Architecture et programmation des calculateurs scientifiques parallèles

François-Xavier Roux
Unité Calcul à Haute Performance

ONERA
THE FRENCH AEROSPACE LAB

retour sur innovation

Plan

- Parallélisme
- Mémoire multi-banc entrelacée
- Mémoire hiérarchique
- Mémoire distribuée
- Programmation « data parallel »: Open MP
- Programmation distribuée: MPI
- Echanges point à point
- Echanges globaux
- Communicateurs
- Conclusion

Tendances actuelles

- En 1965, Gordon Moore (un des fondateurs d'Intel), déclare que la puissance est multipliée par 2 tous les 18 mois
- Limitations physiques : plus la fréquence du processeur est élevée, plus il chauffe. Le dégagement de chaleur important d'un processeur est donc un obstacle pour l'augmentation de sa fréquence
- Aujourd'hui la « Loi de Moore » est toujours vérifiée, mais non plus par la montée en fréquence, mais par la multiplication des cœurs
- Le principal moteur du développement technologique actuellement : les jeux et la vidéo 3D (exemple : cartes graphiques Nvidia, plusieurs centaines d'unités de calcul)

Architecture parallèle

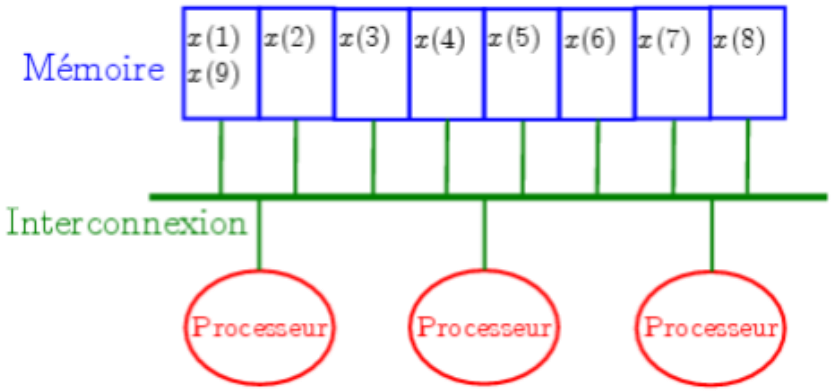
- Principe: multiplication des unités fonctionnelles arithmétiques et/ou des processeurs, pour augmenter la puissance de calcul
- Fonctionnement simultané (concurrent) pour une même application
- Beaucoup d'opérations arithmétiques sur des nombres réels par seconde (Floating point operation per second: FLOPS)
- Le seul vrai problème: comment alimenter les unités en données?
- Plus une mémoire est grosse, plus le temps d'accès à une donnée est élevé

Top 10 novembre 2019

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**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	148,600.0	200,794.9	10,096
2	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
3	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**Frontera** - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR , Dell EMC Texas Advanced Computing Center/Univ. of Texas United States	448,448	23,516.4	38,745.9	
6	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray/HPE Swiss National Supercomputing Centre (CSCS) Switzerland	387,872	21,230.0	27,154.3	2,384

Mémoire multi-banc entrelacée

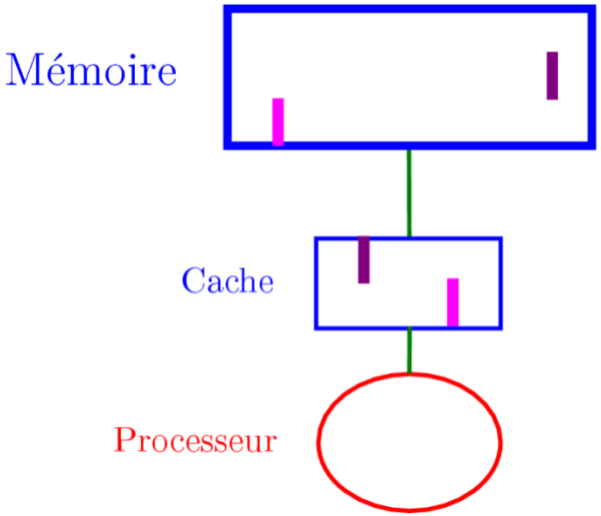
- Plusieurs unités mémoires travaillant simultanément pour fournir des séries de données : vecteurs

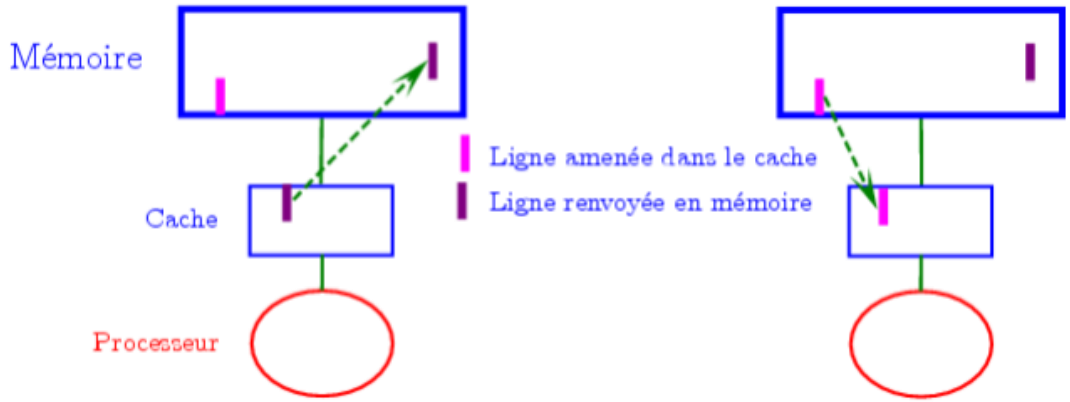


- Complexité quadratique du contrôleur mémoire et du réseau d'interconnexion
- Solution pas extensible: le système deux fois plus gros coûte plus de deux fois plus cher et n'est pas forcément réalisable

Mémoire cache

- Petite mémoire rapide proche du processeur : mémoire cache
- Mécanisme optimisé de transfert par paquet de données contigües : les « lignes de cache »

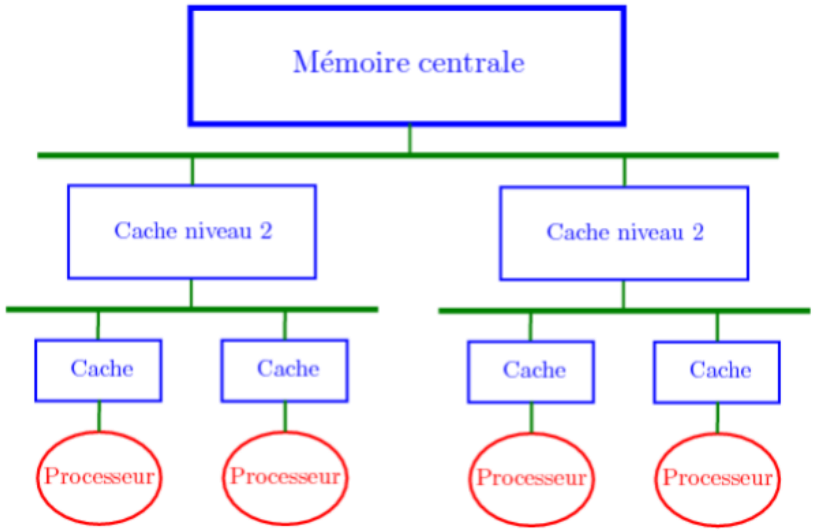




- Nécessité de favoriser les accès à des données contigües en mémoire
- Réutiliser les données présentes dans le cache
- Localisation temporelle et spatiale des données

Mémoire hiérarchique: architecture SMP

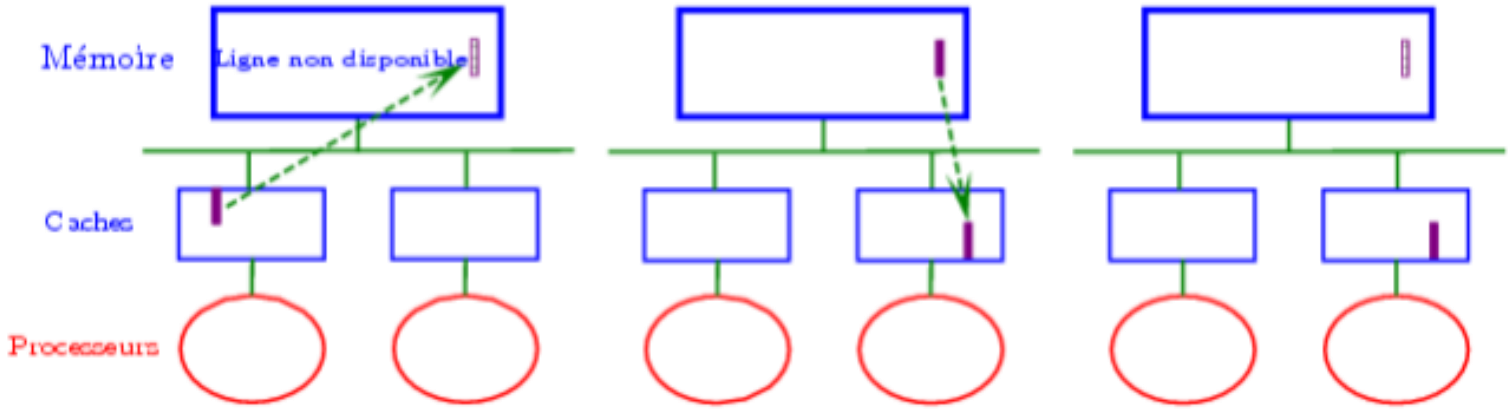
- Multiplication des caches
- Caches multi-niveau pour augmenter le nombre de processeurs



- Complexification de l'architecture avec l'accroissement du nombre de niveaux
- Solution pas complètement extensible

Conflits d'accès à la mémoire

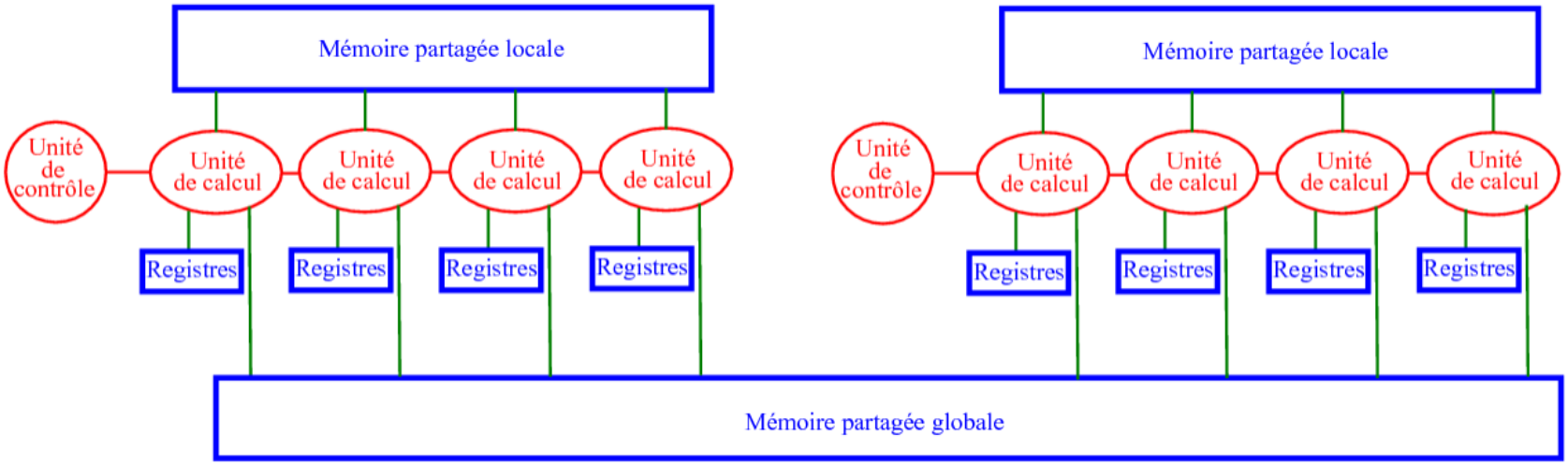
- Conflits matériels: la mémoire centrale ne peut alimenter simultanément plusieurs caches
- Conflits logiques: une ligne ne doit être présente que dans un seul cache pour assurer la cohérence des données



- Les processeurs doivent travailler sur des paquets de données distincts
- Localisation temporelle et spatiale des données encore plus indispensable

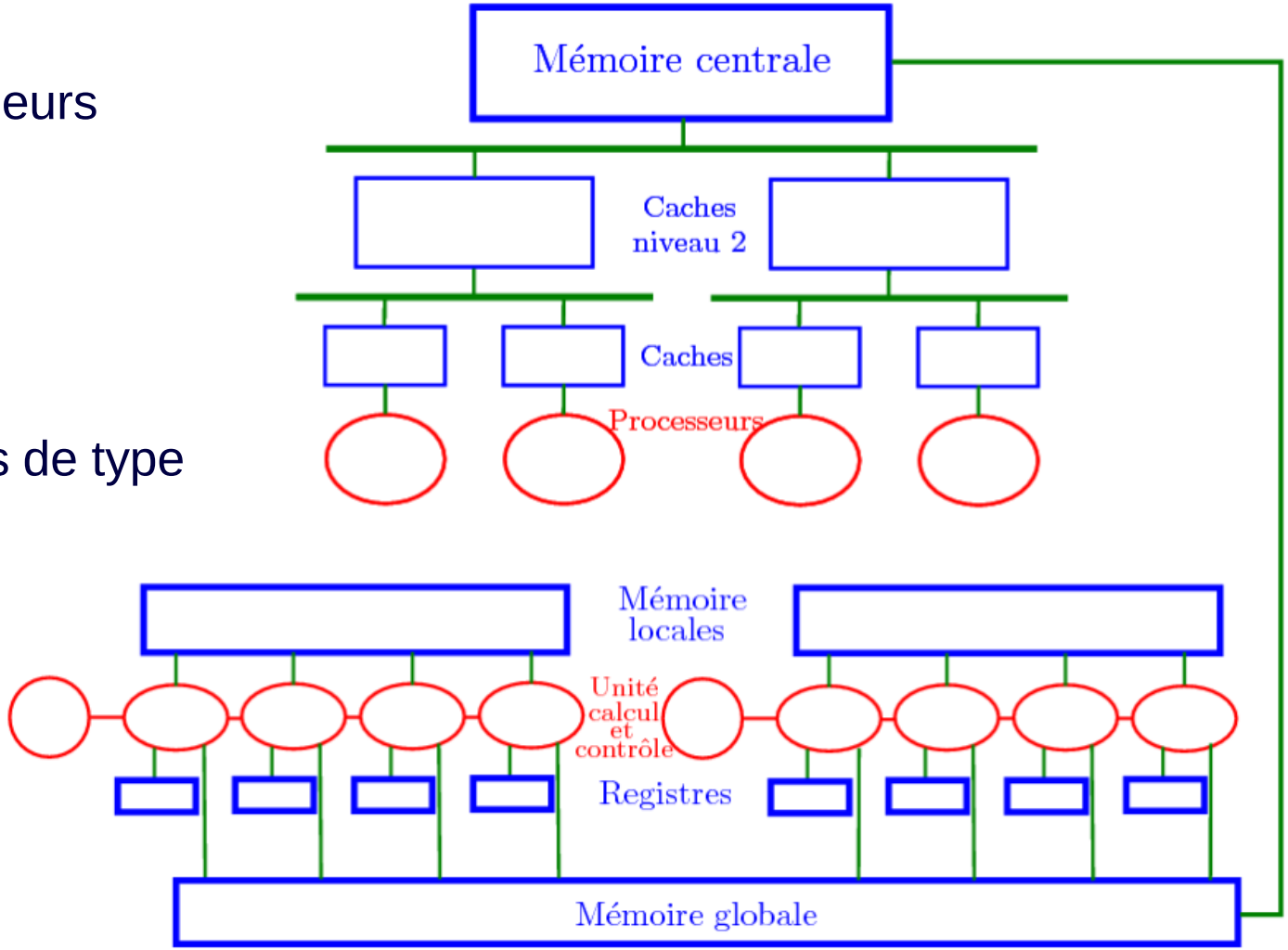
GPU

- Grand nombre d'unités arithmétiques effectuant les mêmes instructions (programmation SIMD, similaire au vectoriel)
- Mémoire centrale de la carte multi-banc, transferts par bus PCI
- Mémoire locale partagée par des groupes d'unités (blocs) et registres des unités affectables explicitement dans le code CUDA, pas de rémanence
- Nécessité d'un alignement correct des données et d'accès sans conflits au bancs de la mémoire partagée globale



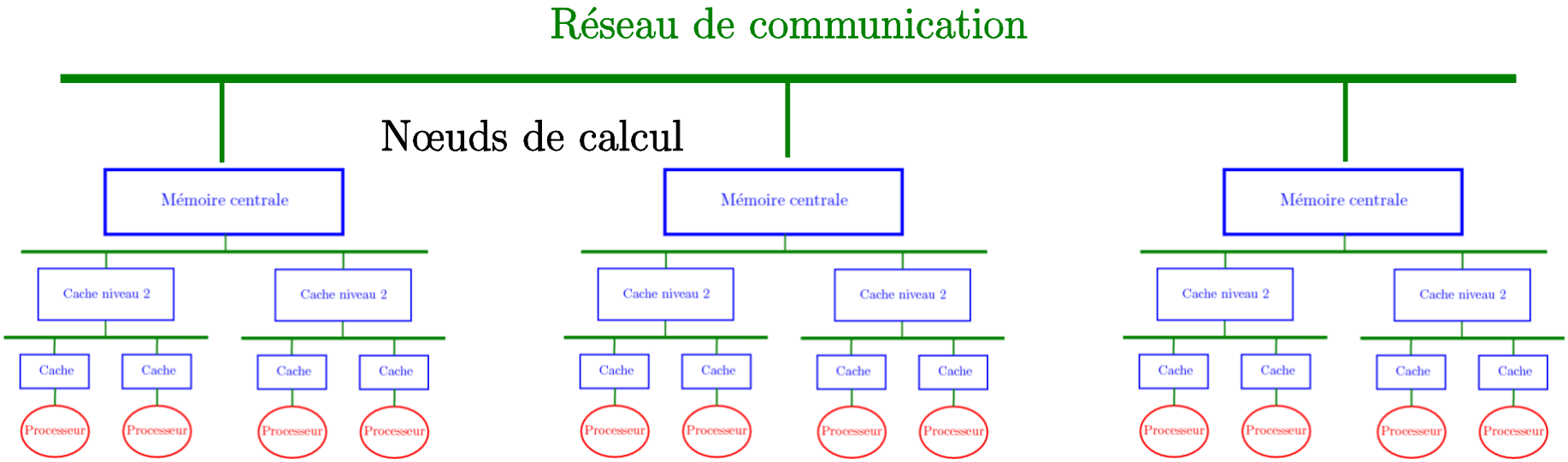
Nœud de calcul hybride

- Multi-processeurs
- Multi-cœurs
- Accélérateurs de type GPU



Suppression de la mémoire partagée: mémoire distribuée

- Un « nœud » de calcul = un ou plusieurs processeurs à mémoire hiérarchique
- Réseau d'interconnexion



- Utilisation de technologies standard: coût réduit
- Extensibilité théoriquement infinie
- Plus de vision unique de la machine: nouvelle conception logicielle

Réseau

- Réseau standard (Ethernet…)
- Réseaux spécialisés (Quadrics, Infiniband…)

- Temps de transfert de n données:

$$T(n) = T_0 + n/C_T$$

- C_T vitesse de transfert, quelques Giga/s
- T_0 temps de latence (start-up time), quelques centaines de µs pour les réseaux standard, de l'ordre de la µs pour les réseaux spécialisés

- Haut débit facile à obtenir : il suffit de multiplier les câbles
- Faible latence : technologiquement difficile, coût élevé

- Les transferts courts coûtent très cher

Parallélisation efficace

- Paralléliser les codes : les rendre exécutables sur des machines parallèles
- Efficacité : taux d'utilisation « utile » des processeurs

- Degré de parallélisme : nombre de tâches susceptibles d'être exécutées simultanément
- Equilibrage des tâches
- Granularité du parallélisme : volume de chacune des tâches

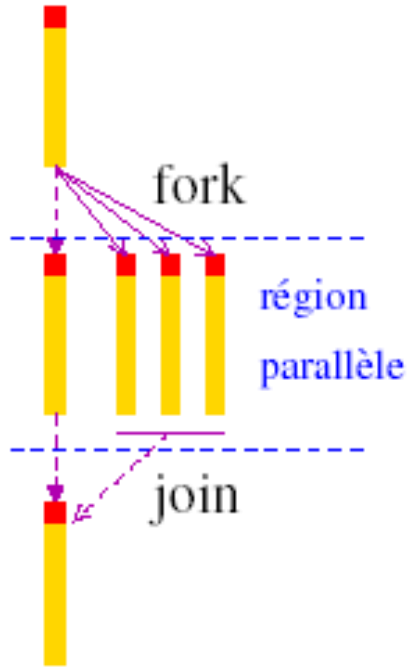
- Sur une machine à mémoire distribuée, la granularité se définit comme le rapport :

  nombre d'opérations arithmétiques / nombre de données transférées

- Granularité suffisante >> C_C/C_T , sachant que $C_C > C_T$

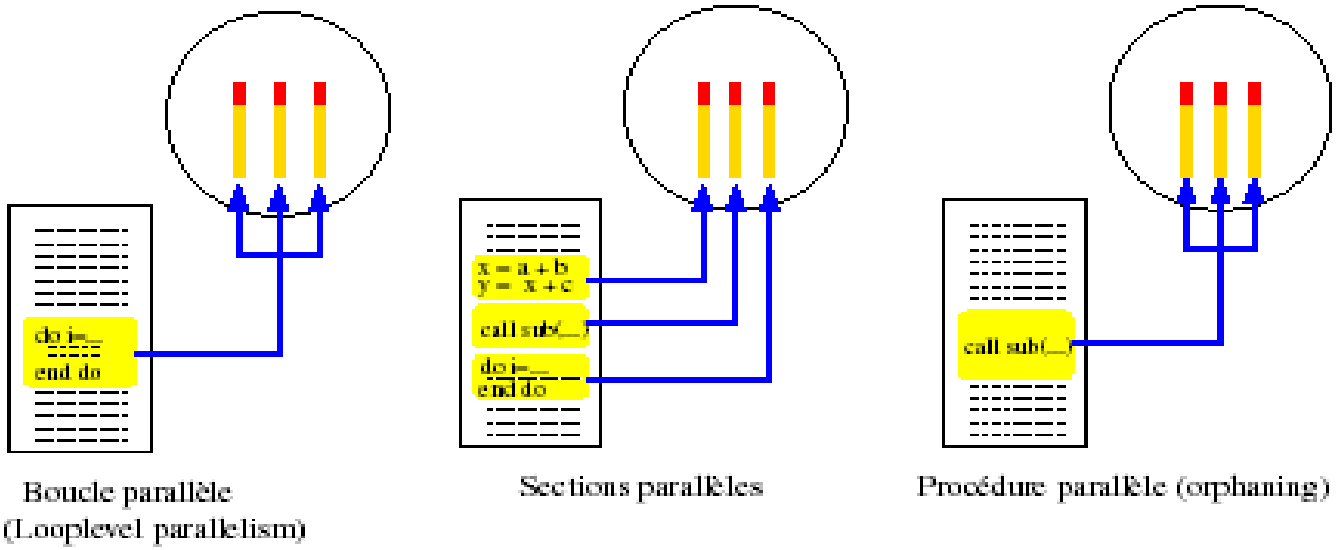
Programmation des machines à mémoire partagée

- Vision classique du code: structures de données, méthodes opérant sur ces structures
- Exécution parallèle des zones de calcul intensif : boucles de programmes imbriquées
- « Parallélisme de données », « Data parallelism »
- Analyse de la dépendance des boucles de programme : parallélisation, permutation



Open MP

- Directives de compilation permettant de définir des boucles parallèles ou des zones parallèles

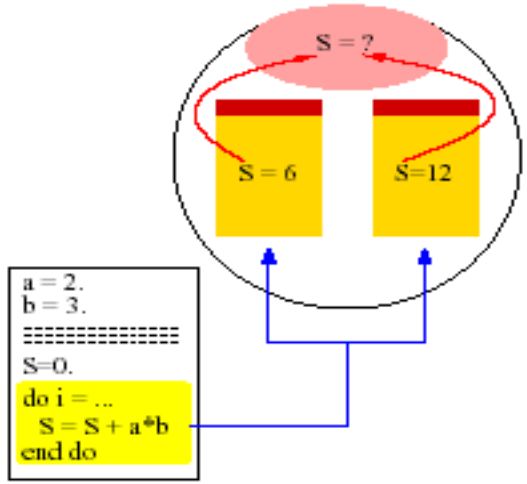


- Bibliothèques optimisées de programmes de type algèbre linéaire (BLAS)

Exemple de programmation Open MP

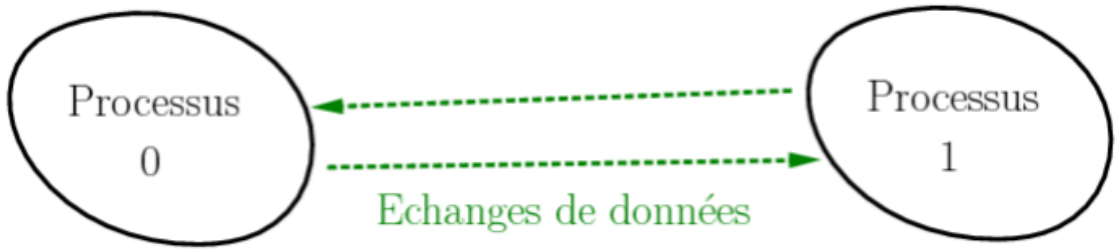
!$OMP PARALLEL DO PRIVATE(i,p), SHARED(n,x,y)
!$OMP& REDUCTION(+: sum)

```
do i = 1, n
   p = x (i) * y(i)
   sum = sum + p
end do
```



Modèle de programmation pour système à mémoire distribuée

- Tâches communicantes



- Chaque tâche a son code (ou au moins son exécutable) et ses données propres
- Programmation des transferts de données dans les codes
- « Programmation par échanges de messages » : bibliothèques de fonctions d'échanges

MPI : Message Passing Interface

- Gestion de l'environnement parallèle
- Echanges point à point
- Echanges collectifs
- Les échanges sont asynchrones : messagerie
- Gestion par MPI de la boîte postale de réception (stockage temporaire)

- Fonctionne sur tout système (multi-tâche), indépendamment du nombre de processeur
- Grappes de stations de travail ou de SMP : « clusters »
- Programmation C++, C, Fortran, interfaces Python, MATLAB…
- Interface très simple
- Validation du code parallèle même sur machine mono-processeur
- Plus efficace sur les systèmes à mémoire hiérarchique (localisation temporelle et spatiale des données)

Transferts de données point à point

- Données typées (tableau)
- Un message = tableau, longueur, type (passage des tableaux par adresse)
- Numéro de processus destinataire (émission) ou source (réception)
- Discrimination par étiquette de message facultative (ordre respecté)
- Echanges dans un « communicateur » : identifiant d'un système de messagerie, par défaut tous les processus

```
MPI_Send (sendbuf,count,MPI_INTEGER,dest,tag,
                                        MPI_COMM_WORLD)
MPI_Recv (recvbuf,count,MPI_INTEGER,source,tag,
                                        MPI_COMM_WORLD,status)
```

- Bordereau de réception (paramètres d'émission) : status

Gestion de l'environnement parallèle

- Numéro de processus, nombre de processus

  MPI_Comm_rank (MPI_COMM_WORLD,rank)

  MPI_Comm_size (MPI_COMM_WORLD,size)

- Création de communicateurs: contexte pour créer un communicateur privé pour une bibliothèque

  MPI_Comm_dup (MPI_COMM_WORLD,NEW_COM)

- Création de communicateurs: groupes pour les échanges collectifs

  MPI_Comm_split (NEW_COM,color,key,GROUP_COM)

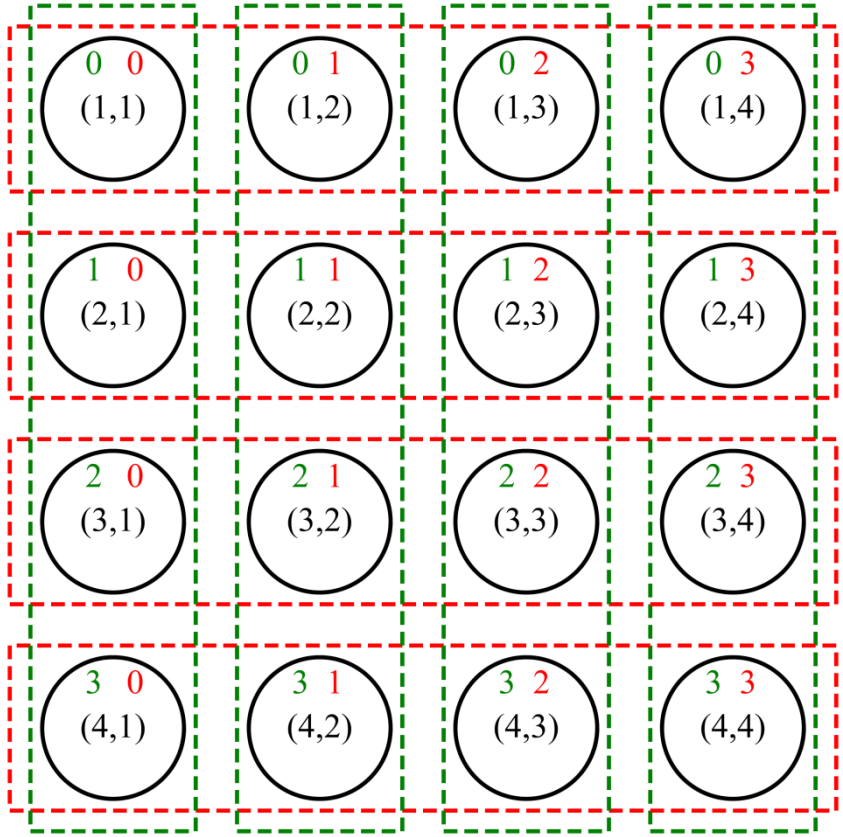
Gestion des communicateurs

- Répartition en grille de blocs d'un tableau à deux dimensions (matrice, image…)
- Création des communicateurs des colonnes de blocs

MPI_Comm_split (COM,j,i,COLUMN)

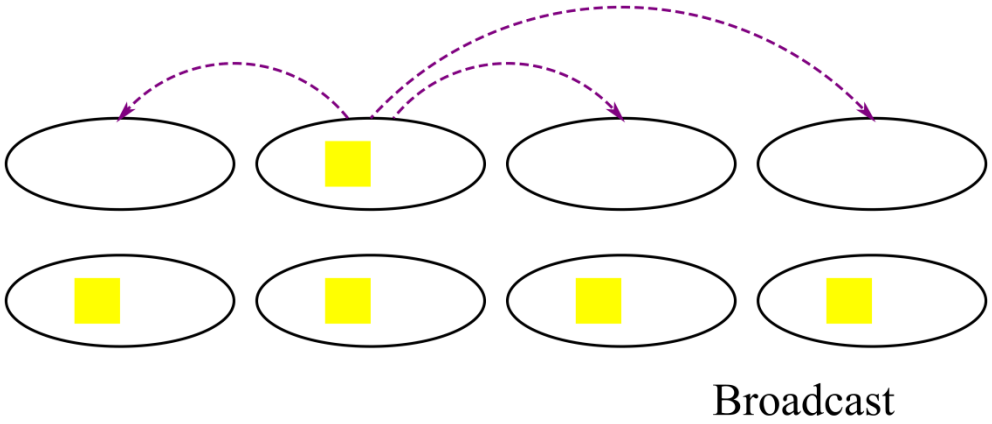
- Création des communicateurs par lignes de blocs

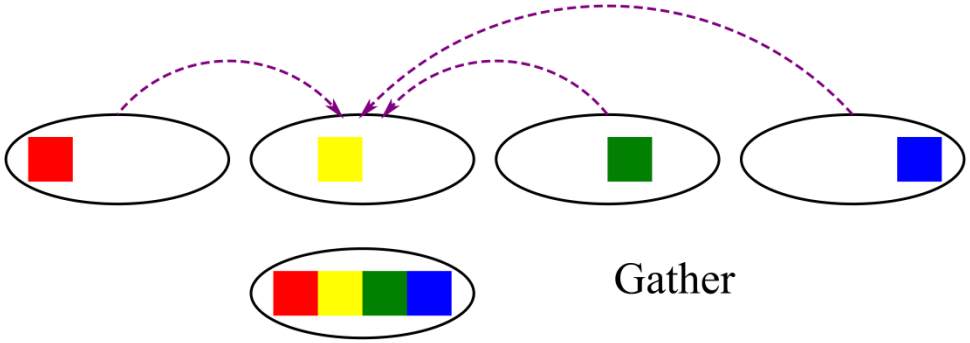
MPI_Comm_split (COM,i,j,ROW)



Communications collectives

- Tous les processus du communicateur participent

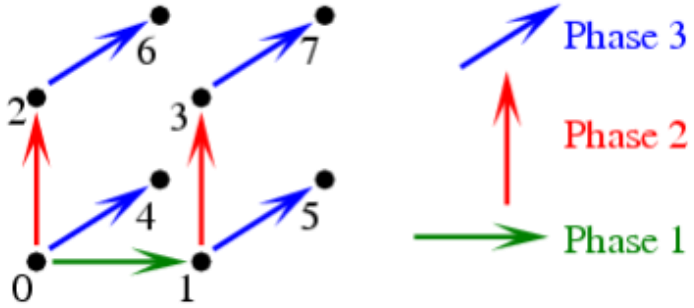


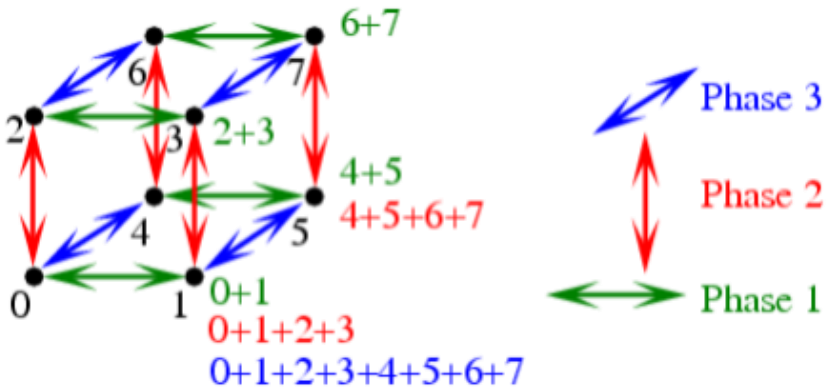


- Opérations de réduction : calcul au vol du produit, de la somme, du Max ou Min
- Résultat directement diffusable à tous

Intérêt des communications collectives

- Plus simple à programmer, moins de risques d'erreurs
- Mise en œuvre d'algorithmes optimisés





- Transfert de un à tous (Broadcast) ou de tous à tous (Allreduce) en Log_2(nombre_de_processus) étapes

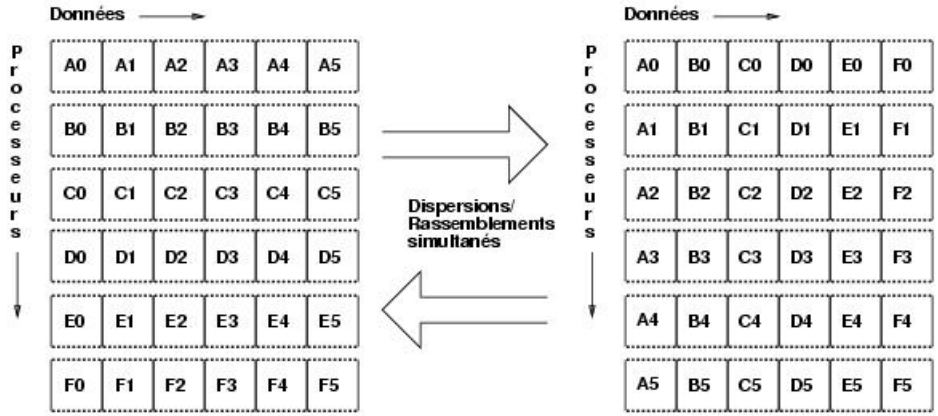
Programmation des échanges collectifs

- Programmation simple, nom des fonctions et des paramètres intuitifs

MPI_Bcast (buf,count,MPI_REAL,root,COM)

MPI_Allreduce (sendbuf,recvbuf,count,MPI_REAL,MPI_MAX,COM)

- Utiliser la fonction adaptée pour améliorer les performances



MPI_Alltoallv (sendbuf,sendcnts,sdispls,MPI_REAL,recvbuf,recvcnts,
rdispls, MPI_REAL,COM)

Conclusion

- Localisation temporelle et spatiale des données
- Localisation temporelle et spatiale des données
- Localisation temporelle et spatiale des données …
- La programmation est simple, c'est la conception qui représente un défi
- Approches générales en simulation numérique : méthodes par blocs, méthodes de résolution par sous-domaines