



Inria

ENST2



INSTITUT
POLYTECHNIQUE
DE PARIS



Cours SIM3

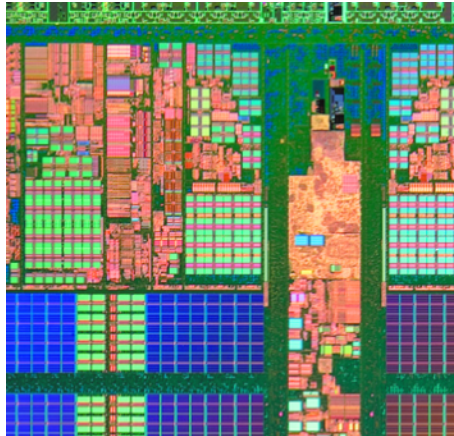
Initiation au calcul haute performance

Contexte HPC français et européen — Introduction à MPI et au GPGPU

Rappel : Vision globale de calcul haute performance ... (RAPPEL)

.....

Cœur



Calcul séquentiel
Calcul vectoriel

Séances 1 et 2

Questions importantes:
Comment **gérer les opérations** ?
(*Quel cœur fait quoi ? Quand ?*)
Comment **gérer les données**
nécessaires pour les opérations ?

Processeur (plusieurs cœurs)

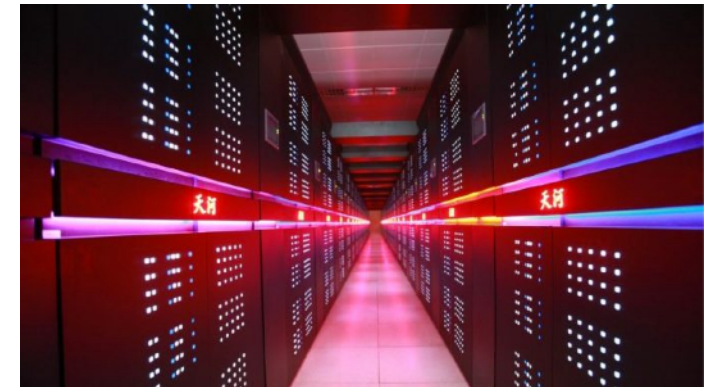


Calcul parallèle
à **mémoire partagée**

Les cœurs peuvent
travailler en parallèle
sur des tâches différentes.
Ils partagent des mémoires
rapides (RAM + L2 ou L3)

Séances 3 et 4

Cluster (plusieurs processeurs)



Calcul parallèle
à **mémoire distribuée**

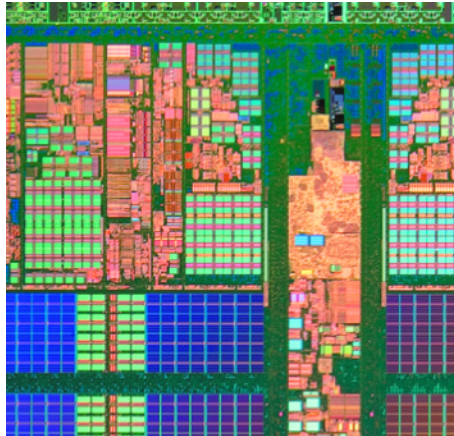
Les processeurs peuvent
travailler en parallèle
sur des tâches différentes
Les données sont distribuées
entre les RAM des processeurs,
qui doivent communiquer
(avec par ex. **MPI**).

Séance 5 (intro)

**Cours MS01 et AMS-I03
(3A ModSim et M2 AMS)**

Vision plus globale de calcul à haute performance ... (UPDATE)

Cœur



Calcul séquentiel
Calcul vectoriel

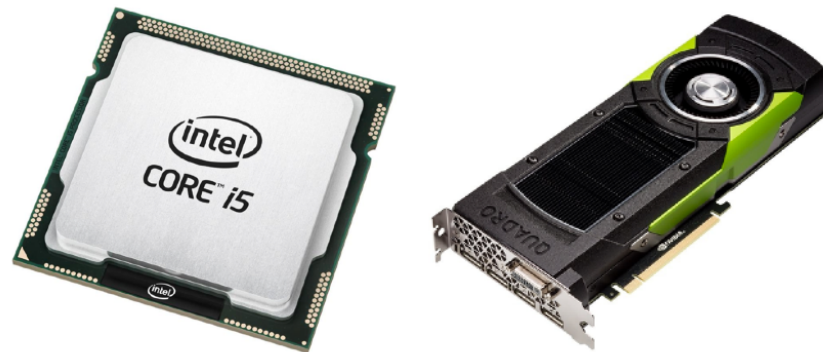
Séances 1 et 2

Questions importantes:

Comment **gérer les opérations** ?
(*Quel cœur fait quoi ? Quand ?*)

Comment **gérer les données**
nécessaires pour les opérations ?

Processeur et co-processeur (plusieurs cœurs)



Calcul parallèle
à mémoire partagée
et GPGPU

Les cœurs peuvent
travailler en parallèle
sur des tâches différentes.

Ils partagent des mémoires
rapides (RAM, L2, L3, ...)

Séances 3, 4 et 5

**Cours AMS-I03
(3A / M2 AMS)**

Cluster (plusieurs processeurs)



Calcul parallèle
à mémoire distribuée

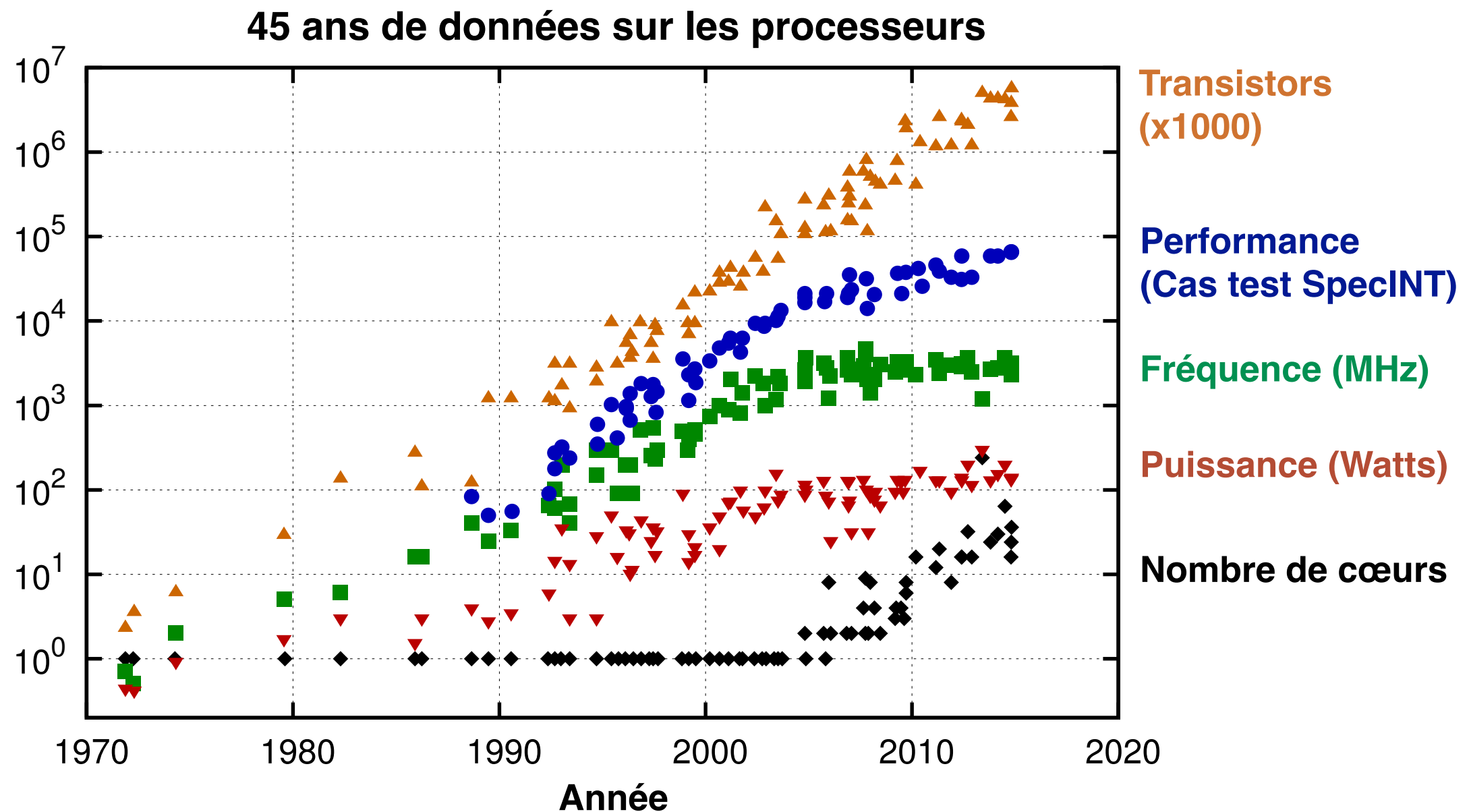
Les processeurs peuvent
travailler en parallèle
sur des tâches différentes

Les données sont distribuées
entre les RAM des processeurs,
qui doivent communiquer
(avec par ex. **MPI**).

Séance 5 (intro)

**Cours MS01 et AMS-I03
(3A ModSim et M2 AMS)**

Évolution des processeurs (RAPPEL)



Jusqu'aux années ~2000, les performances ↗ grâce une ↗ de la fréquence.

À cause de limitations physiques (*miniaturisation vs. dissipation*), il est difficile d'↗ encore la fréquence.

Pour augmenter les performances, le nombre de cœurs est ↗ .

Pour des raisons économiques, la fréquence est parfois ↘ .

} **Tendance**

Classement des supercalculateurs (RAPPEL) vers 1 EFLOP/s 10^{18}



<https://top500.org/>

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,730,112	1,102.00	1,685.65	21,100
2	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	LUMI - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	Leonardo - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610
5	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

Rmax *Performance effective* (débit arithmétique évalué avec la librairie LINPACK — librairie d'algèbre linéaire)

Rpeak *Performance de crête* (débit arithmétique de crête — max. théorique sur base des caractéristiques physiques)

Power Consommation électrique de l'installation

$R_{max} < R_{peak}$

Objectif des programmeurs :
 R_{max} aussi proche de R_{peak}

Introduction au calcul parallèle à mémoire distribuée

Introduction au calcul sur GPU

Contexte HPC français et européen

Le supercalculateur standard ... à l'ENSTA



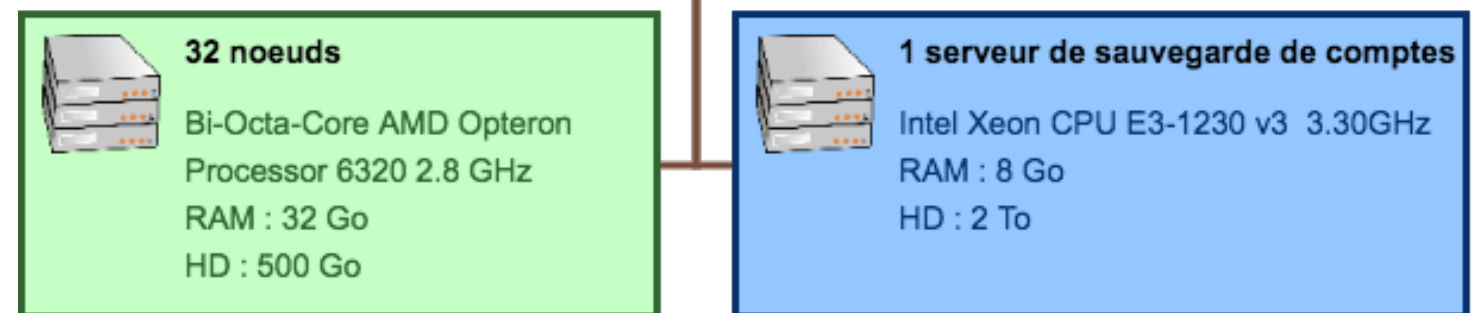
Cluster **gin** de l'unité de mathématiques appliquées (ENSTA)

Le cluster **gin** était composé de ...

32 nœuds, chacun avec ...

- un processeur (*CPU*)
- de la mémoire vive (*RAM*)
- un disque dur (*HDD*)
- des connexions

un **réseau d'interconnexion**



<https://uma.ensta-paris.fr/resources/hpc.html>

Le supercalculateur standard ... à l'ENSTA

Performances du cluster gin

	Modèle	# Cœurs	Fréquence	Débit arithmétique par processeur	Débit arithmétique total
Processeur (CPU)	AMD Opteron Processor 6320	8	2.8 GHz	~ 160 GFLOP/s	~10 TFLOP/s



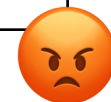
	Capacité mémoire par nœud	Capacité mémoire total
Mémoires caches (L3 ...)	~24 MB	—
Mémoire vive (RAM)	32 GB	~1 TB
Disque dur (HDD)	500 GB	16 TB

Capacité de calcul ↗



Capacité de mémoire ↗

	Bande passante
Transferts entre caches L3, L2, L1	~100 GB/s
Transferts dans le nœud (RAM/Proc)	~10 GB/s
Réseau d'interconnexion (ethernet)	1 GB/s



Taux de transfert ↘

kilo (k)	10 ³	téra (T)	10 ¹²
mega (M)	10 ⁶	péta (P)	10 ¹⁵
giga (G)	10 ⁹	exa (E)	10 ¹⁸

Attention : Ces chiffres sont probablement approximatifs (sources pas fiables), mais les ordres de grandeur devraient être corrects.

Le supercalculateur standard ... au mesocentre IDCS

General view



Front



Back



Cluster Cholesky — IDCS mesocentre — IP Paris

http://meso-ipp.gitlab.labos.polytechnique.fr/user_doc/

Le supercalculateur standard ... au mesocentre IDCS



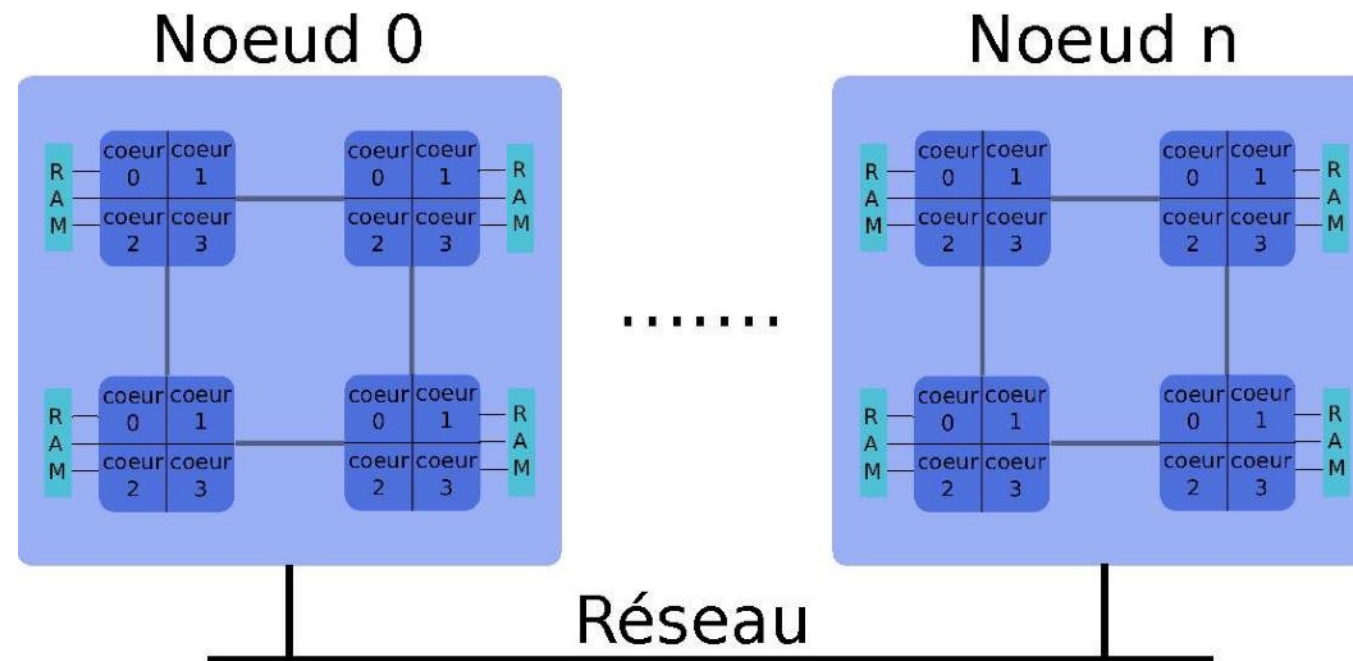
The cluster Cholesky is composed of ...

- ▶ **43 nodes:**
 - 1 login front-end node
 - 40 “CPU nodes” with 2 CPUs
 - 2 “GPU nodes” with 2 CPUs and 4 GPUs
- ▶ A **parallel file system** BeeGFS: 385 TB of usable space
- ▶ A Mellanox InfiniBand HDR (*High Data Rate*) **interconnection network** 200Gb/s

	# Cores	Frequency	Arithmetic throughput	Quantity
CPU : Intel Xeon CPU Gold 6230	20	2.1 GHz	~ 1 715 GFLOPS/s	84
GPU : Nvidia Tesla V100	5 120		~ 14 TFLOP/s	8

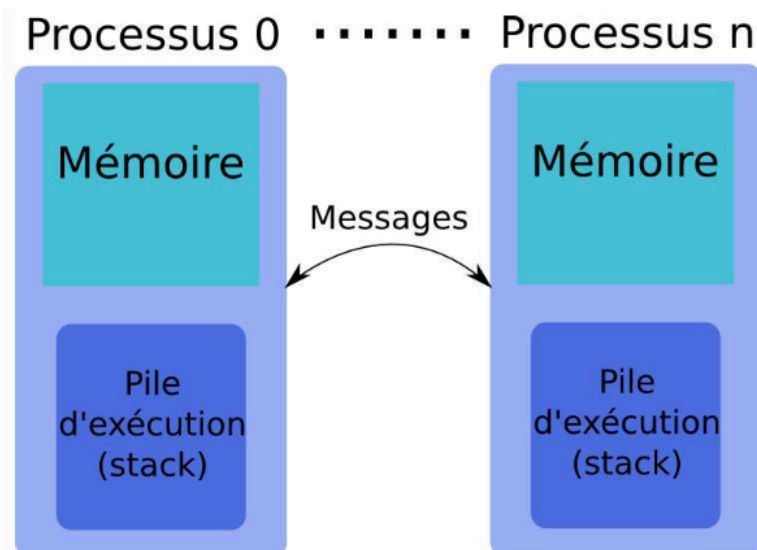
Calcul parallèle à mémoire distribuée *vs* mémoire partagée

Architecture parallèle

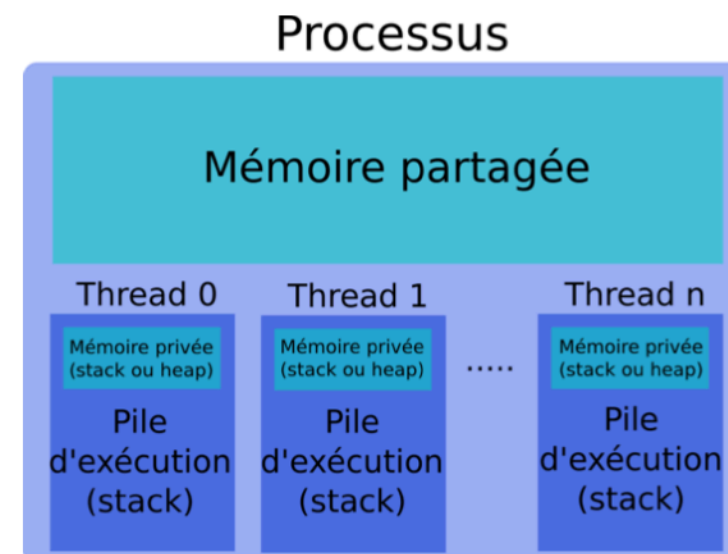


Modèles de programmation

*Mémoire distribuée (e.g. **MPI**)*



*Mémoire partagée (e.g. **OpenMP**)*



MPI « Message Passing Interface »

.....

- Protocole de communication standard pour la programmation d'applications parallèles sur **architectures à mémoire distribuée**.
- MPI est rapidement devenu un standard, supporté par toutes les architectures.
- Les spécifications MPI sont gérées par un **consortium international**, le **MPI Forum**, composé d'académique, de grands laboratoires et d'entreprises (*Intel, Cray, ATOS, Microsoft ...*).

Version		Spécifications	Commentaire
1991			Début des discussions pour un nouveau standard
1994	MPI 1.0	236 pages	Communications point-à-point, communications collectives, types dérivés, ...
1995	MPI 1.1	...	
2008	MPI 1.2	...	
1997	MPI 2.0	370 pages	Communications “one-side”, gestion dynamique de processus, entrées-sorties parallèles, ...
2008	MPI 2.1	...	
2009	MPI 2.2	...	
2012	MPI 3.0	852 pages	Communications collectives non-bloquantes, interfaçage d'outils externes, ...
2015	MPI 3.1	...	
	MPI 4.0		En discussion ...

MPI, en pratique ...

1. Compilateur parallèle

- Il s'agit d'un compilateur classique (gcc, icc, etc.) qui appelle automatiquement la librairie mpi.

```
>> mpicc myCode.c  
>> mpiicc myCode.c
```

2. Fonctions, constantes et types spécifiques

- Il faut charger les entêtes de la librairie
- Elles permettent des actions pendant l'exécution ou donnent des informations.

```
#include <mpi.h>
```

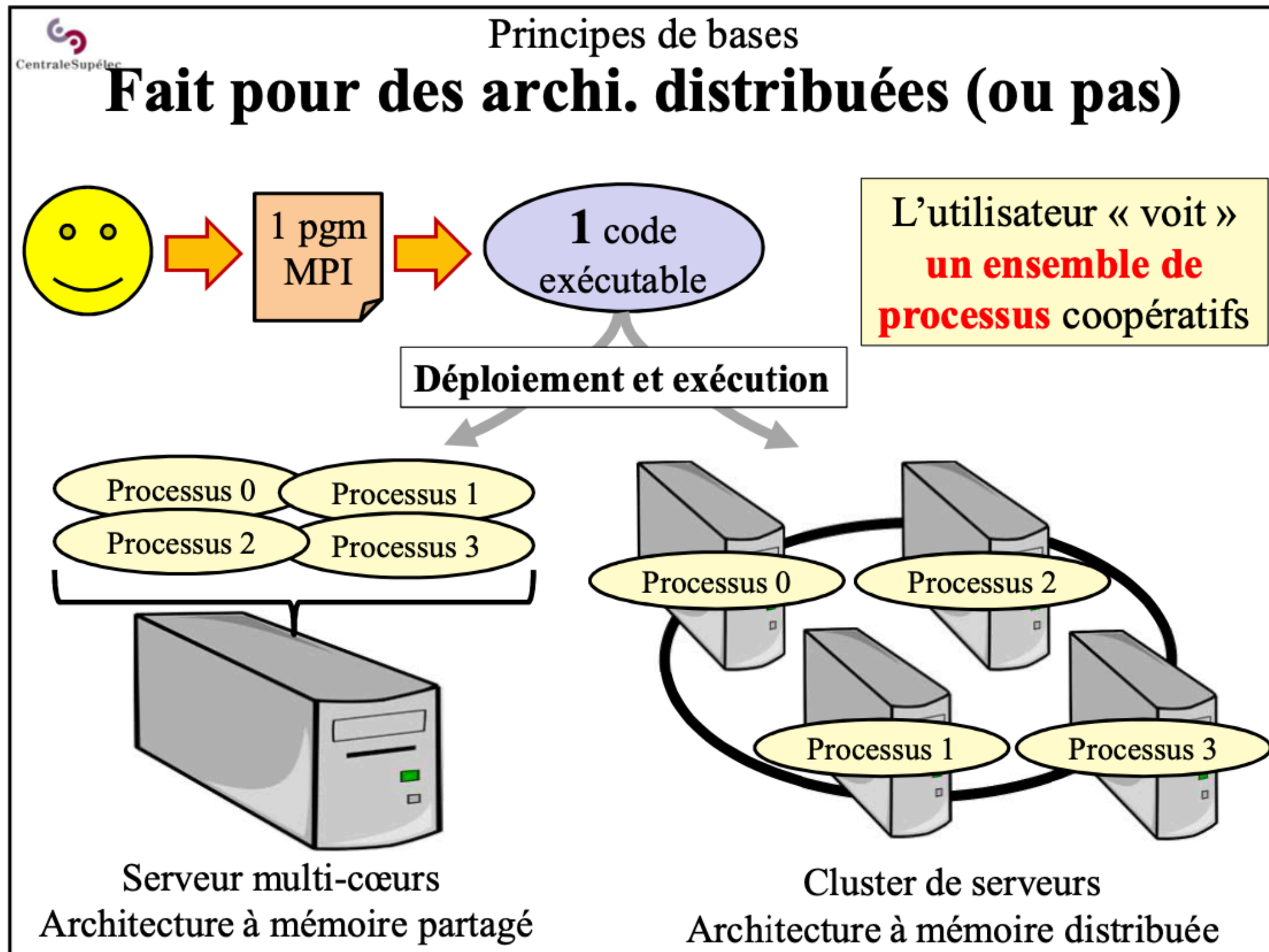
```
MPI_Init(...);  
MPI_Comm_size(...);  
MPI_Comm_rank(...);  
MPI_Send(...);  
MPI_Recv(...);  
MPI_Finalize(...);
```

3. Programme pour l'exécution parallèle

- On utilise un programme pour lancer le programme.
- Options pour indiquer des informations sur l'exécution parallèle
(*ex. nombre de processus parallèles*)

```
>> mpirun -np 8 ./a.out
```


Principes de bases de MPI [1/6]



Principes de bases de MPI [2/6]



Principes de bases « Hello World »

Programme source :

```
#include <stdio.h>
#include "mpi.h"
main(int argc, char **argv) {
    int Me, NbPE;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &NbPE);
    MPI_Comm_rank(MPI_COMM_WORLD, &Me);
    printf("Hello World from process %d/%d\n", Me, NbPE);
    fflush(stdout);
    MPI_Barrier(MPI_COMM_WORLD);
    MPI_Finalize();
}
```

Le « groupe » de tous les
process du pgm MPI

Pour afficher tous les
msgs avant la fin du pgm

Exemple d'exécution avec 3 processus :

Hello World from process 0/3
Hello World from process 2/3
Hello World from process 1/3

Pas d'hypothèse sur l'ordre
d'exécution ni d'affichage !

Principes de bases de MPI [3/6]



Principes de bases

Principales instructions d'un code MPI

Instructions **incontournables** d'un programme MPI :

Inclusion du fichier d'entête MPI

```
#include "mpi.h"
```

Première instruction MPI du `main(int argc, char **argv)`

```
MPI_Init(&argc, &argv);
```

Pour connaître le nombre de processus lancés

```
MPI_Comm_size(MPI_COMM_WORLD, &NbPE);
```

Pour connaître le numéro du processus

```
MPI_Comm_rank(MPI_COMM_WORLD, &Me);
```

Dernière instruction MPI du `main`

```
MPI_Finalize();
```

Instructions de **communication** entre processus :

Comms. point-à-point

Ex : ...

```
MPI_Send(...);
```

```
MPI_Recv(...);
```

Comms. de groupe

Ex: ...

```
MPI_Bcast(...);
```

...

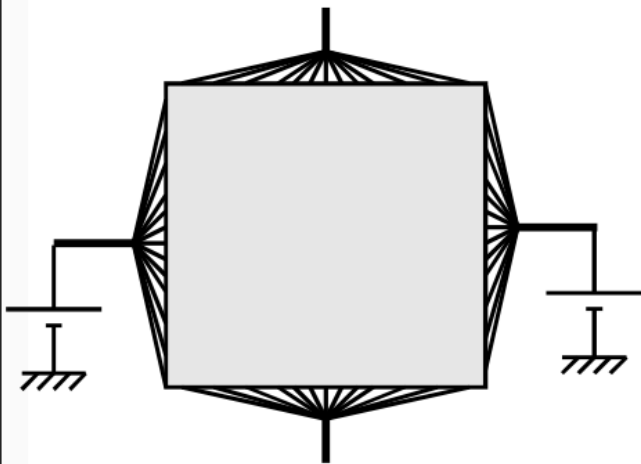
Le parallélisme
est très explicite
en MPI !

Principes de bases de MPI [4/6]



Exemple de programmation MPI

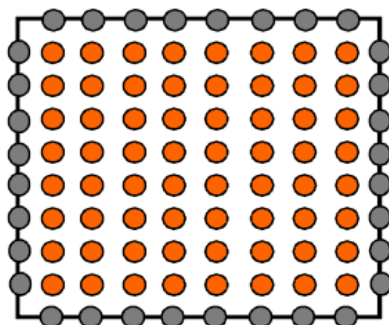
Relaxation de Jacobi parallèle



Calcul des lignes de potentiel dans une plaque diélectrique :

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = 0$$

- Discrétisation et équation aux différences
- Itération jusqu'à la convergence ($V_{i,j}^{n+1} - V_{i,j}^n < \epsilon$)



- Condition aux limites : V fixé

- $$V_{i,j}^{n+1} = \frac{V_{i-1,j}^n + V_{i+1,j}^n + V_{i,j-1}^n + V_{i,j+1}^n}{4}$$

Principes de bases de MPI [5/6]

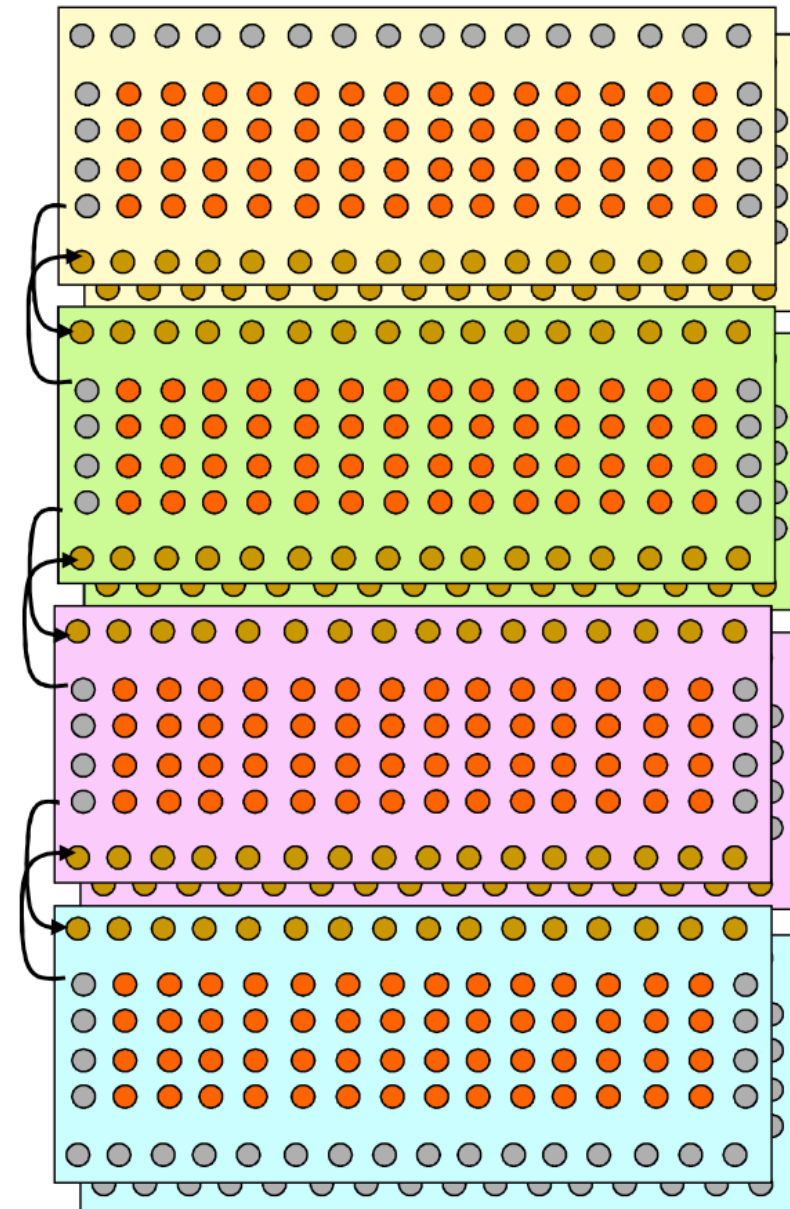


Exemple de programmation MPI Relaxation de Jacobi parallèle

Parallélisation par envoi de messages:

- **Partitionnement spatial des données**
- Ex : avec 4 *processus* de calcul

- 2 extraits des tables V^n et V^{n+1} dans chaque mémoire locale + les frontières
- **Parallélisation de la boucle principale**
- 1 barrière
(implicite sur comm bloquantes)



Principes de bases de MPI [6/6]

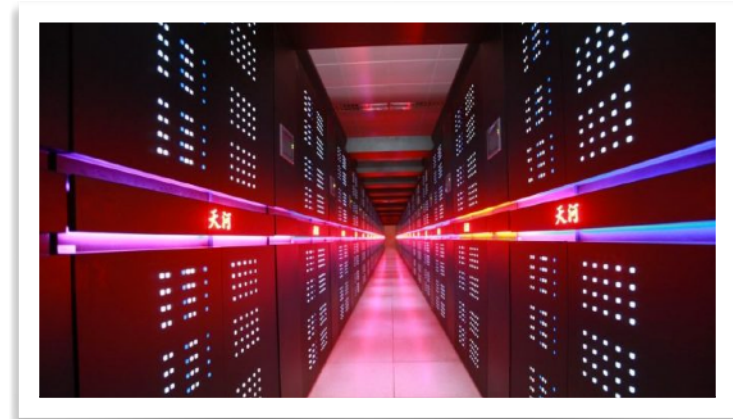
CentraleSupélec

Exemple de programmation MPI Implantation en Send-Recv

```
OI = 0; NI = 1;
for (c = 0; c < NbCycle; c++) {
    // - Computation loop
    .....
    // - Frontier exchange
    if (NbPE > 1) {
        if (Me == 0) {
            MPI_Send(&V[NI][small][1], ..., Me+1, ...);
            MPI_Recv(&V[NI][small+1][1], ..., Me+1, ...);
        } else if (Me == NbPE - 1) {
            MPI_Send(&V[NI][1][1], ..., Me-1, ...);
            MPI_Recv(&V[NI][0][1], ..., Me-1, ...);
        } else {
            MPI_Send(&V[NI][small][1], ..., Me+1, ...);
            MPI_Send(&V[NI][1][1], ..., Me-1, ...);
            MPI_Recv(&V[NI][small+1][1], ..., Me+1, ...);
            MPI_Recv(&V[NI][0][1], ..., Me-1, ...);
        }
    }
    // - Index switching
    OI = 1-OI; NI = 1-NI;
}
```

Diagram illustrating the MPI Send-Recv implementation for a 2D grid. The grid is divided into four quadrants: top-left (yellow), top-right (green), bottom-left (purple), and bottom-right (cyan). The top-left quadrant is labeled 'side+2' with a red double-headed arrow. The bottom-right quadrant is labeled '1' with a red double-headed arrow. The bottom-left quadrant is labeled 'small' and 'small+1' with red text. A blue arrow indicates data flow from the top-left quadrant to the bottom-right quadrant. A red arrow indicates data flow from the bottom-left quadrant to the top-right quadrant.

Calcul parallèle distribué — Quelques messages clés



Pourquoi c'est important ?

- Pour résoudre des problèmes avec beaucoup de calculs ... **seul moyen de passer à l'échelle.**
- Pour résoudre des problèmes avec beaucoup de données ... **seul moyen de passer à l'échelle.**

Les difficultés :

- **Programmation (un peu) plus compliquée** : Programmation explicite des communications.
- **Algorithmes parallèles (un peu) plus compliqués** : Pour être efficaces, les algorithmes parallèles doivent être conçus en tenant compte de l'architecture du cluster.

Comment démarrer ?

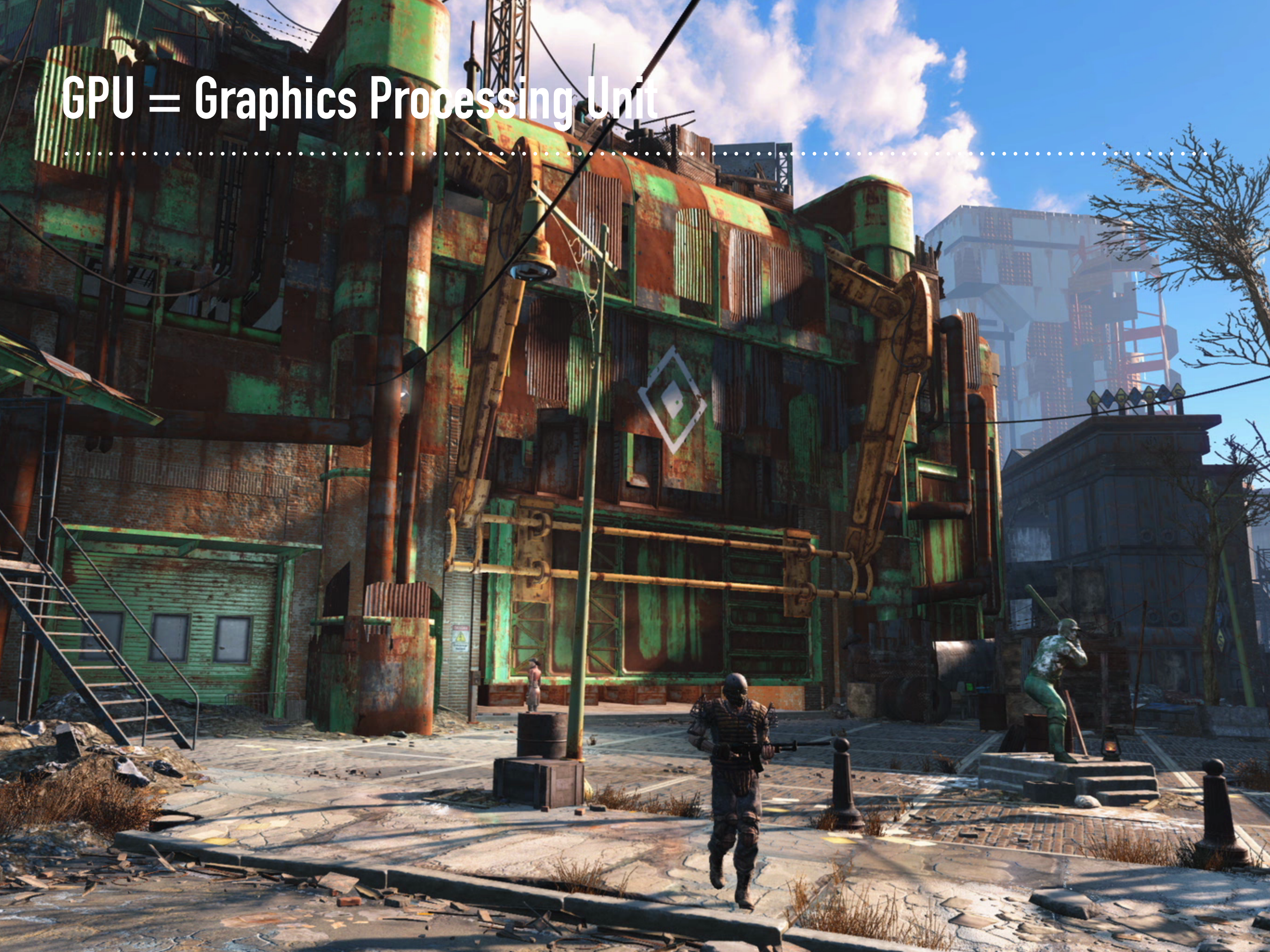
- PRACE Training Portal : <https://training.prace-ri.eu/>
- Livre *"An Introduction to Parallel Programming"* de P. Pacheco
- Dans les parcours IP Paris : M2 AMS (cours AMS301) ou M2 HPDA (cours AMS301)

Introduction au calcul parallèle à mémoire distribuée

Introduction au calcul sur GPU

Contexte HPC français et européen

GPU = Graphics Processing Unit



GPU = Graphics Processing Unit

The main purpose of graphics processing units is to project textured polygons onto the screen in a fiercely competitive consumer-facing industry.



This is an embarrassingly parallel process and specialized MPP chips have been created by ATi (now AMD), Intel, NVIDIA et al to perform floating point intensive operations to render scenes in realtime.

GPGPU — Quelques chiffres ...

Trend for high-performance scientific computation:
clusters of many-core CPU with massively parallel accelerators

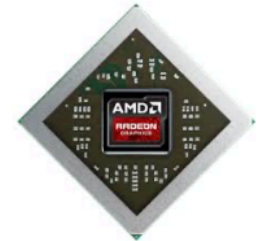
Laptop



CPU
Intel Core i7-3720QM
83.2 to 115 GFLOP/s
16 GB - 4 cores

Accelerators (e.g. GPU)

AMD Radeon R9 M370X
992 GFLOP/s
2 GB - 640 cores



Desktop



Intel Xeon Proc. E5-2680
172.8 to 224 GFLOP/s
32 GB - 8 cores

Nvidia GeForce GTX 980
4.616 TFLOP/s
4 GB - 2496 cores

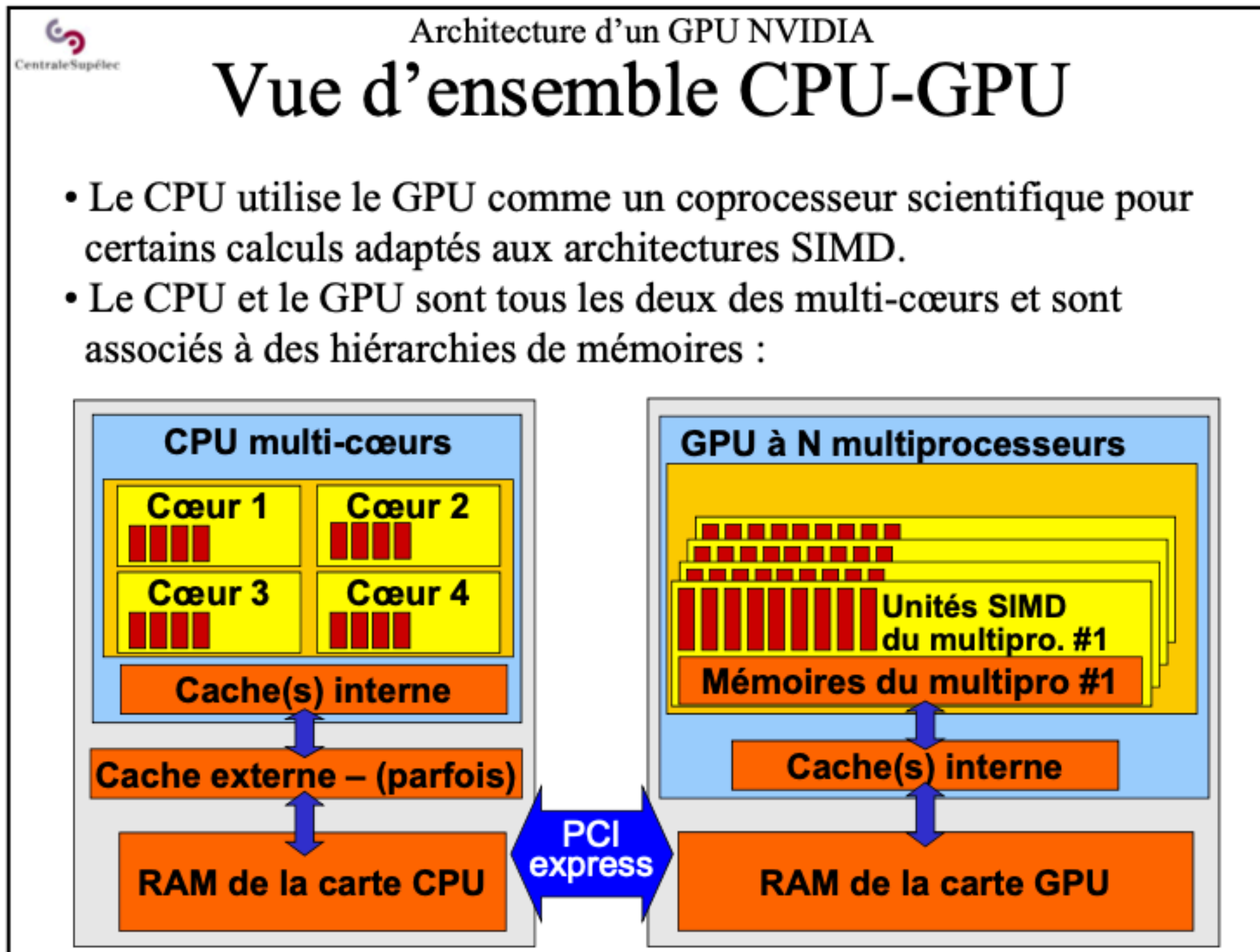


Intel Xeon Phi Coproc. 7120P
1.2 to 2.4 TFLOP/s
16 GB - 61 cores



Accelerators provide approx. 10 - 100 times more FLOP/s
... but on-chip parallelism and smaller memory
... and then require adapted algorithms and implementations

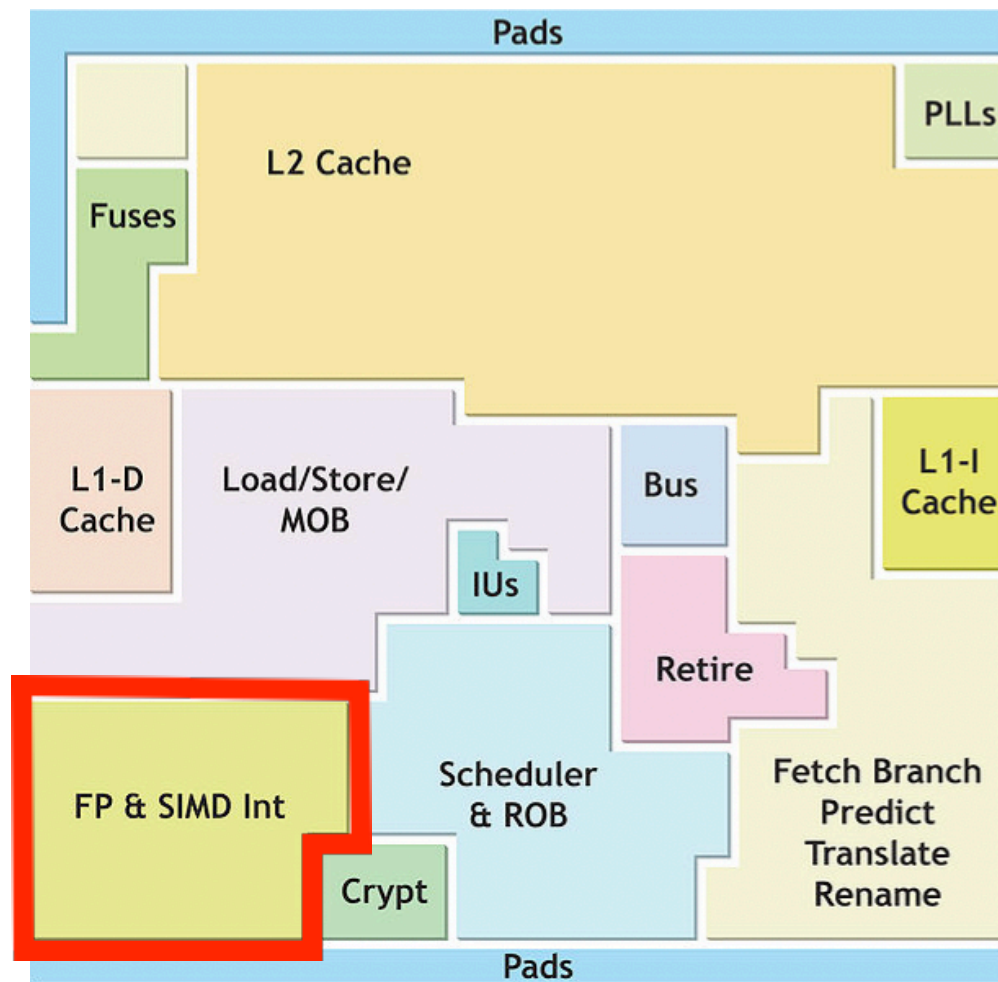
GPGPU — Le GPU est à côté du CPU



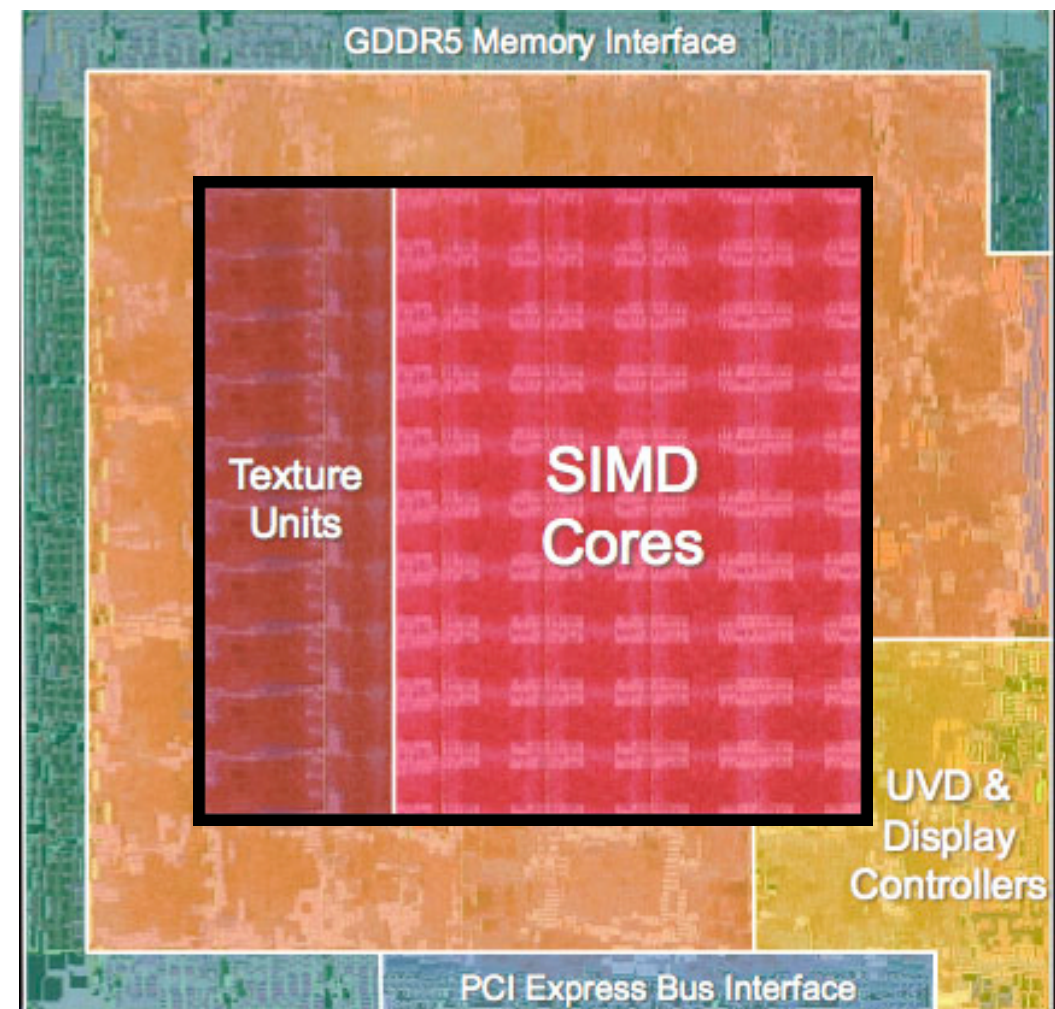
GPGPU — Le GPU est différent du CPU

Comparison of block diagram of vintage CPU and GPU

CPU

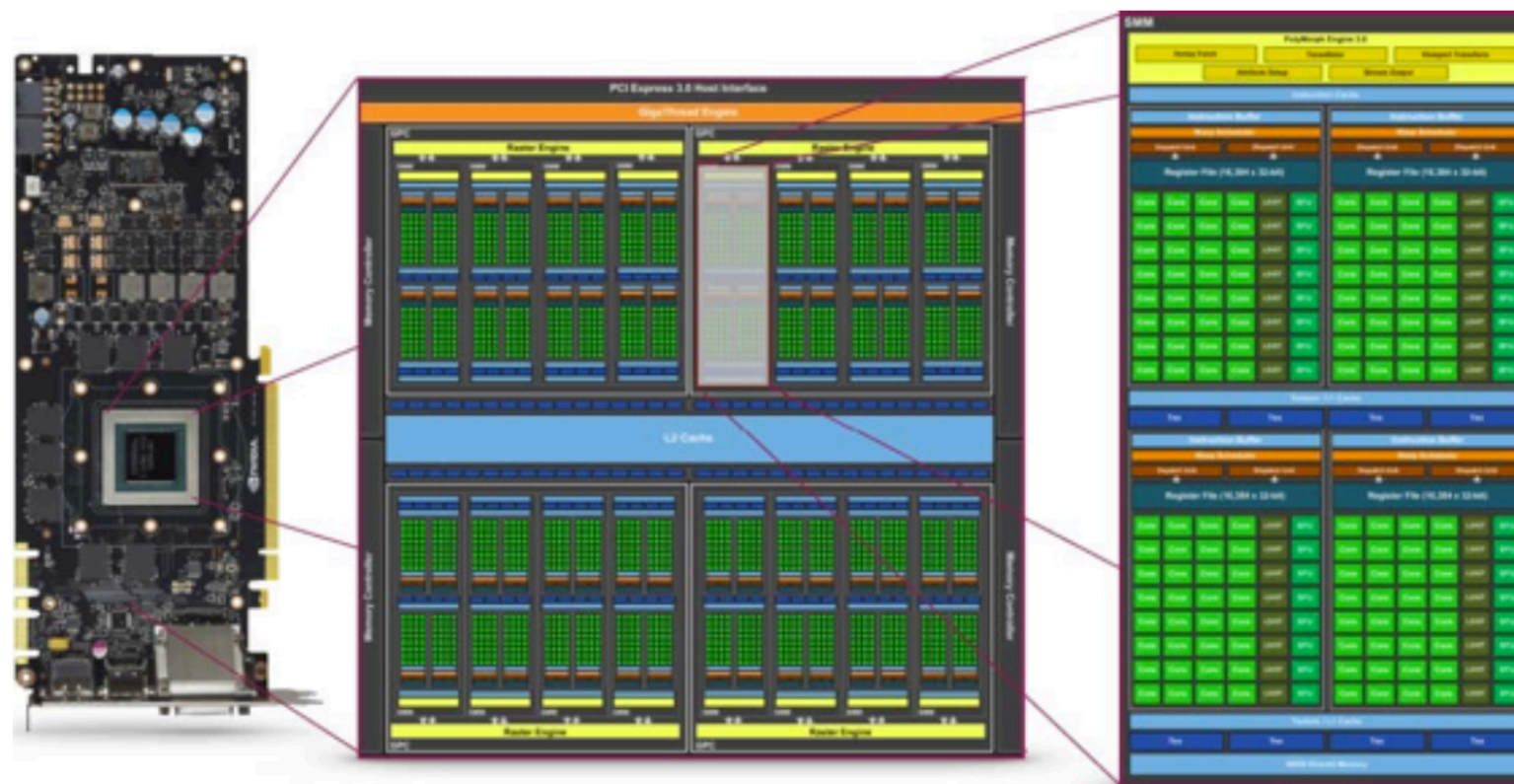


GPU



GPGPU — Une architecture hiérarchique

Hierarchies in the execution model and the memory management
Every implementation must be conceived taking into account these hierarchies



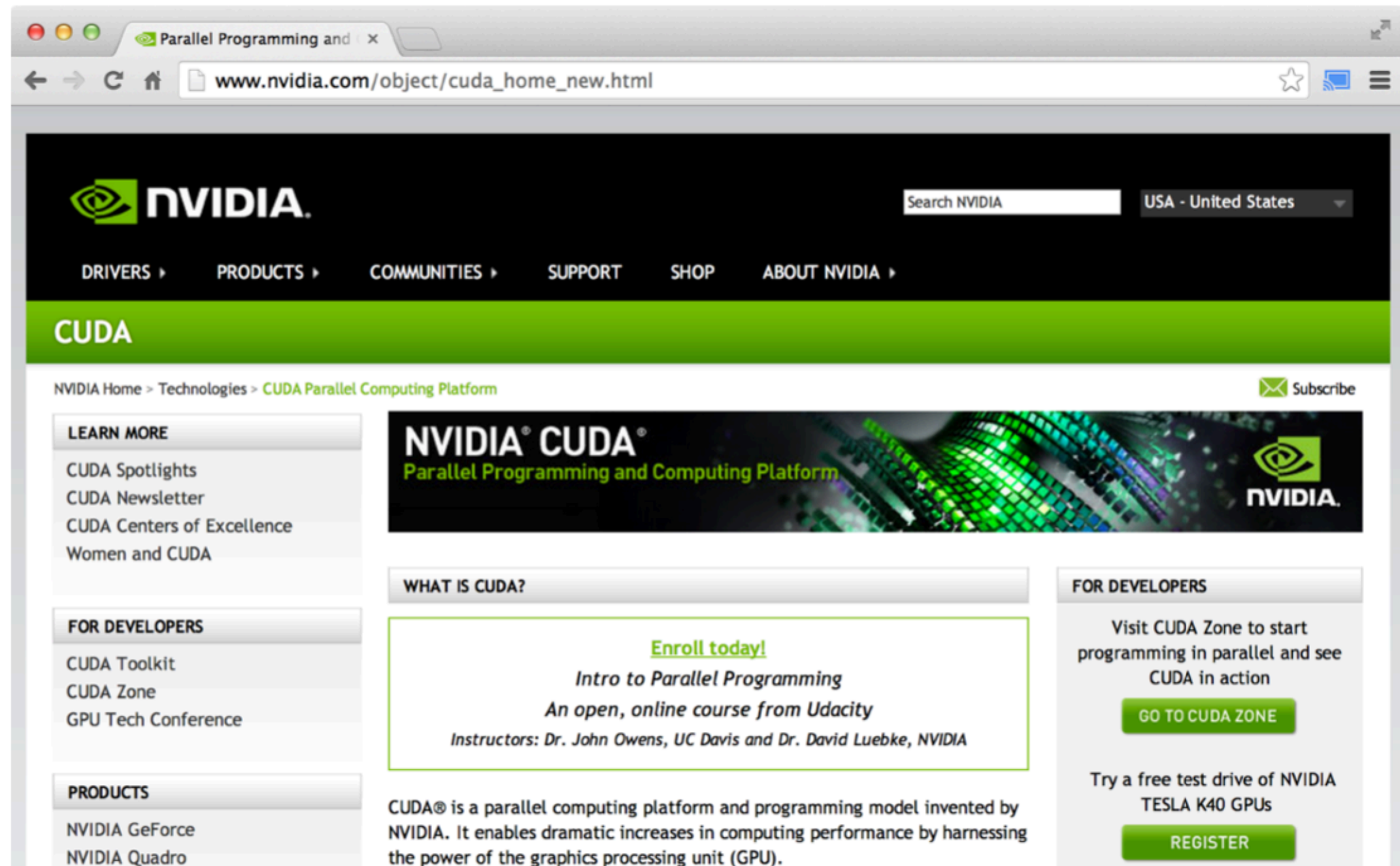
Hardware architecture for Nvidia GTX980

Hierarchies:

- ▶ **Hardware:** device — streaming multiprocessor [16 per device] — core [128 per SM]
- ▶ **Memory:** global mem. [4GB] — shared mem. [96KB per SM] — register [64K per SM]
- ▶ **Programming:** kernel — thread block — thread

GPGPU — Différentes bibliothèques/modèles de programmation

Un exemple : Bibliothèque CUDA pour les GPU NVIDIA



GPGPU — Différentes bibliothèques/modèles de programmation

Exemple de code CUDA

Overview of C-like CUDA code that runs on the HOST:

```
#include "cuda.h"

int main(int argc, char **argv){
    int N = 3789; // size of array for this DEMO

    float *d_a; // Allocate DEVICE array
    cudaMalloc((void**) &d_a, N*sizeof(float));

    int B = 512;
    dim3 dimBlock(B,1,1); // 512 threads per thread-block
    dim3 dimGrid((N+B-1)/B, 1, 1); // Enough thread-blocks to cover N

    // Queue kernel on DEVICE
    simpleKernel <<< dimGrid, dimBlock >>> (N, d_a);

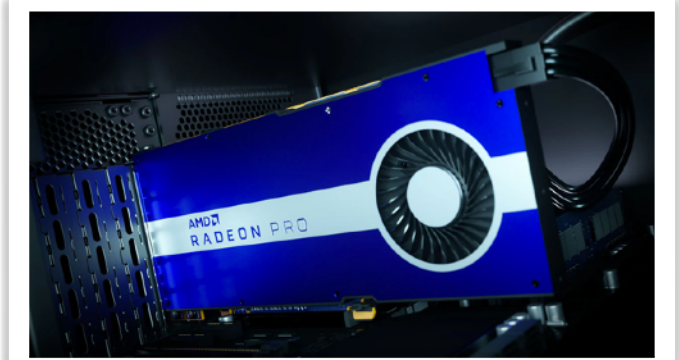
    // HOST array
    float *h_a = (float*) calloc(N, sizeof(float));

    // Transfer result from DEVICE array to HOST array
    cudaMemcpy(h_a, d_a, N*sizeof(float), cudaMemcpyDeviceToHost);

    // Print out result from HOST array
    for(int n=0; n<N; ++n) printf("h_a[%d] = %f\n", n, h_a[n]);
}
```

simpleKernel.cu

General-Purpose Computing on GPU — Messages clés



Pourquoi c'est important ?

- **Coût énergétique** : Un GPU est plus efficace au niveau énergétique (en watt) qu'un CPU.
- **Coût économique** : Le GFLOP/s est moins cher (en \$) avec un GPU qu'avec un CPU.

Les difficultés :

- **Programmation vraiment compliquée** : Plusieurs niveaux de mémoires et de parallélisation.
- **Algorithmes parallèles vraiment compliqués** : Pour être efficaces, les algorithmes parallèles doivent être conçus en tenant compte de la hiérarchie matérielle du GPU.
- ***La limitation des tailles de mémoire et des bandes passantes est le principal problème !***

Comment démarrer ?

- Documentation **CUDA** (NVIDIA) : <https://developer.nvidia.com/cuda-toolkit>
- Documentation **OpenACC** : <https://www.openacc.org/>
- Dans les parcours IP Paris : M2 AMS (cours AMS-I03) ou M2 HPDA (cours CSC5001)

Introduction au calcul parallèle à mémoire distribuée

Introduction au calcul sur GPU

Contexte HPC français et européen

TOP500.org ... au niveau français

10 PFLOP/s 10¹⁶



Supercalculateur "Jean Zay"



Supercalculateur "Joliot-Curie"



Supercalculateur "Occigen"

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
11	Total Exploration Production	PANGEA III - IBM POWER9 18C 3.45GHz, Dual-rail Mellanox EDR Infiniband, NVIDIA Volta GV100 , IBM	291,024	17,860.0	25,025.8	1,367
17	CEA	Tera-1000-2 - Intel Xeon Phi 7250 68C 1.4GHz , Bull BXI 1.2 — Atos	561,408	11,965.5	23,396.4	3,178
41	Total Exploration Production	Pangea - SGI ICE X, Xeon Xeon E5-2670/ E5-2680v3 12C 2.5GHz, Infiniband FDR, HPE	220,800	5,283.1	6,712.3	4,150
46	CNRS/IDRIS-GENCI	Jean Zay - HPE SGI 8600, Xeon Gold 6248 20C 2.5GHz, NVIDIA Tesla V100 SXM2 , Intel Omni-Path — HPE	93,960	4,478.0	7,345.6	
52	CEA/TGCC-GENCI	JOLIOT-CURIE SKL - Bull Sequana X1000, Xeon Platinum 8168 24C 2.7GHz, Mellanox EDR — Atos	79,488	4,065.6	6,635.5	917
104	CINES-GENCI	occigen2 - bullx DLC 720, Xeon E5-2690v4 14C 2.6GHz, Infiniband FDR — Atos	85,824	2,494.7	3,570.3	1,430
127	Meteo France	Prolix2 - bullx DLC 720, Xeon E5-2698v4 20C 2.2GHz, Infiniband FDR — Atos	72,000	2,168.0	2,534.4	830
163	EDF	Gaïa - Bull intel Cluster, Xeon Gold 6140 18C 2.3GHz, Intel Omni-Path — Atos	41,472	1,980.5	3,052.3	512

Au niveau public français



Supercalculateur "Jean Zay"
IDRIS – GENCI



Supercalculateur "Joliot-Curie"
TGCC – GENCI



Supercalculateur "Occigen"
CINES – GENCI

Organismes nationaux

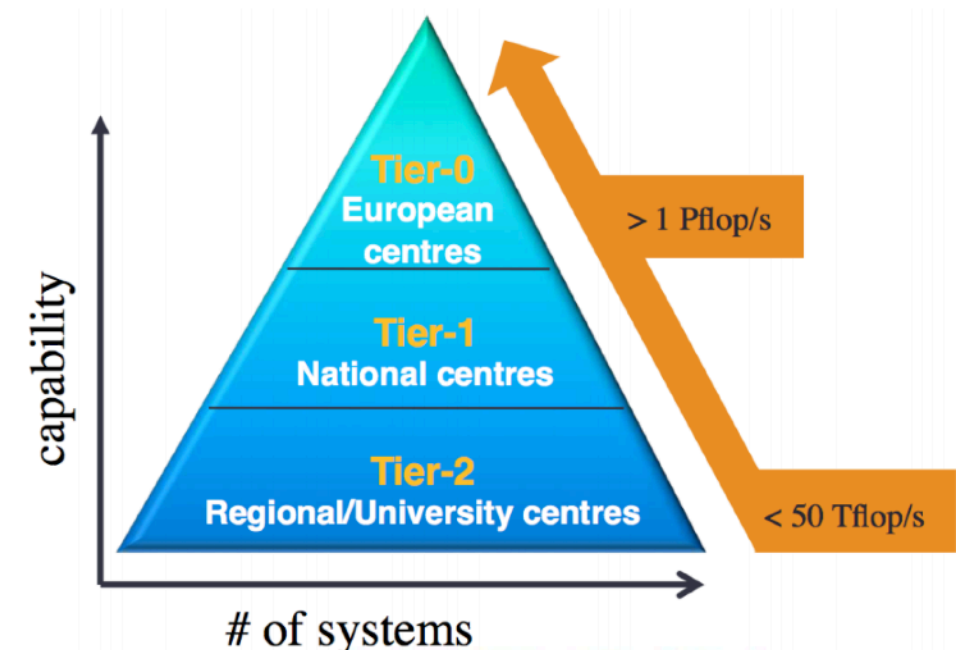
- [GENCI](#), organisme public "Grand équipement national de calcul intensif"
- [IDRIS](#) (Institut du développement et des ressources en informatique scientifique), CNRS
- [TGCC](#) (Très Grand Centre de Calcul), CEA
- [CINES](#) (Centre Informatique National de l'Enseignement Supérieur), Ministère Ens. Sup.

Autres niveaux

- Méso-centres https://calcul.math.cnrs.fr/pages/mesocentres_en_france.html
- Centres de recherches, universités, laboratoires, ...



Cluster GIN de l'ENSTA / UMA
~ 10 TFLOP/s



Au niveau public européen — EuroHPC



EuroHPC
Joint Undertaking

#EuroHPC (high performance computing) Joint Undertaking

The European High Performance Computing Joint Undertaking (EuroHPC JU) will pool European resources to develop top-of-the range exascale supercomputers for processing big data, based on competitive European technology.

Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Montenegro, the Netherlands, North Macedonia, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and Turkey

+ Union Européenne

+ ETP4HPC et BDVA

<https://eurohpc-ju.europa.eu/>



Au niveau public européen — PRACE



« Partnership for Advanced Computing in Europe »

ENABLING HIGH-IMPACT SCIENTIFIC DISCOVERY AND ENGINEERING RESEARCH & DEVELOPMENT

View open PRACE access calls >

TRAINING & SUPPORT
Discover >

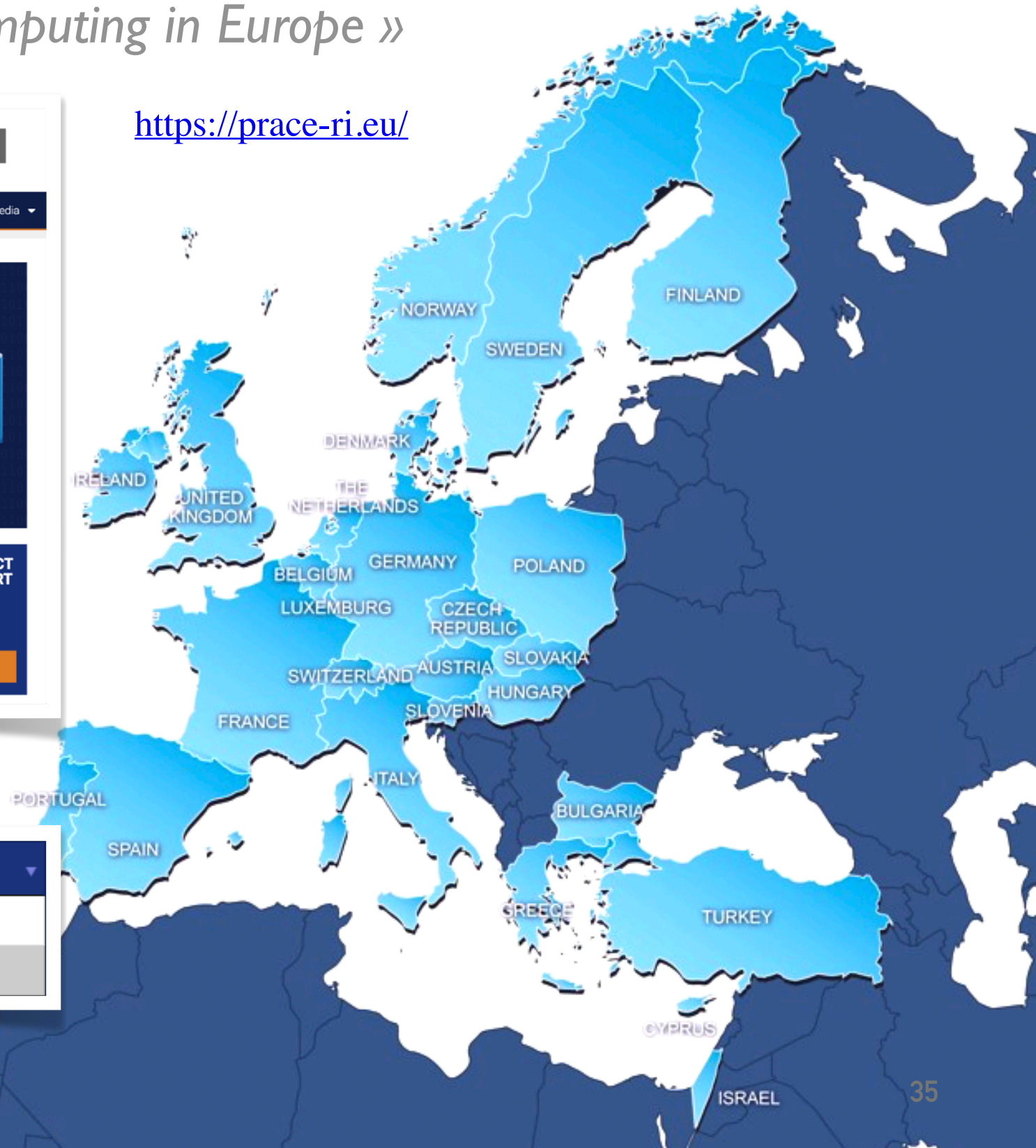
INDUSTRY & SMEs
Discover >

INFRASTRUCTURE SUPPORT
Discover >

<https://prace-ri.eu/>

OPEN CALLS FOR PROPOSALS:

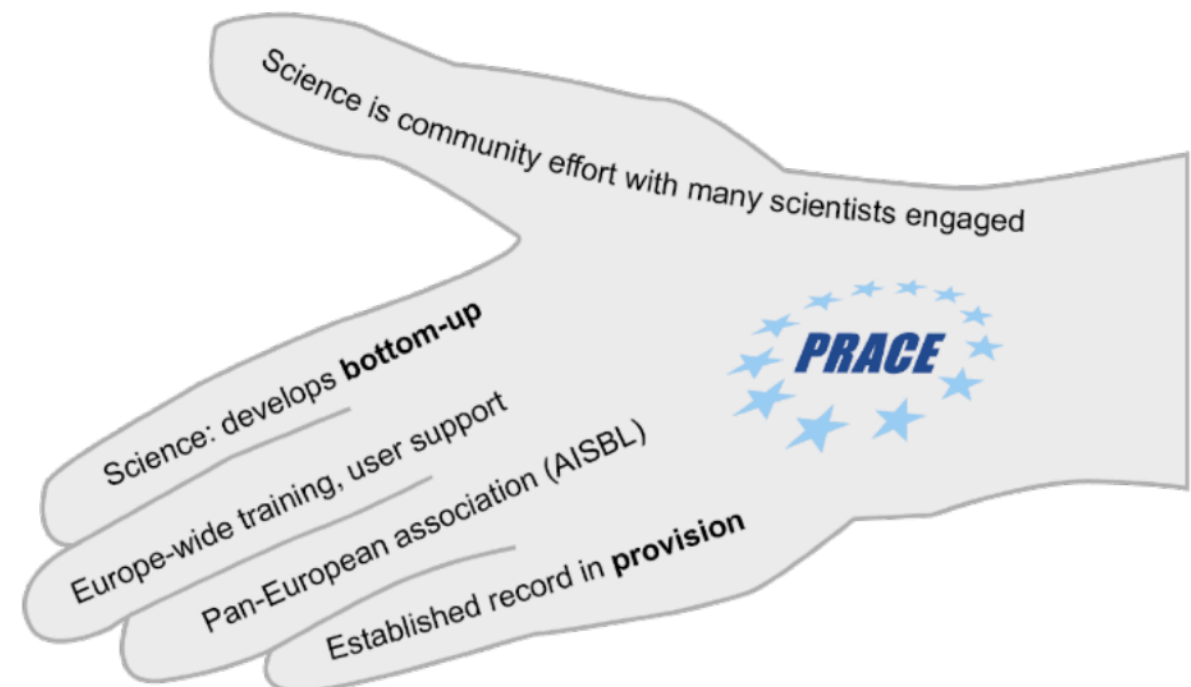
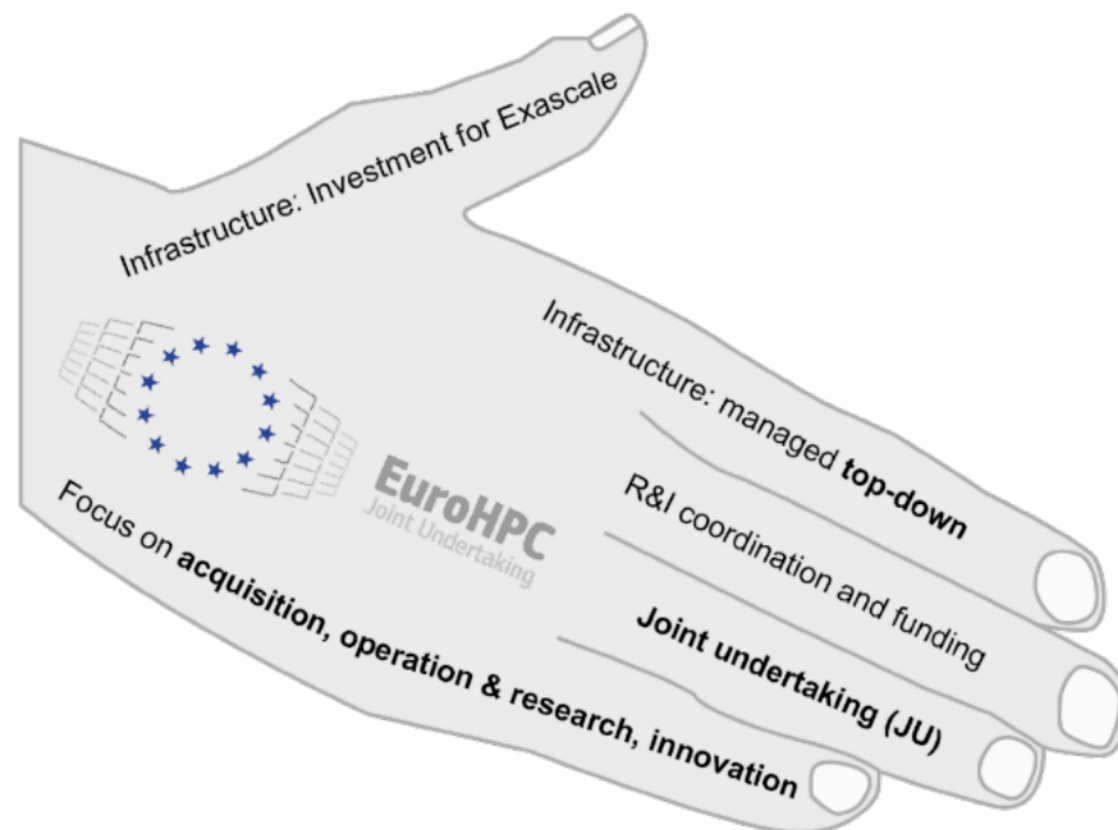
Title	Opening Date	Closing Date Time
SHAPE 11th Call	01/04/2020	01/06/2020 - 10:00
PRACE 21st Call for Proposals for Project Access	03/03/2020	05/05/2020 - 10:00



Au niveau public européen

Machines TIER-0 du PRACE

Country	Site (Partner)	HPC System	RPeak (TFlop/s)
France	CEA (GENCI)	Irene (SkyLake)	6,635.5
France	CEA (GENCI)	Irene (KNL)	2,339.6
Germany	JUELICH (FZJ)	JUWELS	9,891.1
Germany	LRZ (GCS)	SuperMUC phase 2	3,580.0
Germany	HLRS (GCS)	Hazelhen	7,420.0
Italy	CINECA (CINECA)	Marconi	20,000.0
Spain	BSC (BSC)	MareNostrum	11,150.0
Switzerland	CSCS (ETH)	Piz Daint	15,988.0



Bande passante

Région parallèle

Mémoire partagée

FLOP

#prama ivdep

Cœur

Thread

MPI

GFLOP/s

barrier

Caches

Co-processeur

Mémoires distribuées

Vectorisation

Reduce

PCIe

Débit arithmétique

Speedup

Processeur

GPGPU

Scalabilité

Performance

#prama omp

Deadlock

schedule

Nœud

Fork-Join

OpenMP

Concurrence

CUDA

Registres

L1 L2 L2

To be continued ...

