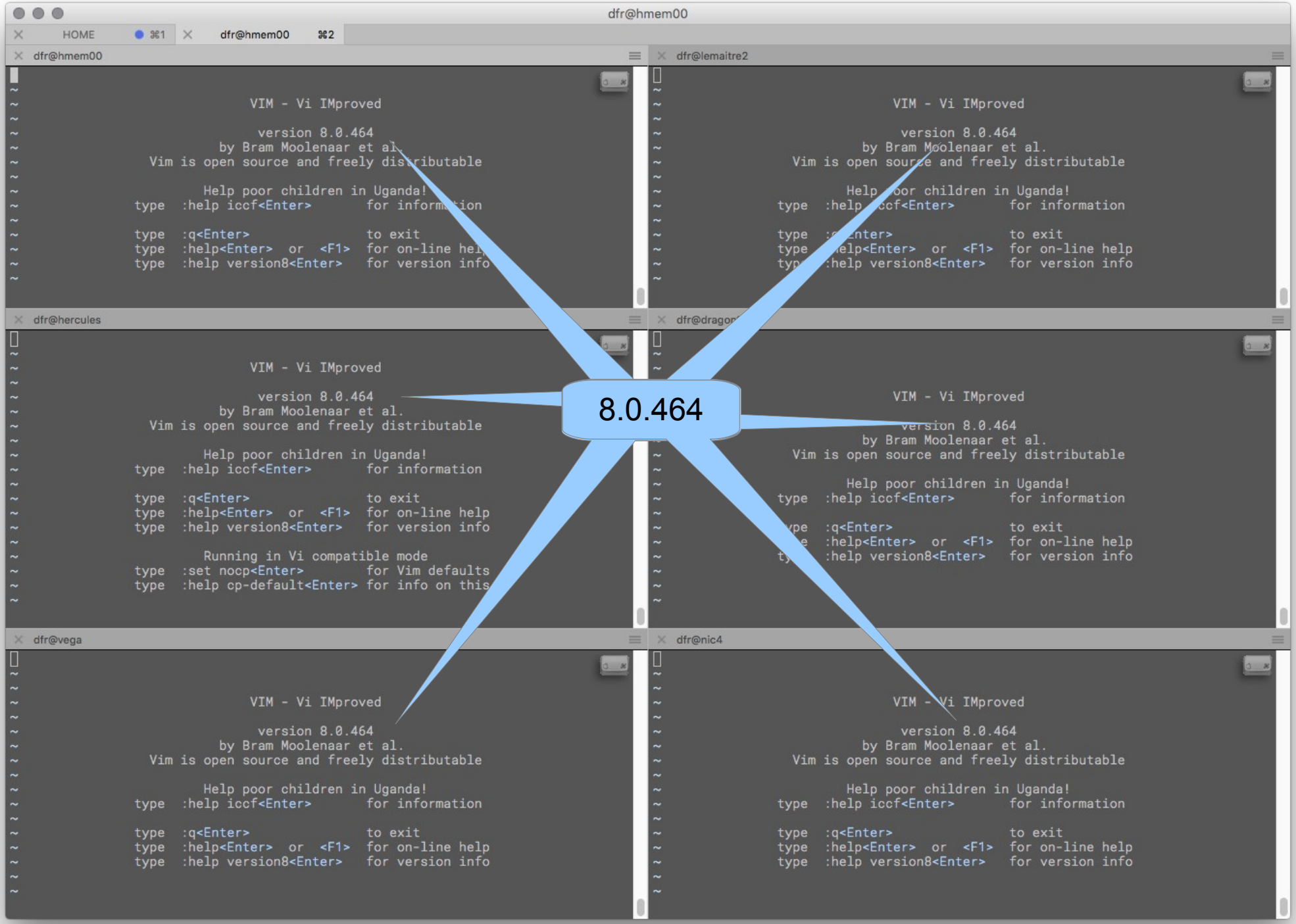


All clusters have a different Vim...

Example: install your own version of Vim on every cluster



Now all clusters have the same Vim...



A catch though...

By default, compilers will tune the code for the CPU of the machine they run on.

compile on \ run on	Hmem	Lemaitre2	Dragon1	Hercules	Vega	NIC4
Hmem	ok	sub-opt	sub-opt	sub-opt	sub-opt	sub-opt
Lemaitre2	crash	ok	sub-opt	sub-opt	sub-opt	sub-opt
Dragon1	crash	crash	ok	ok	sub-opt	ok
Hercules	crash	crash	ok	ok	sub-opt	ok
Vega	crash	crash	crash	crash	ok	crash
NIC4	crash	crash	ok	ok	sub-opt	ok

Mitigation:

GCC:

-march=core2 to build binaries running everywhere

-mtune=[westmere|sandybridge|bdver1]
to tune for the cluster you use the most

Intel (multiple code paths):

-xSSE2 to build binaries running everywhere

-axSSE4.2,AVX,CORE-AVX2
to add additional paths optimized for each cluster

Mitigation:

GCC: “Function multi-versioning”

Toggle line numbers

```
1  __attribute__ ((target ("default")))
2  int foo ()
3  {
4    // The default version of foo.
5    return 0;
6  }
7
8  __attribute__ ((target ("sse4.2")))
9  int foo ()
10 {
11   // foo version for SSE4.2
12   return 1;
13 }
14
15 __attribute__ ((target ("arch=atom")))
16 int foo ()
17 {
18   // foo version for the Intel ATOM processor
19   return 2;
20 }
21
22 __attribute__ ((target ("arch=amdfam10")))
23 int foo ()
24 {
25   // foo version for the AMD Family 0x10 processors.
26   return 3;
27 }
28 int main ()
29 {
30   int (*p)() = &foo;
31   assert ((*p) () == foo ());
32   return 0;
33 }
```

In the above example, 4 versions of function foo are created. The first version of foo with the target attribute "default" is the default version. This version gets executed when no other target specific version qualifies for execution on a particular platform. A new version of foo is created by using the same function signature but with a different target string. Function foo is called or a pointer to it is taken just like a regular function. With the new support, GCC takes care of doing the dispatching to call the right version at runtime.

Mitigation:

Intel compiler: “Manual processor dispatch”

Example

```
#include <stdio.h>
// need to create specific function versions for the following processors:
__declspec(cpu_dispatch(core_2nd_gen_avx, core_i7_sse4_2, core_2_duo_ssse3, generic ))
void dispatch_func() {};      //  stub that will call the appropriate specific function
version

__declspec(cpu_specific(core_2nd_gen_avx))
void dispatch_func() {
  printf("\nCode for 2nd generation Intel Core processors with support for AVX goes
here\n");
}

__declspec(cpu_specific(core_i7_sse4_2))
void dispatch_func() {
  printf("\nCode for Intel Core processors with support for SSE4.2 goes here\n");
}

__declspec(cpu_specific(core_2_duo_ssse3))
void dispatch_func() {
  printf("\nCode for Intel Core 2 Duo processors with support for SSSE3 goes here\n");
}

__declspec(cpu_specific(generic))
void dispatch_func() {
  printf("\nCode for non-Intel processors and generic Intel processors goes here\n");
}

int main() {
  dispatch_func();
  printf("Return from dispatch_func\n");
  return 0;
}
```

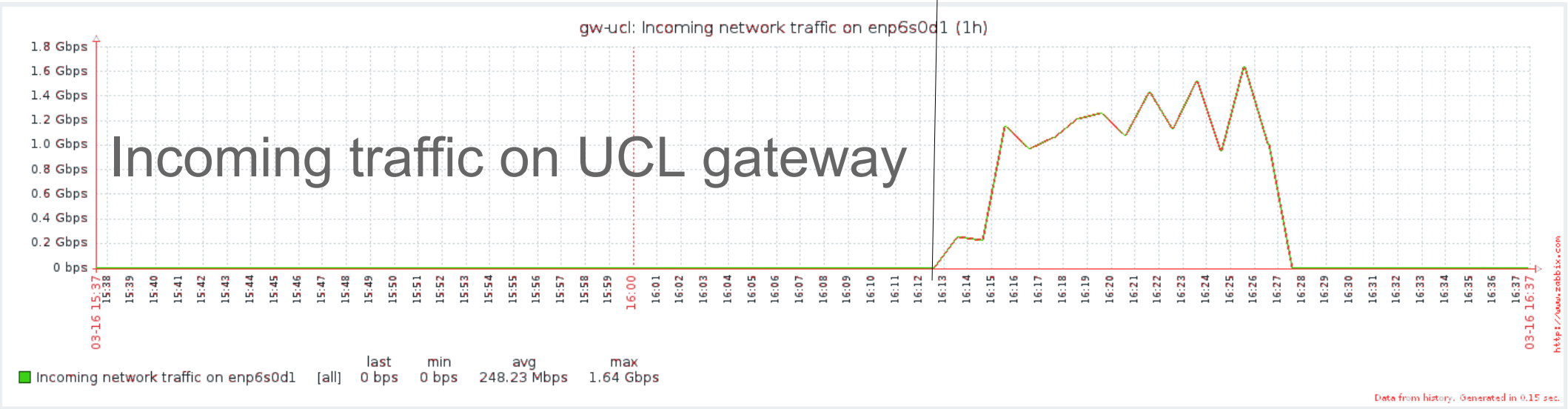
Four spaces

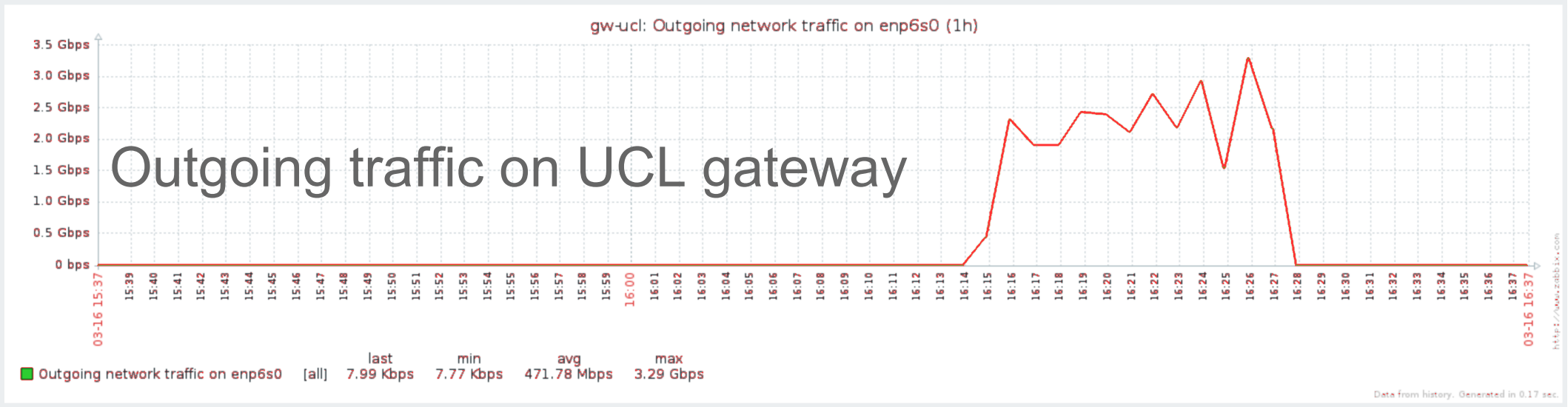
- /CECI/home
  - Quota 100GB/User
  - Daily snapshots
- /CECI/proj
  - Upon request
  - Quota and duration based on request
- /CECI/trsf
  - Quota per user 100GB soft 10TB hard
  - Automatic purge of files older than 6 months
- /CECI/soft
  - Common software + modules

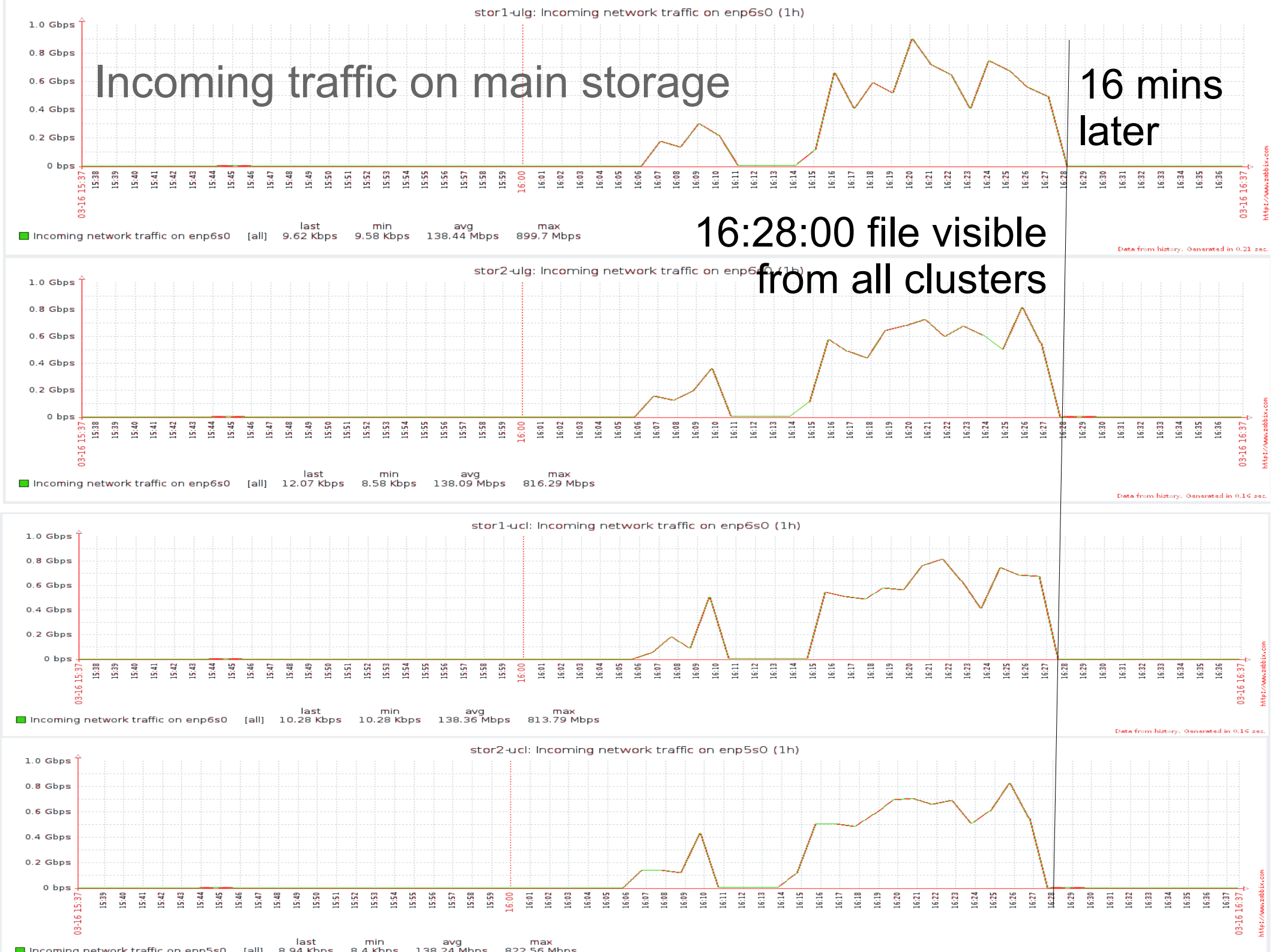
Copy of 100GB file

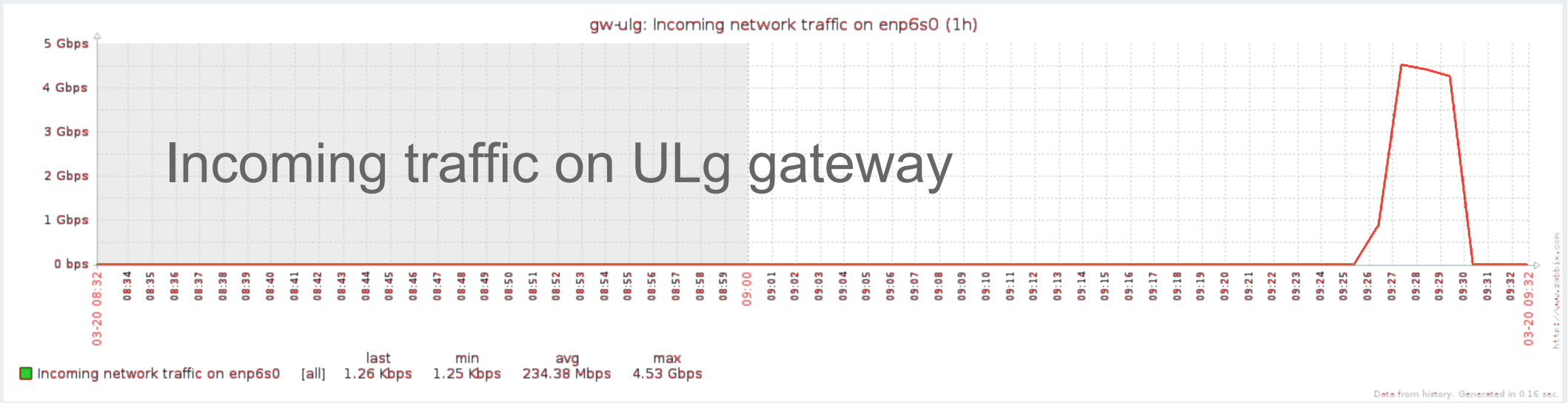
from the scratch of Lemaitre2 to scratch of NIC4

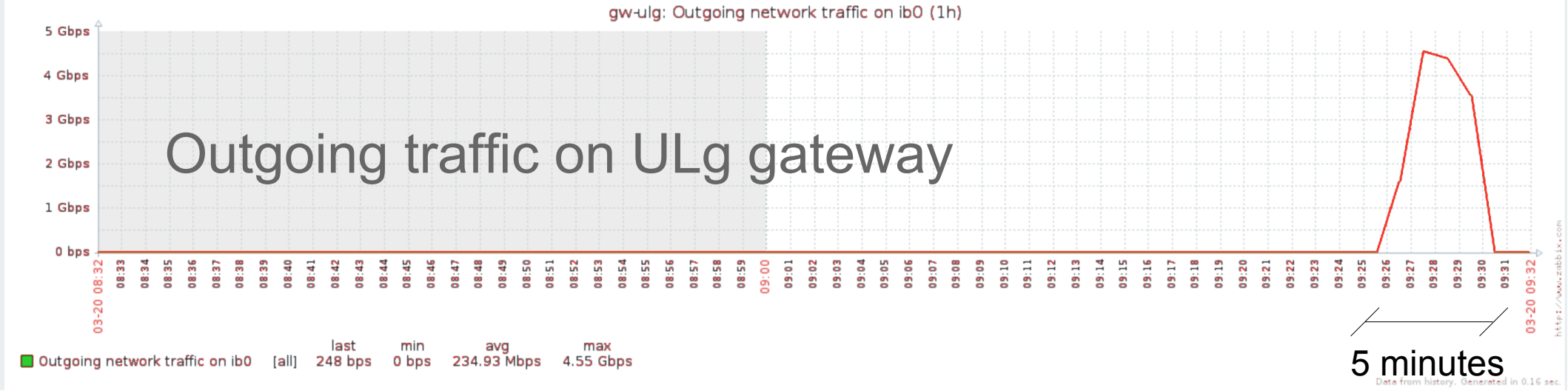
16:12:31 command issued





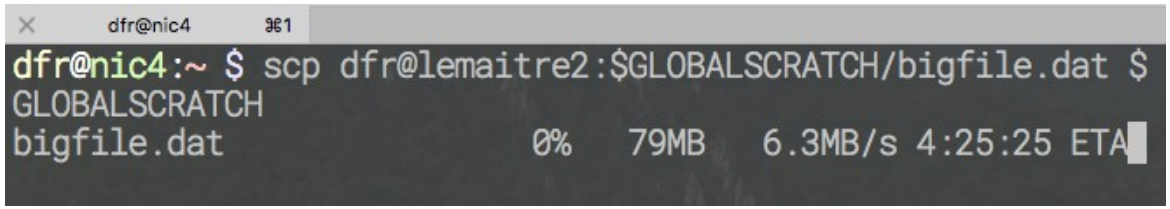

stor1-ulg: Incoming network traffic on enp6s0 (1h)
Incoming traffic on main storage
16 mins later
16:28:00 file visible from all clusters
1.0 Gbps
0.8 Gbps
0.6 Gbps
0.4 Gbps
0.2 Gbps
0 bps
03-16 15:37
16:00
03-16 16:37
Incoming network traffic on enp6s0 [all] last 9.62 Kbps min 9.58 Kbps avg 138.44 Mbps max 899.7 Mbps
Data from history. Generated in 0.21 sec.
stor2-ulg: Incoming network traffic on enp6s0 (1h)
Incoming network traffic on enp6s0 [all] last 12.07 Kbps min 8.58 Kbps avg 138.09 Mbps max 816.29 Mbps
Data from history. Generated in 0.16 sec.
stor1-ucl: Incoming network traffic on enp6s0 (1h)
Incoming network traffic on enp6s0 [all] last 10.28 Kbps min 10.28 Kbps avg 138.36 Mbps max 813.79 Mbps
Data from history. Generated in 0.16 sec.
stor2-ucl: Incoming network traffic on enp5s0 (1h)
Incoming network traffic on enp5s0 [all] last 8.94 Kbps min 8.4 Kbps avg 138.24 Mbps max 822.56 Mbps

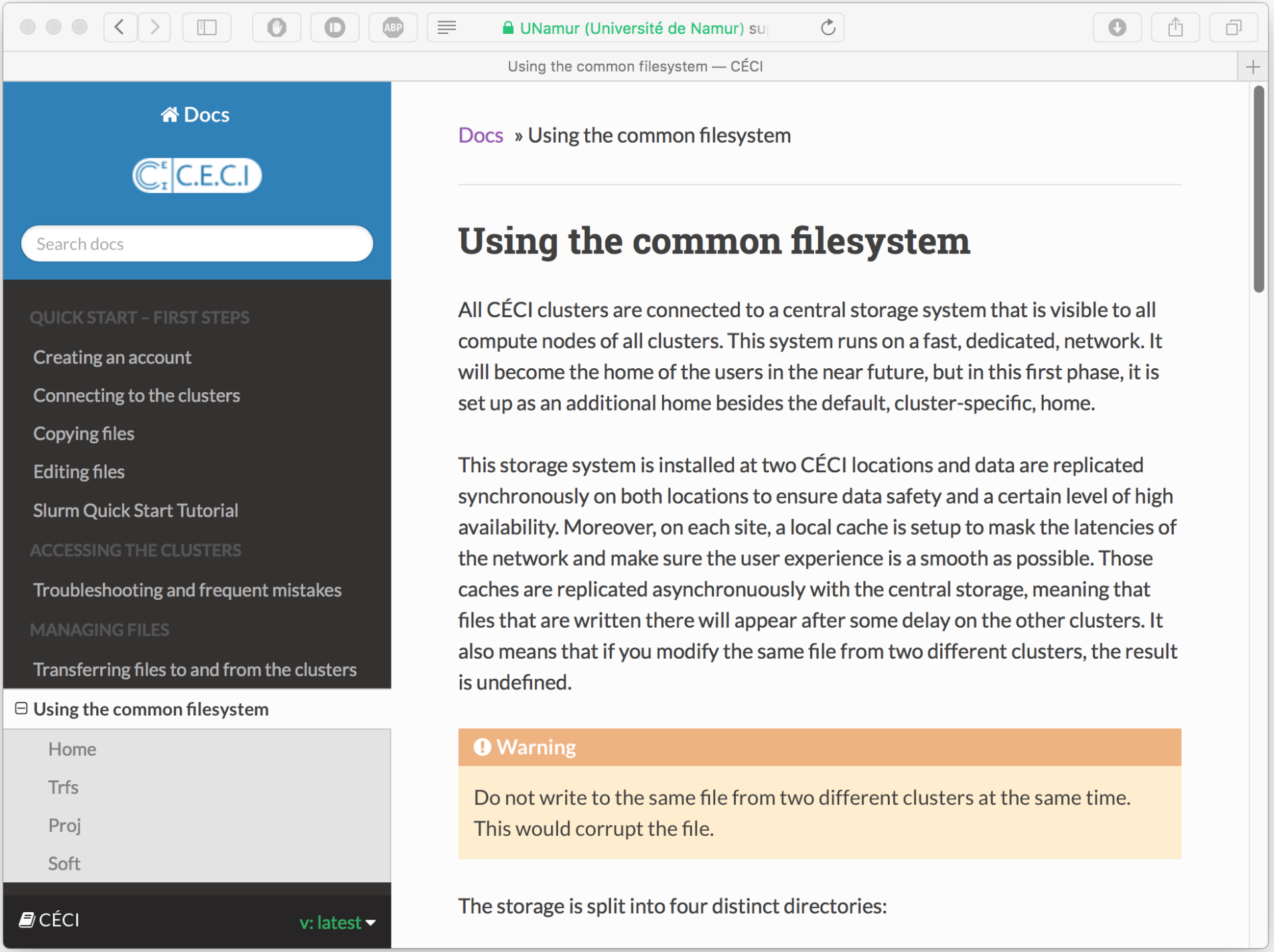




Total 21 minutes

vs.




UNamur (Université de Namur) su
Using the common filesystem — CÉCI
Docs
C.E.C.I
Search docs
QUICK START – FIRST STEPS
Creating an account
Connecting to the clusters
Copying files
Editing files
Slurm Quick Start Tutorial
ACCESSING THE CLUSTERS
Troubleshooting and frequent mistakes
MANAGING FILES
Transferring files to and from the clusters
Using the common filesystem
Home
Trfs
Proj
Soft
CÉCI
v: latest
Docs » Using the common filesystem
Using the common filesystem
All CÉCI clusters are connected to a central storage system that is visible to all compute nodes of all clusters. This system runs on a fast, dedicated, network. It will become the home of the users in the near future, but in this first phase, it is set up as an additional home besides the default, cluster-specific, home.
This storage system is installed at two CÉCI locations and data are replicated synchronously on both locations to ensure data safety and a certain level of high availability. Moreover, on each site, a local cache is setup to mask the latencies of the network and make sure the user experience is a smooth as possible. Those caches are replicated asynchronuously with the central storage, meaning that files that are written there will appear after some delay on the other clusters. It also means that if you modify the same file from two different clusters, the result is undefined.
Warning
Do not write to the same file from two different clusters at the same time. This would corrupt the file.
The storage is split into four distinct directories:

Four spaces

- /CECI/home
  - Quota 100GB/User
  - Daily snapshots
- /CECI/proj
  - Upon request
  - Quota and duration based on request
- /CECI/trsf
  - Quota per user 100GB soft 10TB hard
  - Automatic purge of files older than 6 months
- /CECI/soft
  - Common software + modules

Gotcha's

- Using the `du` command on a cluster will give you

  your usage on the gateway/cache connected to that cluster.

  and not your usage on the main storage.
- Use the `quota -usa` command on a cluster to get your usage on the main storage.
- Some clusters mount the storage via *automount* so the `quota` command **might not display the CÉCI storage**. In that case, run "`ls $CECIHOME`" to trigger the mounting and run `quota` again.
- Quota are enforced at the gateway/cache level (100GB) and at on the main storage (200GB).
  - More than 100GB on a cluster => you cannot write from that cluster anymore
  - More than 200GB on the main storage => data are not transferred from the clusters anymore

What's next...

- Fine-tune configuration

  Your feedback: your local CÉCI system administrator

- Setup procedure to request group space
- Build common software installation

All further information through
the CÉCI users mailing list

C.E.C.I

Consortium des Équipements de Calcul Intensif

Funded by F.R.S.-FNRS

www.ceci-hpc.be

Try the new CÉCI common storage!