Parallélisation d'une méthode d'éléments finis pour la résolution de l'équation de la chaleur

Calcul parallèle scientifique - Soutenance de Projet
Master 2 Analyse Modélisation Simulation

Pierre-Emmanuel Emeriau & Gaétan Facchinetti

Université Paris-Saclay
Ecole Nationale Supérieure des Techniques Avancées
Ecole Normale Supérieure Paris-Saclay

24 novembre 2016

On considère un domaine Ω de $\mathbb{R}^2$.
Nous ne nous interessons qu'au problème *stationnaire*.

- $u(\mathbf{x})$ la température au point $\mathbf{x} \in \Omega$.
- $\mathbf{j}(\mathbf{x})$ le flux de chaleur au point $\mathbf{x} \in \Omega$.
- $f(\mathbf{x})$ Energie produite par unité de surface au point $\mathbf{x} \in \Omega$.
- λ conductivité termique, constante pour un matériaux uniforme.

La conservation de l'energie et la loi de Fourier donnnent alors :

$$\operatorname{div}(\mathbf{j}) = f \quad \text{et} \quad \mathbf{j} = -\lambda \boldsymbol{\nabla} u \tag{1}$$

D'où, l'équation de la chaleur en régme stationnaire :

$$\begin{cases} \Delta u = f & \text{dans } \Omega \\ u = g & \text{sur } \partial\Omega \end{cases} \tag{2}$$

On considère un relèvement u_0 de $g \in H^{1/2}(\Omega)$ dans $H^1(\Omega)$.
On pose $\tilde{u} = u - u_0$. Le problème (2) se réécrit :

Trouver $\tilde{u} \in H_0^1(\Omega)$ tel que

$$\forall v \in H_0^1 \quad \int_\Omega \nabla\tilde{u}\nabla v = \int_\Omega fv \tag{3}$$

On effectue une résolution par éléments finis de Lagrange P_1.
Soit $T = \{\tau_l\}_l$ une famille de triangulation de Ω.
Soit $V_h^0 = \{v \in \mathcal{C}^0(\Omega) \mid \forall l \quad v_{|\tau_l} \in P_1(\tau_l)\}$.
Soit $(\phi_j)_{1 \leq j \leq N_i}$ une base de V_h^0 avec $N_i = dim(V_h^0)$.

On décompose $\tilde{u}$ sur cette base : $\tilde{u} = \sum_{j=1}^{N_i} \tilde{u}_j \phi_j$.

$$\text{Ainsi,} \quad (3) \Rightarrow \forall k \in [\![1, N_i]\!] \quad \sum_{j=1}^{N_i} \tilde{u}_j \int_\Omega \nabla\phi_j \nabla\phi_k = \int_\Omega f\phi_k \tag{4}$$

Ceci permettrait de réécrire le problème sous forme matricielle :

On pose $\forall (p,q) \in [\![1, N_i]\!] \quad \tilde{\mathbb{K}}_{p,q} = \int_\Omega \nabla\phi_p \nabla\phi_q$.
Ainsi que $\forall p \in [\![1, N_i]\!] \quad \tilde{f}_p = \int_\Omega f\phi_p$.
On écrit $\tilde{\mathbf{u}} = (\tilde{u}_1, \ldots, \tilde{u}_N)^T$.

$$\tilde{\mathbb{K}}\mathbf{u} = \tilde{\mathbf{f}} \tag{5}$$

Prise en compte dans le code (lors de l'assemblage et de la pseudo-élimination) du fait que l'on résout sur u et non $\tilde{u}$. En notant u_p la valeur de u au noeud p et le vecteur $\mathbf{u} = (u_1, \ldots, u_N)^T$ avec N le nombre total de points ($N = N_i + N_b$ avec N_b le nombre de points sur le bord) le système s'écrit alors :

$$\mathbb{K}_{elim}\mathbf{u} = \mathbf{f}_{elim} \tag{6}$$

Définition des variables de parallélisation

- myRank : le rank du processeur sur lequel on travaille
- nbTask : le nombre de tâches égal au nombre de partitions + 1

C'est le processeur 0 qui s'occupe de l'interface

Organisation du découpage :

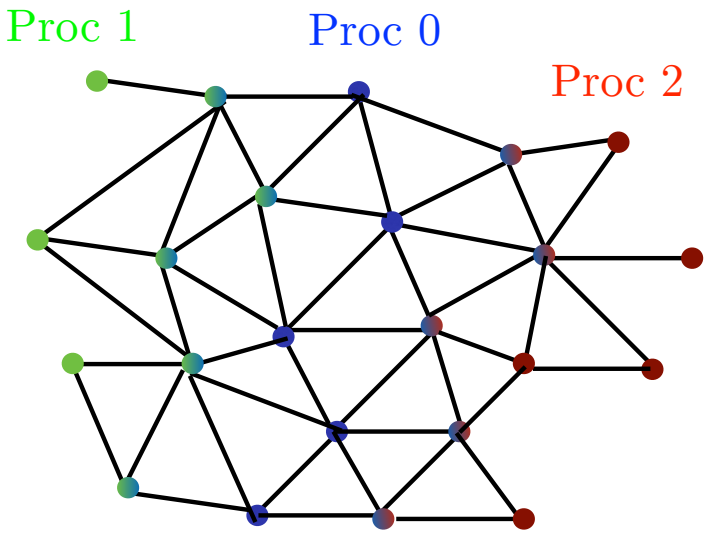

FIGURE: Schéma du découpage entre processeurs

Definition des vecteurs et matrices sur chaque processeur :

- Définition des vecteurs de taille globale N
- Définition des matrices *sparse* de taille globale N

Problèmes qu'il nous a fallu résoudre au préalable :

- Construction de tableau étiquetant les noeurs sur chqaue proc
- Attribution à chaque processeur de ses triangles et noeuds
- Initialisation/communication des listes de noeuds à communiquer
- Pseudo-elimination sur l'ensemble des noeuds du bord
- Nettoyage des lignes incompètes de $\mathbb{K}_{elim}$
- Suppression des coefficients parasites de $\mathbf{f}_{elim}$

Algorithme avec communications :

1: $M = diag(\mathbb{K}_{elim})$ et $N = -\mathbb{K}_{elim} + M$
2: **for** $k = 1, 1000$ **do**
3: $\quad \mathbf{u}_k = M^{-1} N \mathbf{u}_k + M^{-1} \mathbf{f}_{elim}$
4: $\quad$ MPI_BCAST$(p_0 \rightarrow p_{\text{myRank}\neq 0},\ \mathbf{u}_{k|\text{interface}})$
5: $\quad$ **if** myRank $== 0$ **then**
6: $\quad\quad$ **for** $i = 1,$ nbTask-1 **do**
7: $\quad\quad\quad$ MPI_RECV$(p_i \rightarrow p_0,\ \mathbf{u}_{k|\text{voisins interface}})$
8: $\quad\quad$ **end for**
9: $\quad$ **else**
10: $\quad\quad$ MPI_SEND$(p_{\text{myRank}} \rightarrow p_0,\ \mathbf{u}_{k|\text{voisins interface}})$
11: $\quad$ **end if**
12: $\quad$ Calcul de la norme du residu : $\|\mathbf{r}_k\|$
13: **end for**

14: Reconstruction de la solution sur p_0

Calcul de la norme du résidu :

1: ...
2: $\mathbf{r}_k = \mathbb{K}_{elim}\mathbf{u}_k - \mathbf{f}_{elim}$
3: $S = (\mathbf{r}_k, \mathbf{r}_k)$
4: MPI_ALLREDUCE($p_{\text{myRank}} \rightarrow p_0$, MPI_SUM, S, Res : S_t)
5: $\|\mathbf{r}_k\| = \sqrt{S_t}$
6: **if** $\|\mathbf{r}_k\| < \epsilon\|\mathbf{r}_0\|$ **then**
7: EXIT
8: **end if**
9: ...

Reconstruction de la solution :

1: ...
2: MPI_REDUCE($p_{\text{myRank}} \rightarrow p_0$, MPI_SUM, $\mathbf{u}_{k|\text{myRank}}$, Res : $\mathbf{u}$)

Frame de test

Visualisation de la solution du ParaView

Comparaison des temps de calcul sur gin :