

HPC 2025

Introduction

Pierre Künzli

ISC, HEPIA

Modalités du cours

- Pierre Künzli, pierre.kunzli@hesge.ch
- Michaël El Kharroubi, michael.el-kharroubi@hesge.ch

Plan approximatif

- Généralités sur le HPC et le parallélisme
- Les clusters HPC et leur utilisation (ordonnanceur Slurm)
- Architecture GPU
- Programmation CUDA

SI cluster de test Kubernetes prêt

- Généralités sur clusters Kubernetes
- Projet : développement d'une application mettant en oeuvre Kubernetes, Slurm et / ou CUDA

SINON

- Notions avancées de GPU (multi-GPU, profiling, ...)
- Eventuellement MPI, analyse de performances.
- Projet : développement d'une application CUDA

Au début

- Théorie et TP/TD pour aborder la programmation sur GPU.

Par la suite

- Projet en groupe mettant en oeuvre une application parallèle.
- Technologies à définir.

- Un ou deux tests écrits sur la partie théorique du cours (GPU).
- Evaluation du projet.

Présence aux cours et séances de travail obligatoire (contrôle des présences)

Présence et participation pris en compte dans l'évaluation du projet

Contrepartie : pas (ou peu) de travail à côté des séances de travail.

Le HPC

- **HPC pour *High Performance Computing*** ou *Calul Haute Performance*.
- **Utilisation de *supercalulateurs*** ou *clusters de calcul* pour effectuer des calculs plus rapidement, ou sur de plus gros volumes de données, que sur un ordinateur classique.
- **Modèle d'exécution *batch processing*** ou *traitement par lot*
 - Exécution automatique d'une suite de commande,
 - méthode de traitement asynchrone,
 - par d'interaction avec l'utilisateur pendant le traitement du lot.

Les supercalculateurs

- **Supercalculateurs** ou **clusters de calcul**.
- **Système multi-utilisateurs**.
- **Architecture composée de processeurs interconnectés** (ou d'ordinateurs interconnectés, les noeuds).
- **Grande puissance de calcul**.
- **Réseau efficace** (couplage fort).

Quelques exemples de supercalculateurs

Les supercalculateurs

Leur performance se mesure en nombre d'opérations en virgule flottante par secondes, ou **FLOPS** (FLOating-Point Operations per Second).

Le classement des 500 machines (connues) les plus puissantes :

<http://www.top500.org/>

Frontier, 2021, USA



- **Classement** : n°1 du top 500
- **Prix** : US\$600 million, soit environ 0.3% de la fortune de Bernard Arnault
- **Puissance électrique** : 22.7 MW, soit environ 2 TGV (moteurs de 8800 kW)
- **Puissance de calcul** 2 exaFLOPS (2000 petaFLOPS, max théorique)

Architecture

- 9,472 noeuds AMD Epyc 7A53s “Trento” 64 core 2 GHz CPUs (606,208 cores)
- 512 GB RAM par noeud
- 37,888 Radeon Instinct MI250X GPUs (8,335,360 cores)
- **Stockage** : 75 TB flash + 700 PB
- **Réseau** : environ 100 GB/s

Piz Daint, 2012-2013-2016-2018, Suisse



Remplacé en 2024 Par *Alps*.

Piz Daint, 2012-2013-2016-2018, Suisse

- **Classement** : n°37 au top 500
- **Architecture hybride** : 7'133 Intel Xeon E5 12/18-core CPUs & 5'320 Nvidia Tesla P100
- **Mémoire** : XC50 : 64 GB (CPU) et 16 GB (GPU), XC40 : 64/128 GB (CPU)
- **Puissance électrique** : 2.3 MW, Stockage : ?
- **Puissance de calcul** : 27 petaFLOPS (max théorique)
- **Interconnexion** : Aries Dragonfly topology
- **Prix** : ? + 40 millions (mise-à-jour d'octobre 2016)

FLOPS

Floating-Point Operation Per Second

Computer performance

Name	Unit	Value
kiloFLOPS	kFLOPS	10^3
megaFLOPS	MFLOPS	10^6
gigaFLOPS	GFLOPS	10^9
teraFLOPS	TFLOPS	10^{12}
petaFLOPS	PFLOPS	10^{15}
exaFLOPS	EFLOPS	10^{18}
zettaFLOPS	ZFLOPS	10^{21}
yottaFLOPS	YFLOPS	10^{24}

Core i7-13700K 1,3 TFLOPS, max théorique

Nvidia RTX 4090 80 TFLOPS, max théorique

Où les trouver ?

- La plupart dans des centres de recherches publics.
- Aussi dans l'industrie
 - *Eagle* chez Microsoft (Azure),
 - *Discovery5* chez ExxonMobil,
 - *DeepL Mercury* chez DeepL, ...

Pour faire quoi ?

Les superordinateurs jouent un rôle important dans les sciences computationnelles, tel que

- La modélisation de l'atmosphère, p.ex. la météorologie, climatologie.
- La modélisation moléculaire, p.ex. dynamique moléculaire, étude des structures des polymères/cristaux.
- En ingénierie dans les domaines de l'aéronautique, de la physique nucléaire.
- Entraînement de modèles en IA.

Ressources de calcul HPC des hautes écoles genevoises

- Baobab, 2013
- Yggdrasil, 2021
- Bamboo, 2024

Baobab et Yggdrasil

Baobab, mis en service en 2013 à Uni Dufour.

- 900 coeurs publics, 4200 coeurs au total,
- 273 GPUs,
- réseau 100GB Infiniband.

Yggdrasil, mis en service en 2021 à l'observatoire de Genève, Versoix.

- 3000 coeurs publics, 4300 coeurs au total,
- 52 GPUs,
- réseau 100GB Infiniband.

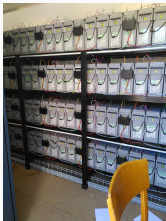
Bamboo, mis en service en 2024 au campus Biotech.

- 5700 coeurs publics,
- 20 GPUs,
- réseau 100GB Infiniband.

Baobab, 2013, Uni Dufour



Yggdrasil, 2021, Observatoire de Genève, Versoix



Point pratique

- Votre compte doit être activé sur les clusters,
- vous avez reçu un mail avant Noël pour le faire,
- vous devez transmettre votre clé publique pour utiliser SSH.

Si ce point n'est pas encore réglé, on doit s'en occuper aujourd'hui

Architecture parallèle

- Les clusters sont des architectures *massivement parallèles*.
- Plusieurs milliers, voir millions, d'unités de calcul peuvent travailler en même temps sur un même problème.
- Pourquoi adopter une architecture parallèle ?

La loi de Moore

Accélérer la vitesse de traitement

Peut-on compter sur le progrès technologique des processeurs pour accélérer une application ?

La loi de Moore

En 1965 G. E. Moore, l'un des fondateurs d'Intel, énonce la loi suivante :

“la complexité des semi-conducteurs d'entrée de gamme double tous les ans à coût constant”

Cette loi, ou plutôt axiome, a ensuite été étendue à :

“le nombre de transistors sur une puce de silicium double tous les deux ans”

La loi de Moore (généralisation)

Cette loi s'est ensuite généralisée à tout ce qui touche une puce de silicium : capacité, fréquence d'horloge, vitesse, etc.

"le/la X d'une puce de silicium double tous les deux ans"

Pour faire simple, on a donc des machines de moins en moins coûteuses et de plus en plus puissantes.

La loi de Moore (généralisation)

Cette loi s'est ensuite généralisée à tout ce qui touche une puce de silicium : capacité, fréquence d'horloge, vitesse, etc.

“le/la X d'une puce de silicium double tous les deux ans”

Pour faire simple, on a donc des machines de moins en moins coûteuses et de plus en plus puissantes.

Nous avons plus de 50 ans d'historique, est-ce vrai ?

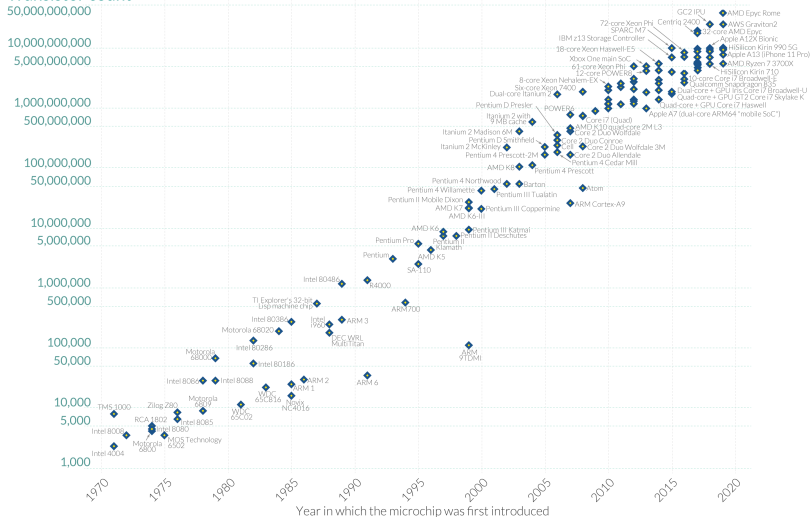
Evolution du nombre de transistors par processeur

Moore's Law: The number of transistors on microchips has doubled every two years

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

Our World
in Data

Transistor count



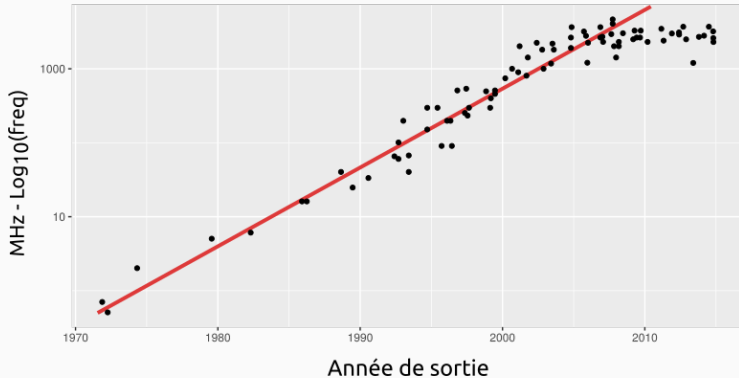
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

OurWorldinData.org – Research and data to make progress against the world's largest problems.

Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

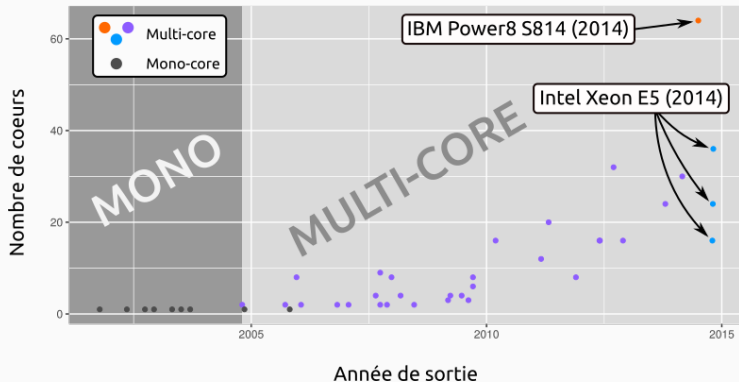
Evolution de la fréquence d'horloge des processeurs

Fréquence d'horloge des processeurs



Evolution du nombre de coeurs par processeur

Nombre de coeurs des processeurs



La loi de Moore arrive-t-elle à sa fin ?

Cette croissance ne peut pas durer indéfiniment, ceci est également admis par Gordon Moore.

On s'approche de limites physiques dans la réalisation des processeurs :

- la dissipation thermique qui augmente avec l'augmentation de la fréquence d'horloge
- l'effet tunnel (*quantum tunneling*) pourrait être un sérieux obstacle dans la miniaturisation des transistors
- ... (demander à un physicien)

Accélérer la vitesse de traitement

Peut-on compter sur le progrès technologique des processeurs pour accélérer une application ?

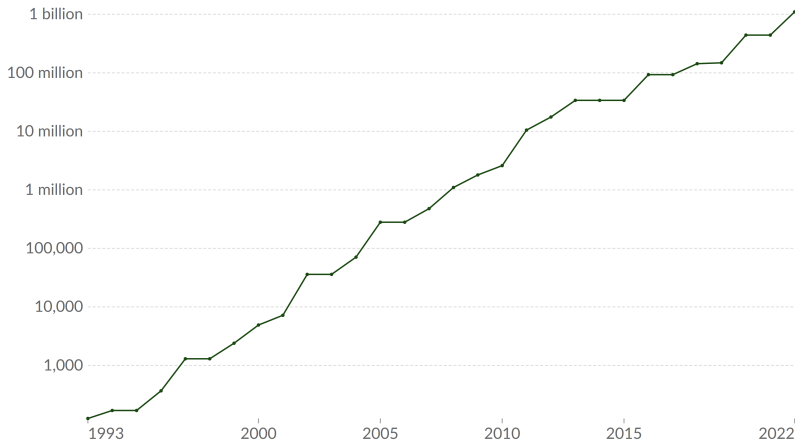
Oui, mais il faut *paralléliser* le code.

Evolution de la puissance des plus puissants superordinateurs

Computational capacity of the fastest supercomputers

Our World
in Data

The number of floating-point operations¹ carried out per second by the fastest supercomputer in any given year. This is expressed in gigaFLOPS, equivalent to 10^9 floating-point operations per second.



Data source: TOP500 Supercomputer Database (2023)

OurWorldInData.org/technological-change | CC BY

1. Floating-point operation: A floating-point operation (FLOP) is a type of computer operation. One FLOP is equivalent to one addition, subtraction, multiplication, or division of two decimal numbers.

Notion de parallélisme

Une définition du parallélisme

Le parallélisme est l'action de découper un problème en plus petites tâches, et d'exécuter plusieurs tâches en même temps (en parallèle) au lieu de les accomplir séquentiellement.

Le but étant d'accélérer le traitement du problème, ou de pouvoir traiter des problèmes plus grands.

La recette de cuisine

Une (hypothétique) recette de cuisine nécessite de :

- couper des tomates
- peler des patates
- cuire des champignons

La recette de cuisine

Pour la réaliser, on peut :

- engager un commis de cuisine qui accomplira les trois étapes
- engager trois commis de cuisines et chacun accomplira une étape

Pour la première approche on parle de réalisation séquentielle des tâches, dans la deuxième on parle d'approche parallèle.

On espère pouvoir terminer la recette plus vite avec plus de commis.

Un travail d'équipe

En supposant que les trois commis sont aussi expérimentés les uns que les autres, quel gain, quel **speedup** (accélération) avons-nous obtenus ?

- Si chaque étape de la recette prend le même temps, on peut espérer que cette dernière sera terminée trois fois plus vite.
- Si les étapes ne prennent pas le même temps, on attendra donc sur le commis ayant l'étape la plus longue.

Ce dernier point peut pénaliser les performance d'une application parallèle et on parle de problème de **balance de charge**.

Problèmes à traiter

- Algorithmique : répartition du travail
- Architecture : interconnexion
- Langage de programmation
- Système d'exploitation : gestion, placement, ordonnancement des tâches

Architectures de machines parallèles

Système distribué

- Architecture composée de processeurs interconnectés
- Utilisation : résolution simultanée de plusieurs problèmes liés
- Succès croissant des architectures distribuées avec la mise en réseau systématique des ordinateurs actuels
- Cloud, cloud hybride, cluster K8S distribué

Ordinateur parallèle

- Machine composée de processeurs **fortement couplés**
- Utilisation : résolution simultanée et concurrente d'un même problème
- Clusters HPC, supercalculateurs

Cloud IaaS vs cluster HPC

▪ Cloud

- mise à disposition de ressources pour traitement de tâches qui peuvent être interactives
- en principe, “toujours” des ressources disponibles
- ressources faiblement couplées

▪ Cluster HPC

- ressources disponibles pour traitement de tâches asynchrones (non interactives)
- objectif de “remplir au maximum” (rentabiliser) les ressources disponibles
- unités de calcul fortement couplées

Ce qui distingue le cluster HPC du reste : le couplage fort des unités de calcul par le réseau.

Les technologies d'orchestration modernes peuvent “brouiller les pistes” : par exemple cluster HPC slurm intégré dans un cluster K8S distribué.

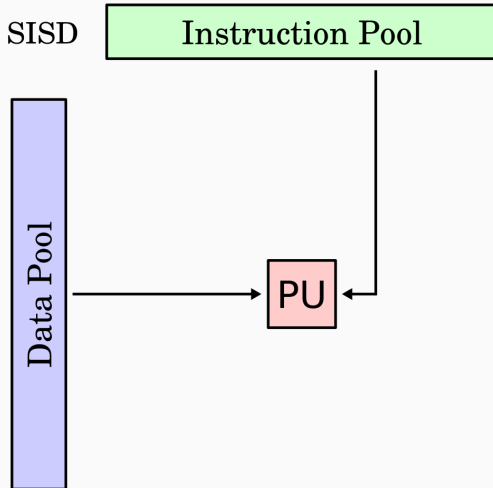
La taxonomie de Flynn

La taxonomie proposée par Michael Flynn en 1966 classe les architectures d'ordinateur.

Les classes sont basées sur le **flux de données** et le **flux d'instructions** :

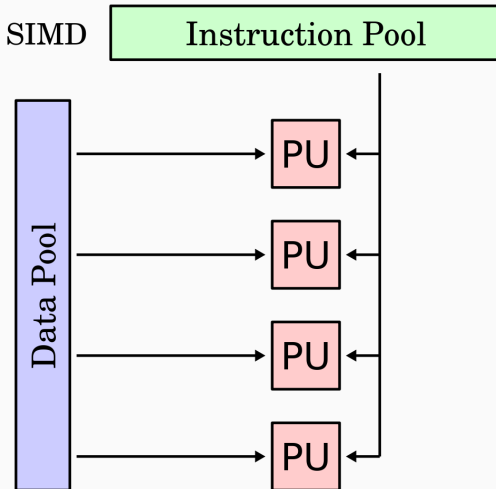
- SISD - Single Instruction Single Data : unique flux d'instructions, unique flux de données
- SIMD - Single Instruction Multiple Data : unique flux d'instructions, multiples flux de données
- MISD - Multiple Instruction Single Data : multiples flux d'instructions, unique flux de données
- MIMD - Multiple Instruction Multiple Data : multiples flux d'instructions, multiples flux de données

Single Instruction Single Data



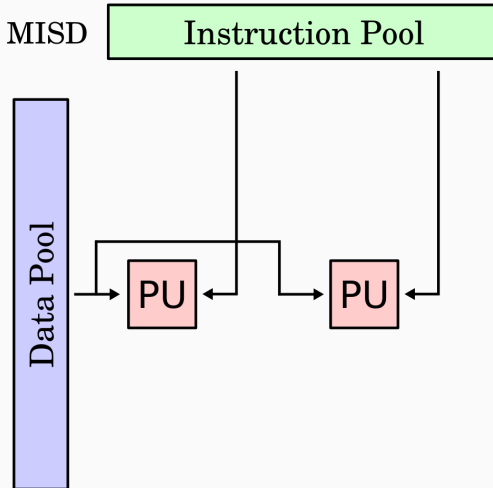
Exemple : CPU monocoeur

Single Instruction Multiple Data



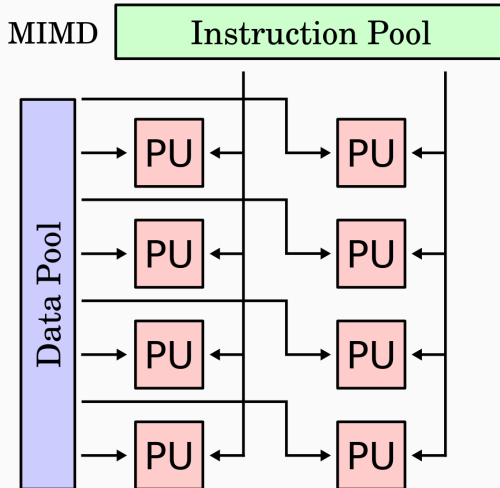
Exemple : GPU

Multiple Instruction Single Data



Exemple : pipeline de traitement ?

Multiple Instruction Multiple Data



Exemple : CPU multicœur, système multiprocesseur

Deux types d'architecture mémoire :

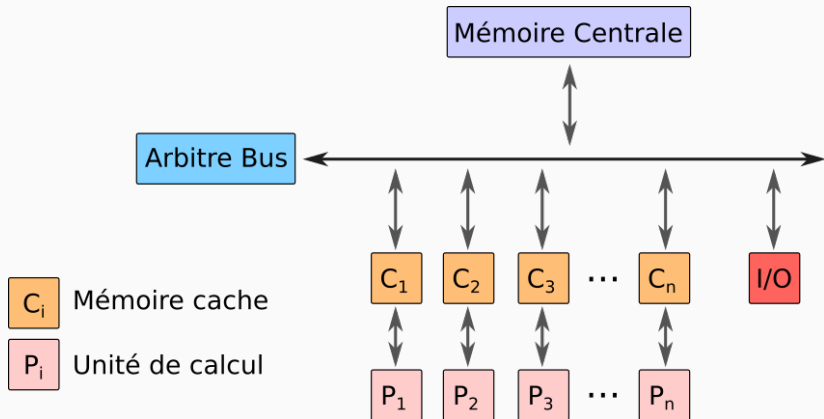
- Mémoire partagée
- Mémoire distribuée

Architecture MIMD à mémoire partagée

Architecture MIMD à mémoire partagée :

- Processeurs synchronisés
- Gestion des conflits d'accès par l'OS
- Programmation conventionnelle
- Exemple : ordinateur multicoeur

Architecture MIMD à mémoire partagée



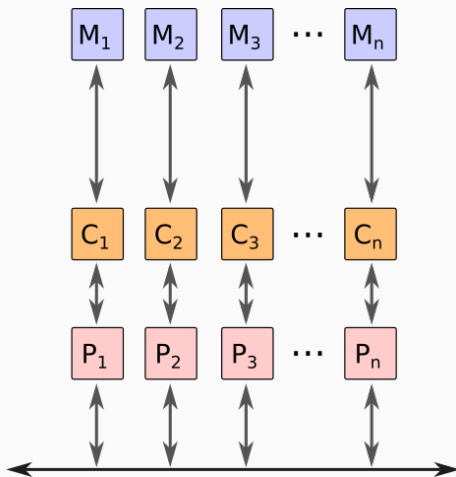
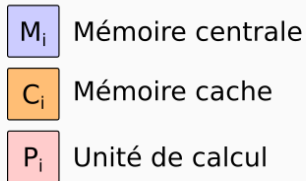
Architecture SIMD à mémoire partagée :

- Processeurs synchronisés
- Gestion des conflits d'accès le runtime / framework
- Framework spécifique
- Exemple : GPU

Architecture MIMD à mémoire distribuée :

- Processeurs non-synchronisés
- Communication par échange de message
- Gestion des échanges de données par le programmeur
- Extensibilité du système par ajout de modules processeur-mémoire
- Exemple : cluster HPC, chaque module (noeud) peut être un ordinateur MIMD ou SIMD à mémoire partagée (cluster de CPUs ou de GPUs)

Architecture MIMD à mémoire distribuée



Les types de parallélisme

Deux types de parallélisme

On peut distinguer deux types de parallélisme

- Le parallélisme de contrôle ou *task parallelism*
- Le parallélisme de données ou *data parallelism*

Le parallélisme de contrôle

Le parallélisme de contrôle

- Les unités de calcul exécutent différentes tâches
- Les tâches sont exécutées de manière concurrentes sur les différentes unités de calcul

Le parallélisme de données

Le parallélisme de données

- Distribue les données sur différentes unités de calcul
- Les données sont traitées en parallèle
- La même opération ou tâche est appliquée sur les données sur chaque unité de calcul

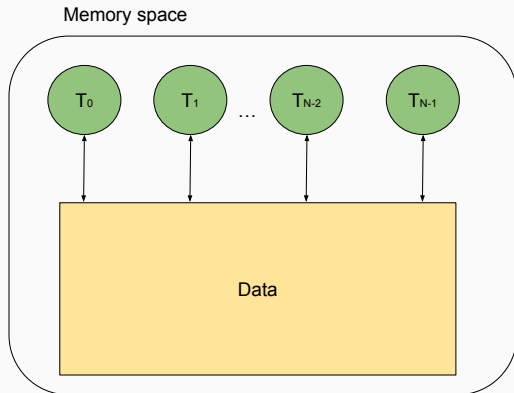
Le parallélisme de données s'applique bien aux applications scientifiques où l'on travaille avec des structures de données régulières (tableaux/vecteurs ou matrices).

Modèles de programmation

Comme pour les architectures :

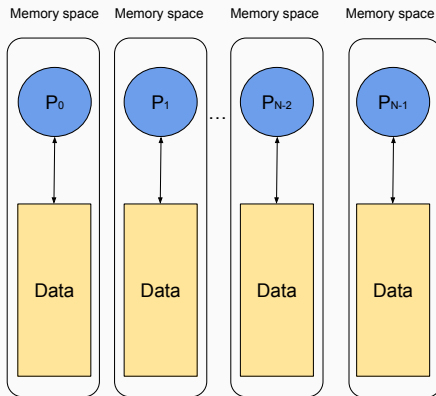
- Mémoire partagée
- Mémoire distribuée

Mémoire partagée



- Les flux d'exécution sont des threads
- Echange de données entre threads implicite via la mémoire
- Risque d'accès concurrent
- Nécessite des mécanismes de contrôle d'accès à la mémoire (verrou, sémaphore, synchronisation, ...)
- Possible uniquement sur une architecture physique à mémoire partagée

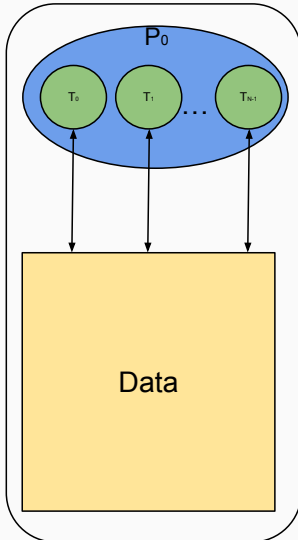
Mémoire distribuée



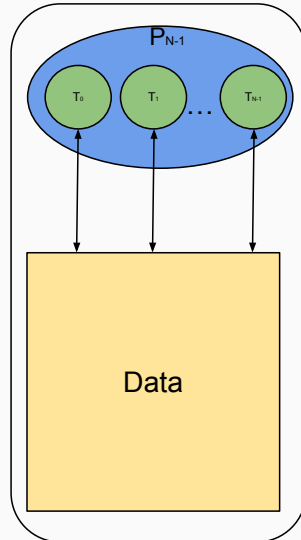
- Les flux d'exécution sont des processus
- Echange de données entre processus explicite via échange de messages
- Pas de problème de concurrence
- Possible sur une architecture physique à mémoire partagée ou à mémoire distribuée (CPU)

Modèle hybride possible

Memory space



Memory space



...

Programmation sur GPU

Architecture SIMD à mémoire partagée.

En multi-GPU sur plusieurs noeuds : mémoire distribuée.