

Introduction au Calcul Haute Performance

Master DFE – année 2019/2020

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

Disponible sur le GitLab de la Maison de la Simulation



<https://gitlab.maisondelasimulation.fr/mlobet/cours-hpc-m2-dfe>

Description des pictogrammes pour les *slides*



Désigne une *slide* importante à retenir



Désigne une *slide* secondaire ou de culture générale



Désigne une *slide* exercice

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

Prérequis pour ce cours

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 multitâche par directives via openMP
- III. Introduction au parallélisme par échange de message via MPI
- IV. Mesure de la performance
- V. Travaux pratiques sur le cas de l'équation d'onde

I. 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 **machines de calcul parallèles ou 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.



- 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 prédiction/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

Machine parallèle

Parallel machine



Une **machine parallèle** est tout simplement une machine composée de **plusieurs unités de calcul**.

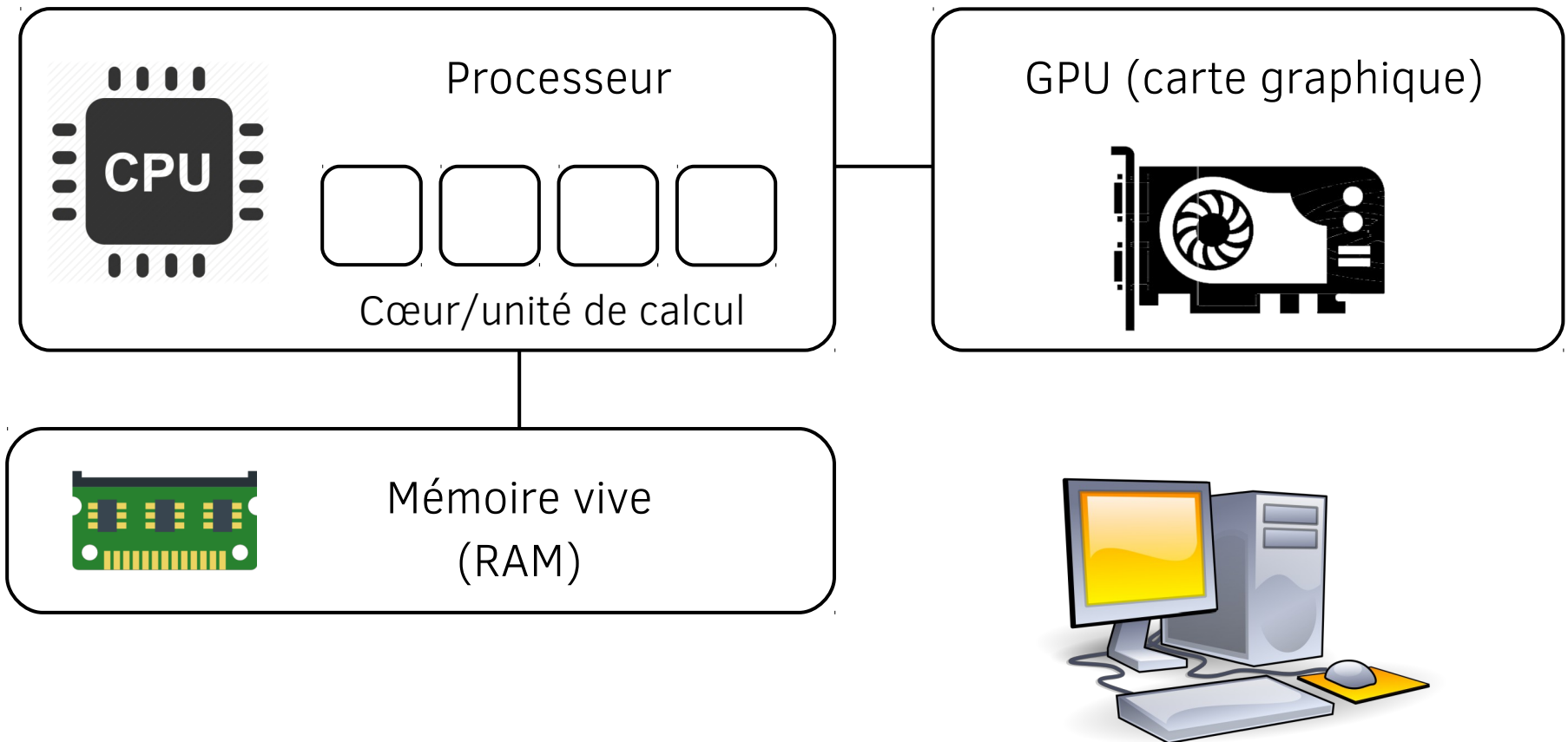


Les processeurs d'ordinateur, de téléphones portables ou les cartes graphiques sont tous aujourd'hui des composants parallèles car composés de plusieurs cœurs de calcul.

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.



Super-calculateur Atos du CEA



Summit, le plus puissant calculateur au monde est aux États-Unis

Pourquoi des super-calculateurs ?



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

En science :

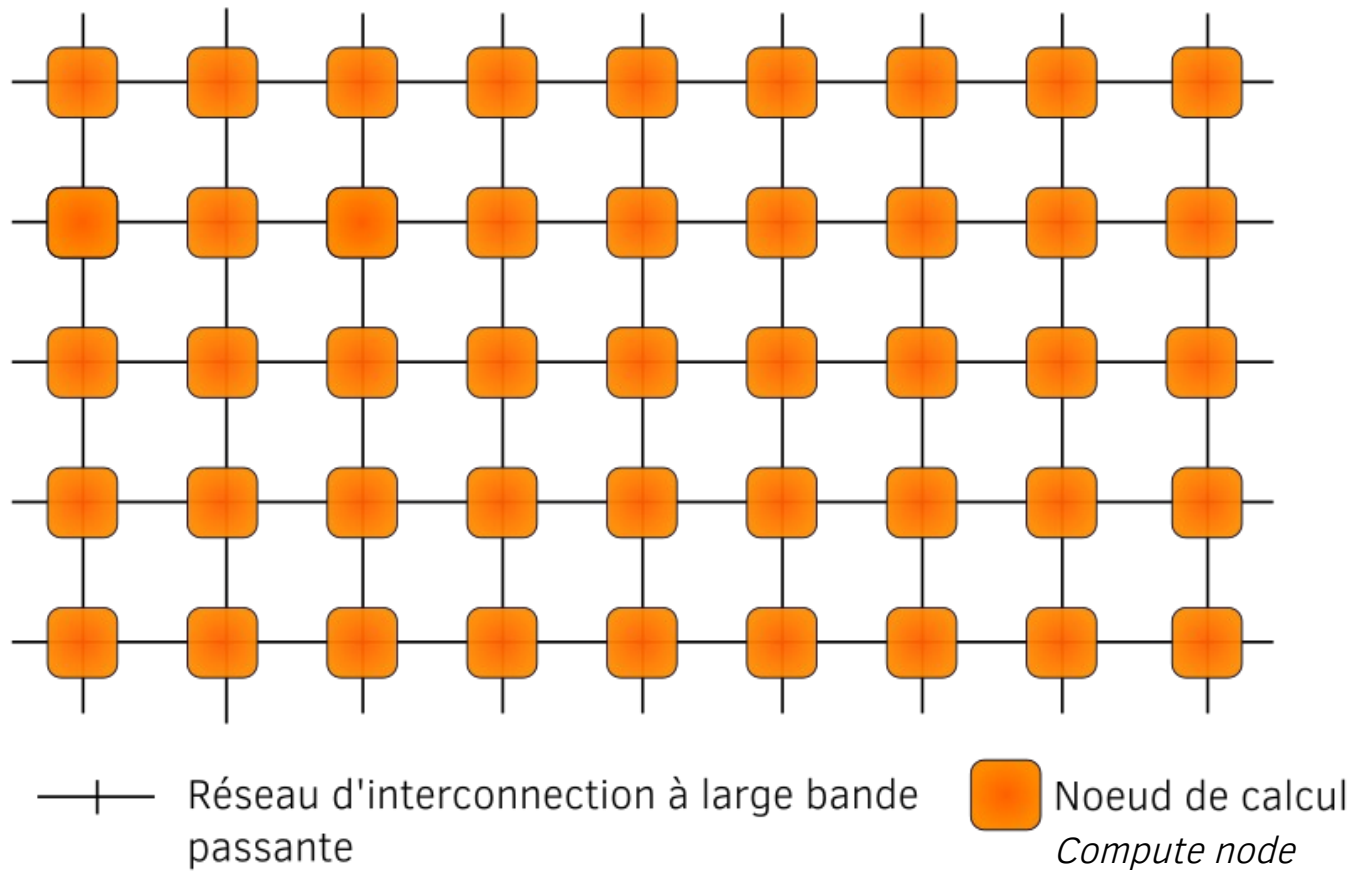
- Résoudre des problèmes avec beaucoup d'éléments ou sur des temps très longs
- Effectuer un grand nombre de simulations (étude paramétrique)
- Résoudre des problèmes multi-D complexe
- Résoudre des problèmes multi-physiques
- Traiter des données issues d'expériences (accélérateurs, satellites..)



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

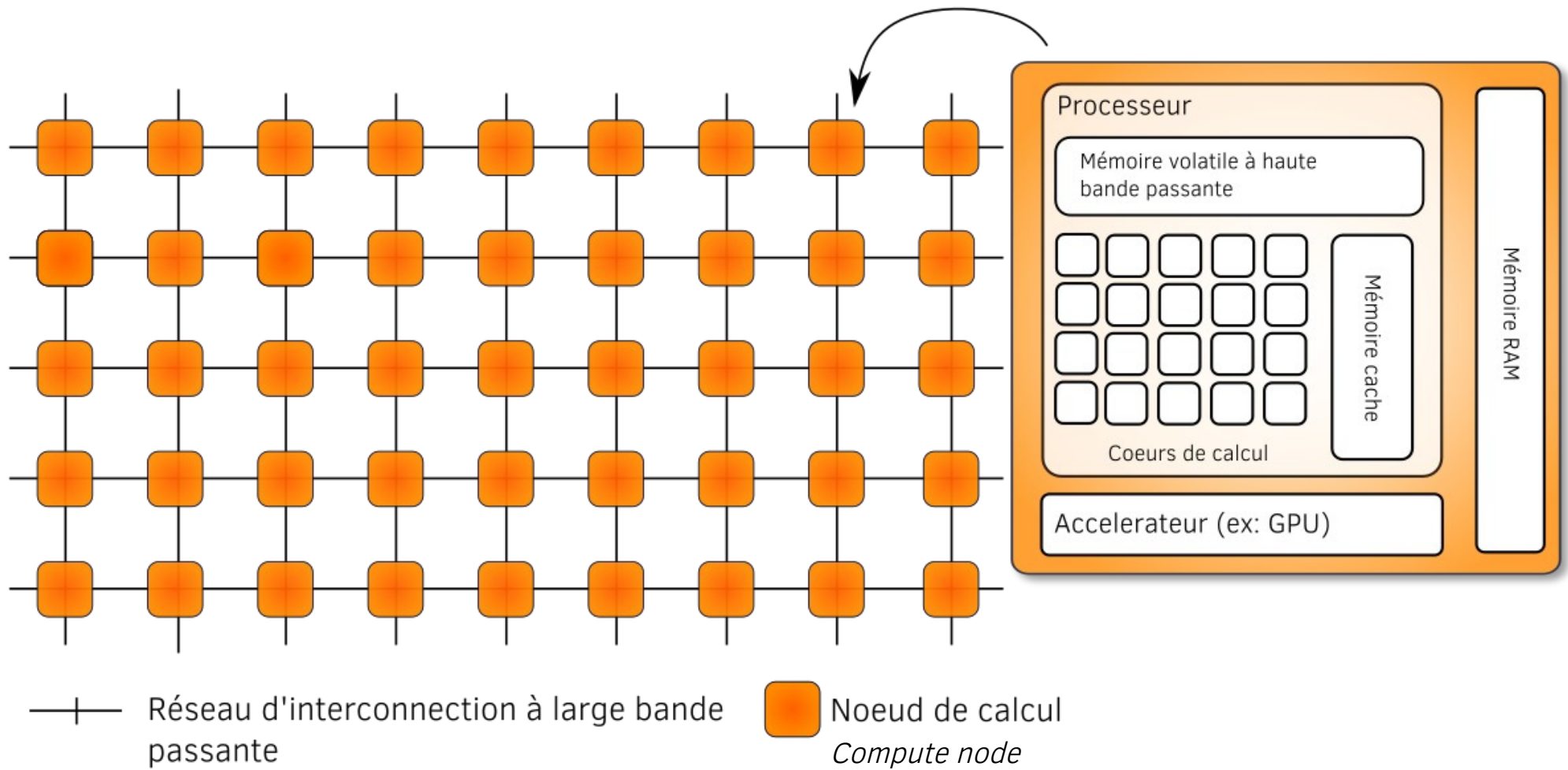
Architecture des super-calculateurs – principe d'une ferme de calcul

Computer cluster



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

Compute node

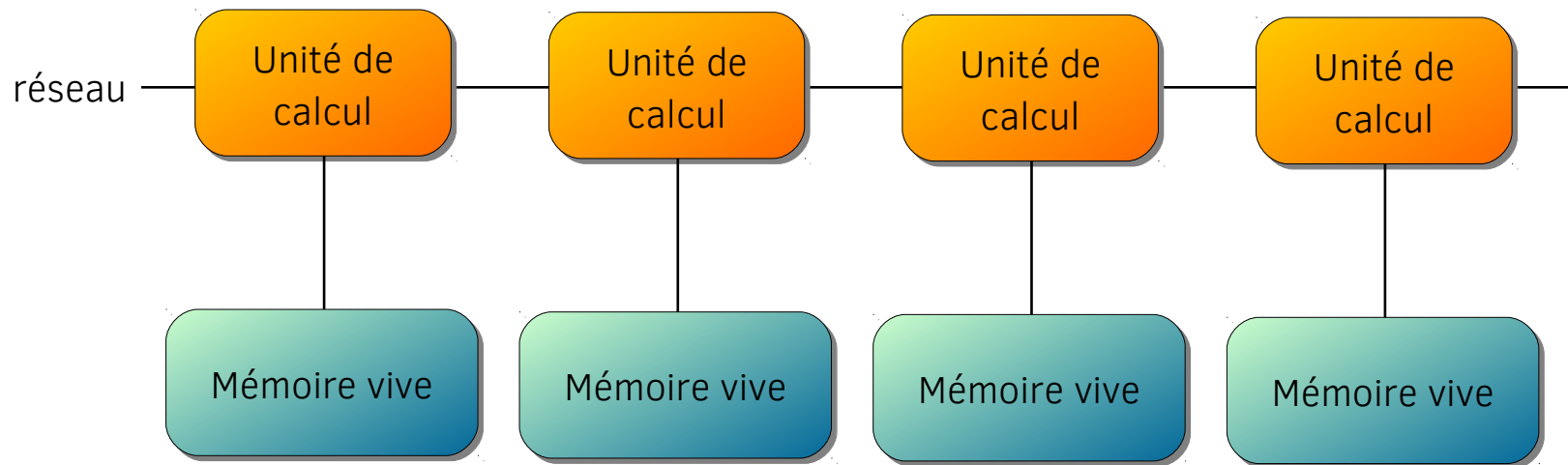


Architecture mémoire : système à mémoire distribuée

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.

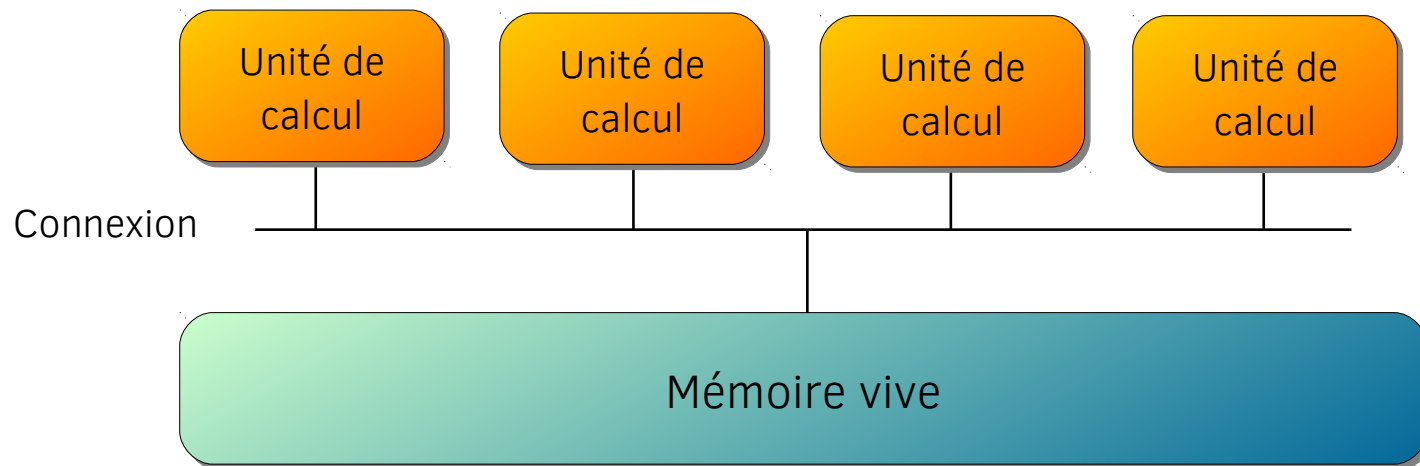


Architecture mémoire : système à mémoire partagée

shared memory system



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



Architecture mémoire : système à mémoire partagée

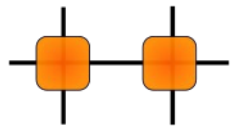
shared memory system



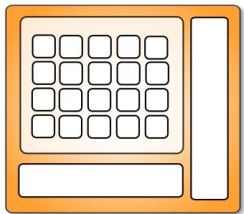
Un ordinateur portable, de bureau ou un smartphone est une machine parallèle (plusieurs cœurs) à mémoire partagée (même RAM)



Architecture mémoire des super-calculateurs : différents niveaux parallèles



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



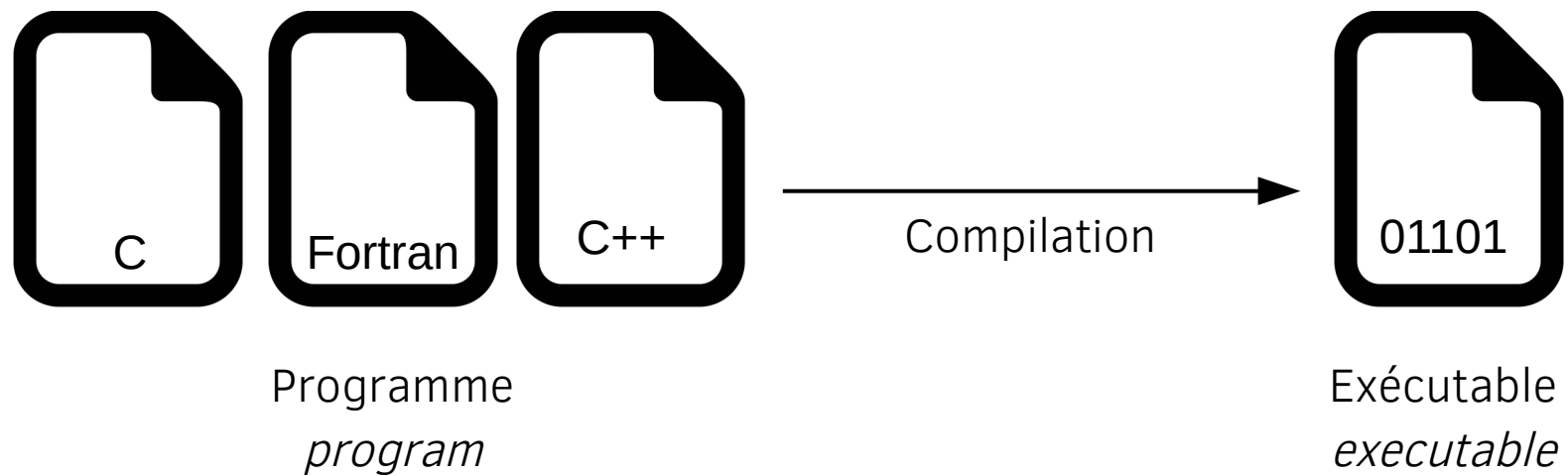
Parallélisme intra-nœud (*intranode parallelism*) : modèle de parallélisme à mémoire partagée

I. Introduction au monde du HPC

2) Programmation et exécution parallèle

Compilation

Compilation

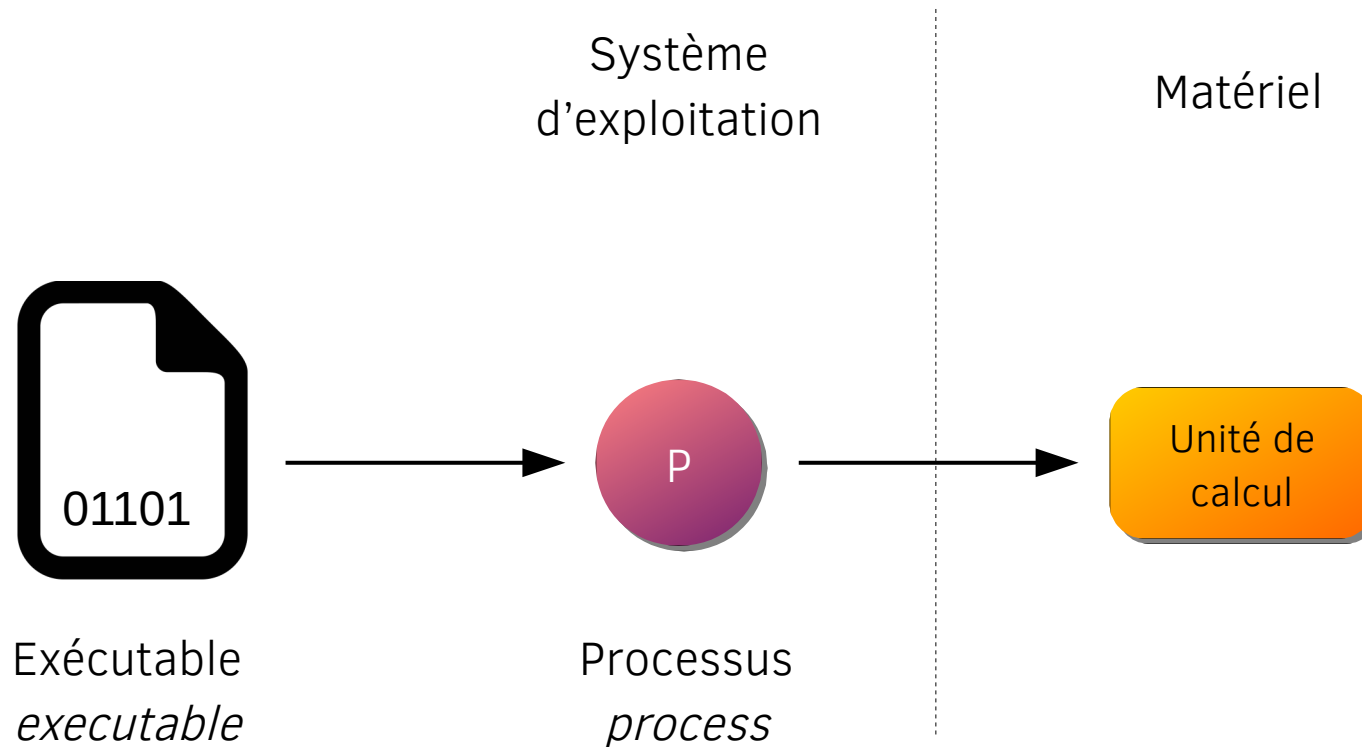


Exécution séquentielle

Sequential execution



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

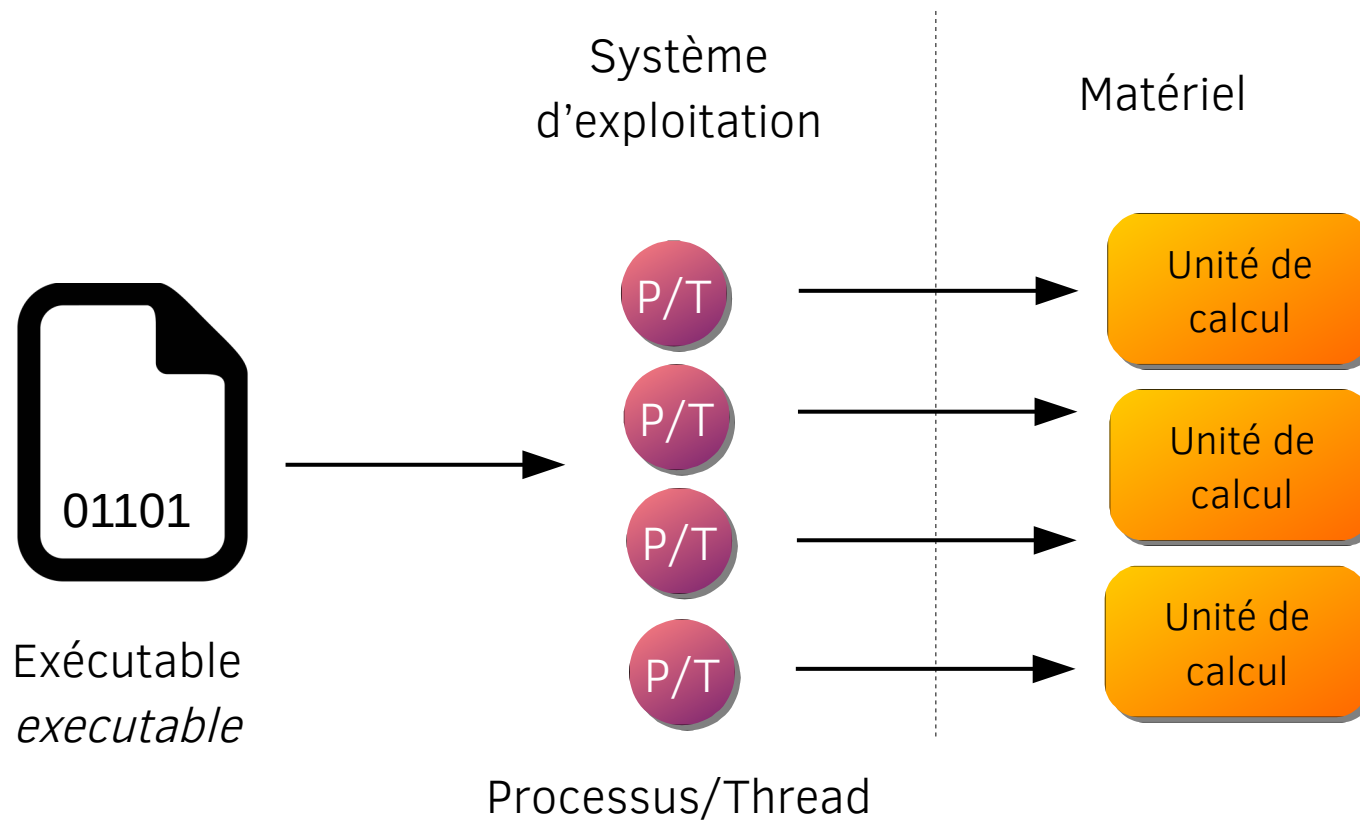


Exécution parallèle

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**.

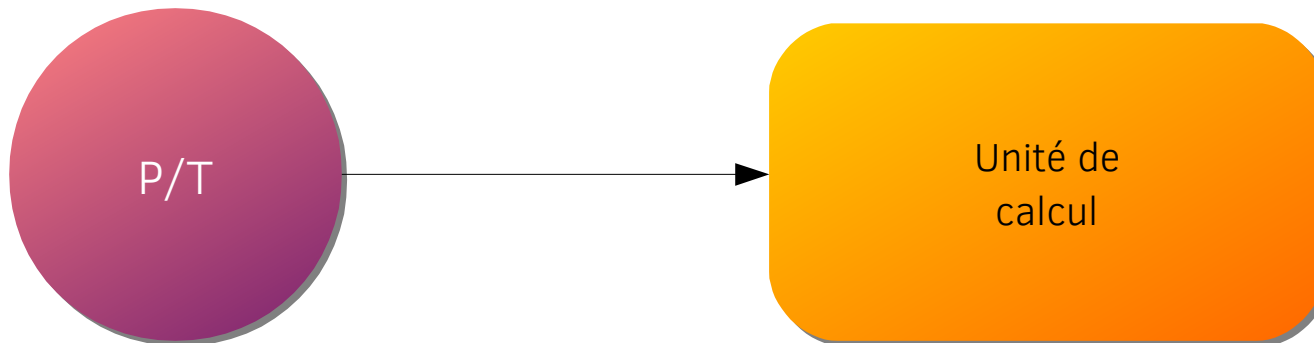


Exécution parallèle

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

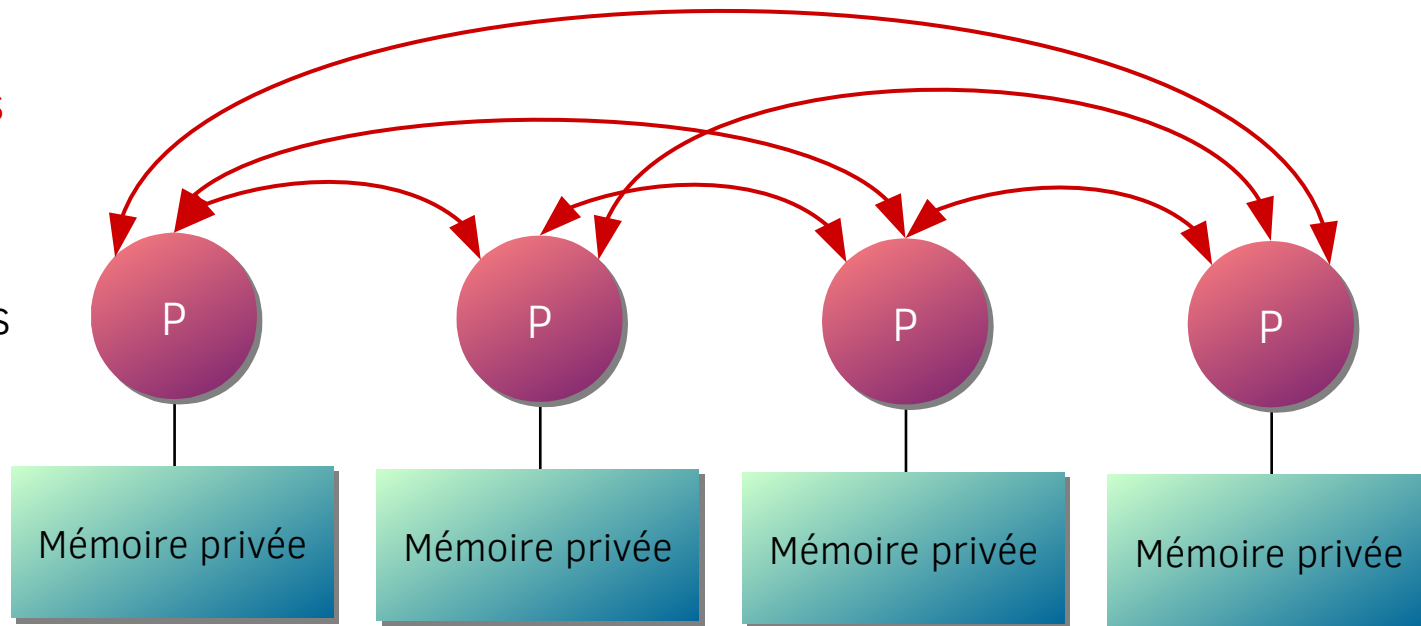
Message passing programming model



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**.

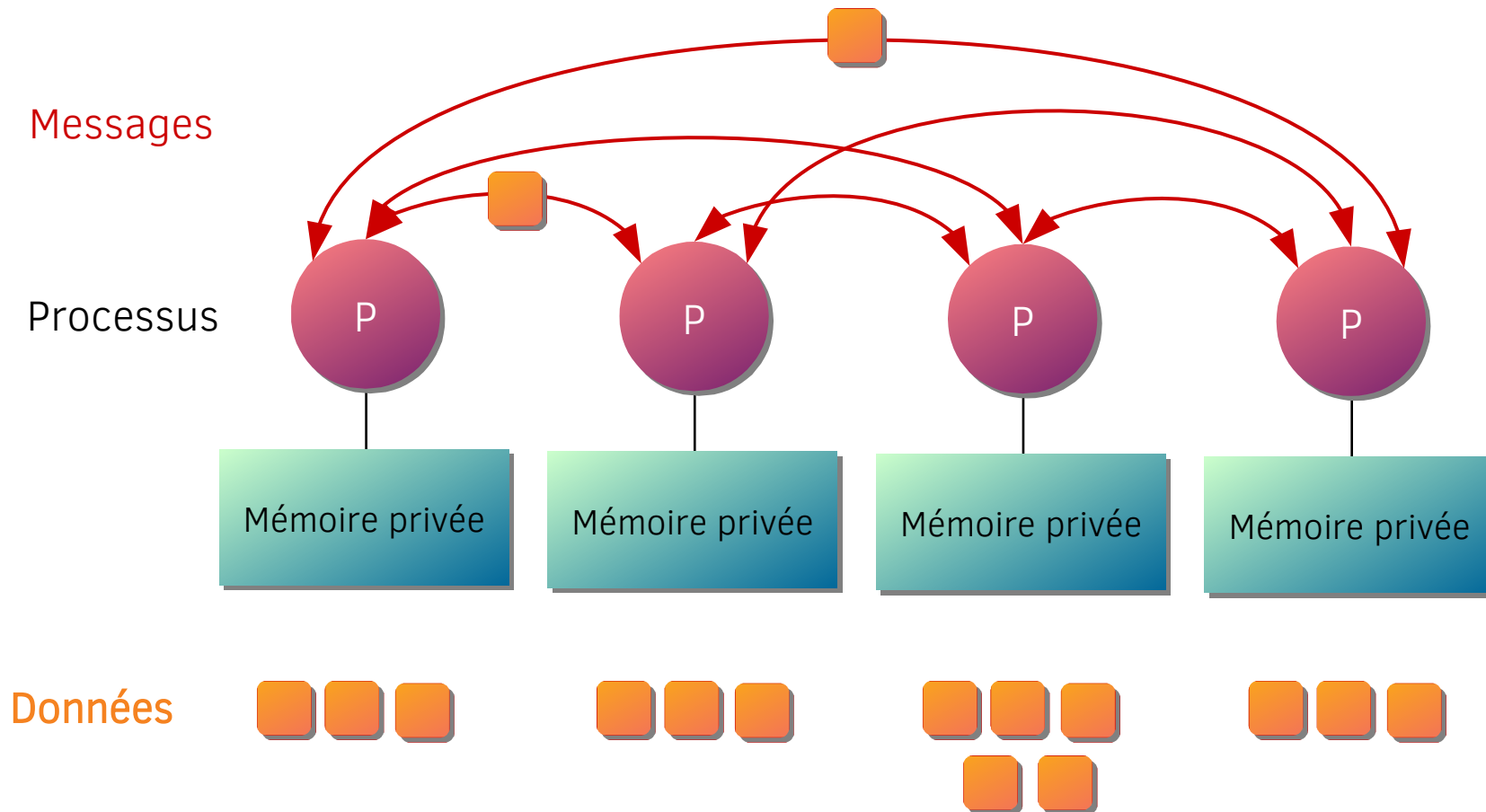
Messages

Processus



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

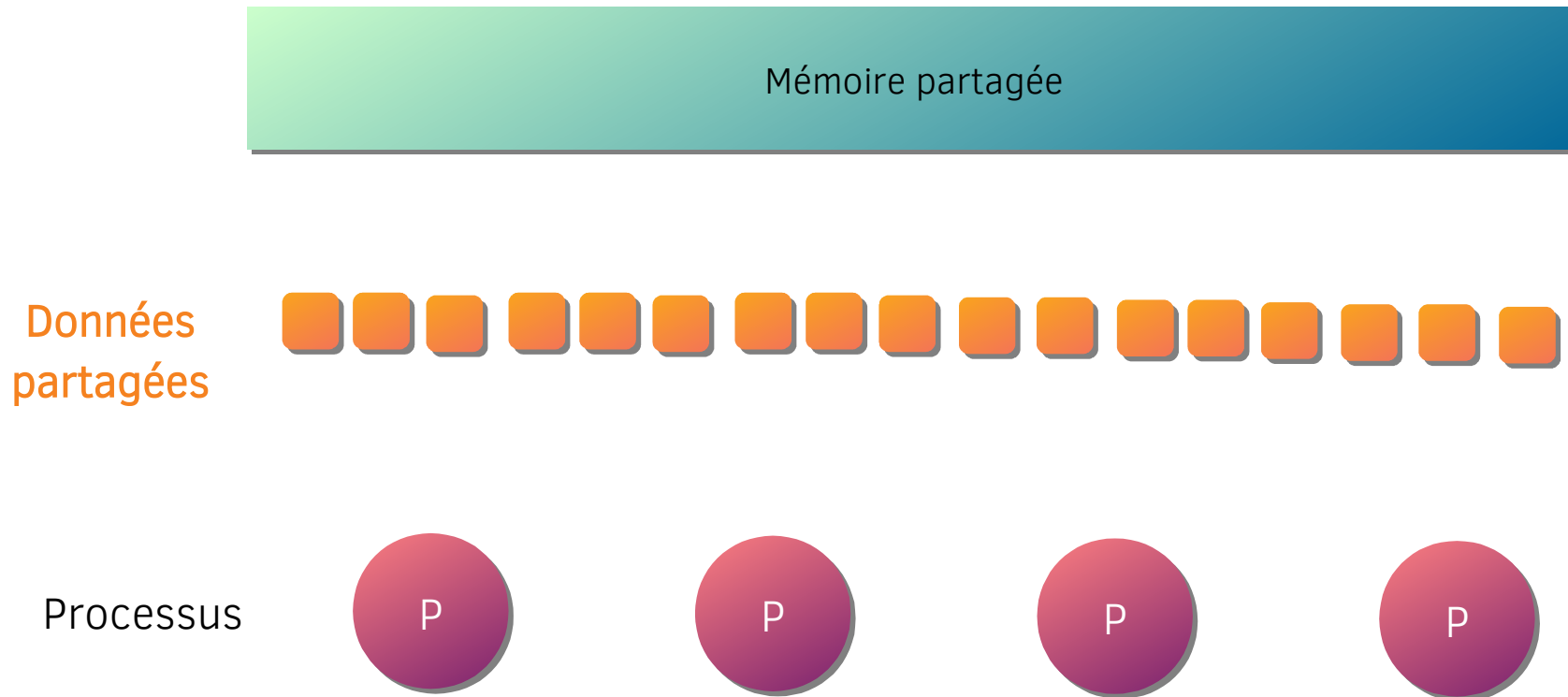


Le standard le plus utilisé pour ce paradigme est **MPI (Message Passing Interface)**.



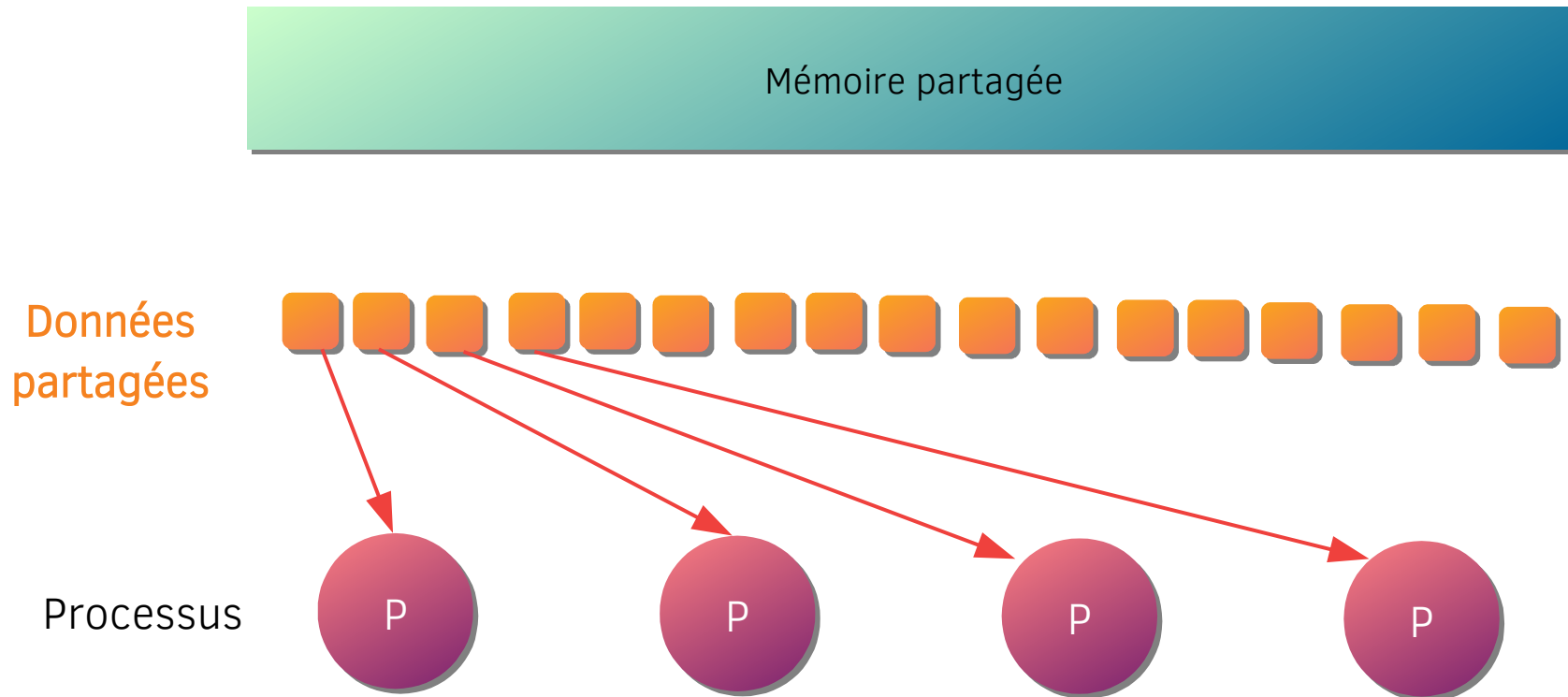
Ce genre de paradigme, bien que **conçu pour un parallélisme distribué**, **fonctionne** parfaitement sur une architecture **mémoire partagée**.

Paradigme *fork-join*



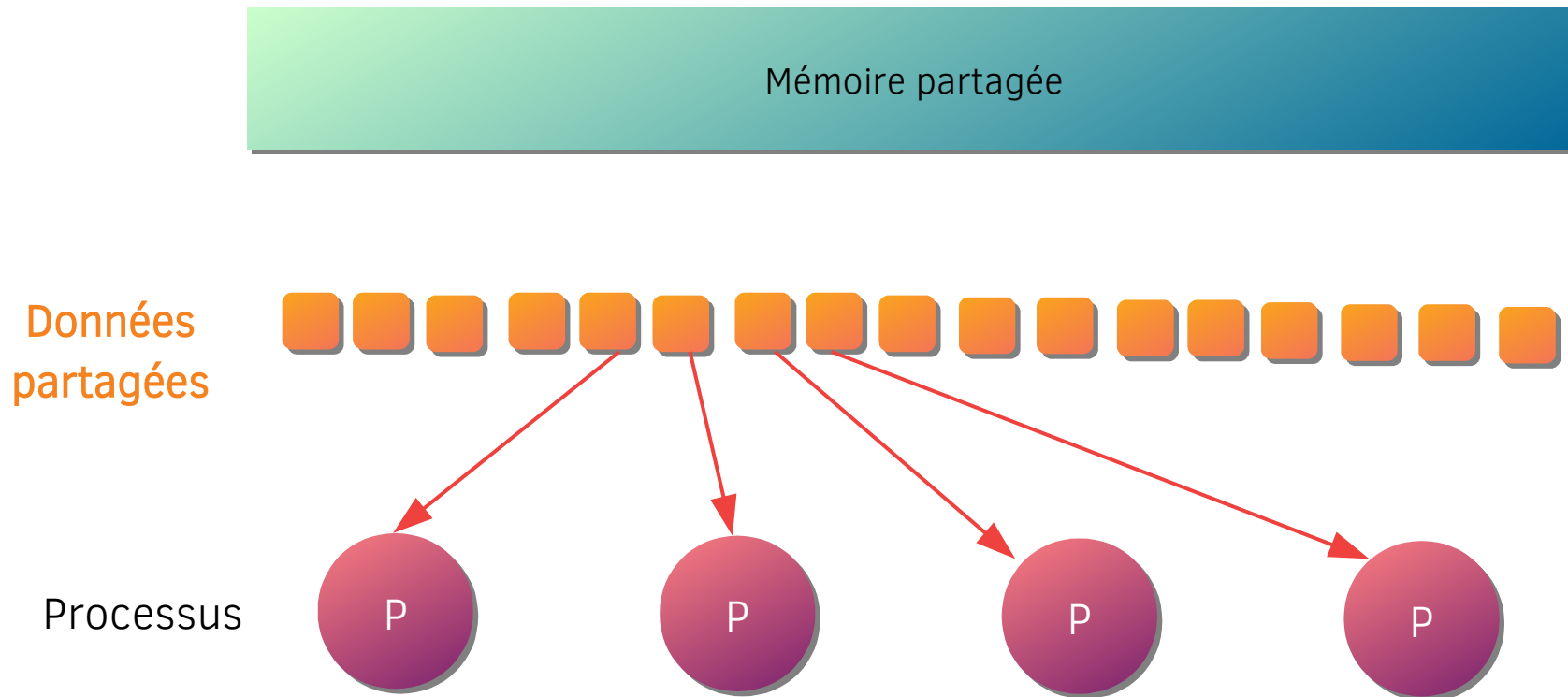
La bibliothèque la plus courante en calcul scientifique est **OpenMP**. Ce modèle ne fonctionne que pour une mémoire partagée.

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` :



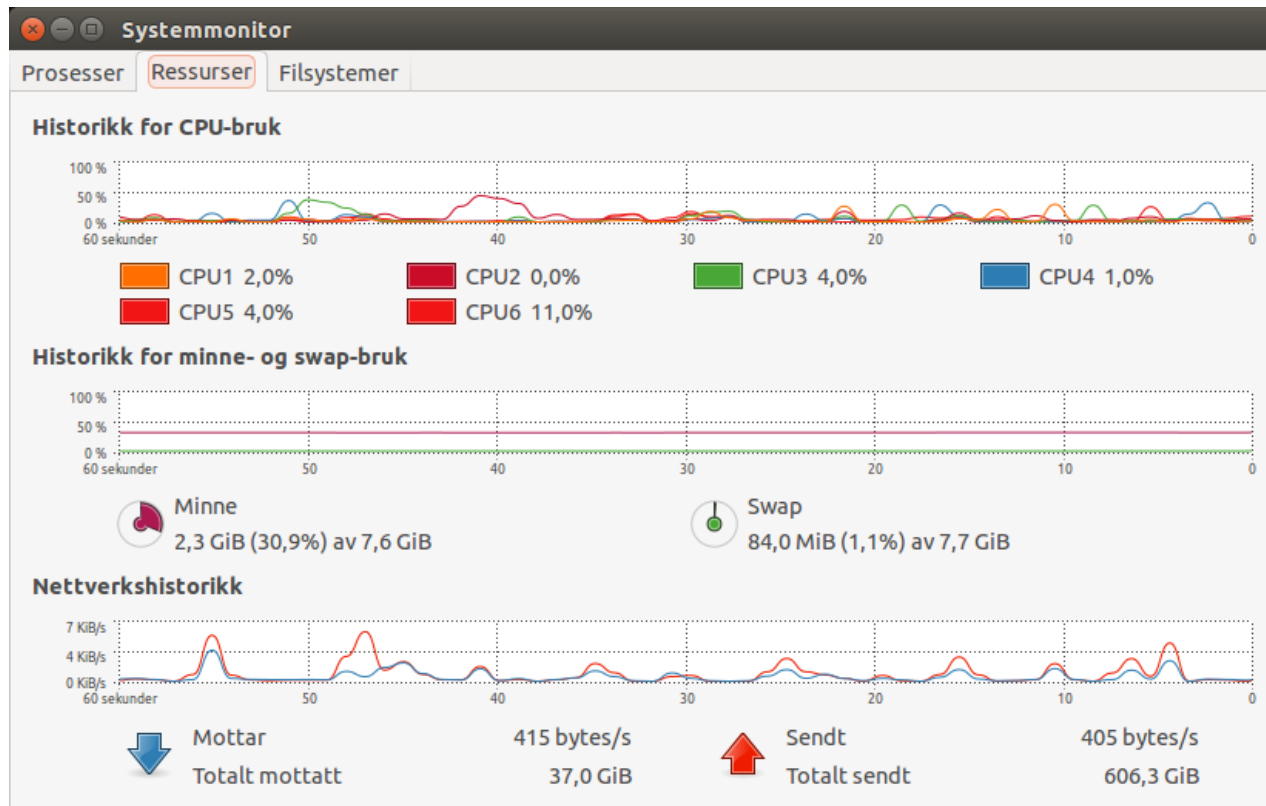
```
> cat /proc/cpuinfo

...
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>