

# Introduction to high-performance computing

Frédéric Wautelet  
CÉCI HPC training 2022

# Outline

Julia

Accelerators

Python

SLURM

OpenMP

MPI

Checkpointing

Debugging and profiling

Compilers and libraries

Introduction to GNU/Linux and the command line

Introduction to HPC

# 2022 CÉCI training

**Learning How to use HPC infrastructure**

**Learning How to program on HPC cluster**

**Going Parallel**

**Data Management**

**Data Science**

# 2022 CÉCI training



**Learning How to use HPC infrastructure**



Learning How to program on HPC cluster

Going Parallel

Data Management

Data Science

# How to use HPC infrastructure

## Part I

1. Introduction to high-performance computing
2. Connecting with SSH from Windows and Linux on CECI clusters
3. Introduction to Linux and the command line

## Part II

4. Choosing and activating software with system modules on CECI clusters
5. Writing and editing text files with Vim
6. Preparing, submitting and managing jobs with Slurm
7. Using a Checkpoint/restart program to overcome time limits

# How to use HPC infrastructure

## Part I

1. Introduction to high-performance computing
2. Connecting with SSH from Windows and Linux on CECI clusters
3. Introduction to Linux and the command line

## Part II

4. Choosing and activating software with system modules on CECI clusters
5. Writing and editing text files with Vim
6. Preparing, submitting and managing jobs with Slurm
7. Using a Checkpoint/restart program to overcome time limits

# 2022 CÉCI training

Learning How to use HPC infrastructure

 **Learning How to program on HPC cluster** 

Going Parallel

Data Management

Data Science

# How to program on HPC cluster

## **Part I: Introduction**

1. Introduction to scientific software development and deployment
2. Introduction to scripting and interpreted languages (Python, R, Octave)
3. How to speed up you code at low cost?

## **Part II: Fortran/C/Julia**

4. Introduction to C programming language
5. Introduction to JULIA
6. Introduction to structured programming with Fortran



# How to program on HPC cluster

## Part I: Introduction

1. Introduction to scientific software development and deployment
2. Introduction to scripting and interpreted languages (Python, R, Octave)
3. How to speed up you code at low cost?

## Part II: Fortran/C/Julia

4. Introduction to C programming language
5. Introduction to JULIA
6. Introduction to structured programming with Fortran

# How to program on HPC cluster

## **Part III: Python**

7. Introduction to Python
8. Python as an Object Oriented Language
9. Efficient use of Python on the cluster

## **Part IV: Tools**

10. Introduction to code versioning
11. Debugging and profiling scientific code, and commercial optimized libraries

# How to program on HPC cluster

## Part III: Python

7. Introduction to Python
8. Python as an Object Oriented Language
9. Efficient use of Python on the cluster

## Part IV: Tools

10. Introduction to code versioning
11. Debugging and profiling scientific code, and commercial optimized libraries

# 2022 CÉCI training

Learning How to use HPC infrastructure

Learning How to program on HPC cluster



**Going Parallel**



Data Management

Data Science

# Session 3

## Part I

1. Introduction to parallel computing
2. Parallel programming with MPI

## Part II

4. Parallel programming with OpenMP
5. Directive Based Parallel programming on GPU (OpenACC)
6. Parallel programming on GPU with CUDA

# Session 3

## Part I

1. Introduction to parallel computing
2. Parallel programming with MPI

## Part II

4. Parallel programming with OpenMP
5. Directive Based Parallel programming on GPU (OpenACC)
6. Parallel programming on GPU with CUDA

# 2022 CÉCI training

Learning How to use HPC infrastructure

Learning How to program on HPC cluster

Going Parallel



**Data Management**



Data Science

# Data Management

1. Introduction to data storage and access
2. Efficient data storage on CECI clusters
3. Open Science and Open Research Data / Data Management Plan
4. Data versioning



# 2022 CÉCI training

Learning How to use HPC infrastructure

Learning How to program on HPC cluster

Going Parallel

Data Management



**Data Science (TBC)**



# Introduction to HPC

Julia

Accelerators/Co-processors

Python for HPC

Slurm workload manager

Share memory (OpenMP)

Message passing (MPI)

Checkpointing

Debugging and profiling

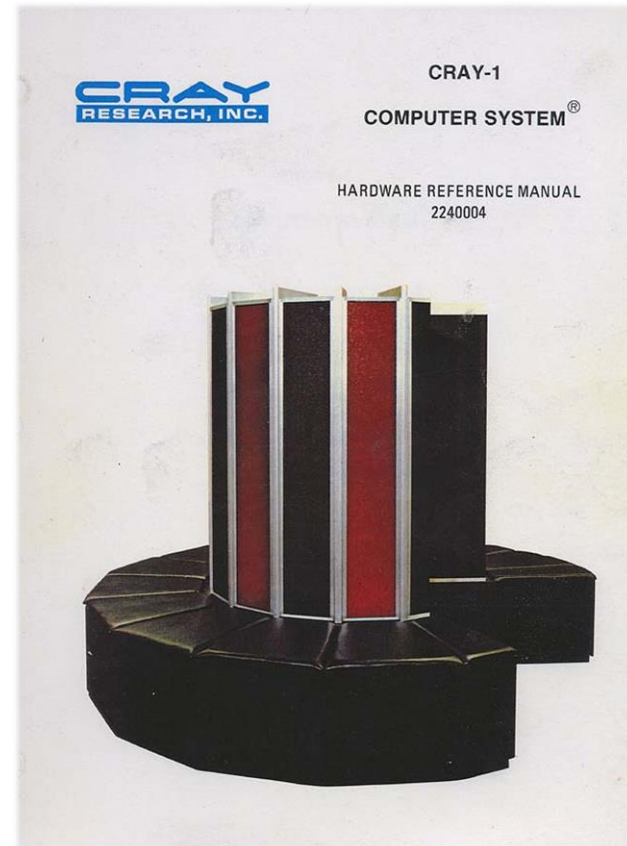
Compilers and libraries

Introduction to GNU/Linux and the command line

Introduction to HPC

# High Performance Computing

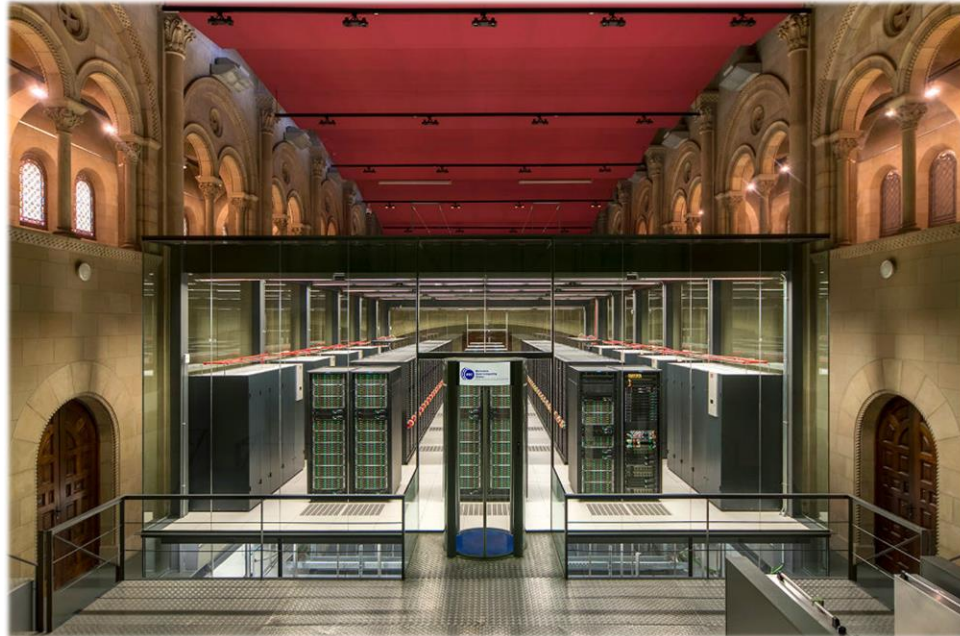
- High-performance computing (HPC) uses supercomputers and computer clusters to solve advanced computation problems.



**Cray-1a (1977)**  
250 MFlops

# Cluster

- A computer cluster is a group of linked computers, working together closely so that in many respects they form a single computer.



**MareNostrum 4 (2017)**

13.7 PFlops

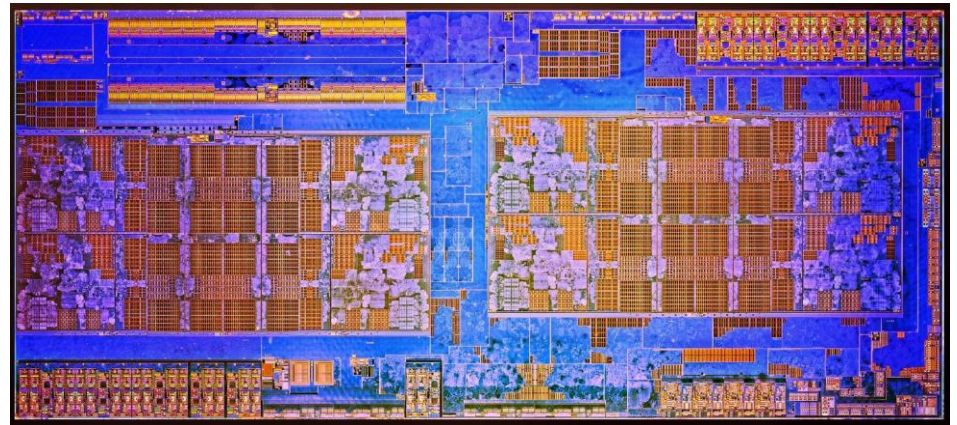
# Nodes and Cores

- Compute node
  - Part of a cluster
  - Equivalent to a high-end workstation



Dell PowerEdge R6415

- Core
  - A processor (CPU)
  - Multiple cores per socket



AMD Ryzen 7000 Series 8-core "chiplet"

# Measure supercomputer power

- FLOPS
- floating-point operations per second

GigaFLOPS = one billion ( $10^9$ ) floating-point operations per second

TeraFLOPS = one trillion ( $10^{12}$ ) floating-point operations per second

PetaFLOPS = one quadrillion ( $10^{15}$ ) floating-point operations per second

ExaFLOPS = one quintillion ( $10^{18}$ ) floating-point operations per second

# TOP500



- Frontier
  - Oak Ridge National Laboratory, Tennessee, USA
  - TOP500 #1 (June 2022)
  - 8,700,000 cores
  - First Exascale supercomputer: 1.1 Eflop/s
  - 21 MW
  - US\$600M





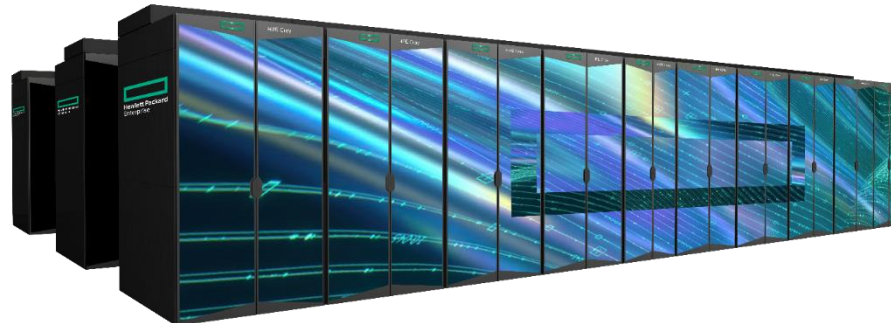
# Exascale in Europe



**EuroHPC**  
Joint Undertaking

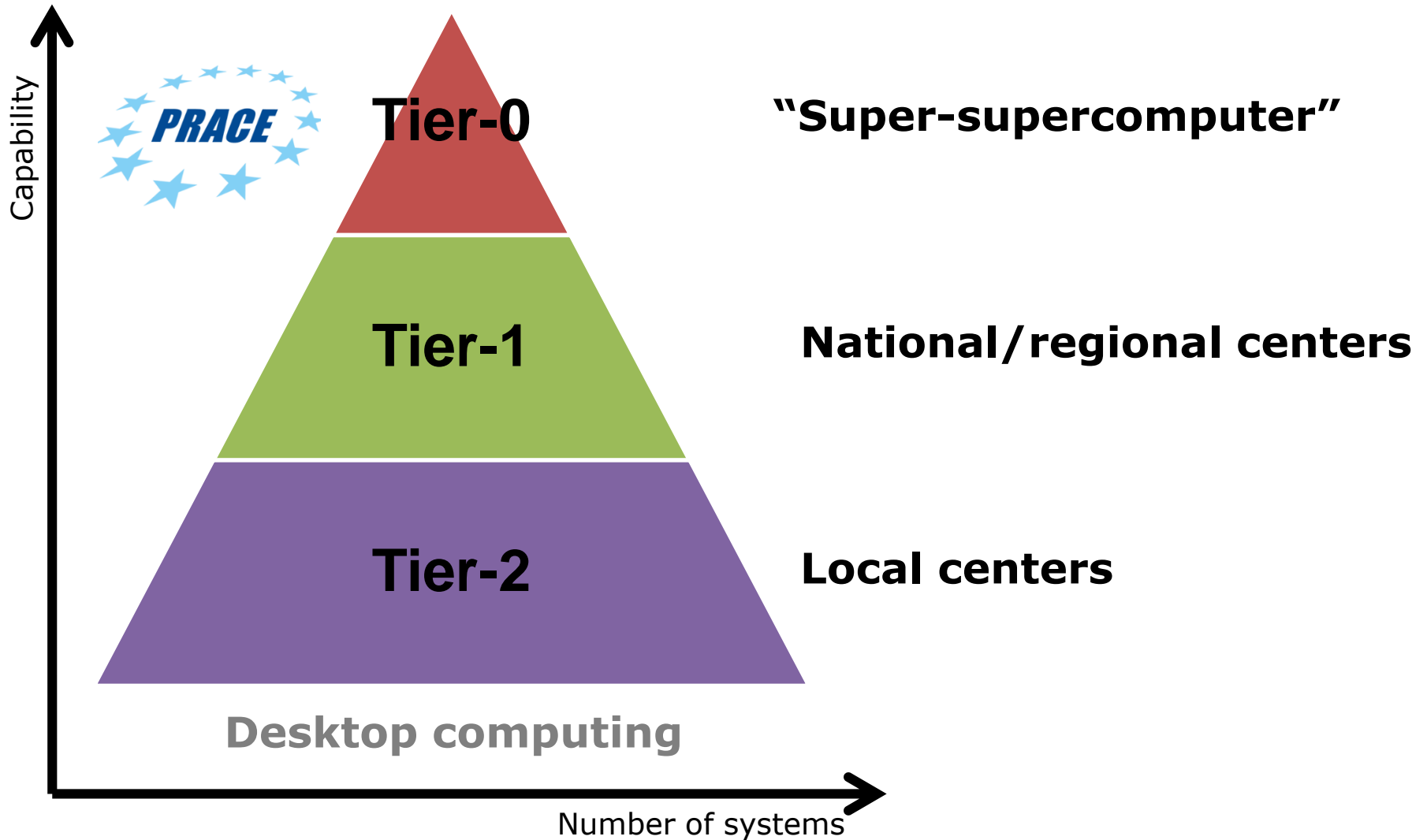
- First European pre-Exascale system: Lumi
  - CSC, Kajaani, Finland
  - 1,100,000 cores, 150 Pflop/s, 120 PB storage
  - #3 Top 500
- Belgian researchers eligible to apply for LUMI resources

LUMI





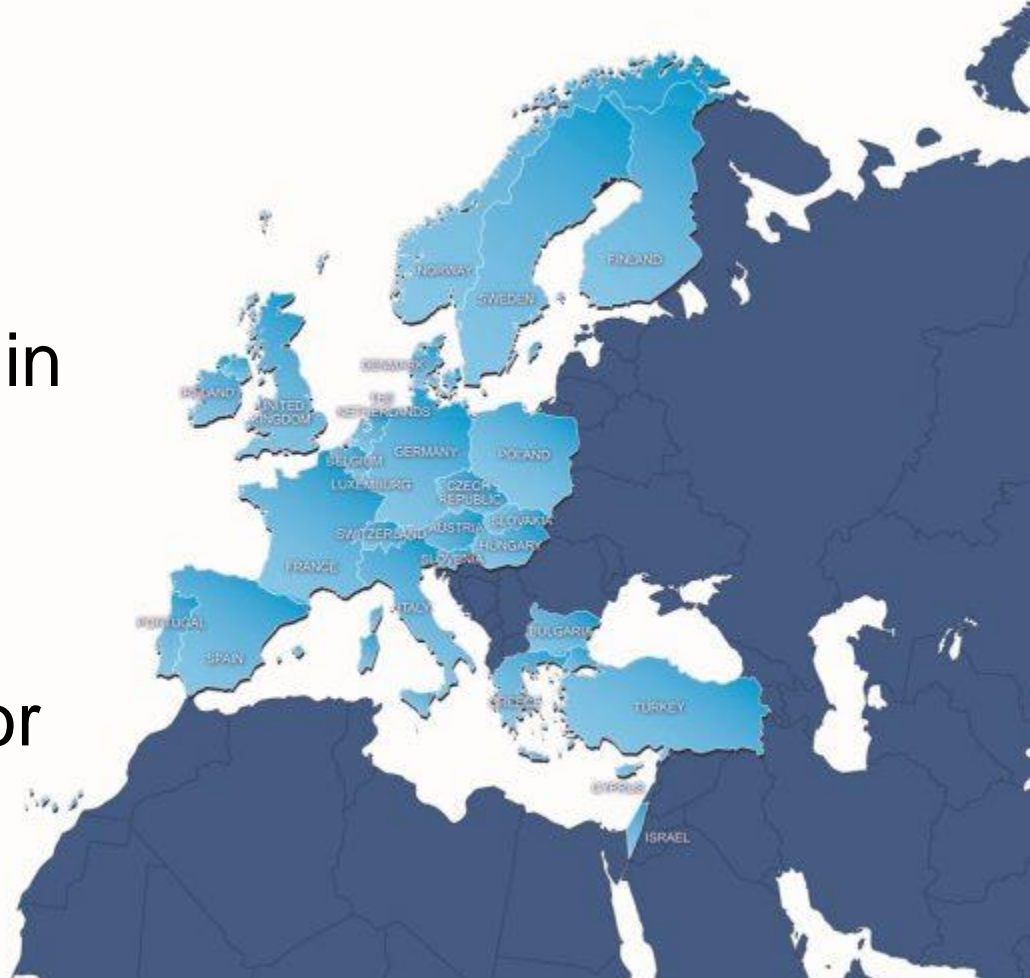
# The European HPC ecosystem



# PRACE



- Partnership for Advanced Computing in Europe
- 28 supercomputers in 26 countries
- Call for Proposals for Project Access



# Tier-0



- JUWELS Booster Module
  - Forschungszentrum Juelich, Germany
  - TOP500 #11 (June 2022)
  - ~450,000 cores
  - 44 Pflop/s
  - 2 MW



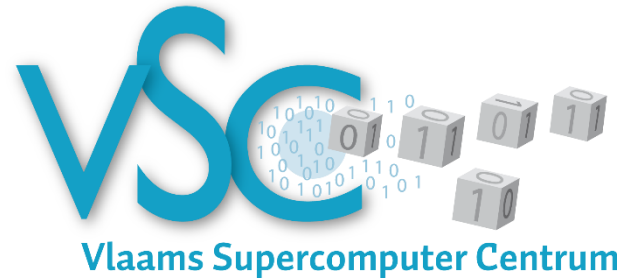
# Tier-1

- Tier-1
  - National and/or regional supercomputers
  - Allows the build-up of the necessary expertise and knowledge required to use a tier-0 supercomputer



# HPC in Belgium

- Vlaamse Gemeenschap/Vlaams Gewest
  - Vlaams Supercomputer Centrum (VSC)



- Fédération Wallonie-Bruxelles
  - Consortium des Équipements de Calcul Intensif (CÉCI)



# Vlaamse Gemeenschap Vlaams Gewest

- Tier-1: BrENIAC
  - KU Leuven
  - 27,000 cores
  - 1 Pflop/s
  - 400 kW



- Tier-2
  - Clusters available in UA, VUB, UGent, KU Leuven and UHasselt

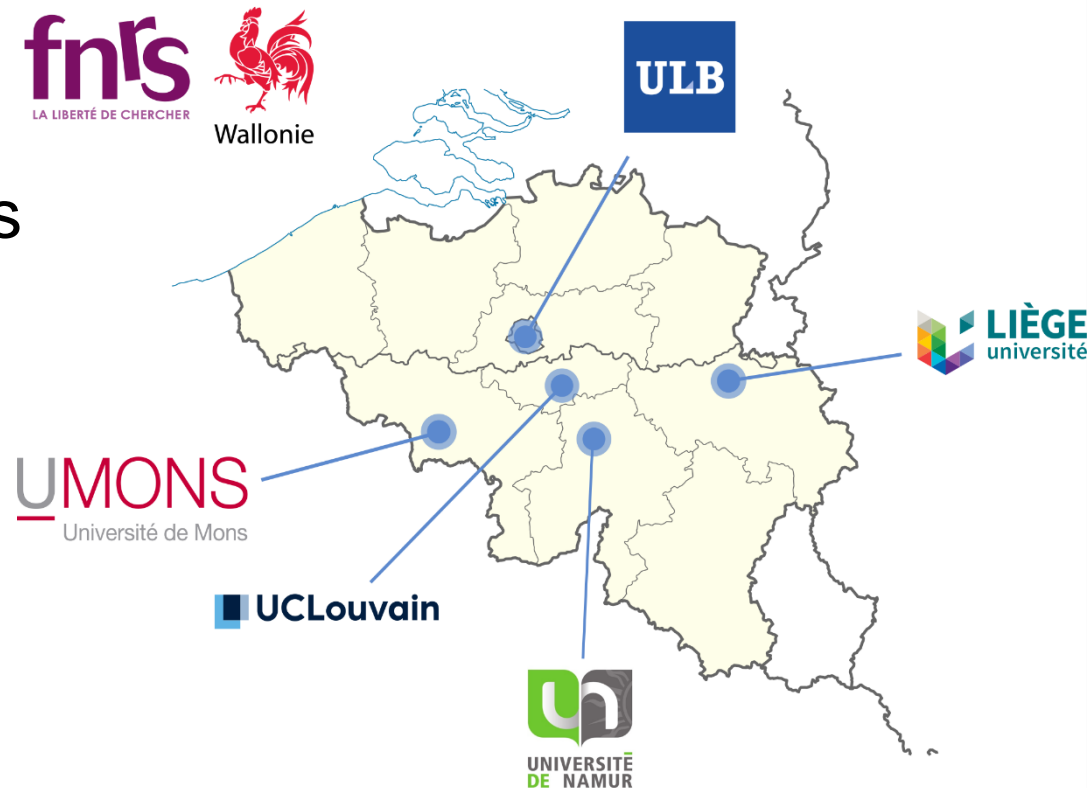
# Fédération Wallonie-Bruxelles

- Tier-1: Zenobe
  - CENAERO
  - 14,000 cores
  - 300 Tflop/s
  - 300 kW
  - 5.5 M€
- Tier-2: CÉCI
  - UCLouvain
  - ULB
  - ULiège
  - UMONS
  - UNamur
  - 100+ Tflop/s





- Consortium des Équipements de Calcul Intensif
- Five universities
- Tier-2 HPC clusters







**Lemaitre 3**  
**1984 cores**  
 Skylake  
 Haswell

95 GB RAM

100Gb/s OPA

**Q2 2018**

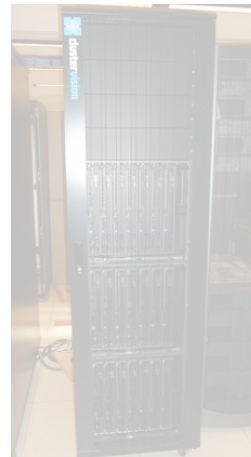


**NIC5**  
**4672 cores**  
 Epyc

1 TB RAM

100Gb/s IB

**Q4 2020**



**Vega**  
**2112 cores**  
 Bulldozer

256 GB RAM

10Gb/s IB

Not available



**Hercules 2**  
**1528 cores**  
 Sandybridge  
 Epyc

2 TB RAM

10 GbE

**Q3 2019**



**Dragon 2**  
**592 cores**  
 Skylake  
 Tesla V100

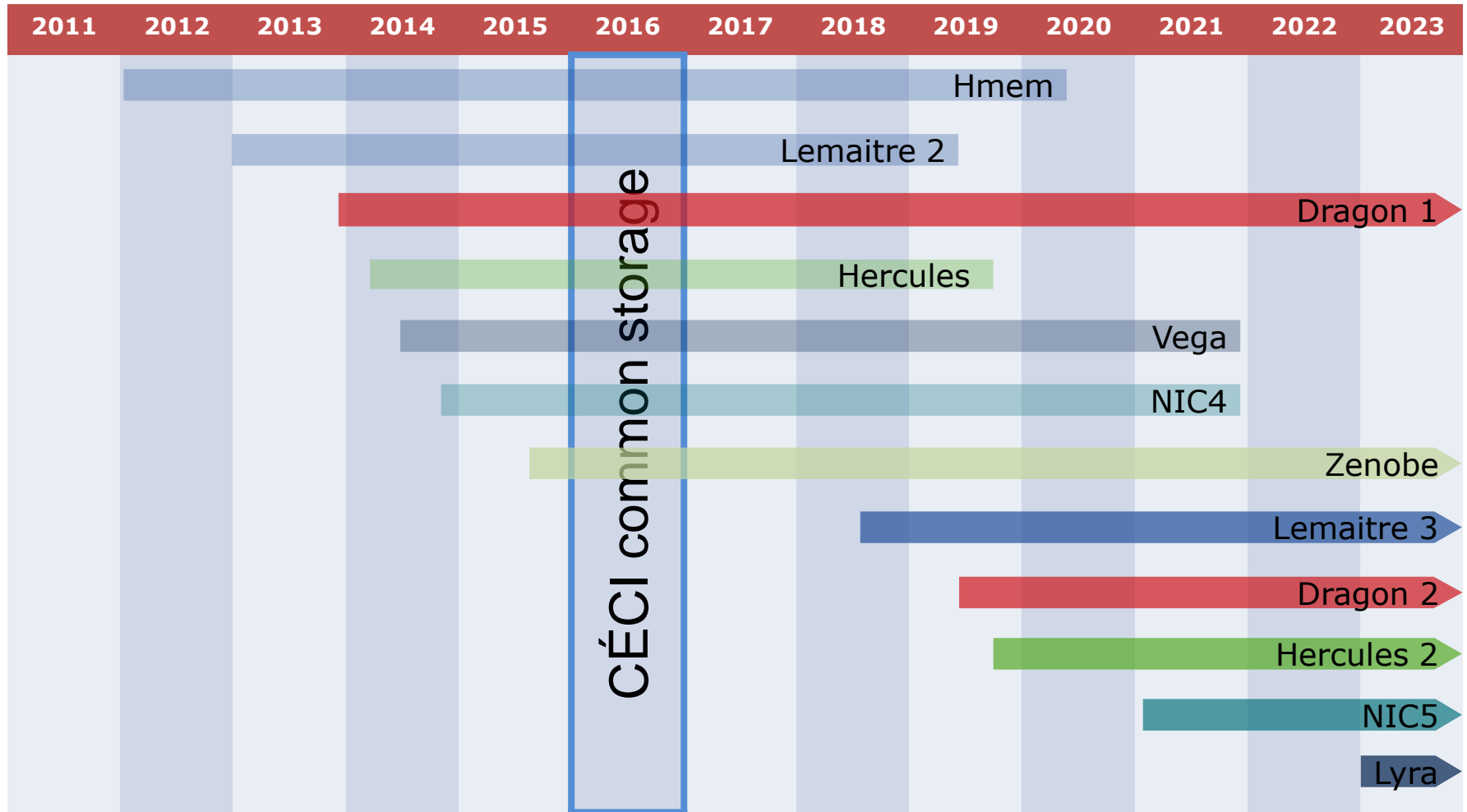
384 GB RAM

10 GbE

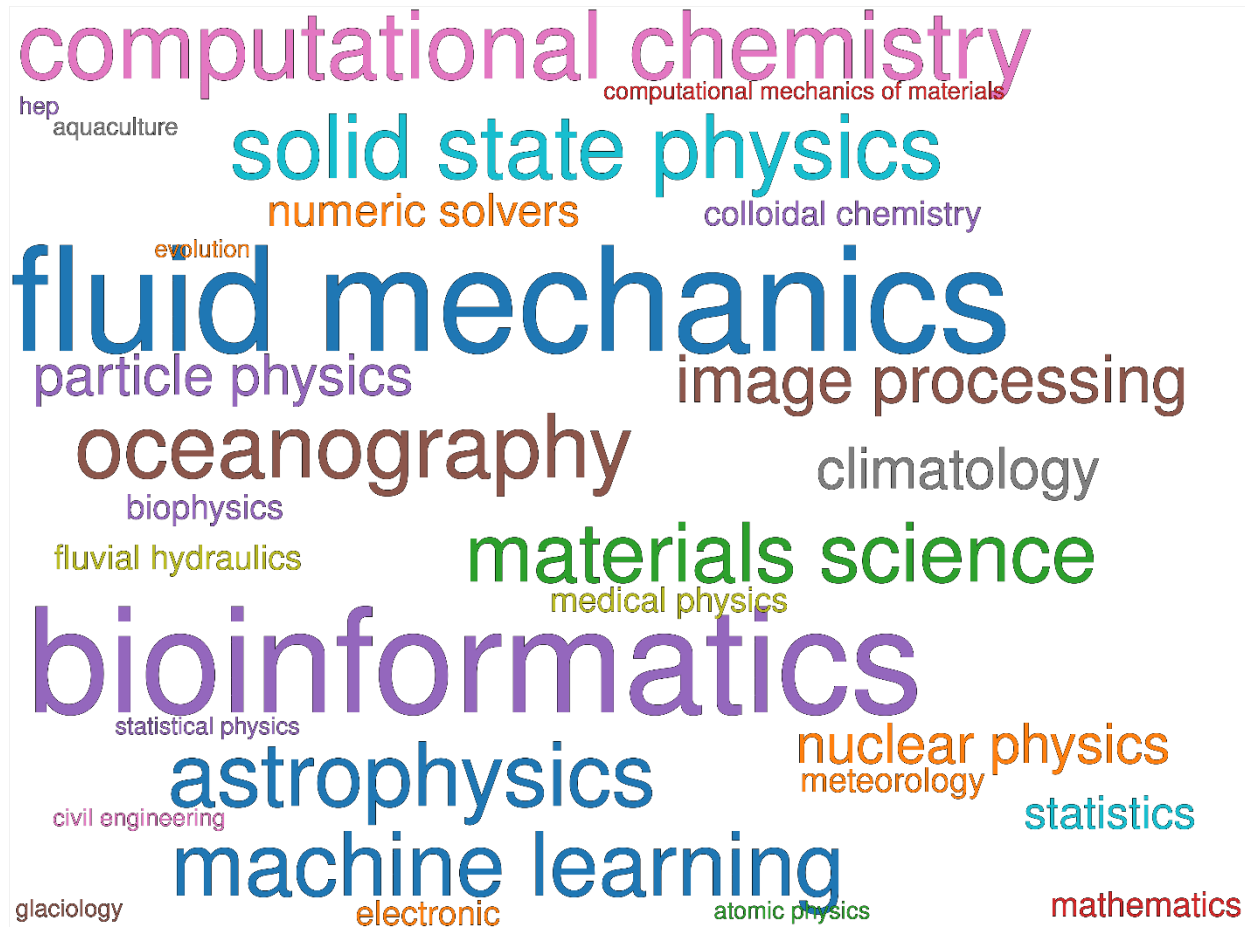
**Q1 2019**

**8776 cores total**

# The CÉCI upgrade



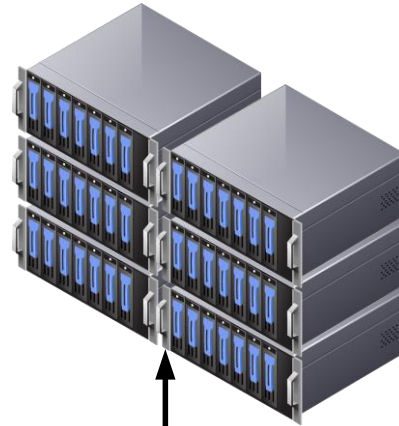
- 400+ CÉCI actives users



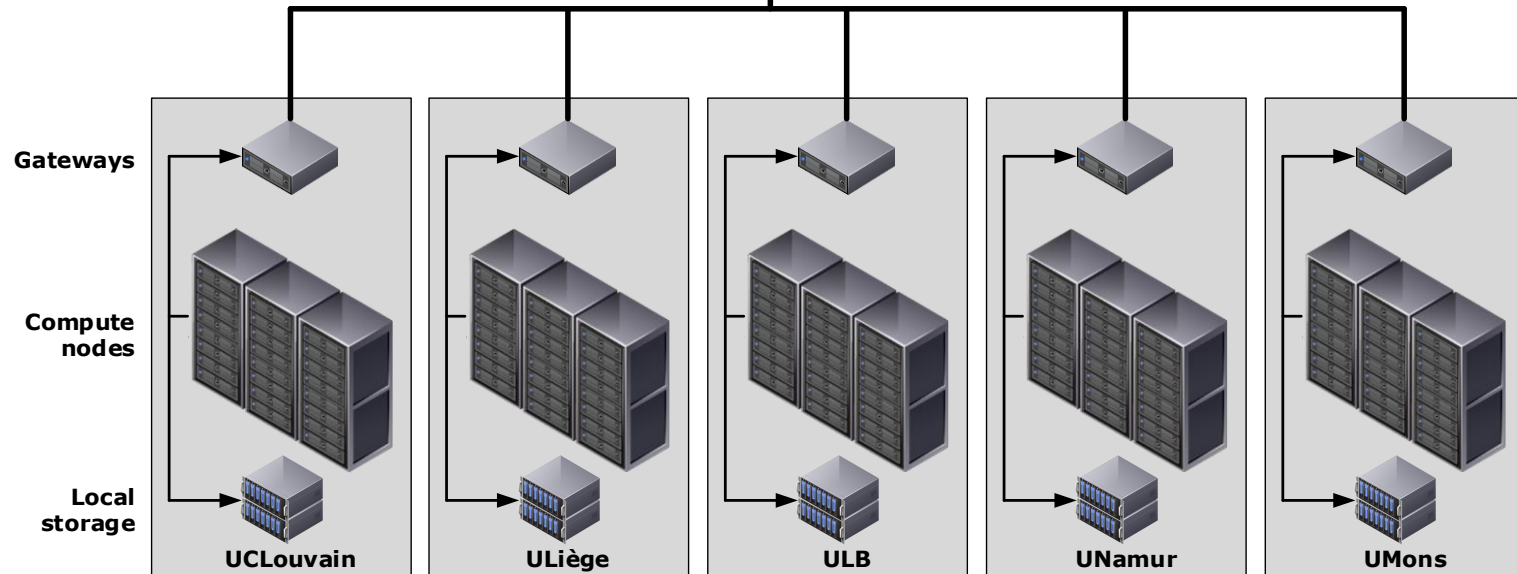
Fields of applications - CÉCI survey 2022

# CÉCI distributed storage

- Distributed storage solution
- Visible from all the frontends and compute nodes of all CÉCI clusters
- 400 TB net



IBM  
Spectrum  
Storage

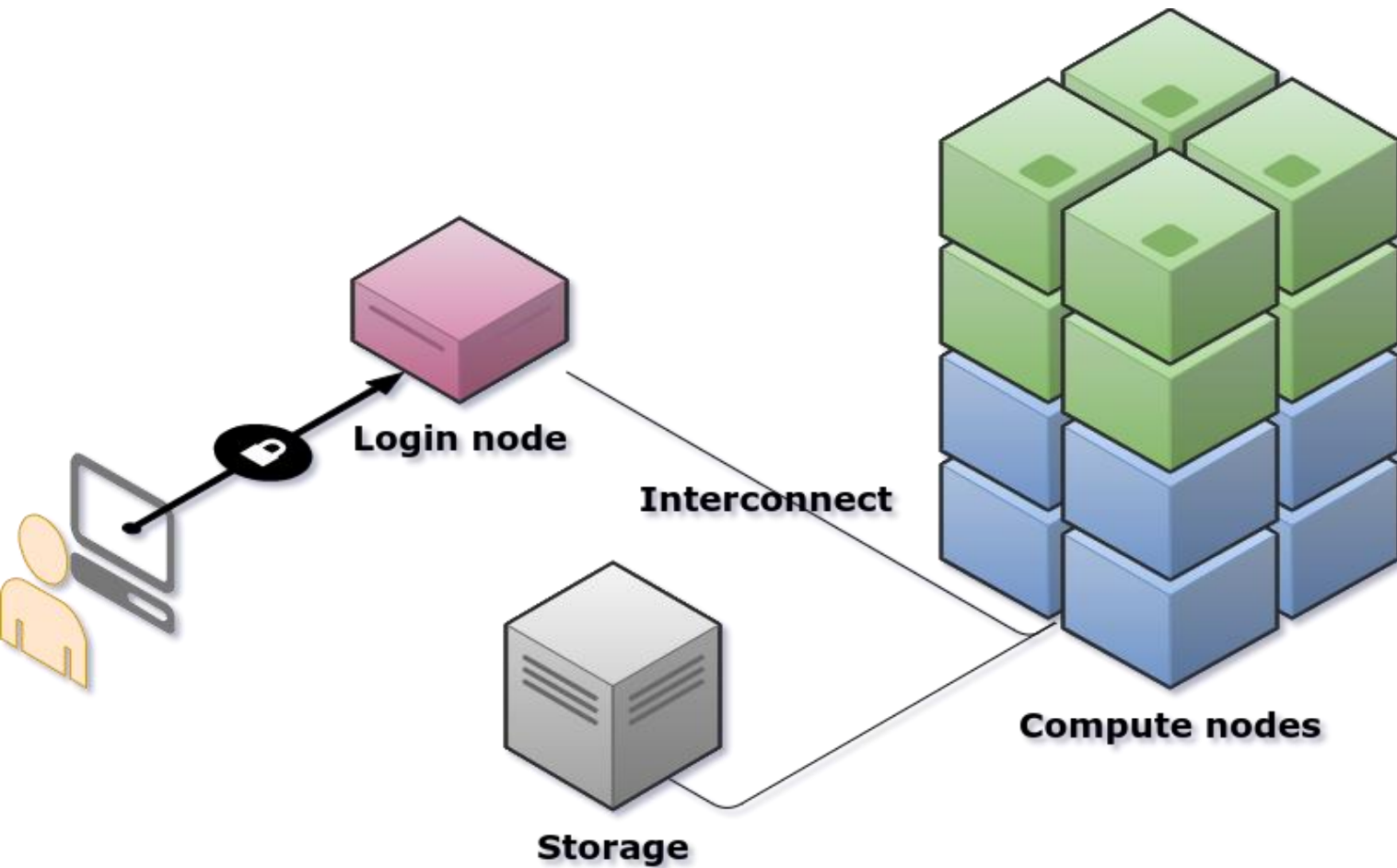


# CÉCI distributed storage



- Common storage directories for all CÉCI clusters
- No need to transfer data between clusters with scp
- Common software repository
- Almost all software installed on any cluster are available on all clusters

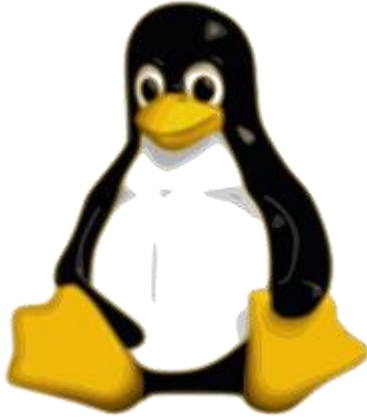
# A cluster in a nutshell



# Login node

- Submit jobs to batch system
- Manage your files
- Interactive work at small scale
- CÉCI login nodes
  - [hercules2.ptci.unamur.be](https://hercules2.ptci.unamur.be)
  - [dragon2.umons.ac.be](https://dragon2.umons.ac.be)
  - [lemaitre3.cism.ucl.ac.be](https://lemaitre3.cism.ucl.ac.be)
  - [nic5segi.ulg.ac.be](https://nic5segi.ulg.ac.be)

# Operating system



- All CÉCI cluster are running GNU/Linux



**CentOS**

- Linux CentOS 7
  - Red Hat Enterprise Linux (RHEL) compatible



# Clusters at CÉCI

The aim of the Consortium is to provide researchers with access to powerful computing equipment ([clusters](#)). Clusters are installed and managed locally at the different sites of the universities taking part in the Consortium, but they are accessible by all researchers from the member universities. A single login/passphrase is used to access all clusters through SSH.

All of them run Linux, and use [Slurm](#) as the job manager. Basic parallel computing libraries (OpenMP, MPI, etc) are installed, as well as the optimized computing subroutines (e.g. BLAS, LAPACK, etc.). Common interpreters such as R, Octave, Python, etc. are also installed. See each cluster's FAQ for more details.

Cluster	Host	CPU type	CPU count*	RAM/node	Network	Filesystem**	Accelerator	Max time	Preferred jobs***
<a href="#">NIC5</a>	ULiège	<a href="#">Rome</a> 2.9 GHz	4672 (73 x 64)	256 GB..1 TB	<a href="#">HDR Ib</a>	<a href="#">BeeGFS</a> 520 TB	None	2 days	☒ MPI
<a href="#">Hercules2</a>	UNamur	<a href="#">Naples</a> 2 GHz <a href="#">SandyBridge</a> 2.20 GHz	1024 (30 x 32 + 2 x 64) 512 (32 x 16)	64 GB..2 TB	<a href="#">10 GbE</a>	<a href="#">NFS</a> 20 TB	None	15 days	☒ serial / ☒ SMP
<a href="#">Dragon2</a>	UMons	<a href="#">SkyLake</a> 2.60 GHz	592 (17 x 32 + 2 x 24)	192..384 GB	<a href="#">10 GbE</a>	<a href="#">RAID0</a> 3.3 TB	4x <a href="#">Volta</a> V100	21 days	☒ serial / ☒ SMP
<a href="#">Lemaitre3</a>	UCL	<a href="#">SkyLake</a> 2.3 GHz <a href="#">Haswell</a> 2.6 GHz	1872 (78 x 24) 112 (4 x 28)	95 GB 64 GB	<a href="#">Omnipath</a>	<a href="#">BeeGFS</a> 440 TB	None	2 days 6 hours	☒ MPI
<a href="#">Dragon1</a>	UMons	<a href="#">SandyBridge</a> 2.60 GHz	416 (26 x 16) 32 (2x16)	128 GB	<a href="#">GbE</a>	<a href="#">RAID0</a> 1.1 TB	4x <a href="#">Tesla</a> C2075, 4x <a href="#">Tesla</a> Kepler K20m	41 days	☒ serial / ☒ SMP
<a href="#">NIC4*</a>	ULiège	<a href="#">SandyBridge</a> 2.0 GHz <a href="#">IvyBridge</a> 2.0 GHz	2048 (120 x 16 + 8 x 16)	64 GB	<a href="#">QDR Ib</a>	<a href="#">FHGFS</a> 144 TB	None	3 days	☒ MPI
<a href="#">Vega*</a>	ULB	<a href="#">Bulldozer</a> 2.1 GHz	896 (14 x 64)	256 GB	<a href="#">QDR Ib</a>	<a href="#">GPFS</a> 70 TB	None	14 days	☒ serial / ☒ SMP / ☒ MPI
<a href="#">Hercules*</a>	UNamur	<a href="#">SandyBridge</a> 2.20 GHz	512 (32 x 16)	64..128 GB	<a href="#">GbE</a>	<a href="#">NFS</a> 20 TB	None	63 days	☒ serial / ☒ SMP
<a href="#">Lemaitre2*</a>	UCL	<a href="#">Westmere</a> 2.53 GHz	1380 (115 x 12)	48 GB	<a href="#">QDR Ib</a>	<a href="#">Lustre</a> 120 TB	3x <a href="#">Quadro</a> Q4000	3 days	☒ MPI
<a href="#">Hmem*</a>	UCL	<a href="#">MagnyCours</a> 2.2 GHz	816 (17 x 48)	128..512 GB	<a href="#">QDR Ib</a>	<a href="#">FHGFS</a> 30 TB	None	15 days	☒ SMP

# Hercules 2 @ UNamur


- High memory jobs
  - up to 2 TB
- Long duration jobs
  - 15 days
- GPU
  - 16 GPUs total
- No multi-node jobs



# HPC @ UNamur

- Local support :
  - Plateforme Technologique en Calcul Intensif (PTCI)
    - Juan CABRERA
    - Frédéric WAUTELET
  - [ptci-support@unamur.be](mailto:ptci-support@unamur.be)
- Other HPC resources
  - Hyades 2
    - 288 cores total
    - Up to 92 GB RAM per node

# Dragon 2 @ UMons

- High performance SMP nodes
- Long duration job
  - 21 days
- GPU
  - 4x  NVIDIA. Volta V100
- No multi-node jobs



# HPC @ UMONS

- Local support
  - [Sebastien.KOZLOWSKYJ@umons.ac.be](mailto:Sebastien.KOZLOWSKYJ@umons.ac.be)
- Other HPC resources
  - Biovia Materials Studio cluster
    - 144 cores total
    - 192 GB RAM per node
  - HTC cluster
    - 512 cores total
    - Up to 256 GB RAM per node



# Lemaitre 3 @ UCLouvain

- Massively parallel jobs
  - MPI
- I/O intensive jobs
- Short duration job
  - 2 days
- Fast parallel filesystem
  - \$GLOBALSCRATCH



# HPC @ UCLouvain

## CISM



• Local support

- Institut de Calcul Intensif et de Stockage de Masse ([egs-cism@listes.uclouvain.be](mailto:egs-cism@listes.uclouvain.be))



Thomas Keutgen  
(Head)



Damien François



Olivier Mattelaer



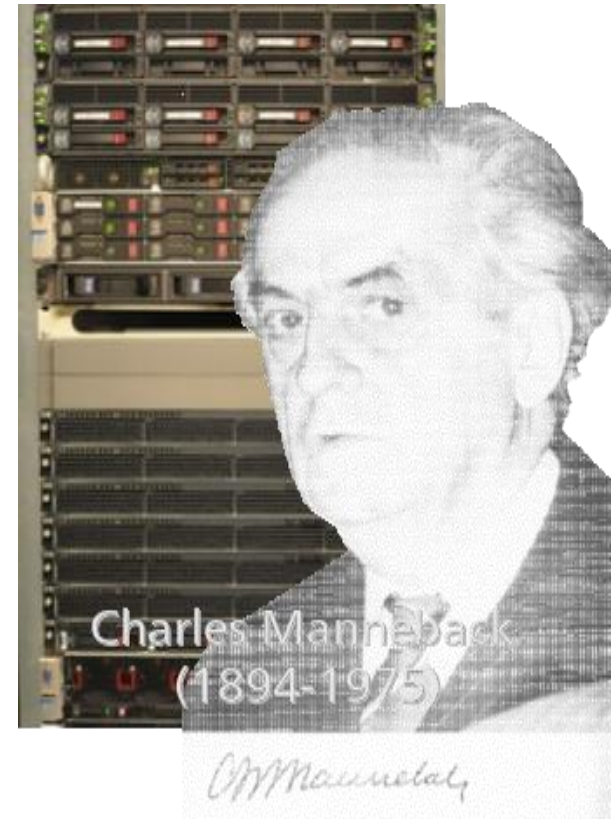
Bernard Van  
Renterghem



Patrick Vranckx



- Other resources
  - Manneback HPC cluster
    - Heterogeneous hardware
    - +5700 cores
    - 82 Tflop/s
  - Mass storage
    - 317 TB storage total





# NIC5 @ ULiège

- Massively parallel jobs
  - MPI
- I/O intensive jobs
- Short duration jobs
  - 2 days
- Fast parallel filesystem
  - \$GLOBALSCRATCH



# HPC @ ULiège

- Local support
  - [David.Colignon@uliege.be](mailto:David.Colignon@uliege.be)
- More info
  - <http://www.ulg.ac.be/nic4>



# Zenobe

- Massively parallel jobs
  - MPI
- I/O intensive jobs
- Very short duration job
  - 1 day



# How to get a CÉCI account?



The screenshot shows the CÉCI website homepage. At the top, there is a navigation bar with links for Clusters, News, Training, FAQ, Documentation, Support, and Contact, along with a 'Create/Manage Account' button. The main header features the C.E.C.I. logo and the text 'Consortium des Équipements de Calcul Intensif' with statistics: 9 clusters, 10k cores, 1 login, 1 home directory. The 'About' section describes CÉCI as a consortium of high-performance computing centers supported by F.R.S-FNRS and the Walloon Region. A map shows member institutions: fnis, ULB, LIÈGE, UMONS, and UCLouvain. The 'Quick links' section lists resources like connection guides and submission wizards. The 'Quick search' section has a search box. The 'Photo Gallery' shows server racks. The 'Latest News' section includes announcements about LEMAITRE3, Dragon1, a survey on Big Data and Machine Learning, the 10th CÉCI Scientific Meeting, and a PRACE call for proposals. The footer contains RSS, copyright, and contact information.

Go to <http://www.ceci-hpc.be>

# Create/Manage Account



The screenshot shows the top navigation bar of the CÉCI website with links for Clusters, News, Training, FAQ, Documentation, Support, and Contact. The main header features the C.E.C.I logo and the text "Consortium des Équipements de Calcul Intensif". A red arrow points from the logo area to a modal dialog box titled "Create/Manage Account". The dialog box contains the text: "To create an account your computer must be connected to your university network." and an "Ok" button.

## About

CÉCI is the 'Consortium des Équipements de Calcul Intensif'; a consortium of high-performance computing centers of UCLouvain, ULB, ULiège, UMONS, and UNamur. The CÉCI is supported by the F.R.S-FNRS and the Walloon Region. [Read more.](#)



## Quick links



## Save the date!

The next CÉCI scientific day will take place on Thursday April 25th in Brussels.  
More information soon!

## Latest News

### LEMAITRE3 installed at UCL

Lemaitre3 is now operational and replaces Lemaitre2, which will be decommissioned this Summer. It has 80 nodes (SkyLake 2.3 GHz, 24CPUs, 96GB RAM) interconnected with the Intel OmniPath Architecture and more than half a petabyte of scratch space.

MONDAY, 04 JUNE 2018

TUESDAY, 05 MAY 2018

### Dragon1 cluster featured in a Belnet article

The Dragon1 CÉCI cluster is highlighted in an interview from Belnet to Chantal Poirat, professor in Information and Communication Technology at the University of Mons.

# I want to... create an account

## I want to...

[create an account](#)

You are about to request an account on the CÉCI clusters.

The first step is to enter your email address. You will receive an email with a link to an online form which you will have to fill and submit.

Once your request has been approved, you will receive proper information on how to access the CÉCI clusters.

[renew my account](#)

[join an existing project](#)

[reset my passphrase](#)

[retrieve my private key](#)

[change my email address](#)

[invite a guest or renew a guest account](#)

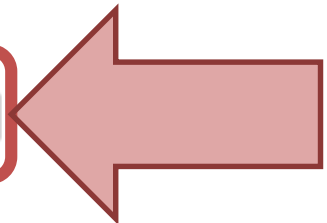
## create an account

**My email address:**

frederic.wautelet

@unamur.be

Send



# That's it

- Click on the link sent to you by email.
- Fill-in the form and hit the “Submit” button
- Get your SSH private key from your email
- Configure your SSH client
- Connect and profit!

# Introduction to Linux and the command line

Julia

Accelerators/Co-processors

Python for HPC

Slurm workload manager

Share memory (OpenMP)

Message passing (MPI)

Checkpointing

Debugging and profiling

Compilers and libraries

**Introduction to GNU/Linux and the command line**

Introduction to HPC



# SSH tools

- Windows
  - PuTTY
  - **MobaXterm**
  - X-Win32
  - OpenSSH on Windows (Windows 10)
- Linux/MacOS
  - ssh
  - scp

# MobaXterm



- Easy to use
- No installation needed
- Command line interface
- Allow use of graphical application remotely
- Files transfer

# Bash



- Shell is the interface between the user and the Linux system
- Interprets and run commands
- For Linux, “Bash” is the default
- Shell scripts

# Modules



- Modify user's environment
- Allow use of application with different versions
- Commands:
  - \$ module load/unload
  - \$ module list
  - \$ module available
  - \$ module spider

# module available



```
----- Meta Modules -----
dot      releases/elic-2017b  releases/2016b  releases/2018a  tis/2018.01 (S,L)
null     releases/2016a         releases/2017b (S,L,D)       releases/2018b  use.own

----- TIS: Toolchain Independent Software (2018.01) -----
EasyBuild/3.5.1  MCR/R2013a  MCR/R2015a  MCR/R2017a  crystal/17-v1.0.1  julia/0.6.3
Java/1.8.0_31    MCR/R2013b  MCR/R2015b  MCR/R2017b  crystal/17-v1.0.2 (D)  julia/1.0.0 (D)
Java/1.8.0_92   MCR/R2014a  MCR/R2016a  MCR/R2018a  (D)  freesurfer/6.0.0  xpress/xp850
Java/1.8.0_121  MCR/R2014b  MCR/R2016b  NCBI-BLAST-database/20170306  gurobi/gurobi800

----- Releases (2017b) -----
ABINIT/8.4.4-intel-2017b  Python/2.7.14-GCCcore-6.4.0-bare
ANTLR/2.7.7-intel-2017b  Python/3.6.3-foss-2017b
Boost/1.65.1-foss-2017b  Python/3.6.3-intel-2017b  (D)
Boost/1.66.0-intel-2017b  (D)  Qhull/2015.2-foss-2017b
CD0/1.9.2-intel-2017b  Qt/4.8.7-foss-2017b
CGAL/4.11-foss-2017b-Python-2.7.14  R/3.4.3-foss-2017b-X11-20171023
CP2K/5.1-intel-2017b  Ruby/2.5.0-intel-2017b
Doxygen/1.8.13-GCCcore-6.4.0  SCOTCH/6.0.4-foss-2017b
Eigen/3.3.4  SCOTCH/6.0.4-intel-2017b  (D)
FFTW/3.3.6-gompi-2017b  SQLite/3.20.1-GCCcore-6.4.0
FFTW/3.3.6-intel-2017b  (D)  SWIG/3.0.12-foss-2017b-Python-2.7.14
FLUENT/14.0  SWIG/3.0.12-foss-2017b-Python-3.6.3
FLUENT/18.2  (D)  SWIG/3.0.12-intel-2017b-Python-3.6.3  (D)
GCC/6.4.0-2.28  ScaLAPACK/2.0.2-gompi-2017b-OpenBLAS-0.2.20
GDAL/2.2.3-foss-2017b-Python-2.7.14  Singularity/2.5.2-foss-2017b
GDAL/2.2.3-foss-2017b-Python-3.6.3  (D)  UDUNITS/2.2.25-intel-2017b
GEOS/3.6.2-foss-2017b-Python-2.7.14  UDUNITS/2.2.26-intel-2017b  (D)
GEOS/3.6.2-foss-2017b-Python-3.6.3  X11/20171023-GCCcore-6.4.0
GEOS/3.6.2-intel-2017b-Python-3.6.3  (D)  YAXT/0.5.1-intel-2017b
GLib/2.53.5-GCCcore-6.4.0  foss/2017b
GMP/6.1.2-GCCcore-6.4.0  gc/7.6.0-GCCcore-6.4.0
GSL/2.4-GCCcore-6.4.0  gflags/2.2.1-intel-2017b
Guile/1.8.8-GCCcore-6.4.0  gompi/2017b
HDF5/1.8.19-intel-2017b  grib_api/1.24.0-intel-2017b

lines 1-36
```

# How to use HPC infrastructure

## Part I

1. Introduction to high-performance computing
2. Connecting with SSH from Windows and Linux on CECI clusters
3. Introduction to Linux and the command line

## Part II

4. Choosing and activating software with system modules on CECI clusters
5. Writing and editing text files with Vim
6. Preparing, submitting and managing jobs with Slurm
7. Using a Checkpoint/restart program to overcome time limits

# Batch jobs

Julia

Accelerators/Co-processors

Python for HPC

Slurm workload manager

Share memory (OpenMP)

Message passing (MPI)

Checkpointing

Debugging and profiling

Compilers and libraries

Introduction to GNU/Linux and the command line

Introduction to HPC

# Interactive or batch

- Interactive
  - Short tasks
  - Tasks that require frequent user interaction
  - Graphically intensive tasks
- Batch
  - Longer running processes
  - Parallel processes
  - Running large numbers of short jobs simultaneously
  - Submitted to a job scheduler



# Job scheduler



- Dispatch the batch jobs on compute nodes
- Parameters
  - Memory
  - Processor type
  - Execution time
  - Number of processors
  - Software license tokens
- Slurm workload manager

# Submit a batch job



- Connect to a login node

```
$ ssh hercules.ptci.unamur.be
```

# Job scripts

- Define resources to be reserved for your job:
  - CPU time
  - memory
  - platform
  - number of CPUs
  - List instructions to be executed
- Bash shell script

# Job scripts



- run.sh

```
#!/bin/bash
#SBATCH --job-name=hello
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --ntasks-per-node=1
#SBATCH --time=1:00:00
#SBATCH --mem-per-cpu=1000

echo "Hello World!"
```

# Submitting jobs



- Submit the job script

```
$ sbatch run.sh
Submitted batch job 3513668
```

- Return the job id

- Job is running

```
$ squeue -u $USER
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST
3513667	cpu	hello	fwautele	R	0:12	1	n065

- Job is finished

```
$ squeue -u $USER
$
```

# Batch jobs



- Check output file

```
$ ls -altr
...
-rw-rw-r-- 1 fwautele fwautele          13 Feb 26 11:16 slurm-3513668.out
```

- Hello world!

```
$ cat slurm-3513668.out
Hello World!
```

# Safeguards



- Slurm will automatically cancel jobs:
  - When the memory reserved is exceeded
  - When time is over
- Slurm constraint job in the number of core requested

# Delete a job



- scancel

```
$ scancel 2243523
```

- You can only delete your own jobs... (hopefully)



# Monitoring jobs



- `squeue`

```
$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST (REASON)
2619747	cpu	PYV3_FBI	jquertin	R	16:15:37	1	n076
2619745	cpu	PYV3_DHB	jquertin	R	4-14:36:35	1	n020
2620638	cpu	PYV3_FA_	jquertin	R	43:33	1	n025
2618213	cpu	PYV3_SDP	jquertin	R	9-19:40:43	1	n054
2620635	cpu	PYV3-CC2	jquertin	R	56:59	1	n020
2620632	cpu	PYV3-CC2	jquertin	R	59:22	1	n014
2620633	cpu	PYV3-CC2	jquertin	R	59:22	1	n014
2620630	cpu	PYV3-CC2	jquertin	R	59:52	1	n054
2620631	cpu	PYV3-CC2	jquertin	R	59:52	1	n064
2620627	cpu	PYV3-CC2	jquertin	R	1:01:24	1	n064
2620628	cpu	PYV3-CC2	jquertin	R	1:01:24	1	n064
2620622	cpu	PYV3-CC2	jquertin	R	1:18:17	1	n076

# scriptgen

- Slurm Script Generation Wizard

<http://www.cec-hpc.be/scriptgen.html>

<p><b>1. Describe your job</b></p> <p>Email address: <input type="text" value="user@example.com"/></p> <p>Job name: <input type="text" value="Some name"/></p> <p>Project: <input type="text" value="Some project"/></p> <p><b>Parallelization paradigm(s)</b></p> <p><input type="checkbox"/> Embarrassingly parallel / Job array</p> <p><input checked="" type="checkbox"/> Shared memory / OpenMP</p> <p><input type="checkbox"/> Message passing / MPI</p> <p><b>Job resources</b></p> <p>Duration : <input type="text" value="0"/> days, <input type="text" value="1"/> hour, <input type="text" value="0"/> minutes.</p> <p>Number of threads : <input type="text" value="2"/></p> <p>Memory per thread: <input type="text" value="32"/> <input type="text" value="GB"/></p> <p><b>Filesystem</b></p> <p>Filesystem: <input type="text" value="\$HOME"/></p> <p>Total CPUs: 2   Total Memory: 65536 MB   Total CPU.Hours: 2</p>	<p><b>2. Choose a cluster</b></p> <p><input type="radio"/> Vega</p> <p><input type="radio"/> Lemaitre3</p> <p><input checked="" type="radio"/> Hercules</p> <p><input type="radio"/> Dragon1</p> <p><input type="radio"/> HMEM</p> <p>NIC4</p> <p>Lemaitre2</p> <p>Zenobe*</p>	<p><b>3. Copy-paste your script</b></p> <pre>#!/bin/bash # Submission script for Hercules #SBATCH --time=01:00:00 # hh:mm:ss # #SBATCH --ntasks=1 #SBATCH --cpus-per-task=2 #SBATCH --mem-per-cpu=32768 # 32GB #SBATCH --partition=cpu  export OMP_NUM_THREADS=2 export MKL_NUM_THREADS=2</pre>
---	--	---

# Array jobs



- Run several instances of the same program with different inputs
- Same allocation options
  - Memory size
  - Time limit
  - ...

# --array options



```
# SBATCH --array=0-31
```

```
# SBATCH --array=1,3,5,7
```

```
# SBATCH --array=1-7:2
```

```
# SBATCH --array=1-15%4
```

# Example



```
$ sbatch --array=0-3 run.sh
Submitted batch job 3512681
```

```
$ squeue -u fwautele
```

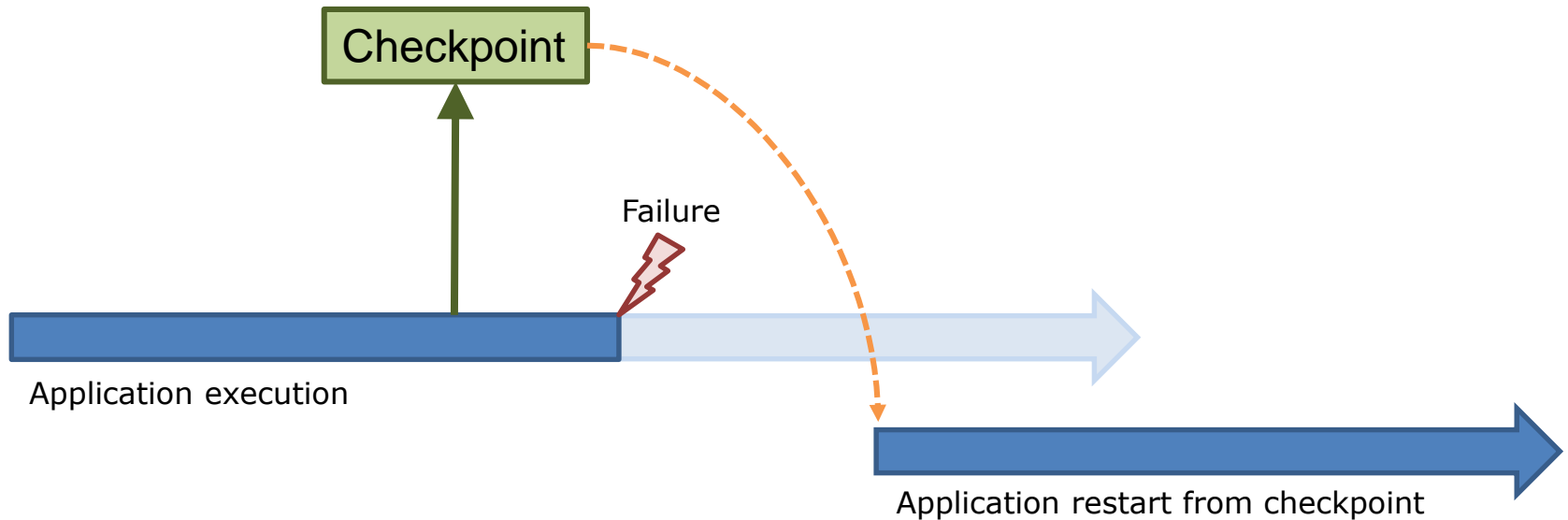
JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST
3512681_0	cpu	run.sh	fwautele	R	0:12	1	n064
3512681_1	cpu	run.sh	fwautele	R	0:12	1	n077
3512681_2	cpu	run.sh	fwautele	R	0:12	1	n047
3512681_3	cpu	run.sh	fwautele	R	0:12	1	n047

# Job Dependencies

- A job can be dependent upon other job(s) status
- Dependency type:
  - after
    - after the specified jobs have started
  - afterany
    - after the specified jobs have terminated
  - afternotok
    - after the specified jobs have failed
  - afterok
    - after the specified jobs have terminated successfully

# Checkpointing

- To overcome job time limitation
- Allow rollback-recovery for long-running applications
- Enable job migration



# How to use HPC infrastructure

## Part I

1. Introduction to high-performance computing
2. Connecting with SSH from Windows and Linux on CECI clusters
3. Introduction to Linux and the command line

## Part II

4. Choosing and activating software with system modules on CECI clusters
5. Writing and editing text files with Vim
6. Preparing, submitting and managing jobs with Slurm
7. Using a Checkpoint/restart program to overcome time limits



# Compilers and libraries

Julia

Accelerators/Co-processors

Python for HPC

Slurm workload manager

Share memory (OpenMP)

Message passing (MPI)

Checkpointing

Debugging and profiling

**Compilers and libraries**

Introduction to GNU/Linux and the command line

Introduction to HPC

# Objectives

- Building from source is preferred in an HPC environment
- Allow users to install applications
  - Link with numerical libraries
  - Built with optimized compiler
- Special case
  - Python
  - R
  - Perl

# Compilers available

- GNU Compiler Collection (GCC)
- Intel Parallel Studio XE Cluster Edition
- The Portland Group PGI Accelerator CDK

# GNU Compiler Collection (GCC)

- Open Source (GPL)
- Pretty good performance
- Compiler suite
  - gcc: C compiler
  - g++: C++ compiler
  - gfortran: Fortran compiler
- module load foss

# Intel Parallel Studio XE Cluster Edition



- Commercial
- High performance compiler
- Compiler suite
  - icc: C compiler
  - icpc: C++ compiler
  - ifort: Fortran compiler
- `module load intel`

# PGI



- The Portland Group PGI Accelerator CDK
- Commercial (NVIDIA)
- Offloading on GPU
- Compiler suite
  - pgcc: C compiler
  - pgCC: C++ compiler
  - pgf77: Fortran 77 compiler
  - pgf90: Fortran 90 compiler
- module load PGI

# Optimized libraries

- Do not reinvent the wheel
- Use multicore-tuned libraries.
- Use optimized libraries
  - Boost
  - FFTW
  - GMP
  - GSL
  - HDF5
  - ...

# Compiler Toolchains

- Compiler toolchain =
  - Compiler
  - + MPI library
  - + BLAS/LAPACK library
    - linear algebra routines
  - + FFT library
    - Fast Fourier Transforms
- Examples
  - foss/2021b
  - intel/2021b



# Compiler Toolchains

- Open Source compiler toolchain
  - foss/2021b
    - GCC 11.2.0
    - OpenMPI 4.1.1
    - OpenBLAS 0.3.18 (including LAPACK)
    - FlexiBLAS 3.0.4
    - ScaLAPACK 2.1.0
    - FFTW 3.3.10

# Compiler Toolchains

- Intel Parallel Studio XE Cluster Edition 2021
  - A toochain: intel/2021b
    - icc 2021.4.0 (C compiler)
    - icpc 2021.4.0 (C++ compiler)
    - ifort 2021.4.0 (Fortran compiler)
    - impi 2021.4.0 (Intel MPI)
    - MKL 2021.4.0 (Math Kernel Library)

# How to program on HPC cluster

## **Part I: Introduction**

1. Introduction to scientific software development and deployment
2. Introduction to scripting and interpreted languages (Python, R, Octave)
3. How to speed up you code at low cost?

## **Part II: Fortran/C/Julia**

4. Introduction to C programming language
5. Introduction to JULIA
6. Introduction to structured programming with Fortran

# Scripting languages

Julia

Accelerators/Co-processors

Python for HPC

Slurm workload manager

Share memory (OpenMP)

Message passing (MPI)

Checkpointing

Debugging and profiling

Compilers and libraries

Introduction to GNU/Linux and the command line

Introduction to HPC

# Python



- Python 2 (deprecated)
  - Python/2.7.16-GCCcore-8.3.0
  - Python/2.7.18-GCCcore-9.3.0
  - Python/2.7.18-GCCcore-10.2.0
- Python 3
  - Python/3.8.6-GCCcore-10.2.0
  - Python/3.9.5-GCCcore-10.3.0
  - Python/3.9.6-GCCcore-11.2.0

# Installing languages extensions



- Install with PIP
  - PIP is the easiest and recommended way to install Python packages

```
$ pip install --user example
```

- Install from source
  - If package not available on PIP
  - Steps:
    - Download the source and unpack it
    - Change to the source directory
    - `python setup.py install --prefix=$HOME/.local`

```
$ python setup.py install --prefix=$HOME/.local
```

# R

- Available versions
  - R/4.0.0-foss-2020a
  - R/4.0.3-foss-2020b
  - R/4.1.0-foss-2021a
- Already bundle with a set of libraries
  - Type “`installed.packages()`” to list them
- Additional libraries
  - R-bundle-Bioconductor/3.11-foss-2020a-R-4.0.0
  - R-bundle-Bioconductor/3.12-foss-2020b-R-4.0.3
  - R-bundle-Bioconductor/3.13-foss-2021a-R-4.1.0.eb

# Octave

- Interactive programming language
- Suited for numerical calculations
- Alternative to MATLAB
  
- Available version(s)
  - Octave/5.1.0-foss-2019b



# Julia

- General-purpose and high-level as Python
- Interactive as R
- But fast as C
  
- Available versions
  - Julia/1.5.1-linux-x86\_64
  - Julia/1.6.7-linux-x86\_64
  - Julia/1.8.2-linux-x86\_64

# How to program on HPC cluster

## **Part I: Introduction**

1. Introduction to scientific software development and deployment
2. Introduction to scripting and interpreted languages (Python, R, Octave)
3. How to speed up you code at low cost?

## **Part II: Fortran/C/Julia**

4. Introduction to C programming language
5. Introduction to JULIA
6. Introduction to structured programming with Fortran

# How to program on HPC cluster

## Learning How to program on HPC cluster (cont'd)

### Part III: Python

7. Introduction to Python
8. Python as an Object Oriented Language
9. Efficient use of Python on the cluster

### Part IV: Tools

10. Introduction to code versioning
11. Debugging and profiling scientific code, and commercial optimized libraries

# Debugging and profiling

Julia

Accelerators/Co-processors

Python for HPC

Slurm workload manager

Share memory (OpenMP)

Message passing (MPI)

Checkpointing

**Debugging and profiling**

Compilers and libraries

Introduction to GNU/Linux and the command line

Introduction to HPC

# Profiling = finding hotspots

- Hotspot = Where in an application or system there is a significant amount of activity
  - Where: address in memory → line of source code
  - Significant: activity that occurs infrequently probably does not have much impact on system performance
  - Activity: time spent or other internal processor event

# Intel Vtune™ Amplifier

- What is the VTune™ Performance Analyzer?
  - Helps you identify and characterize performance issues by:
    - Collecting performance data
    - Organizing and displaying the data from system-wide down to source code or processor instruction
    - Identifying potential performance issues and suggesting improvements
  - Able to analyse serial, OpenMP and MPI application

```
$ ml load VTune
```

# How to program on HPC cluster

## Part III: Python

7. Introduction to Python
8. Python as an Object Oriented Language
9. Efficient use of Python on the cluster

## Part IV: Tools

10. Introduction to code versioning
11. Debugging and profiling scientific code, and commercial optimized libraries

# Parallel computing

Julia

Accelerators/Co-processors

Python for HPC

Slurm workload manager

Share memory (OpenMP)

Message passing (MPI)

Checkpointing

Debugging and profiling

Compilers and libraries

Introduction to GNU/Linux and the command line

Introduction to HPC



# Job type

- Sequential job
  - A single core on one node
- Threaded jobs
  - Several cores on one node
  - OpenMP
- MPI jobs
  - Several cores on several nodes
  - OpenMPI, MPICH, ...

# Going Parallel

## Part I

1. Introduction to parallel computing
2. Parallel programming with MPI

## Part II

4. Parallel programming with OpenMP
5. Directive Based Parallel programming on GPU (OpenACC)
6. Parallel programming on GPU with CUDA

# Accelerators

Julia

Accelerators/Co-processors

Python for HPC

Slurm workload manager

Share memory (OpenMP)

Message passing (MPI)

Checkpointing

Debugging and profiling

Compilers and libraries

Introduction to GNU/Linux and the command line

Introduction to HPC

# Accelerators

- Hardware component with a specialized microprocessor
- Mostly General Purpose Graphical Processing Units (GPGPUs)
- Offer excellent floating point performance per Watt
- Parts of computation “offloaded” to accelerator

# GPGPUs resources at CÉCI

Cluster	Model	Cores	Memory	Float performance (FP32)	Double performance (FP64)
Dragon2	4 x NVIDIA Tesla V100	5120	16 GB	14 TFLOPS	<u>7 TFLOPS</u>
Hercules2	4 x NVIDIA RTX A6000	10752	48 GB	<u>40 TFLOPS</u>	1 TFLOPS
	8 x NVIDIA Tesla A40	10752	48 GB	<u>38 TFLOPS</u>	600 GFLOPS

# Going Parallel

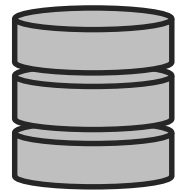
## Part I

1. Introduction to parallel computing
2. Parallel programming with MPI

## Part II

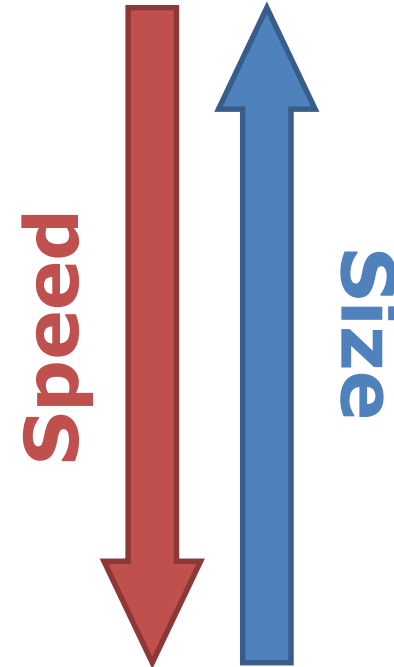
4. Parallel programming with OpenMP
5. Directive Based Parallel programming on GPU (OpenACC)
6. Parallel programming on GPU with CUDA

# Four levels of storage



Data storage

- `$CECIHOME`
  - 400 TB
  - CÉCI distributed storage
- `$HOME`
  - Programs and scripts
- `$WORKDIR`
  - Input and output data
- `$LOCALSCRATCH` or `$GLOBALSCRATCH`
  - Job temporary data



# Scientific data: text or binary?

**Table 5**  
*Simulation results for using full data, CRs only, and proposed method under four missing mechanisms*

Method	Bias <sup>a</sup>		Variance <sup>b</sup>		95% CI <sup>c</sup>	
	$(\hat{\beta}_W)$	$(\hat{\beta}_X)$	$(\hat{\beta}_W)$	$(\hat{\beta}_X)$	$(\hat{\beta}_W)$	$(\hat{\beta}_X)$
<b>(M.1)</b> $P(R = 1) = 0.66$						
Full	0.01346	0.02229	0.04008	0.03685	0.955	0.950
Comp	0.03062	-0.003561	0.1149	0.06732	0.960	0.955
Impu	0.01431	0.021	0.04088	0.05169	0.980	0.975
<b>(M.2)</b> logit $P(R = 1) = 2Y$						
Full	0.007908	-0.02116	0.03838	0.03624	0.975	0.925
Comp	0.01945	0.07096	0.107	0.06581	0.960	0.950
Impu	0.006966	0.01597	0.04227	0.05226	0.975	0.985
<b>(M.3)</b> logit $P(R = 1) = 2X$						
Full	0.007908	-0.02116	0.03838	0.03624	0.975	0.925
Comp	0.01225	0.0589	0.08856	0.06818	0.980	0.975
Impu	0.009563	-0.04699	0.03865	0.04923	0.985	0.970
<b>(M.4)</b> logit $P(R = 1) = X + Y$						
Full	0.01346	0.02229	0.04008	0.03685	0.955	0.950
Comp	0.02404	1.613	0.1102	0.08202	0.955	0.580
Impu	0.01814	0.08289	0.0578	0.06075	0.955	0.970

<sup>a</sup>Bias =  $(\hat{\beta} - \beta_0)/\beta_0$ .

<sup>b</sup>Simulation variance.

<sup>c</sup>Confidence interval using jackknife standard error.



# Scientific data

- What is scientific data?
  - N-dimensional arrays + metadata:
    - Measurements at specific time, location, condition
    - Physics: temperature, pressure
    - Chemistry: reaction speed
    - Biology: type (species, cell types, nucleotides)
    - Economics: price
    - ...

# Example

- Problem:



- Example:
  - A data crushing software written in Fortran generate results
  - A post-processing application written in Python read this results

# Solution 1: Text file

- Pro:
  - Human readable
  - Easy to write
  - Platform independent (Endianness)
  - Very flexible
  - Easy to add a variable
- Cons:
  - Sometime hard to parse
  - No accuracy
  - Performance problem
  - Data size

# Solution 2: NetCDF

- NetCDF (Network Common Data Form)
- For array oriented scientific data
- Available in many programming and scripting languages
  - C++, Java, Fortran, Perl, Python, R, ...
- Emphasizes simplicity over power (unlike HDF5)

# Solution 3: HDF5

- Open file format
- Can represent very complex data objects
  - Like a files hierarchy
- No limit on the number or size of data objects
- Allow access time and storage space optimizations
- Many tools available

# HDFView

The screenshot shows the HDFView application window. The title bar reads "HDFView". The menu bar includes "File", "Window", "Tools", and "Help". The address bar shows the file path "/home/fwautele/test.h5".

The left sidebar displays a tree view of the file structure:

- test.h5
  - dataset1
  - dataset2
  - group1
    - dataset1.1
    - group1.2
      - dataset1.2.1
  - group2
    - dataset2.1

TableView - dataset1 - / - /home... ↗

	0	1	2
0	1	0	0
1	0	1	0
2	0	0	1

TableView - dataset1.1 ... ↗

	0	1
0	1.0	1.0
1	1.0	1.0

TableView - dataset1.2.1 ... ↗

	0	1
0	0.7749942	0.165297...
1	0.912218...	0.3192058
2	0.329779...	0.204235...
3	0.7672147	0.069973...

dataset2 (14058)  
String, length = 4, 2 x 2  
Number of attributes = 0

Log Info Metadata

<https://www.hdfgroup.org/downloads/hdfview/>

And also...

Data versioning



Open Science



# Data Management

1. Introduction to data storage and access
2. Efficient data storage on CECI clusters
3. Open Science and Open Research Data / Data Management Plan
4. Data versioning



# Green HPC



- Green500
  - Rank supercomputers in terms of energy efficiency
  - Performance per Watt (GFLOPS/W)
  - <https://www.top500.org/lists/green500>



Frontier, Oak Ridge National Laboratory, USA

# A green supercomputer: LUMI

- 200,000 cores
- Negative carbon footprint
- 100% renewable energy
- Wasted heat can be used by 20% of the houses of the surrounding city



# Carbon footprint of your computation

- <http://www.green-algorithms.org>

## Green Algorithms

How green are your computations?

### Details about your algorithm

To understand how each parameter impacts your carbon footprint, check out the formula below and the [methods article](#).

Runtime (HH:MM)


Type of cores

Number of cores


Model

Memory available (in GB)


Select the platform used for the computations




**303.10 g CO<sub>2</sub>e**  
Carbon footprint




**2.28 kWh**  
Energy needed



**0.33 tree-months**  
Carbon sequestration



**1.73 km**  
in a passenger car



**1 %**  
of a flight Paris-London

Share your results with [this link!](#)



Thanks you for your attention  
and happy computing