Modèle pour la redistribution de données complexes

Aurélien Esnard

LaBRI & INRIA Futurs
Université de Bordeaux

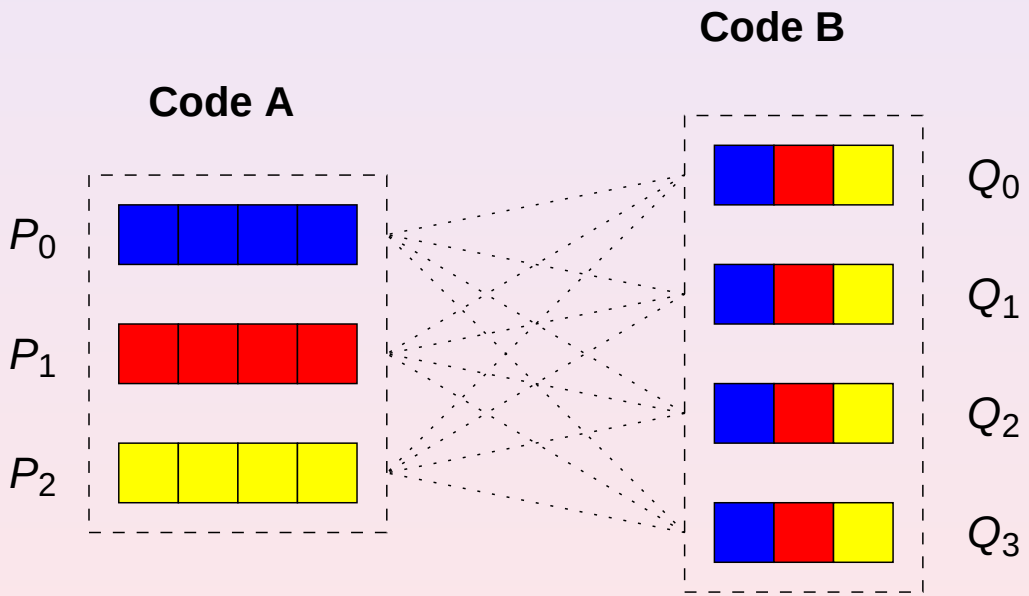
RenPAR'16, Le Croisic, France, avril 2005

Plan

Plan

Le problème de la redistribution des données

- Deux ensembles de processeurs disjoints P et Q ($M \neq N$)
- Un même ensemble de données partagé entre deux codes
- Schémas de distribution des données *a priori* différents



Le problème de la redistribution des données

Les étapes de la redistribution

1. Description des données distribuées
2. Génération des messages
3. Ordonnancement
4. Communication (flux parallèles)

Le problème de la redistribution des données

Les étapes de la redistribution

1. Description des données distribuées
2. Génération des messages
3. Ordonnancement
4. Communication (flux parallèles)

Remarque

Nous laissons de côté le problème d'ordonnancement des communications (cf. Wagner & Jeannot).

Le problème de la redistribution des données

Les étapes de la redistribution

1. Description des données distribuées
2. Génération des messages
3. Ordonnancement
4. Communication (flux parallèles)

Modèle de redistribution symbolique

Le développement d'une « solution standard » au problème de la redistribution nécessite de définir une modèle de description également standard.

Travaux existants

Bibliothèques	Couplage	Objets, Distributions
HPF, ScaLAPACK	M×M	tableaux, bloc-cyclique
CUMULVS	M×1	tableaux, bloc-cyclique + patchs
CUMULVS	M×1	particules dans $\mathbb{Z}^n$
PAWS	M×N	tableaux, rectilinéaire
CCA M×N	M×N	tableaux, PAWS + CUMULVS

Constat

Le problème de la redistribution des données a principalement été étudié dans le cas des distributions régulières de tableaux denses.

Travaux reliés

MetaChaos/InterComm, MpCCI

Travaux existants

Bibliothèques	Couplage	Objets, Distributions
HPF, ScaLAPACK	M×M	tableaux, bloc-cyclique
CUMULVS	M×1	tableaux, bloc-cyclique + patchs
CUMULVS	M×1	particules dans $\mathbb{Z}^n$
PAWS	M×N	tableaux, rectilinéaire
CCA M×N	M×N	tableaux, PAWS + CUMULVS

Constat

Le problème de la redistribution des données a principalement été étudié dans le cas des distributions régulières de tableaux denses.

Travaux reliés

MetaChaos/InterComm, MpCCI

Travaux existants

Bibliothèques	Couplage	Objets, Distributions
HPF, ScaLAPACK	M×M	tableaux, bloc-cyclique
CUMULVS	M×1	tableaux, bloc-cyclique + patchs
CUMULVS	M×1	particules dans $\mathbb{Z}^n$
PAWS	M×N	tableaux, rectilinéaire
CCA M×N	M×N	tableaux, PAWS + CUMULVS

Constat

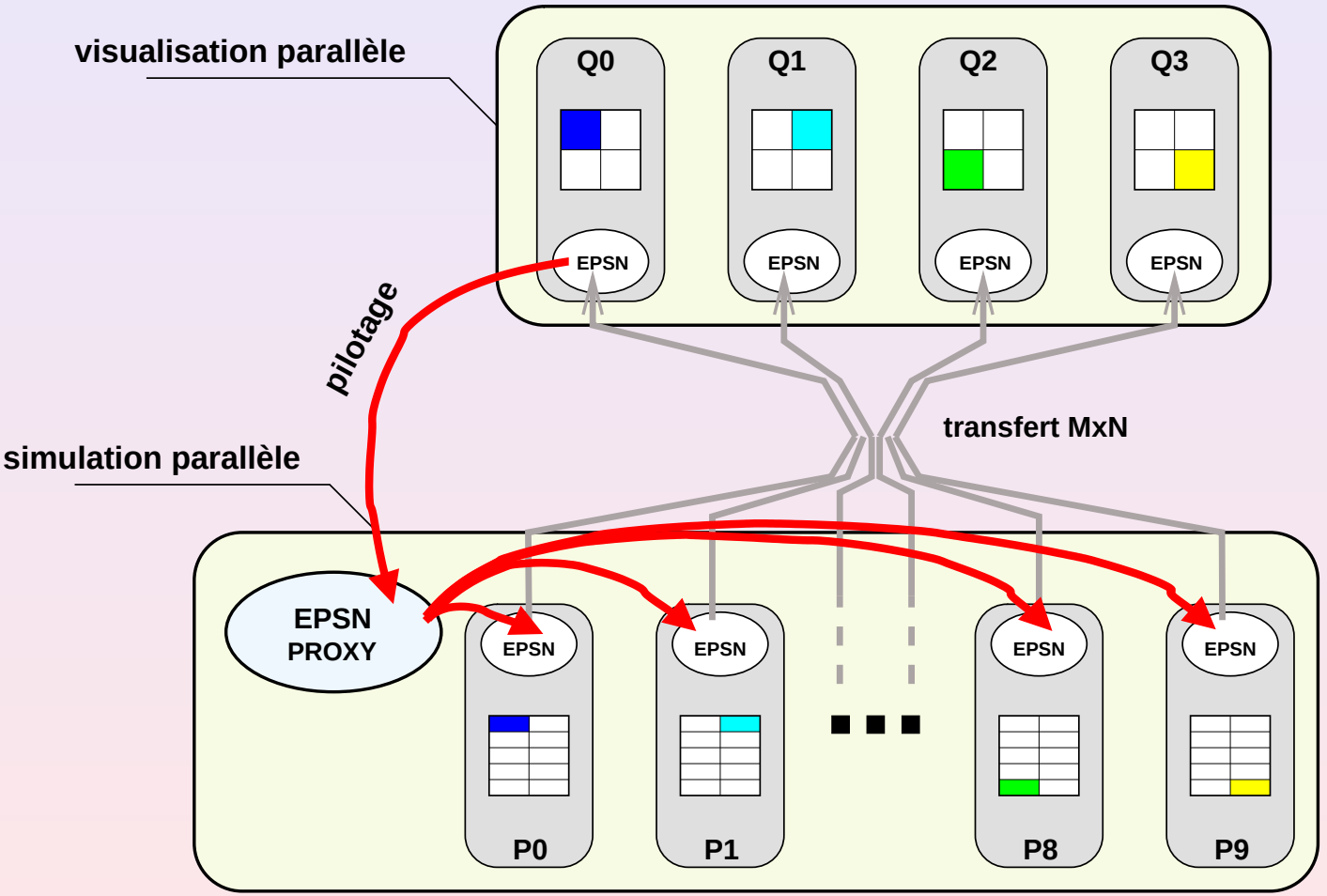
Le problème de la redistribution des données a principalement été étudié dans le cas des distributions régulières de tableaux denses.

Travaux reliés

MetaChaos/InterComm, MpCCI

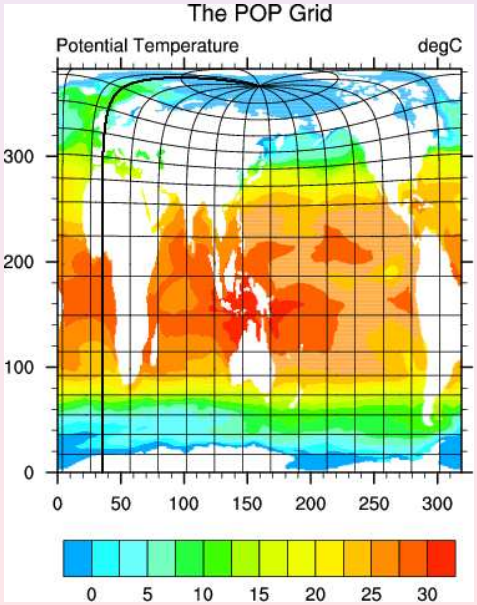
Travaux motivés par le projet EPSN (ACI GRID)

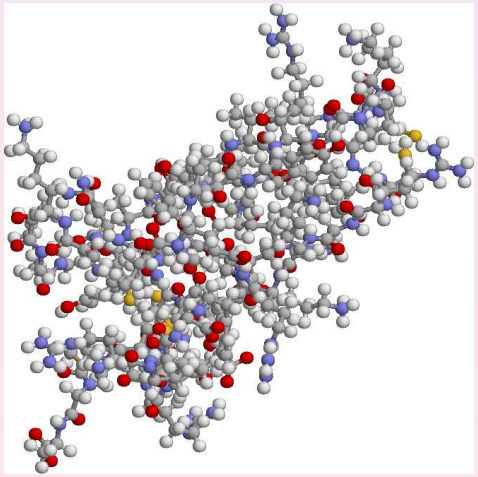
EPSN : Environnement pour le Pilotage de Simulation Numériques



Besoin d'étendre les travaux sur la redistribution...

- Pour supporter des distributions de données plus irrégulières
  - Structures creuses, *overlap* entre les blocs
  - Sous-domaine de visualisation
- Pour redistribuer d'autres types d'objets plus complexes
  - grilles structurées, particules, molécules, maillages non structurés...

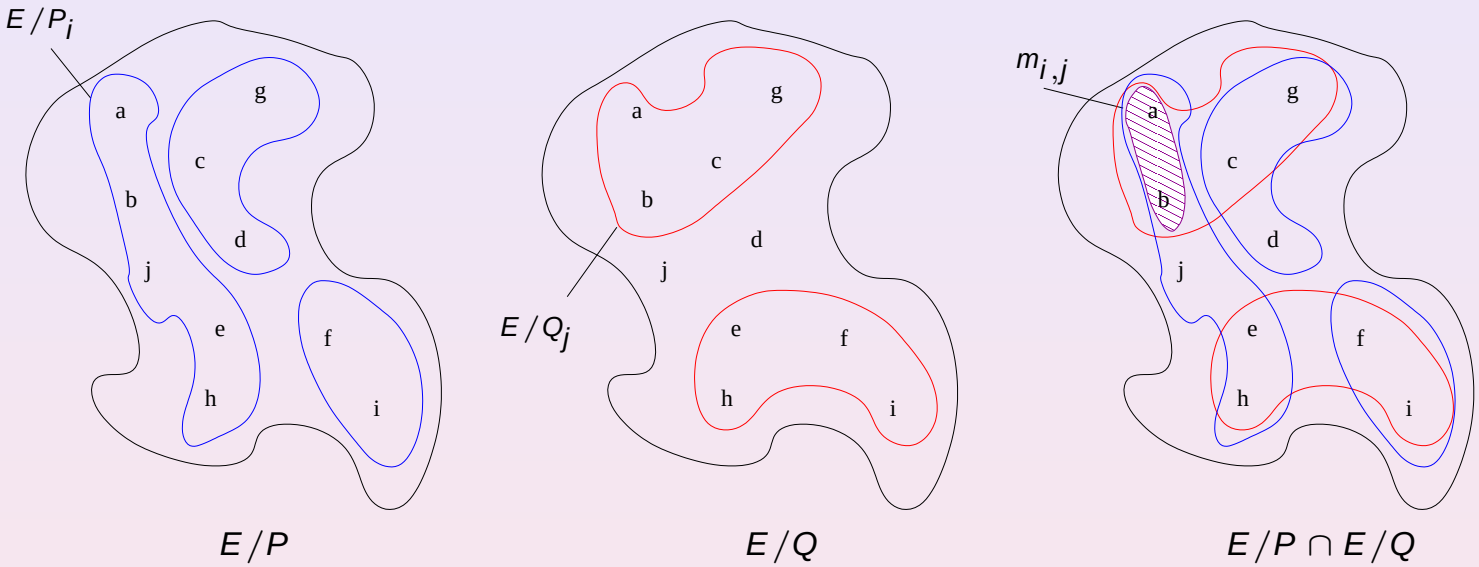




Plan

Formulation ensembliste

Soit E un ensemble d'éléments totalement ordonnés, distribués sur P et Q selon E/P et E/Q.

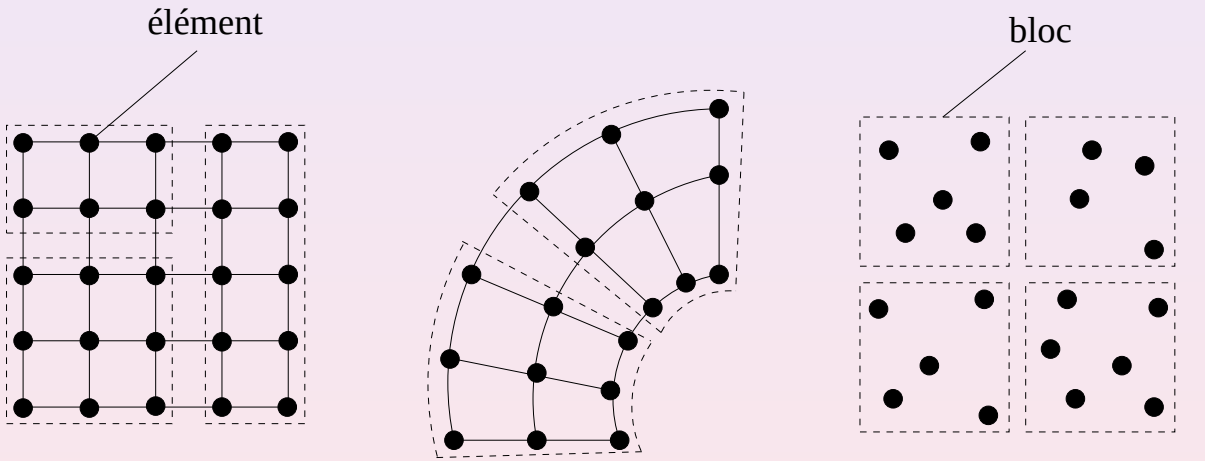


Principe d'intersection

Le message $m_{i,j}$ est l'ensemble des éléments échangés entre P_i et Q_j : $m_{i,j} = E/P_i \cap E/Q_j$

Objets spatiaux distribués par bloc

- Objet complexe : ensemble d'éléments $\rightarrow$ série de données
- Objet spatial : fonction coordonnée dans l'espace $\mathbb{Z}^n$ ou $\mathbb{R}^n$
- Distributions généralisées : ensemble arbitraire de blocs
- Stockage des données : modèle en *stride*



(a) tableaux (b) grille structurée (c) ensemble de particules

Les blocs jouent le rôle de conteneur pour les éléments.

Algorithme parallèle de génération des messages

Initialisation

- Chaque processeur connait sa description locale.
- Il a besoin de connaître l'ensemble de la distribution distante.
- Il n'a pas besoin de connaître l'ensemble de la distribution locale !

Algorithme

Pour chaque processeur P_i, faire en parallèle :
  Pour chaque processeur distant Q_j, faire :

1. Calcul de l'ensemble des blocs de redistribution ($\Pi_{i,j}$) par intersection géométrique des blocs de P_i et de Q_j
2. Tri des blocs de redistribution dans un ordre canonique
3. Génération du message $m_{i,j}$ par extraction des éléments associés aux blocs de redistribution

Algorithme parallèle de génération des messages

Initialisation

- Chaque processeur connait sa description locale.
- Il a besoin de connaître l'ensemble de la distribution distante.
- Il n'a pas besoin de connaître l'ensemble de la distribution locale !

Algorithme

Pour chaque processeur P_i, faire en parallèle :
Pour chaque processeur distant Q_j, faire :

1. Calcul de l'ensemble des blocs de redistribution ($\Pi_{i,j}$) par intersection géométrique des blocs de P_i et de Q_j
2. Tri des blocs de redistribution dans un ordre canonique
3. Génération du message $m_{i,j}$ par extraction des éléments associés aux blocs de redistribution

Algorithme parallèle de génération des messages

Initialisation

- Chaque processeur connait sa description locale.
- Il a besoin de connaître l'ensemble de la distribution distante.
- Il n'a pas besoin de connaître l'ensemble de la distribution locale !

Algorithme

Pour chaque processeur P_i, faire en parallèle :
  Pour chaque processeur distant Q_j, faire :

1. Calcul de l'ensemble des blocs de redistribution ($\Pi_{i,j}$) par intersection géométrique des blocs de P_i et de Q_j
2. Tri des blocs de redistribution dans un ordre canonique
3. Génération du message $m_{i,j}$ par extraction des éléments associés aux blocs de redistribution

Algorithme parallèle de génération des messages

Initialisation

- Chaque processeur connait sa description locale.
- Il a besoin de connaître l'ensemble de la distribution distante.
- Il n'a pas besoin de connaître l'ensemble de la distribution locale !

Algorithme

Pour chaque processeur P_i, faire en parallèle :
Pour chaque processeur distant Q_j, faire :

1. Calcul de l'ensemble des blocs de redistribution ($\Pi_{i,j}$) par intersection géométrique des blocs de P_i et de Q_j
2. Tri des blocs de redistribution dans un ordre canonique
3. Génération du message $m_{i,j}$ par extraction des éléments associés aux blocs de redistribution

Algorithme parallèle de génération des messages

Initialisation

- Chaque processeur connait sa description locale.
- Il a besoin de connaître l'ensemble de la distribution distante.
- Il n'a pas besoin de connaître l'ensemble de la distribution locale !

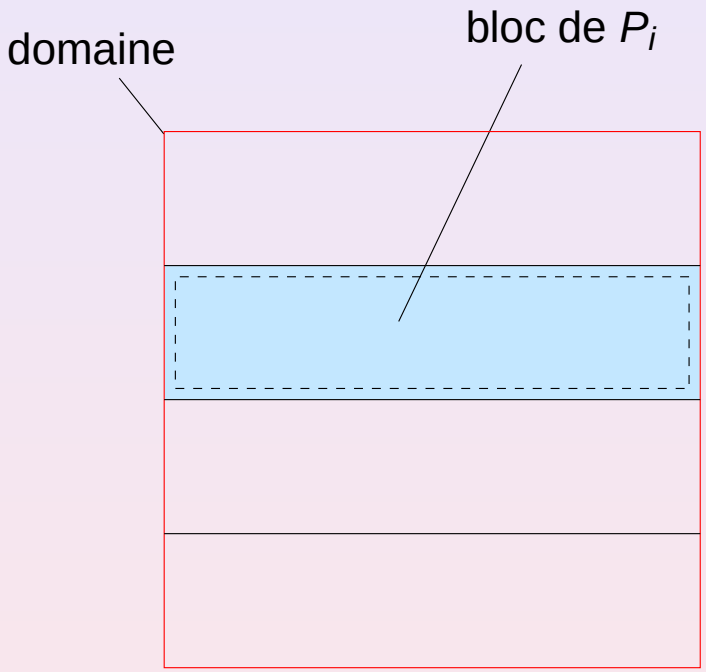
Algorithme

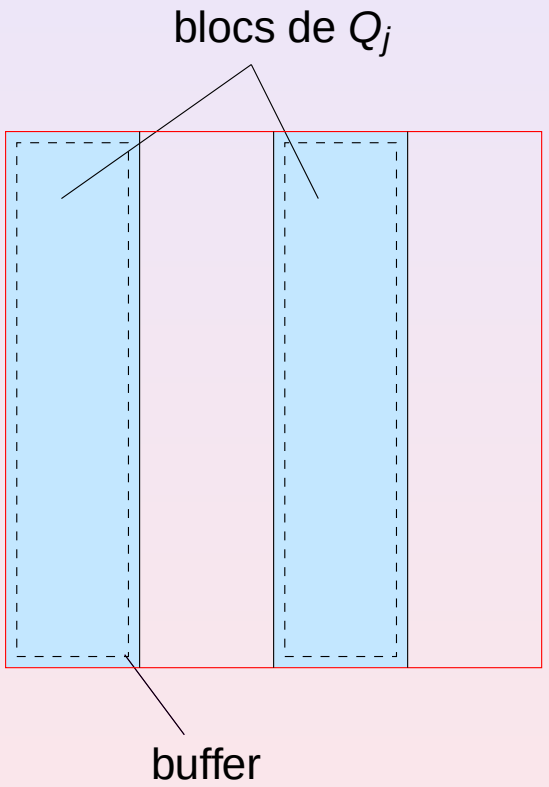
Pour chaque processeur P_i, faire en parallèle :
Pour chaque processeur distant Q_j, faire :

1. Calcul de l'ensemble des blocs de redistribution ($\Pi_{i,j}$) par intersection géométrique des blocs de P_i et de Q_j
2. Tri des blocs de redistribution dans un ordre canonique
3. Génération du message $m_{i,j}$ par extraction des éléments associés aux blocs de redistribution

Calcul des blocs de redistribution $\Pi_{i,j}$

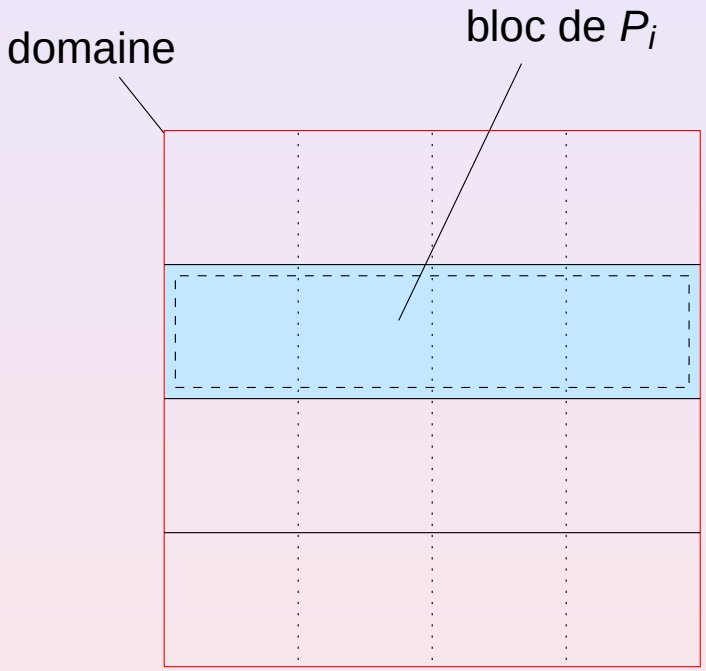
$\Pi_{i,j}$ = intersection géométrique des blocs de P_i et de Q_j

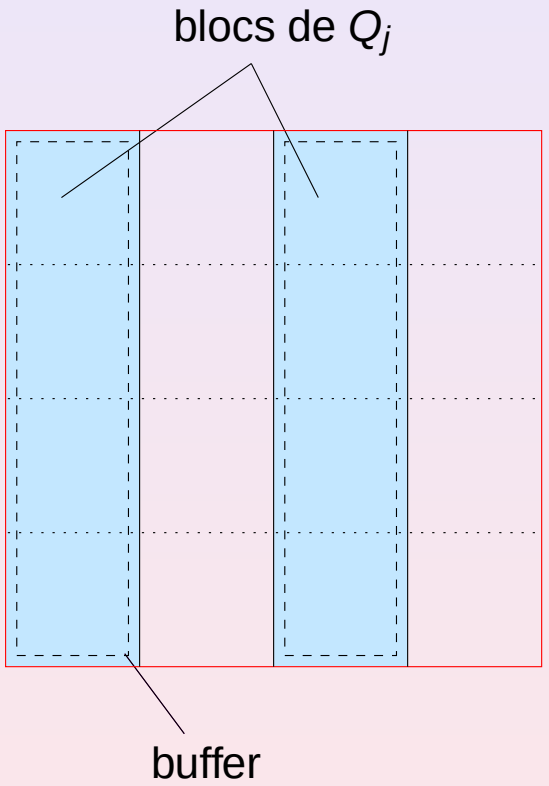




Calcul des blocs de redistribution $\Pi_{i,j}$

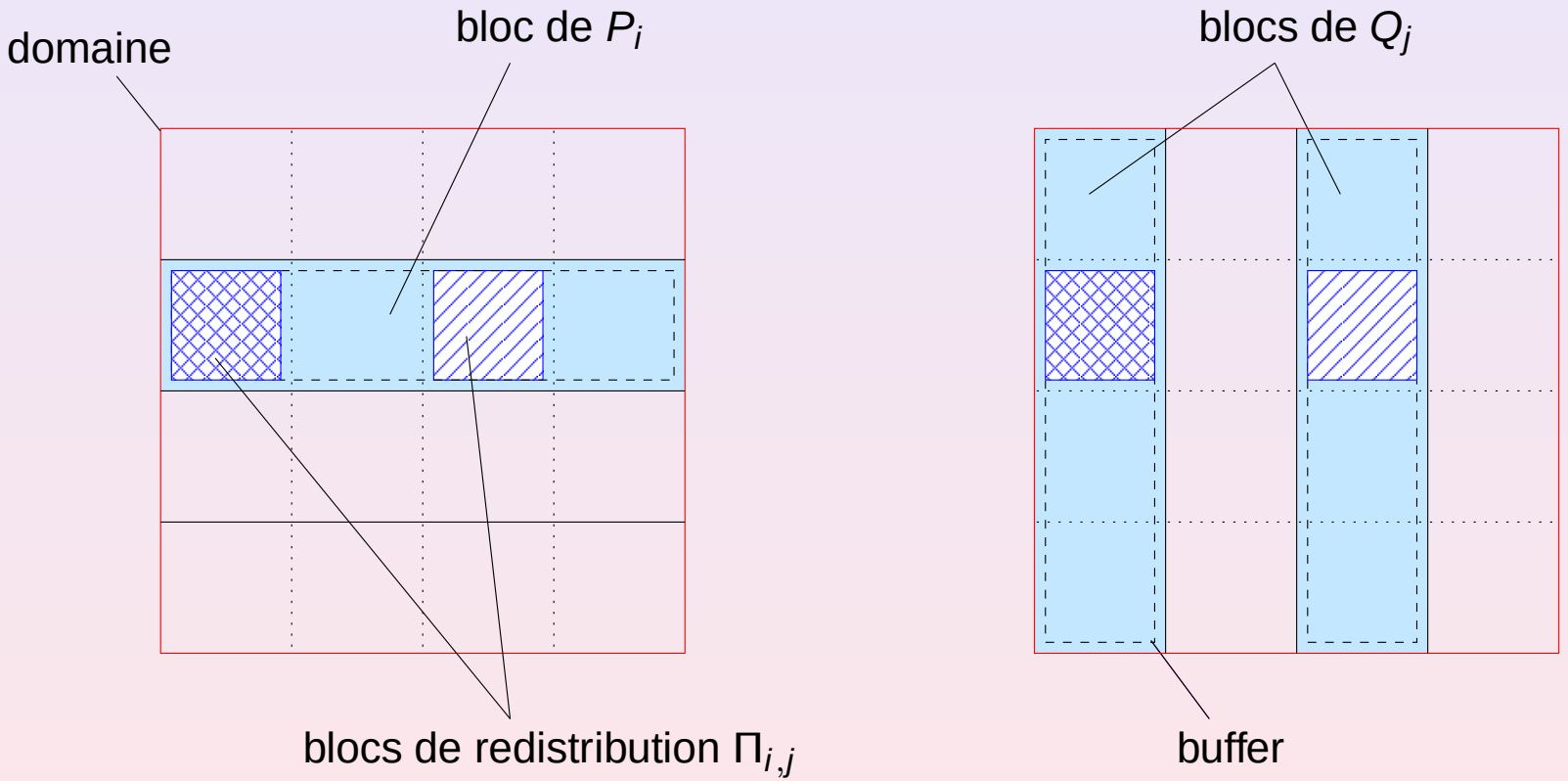
$\Pi_{i,j}$ = intersection géométrique des blocs de P_i et de Q_j





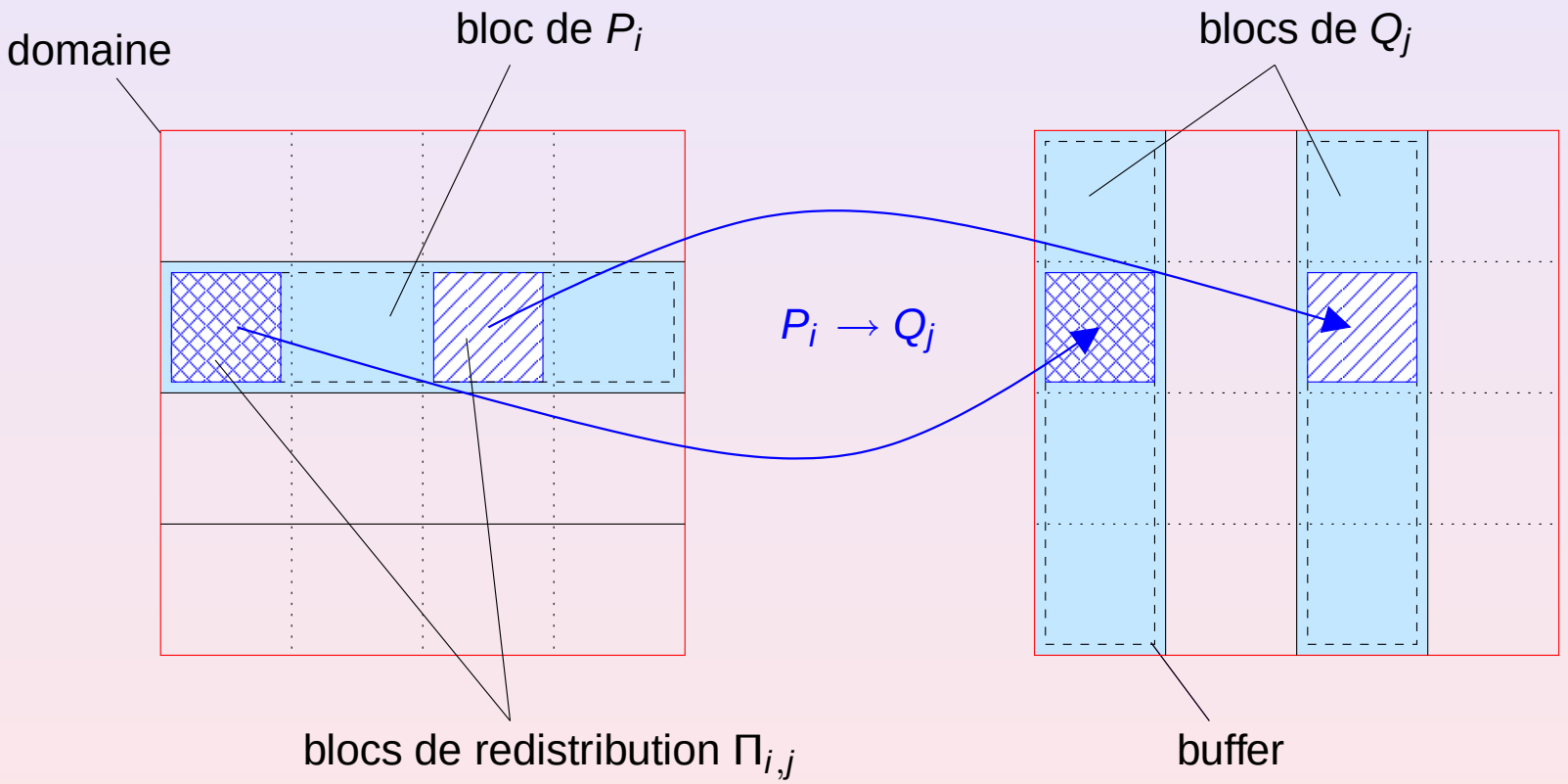
Calcul des blocs de redistribution $\Pi_{i,j}$

$\Pi_{i,j}$ = intersection géométrique des blocs de P_i et de Q_j



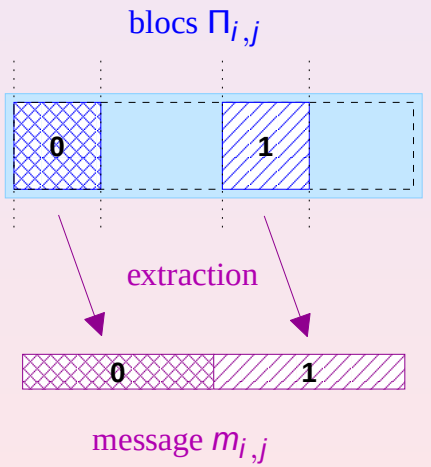
Calcul des blocs de redistribution $\Pi_{i,j}$

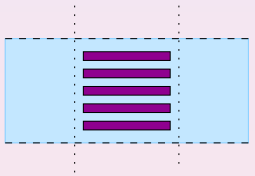
$\Pi_{i,j}$ = intersection géométrique des blocs de P_i et de Q_j



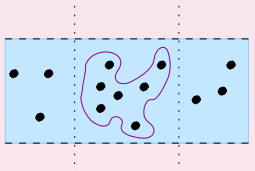
Opération abstraite d'extraction

- Message $m_{i,j}$ structuré par les blocs de redistribution $\Pi_{i,j}$
- Sérialisation des éléments extraits des blocs de $\Pi_{i,j}$ dans le message (selon ordre local des éléments)
- L'ordre canonique garantit que le message émis est structuré identiquement au message reçu





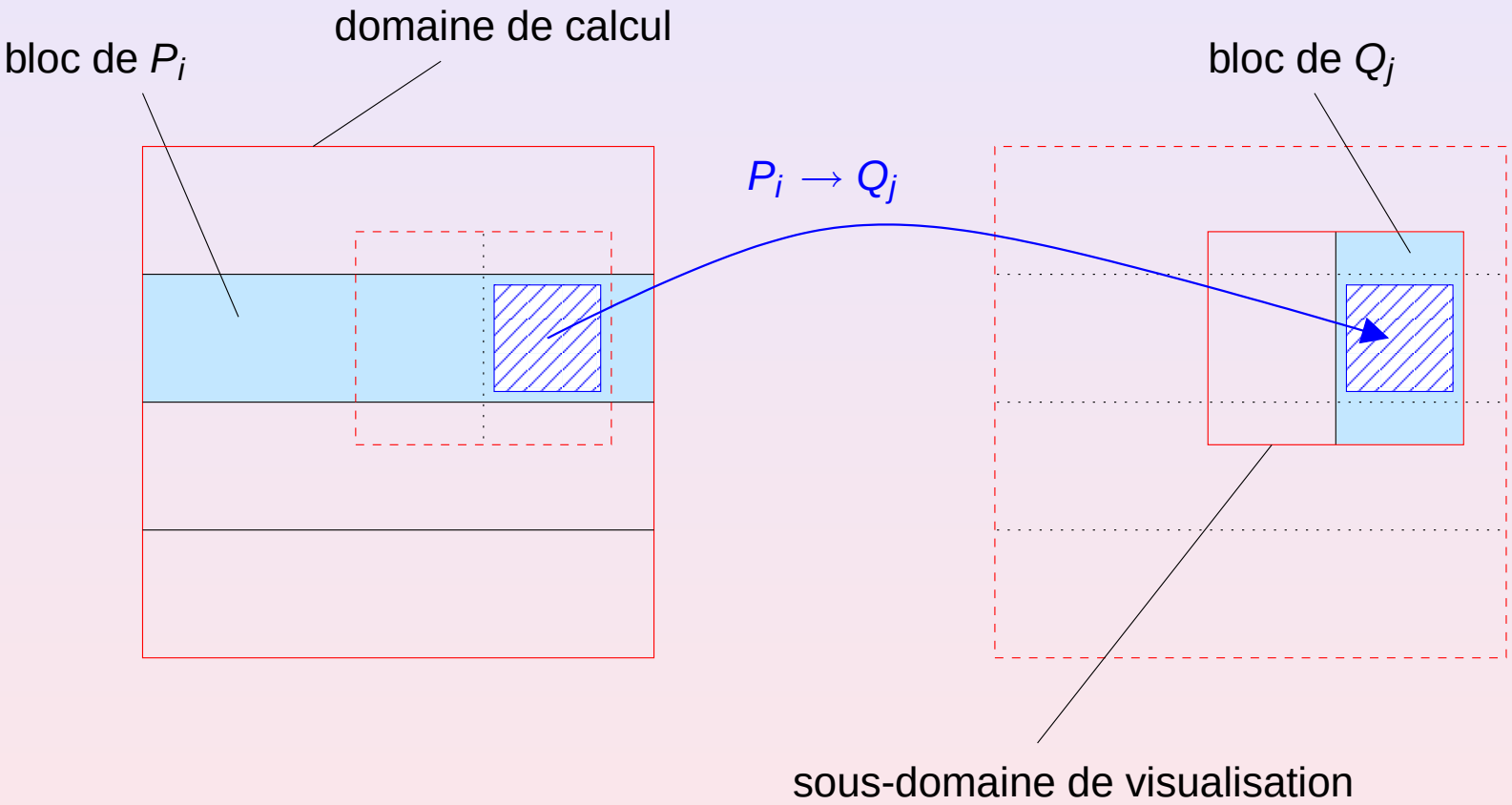
a) cas des tableaux



b) cas des particules

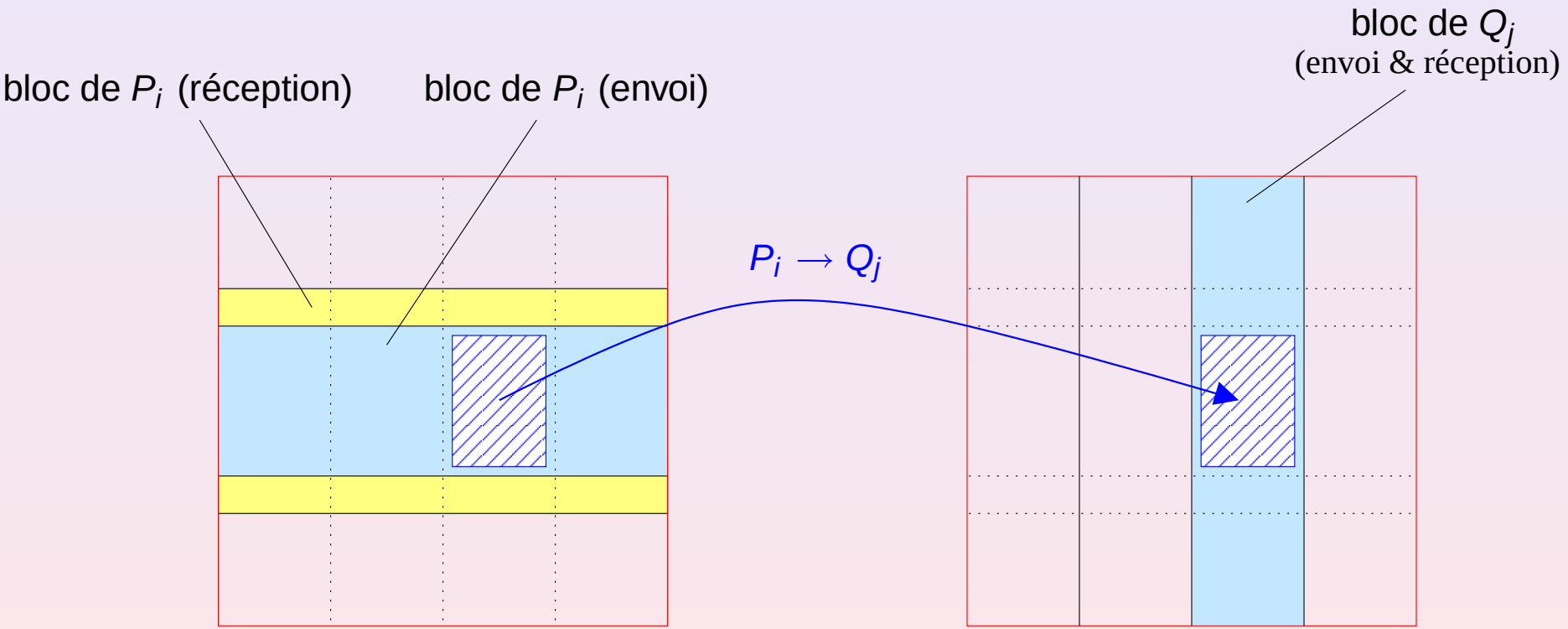
Exemple

Sous-domaine de visualisation



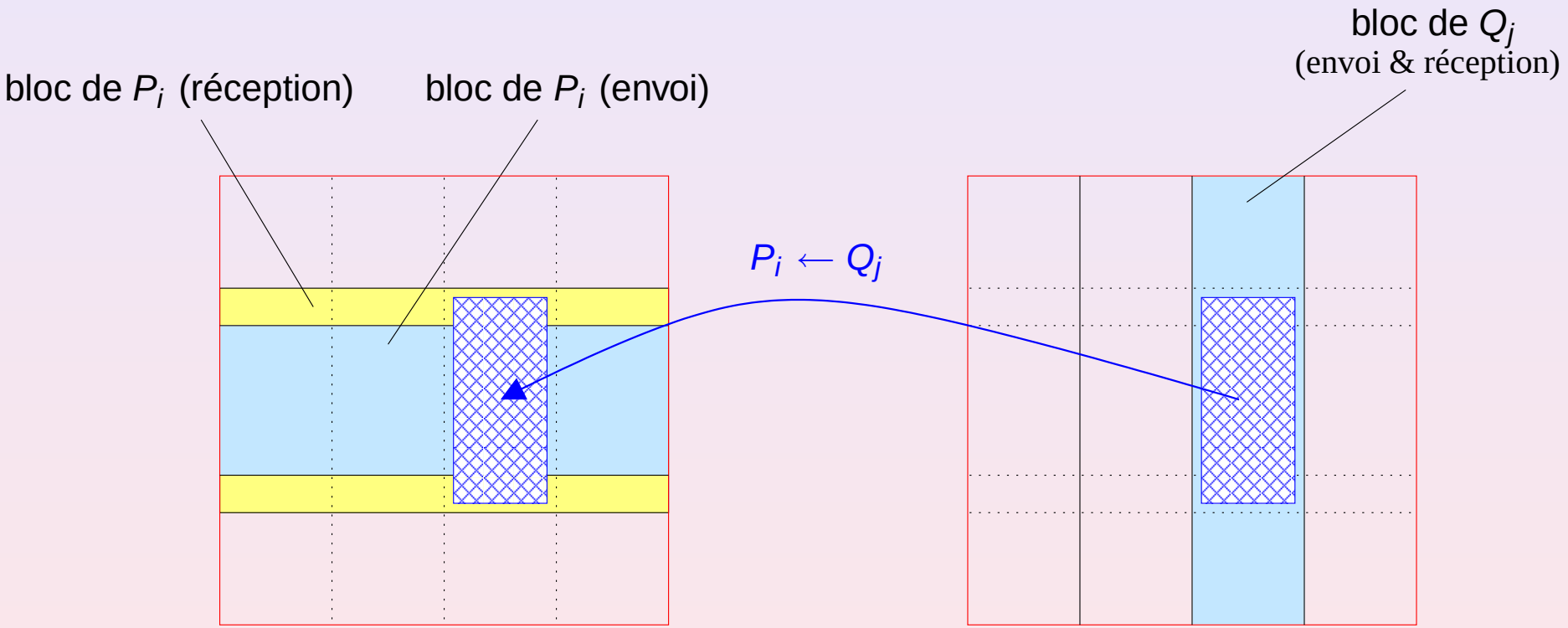
Extension de l'algorithme

Blocs d'envoi et de réception (ghost cells)



Extension de l'algorithme

Blocs d'envoi et de réception (ghost cells)

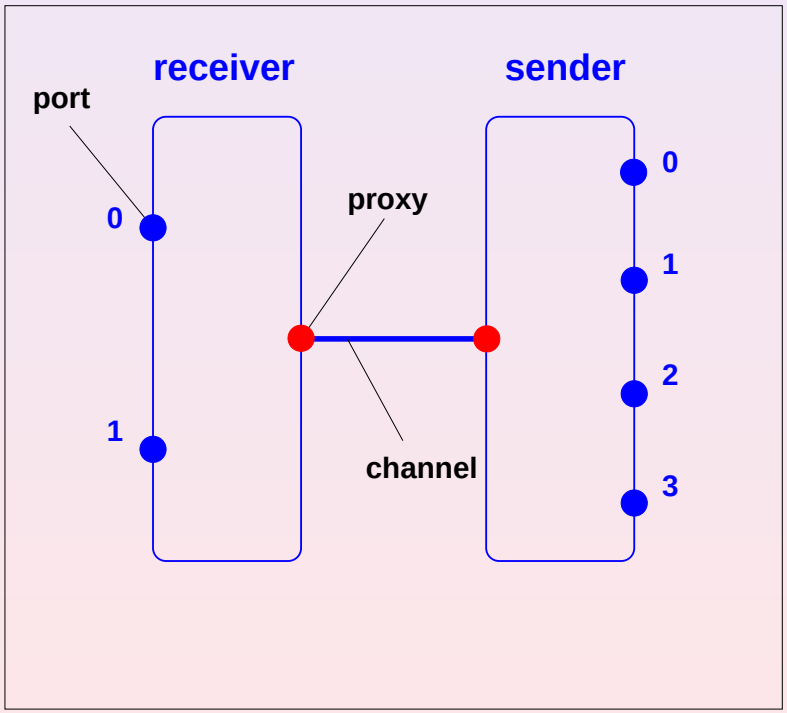


Plan

Bibliothèques RedSYM & RedCORBA

Environnement logiciel pour le couplage de codes

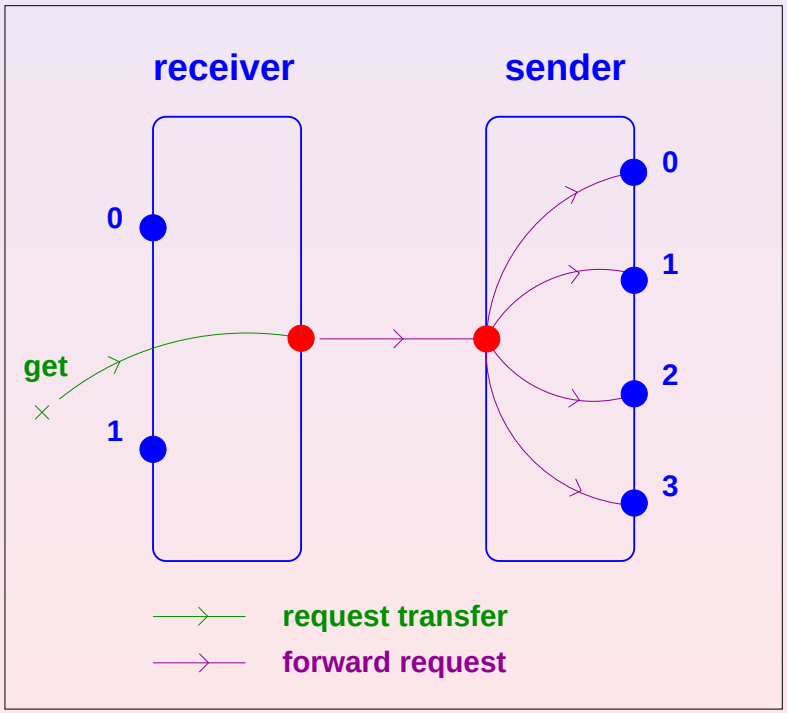
- RedSYM : redistribution symbolique
- RedCORBA : couche de communication basée sur CORBA



Bibliothèques RedSYM & RedCORBA

Environnement logiciel pour le couplage de codes

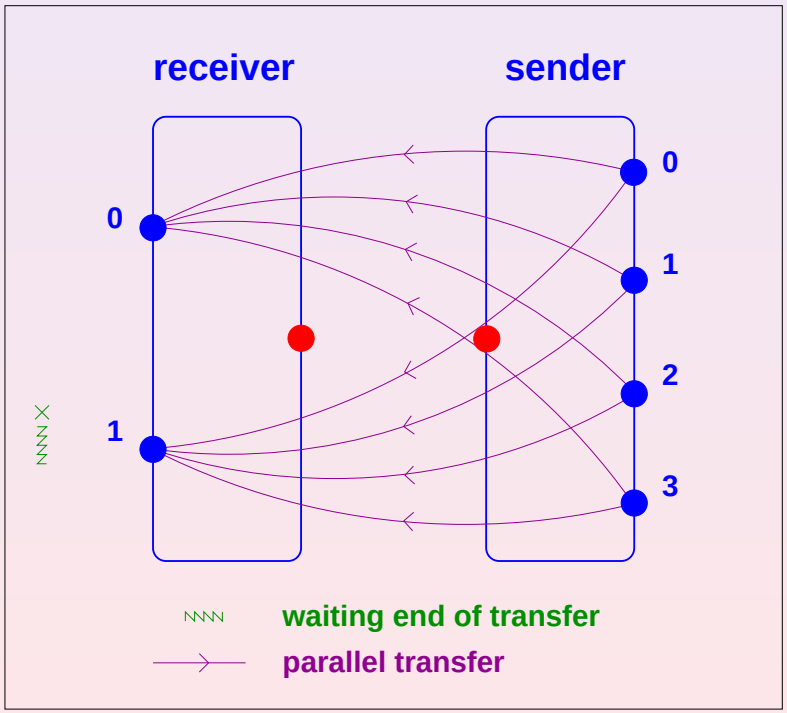
- RedSYM : redistribution symbolique
- RedCORBA : couche de communication basée sur CORBA



Bibliothèques RedSYM & RedCORBA

Environnement logiciel pour le couplage de codes

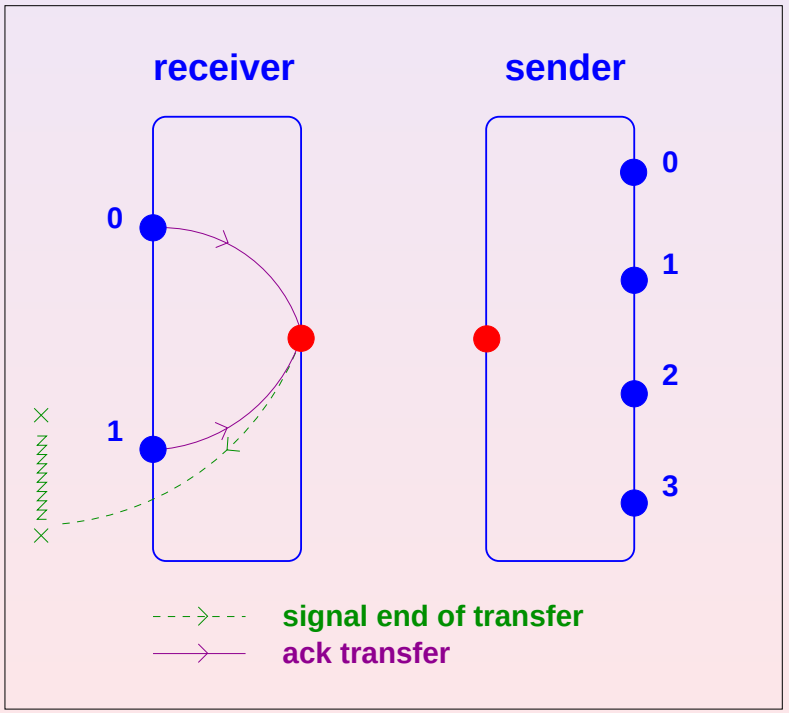
- RedSYM : redistribution symbolique
- RedCORBA : couche de communication basée sur CORBA



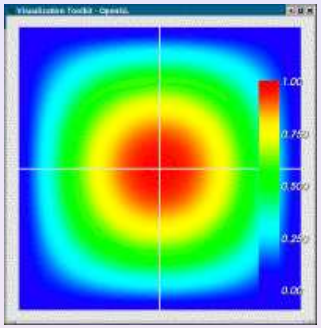
Bibliothèques RedSYM & RedCORBA

Environnement logiciel pour le couplage de codes

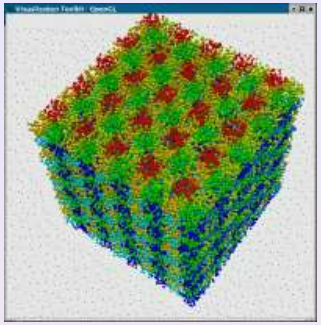
- RedSYM : redistribution symbolique
- RedCORBA : couche de communication basée sur CORBA



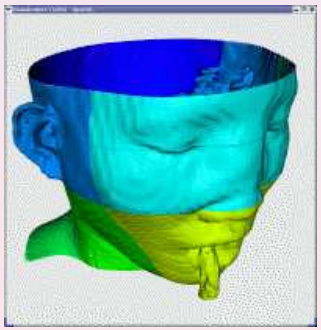
Quelques applications



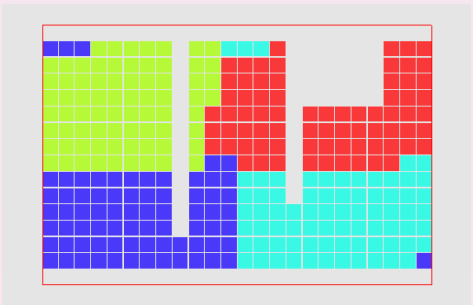
Equation de la chaleur 2D (6×4)



Cas d'école avec des particules en 3D (8×8)



Calcul à distance d'une iso-surface parallèle (4×8)

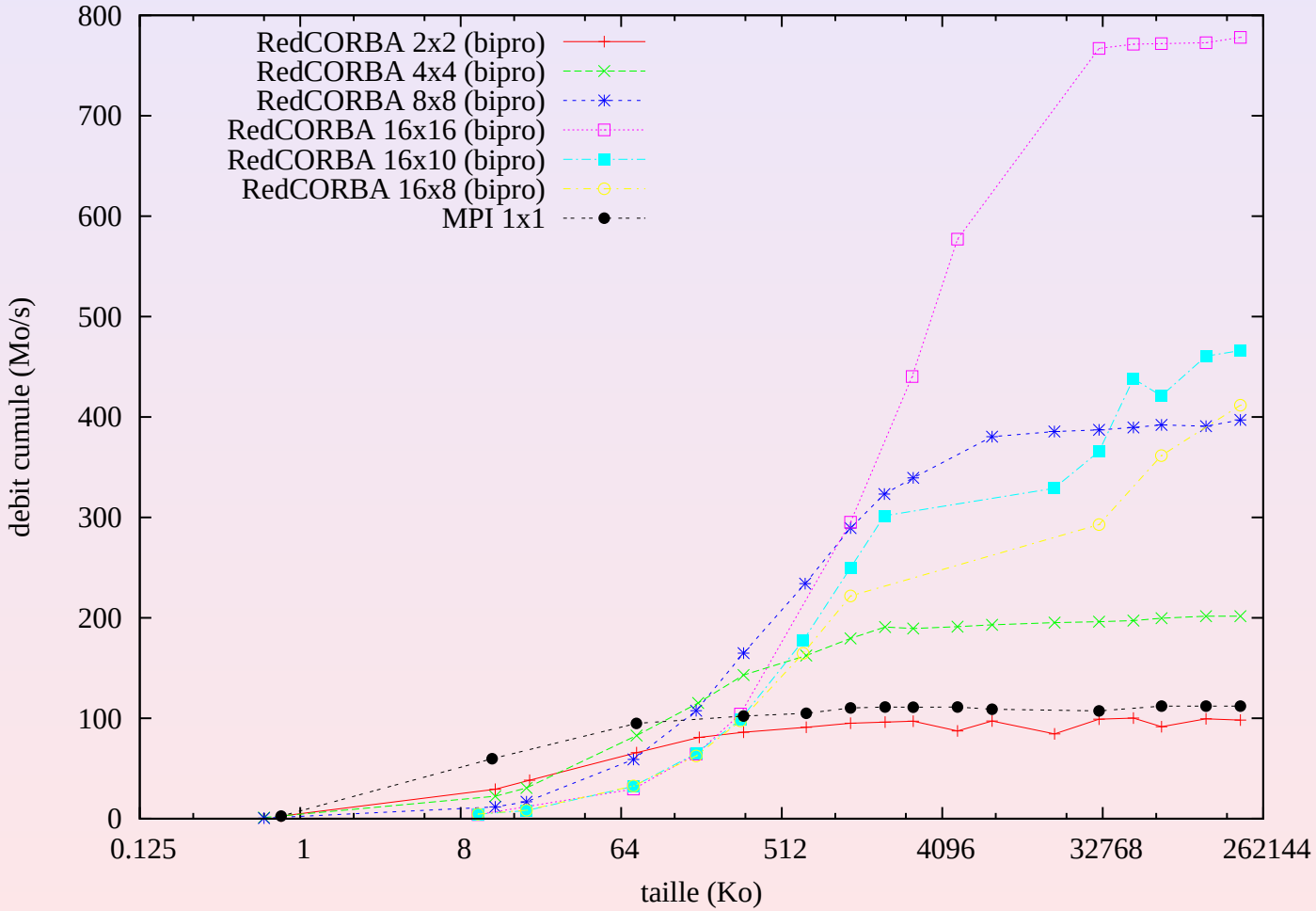


Distribution test dans POP (4×4)

Evaluation des performances dans RedCORBA

Redistribution des tableaux 2D (bloc-ligne vers bloc-colonne)

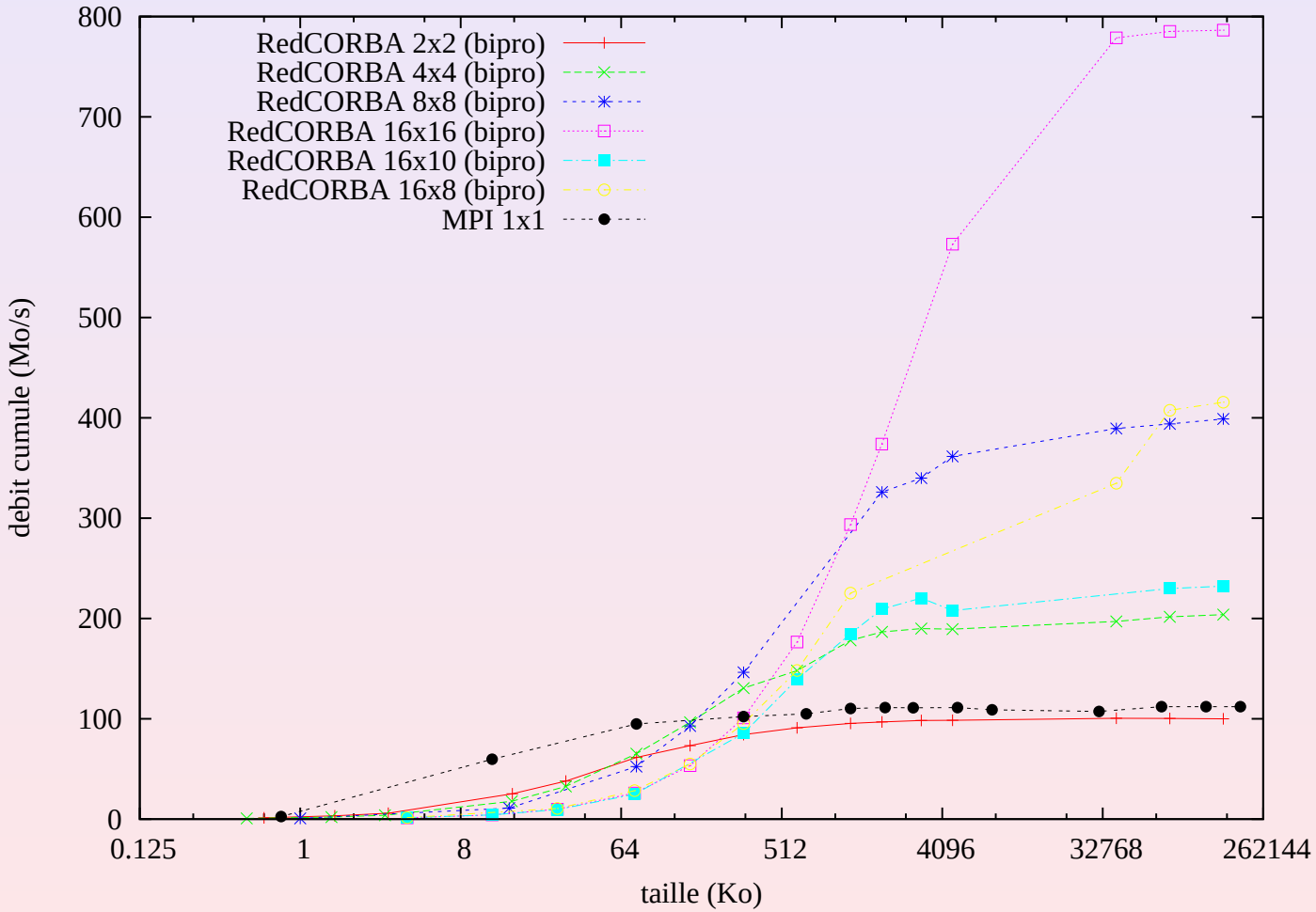
Cluster 16 bixeons. Giga-Ethernet. OmniORB4 (108 Mo/s). LAM/MPI (112 Mo/s). De 128 o à 200 Mo (5000x5000).



Evaluation des performances dans RedCORBA

Redistribution des particules 2D (pavages réguliers en M^2 vers N^2 blocs)

Cluster 16 bixeons. Giga-Ethernet. OmniORB4 (108 Mo/s). LAM/MPI (112 Mo/s). De 128 o à 200 Mo (10M de particules).



Conclusion

Résumé

- Approche spatiale de la redistribution (tableaux, grilles, particules)
- Développement de deux bibliothèques RedSYM & RedCORBA
- Utilisation dans le projet EPSN (ACI GRID EPSN)

En cours et perspectives

- Intégration en cours dans PaCO++ (ARC RedGRID)
- Approche « placement » de la redistribution dans le cas des maillages non structurés