



SG6: High Performance Computing

## Distributed application deployment with MPI

**Stéphane Vialle**



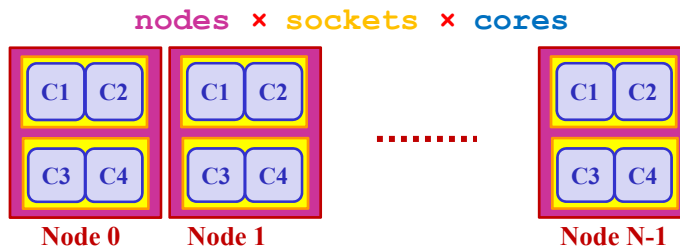
Stephane.Vialle@centralesupelec.fr  
<http://www.metz.supelec.fr/~vialle>

## Distributed application deployment with MPI

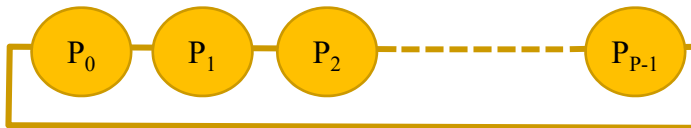
- **Déploiement sur cluster de multicœurs et mécanismes de MPI**
- Exercices
- Modélisation de performances

## Problématique et possibilités de MPI

Ressources matérielles:



Topologie des processus (ex : anneau)

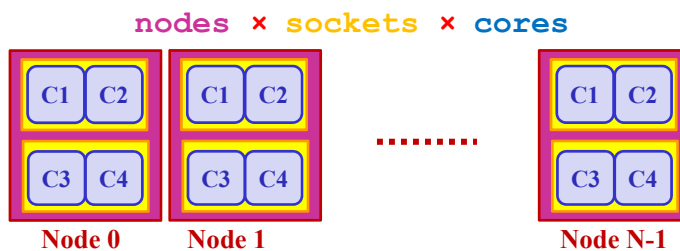


- Quel *mapping* de la topologie virtuelle sur les rsrscs matérielles ?
- Quel *ranking* des processus MPI sur la topologie matérielle ?
- Quel *binding* d'un process sur les rsrscs matérielles locales ?

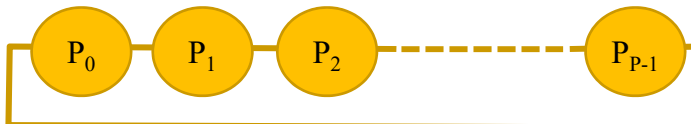
(voir les exemples)

## Problématique et possibilités de MPI

Ressources matérielles:



Topologie des processus (ex : anneau)



```

mpirun -np #P -machinefile machines.txt
    -map-by <node|socket|core...>
    -rank-by <node|socket|core...>
    -bind-to <none|socket|core...>
    --report-bindings .....
  
```

CentraleSupélec

## Déploiement sur cluster de multicœurs et mécanismes de MPI

# Problématique et possibilités de MPI

Ressources matérielles:

nodes × sockets × cores

Topologie des processus (ex : anneau)

Nbr de processus par ressource

```

mpirun -np #P -machinefile machines.txt
        -map-by ppr:N:<node|socket|core...>
        -rank-by <node|socket|core...>
        -bind-to <none|socket|core...>
        --report-bindings .....
  
```

CentraleSupélec

## Déploiement sur cluster de multicœurs et mécanismes de MPI

# Exemple de déploiement: sol 1

**Sol 1 : un processus MPI par nœud**

- Chaque machine présente une fois dans le fichier des machines ciblées
- Nbr de machines listées  $\geq P$  ( $P$  : nbr de processus à créer)
- Utile si un processus nécessite tout le cache ou toute la RAM d'un nœud / d'un socket  
→ Reste un mode « luxueux » !

machines.txt

```

Node 0
Node 1
...
Node P-1
...
Node max
  
```

Node 0    Node 1    Node 2    ...    Node P-1    ...    Node max

CentraleSupélec

Déploiement sur cluster de multicœurs et mécanismes de MPI

## Exemple de déploiement: sol 1

**Sol 1 : un processus MPI par nœud**

- Chaque machine présente une fois dans le fichier des machines ciblées
- Nbr de machines listées  $\geq P$   
(P : nbr de processus à créer)

```
mpirun -np #P -machinefile machines.txt
        -map-by ppr:1:node
        -rank-by node
        -bind-to none|socket .....
```

**machines.txt**

```
Node 0
Node 1
...
Node P-1
...
Node max
```

**Attention : le binding d'un processus peut être limité à un socket...**

CentraleSupélec

Déploiement sur cluster de multicœurs et mécanismes de MPI

## Exemple de déploiement: sol 2

**Sol 2 : un processus MPI par cœur en « mode compact »**

- Nbr de machines nécessaires =  $P / C$   
(P : nbr de processus à créer, C : nb de cœurs/nœud)
- Un maximum de communications se font au sein d'un même nœud : plus efficace!

*Ce déploiement dépend de la topologie virtuelle et de l'architecture réelle.*

**machines.txt**

```
Node 0
Node 1
...
Node P/C-1
...
Node max
```

## Exemple de déploiement: sol 2

Sol 2 : un processus MPI par cœur en « mode compact »

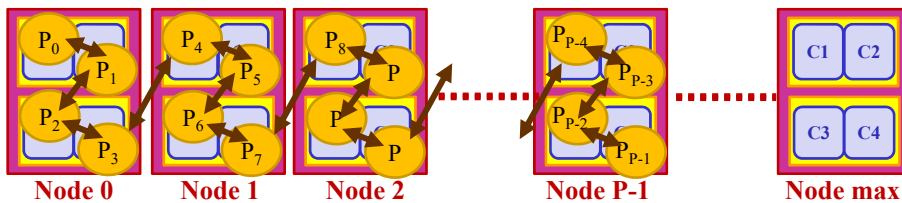
- Nbr de machines nécessaires =  $P / C$   
( $P$  : nbr de processus à créer,  $C$  : nb de cœurs/nœud)

```
mpirun -np #P -machinefile machines.txt
        -map-by ppr:1:core
        -rank-by core
        -bind-to core .....
```

Provoque un déploiement « compact »

machines.txt

```
Node 0
Node 1
...
Node P/C-1
...
Node max
```



## Exemple de déploiement: sol 2

Sol 2 : un processus MPI par cœur en « mode compact »

- Nbr de machines nécessaires =  $P / C$   
( $P$  : nbr de processus à créer,  $C$  : nb de cœurs/nœud)

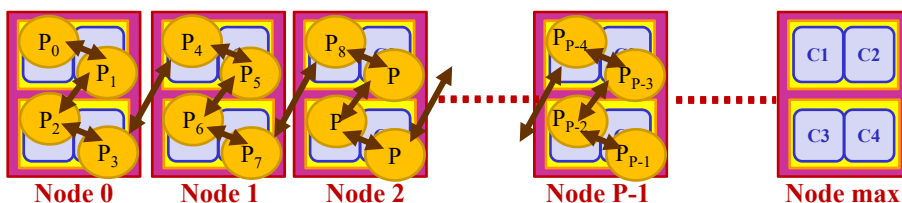
```
mpirun -np #P -machinefile machines.txt
        -map-by ppr:#C:node
        -rank-by core
        -bind-to core .....
```

Provoque un déploiement « compact »

machines.txt

```
Node 0
Node 1
...
Node P/C-1
...
Node max
```

Simple variante : pas d'impact



CentraleSupélec

## Déploiement sur cluster de multicœurs et mécanismes de MPI

### Exemple de déploiement: sol 3

**Sol 3 : un processus MPI par cœur en « round robin »**

- Nbr de machines nécessaires =  $P / C$   
( $P$  : nbr de processus à créer,  $C$  : nb de cœurs/nœud)
- Simple, mais **toutes les communications se font à travers le réseau...** Pas le plus efficace!!

machines.txt

```
Node 0
Node 1
...
Node P/C-1
```

CentraleSupélec

## Déploiement sur cluster de multicœurs et mécanismes de MPI

### Exemple de déploiement: sol 3

**Sol 3 : un processus MPI par cœur en « round robin »**

- Nbr de machines nécessaires =  $P / C$   
( $P$  : nbr de processus à créer,  $C$  : nb de cœurs/nœud)

```
mpirun -np #P -machinefile machines.txt
      -map-by ppr:1:core
      -rank-by node
      -bind-to core .....
```

machines.txt

```
Node 0
Node 1
...
Node P/C-1
```

Provoque un déploiement « round robin »

Le fichier des machines doit avoir exactement la bonne taille

## Exemple de déploiement: sol 4

**Sol 4 : un processus MPI par node et C threads/process**

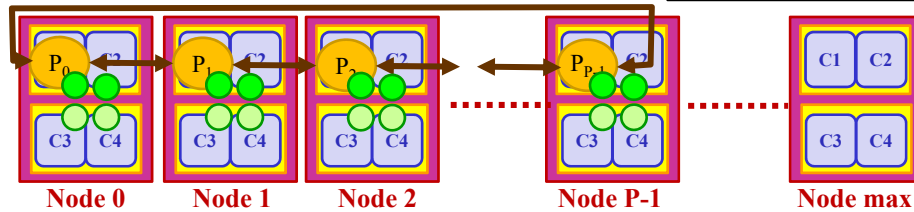
- Nbr de machines listées  $\geq P$   
(P : nbr de processus à créer, C : nb de cœurs/nœud)
- Un processus/nœud, puis chaque processus créera C threads

```
mpirun -np #P -machinefile machines.txt
        -map-by ppr:1:node
        -rank-by node
        -bind-to none .....
```

machines.txt

```
Node 0
Node 1
...
Node P-1
...
Node max
```

Refuse parfois de *binder* sur le nœud entier : threads sur un seul socket !



## Exemple de déploiement: sol 4'

**Sol 4' : un processus MPI par socket et Cs threads/process**

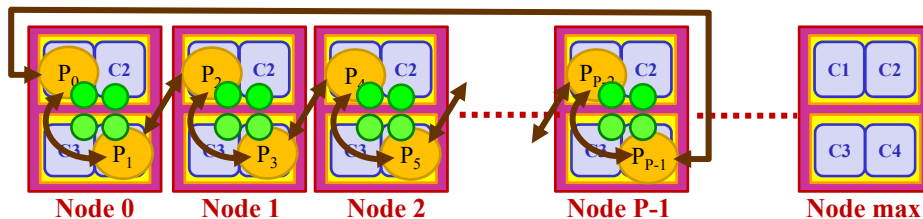
- Nbr de machines listées  $\geq P / S$   
(P : nbr de processus à créer, S sockets/nœud, Cs : nb de cœurs/socket)
- Un processus/socket, puis chaque processus créera Cs threads

```
mpirun -np #P -machinefile machines.txt
        -map-by ppr:1:socket
        -rank-by socket
        -bind-to socket .....
```

machines.txt

```
Node 0
Node 1
...
Node P/S-1
...
Node max
```

Solution plus efficace sur nœuds NUMA (nœuds modernes)



## Options de *binding* de MPI

Options pour contrôler les ressources accessibles à un processus :

```
mpirun --bind-to < none | hwthread | core |
                    socket | numa | board |
                    l1cache | l2cache | l3cache >
```

Toutes ne fonctionnent pas sur tous les systèmes!

## Rapport de *binding* de MPI

```
mpirun -np 4 -machinefile machsar.txt
        -map-by ppr:1:socket -rank-by socket -bind-to socket
        --report-bindings
        ./MatrixProduct -k 1 -klc 4 -nt 4
```

32 nodes × 2 sockets × 4 cores × hyperthreading

```
[sar01:85379] MCW rank 0 bound to
socket 0[core 0[hwt 0-1]], socket 0[core 1[hwt 0-1]],
socket 0[core 2[hwt 0-1]], socket 0[core 3[hwt 0-1]]:
[BB/BB/BB/BB][../../../../]
[sar01:85379] MCW rank 1 bound to
socket 1[core 4[hwt 0-1]], socket 1[core 5[hwt 0-1]],
socket 1[core 6[hwt 0-1]], socket 1[core 7[hwt 0-1]]:
[../../../../][BB/BE/BB/BB]
[sar02:78868] MCW rank 3 bound to
socket 1[core 4[hwt 0-1]], socket 1[core 5[hwt 0-1]],
socket 1[core 6[hwt 0-1]], socket 1[core 7[hwt 0-1]]:
[../../../../][BB/BE/BB/BB]
[sar02:78868] MCW rank 2 bound to
socket 0[core 0[hwt 0-1]], socket 0[core 1[hwt 0-1]],
socket 0[core 2[hwt 0-1]], socket 0[core 3[hwt 0-1]]:
[BB/BB/BB/BB][../../../../]
```



## Meilleur déploiement ?

**La performance du déploiement dépend de plusieurs facteurs :**

- **L'occupation des ressources de calcul disponibles**
  - Déployer une tâche (processus ou thread) par ressource :
    - au moins une tâche par cœur physique
    - ou une tâche par cœur logique (selon le noyau de calcul utilisé)
- **Le schéma de communication obtenu**
  - Minimiser les communications inter-nœuds
- **Le respect de l'architecture NUMA des nœuds**
  - Au moins un processus par sous-nœud NUMA (en général par socket)

Rmq : quand on travaille toujours sur un même cluster on finit par savoir ce qui est le plus efficace sur lui !

## Distributed application deployment with MPI

- Déploiement sur cluster de multicœurs et mécanismes de MPI
- **Exercices**
- Modélisation de performances

## Déploiement 1

**Cluster :** 16 nœuds ×  
4 sockets (processeurs) ×  
16 cœurs physiques ×  
2 threads (hyperthreading)

**Règles :**

- On alloue  $n$  nœuds ( $1 \leq n \leq 16$ )
- On cherche toujours à utiliser tous les cœurs des nœuds alloués

**Pb 1 :**

- Ressources allouées : 8 nœuds
- Stratégie de déploiement de l'application *myAppli* :
  - un processus MPI par cœur physique,
  - processus de numéros consécutifs sur un même nœud
  - un thread par cœur logique (hyp : option *-nt* de l'application)

**Question 1 :** combien de processus doit-on créer (option *-np* de *mpirun*) ?

→  $8 \times 4 \times 16 = 512$  processus MPI

**Question 2 :** combien de *threads* doit créer chaque processus (option *-nt*) ?

→ 2 threads

## Déploiement 1

**Cluster :** 16 nœuds ×  
4 sockets (processeurs) ×  
16 cœurs physiques ×  
2 threads (hyperthreading)

**Règles :**

- On alloue  $n$  nœuds ( $1 \leq n \leq 16$ )
- On cherche toujours à utiliser tous les cœurs des nœuds alloués

**Pb 1 :**

- Ressources allouées : 8 nœuds
- Stratégie de déploiement de l'application *myAppli* :
  - un processus MPI par cœur physique,
  - processus de numéros consécutifs sur un même nœud
  - un thread par cœur logique (hyp : option *-nt* de l'application)

**Question 3 :** quelle commande *mpirun* doit-on entrer ?

→ `mpirun -np 512 -machinefile machines.txt  
-map-by ppr:1:core -rank-by core -bind-to core  
myAppli -nt 2`

## Déploiement 2

**Cluster :** 16 nœuds ×  
4 sockets (processeurs) ×  
16 cœurs physiques ×  
2 threads (hyperthreading)

**Règles :**

- On alloue  $n$  nœuds ( $1 \leq n \leq 16$ )
- On cherche toujours à utiliser tous les cœurs des nœuds alloués

**Pb 2 :**

- Ressources allouées : 16 nœuds
- Stratégie de déploiement de l'application *myAppliOptim* :
  - un processus MPI par socket,
  - processus de numéros consécutifs sur un même nœud
  - un thread par cœur physique (hyp : option *-nt* de l'application)

**Question 1 :** combien de processus doit-on créer (option *-np* de *mpirun*) ?

→  $16 \times 4 = 64$  processus MPI

**Question 2 :** combien de *threads* doit créer chaque processus (option *-nt*) ?

→ 16 threads

## Déploiement 2

**Cluster :** 16 nœuds ×  
4 sockets (processeurs) ×  
16 cœurs physiques ×  
2 threads (hyperthreading)

**Règles :**

- On alloue  $n$  nœuds ( $1 \leq n \leq 16$ )
- On cherche toujours à utiliser tous les cœurs des nœuds alloués

**Pb 2 :**

- Ressources allouées : 16 nœuds
- Stratégie de déploiement de l'application *myAppliOptim* :
  - un processus MPI par socket,
  - processus de numéros consécutifs sur un même nœud
  - un thread par cœur physique (hyp : option *-nt* de l'application)

**Question 3 :** quelle commande *mpirun* doit-on entrer ?

→ `mpirun -np 64 -machinefile machines.txt  
-map-by ppr:1:socket -rank-by socket -bind-to socket  
myAppliOptim -nt 16`

## Déploiement 3

**Cluster :** 16 nœuds ×  
 4 sockets (processeurs) ×  
 16 cœurs physiques ×  
 2 threads (hyperthreading)

**Règles :**

- On alloue  $n$  nœuds ( $1 \leq n \leq 16$ )
- On cherche toujours à utiliser tous les cœurs des nœuds alloués

**Pb 3 :**

- Ressources allouées : 16 nœuds
- Stratégie de déploiement de l'application *myAppliOptim* :
  - 4 processus MPI par socket,
  - processus de numéros consécutifs sur des nœuds différents
  - un thread par cœur physique (hyp : option *-nt* de l'application)

**Question 1 :** combien de processus doit-on créer (option *-np* de *mpirun*) ?

→  $16 \times 4 \times 4 = 256$  processus MPI

**Question 2 :** combien de *threads* doit créer chaque processus (option *-nt*) ?

→ 4 threads

## Déploiement 3

**Cluster :** 16 nœuds ×  
 4 sockets (processeurs) ×  
 16 cœurs physiques ×  
 2 threads (hyperthreading)

**Règles :**

- On alloue  $n$  nœuds ( $1 \leq n \leq 16$ )
- On cherche toujours à utiliser tous les cœurs des nœuds alloués

**Pb 3 :**

- Ressources allouées : 16 nœuds
- Stratégie de déploiement de l'application *myAppliOptim* :
  - 4 processus MPI par socket,
  - processus de numéros consécutifs sur des nœuds différents
  - un thread par cœur physique (hyp : option

**Question 3 :** quelle commande *mpirun* doit-on entrer ?

→ `mpirun -np 256 -machinefile machines.txt  
 -map-by ppr:4:socket -rank-by node -bind-to socket  
 myAppliOptim -nt 4`

Sur chaque socket l'OS répartit 4×4 threads sur 16 cœurs

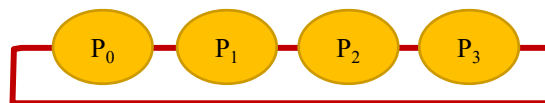
## Distributed application deployment with MPI

- Déploiement sur cluster de multicoeurs et mécanismes de MPI
- Exercices
- **Modélisation de performances**
  - **volume de communication en fonction du déploiement**
  - temps d'exécution pour un déploiement donné

### Cas d'un pgm en anneau (de processus)

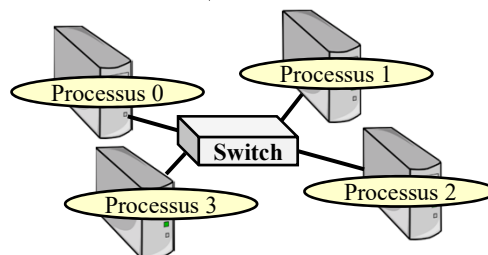
Produit de matrices denses en anneau:

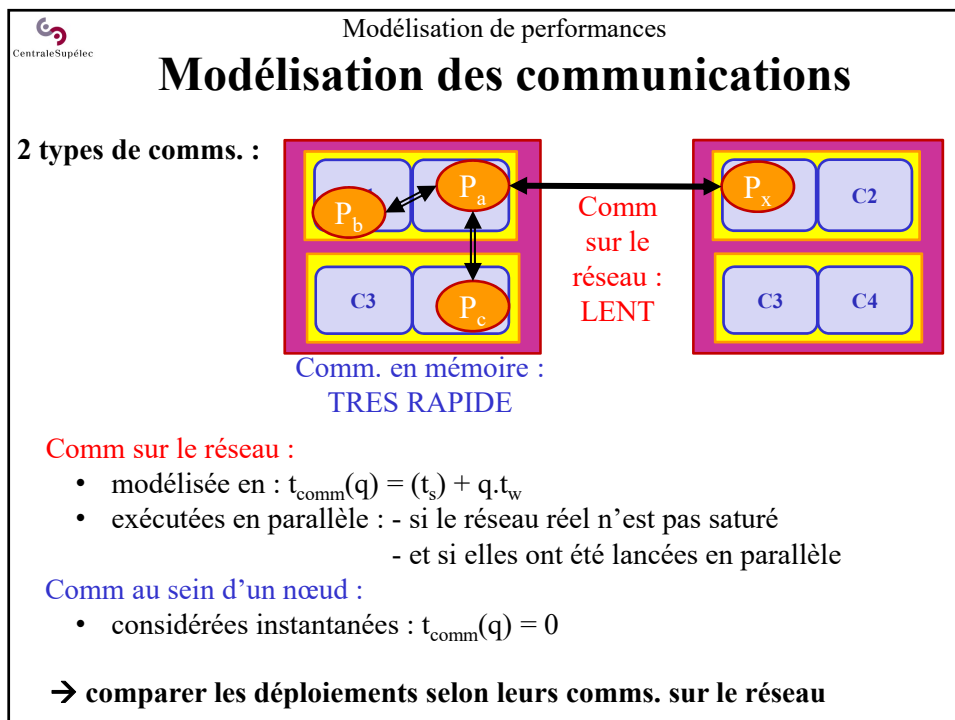
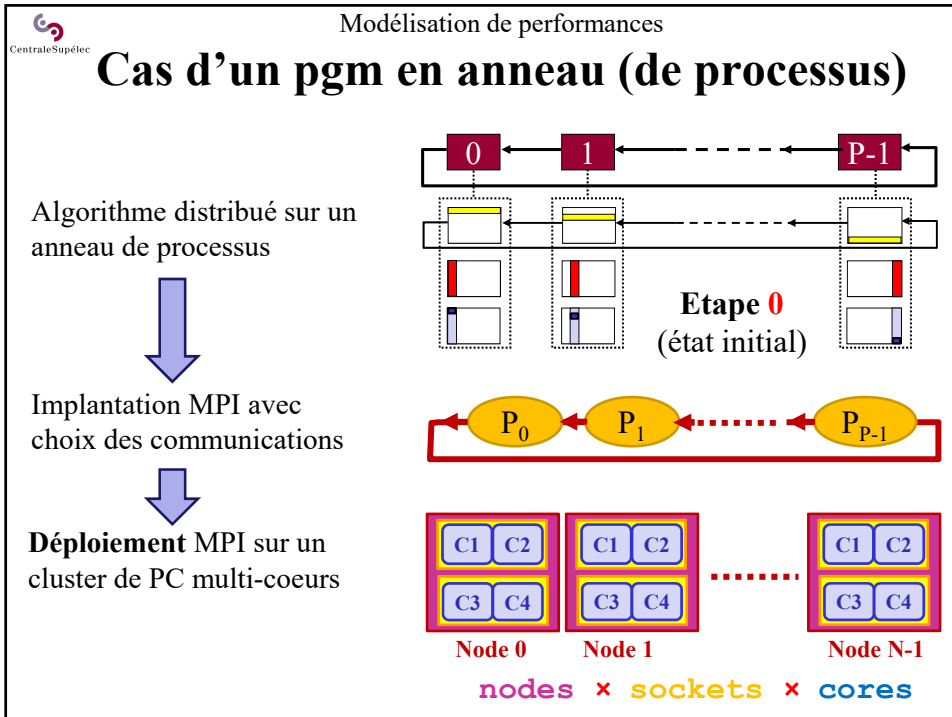
Anneau de processus



Déploiement visé :  
• 1 process/machine  
• processus  $k$  voisin des processus  $k-1$  et  $k+1$

Réseau réel de machines  
&  
1 processeur par machine





## Volume des comms. du déploiement

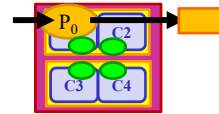
### Volume de communications sur le réseau d'interconnexion :

Cluster :  $N_{nodes} \times N_{socket/node} \times N_{coeurs/socket} = N_n \times N_s \times N_c$

Déploiement : 1 processus/nœud et  $N_s \times N_c$  threads par processus

- $N_n$  processus et étapes
- Volume d'une tranche de matrice A :  $Q_{slice} = Q_s = Q_A / N_n$

A chaque étape : - chaque processus émet  $Q_s$  data  
 - chaque nœud émet  $Q_s$  data vers un autre nœud



Au final :

Volume total de données traversant le réseau durant tout le calcul :

$$N_n \times (N_n \times Q_s) = N_n \times Q_A$$

Taille de chaque message :

$$Q_A / N_n$$

## Volume des comms. du déploiement

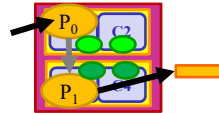
### Volume de communications sur le réseau d'interconnexion :

Cluster :  $N_{nodes} \times N_{socket/node} \times N_{coeurs/socket} = N_n \times N_s \times N_c$

Déploiement : 1 processus/socket et  $N_c$  threads par processus et numérotation selon les sockets

- $N_n \times N_s$  processus et étapes
- Volume d'une tranche de matrice A :  $Q_{slice} = Q_s = Q_A / (N_n \times N_s)$

A chaque étape : - chaque processus émet  $Q_s$  data  
 - chaque nœud émet  $Q_s$  data vers un autre nœud



Au final :

Volume total de données traversant le réseau durant tout le calcul :

$$(N_n \times N_s) \times (N_n \times Q_s) = N_n \times Q_A$$

Taille de chaque message :

$$Q_A / (N_n \times N_s)$$

Plus de plus petits msg, mais même volume total

## Volume des comms. du déploiement

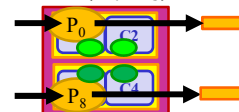
### Volume de communications **sur le réseau d'interconnexion** :

Cluster :  $N_{nodes} \times N_{socket/node} \times N_{coeurs/socket} = N_n \times N_s \times N_c$

Déploiement : 1 processus/socket et  $N_c$  threads par processus et numérotation selon les nœuds

- $N_n \times N_s$  processus et étapes
- Volume d'une tranche de matrice A :  $Q_{slice} = Q_s = Q_A / (N_n \times N_s)$

A chaque étape : - chaque processus émet  $Q_s$  data  
 - chaque nœud émet  $N_s \times Q_s$  data  
 vers d'autres nœuds



Au final :

Volume total de données traversant le réseau durant tout le calcul :

$$(N_n \times N_s) \times (N_n \times (N_s \times Q_s)) = N_n \times N_s \times Q_A$$

Taille de chaque message :

$$Q_A / (N_n \times N_s)$$

Bcp plus de plus petits msg, et volume total plus important !  
 → Plus lent !

## Distributed application deployment with MPI

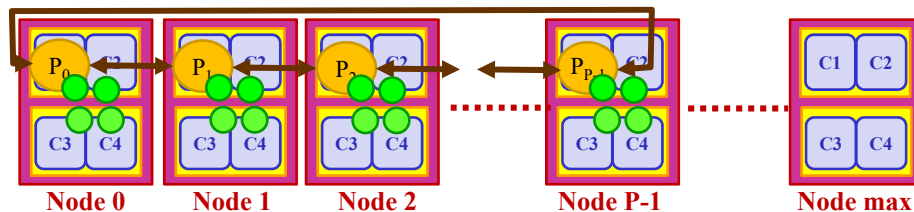
- Déploiement sur cluster de multicœurs et mécanismes de MPI
- Exercices
- **Modélisation de performances**
  - volume de communication en fonction du déploiement
  - **temps d'exécution pour un déploiement donné**



## Configuration logicielle et matérielle

### Déploiement : un processus MPI par node et C threads/process

- P nœuds (machines), avec C cœurs par nœud
- Un réseau architecturé autour d'un switch
- P processus dans le programme, les processus sont multithreadés
- Communication des processus en anneau
- Un processus/nœud, puis chaque processus crée C threads
- C : nb de cœurs/nœud
- Hypothèse : pas de limitation du *binding*



## Modélisation du temps d'exécution

### Performances sans recouvrement :

- Temps mono-noeud :

$$T_{seq} = N \cdot (2 \cdot \sqrt{N} - 1) t_{flop}$$

$$T_{seq} \approx 2 \cdot N \cdot \sqrt{N} t_{flop}$$

$t_{flop}$  tient compte du multithreading sur C cœurs

Modèle de comm :

- $t_{com}(q) = t_s + q \cdot t_w$
- toutes les comms en parallèles

- Temps parallèle :

$$t_{1-calc} = \frac{\sqrt{N}}{P} \cdot \frac{\sqrt{N}}{P} \cdot (2 \cdot \sqrt{N} - 1) t_{flop}$$

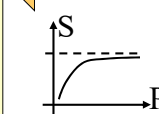
$$t_{1-calc} \approx 2 \cdot \frac{N \cdot \sqrt{N}}{P^2} t_{flop}$$

$$t_{1-circ} = t_s + \sqrt{N} \cdot \frac{\sqrt{N}}{P} t_w$$

$$t_{1-circ} \approx \frac{N}{P} t_w$$

- Speed up :

$$S_{no} = \frac{T_{seq}}{T_{par}^{no}} \approx P \cdot \frac{1}{1 + \frac{P t_w}{2 \cdot \sqrt{N} t_{flop}}}$$



$$T_{par}^{no} \approx P \cdot (t_{1-calc} + t_{1-circ})$$

$$T_{par}^{no} \approx 2 \cdot \frac{N \cdot \sqrt{N}}{P} t_{flop} + N t_w$$

## Modélisation de perfs sur anneau théorique

Comparaison calculs-communications (**sans** recouvrement) :

- Calculs :

$$T_{par}^{calc} \approx 2 \cdot \frac{N^{3/2}}{P} t_{flop}$$

$$P \text{ fixé} \Rightarrow O(T_{par}^{calc}) = O(N^{3/2})$$

- Circulation :

$$T_{par}^{circ} \approx N t_w$$

$$O(T_{par}^{circ}) = O(N)$$

$$O(T_{par}^{calc}) > O(T_{par}^{circ})$$

Avec  $P$  fixé, quand  $N \uparrow$  : les calculs deviennent prépondérants  
 → le surcoût des comms devient négligeable

Et le speedup devient parfait :

$$S^{no} \approx P \cdot \frac{1}{1 + \frac{P \cdot t_w}{2 \cdot \sqrt{N} \cdot t_{flop}}} \xrightarrow{N \rightarrow \infty} P$$

« Bon » problème pour un TP !

Distributed application  
 deployment with MPI

**Questions ?**