Parallélisation d'un code de diffusion de la chaleur en deux dimension

Dufumier & Valade

Projet d'AMS301

But du TP

Code séquentiel

Créer un code séquentiel qui résolve l'équation de la chaleur sur un grille 2D sur un seul processeur en utilisant la méthode des différences finies spatiales et temporelles.

But du TP

Code séquentiel

Créer un code séquentiel qui résolve l'équation de la chaleur sur un grille 2D sur un seul processeur en utilisant la méthode des différences finies spatiales et temporelles.

Code parallel

Paralleliser ce code sur un grille structurée.

But du TP

Code séquentiel

Créer un code séquentiel qui résolve l'équation de la chaleur sur un grille 2D sur un seul processeur en utilisant la méthode des différences finies spatiales et temporelles.

Code parallel

Paralleliser ce code sur un grille structurée.

Outils

Utilisation du langage `C` et de la bibliothèque `MPI`. On fera tourner le code sur la machine `gin` pour étudier les scalabilité faible et forte.

But du développement séquentiel préalable

Buts principaux

- Créer des outils réutilisables dans le code parallèle
- Avoir une méthodologie de vérfication de la justesse du code. On utilise une solution exacte en espace et en temps et on calcul de l'écart de la solution approximée à cette solution exacte.

But du développement séquentiel préalable

Buts principaux

- Créer des outils réutilisables dans le code parallèle
- Avoir une méthodologie de vérfication de la justesse du code. On utilise une solution exacte en espace et en temps et on calcul de l'écart de la solution approximée à cette solution exacte.

On s'attachera à vérifier que

- Les arguments sont bien passés en début de fonction
- La solution exacte est bonne
- L'équation locale bien implémentée
- La norme L^2 fonctionne

Structure du programme

- Récupération des arguments
- Allocation de la mémoire (grilles `read`, `write`, `exact_sol`)
- Calcul de la CFL (sortie si trop grande)
- Initialisation de `read` avec la solution exacte

  Pour $t \in [1, N_{steps}]$

    Pour $i, j \in [1, N_{pts} - 2]^2$

      - Mettre a jour la grille `write` au point (i, j) à partir de `read`

    Fin Pour

  - Comparer `write` avec `exact_sol` avec la norme L^2
  - Échanger les pointeurs de `read` et `write`

  Fin Pour
- Afficher les résultats

Résultats

Problème et méthode

On procède à un découpage structuré de la grille en deux dimensions.

On dipose des valeurs suivantes

- Une grille globale de $N_x \times N_y$ points
- P processeurs

Problème et méthode

On procède à un découpage structuré de la grille en deux dimensions.

On dipose des valeurs suivantes

- Une grille globale de $N_x \times N_y$ points
- P processeurs

On cherche à connaître

- La répartition des processeurs sur une grille $P_x \times P_y$
- La taille de la grille locale $N_x^p \times N_y^p$ pour chaque processeur (sauf sur les bords)

Problème et méthode

On procède à un découpage structuré de la grille en deux dimensions.

On dipose des valeurs suivantes

- Une grille globale de $N_x \times N_y$ points
- P processeurs

On cherche à connaître

- La répartition des processeurs sur une grille $P_x \times P_y$
- La taille de la grille locale $N_x^p \times N_y^p$ pour chaque processeur (sauf sur les bords)

Méthode

- On choisit de faire que les grilles locales soient homotétiques à la grille globale
- Pour maximiser le nombre de processeurs utilisés, on ajoute une fonction heuristique

Construction de la grille de processeurs

1. Premier calcul des P_x, P_y :

$$P_x = \left\lfloor \sqrt{P \frac{N_x}{N_y}} \right\rfloor, \qquad P_y = \left\lfloor \frac{P}{P_x} \right\rfloor$$

2. Heurisitique de modification des P_x, P_y :

$$P_x, P_y = \underset{i,j \in V}{\text{argmin}} \left(P - i \times j \right), \qquad V = \{P_x, P_x \pm 1\} \times \{P_y, P_y \pm 1\}$$

3. Calcul des N_x^p, N_y^p :

$$N_x^p = \left\lfloor \frac{N_x}{P_x} \right\rfloor, \qquad N_y^p = \left\lfloor \frac{N_y}{P_y} \right\rfloor$$

Exemple illustratif

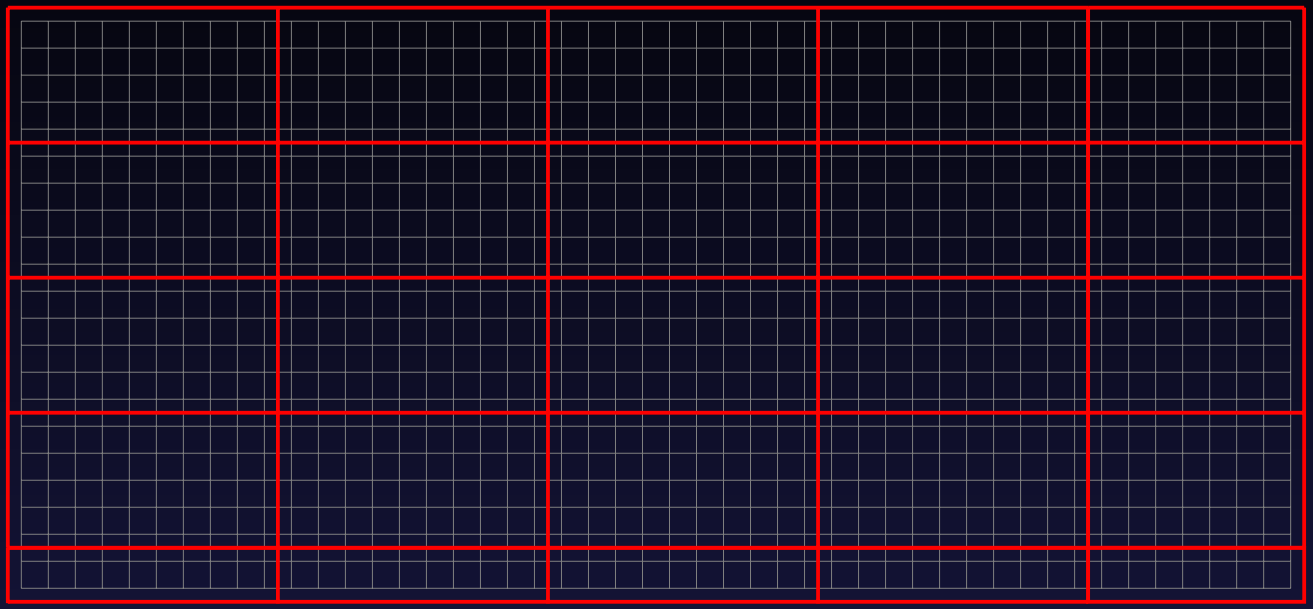


Figure: Répartion d'une grille de taille 47 par 21 points sur 25 processeurs.

Processeurs supplémentaires

Existence inevitable

- Souvent impossible d'utiliser tous les processeurs à disposition.
- Exemple : si P est un nombre premier...

Processeurs supplémentaires

Existence inevitable

- Souvent impossible d'utiliser tous les processeurs à disposition.
- Exemple : si P est un nombre premier...

Réutilisation

- Le premier processeur non utilisé devient le processeur auxiliaire :
- Chargé de l'affichage
- Calcul la CFL
- Rassemble les erreurs L^2 locales et les somme

Processeurs supplémentaires

Existence inevitable

- Souvent impossible d'utiliser tous les processeurs à disposition.
- Exemple : si P est un nombre premier...

Réutilisation

- Le premier processeur non utilisé devient le processeur auxiliaire :
- Chargé de l'affichage
- Calcul la CFL
- Rassemble les erreurs L^2 locales et les somme

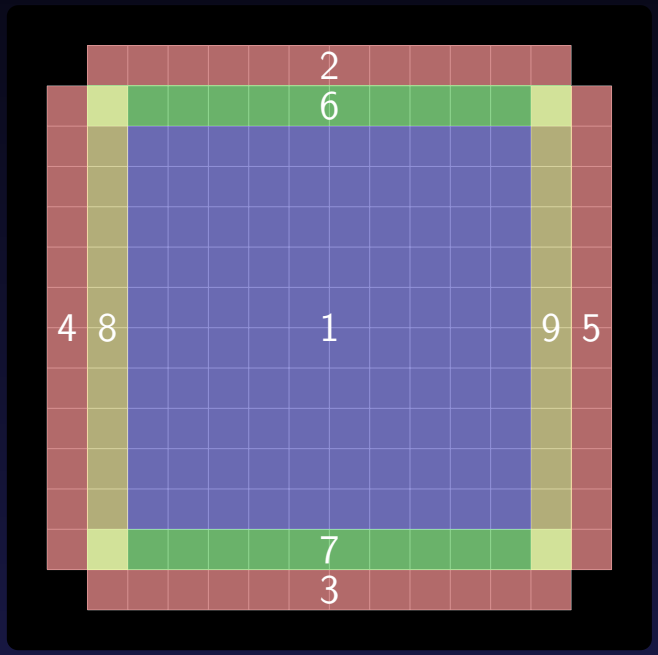
Les autres...

- sont perdus...

Une structure non triviale

Intérêts

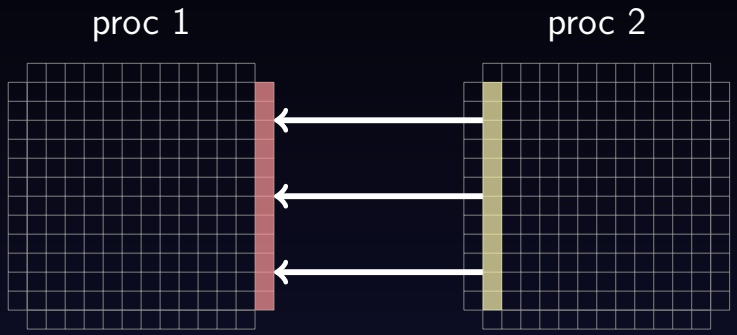
- Accéder sans difficulté aux informations copiées depuis les processeurs voisins
- Avoir des contenants facilitant les communications entre processeurs



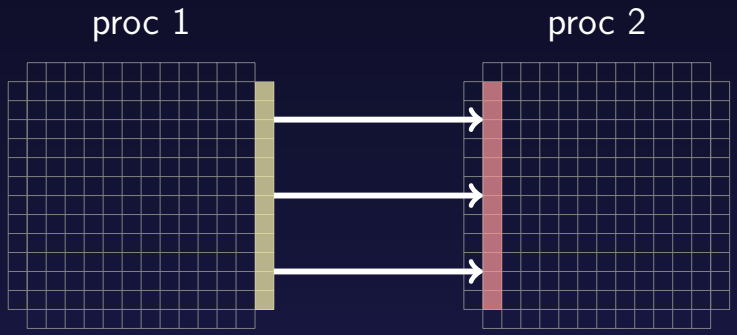
1. `center`
2. `top_out`
3. `bottom_out`
4. `left_out`
5. `right_out`
6. `top_in`
7. `bottom_in`
8. `left_in`
9. `right_in`

Communication entre grilles étendues

Exemple de communication



Communication droite vers gauche : copie `right_in` de proc 1 dans `left_out` de proc 2.



Communication gauche vers droite : copie `right_out` de proc 1 dans `left_in` de proc 2.