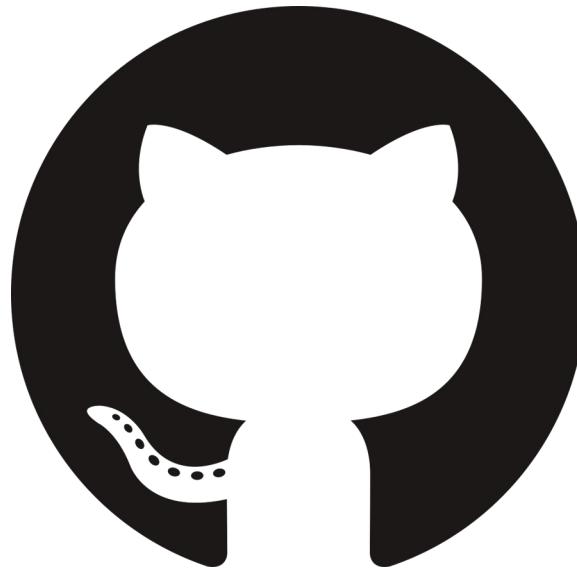


Introduction au Calcul Haute Performance

Master DFE – année 2020/2021

Mathieu Lobet, Maison de la Simulation
Mathieu.lobet@cea.fr

Ce cours est libre



<https://github.com/Maison-de-la-Simulation/HPC-DFE-Paris-Saclay>

Description des pictogrammes au sein des *slides*



Désigne une définition



Désigne une astuce ou une remarque



Désigne une commande de terminal



Désigne un exemple de code



Désigne une *slide* exercice

Prérequis pour ce cours

1) Les bases de la programmation en C/C++

[Cours de l'IDRIS sur le C](#)

[Cours OpenClassroom](#)

1) Les bases de la programmation en Fortran 90

[Cours ENSEIRB-MATMECA](#)

[Cours de L'IDRIS](#)

2) Les bases de l'utilisation d'un terminal sous Linux

[Cours OpenClassroom](#)

3) Savoir compiler et exécuter un programme

Contenu du cours

I. Introduction au calcul parallèle

II. Introduction au parallélisme par échange de message via MPI

IV. Mesure de la performance

V. Travaux pratiques

Introduction au calcul parallèle

1) Architectures parallèles

Calcul Haute Performance (HPC)

High performance computing



Le Calcul Haute Performance est la discipline qui s'intéresse à la programmation et à l'utilisation des **super-calculateurs** dans le but d'exécuter de la manière la plus performante possible des **algorithmes** donnés.



Le HPC joue un rôle essentiel en simulation numérique en permettant l'élaboration de codes de simulation, principalement scientifiques, capables de tourner efficacement sur des très larges super-ordinateurs.

Notion de puissance de calcul



La puissance de calcul d'une unité de calcul ou d'un ordinateur entier se mesure en **opérations par seconde**. Dans le cas d'une opération à virgule flottante on parle plus précisément de **flops/s**. Il faut préciser s'il s'agit de flottants à simple ou double précision. Le plus souvent les chiffres sont donnés pour des doubles.

La puissance d'une machine parallèle est la somme des puissances de chaque unité de calcul.

Il existe un classement mondial des super-calculateur appelé [TOP500](#).

Mesure de la mémoire



Bit 0 1

Bytes = octet => 8 bit (utilisé dans l'embarqué)

Entier demi-précision => 2 octets (utilisé dans l'embarqué)

Entier simple précision => 4 octets (le plus courant)

Entier double précision => 8 octets

Float demi-précision => 2 octets (utilisé en Machine Learning)

Float simple précision => 4 octets

Float double précision => 8 octets

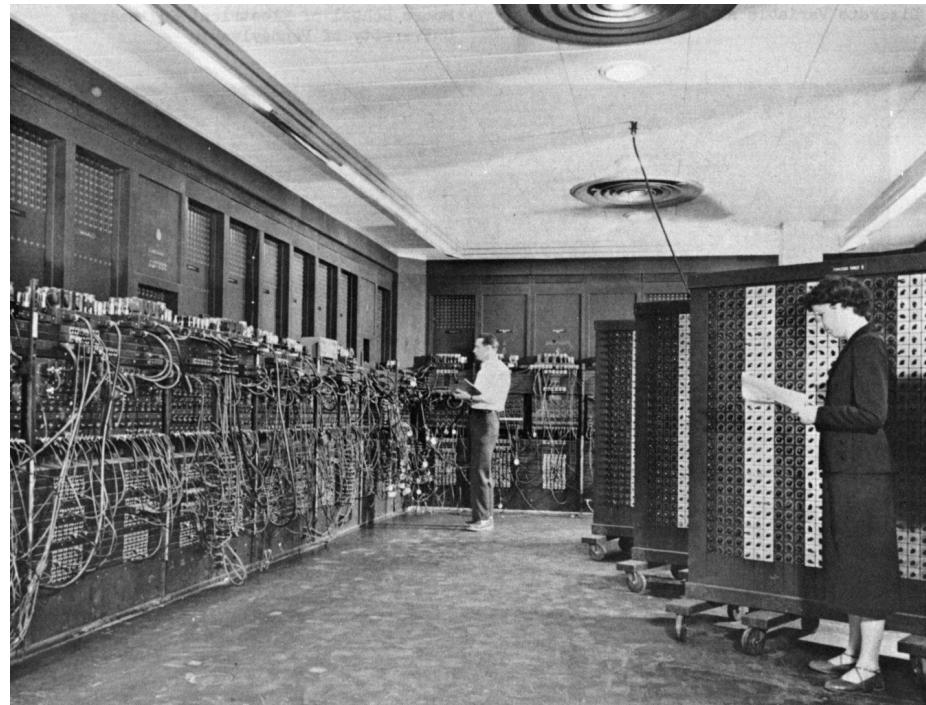
1 Ko = 1024 octets

Les métiers du HPC

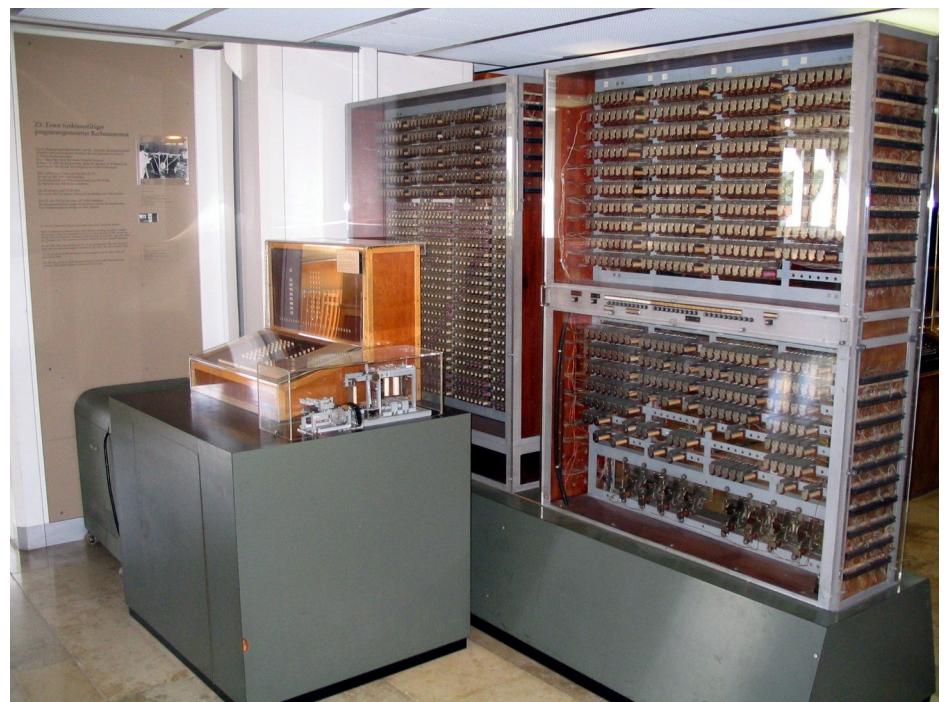
- Postes ingénieur R&D faisant appel à la modélisation numérique : ingénieur calcul, CFD, matériaux, chimie...
- Postes de recherche faisant appel à la modélisation numérique : CFD, astrophysique, fusion, chimie, biologie...
- Poste ingénieur prédition/statistique : finance, assurance...
- Ingénieur en traitement des données (*big data*)
- Ingénieur en intelligence artificielle (*machine learning, deep learning*)
- Ingénieurs rendu physique pour l'animation
- Optimisation dans le jeu vidéo

Un peu d'histoire : 1940-1960, les prémisses

L'histoire des ordinateurs démarrent doucement un peu avant la seconde guerre mondiale et voit ses premiers systèmes appliqués (électro-mécanique ou électronique) pendant la guerre.



ENIAC (1945), l'un des premiers ordinateurs électroniques – 50 Kflops, 150 kW - USA



Z3 (1941), ordinateur électromécanique allemand – 20 flops, 4 kW - Allemagne

1960 – 1970 : la naissance d'une industrie

L'émergence des super-ordinateurs destinés au calcul scientifique au sens propre se fait au début des années 60. C'est aussi la naissance du Fortran.



CDC 6600 (1964), l'un des premiers vrais succès en terme de HPC – 0,5 Mflops, 30 kW - USA



ILLIAC IV (1966), premier ordinateur conçu pour être parallèle, utilise le concept de vectorisation, premier à être connecté à l'ancêtre d'internet – 50 Mflops - USA

1970-1990 : l'ère des machines vectorielles et de Cray

Cray, l'un des concepteurs du CDC, va fonder sa société et devenir l'un des leaders du marché pendant 15 ans



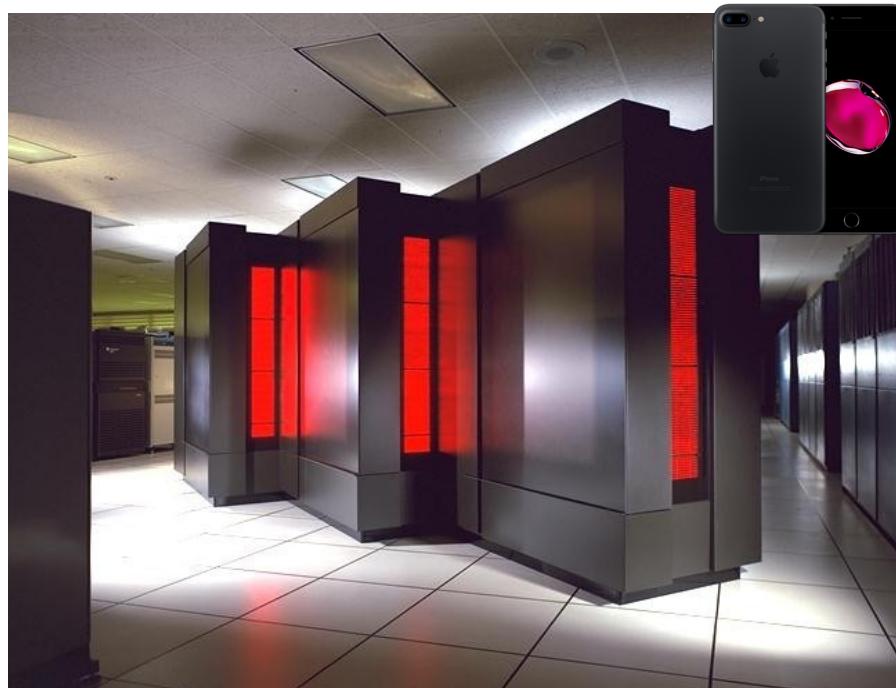
Cray 1 (1975), vectorisation très efficace, puissance équivalente à l'iphone 1, à la fois esthétique et performant – 160 Mflops, 115 kW – USA



Cray Y-MP (1988), 2 à 8 processeurs vectoriels à mémoire partagée, 333 Mflops par processeur, puissance équivalente à l'iphone 5 – 2.6 Gflops - USA

1990 -aujourd'hui : le parallélisme massif distribué et la convergence avec l'industrie de la micro-informatique

D'un point de vue économique et technique, il devient de plus intéressant de mettre en réseau des processeurs aux technologies issues du grand public.



CM 5 (1993), 1024 processeurs, premier au top 500 en 1993, puissance équivalente à l'iphone 7 – 131 Gflops - USA



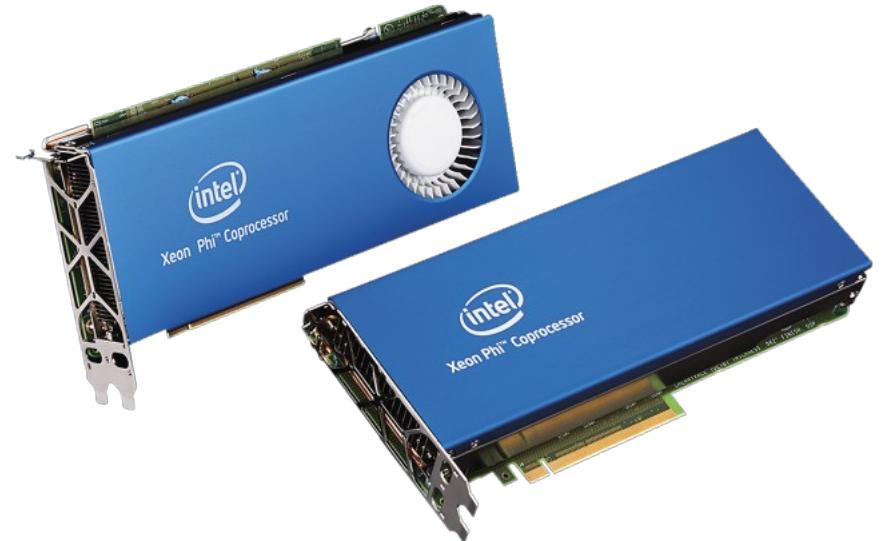
Un exemple surprenant est la construction de véritables cluster de PS3 dans les années 2007-2010 grâce au processeur CELL d'IBM

2010-aujourd'hui : le retour en grâce de la vectorisation et du multi-coeur

Les processeurs deviennent multi-coeur et les registres vectoriels grossissent comme c'était le cas sur les machines Cray mais condensé dans une simple puce



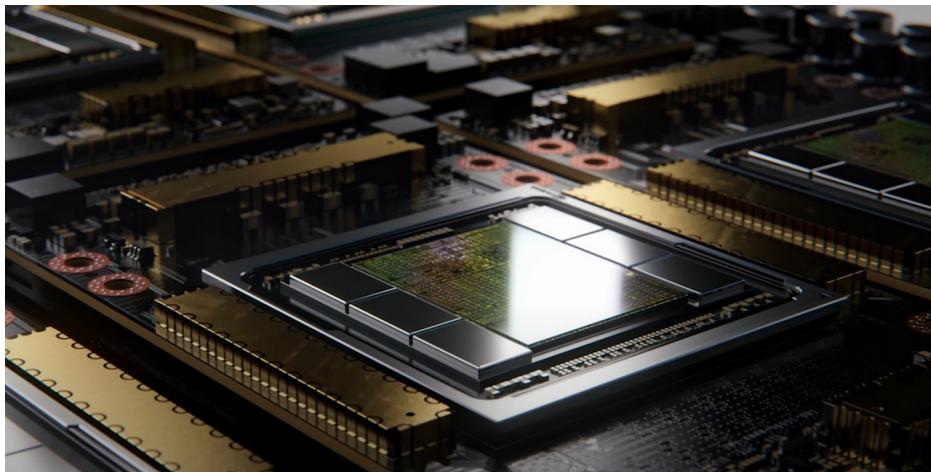
Le processeur Intel Broadwell (2014) est l'un des succès majeurs d'Intel sur le marché du HPC (jusqu'à 22 coeurs, 44 threads en dual socket, registre vecto de 4 doubles)



Le Xeon Phi est une tentative de rupture d'Intel avec un processsor dit many-core équipé de 68 coeurs à longs registres vectoriels de 8 doubles. Le succès en terme de performance restera mitigé.

2010 – aujourd’hui : l’émergence de la technologie GPU et des accélérateurs

La technologie GPU offre des ratios performance, prix, consommation électrique supérieur à la techno CPU pour des applications données. Beaucoup de calculateurs parmi les plus puissants au monde utilisent cette technologie.

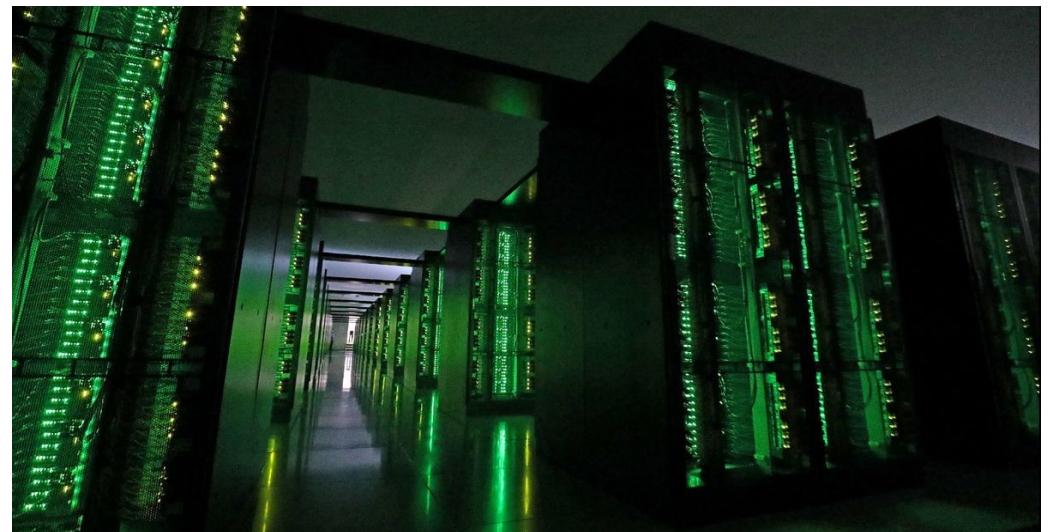
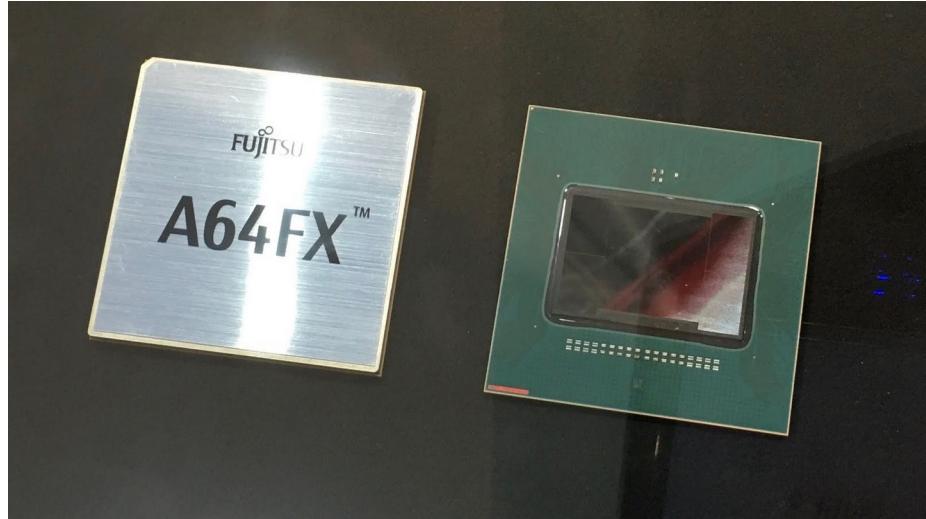


NVIDIA est le leader du marché HPC avec l’architecture Ampere. Un seul GPU offre une puissance de 10 Tflops en double précision.

Summit est le super-calculateur américain le plus puissant équipé de 28 000 cartes NVIDIA Volta 100 pour une puissance totale de 150 Pflops.

2020 : la course à l'Exascale et l'émergence de nouveaux acteurs

Dans une philosophie d'indépendance et de compétition exacerbée entre les grandes puissances du HPC(USA, Chine, Europe, Japon), la technologie RISC/ARM a refait son retour dans la course à la puissance.



La technologie ARM bien connue de nos mobiles investie aujourd'hui le secteur du HPC et du cloud computing avec des arguments de coût, puissance et consommation énergétique.

Le FUGAKU, le super-ordinateur le plus puissant du monde (416 Pflops), est équipé de cette technologie développée ici par Fujitsu. Le processeur ARM le plus puissant est le Fujitsu A64FX.

Machine parallèle

Parallel machine



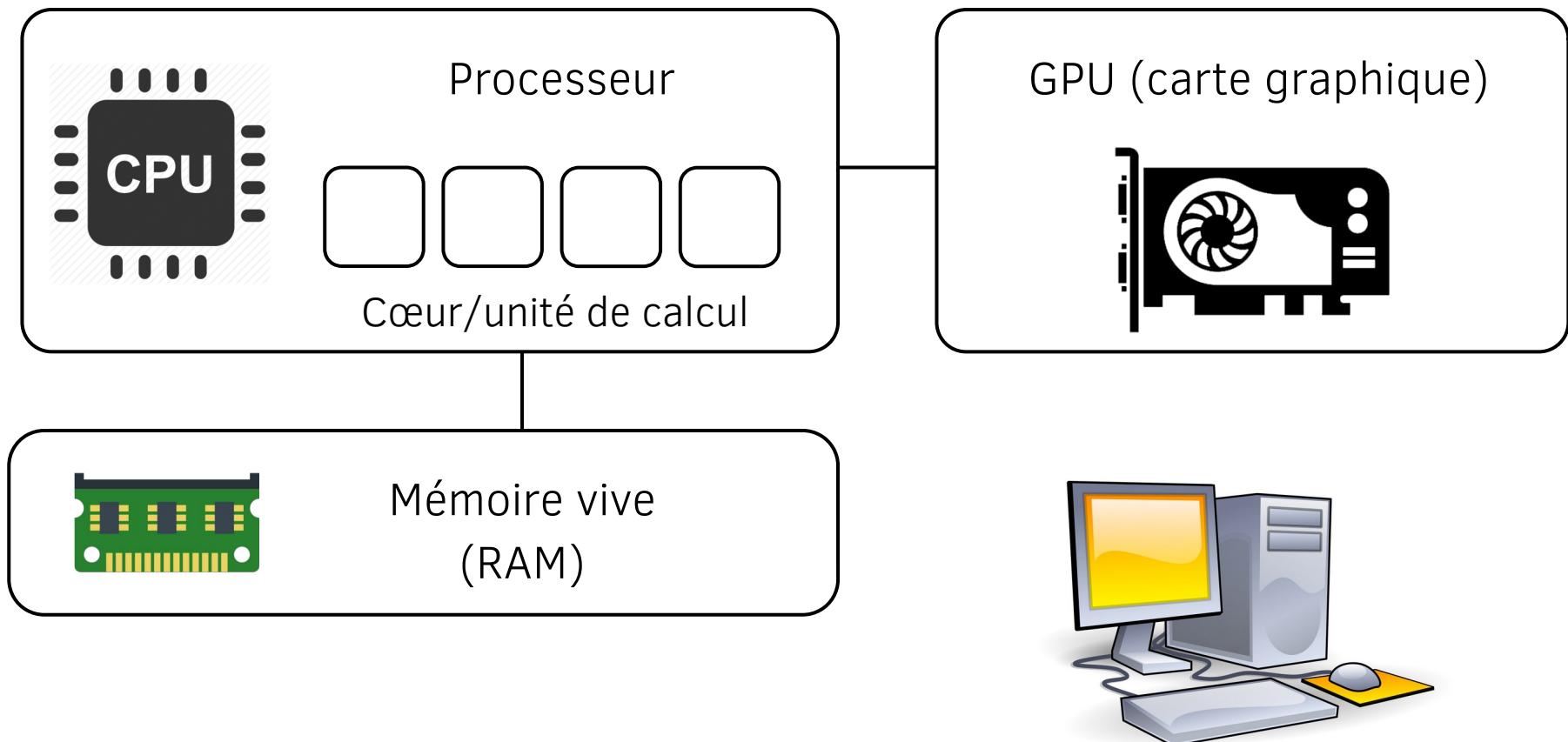
Une machine parallèle est aujourd’hui tout simplement une machine composée de plusieurs unités de calcul.



Les processeurs d’ordinateur, de téléphones portables ou les cartes graphiques sont tous aujourd’hui des composants parallèles car composés de plusieurs cœurs de calcul.

Ordinateur de bureau

Un simple ordinateur portable ou de bureau est déjà une machine parallèle.

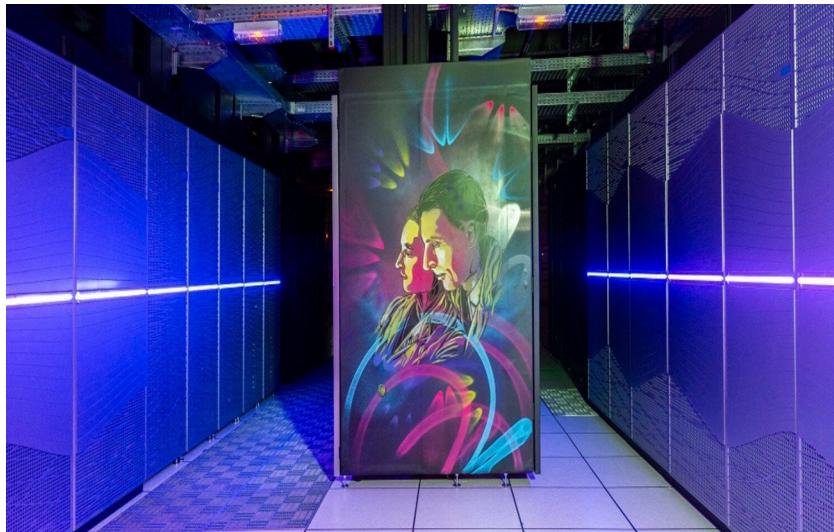


Super-ordinateur ou super-calculateur

Super-computer



Un super-ordinateur (ou super-calculateur) est un système complexe de calcul surpassant les capacités d'un simple ordinateur, résultant de la mise en parallèle d'un très grand nombre d'unité de calcul. Il peut être dédié à l'exécution parallèle de programmes informatiques et/ou l'exécution simultanée de plusieurs programmes.



JOLIOT-CURIE ROME (TGCC, FRANCE). 33ème avec une puissance de calcul de 7 petaflops/s au TOP500. 200 000 coeurs AMD ROME.



Jean-Zay est le puissant super-calculateur du CNRS. Il est équipé de plusieurs partitions hybrides CPU-GPU (NVIDIA V100) pour une puissance globale de 16 Pflops.

Pourquoi des super-calculateurs ?

- Résoudre des **problèmes inaccessible** avec la puissance de calcul ou la mémoire d'un **simple ordinateur**

En science :

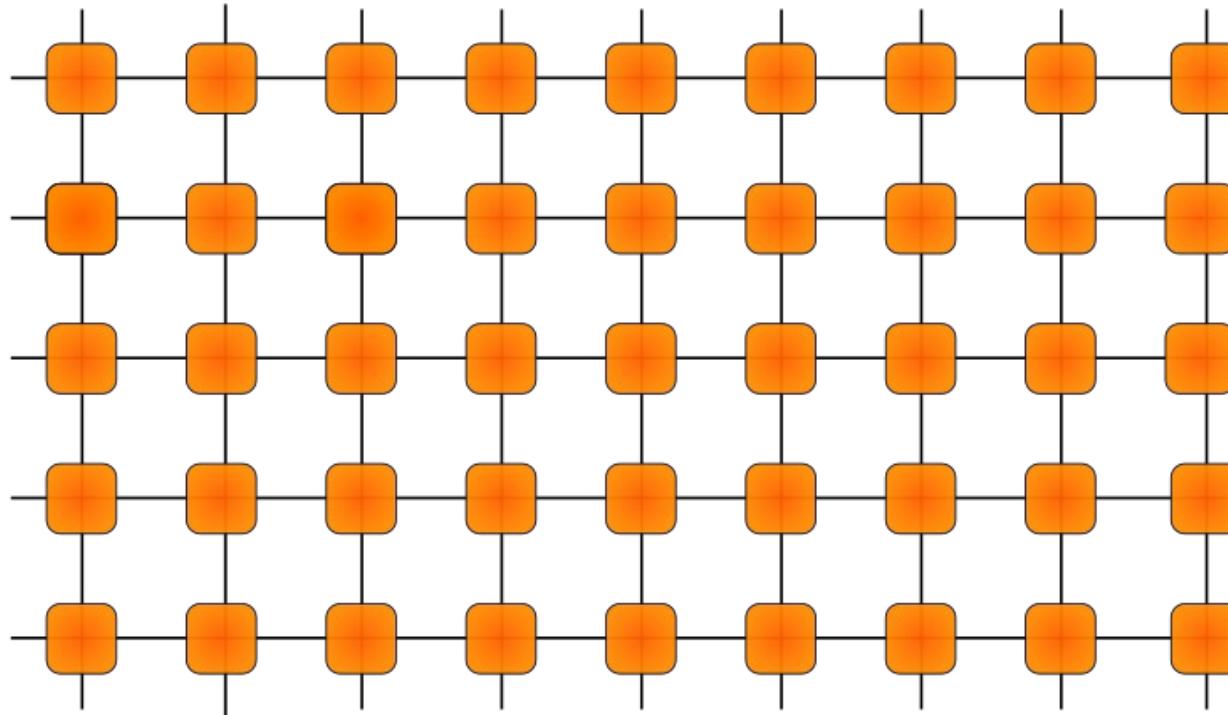
- Résoudre des problèmes avec beaucoup d'éléments ou sur des temps très longs
- Effectuer un grand nombre de simulations (étude paramétrique)
- Résoudre des problèmes multi-D complexe
- Résoudre des problèmes multi-physiques
- Traiter des données issues d'expériences (accélérateurs, satellites...)



Les choix technologiques et l'architecture des super-calculateurs dépendent principalement de leur utilisation finale (scientifique, traitement des données, IA, temps réel)

Architecture des super-calculateurs – principe d'une ferme de calcul

Computer cluster

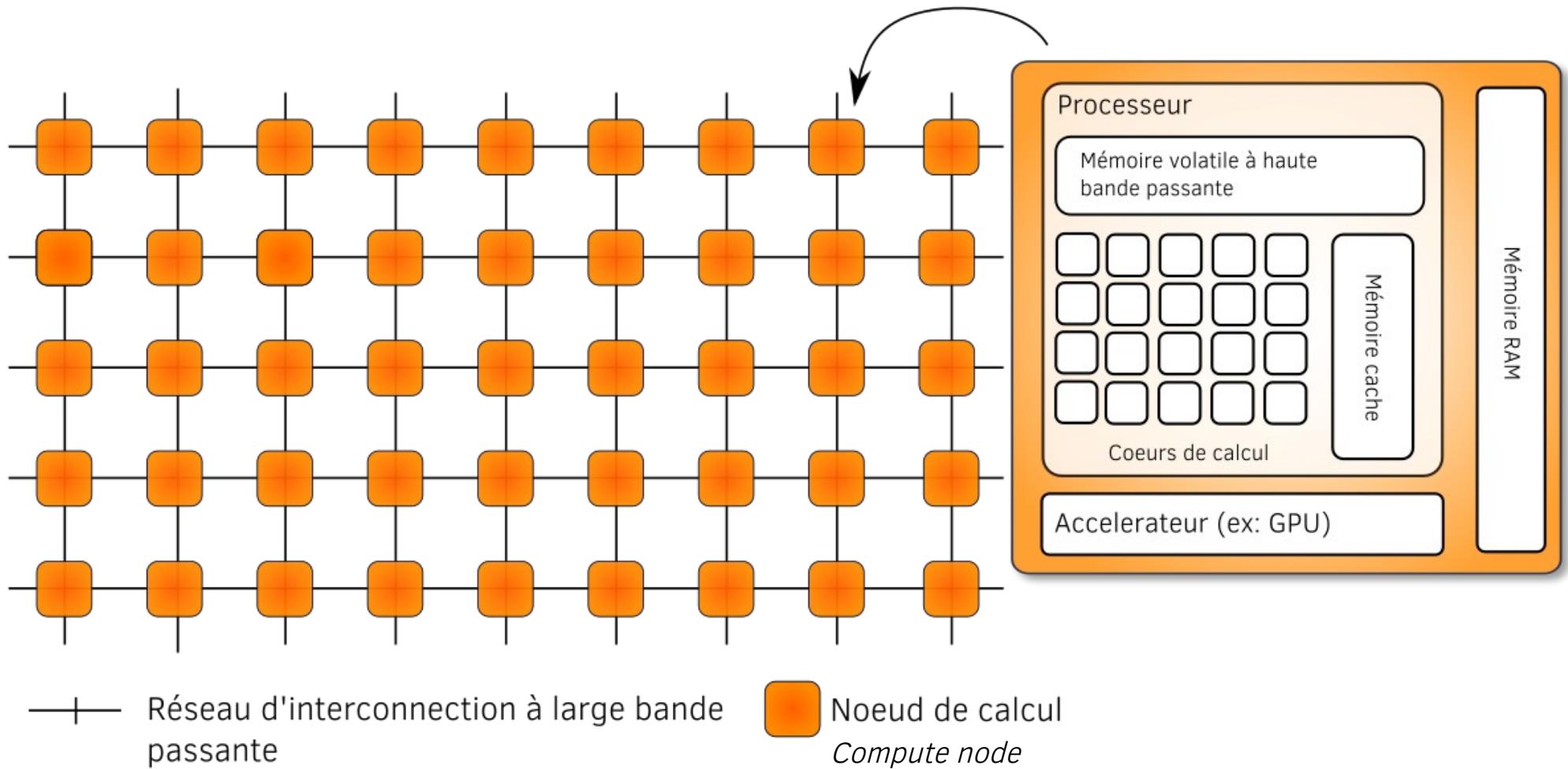


—+ Réseau d'interconnection à large bande
passante

 Noeud de calcul
Compute node

Architecture des super-calculateurs : intérieur d'un nœud de calcul

Compute node



Architecture des coeurs : notion de vectorisation

Vectorization

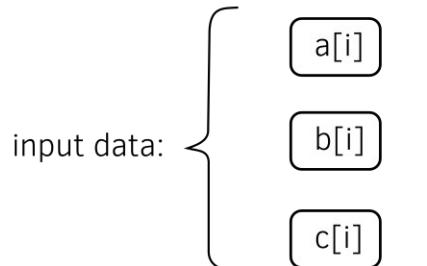
La vectorisation consiste à effectuer **la même opération** simultanément en un faible nombre de cycles sur **un vecteur de données** contigu en mémoire.

Les cœurs les plus modernes peuvent appliquer une multiplication et une addition (FMA) sur 8 doubles.

```
for(int i = 0 ; i < n ; i++) {  
    if (condition dependent of i) {  
        d[i] = a[i] + b[i]*c[i]  
    }  
}
```

scalar FMA

if-condition:



vectorized FMA

Mask

1	0	1	1	1	1	0	1	
a	a[i]	a[i+1]	a[i+2]	a[i+3]	a[i+4]	a[i+5]	a[i+6]	a[i+7]
b	b[i]	b[i+1]	b[i+2]	b[i+3]	b[i+4]	b[i+5]	b[i+6]	b[i+7]
c	c[i]	c[i+1]	c[i+2]	c[i+3]	c[i+4]	c[i+5]	c[i+6]	c[i+7]
d	d[i]	d[i+1]	d[i+2]	d[i+3]	d[i+4]	d[i+5]	d[i+6]	d[i+7]

Notion de puissance de calcul



Être en mesure d'exploiter l'intégralité de la puissance de calcul dépend de beaucoup de paramètres :

- Caractéristiques du réseau reliant les nœuds entre eux (architecture, bande-passante, latence)
- Les couches logicielles utilisées
- Le type d'algorithme à paralléliser
- Le niveau d'optimisation des programmes

C'est tout l'expertise du Calcul Haute-Performance.

Notion d'Exascale



Il existe une compétition mondiale pour la construction de la première machine capable d'atteindre une puissance exaflopique : c'est à dire **10^{15} opérations à virgule flottante par seconde**.

Le défi consiste surtout à atteindre cet objectif sans dépasser une certaine enveloppe énergétique (entre 20 et 40 MW de consommation).

Les constructeurs misent sur une diminution permanente de la consommation par transistor, sur des optimisations matérielles et sur l'utilisation d'unités de calcul spécialisées.

Les compétences en HPC

Comme pour le milieu de la dynamique des fluides, le HPC regroupe un grand nombre de compétences et de spécialités :

- Optimisation intra-noeuds
- Parallélisme distribué et massif
- Programmation des accélérateurs
- Les modèles de programmation parallèles (abstraction, tâche...)
- Entrées-sorties parallèles
- Le traitement parallèle de jeux de données massifs (big data)
- La visualisation parallèle
- Le machine learning parallèle
- Le calcul quantique
- ...

Introduction au calcul parallèle

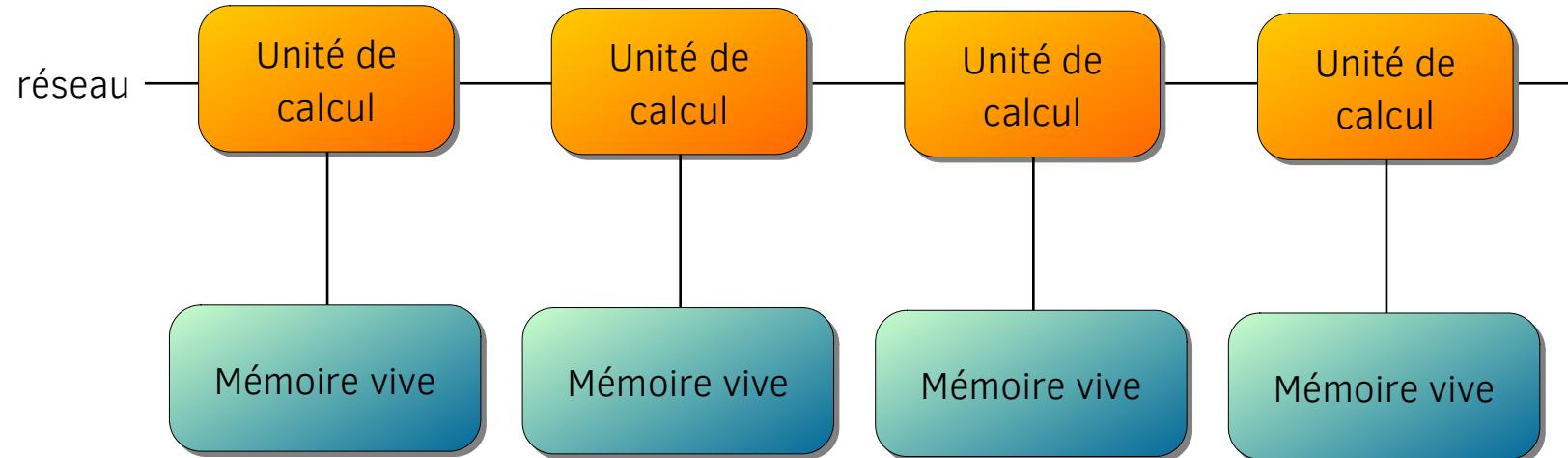
2) Architectures mémoires

Architecture mémoire : système à mémoire distribuée

Distributed memory system



Système composé d'un ensemble d'unités de calcul ou d'ordinateurs connectés entre eux par un réseau de communication rapide et possédant chacune une mémoire vive strictement privée.

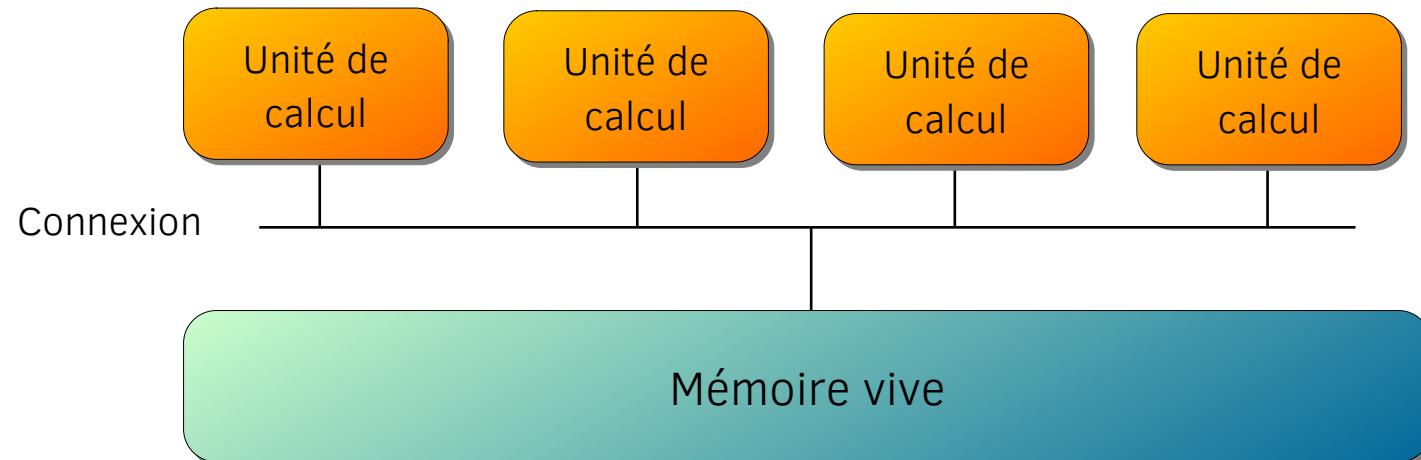


Architecture mémoire : système à mémoire partagée

shared memory system



Système composé d'un ensemble d'unités de calcul partageant la même mémoire vive



Architecture mémoire : système à mémoire partagée

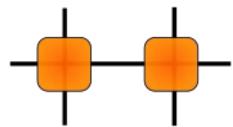
shared memory system



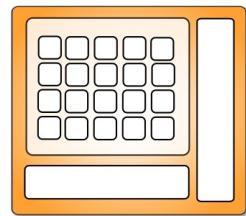
Un ordinateur portable, de bureau ou un smartphone est une machine parallèle (plusieurs cœurs) à mémoire partagée (même RAM)



Architecture mémoire des super-calculateurs : différents niveaux parallèles



Parallélisme inter-nœuds (*internode parallelism*) : modèle de parallélisme à mémoire distribuée

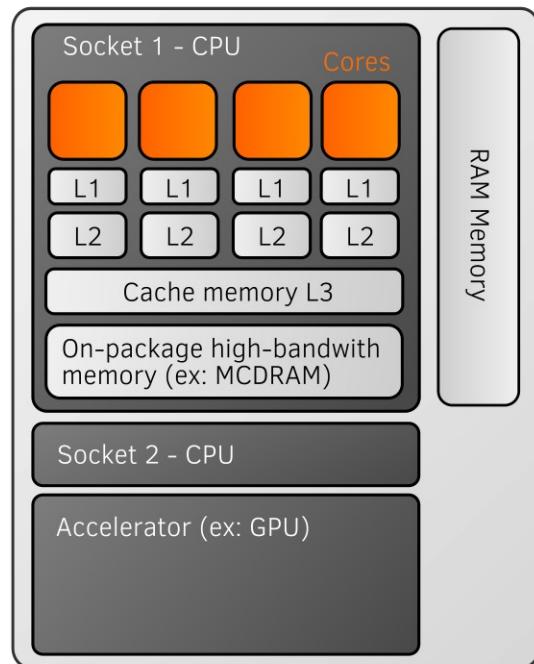


Parallélisme intra-noeud (*intranode parallelism*) : modèle de parallélisme à mémoire partagée

Introduction au monde du HPC

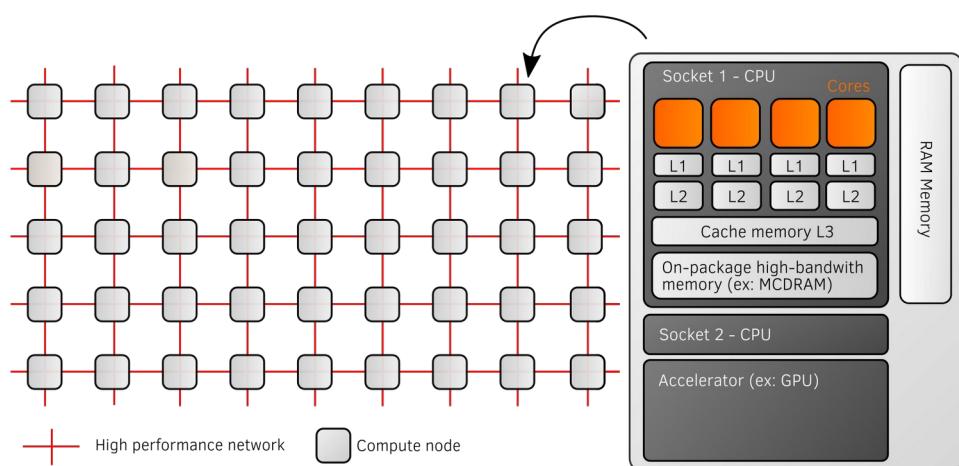
3) Programmation et exécution parallèle

Les bibliothèques pour la programmation en mémoire partagée



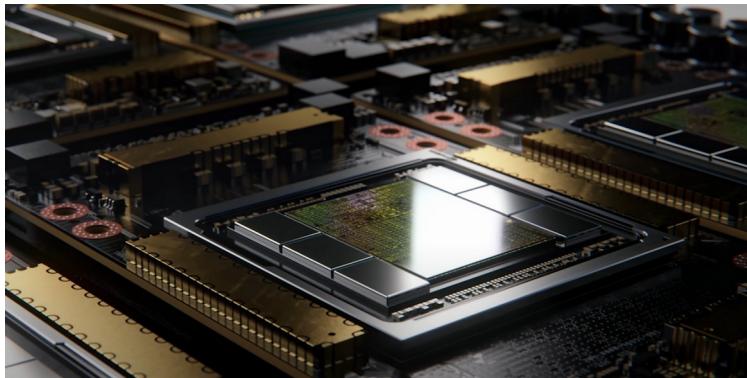
- Par directives : OpenMP
- Mémoire distribuée : MPI
- Parallélisme unifié : PGAS, Chapel, X10, MPC...
- Par tâches : pthread, OpenMP, C++ moderne...

Les bibliothèques pour la programmation en distribué



- Mémoire distribuée : MPI...
- Parallélisme unifié : PGAS, Chapel, X10, MPC...
- Par tâches : starPU...

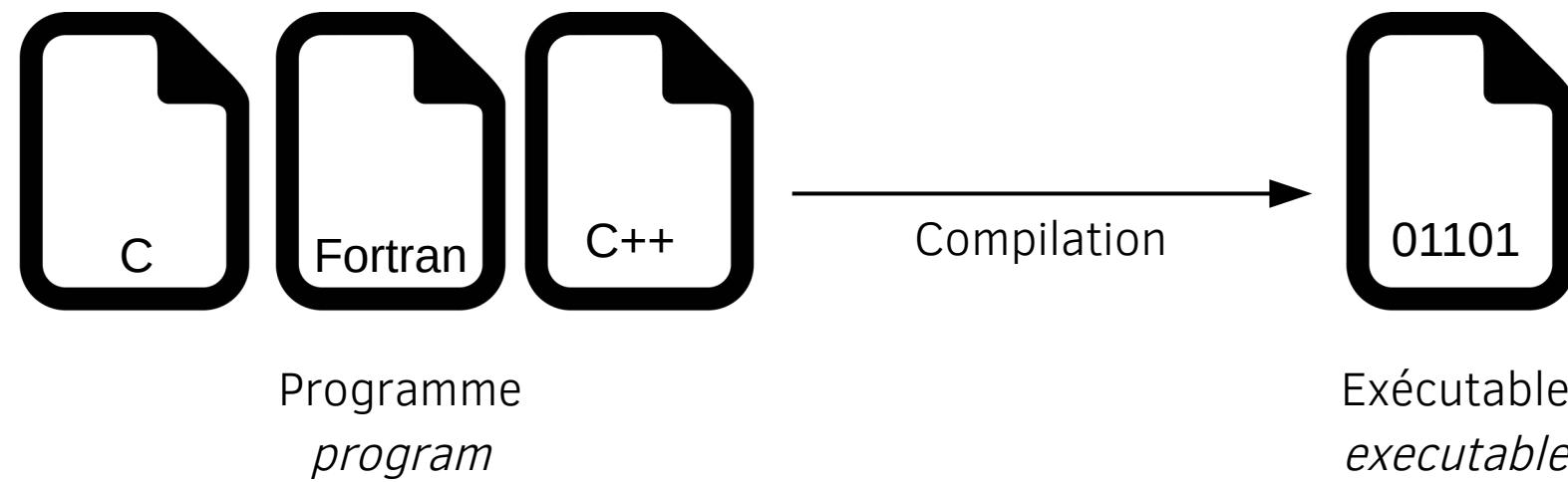
Les bibliothèques pour la programmation GPU



- Code constructeur : CUDA, HIP, OneAPI
- Portable : OpenCL
- Directive : OpenACC, OpenMP
- Abstraction : Kokkos, Raja, Sycl

Compilation

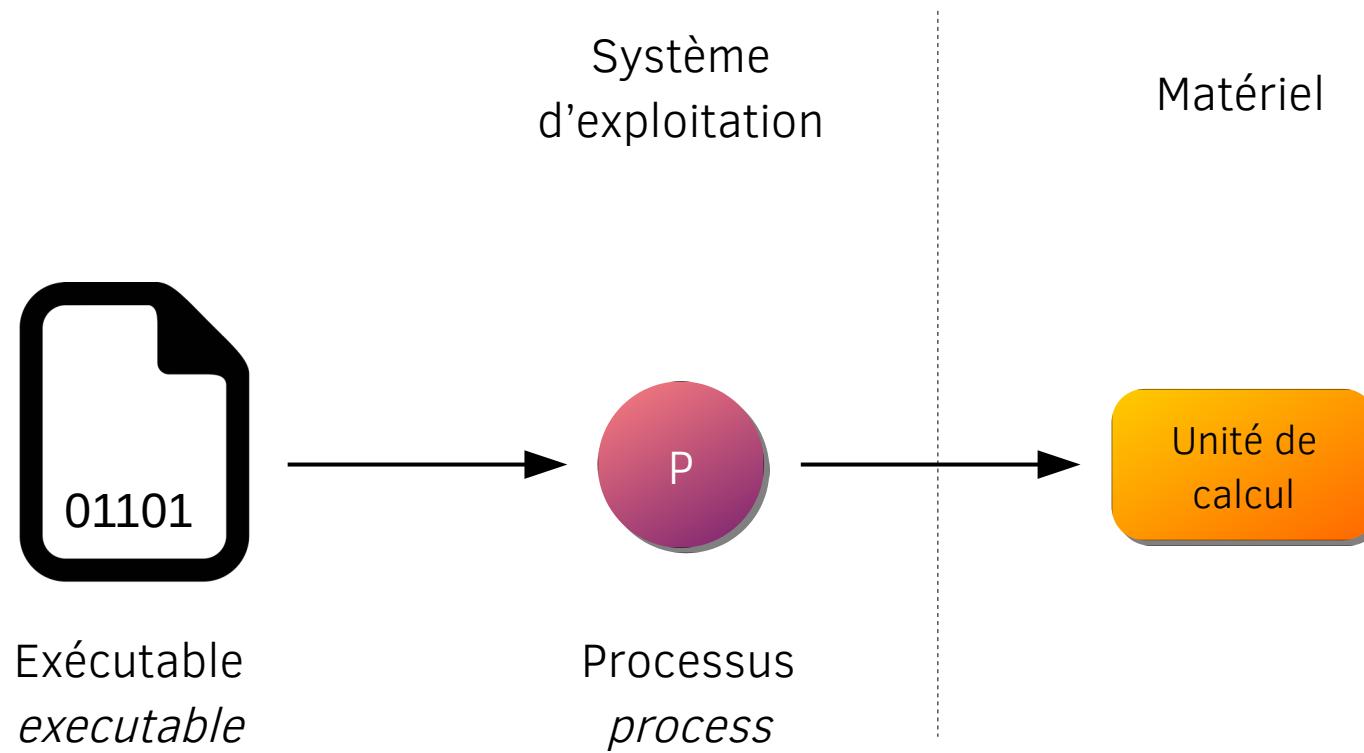
Compilation



Exécution séquentielle

Sequential execution

Un programme en cours d'exécution séquentielle devient un processus actif. Ce processus est exécuté par une seule unité de calcul.

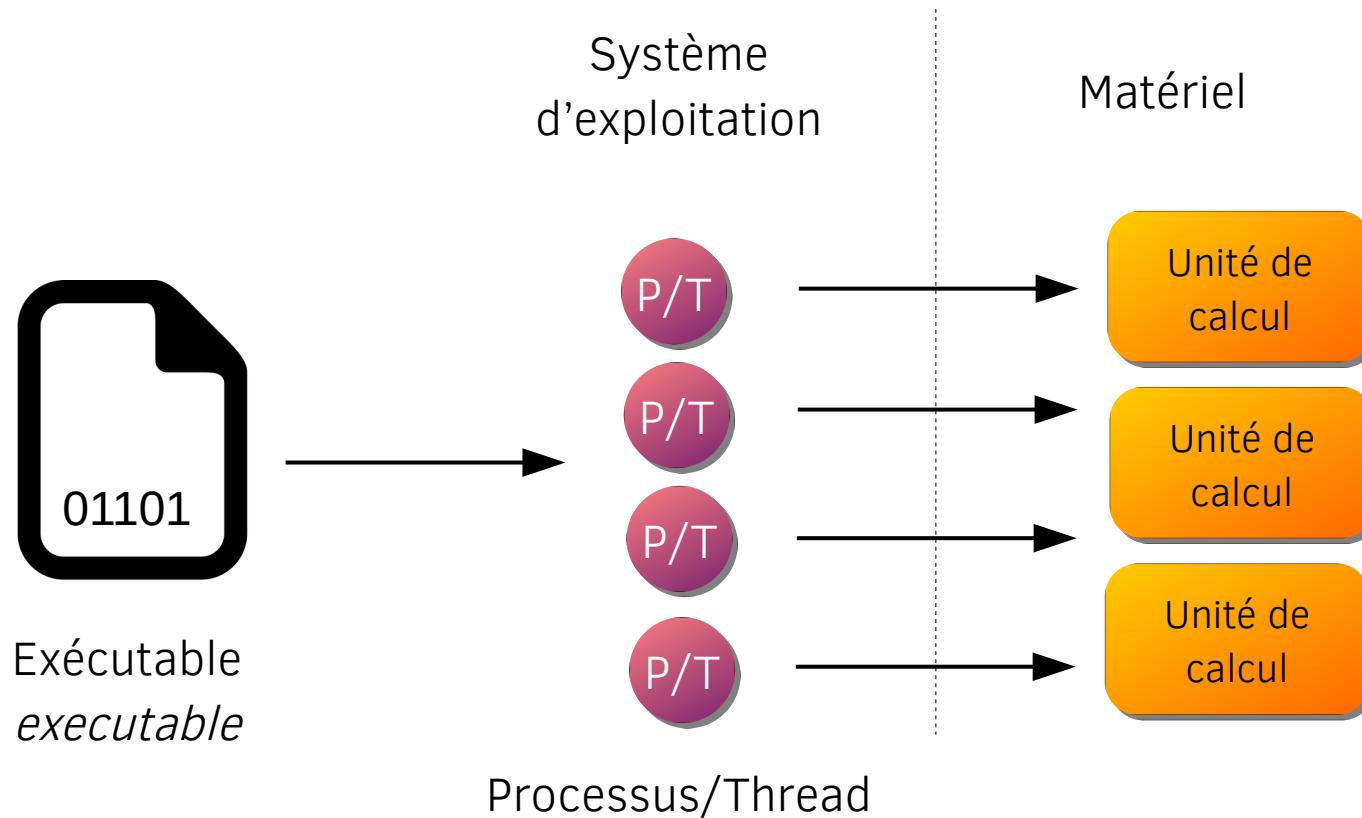


Exécution parallèle

Parallel execution



Un programme en cours d'exécution parallèle est dupliqué en plusieurs processus actifs ou threads. Ces processus sont exécutés par plusieurs unités de calcul.

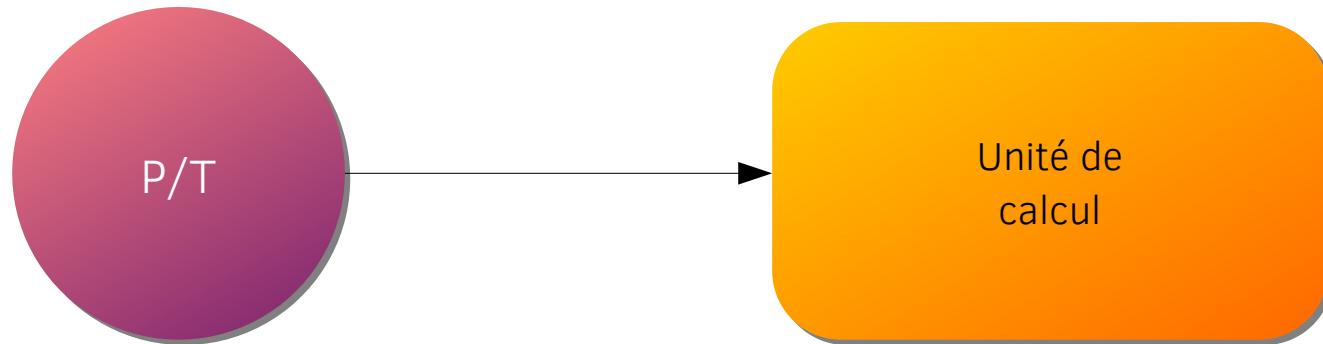


Exécution parallèle

Parallel execution



Par simplification, nous décidons ici qu'une unité de calcul est toujours en charge d'un seul et unique processus. Cela n'est en fait pas forcément le cas.



Modèle de programmation parallèle

Parallel programming model



Un modèle de programmation parallèle est une approche de programmation, souvent basée sur une **bibliothèque** (*library*) ou une API, permettant **de faciliter la résolution d'un problème parallèle** et/ou **de s'adapter à une ou plusieurs architectures parallèles**.

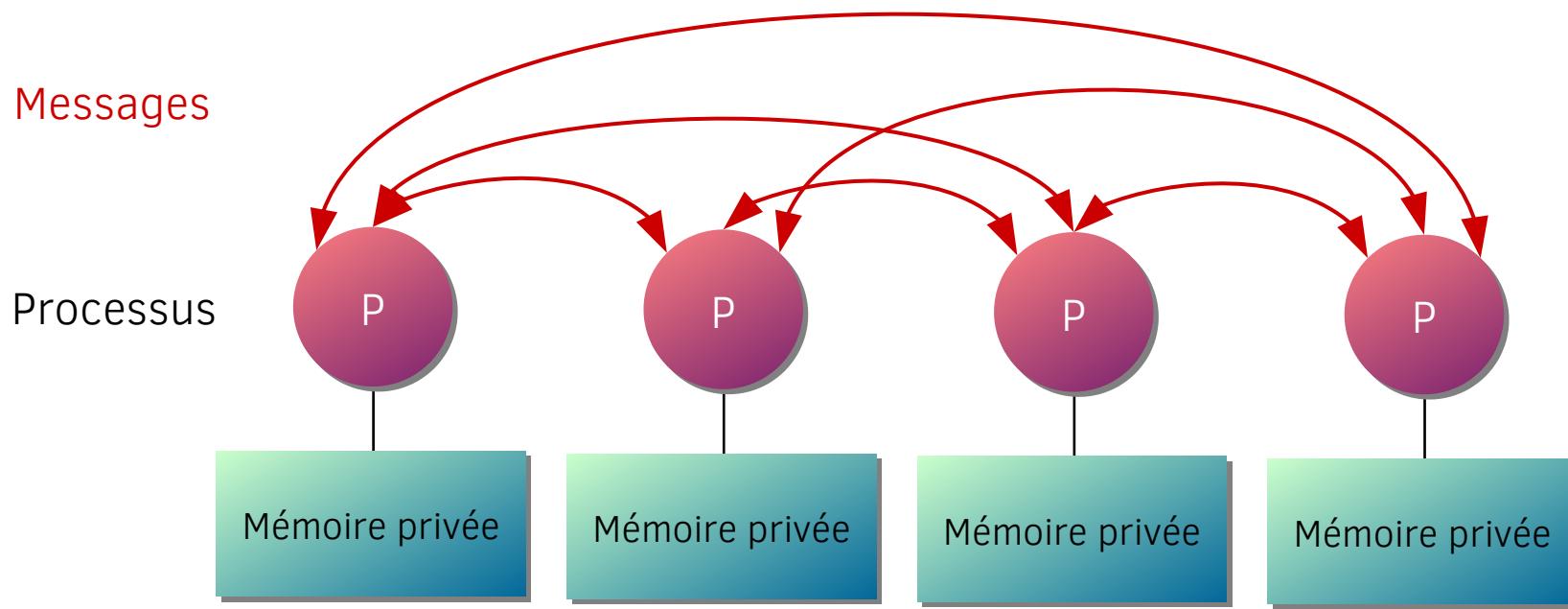
- Mémoire distribuée : paradigme par passage de messages
- Mémoire partagée: paradigme *fork-join* ou *divide and conquer*

Paradigme par passage de messages : définition

Message passing programming model

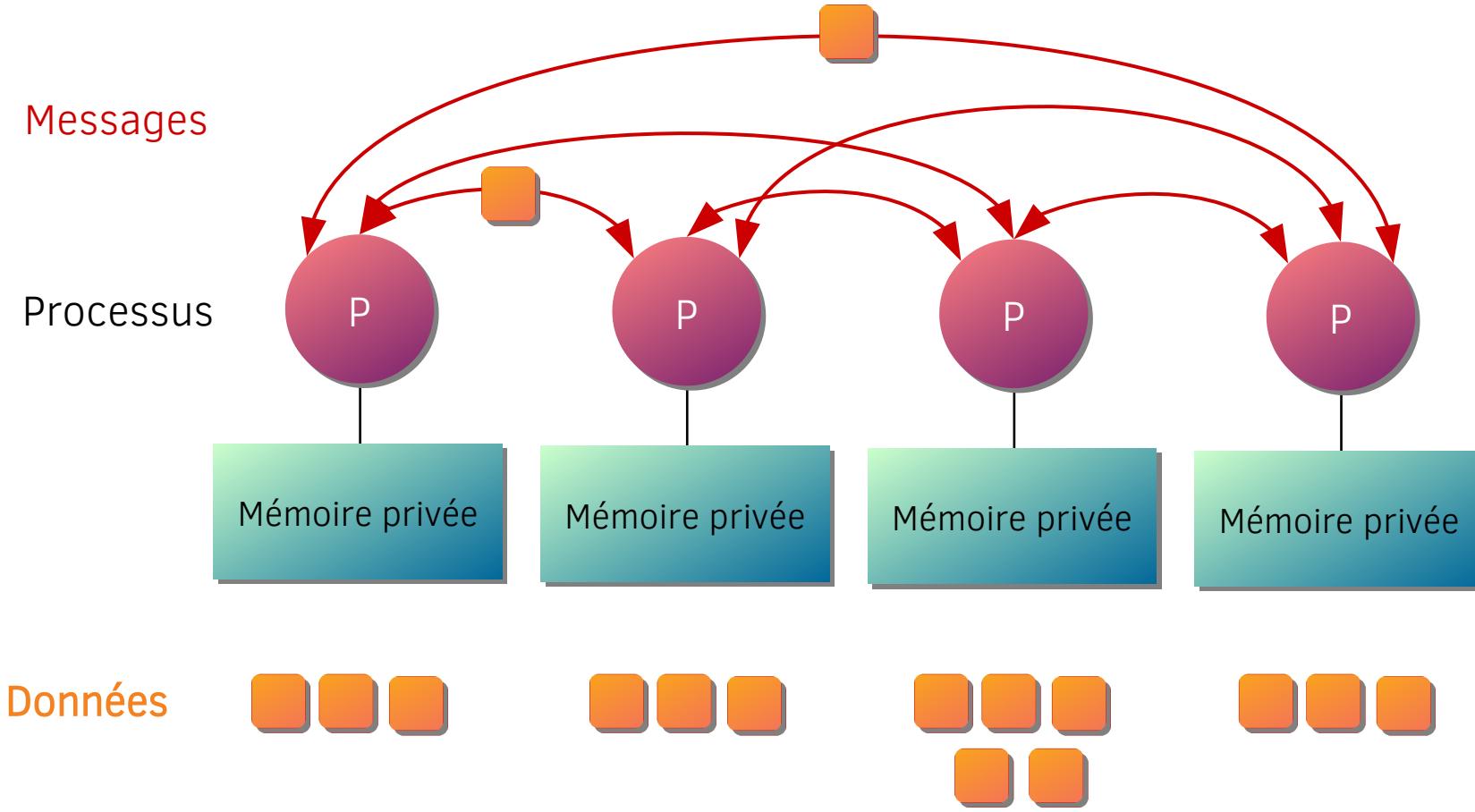


Dans un modèle par échange de messages, les différents exécutants (*process*) ont leurs **données propres** qu'ils communiquent entre eux par **messages**. Ces messages peuvent servir à **échanger** une ou plusieurs données, ou se **synchroniser**.



Paradigme par passage de messages : définition

Message passing programming model



Paradigme par passage de messages : définition

Message passing programming model



C'est l'un des modèles que nous verrons dans ce cours et **le plus utilisé en calcul scientifique**

Paradigme par passage de messages : la bibliothèque MPI

Message passing programming model



Le standard le plus utilisé pour ce paradigme est **MPI** (Message Passing Interface).



Ce genre de paradigme, bien que **conçu pour un parallélisme distribué, fonctionne** parfaitement sur une architecture mémoire partagée.

Paradigme *fork-join*

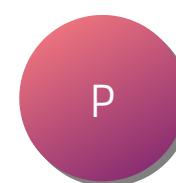


Mémoire partagée

Données
partagées

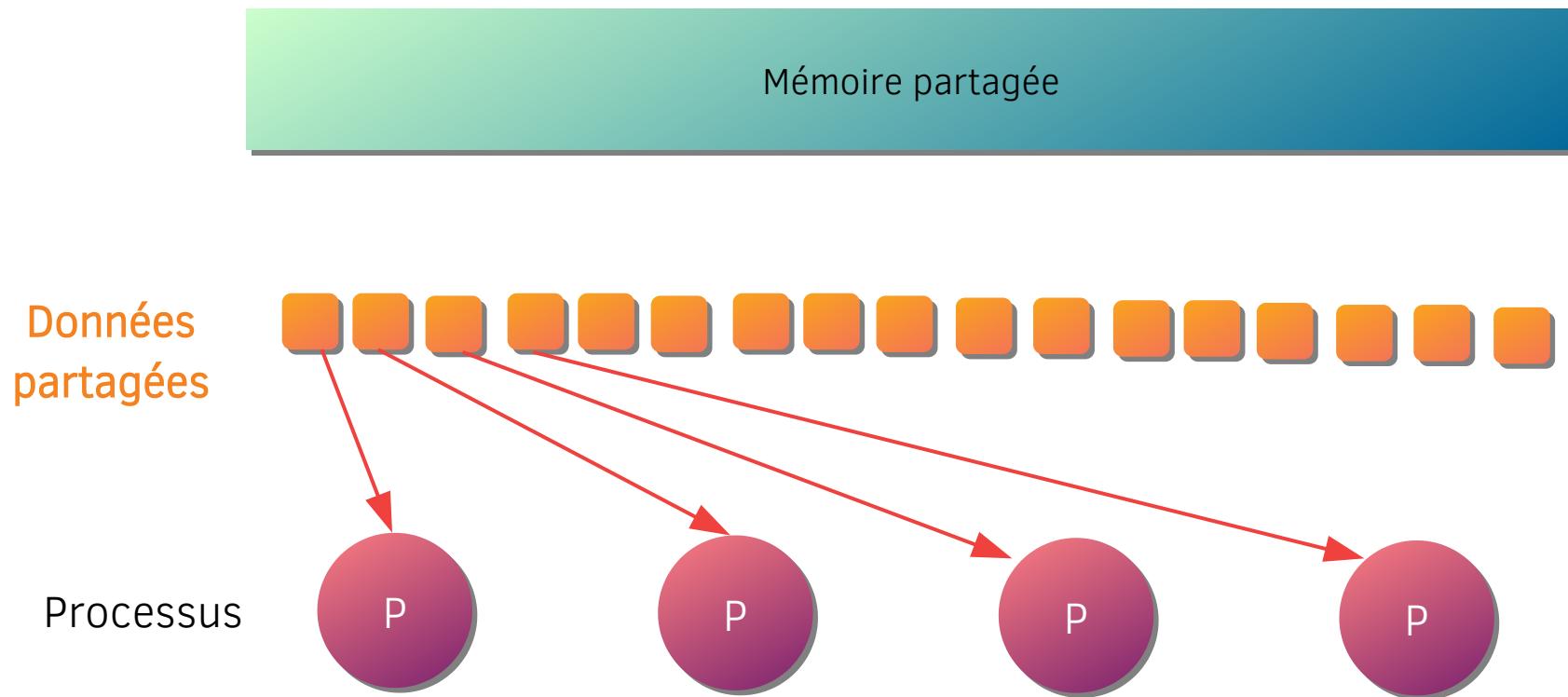


Processus



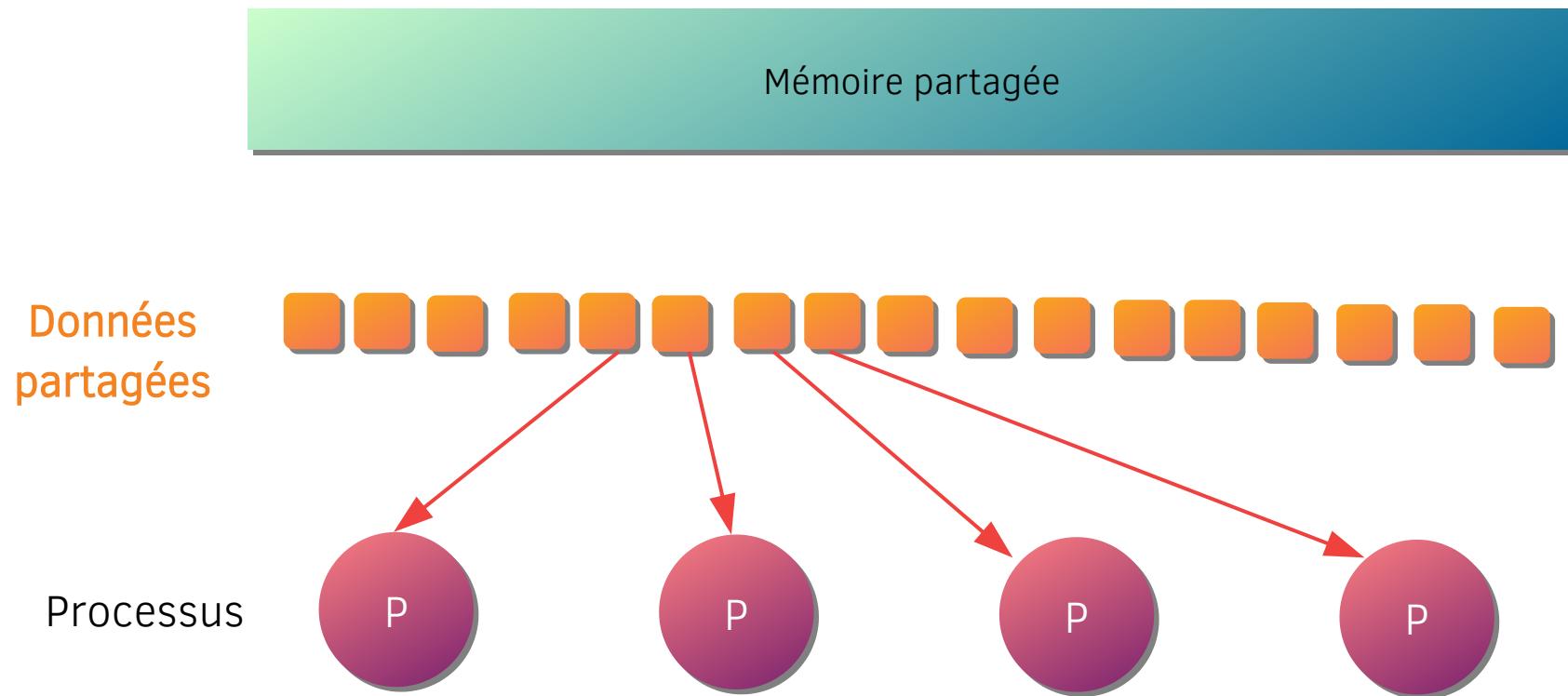
La bibliothèque la plus courante en calcul scientifique est [OpenMP](#). Ce modèle ne fonctionne que pour une mémoire partagée.

Paradigme *fork-join*



La bibliothèque la plus courante en calcul scientifique est [OpenMP](#). Ce modèle ne fonctionne que pour une mémoire partagée.

Paradigme *fork-join*



La bibliothèque la plus courante en calcul scientifique est [OpenMP](#). Ce modèle ne fonctionne que pour une mémoire partagée.

Obtenir des informations sur le processeur de son ordinateur

Dans un terminal, vous pouvez ouvrir le fichier /proc/cpuinfo :

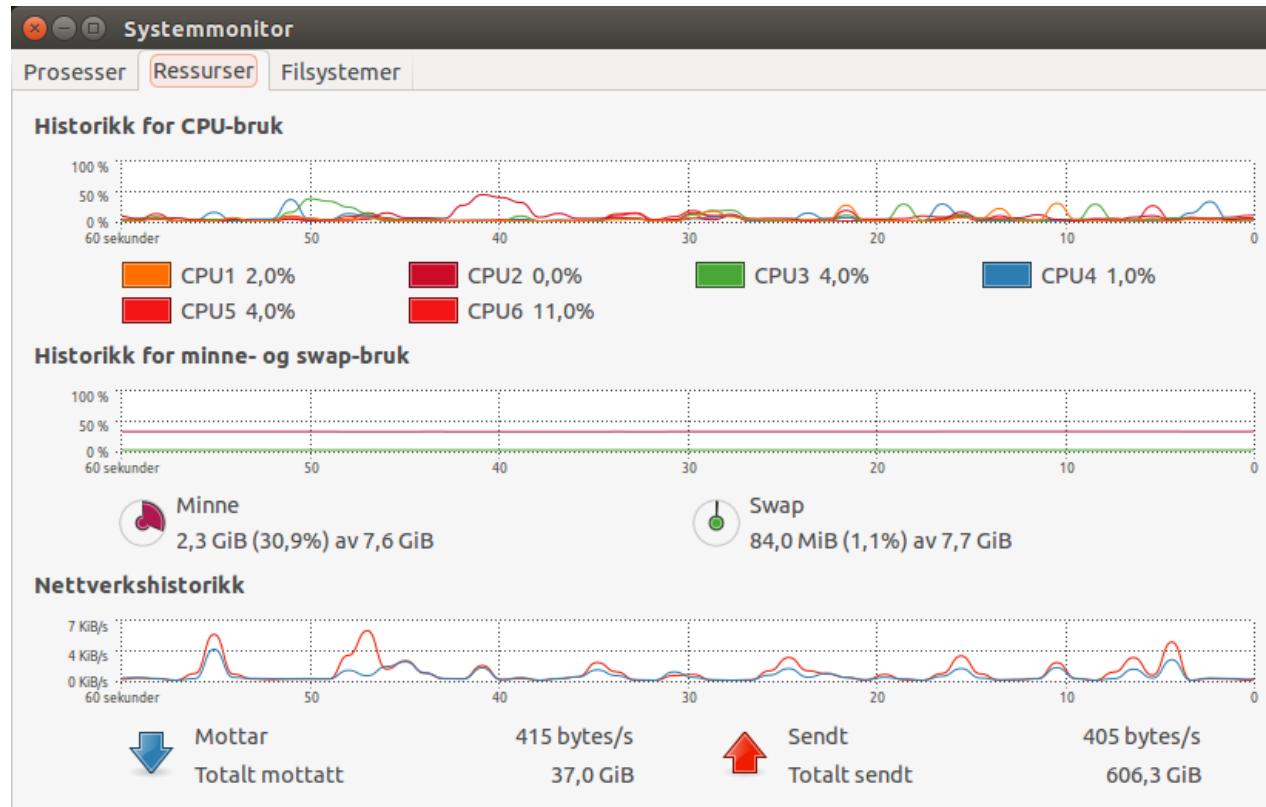


```
> cat /proc/cpuinfo
...
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model    : 78
model name : Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz
...
```

Utilisez ensuite le site du constructeur pour plus d'informations

Obtenir des informations sur le processeur de son ordinateur

Sous environnement gnome, il est possible d'utiliser l'utilitaire graphique moniteur système (paquet gnome-system-monitor)



Rappel sur la compilation

Compilation sans optimisation sous Fortran avec GNU



```
> gfortran program.f90 -o nom_du_binaire
```

Compilation sans optimisation en C avec GNU



```
> gcc program.c -o nom_du_binaire
```

Compilation sans optimisation en C++ avec GNU



```
> g++ program.cpp -o nom_du_binaire
```

Flags d'optimisation

Il est conseillé de compiler en spécifiant un *flag* d'optimisation :

- -O2 ou -O3 permet d'obtenir les meilleures performances grâce à des optimisations du compilateur



```
> gfortran -O3 program.f90 -o nom_du_binaire
```



```
> g++ -O3 program.cpp -o nom_du_binaire
```



<https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html>