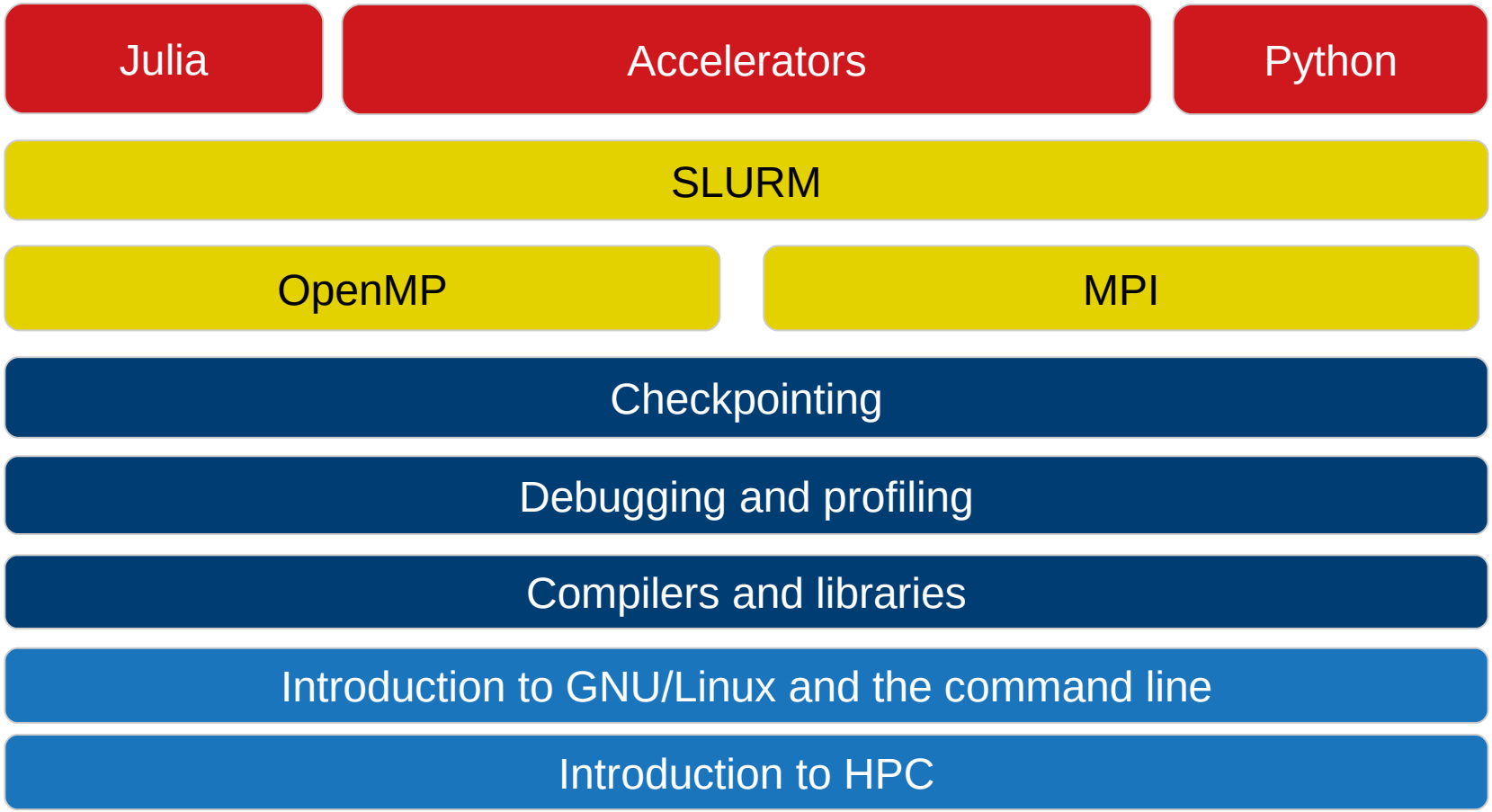
Introduction to high-performance computing

Frédéric Wautelet

CÉCI HPC training 2022

UNIVERSITÉ DE NAMUR

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

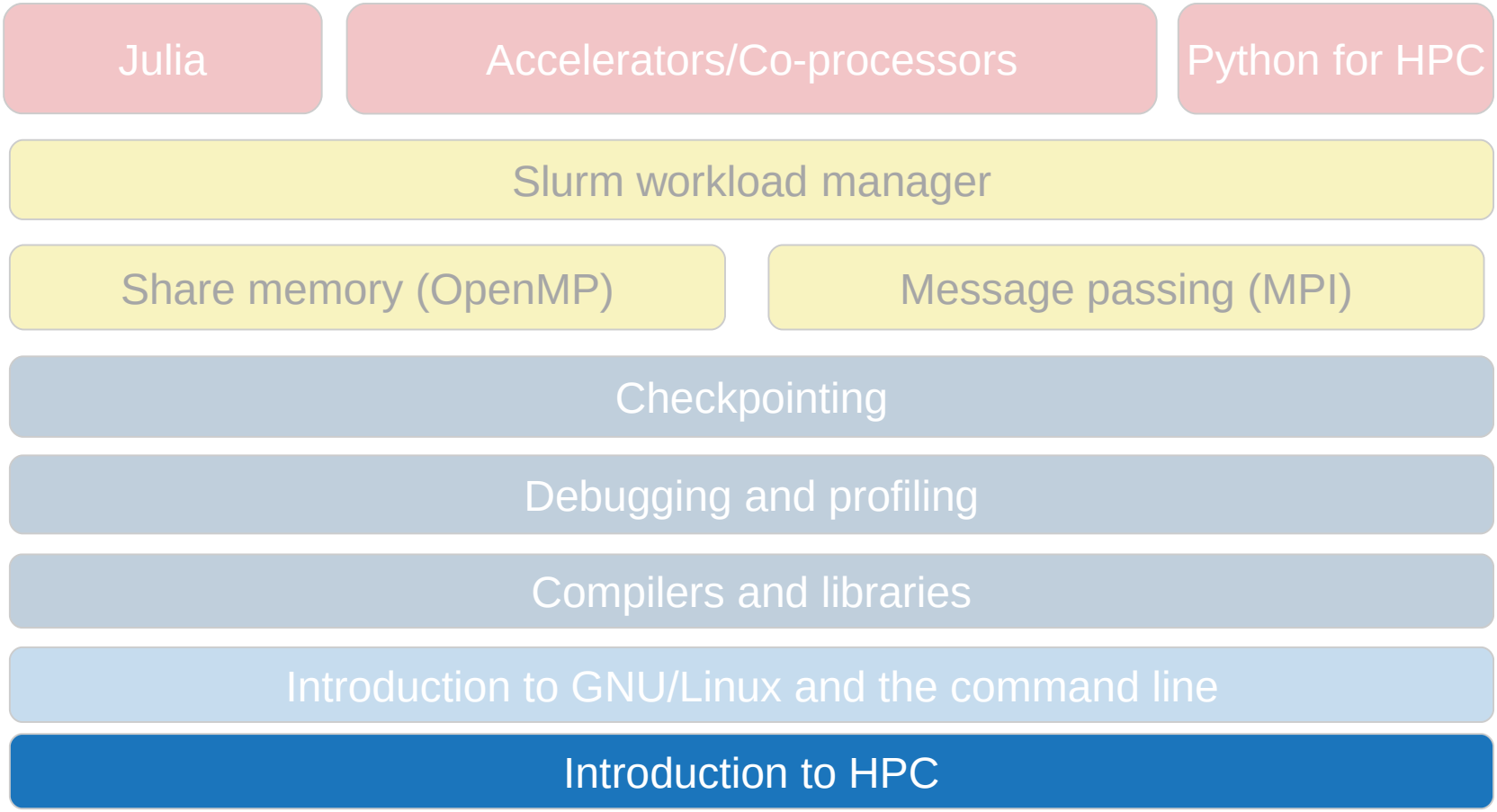
Learning How to program on HPC cluster

Going Parallel

Data Management

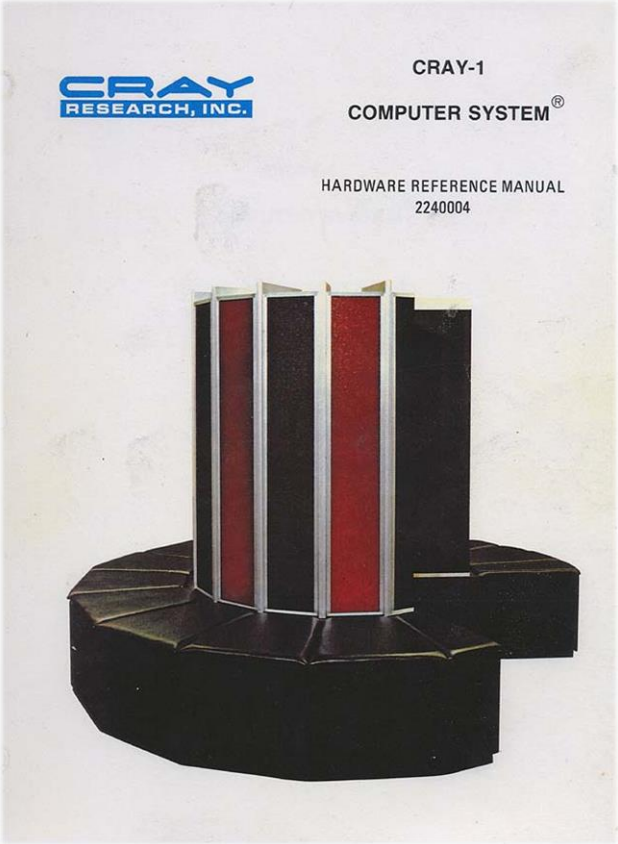
Data Science (TBC)

Introduction to HPC



High Performance Computing

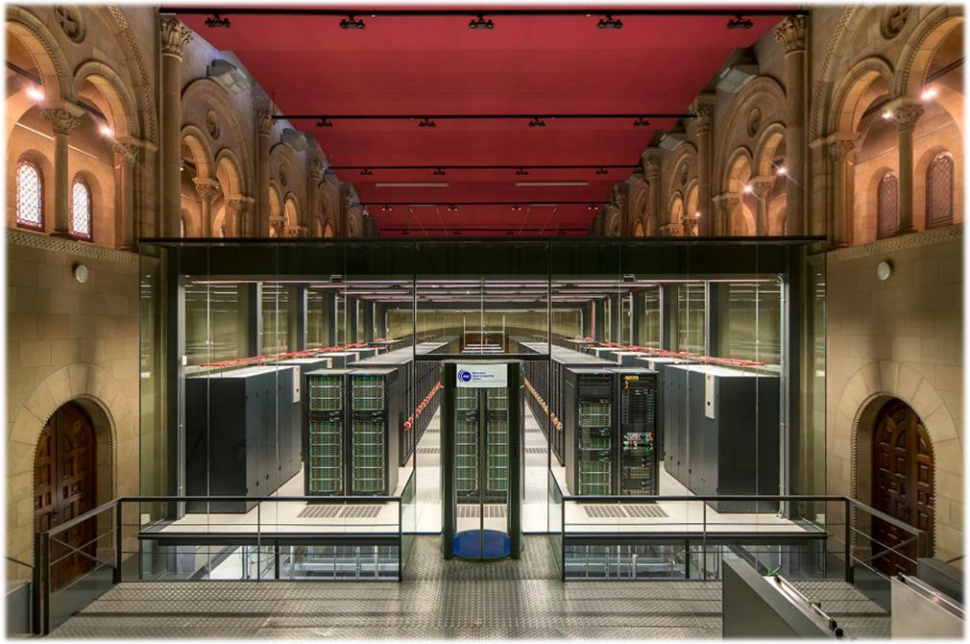
- High-performance computing (HPC) uses supercomputers and computer clusters to solve advanced computation problems.



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.



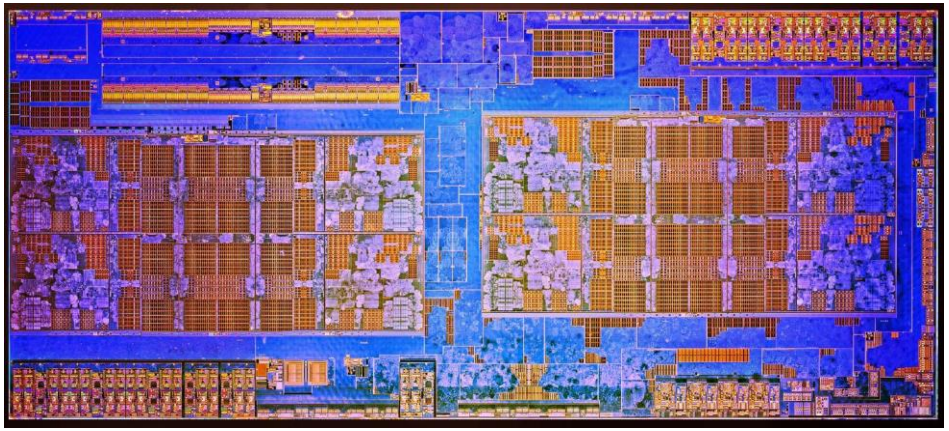
MareNostrum 4 (2017)
13.7 PFlops

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
- **fl**oating-point **o**perations **p**er **s**econd

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

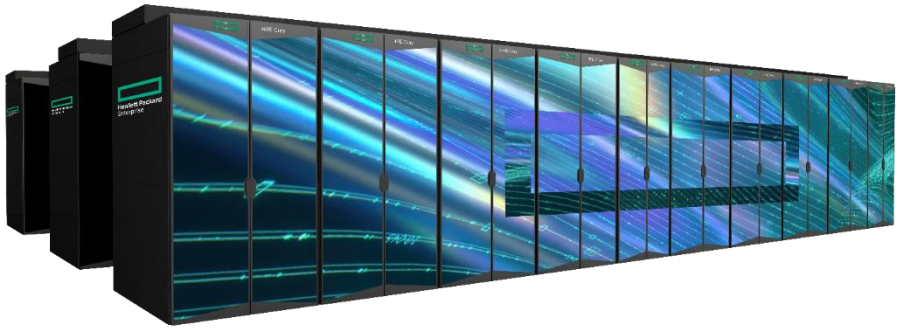
- Frontier
  - Oak Ridge National Laboratory, Tennessee, USA
  - TOP500 #1 (June 2022)
  - 8,700,000 cores
  - First Exascale supercomputer: 1.1 Eflop/s
  - 21 MW
  - US$600M

Exascale in Europe

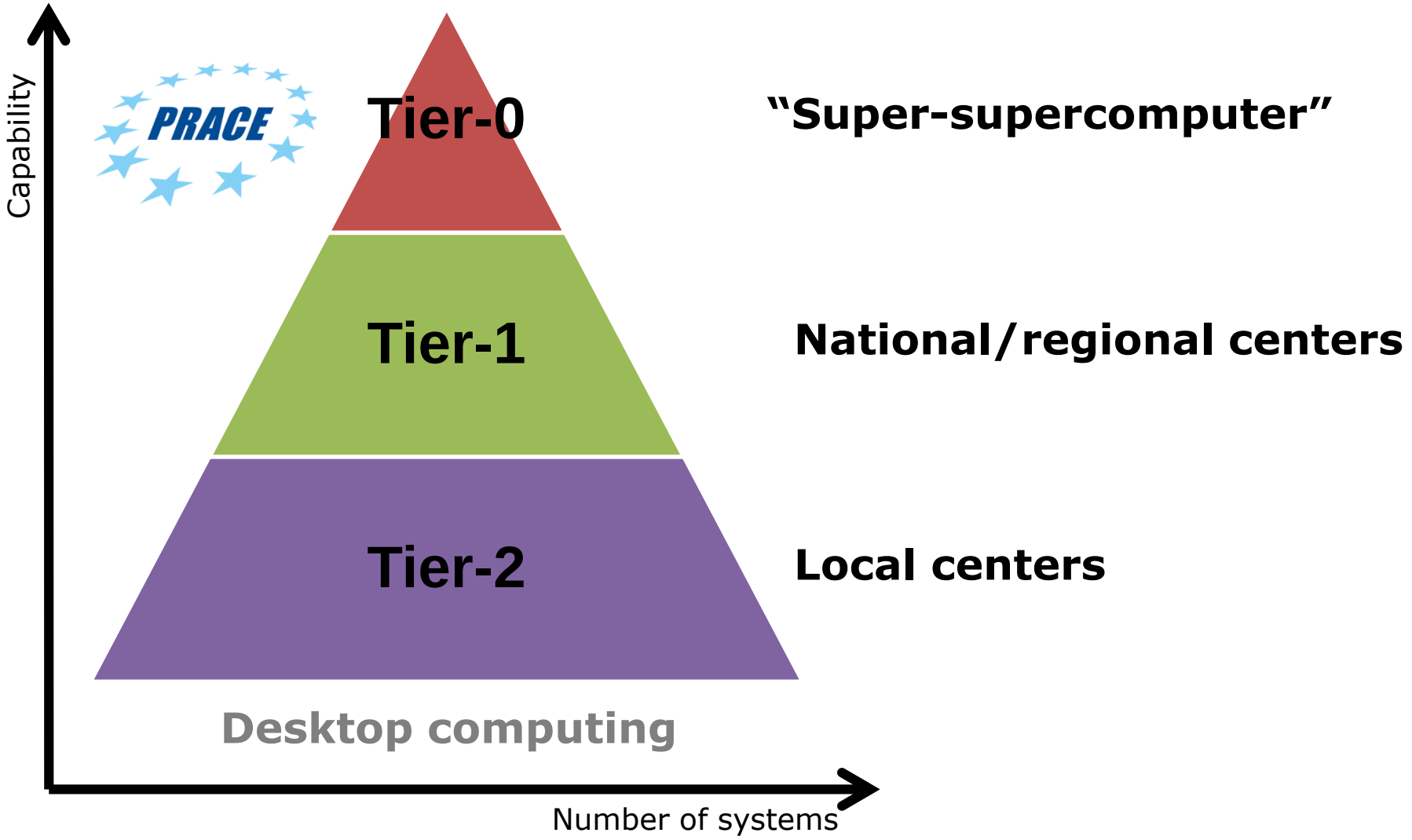
EuroHPC Joint Undertaking

- 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

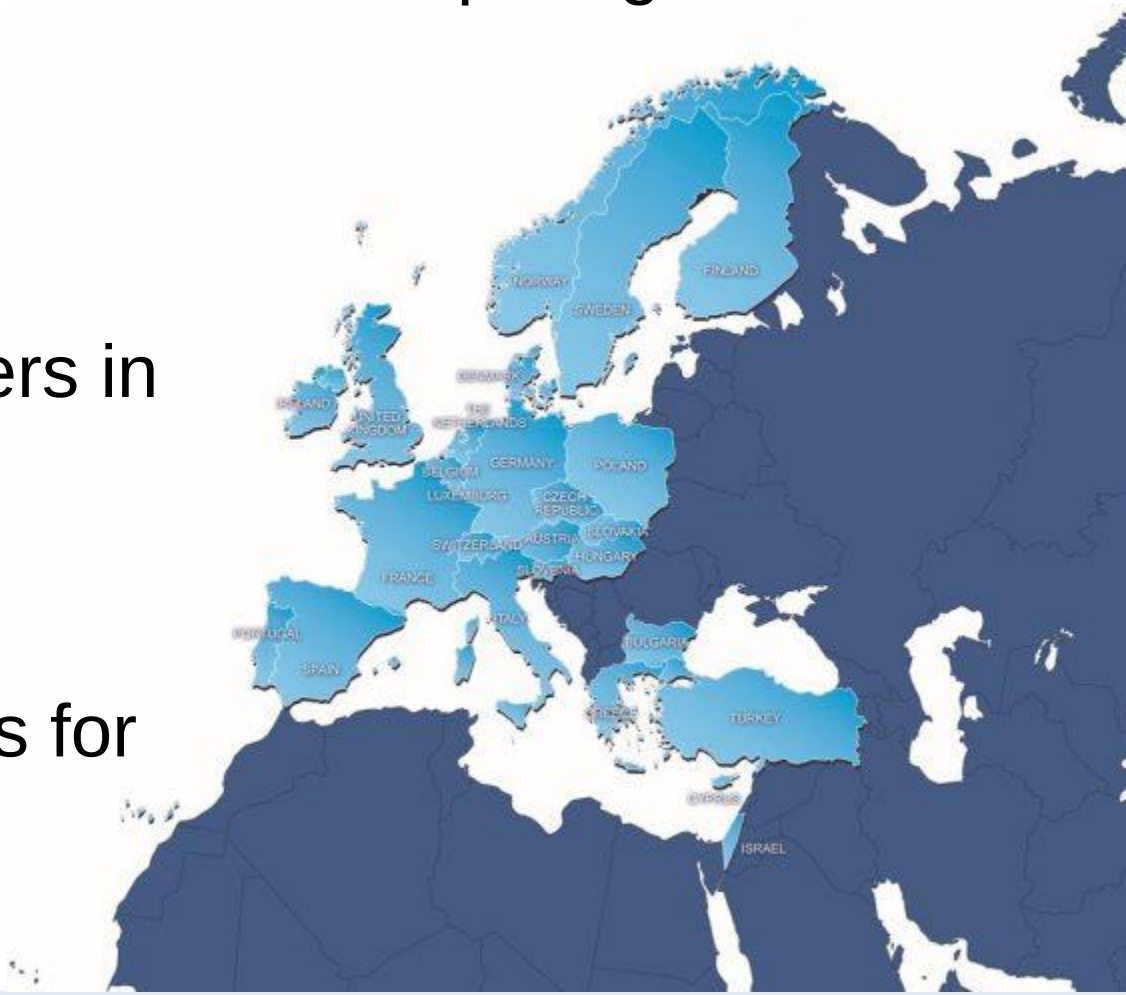


PRACE

- Partnership for Advanced Computing in Europe

- 28 supercomputers in 26 countries

- Call for Proposals for Project Access



Tier-0

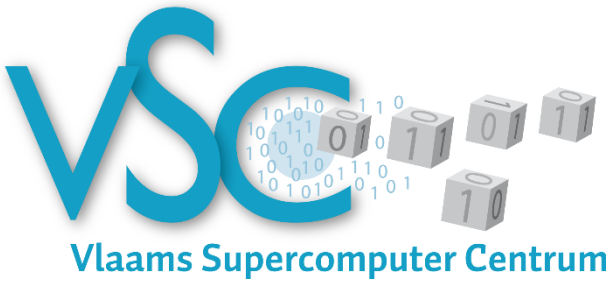
- JUWELS Booster Module
  - Forschungszentrum Juelich, Germany
  - TOP500 #11 (June 2022)
  - ~450,000 cores
  - 44 Pflop/s
  - 2 MW

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

Vlaams Supercomputer Centrum

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

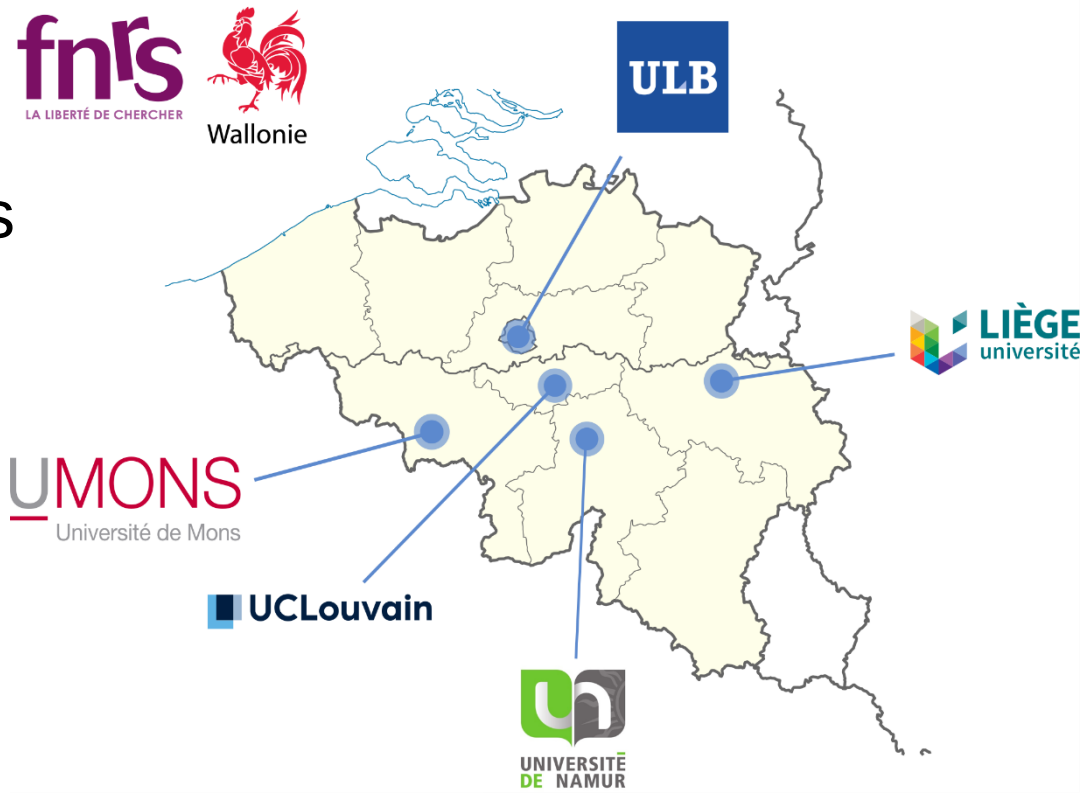
- Tier-2
  - Clusters available in UA, VUB, UGent, KU Leuven and UHasselt

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

CÉCI

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



C.E.C.I

UCLouvain Université | LIÈGE université | ULB | UNIVERSITÉ DE NAMUR | UMONS

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

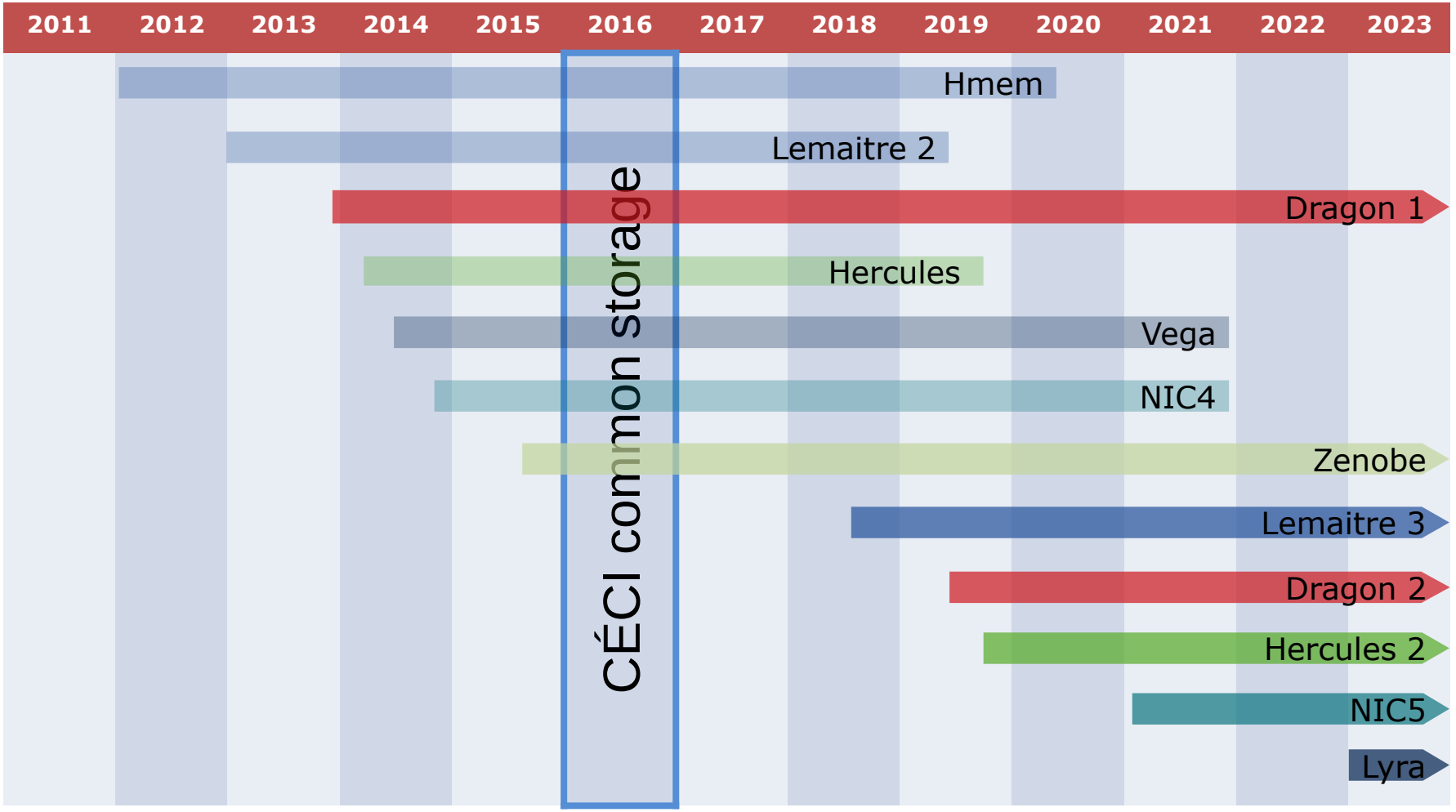
10 GbE

Q1 2019

8776 cores total

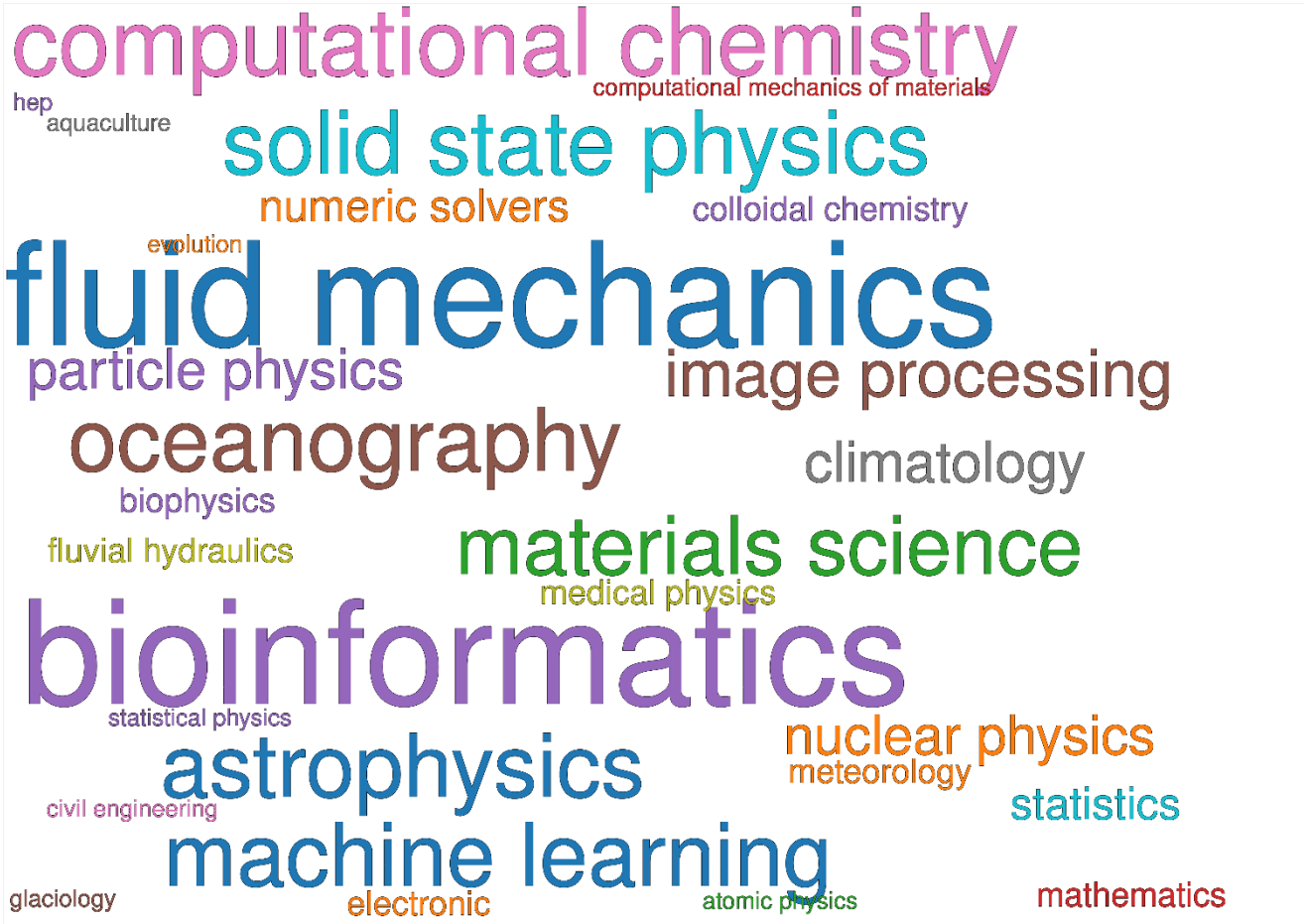
C.E.C.I

The CÉCI upgrade



Users

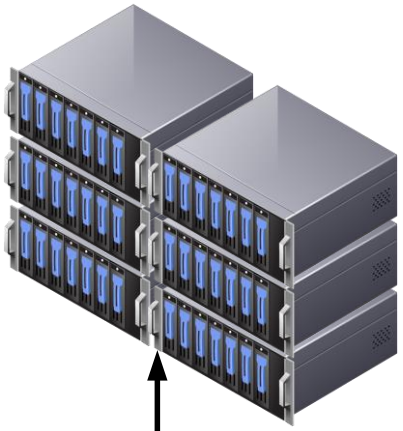
- 400+ CÉCI actives users



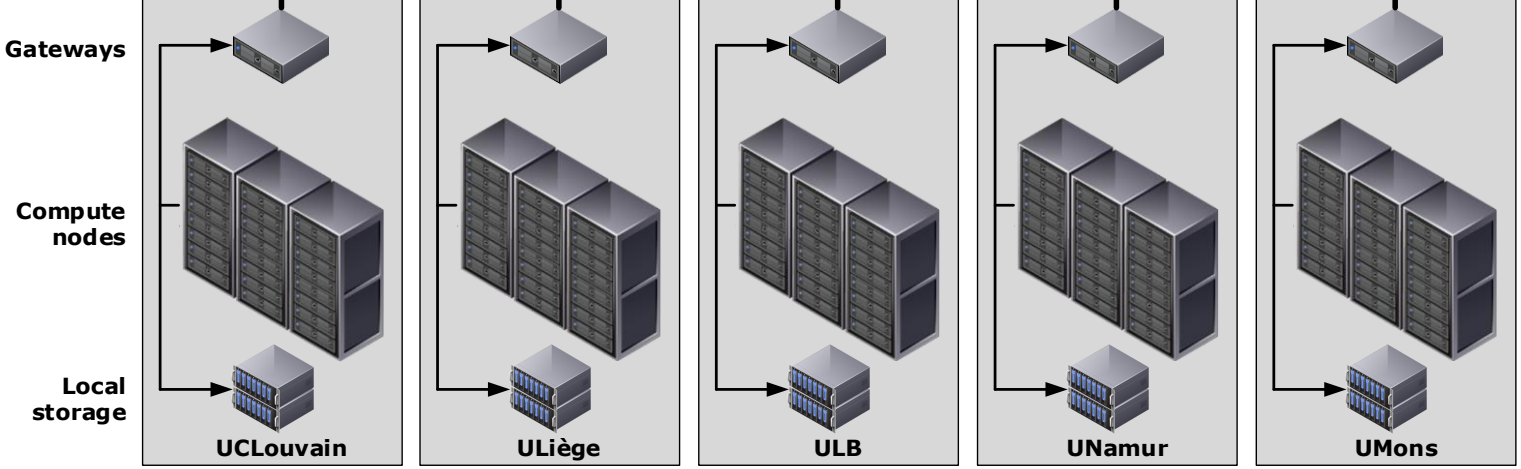
Fields of applications - CÉCI survey 2022

CÉCI distributed storage

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



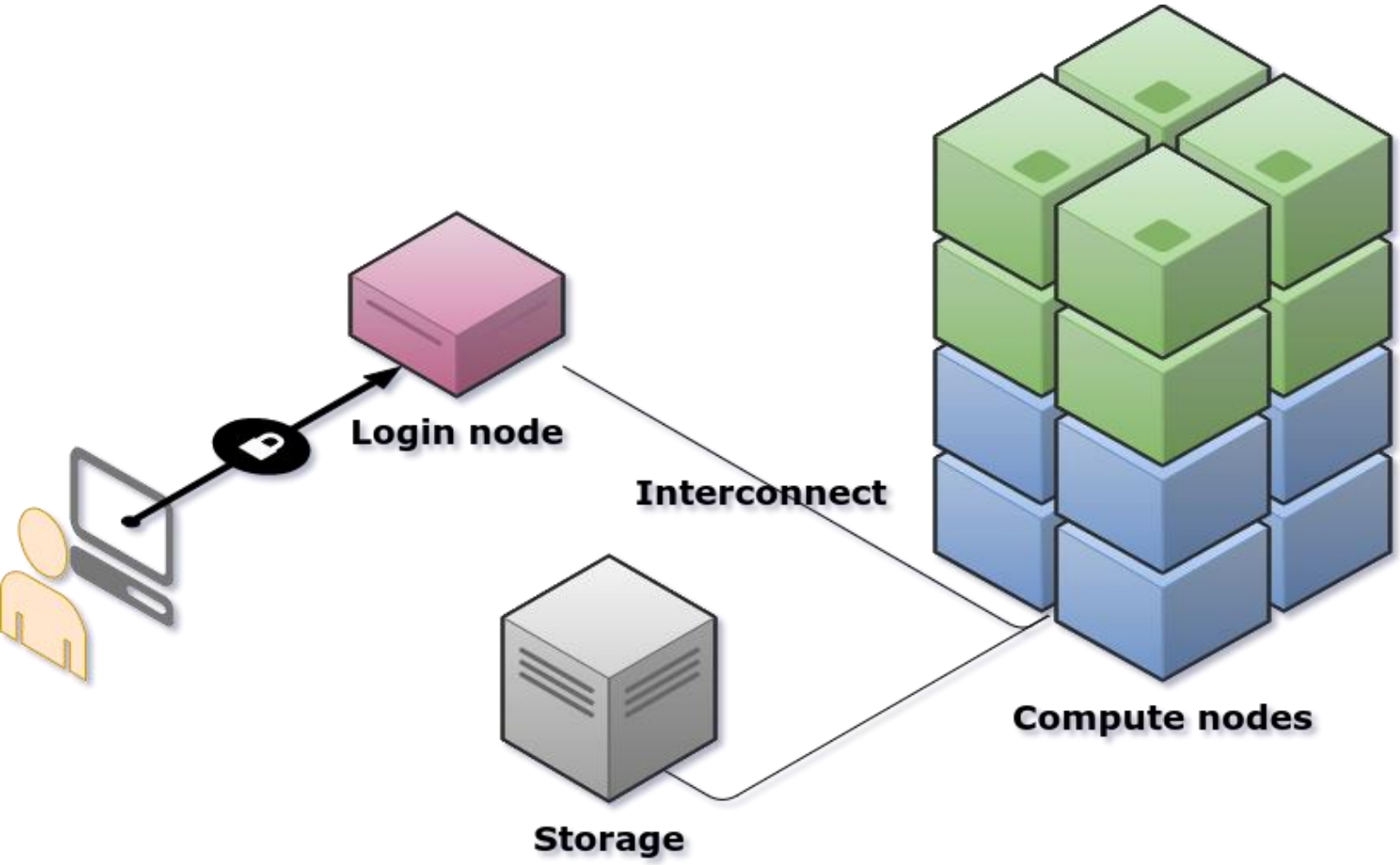
IBM Spectrum Storage



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

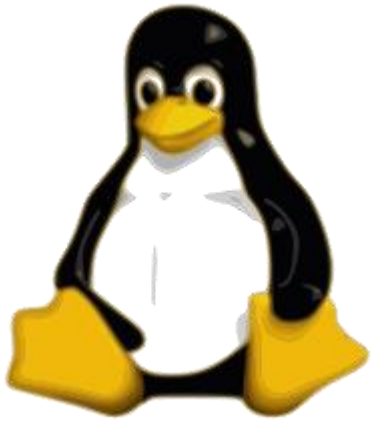
A cluster in a nutshell



Login node

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

Operating system



- All CÉCI cluster are running GNU/Linux

- Linux CentOS 7
  - Red Hat Entreprise 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***
NIC5	ULiège	Rome 2.9 GHz	4672 (73 x 64)	256 GB..1 TB	HDR Ib	BeeGFS 520 TB	None	2 days	MPI
Hercules2	UNamur	Naples 2 GHz SandyBridge 2.20 GHz	1024 (30 x 32 + 2 x 64) 512 (32 x 16)	64 GB..2 TB	10 GbE	NFS 20 TB	None	15 days	serial / SMP
Dragon2	UMons	SkyLake 2.60 GHz	592 (17 x 32 + 2 x 24)	192..384 GB	10 GbE	RAID0 3.3 TB	4x Volta V100	21 days	serial / SMP
Lemaitre3	UCL	SkyLake 2.3 GHz Haswell 2.6 GHz	1872 (78 x 24) 112 (4 x 28)	95 GB 64 GB	Omnipath	BeeGFS 440 TB	None	2 days 6 hours	MPI
Dragon1	UMons	SandyBridge 2.60 GHz	416 (26 x 16) 32 (2x16)	128 GB	GbE	RAID0 1.1 TB	4x Tesla C2075, 4x Tesla Kepler K20m	41 days	serial / SMP
NIC4*	ULiège	SandyBridge 2.0 GHz IvyBridge 2.0 GHz	2048 (120 x 16 + 8 x 16)	64 GB	QDR Ib	FHGFS 144 TB	None	3 days	MPI
Vega*	ULB	Bulldozer 2.1 GHz	896 (14 x 64)	256 GB	QDR Ib	GPFS 70 TB	None	14 days	serial / SMP / MPI
Hercules*	UNamur	SandyBridge 2.20 GHz	512 (32 x 16)	64..128 GB	GbE	NFS 20 TB	None	63 days	serial / SMP
Lemaitre2*	UCL	Westmere 2.53 GHz	1380 (115 x 12)	48 GB	QDR Ib	Lustre 120 TB	3x Quadro Q4000	3 days	MPI
Hmem*	UCL	MagnyCours 2.2 GHz	816 (17 x 48)	128..512 GB	QDR Ib	FHGFS 30 TB	None	15 days	SMP

Hercules 2 @ UNamur

UNIVERSITÉ DE NAMUR

- 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

- Other HPC resources
  - Hyades 2
    - 288 cores total
    - Up to 92 GB RAM per node

Dragon 2 @ UMons

UMONS Université de Mons

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

HPC @ UMons

UMONS
Université de Mons

- Local support
  - 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

UCLouvain
Université

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

HPC @ UCLouvain

UCLouvain
Université

CISM

- Local support
  - Institut de Calcul Intensif et de Stockage de Masse (egs-cism@listes.uclouvain.be)

Thomas Keutgen (Head)

Damien François

Olivier Mattelaer

Bernard Van Renterghem

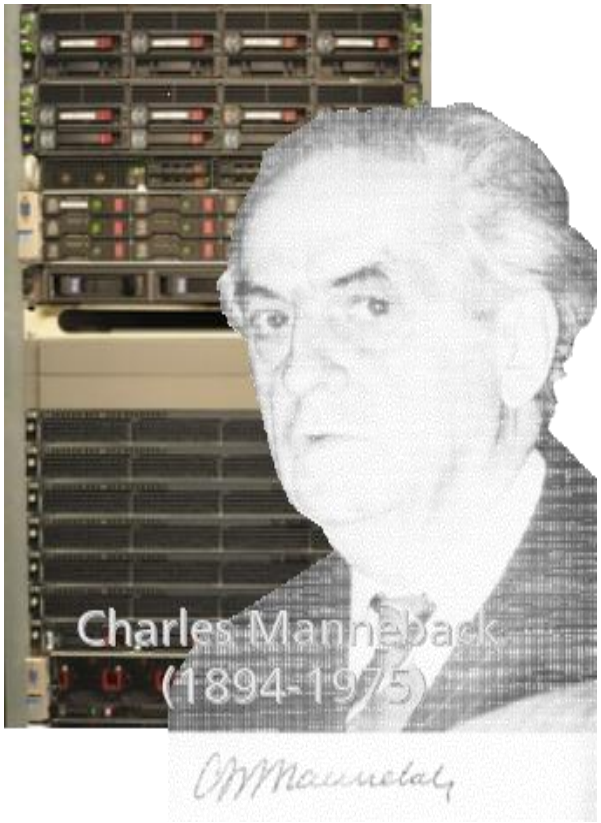
Patrick Vranckx

C.E.C.I

HPC @ UCLouvain

UCLouvain
Université

- 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
- More info
  - http://www.ulg.ac.be/nic4

Zenobe

Cenaero

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

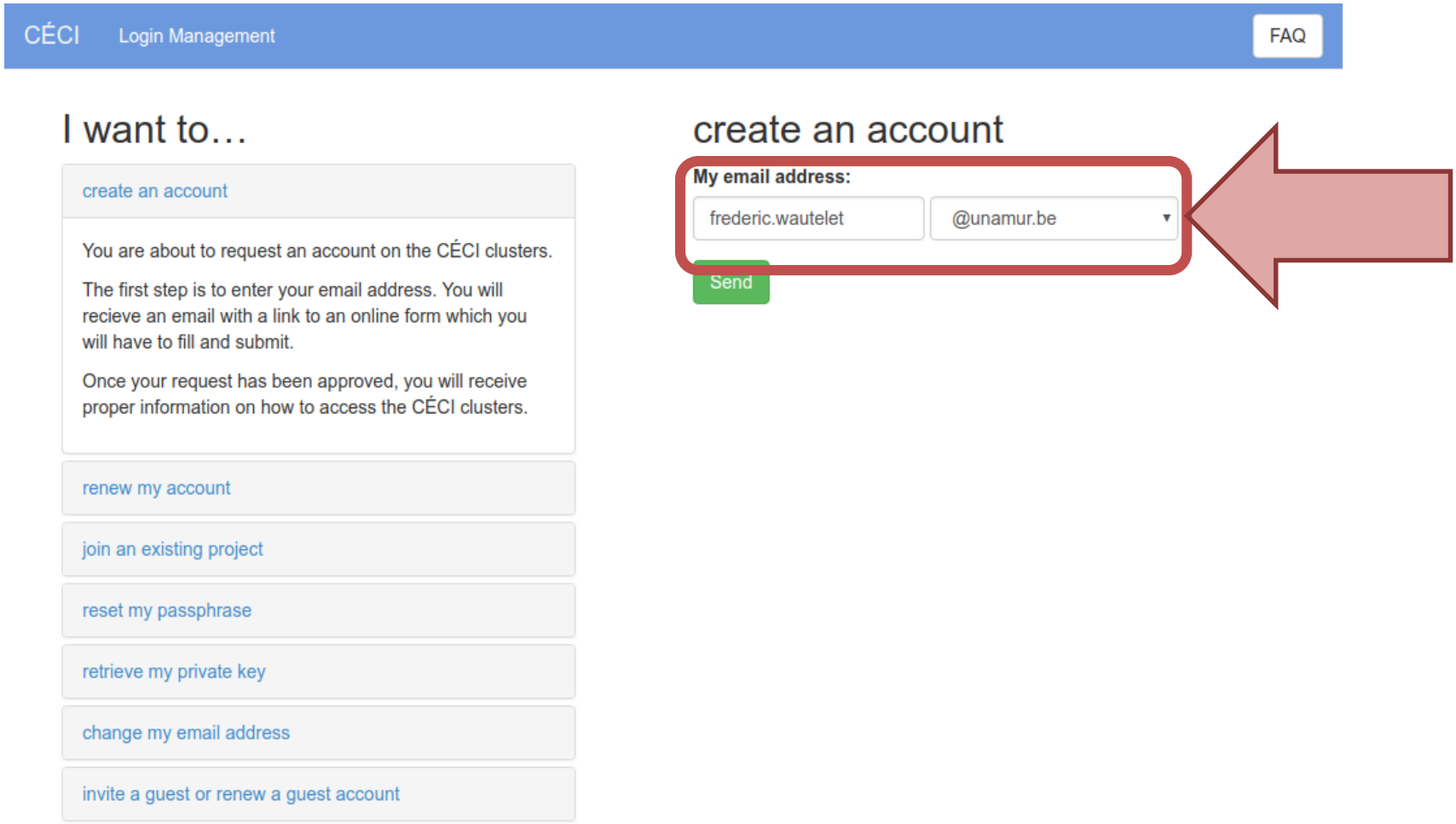
C.E.C.I

How to get a CÉCI account?

Go to **http://www.ceci-hpc.be**

Create/Manage Account

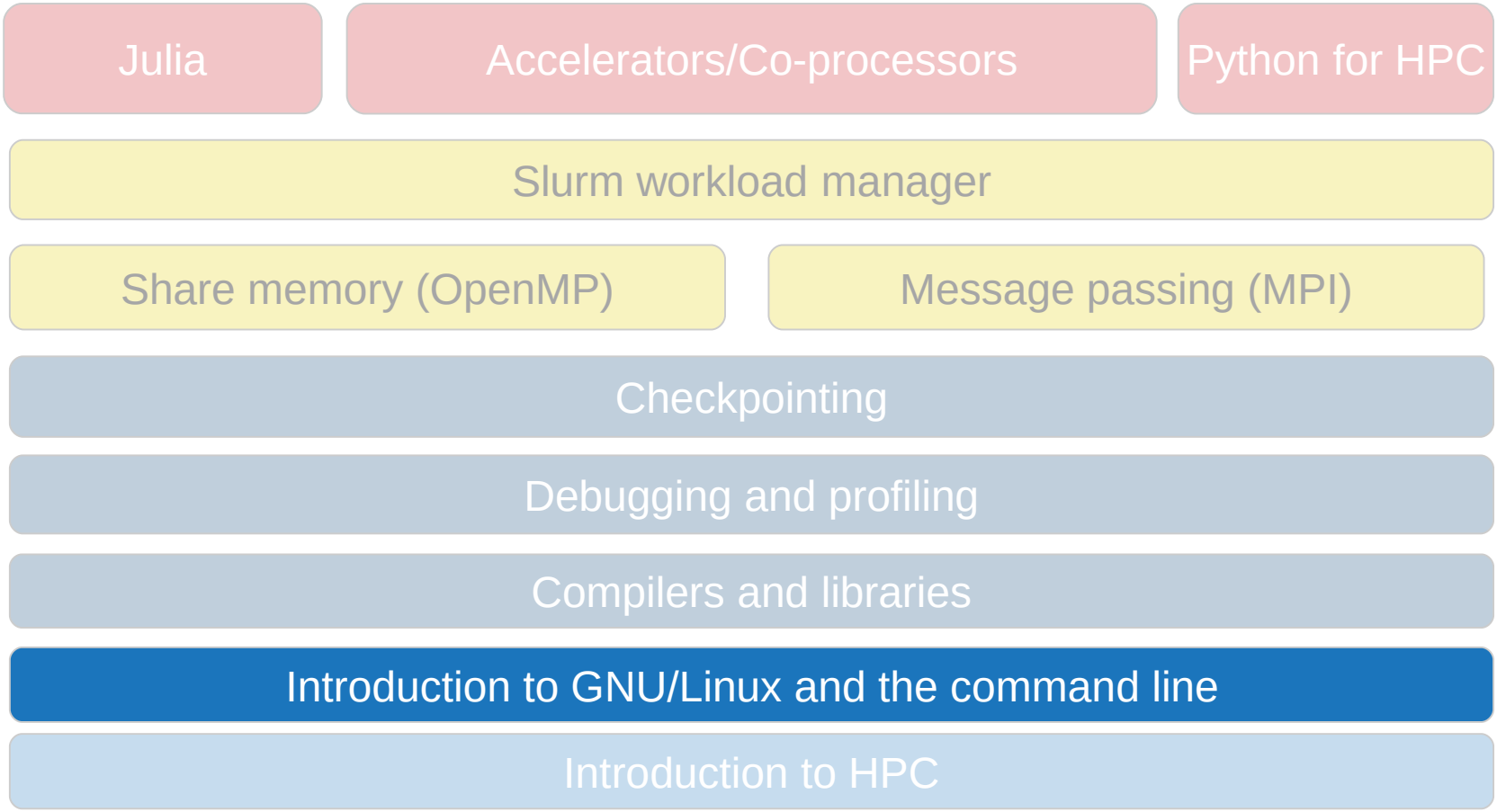
I want to… create an account



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



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

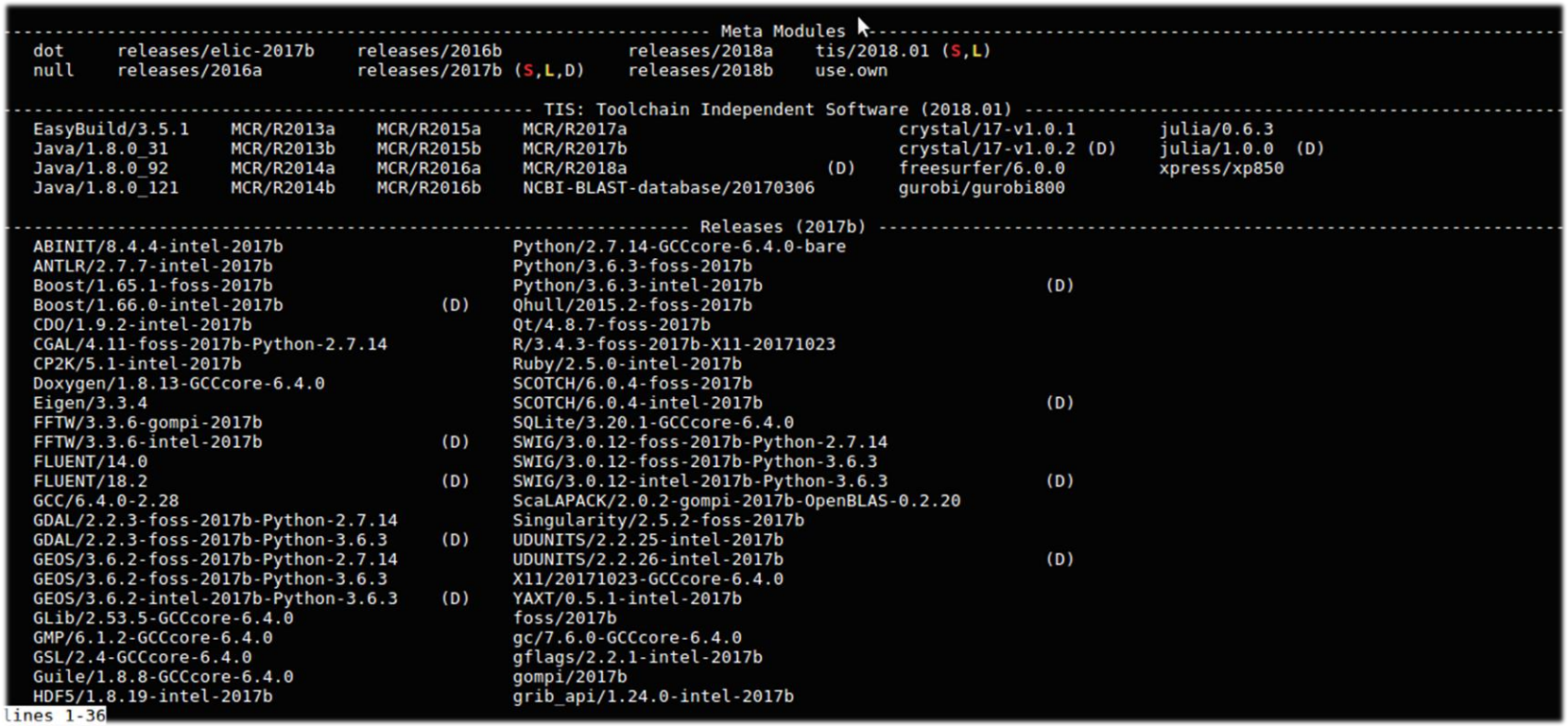
Bash

- 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



How to use HPC infrastructure

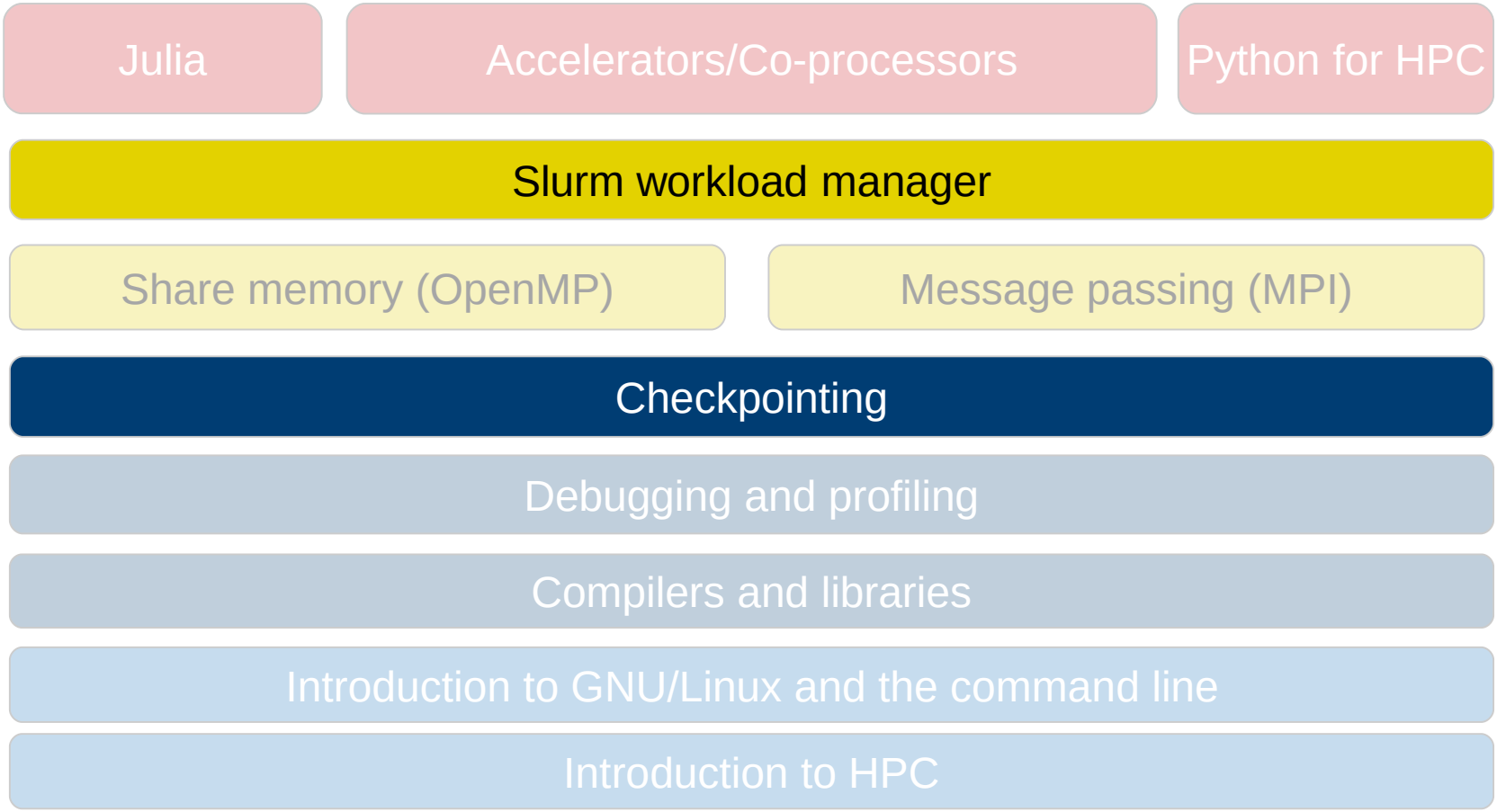
Part I

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

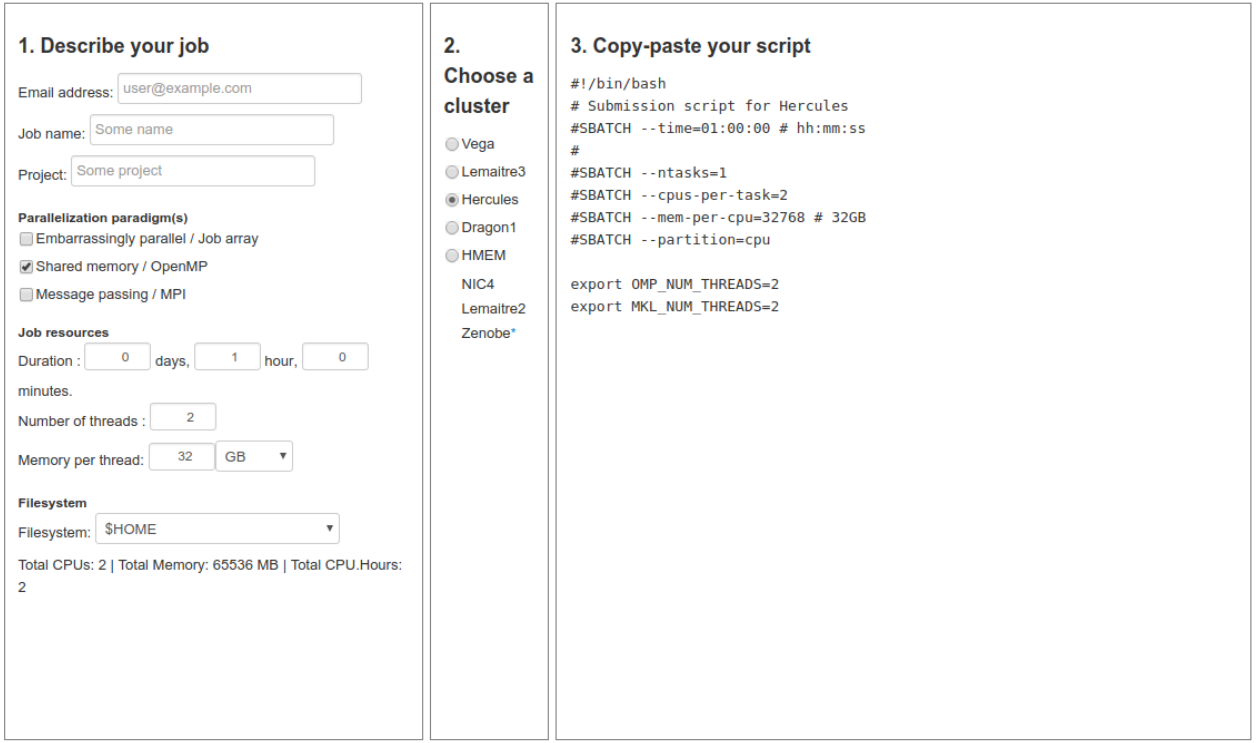
- squeue

```
$  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.ceci-hpc.be/scriptgen.html



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 NODELIS
         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
```

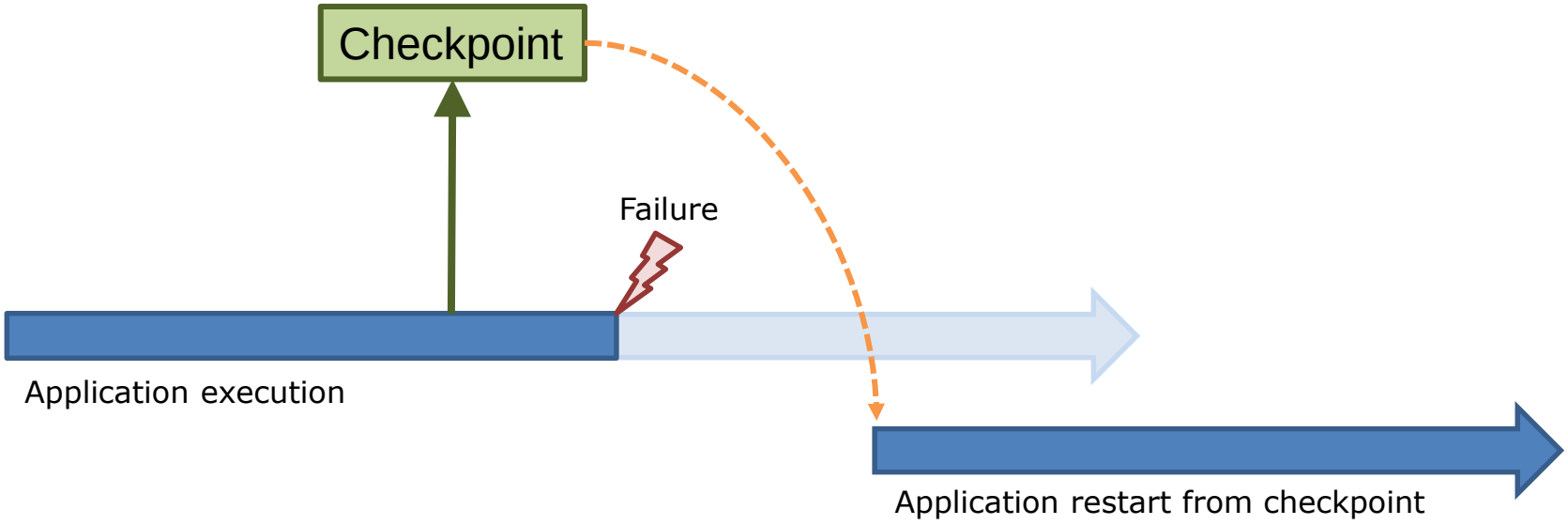
Job Dependencies

- 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

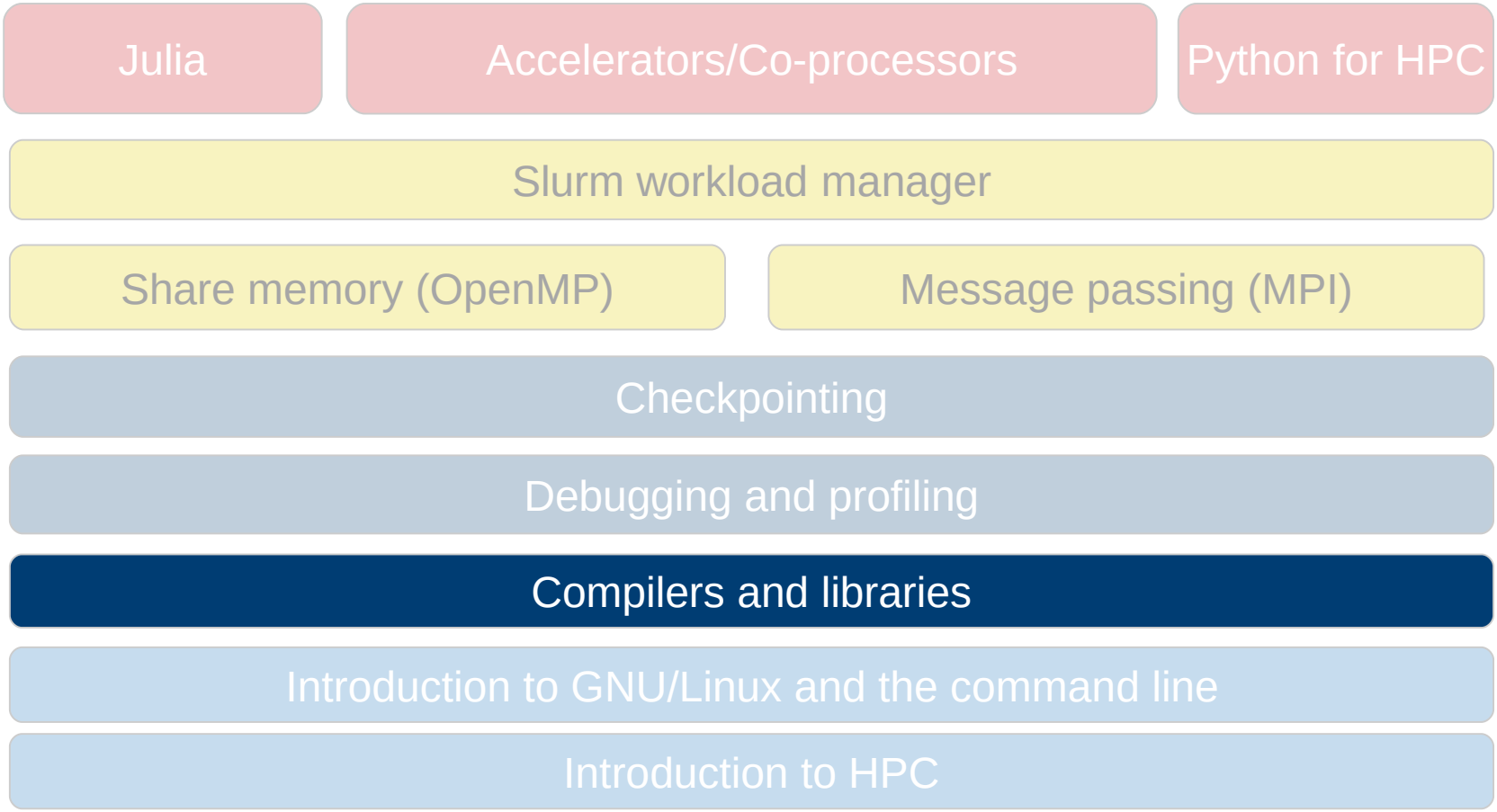
Part I

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



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

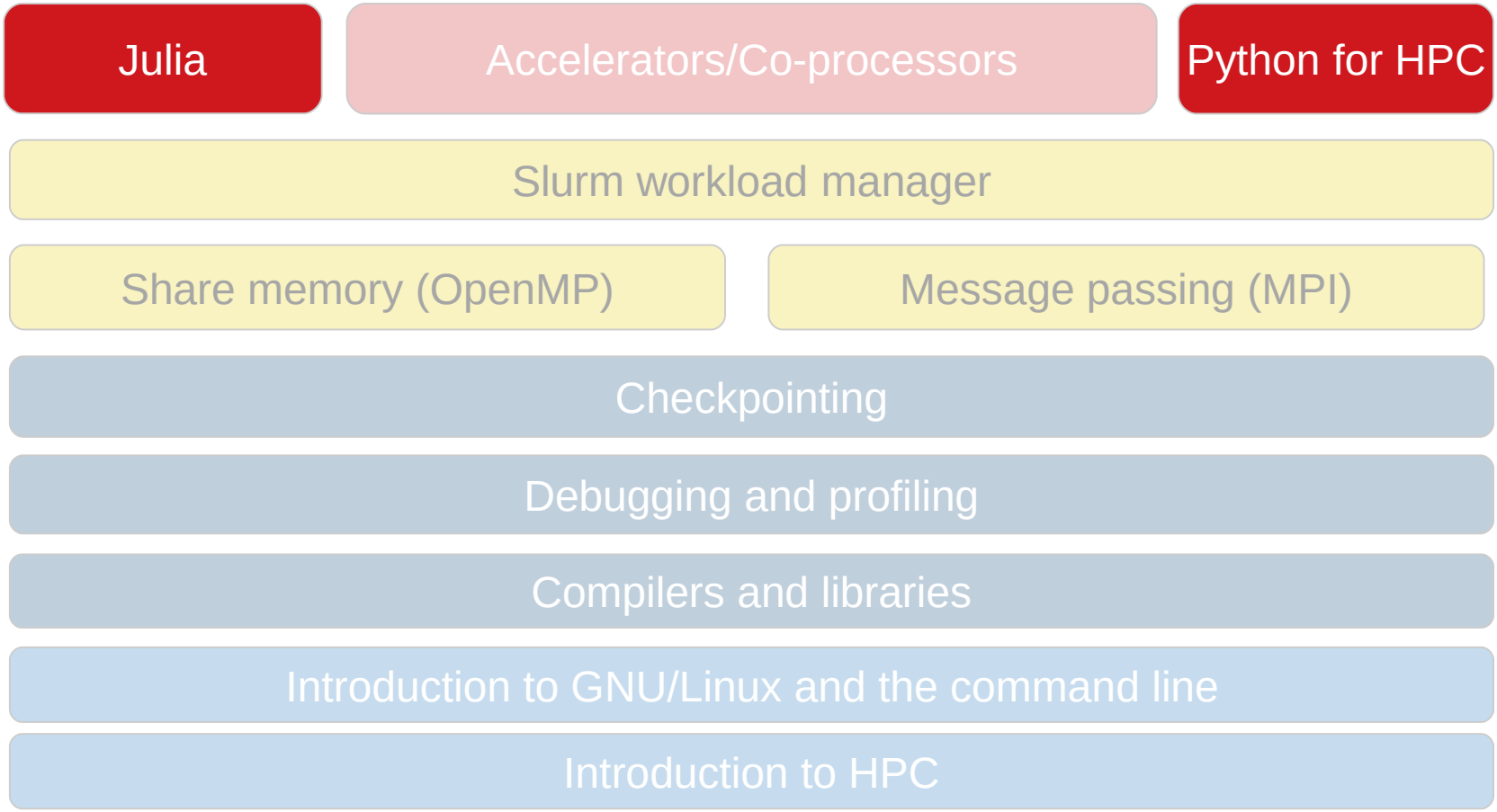
Part I: Introduction

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



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)

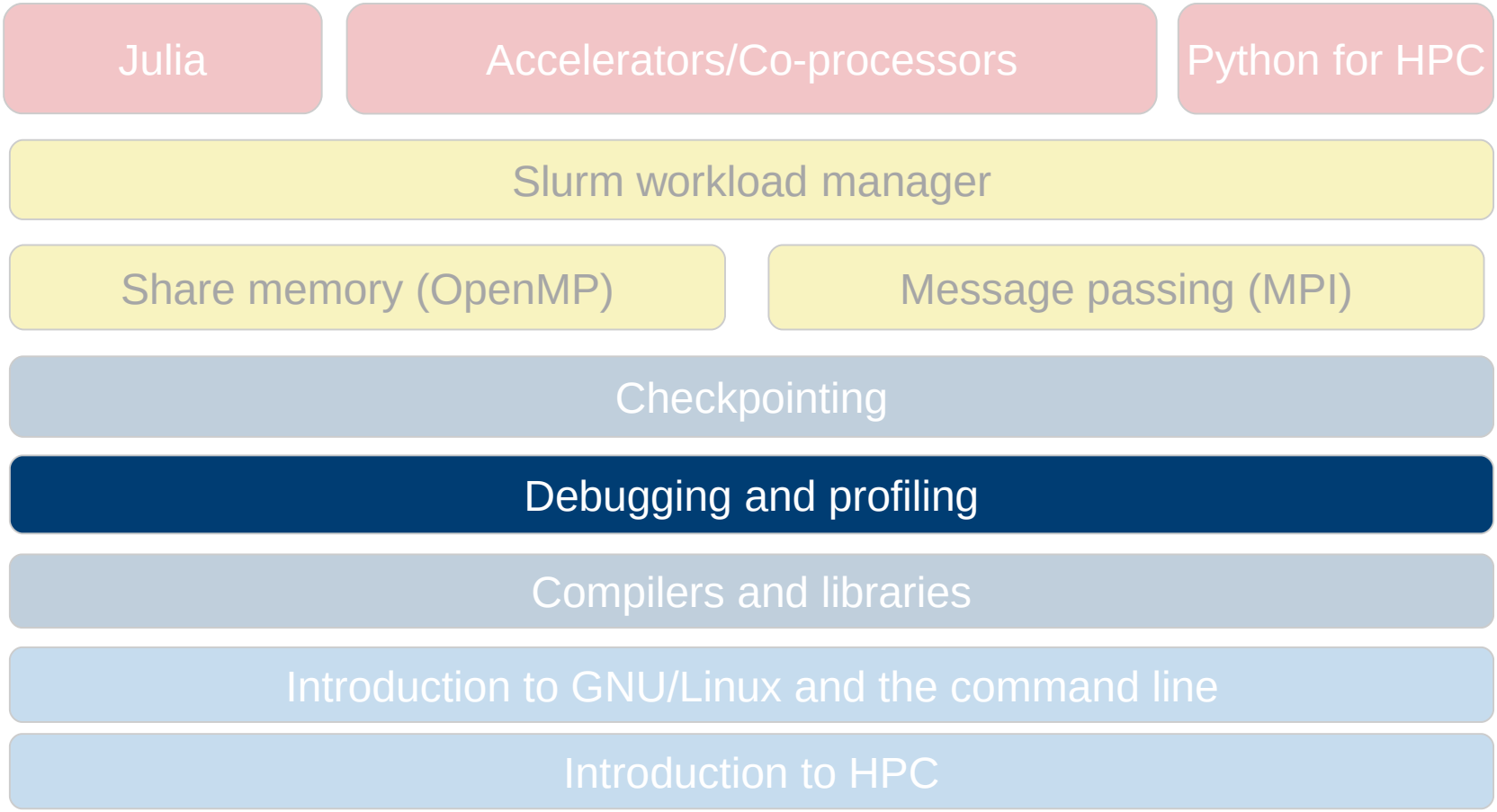
Part III: Python

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



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

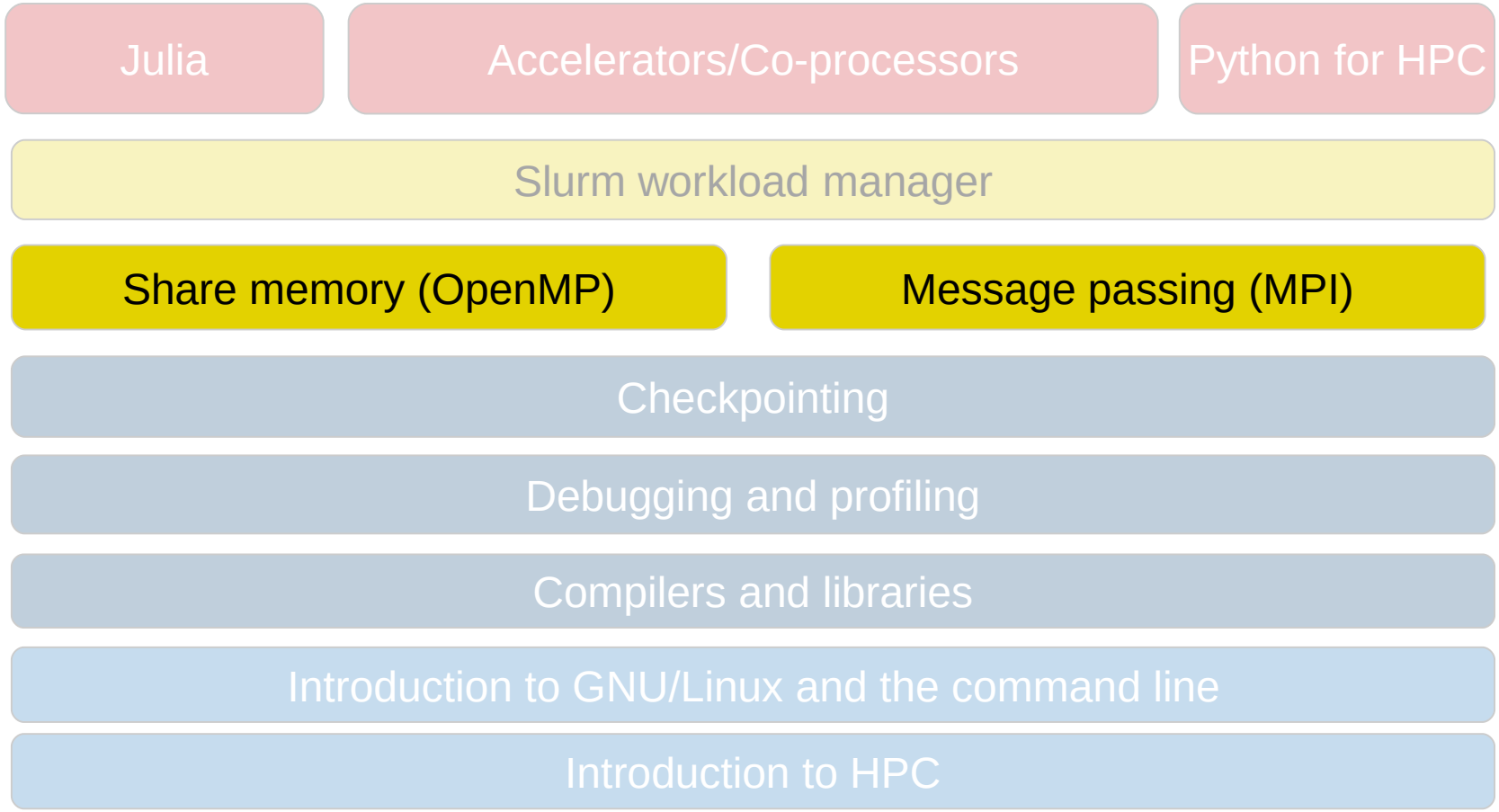
9. Efficient use of Python on the cluster

Part IV: Tools

10. Introduction to code versioning

11. Debugging and profiling scientific code, and commercial optimized libraries

Parallel computing



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

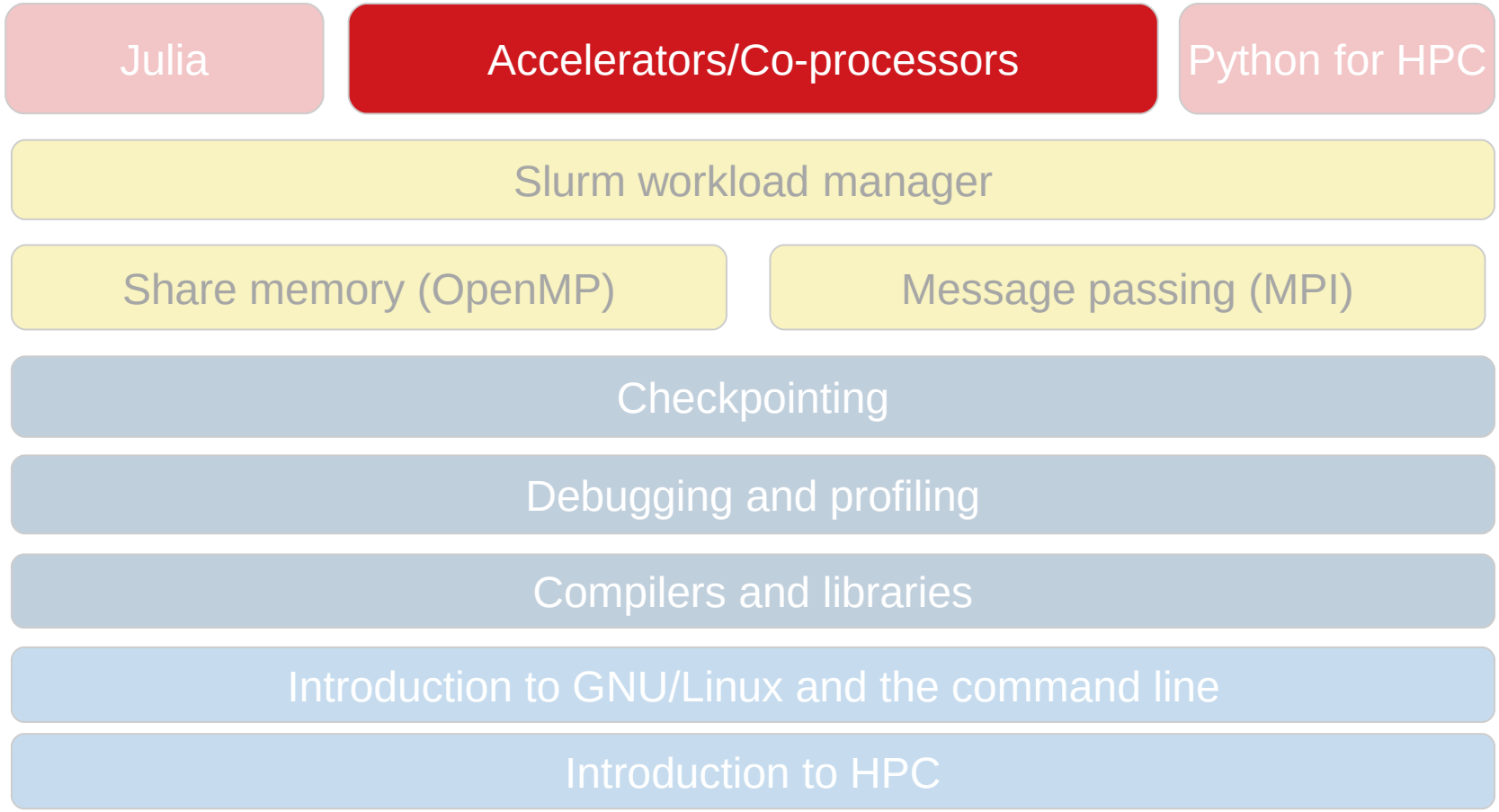
Part II

4. Parallel programming with OpenMP

5. Directive Based Parallel programming on GPU (OpenACC)

6. Parallel programming on GPU with CUDA

Accelerators



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	7 TFLOPS
Hercules2	4 x NVIDIA RTX A6000	10752	48 GB	40 TFLOPS	1 TFLOPS
	8 x NVIDIA Tesla A40	10752	48 GB	38 TFLOPS	600 GFLOPS

Going Parallel

Part I

1. Introduction to parallel computing

2. Parallel programming with MPI

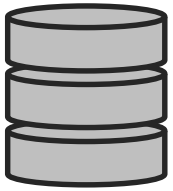
Part II

4. Parallel programming with OpenMP

5. Directive Based Parallel programming on GPU (OpenACC)

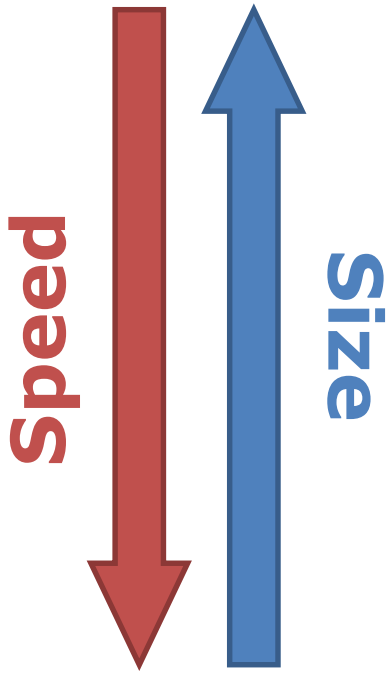
6. Parallel programming on GPU with CUDA

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

	Bias[a]		Variance[b]		95% CI[c]	
Method	$(\hat{\beta}_W)$	$(\hat{\beta}_X)$	$(\hat{\beta}_W)$	$(\hat{\beta}_X)$	$(\hat{\beta}_W)$	$(\hat{\beta}_X)$
(M.1) $P(R=1)=0.66$						
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
(M.2) 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
(M.3) 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
(M.4) 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

[a]Bias $= (\hat{\beta}-\beta_0)/\beta_0$.

[b]Simulation variance.

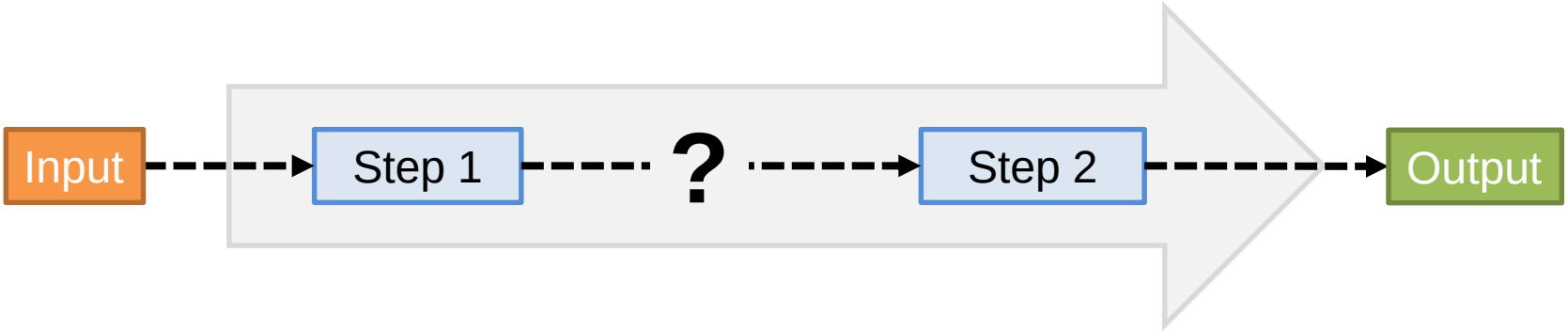
[c]Confidence interval using jackknife standard error.

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

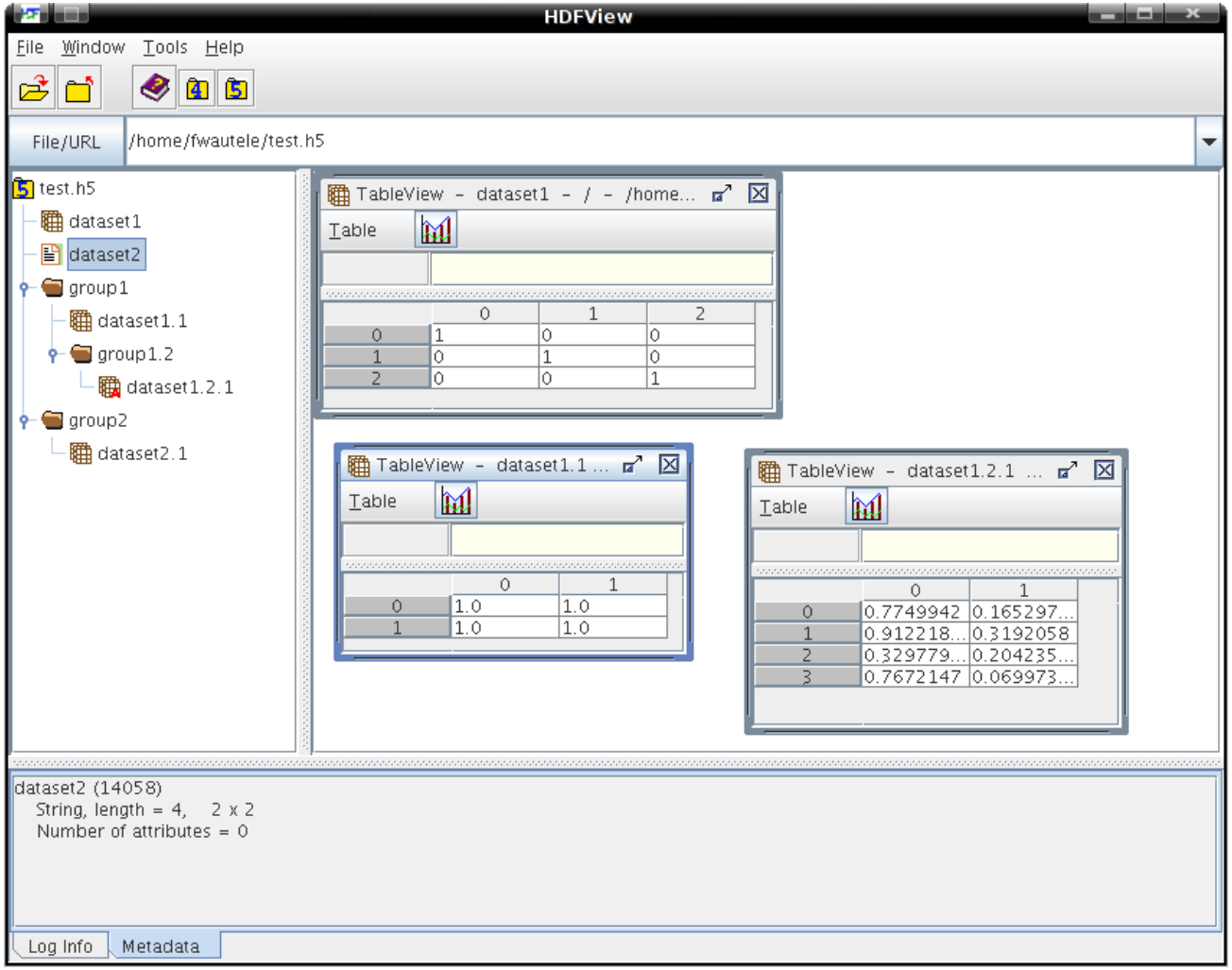
Solution 2: NetCDF

- 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



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

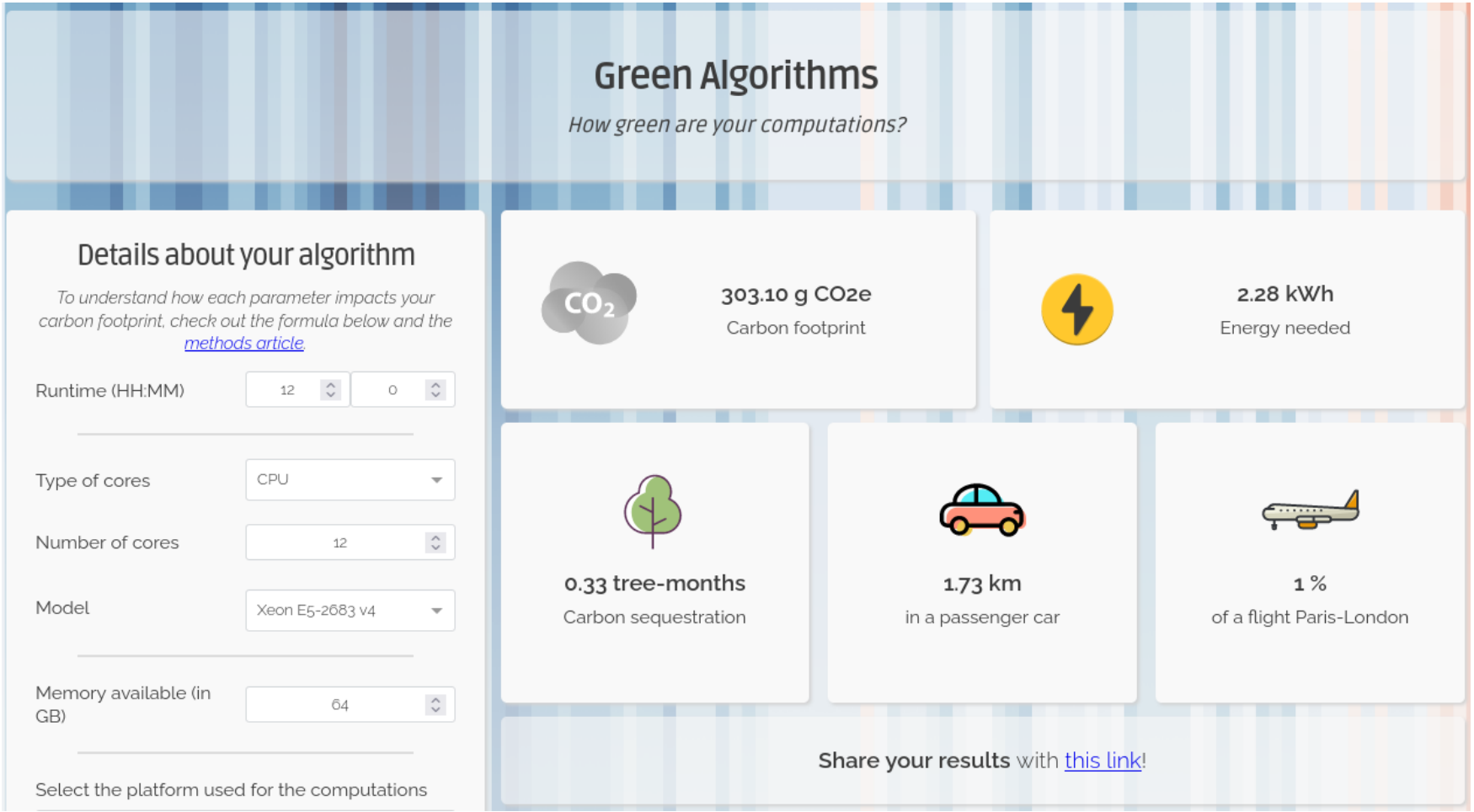
Frontier, Oak Ridge National Laboratory, USA

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



Thanks you for your attention

and happy computing