Atomistic simulations and HPC at IDRIS

Thibaut Very, engineer at support team

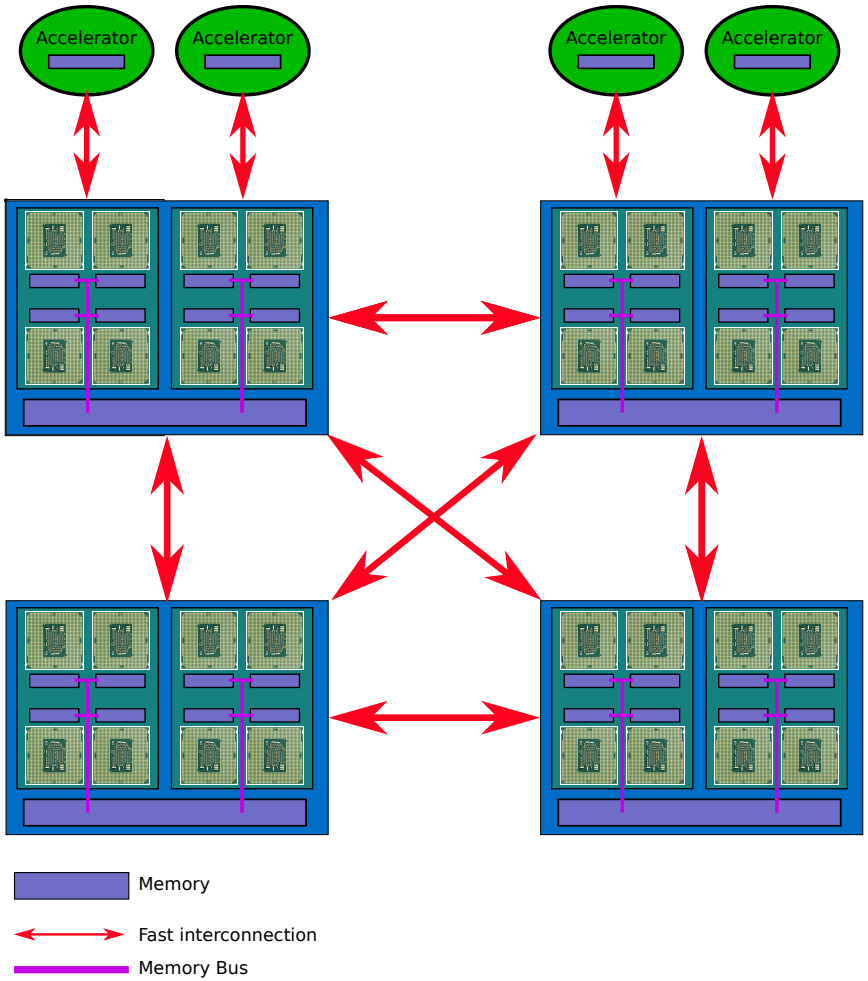
Goals

Goals of this presentation

- Introduction to
  - European High Performance Computing ecosystem
  - French HPC ecosystem
  - GENCI
    - ✓ How to compute at HPC centers
  - Focus on IDRIS
    - ✓ Organization
    - ✓ Machines
    - ✓ User support
- Technical points
  - A few words on hardware
  - Why is parallelism important
- Application to atomistic simulation
  - Performing tests to optimize performance

Introduction - Some vocabulary: Architecture

Here is a cluster of 4 computers with a fast interconnection system. Let's see how it works bottom-up.



Summary

Architecture

- The *core* is the basic unit for computation
- A *processor* might group several cores
- A *node* is what we usually call a computer. It might have several processors
- A *cluster* is a group of nodes interconnected
- An *accelerator* is a device connected to a node which performs offloaded computation

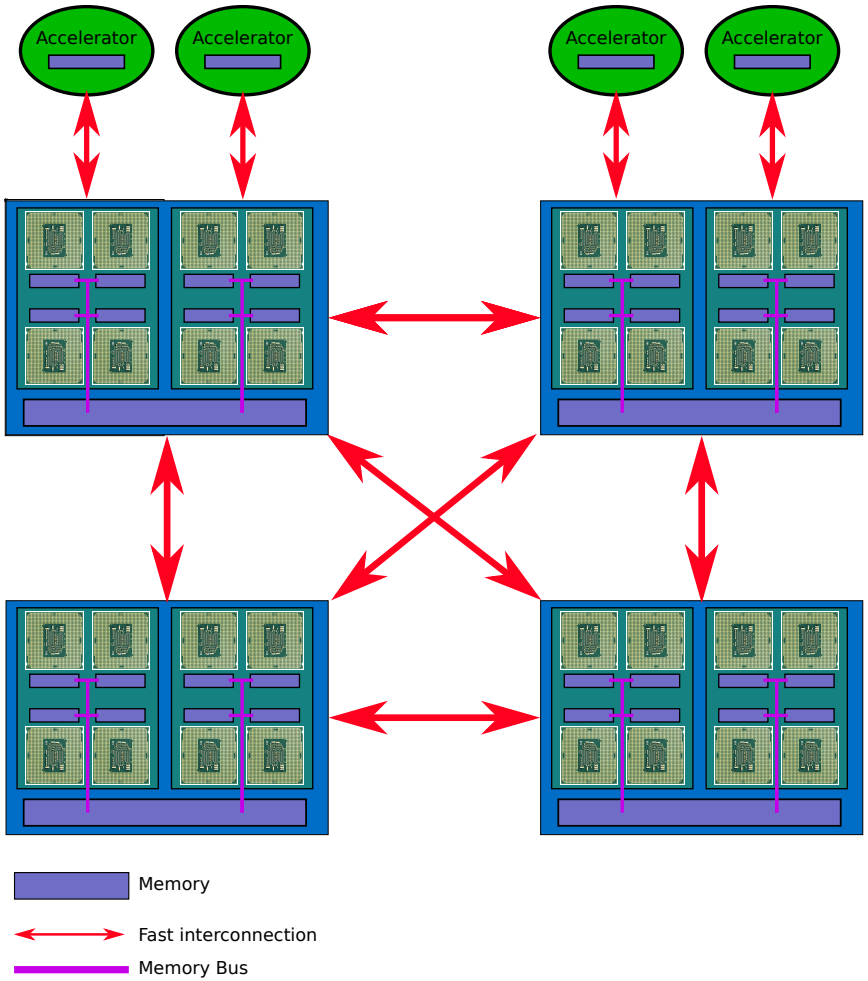
Memory

One can distinguish between 2 categories:

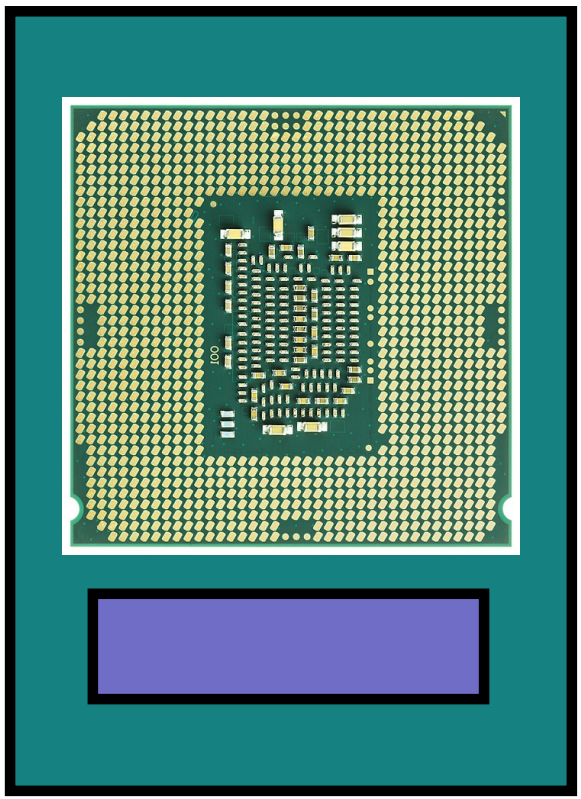
- *Shared Memory*: The units in a group share some memory and all have direct access
- *Distributed Memory*: The units must go through a network to access memory of another unit

Introduction - Some vocabulary: Architecture

Here is a cluster of 4 computers with a fast interconnection system. Let's see how it works bottom-up.

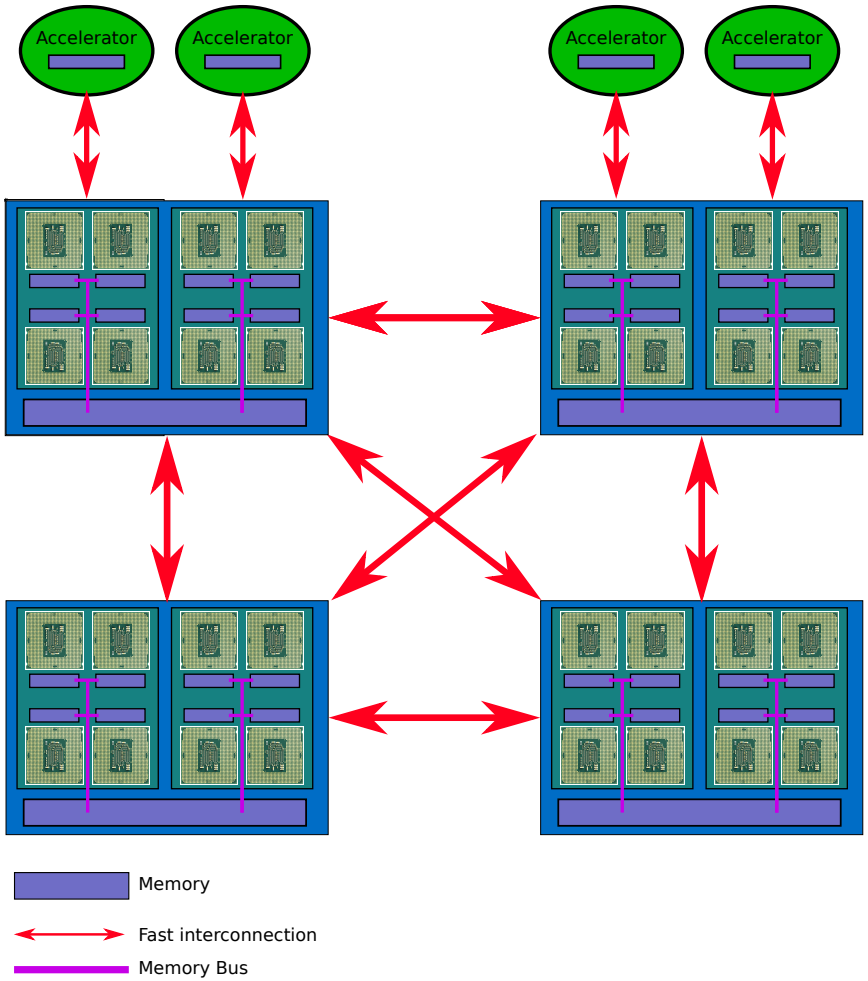


The basic unit is *the core*. It comes with some fast (but small) cache memory.

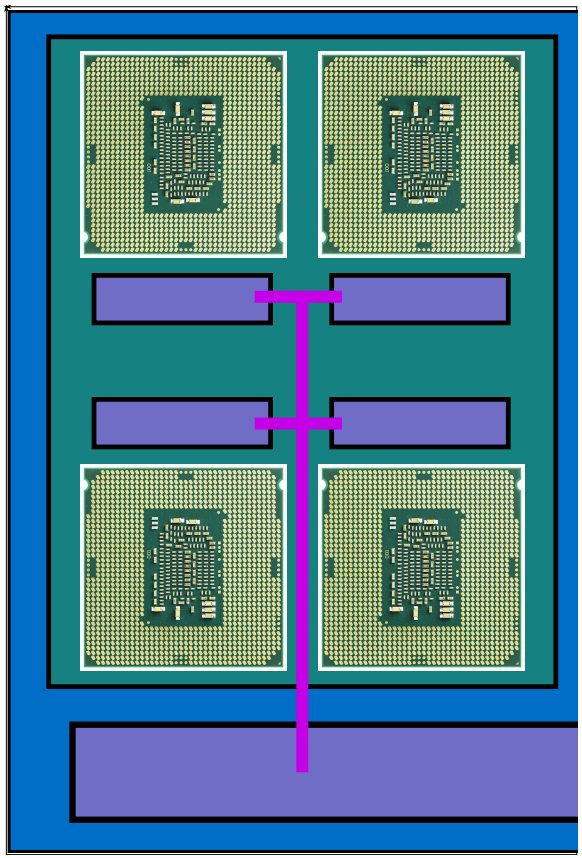


Introduction - Some vocabulary: Architecture

Here is a cluster of 4 computers with a fast interconnection system. Let's see how it works bottom-up.

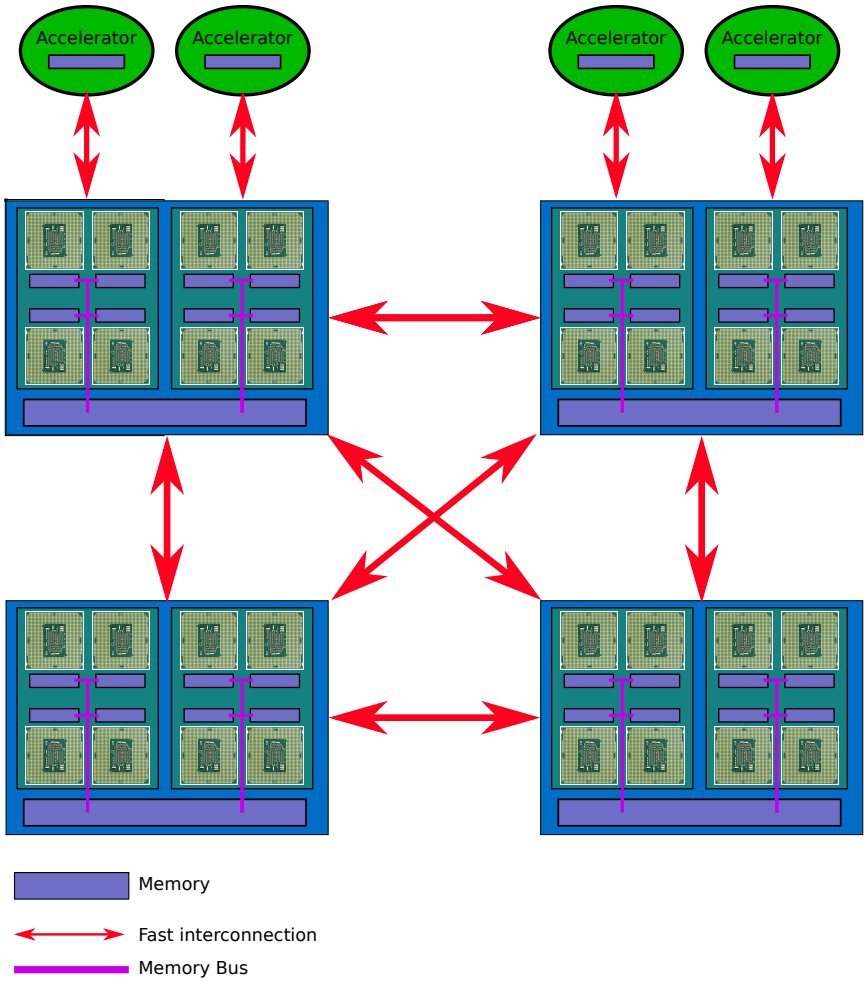


A *processor* is a piece of hardware on which there can be several cores. The cores in a processor have some shared memory.

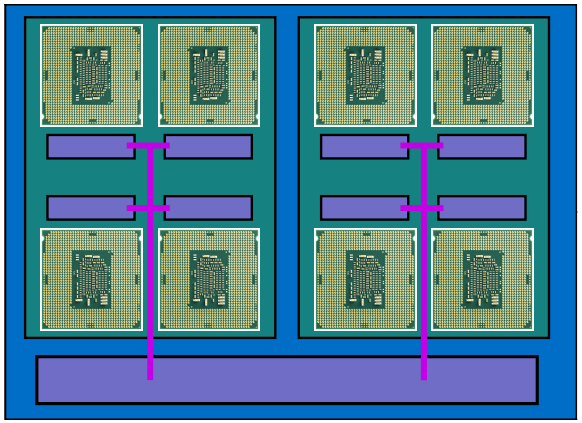


Introduction - Some vocabulary: Architecture

Here is a cluster of 4 computers with a fast interconnection system. Let's see how it works bottom-up.



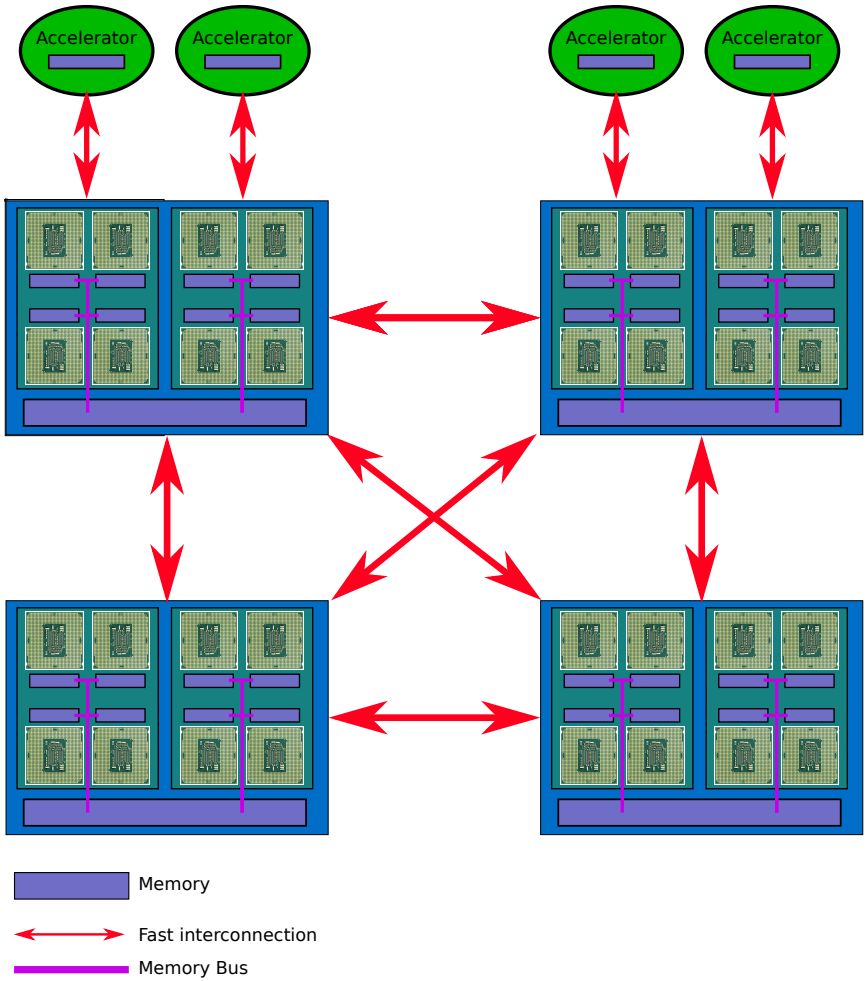
A *node* usually have several processors sharing some memory.



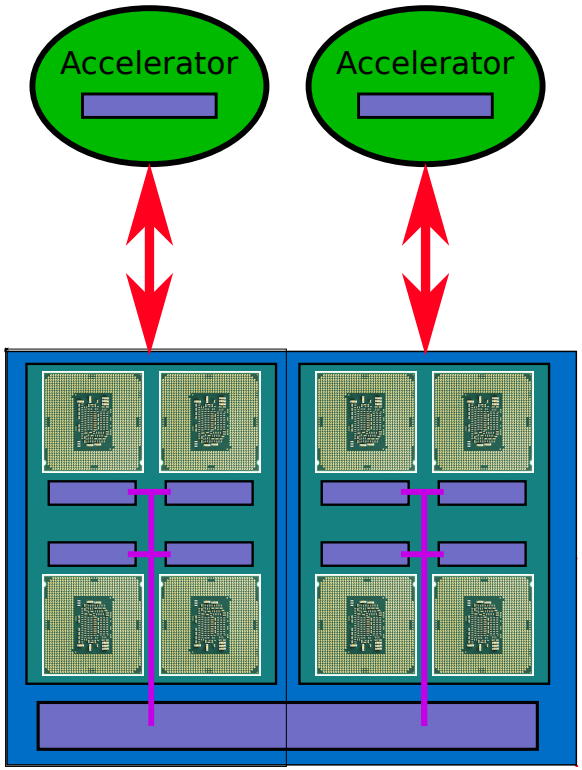
A *node* like this one is representative of a *shared memory* system.

Introduction - Some vocabulary: Architecture

Here is a cluster of 4 computers with a fast interconnection system. Let's see how it works bottom-up.



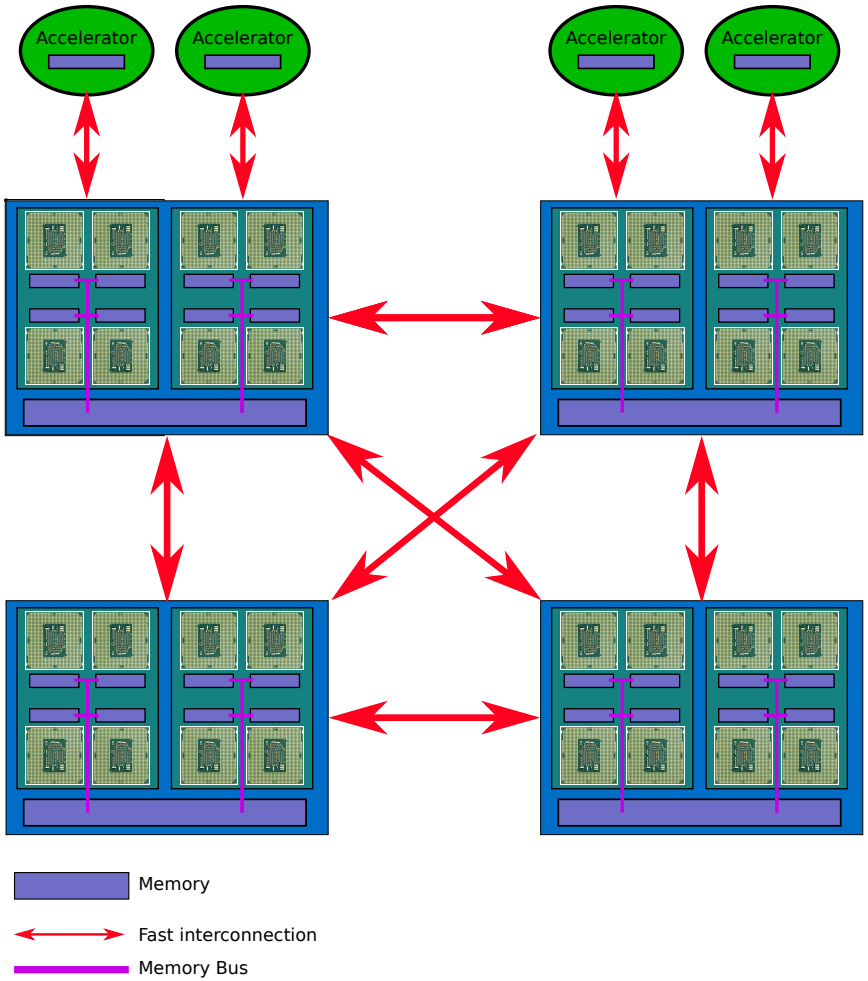
You can add a high performance device on which you can offload some computation. It is usually called an *accelerator*. For example, Graphical Processing Units (GPUs) are now often used to accelerate computation on High Performance Computing (HPC) clusters.



An *accelerator* has its own memory. One needs to manage data transfers.

Introduction - Some vocabulary: Architecture

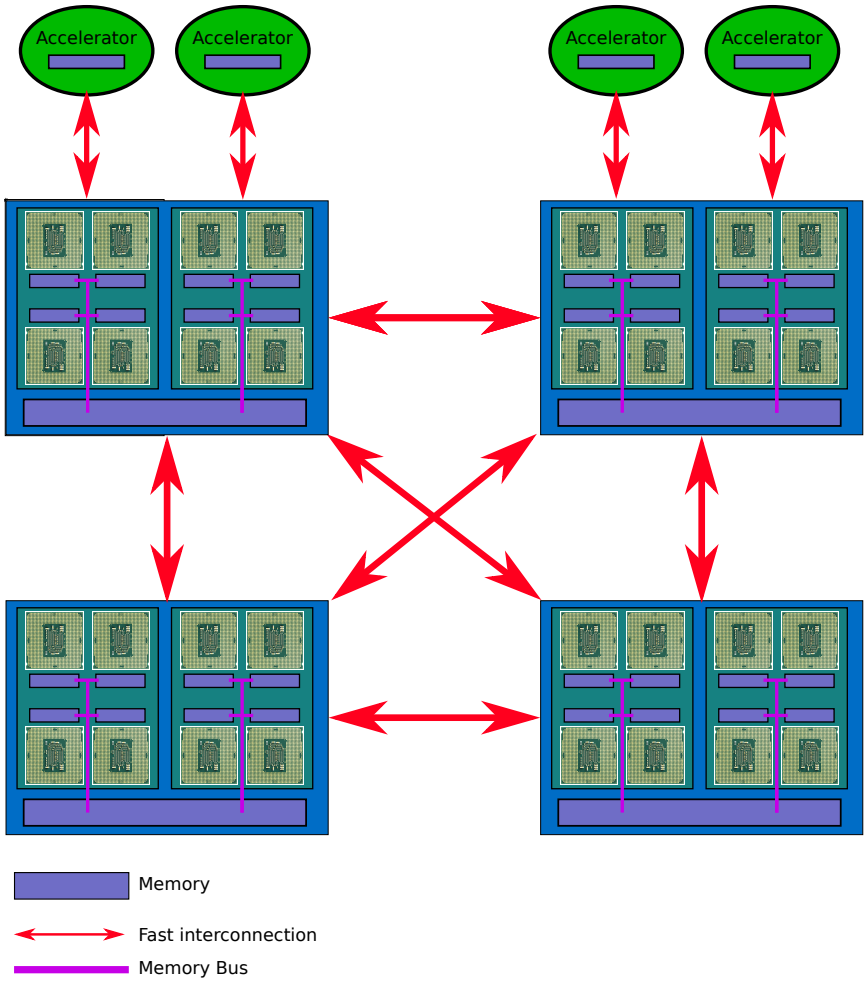
Here is a cluster of 4 computers with a fast interconnection system. Let's see how it works bottom-up.



A *cluster* is a group of nodes which are interconnected. One node cannot access directly the memory of another node. It needs to pass through a network.
A *cluster* is representative of a *distributed memory* system.

Introduction - Some vocabulary: Architecture

Here is a cluster of 4 computers with a fast interconnection system. Let's see how it works bottom-up.



Summary

Architecture

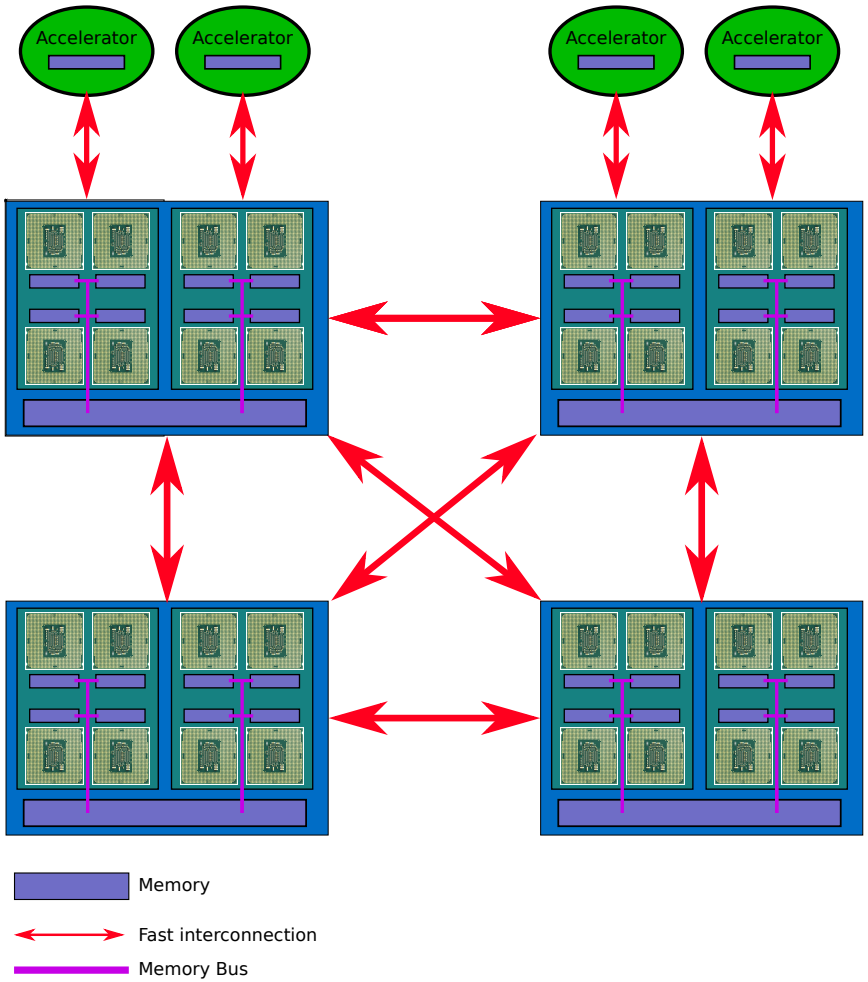
- The *core* is the basic unit for computation
- A *processor* might group several cores
- A *node* is what we usually call a computer. It might have several processors
- A *cluster* is a group of nodes interconnected
- An *accelerator* is a device connected to a node which performs offloaded computation

Memory

One can distinguish between 2 categories:

- *Shared Memory*: The units in a group share some memory and all have direct access
- *Distributed Memory*: The units must go through a network to access memory of another unit

Introduction - Parallel programming models



Parallel programming models

- For shared memory systems: *OpenMP*
- For distributed memory system: Message Passing like *MPI*
- For accelerators: *CUDA*, *OpenCL*, *OpenMP target*, *OpenAcc*

On modern architectures one have to combine the 3 ways to have the best performance.

A bit of vocabulary - Other

FLOPs or FLOP/s

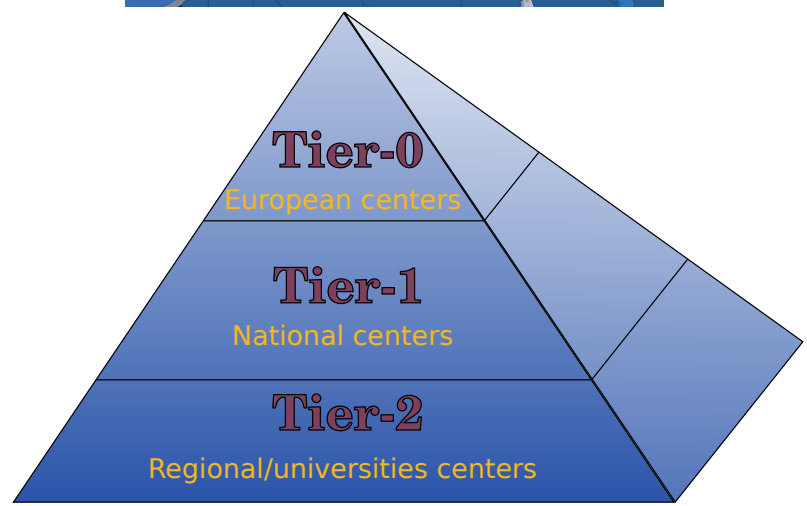
Number of floating point (real number) operations that a unit is able to sustain.
It is usually measured with the LINPACK benchmark which solves a dense $N \times N$ system of linear equations .
Have a look at Top500.

How is HPC organized in Europe/France?

PRACE - Description

Partnership for Advanced Computing in Europe

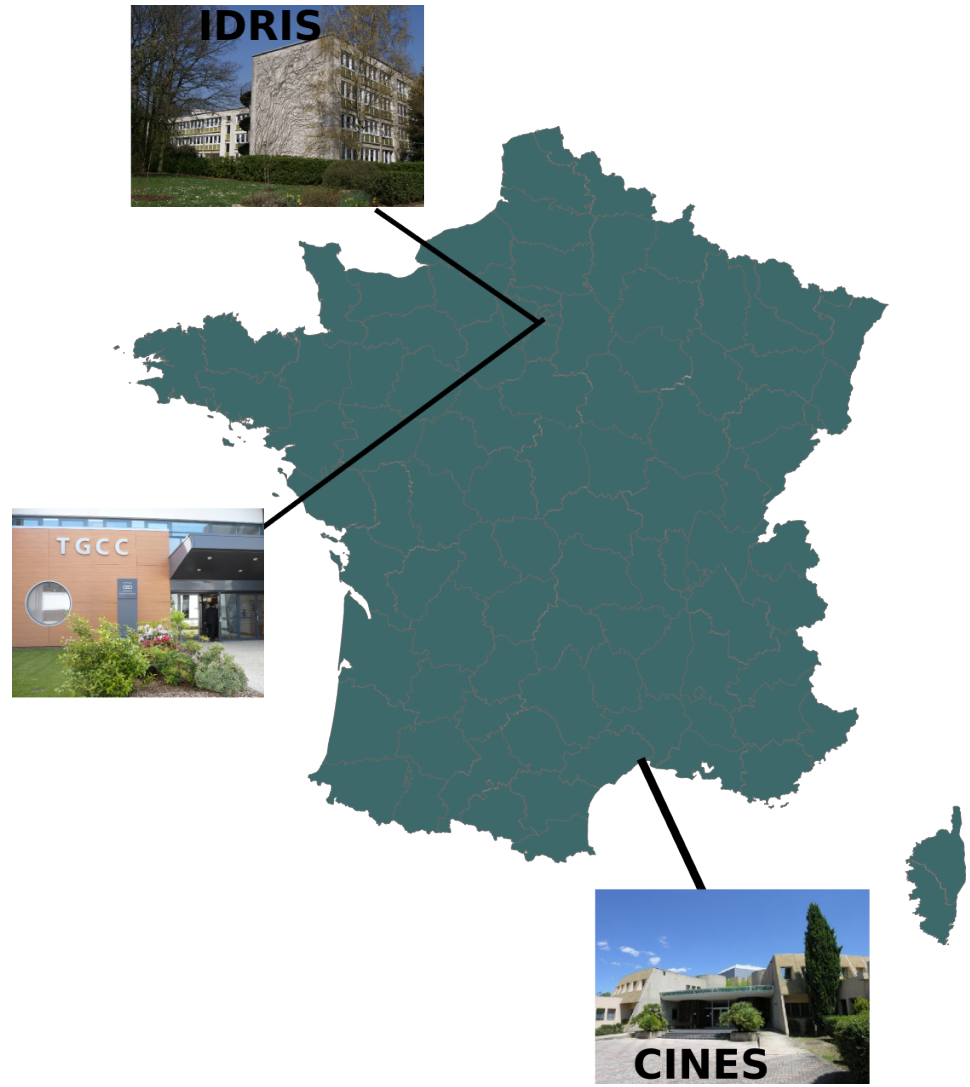
- Created in 2008
- 26 member countries + 2 observers
- Divides computing centers in tiers
- 8 tier-0 machines
  - Hazel Hen, Cray XC40, 7.4 PFLOP/s, HLRS/GCS
  - Joliot-Curie, BULL Sequana X1000 Skylake, 2 PFLOP/s, CEA
  - Joliot-Curie, BULL Sequana X1000 KnightLanding, 2.5 PFLOP/s, CEA
  - JUWELS, BULL Sequana X1000 Skylake, 12.0 PFLOP/s, FZJ
  - Marconi, Lenovo, 13 PFLOP/s, CINECA
  - MareNostrum, Lenovo System, 11.1 PFLOP/s, BSC
  - Piz Daint, Cray XC30, 7.8 PFLOP/s, CSCS
  - SuperMUC, Lenovo ThinkSystem, 6.5 PFLOP/s, LRZ
- Advanced Training courses (http://www.training.prace-ri.eu)
- In France: GENCI



HPC Organisation - GENCI

Grand Équipement National de Calcul Intensif

- Created in 2007
- Several share holders
  - Ministry of education and research (49%)
  - CNRS (20%)
  - CEA (20%)
  - Universities (10%)
  - INRIA (1%)
- 3 missions
  - National equipment
  - Promote HPC on European scale
  - Promote numerical simulation for academia and industry
    - ✓ Tier2 centers (Equip@Meso: Mesocenters)
    - ✓ Simseo
- 3 HPC centers
  - CINES
  - IDRIS
  - TGCC

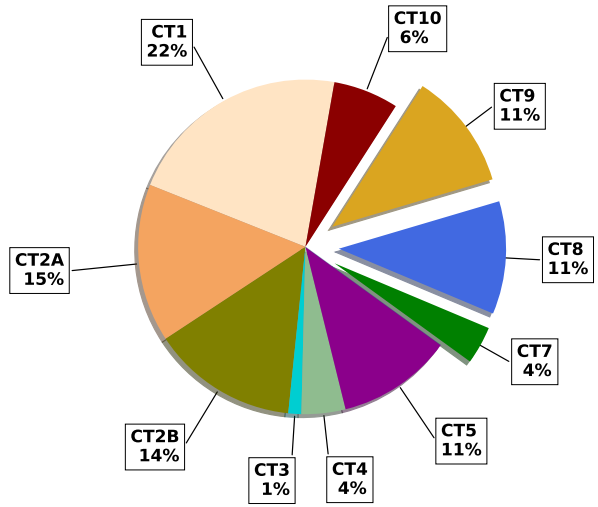


GENCI - Thematic committees

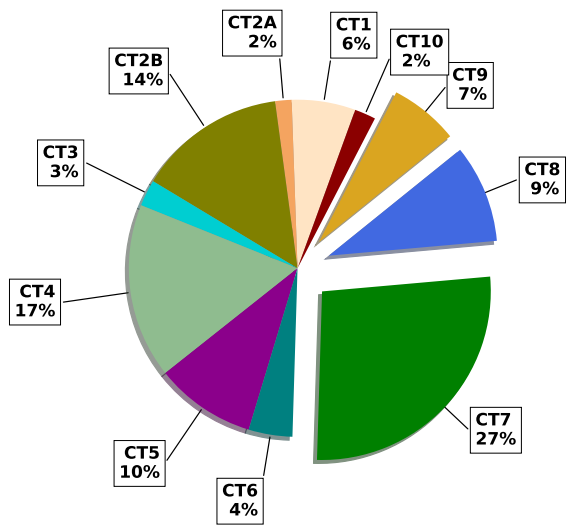
Research divided in 11 committees

- CT1: Environmental sciences
- CT2a: Non-reactive fluid flows
- CT2b: Reactive or multiphase fluid flows
- CT3: Biology and biomedical sciences
- CT4: Astrophysics and geophysics
- CT5: Theoretical and plasma physics
- CT6: Computer science, algorithm and mathematics
- CT7: Molecular dynamics in biology
- CT8: Quantum chemistry and molecular modeling
- CT9: Physics, chemistry and material properties
- CT10: Transversal applications (including Artificial Intelligence)

Hours allocated per CT for CPU partition (total = 239.038 Mh CPU)



Hours allocated per CT for GPU partition (total = 1.968 Mh GPU)

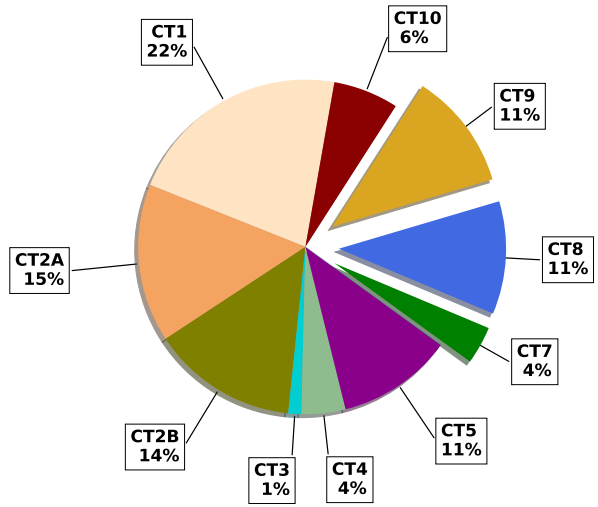


GENCI - Thematic committees

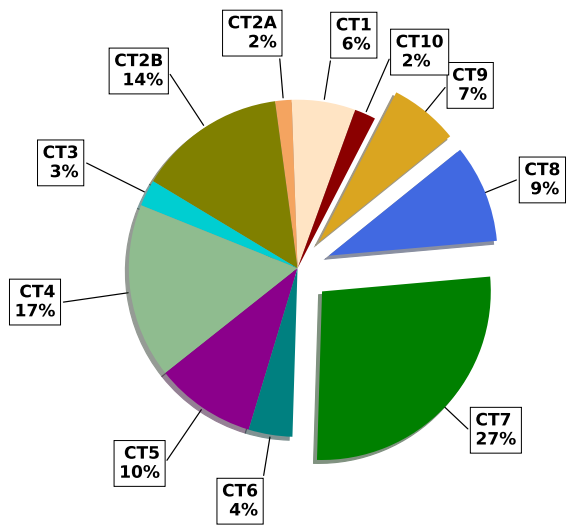
Research divided in 11 committees

- CT1: Environmental sciences
- CT2a: Non-reactive fluid flows
- CT2b: Reactive or multiphase fluid flows
- CT3: Biology and biomedical sciences
- CT4: Astrophysics and geophysics
- CT5: Theoretical and plasma physics
- CT6: Computer science, algorithm and mathematics
- CT7: Molecular dynamics in biology
- CT8: Quantum chemistry and molecular modeling
- CT9: Physics, chemistry and material properties
- CT10: Transversal applications (including Artificial Intelligence)

Hours allocated per CT for CPU partition (total = 239.038 Mh CPU)



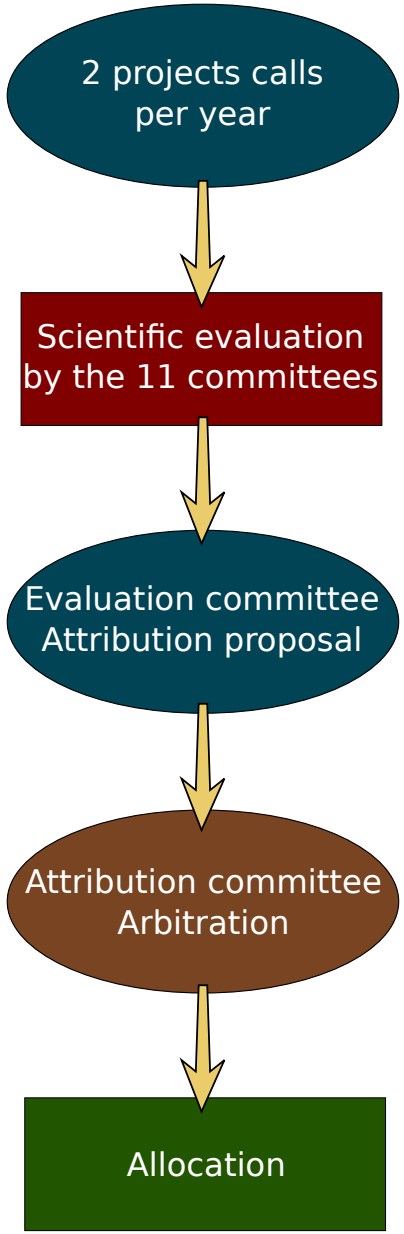
Hours allocated per CT for GPU partition (total = 1.968 Mh GPU)



GENCI - DARI

How to compute at HPC centers

- www.edari.fr
- $\simeq$ 817 million hours CPU
  (Real cost: $\simeq$ 0.01€/hour/core)
- $\simeq$ 1.6 million hours GPU
  (Real cost: $\simeq$ 0.59€/hour/GPU)
- 2 sessions a year (1-year projects)
- Preparatory access (a few tens of thousands hours, depending on the machine)



GENCI - Project

Admininistrative section

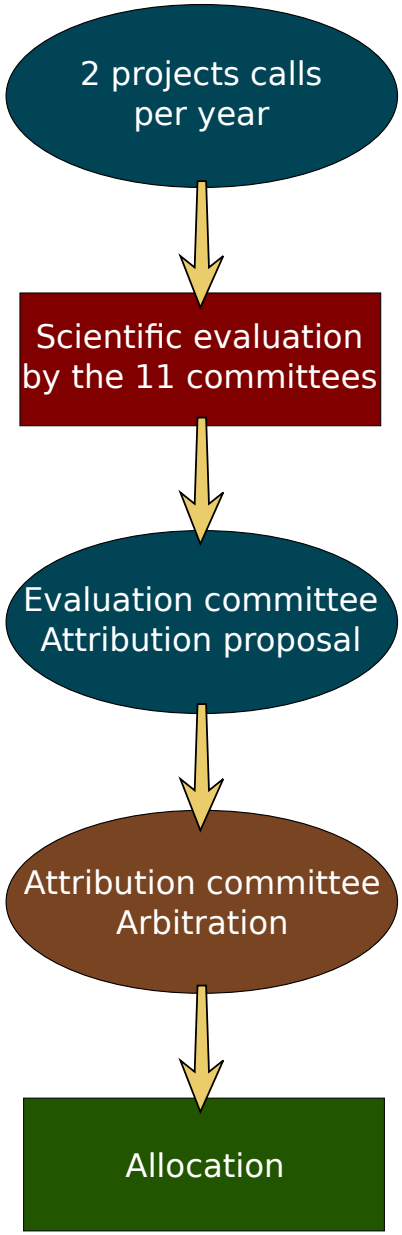
- Project title
- Thematic committee
- Laboratory info.
- Project manager info.
- Technical manager
- Project support (ANR, Tier2 center, industry, . . .)

Technical section

- Software description and performance
- Type of machine architecture needed
- How much data space?
- Experience of the team

Scientific section

- Project description
  - Abstract
  - State of the art
  - Research plan
  - Methods
- Justification of the needs
- Bibliography



Institut du Développement et des Ressources en Informatique Scientifique

- Institute for the Development and Resources in Scientific Computing
- CNRS UPS851 founded end of 1993
- Located in Orsay ($\simeq$20 km South-east of Paris on Paris-Saclay campus)
- Building constructed in 1969 (previously CIRCE)
- Simultaneously:
  - Computing center equiped with supercomputers among the most powerful
  - HPC center of excellence

IDRIS - Jean Zay

CPU partition

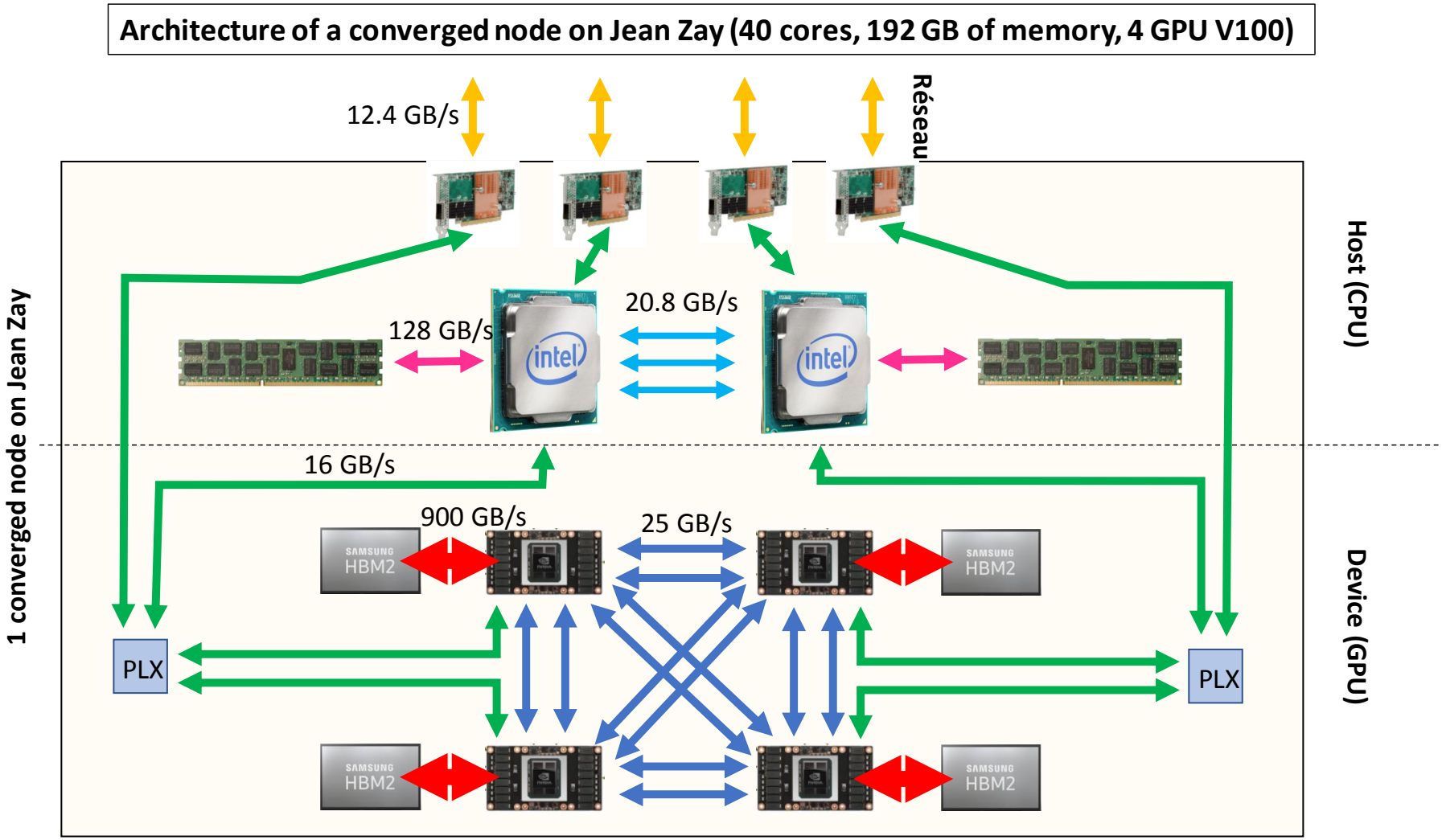
- 1528 nodes 2 x 20 cores Intel Cascade Lake 2.5 GHz
- 4.5 GB / core of memory (4 GB available to users)
- 1 OmniPath link 100 GB/s

GPU partition

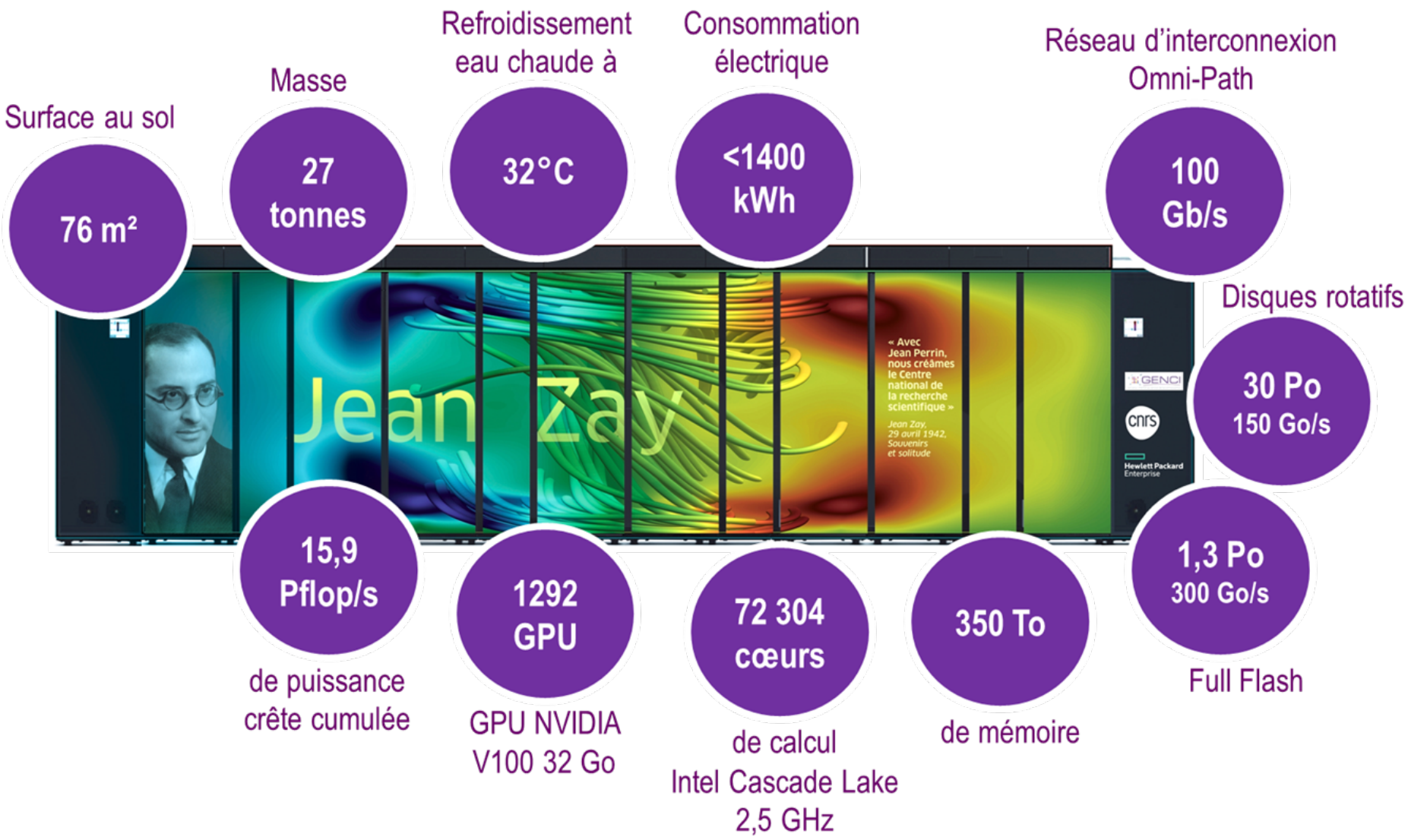
- 261 nodes 2 Intel Cascade Lake x 20 cores 2.5 GHz
- 4.5 GB / core of memory (4 GB available to users)
- 4 V100 NVidia GPU fully interconnected with NVLink2 (1044 GPUs)
- 4 OmniPath link 100 GB/s

Other nodes

- 4 pre/postprocessing nodes (3TB of memory; 4 Intel Skylake x12 cores 3.2 GHz; 1 V100 GPU)
- 5 visualization nodes (1 P6000 GPU)


Architecture of a converged node on Jean Zay (40 cores, 192 GB of memory, 4 GPU V100)
12.4 GB/s
Réseau
1 converged node on Jean Zay
Host (CPU)
Device (GPU)
128 GB/s
20.8 GB/s
16 GB/s
900 GB/s
25 GB/s
PLX
PLX
SAMSUNG HBM2
SAMSUNG HBM2
SAMSUNG HBM2
SAMSUNG HBM2

IDRIS - Jean Zay



A part of the machine is dedicated to research on Artificial Intelligence.

Resources

- Nodes
  - 162/296 nodes with 4 GPUs (depends on load)
  - 31 nodes with 8 GPUs (Facebook donation)
- Installed by IDRIS inside Conda environments:
  - Pytorch
  - Tensorflow
  - Caffe
- Shared space for common databases

User Support

- Important resources since the beginning of IDRIS
- $\simeq$ 15 engineers
- Hotline
  - By phone and e-mail
  - Monday-Friday, from 9 a.m to 6 p.m
  - Follow-up of problems
- Advanced support for projects
  - Porting, optimization (CPU, communications, memory, I/O)
  - From a few days to a few months
- Documentation on www.idris.fr
- Training courses
  - Languages (C,Fortran)
  - Parallelism (OpenMP, MPI, hybrid)
  - Free for academics
  - Industry welcome, but paying (CNRS Formation Entreprises)

IDRIS - Training courses

2020 Trainings

MPI	03.02.2020	4 days	Confirmed
Advanced Fortran	10.03.2020	4 days	Announced
OpenMP	18.03.2020	3 days	Announced
SIMD Vectorization	25.03.2020	1 jour	Announced
Introduction to OpenACC for GPU	26.03.2020	2 days	Announced
Expert Fortran	01.04.2020	3 days	Announced
Basics Fortran	13.05.2020	3 days	Announced
C Language	25.05.2020	5 days	Announced
Hybrid MPI/OpenMP programming	04.06.2020	2 days	Announced
OpenMP / MPI	15.06.2020	5 days	Announced
Advanced Fortran	23.06.2020	4 days	Announced
Introduction to OpenAcc for GPU	29.06.2020	2 days	Announced
MPI	28.09.2020	4 days	Announced
OpenMP	14.10.2020	3 days	Announced
Basics Fortran	21.10.2020	3 days	Announced
HPC Debugging	19.11.2020	2 days	Announced
Advanced Fortran	24.11.2020	4 days	Announced
Vectorisation SIMD	30.11.2020	1 jour	Announced
Introduction to OpenACC for GPU	01.12.2020	2 days	Announced
PETSc	03.12.2020	2 days	Announced

To register for a training course

https://cours.idris.fr/

Supercomputing center specificities

SuperComputer specifities - A little Poll

What kind of machine do you use?

- Work station
- Local cluster
- Computing center (Mesocentre, National, European)

SuperComputer specifities - A little Poll

What kind of machine do you use?

- Work station
- Local cluster
- Computing center (Mesocentre, National, European)

What differences?

SuperComputer specifities - A little Poll

What kind of machine do you use?

- Work station
- Local cluster
- Computing center (Mesocentre, National, European)

What differences?

What I see

- No direct access to the system
  - Softare Installation
  - Some parameters are not accessible
- Much more users
  - Management of jobs: A batch scheduler (Can you guess the number of jobs that ran on Ada in 2018?)
  - Syntax for batch scheduler (example)
  - Access to a list of software
- No physical access
  - Need to ask the support team for help

SuperComputer specifities - A little Poll

What kind of machine do you use?

- Work station
- Local cluster
- Computing center (Mesocentre, National, European)

What differences?

What I see

- No direct access to the system
  - Softare Installation
  - Some parameters are not accessible
- Much more users
  - Management of jobs: A batch scheduler (Can you guess the number of jobs that ran on Ada in 2018? *650 000*)
  - Syntax for batch scheduler (example)
  - Access to a list of software
- No physical access
  - Need to ask the support team for help

Job Scheduler - Options

Schedulers

- LoadLeveler
- SLURM (At IDRIS)
- PBS
- Univa Grid Engine
- ...

Usage

- Allows sheduling numerous jobs
- Allocate resources required
  - Number of nodes, cores
  - Memory
  - Time
- Manages priorities, dependencies, etc...
- Specific syntax (usually shell comments)
- Commands to be run

Example: Jean Zay queue 21/01

- 455 waiting
- 569 running
- 89 users

Job Scheduler - Options

Schedulers

- LoadLeveler
- SLURM (At IDRIS)
- PBS
- Univa Grid Engine
- . . .

Usage

- Allows sheduling numerous jobs
- Allocate resources required
  - Number of nodes, cores
  - Memory
  - Time
- Manages priorities, dependencies, etc. . .
- Specific syntax (usually shell comments)
- Commands to be run

Example: Jean Zay queue 21/01

- 455 waiting
- 569 running
- 89 users

Partition and QoS

Different partition to submit jobs and several QoS to fine tune.

```
QoS CPU:
              Name     MaxWall                  MaxTRES                 GrpTRES
------------------- ----------- ------------------------ ------------------------
        qos_cpu-dev    02:00:00                                        cpu=80000
         qos_cpu-gc    20:00:00
         qos_cpu-t3    20:00:00                cpu=40960
         qos_cpu-ex    20:00:00               cpu=120960
        qos_cpu-dir
         qos_cpu-cp
         qos_cpu-t4  4-04:00:00                   cpu=80                cpu=10240
QoS GPU:
              Name     MaxWall                  MaxTRES                 GrpTRES
------------------- ----------- ------------------------ ------------------------
        qos_gpu-dev    02:00:00                                 cpu=5120,gres/gpu=256
         qos_gpu-gc    20:00:00
         qos_gpu-t3    20:00:00     cpu=7680,gres/gpu=384
         qos_gpu-ex    20:00:00   cpu=20480,gres/gpu=1024
        qos_gpu-dir
         qos_gpu-cp
         qos_gpu-t4  4-04:00:00         cpu=80,gres/gpu=8     cpu=2560,gres/gpu=256
```

Job Scheduler - Options

Schedulers

- LoadLeveler
- SLURM (At IDRIS)
- PBS
- Univa Grid Engine
- . . .

Usage

- Allows sheduling numerous jobs
- Allocate resources required
  - Number of nodes, cores
  - Memory
  - Time
- Manages priorities, dependencies, etc. . .
- Specific syntax (usually shell comments)
- Commands to be run

Example: Jean Zay queue 21/01

- 455 waiting
- 569 running
- 89 users

Partition and QoS

Different partition to submit jobs and several QoS to fine tune.

```
QoS CPU:
              Name     MaxWall                  MaxTRES                 GrpTRES
------------------- ----------- ------------------------ ------------------------
        qos_cpu-dev    02:00:00                                        cpu=80000
         qos_cpu-gc    20:00:00
         qos_cpu-t3    20:00:00                cpu=40960
         qos_cpu-ex    20:00:00               cpu=120960
        qos_cpu-dir
         qos_cpu-cp
         qos_cpu-t4  4-04:00:00                   cpu=80                cpu=10240
QoS GPU:
              Name     MaxWall                  MaxTRES                 GrpTRES
------------------- ----------- ------------------------ ------------------------
        qos_gpu-dev    02:00:00                                 cpu=5120,gres/gpu=256
         qos_gpu-gc    20:00:00
         qos_gpu-t3    20:00:00     cpu=7680,gres/gpu=384
         qos_gpu-ex    20:00:00   cpu=20480,gres/gpu=1024
        qos_gpu-dir
         qos_gpu-cp
         qos_gpu-t4  4-04:00:00         cpu=80,gres/gpu=8     cpu=2560,gres/gpu=256
```

Job Scheduler - Options

Schedulers

- LoadLeveler
- SLURM (At IDRIS)
- PBS
- Univa Grid Engine
- . . .

Usage

- Allows sheduling numerous jobs
- Allocate resources required
  - Number of nodes, cores
  - Memory
  - Time
- Manages priorities, dependencies, etc. . .
- Specific syntax (usually shell comments)
- Commands to be run

Example: Jean Zay queue 21/01

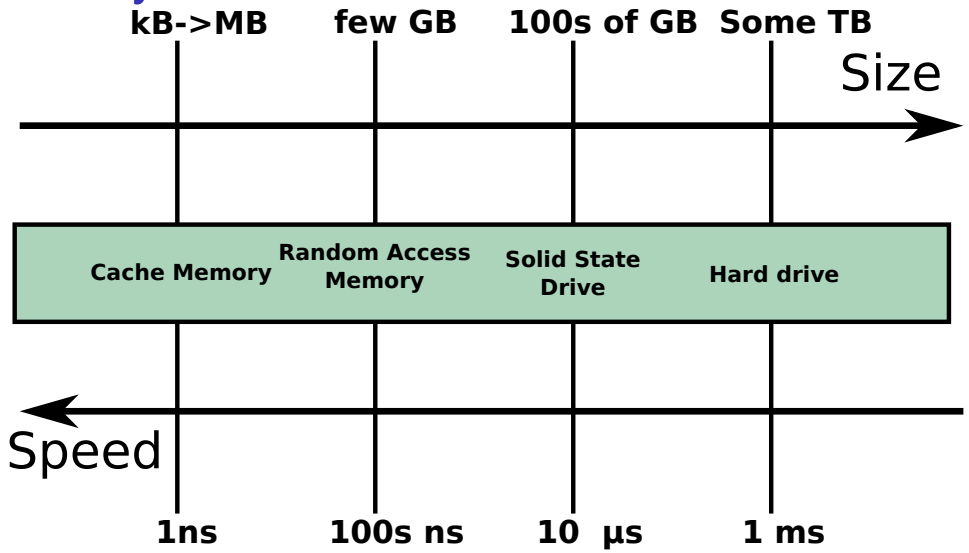
- 455 waiting
- 569 running
- 89 users

Submission script

```
#!/bin/bash
#SBATCH --nodes=1                # Number of Nodes
#SBATCH --ntasks-per-node=1      # Number of MPI tasks per node
#SBATCH --cpus-per-task=40       # Number of OpenMP threads
#SBATCH --hint=nomultithread     # Disable hyperthreading
#SBATCH --job-name=b12           # Jobname
#SBATCH --output=%x.o%j          # Output file %x is the jobname, %j the jobid
#SBATCH --error=%x.o%j           # Error file
#SBATCH --account=sos@cpu        # Account
#SBATCH --time=20:00:00

# Print environment
env

# Manage modules
module purge
module load gaussian/g16-revC01

export GAUSS_SCRDIR=$JOBSCRATCH
g16 -c="0-39" -m=140GB < cyanocobalamine.com
```

Technical Aspects - Memory

Different kinds of memory



Why is it important?

- Strongly affects code perfomance
  - Try to keep everything in RAM
  - OS uses RAM for disk cache!
  - You should tune memory settings!
- Users should set up their job having that in mind
- Some limitations on HPC centers (Size and number of files)

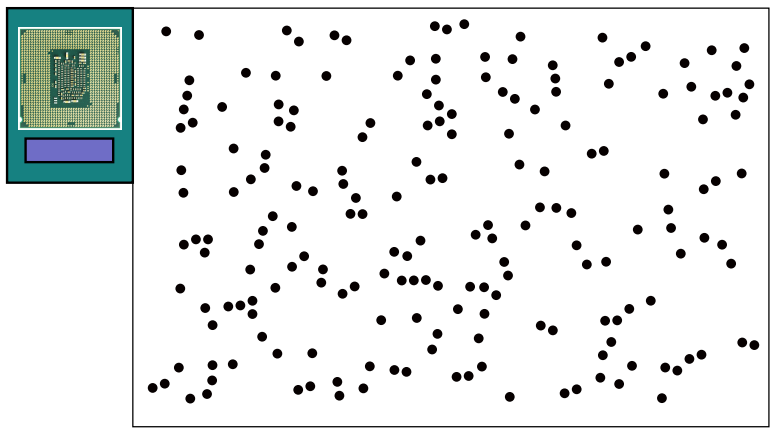
Examples

- VASP, 289 atoms, 1024 processes, 2 geometry steps
  - With I/O ($\simeq 1GB$): 662s
  - Without I/O: 338s
- Crystal, 289 atoms, 1024 processes, 1 SCF
  - PCrystal: 12,300 files ; crashed after 420s
  - PCrystal: 1040 files ; 1239s
  - PCrystal optimized: 1040 files ; 1117s

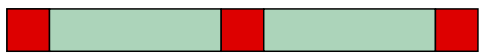
Parallelism - Atoms in a box

We have some atoms in a box and we want to run a molecular dynamics simulation. Some parts of the code are serial (α) and some are parallelizable ($1 - \alpha$).

Our reference is the purely sequential code:



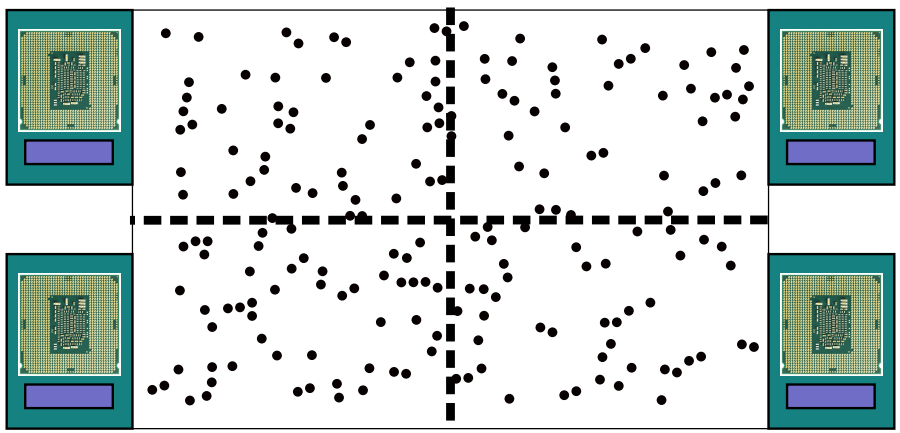
TtS:



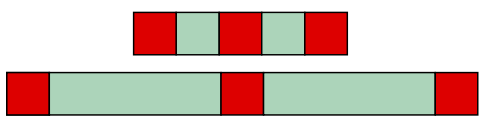
Parallelism - Atoms in a box

We have some atoms in a box and we want to run a molecular dynamics simulation. Some parts of the code are serial (α) and some are parallelizable ($1 - \alpha$).

Lets divide our box between 4 cores



TtS:
Ref:

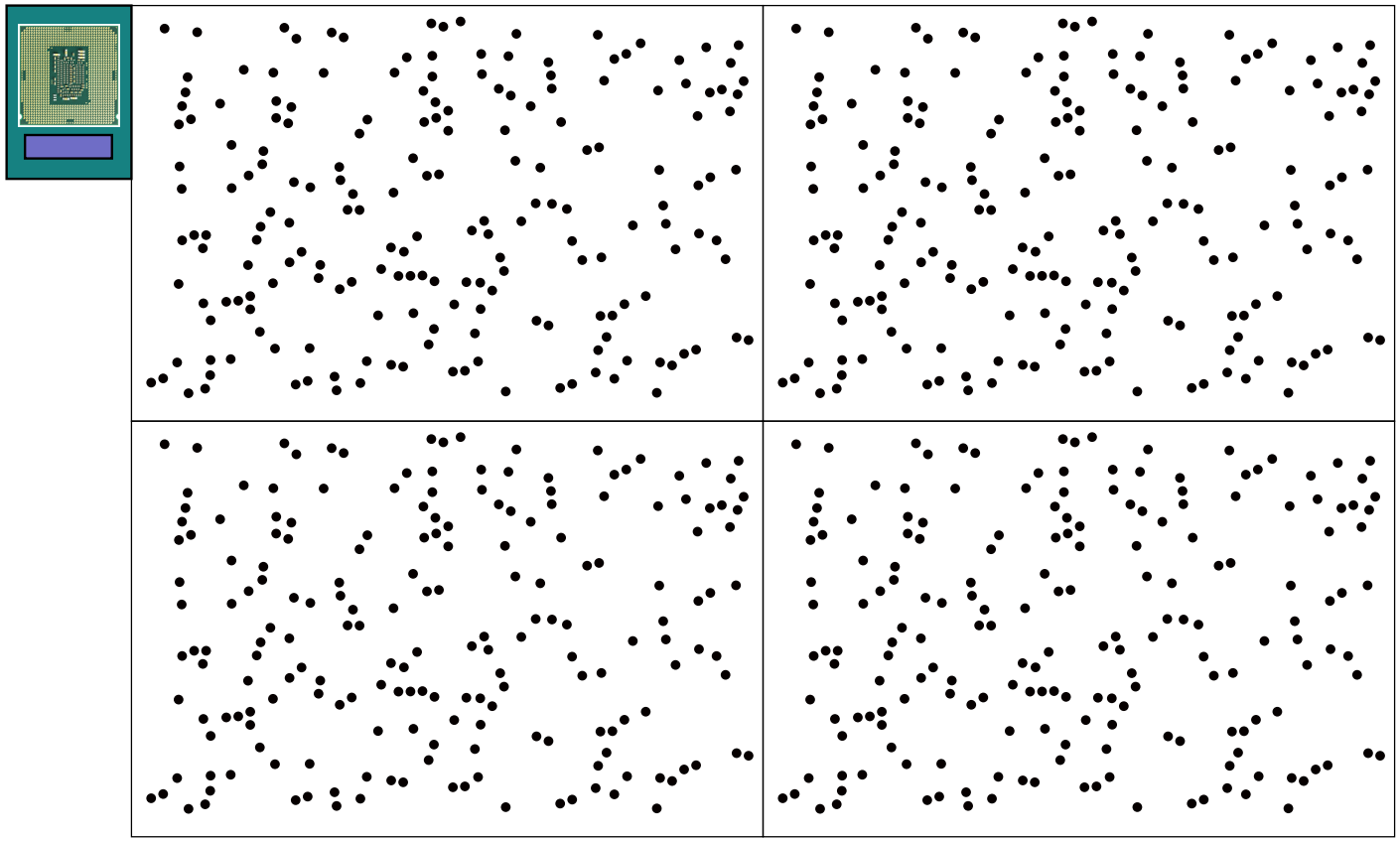


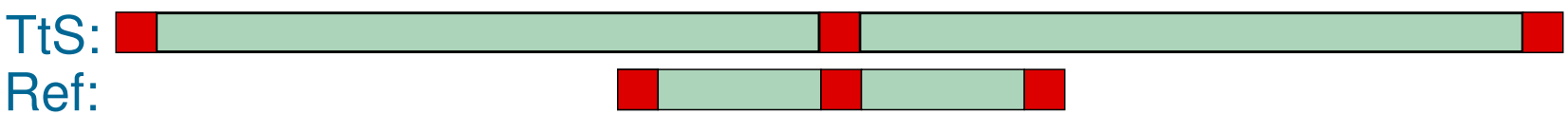
This is a case of *Strong Scaling Parallelism*.

Parallelism - Atoms in a box

We have some atoms in a box and we want to run a molecular dynamics simulation. Some parts of the code are serial (α) and some are parallelizable ($1 - \alpha$).

We can also increase the size of our system

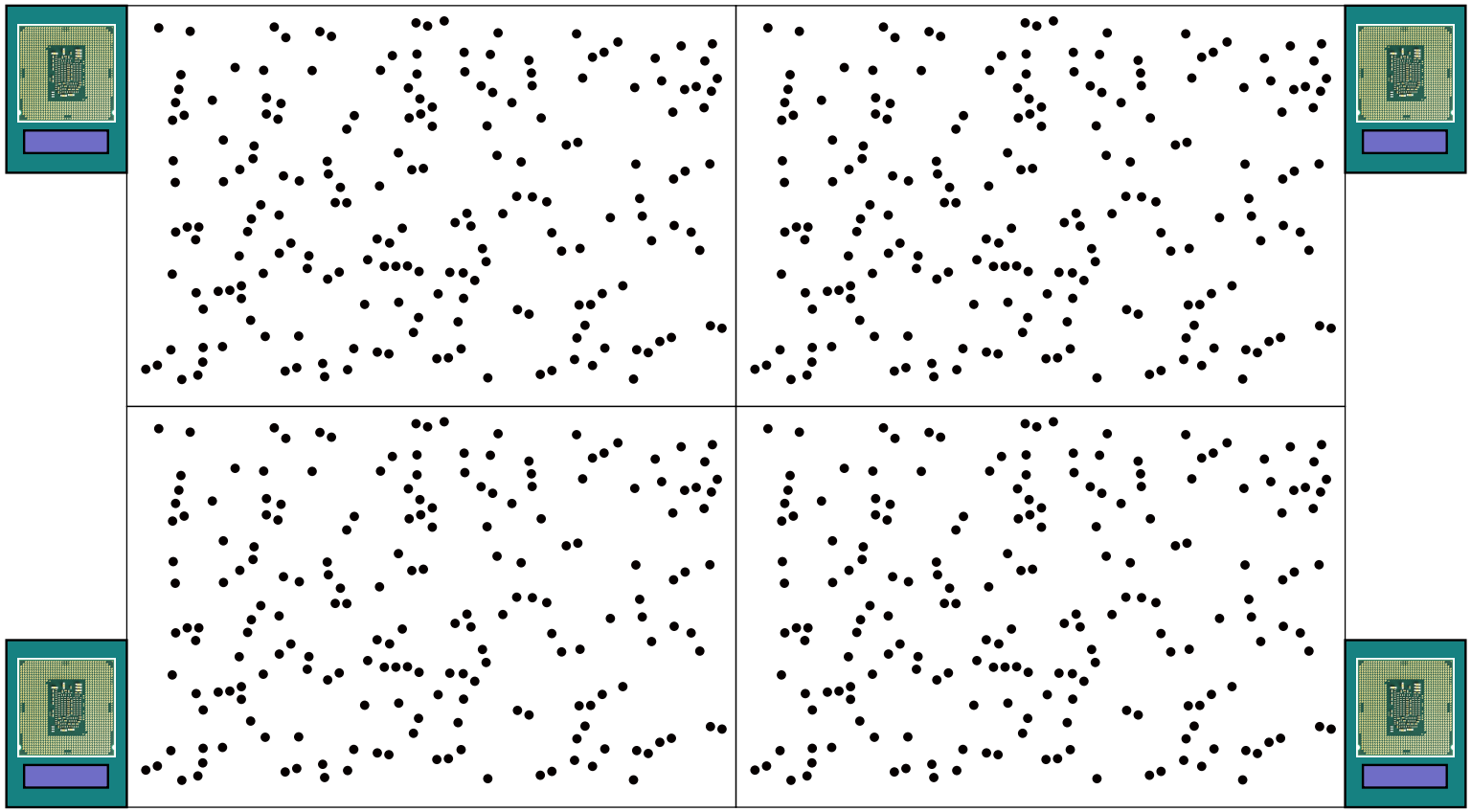




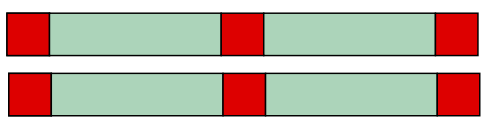
Parallelism - Atoms in a box

We have some atoms in a box and we want to run a molecular dynamics simulation. Some parts of the code are serial (α) and some are parallelizable ($1 - \alpha$).

And give some work to each core



TtS:
Ref:



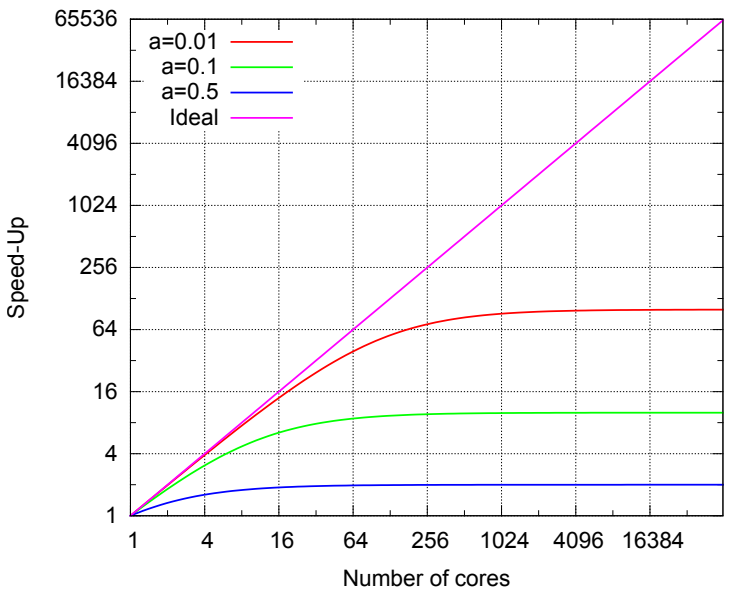
This is a case of *Weak Scaling Parallelism*.

Parallelism - 2 laws to rule them all

Speedup: S(n) = $\frac{T_{seq}}{T_{para}(n)}$ where n is the number of cores used.

Strong scaling: Amdahl's law

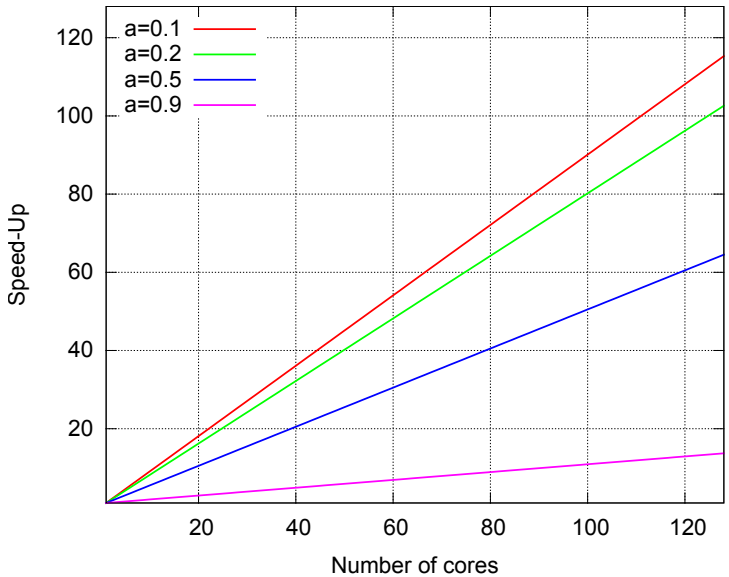
- Fixed-size problem
- A part α of the code is serial
- $S(n) = \frac{1}{\alpha + \frac{1-\alpha}{n}}$
- Perfect scaling: S(n) = n



- Quite pessimistic. Real applications usually not fixed size

Weak scaling: Gustason-Barsis' law

- Size of the problem proportional to the number of cores
- A part α of the code is serial
- $S(n) = n - \alpha(n-1)$



- Difficult to achieve in real applications
- Optimistic

Parallelism - Atomistic Simulations

What can you parallelize?

System splitting	Wave function, e^- density	Mathematics functions
• Multiple images • NEB • Numerical frequencies	• Basis set coefficients • Bands • Grid	• Functions (FFT,. . .) • Eigen solvers • Matrices handling

Parallelism - Embarassing parallelism

Some problems show a straightforward parallelism where the different parts can be run independantly. For example, the Normal Modes of a molecular system computed numerically are completly independant.
In order to solve the problem, no communication between the cores/nodes is needed.
This is called an *Embarassingly Parallel* problem. And it is quite embarassing for SuperComputing centers.

Why?

Supercomputers have a very efficient (and expansive) network for the communication between nodes. *In this case, the resources are wasted.*

Parallelization - Summary

What you can have

- Reduce time to solution for a fixed size problem
- More memory
  - Increase size of systems
  - New kinds of computations

Limitations

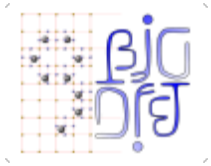
- Extensibility not infinite
- Parallelism overhead
- Balance of workload between threads, processes
- Overall time consumption is higher
- Code adaptation

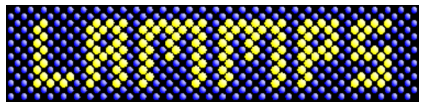
Parallelism is critical

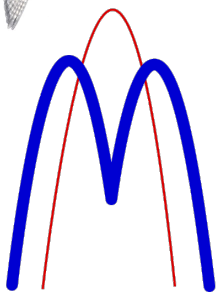
- Do not expect improvement thanks to hardware
- Fits the architectures of machines
- Larger simulations

Atomistic Simulations at IDRIS

Software - Installed at IDRIS

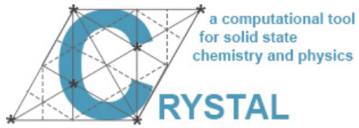


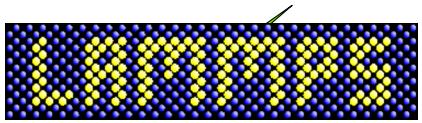


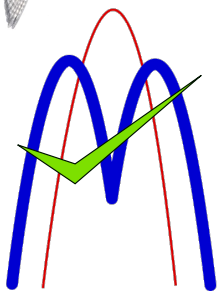


On wikipedia: roughly 125 different pieces of software

Software - Installed at IDRIS





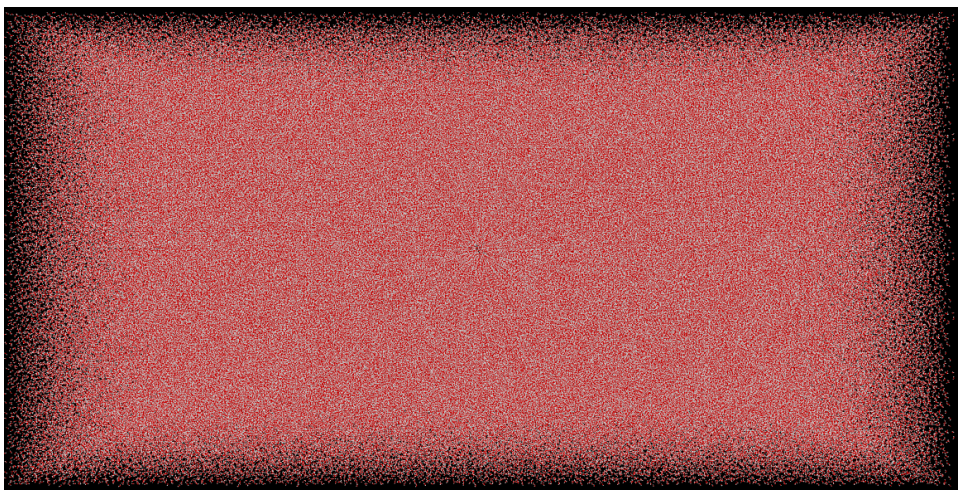


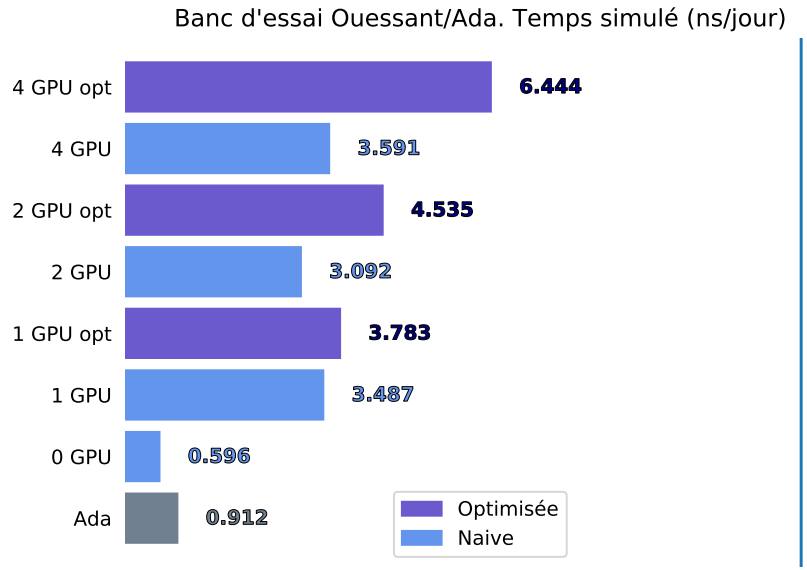
Some are GPU ready

Benchmark Ouessant - GROMACS - water

Description

- Water box (1M molecules)
- PME
- Classical molecular dynamics
- Time step: 2fs
- Canonical Ensemble





Higher the value, better the performance!

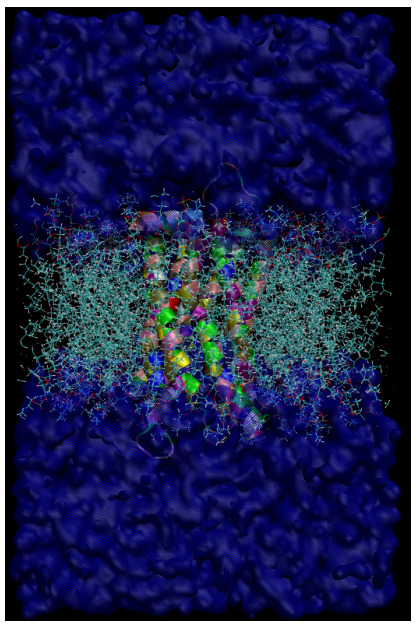
Comments

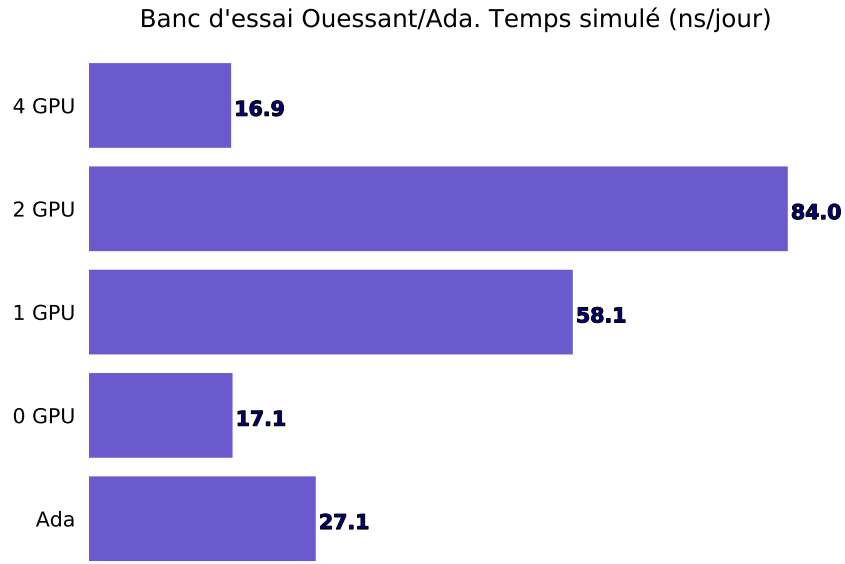
- Simple case. No load imbalance

Benchmark Ouessant - GROMACS - GMX

Description

- Protein + Lipid bilayer + water box (58000 atomes)
- PME
- Classical molecular dynamics
- Time step: 2fs
- Isobaric-Isothermal Ensemble





Higher the value, better the performance!

Comments

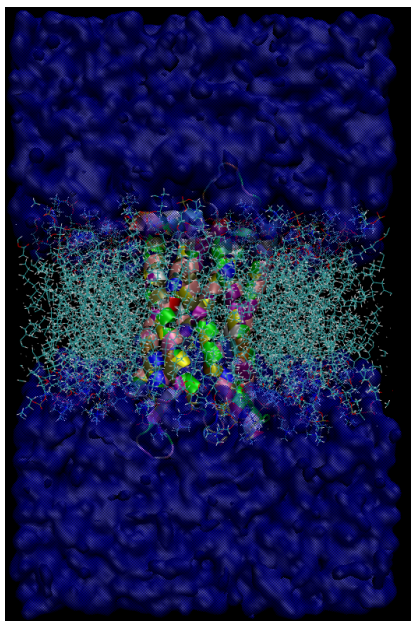
- Real case
- Too small to get good performance on more than 2 GPU.

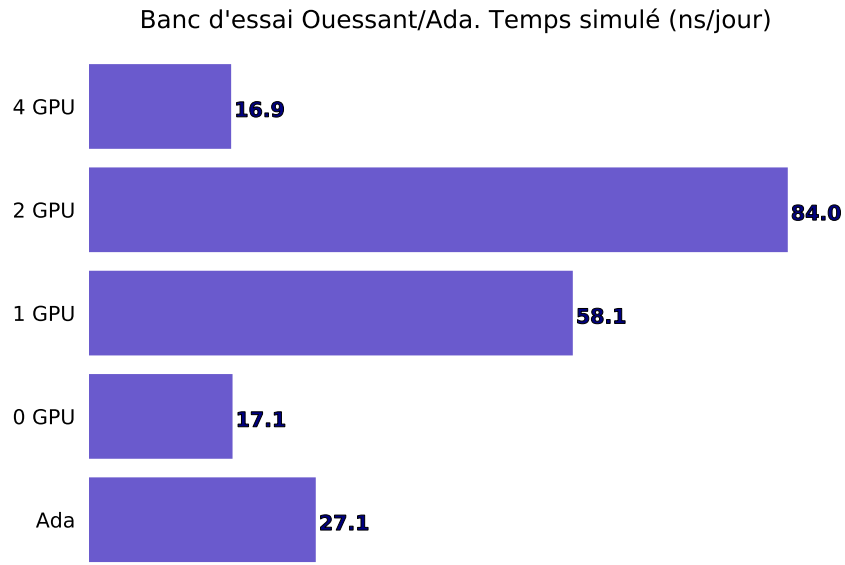
For a new system/machine: *Perform some tests!*

Benchmark Ouessant - GROMACS - GMX

Description

- Protein + Lipid bilayer + water box (58000 atomes)
- PME
- Classical molecular dynamics
- Time step: 2fs
- Isobaric-Isothermal Ensemble





Higher the value, better the performance!

Comments

- Real case
- Too small to get good performance on more than 2 GPU.

For a new system/machine: ***Perform some tests!***

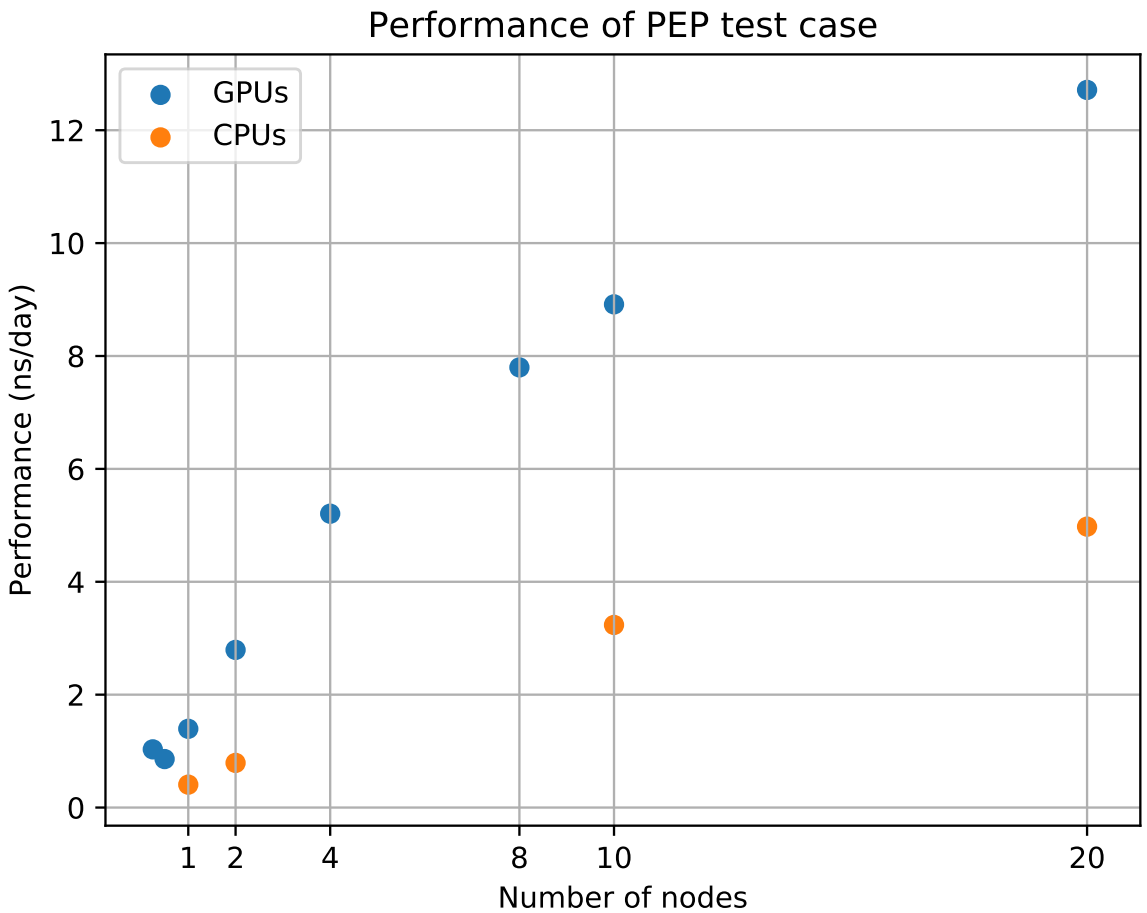
Examples - GROMACS: PEP

Description

- GROMACS 2020
- Protein + membrane + water
- 12.5 millions of atoms
- MD with time step: 2 fs
- PME
- Test MPI, OpenMP threads, GPU, Scalability

Analysis

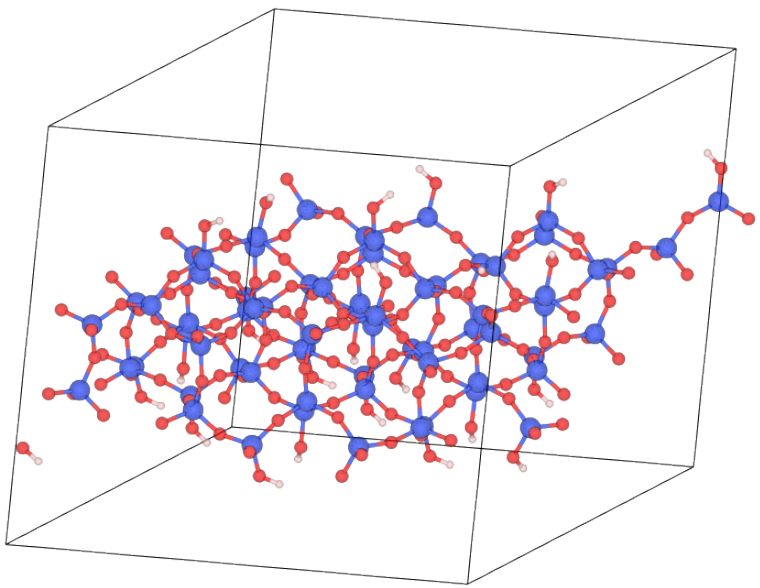
- $\simeq$ 100 configurations
- Better performance on GPU
- Scalability average
- Strange behavior on a quarter of node
- Still some investigations

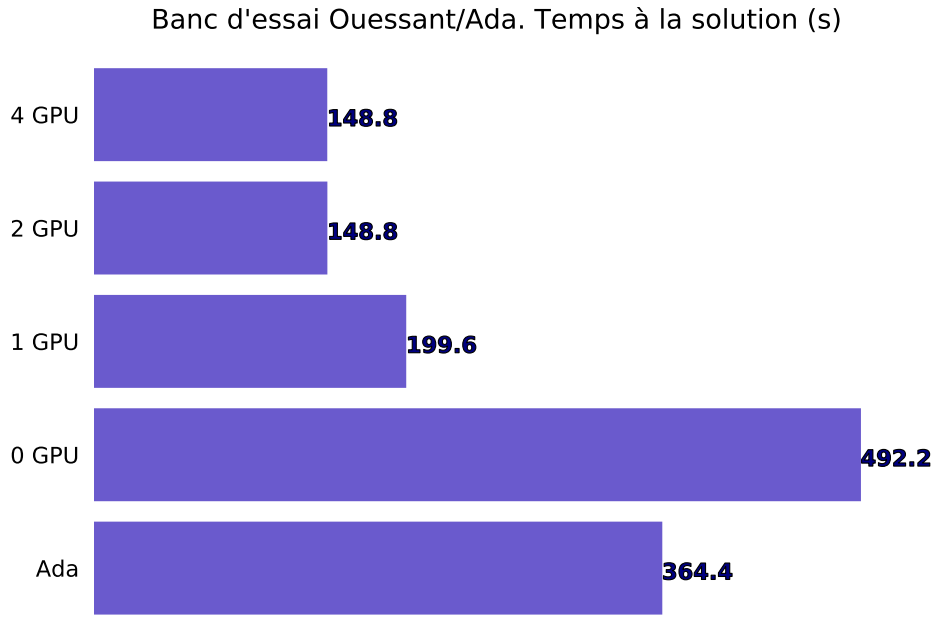


Benchmark Ouessant - VASP - silicaIFPEN

Description

- Force calculation with 28 electronic steps
- 328 atoms
- 1 KPT
- GGA
- VASP standard version





Lower the value, better the performance!

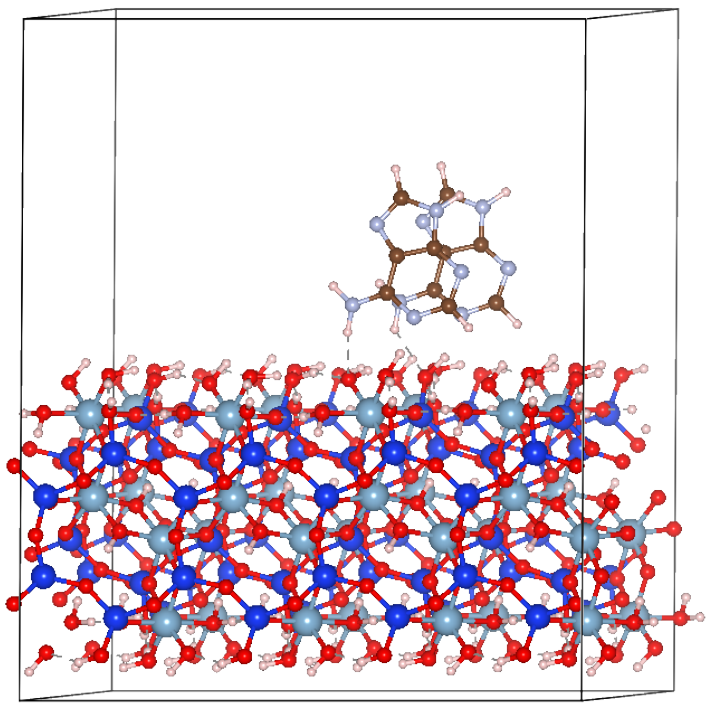
Comments

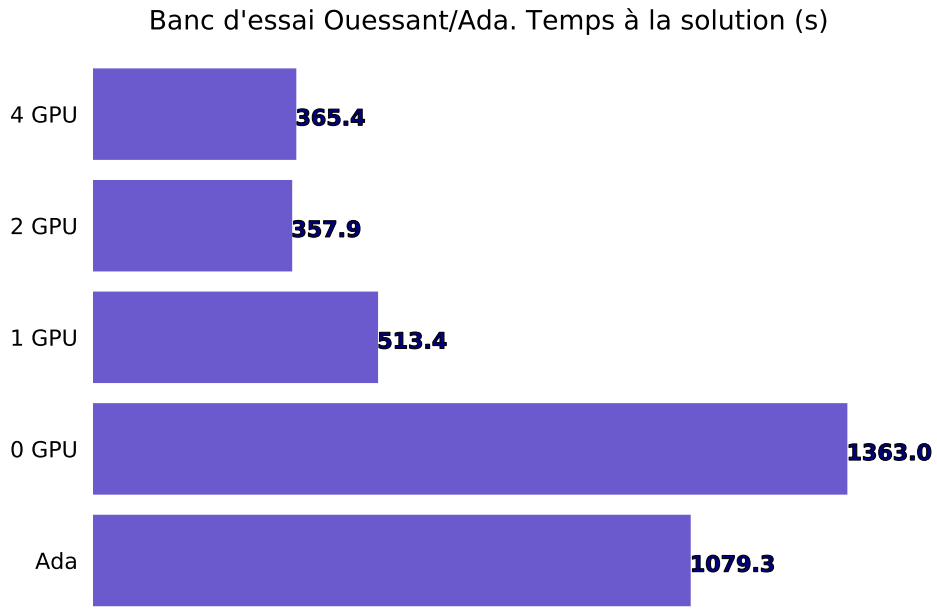
- Ouessant gamma-only version: 349.1 s

Benchmark Ouessant - VASP - TSUB

Description

- Force calculation with 24 electronic steps
- 521 atoms
- 1 KPT
- GGA+DFTD3
- VASP standard version





Lower the value, better the performance!

Comments

- Ouessant gamma version : 805.4 s

Take home messages - User support

If there is something strange in

- How your code runs
- Your code installation
- Administration of account

Who you gonna call?

- IDRIS support
  - By email: assist@idris.fr
  - By phone: 01 69 35 85 55

Larger projects

- Advanced support
  - Large scale code porting
  - Parallelization of code
  - New algorithm implementation
  - Debugging and optimization
  - Code coupling

Conclusions

Architectures

- More cores (Top500 #1: $\simeq$ 2.4 million cores)
- Less power/CPU (Green500)
- Use of accelerators (Top500 #1: $\simeq$ 27 650 GPU)
- Efficient networks
- Software has to adapt

Tune your computation

- Test parallelism parameters of your software
- Use memory settings wisely
- Adapt the size of your system

Tips

- Know your software
- Know your environment
- Perform some tests
- RTFM (Read the famous manual)

Acknowledgments

Thanks to

- Denis Girou
- Pierre-François Lavallée
- Fabien Leydier
- Myriam Peyrounette
- Pascal Voury
- All the IDRIS support team

And you for your attention!

Any questions? thibaut.very@idris.fr