TP2 - Résolution de l'équation de Laplace par une méthode stochastique

Marion Pavaux
Erwan Lambert

Le 29 Avril 2021

1 Introduction

Notre démarche a été de séparer le travail entre tous les processeurs de la manière suivante :

1. A chaque processeur est associé des bornes de calcul.
2. Chaque processeur initialise sa propre grille avec les conditions de bord. On aurait pu aussi les faire initialiser chacun une partie de grille et faire communiquer les processeurs ensuite entre eux, mais il a été estimé que l'initialisation de chaque processeur était proche de l'initialisation séparée avec mise en commun.
3. Puis chaque processeur se répartie le nombre de colonnes de la grille pour faire les calculs des éléments de celle-ci.
4. Une fois les calculs faits les processeurs communiquent entre eux et somment les éléments du tableau sur le processeur 0, qui affiche la grille complète.
5. L'écriture du fichier de sortie se fait en séquentiel

1.1 Démarche

Nous avons choisi d'automatiser la soumission des différents job sur des nombre de coeurs différents. Pour ce faire nous avons utilisé un script Python. Il copie la template du job et change uniquement le nombre de noeux et le nombre de tâche par noeux.Chaque job fait intervenir le programme monte_carlo_parallel.py sur un nombre de noeux différent.

Les principales difficultés ont été d'ajuster le temps limite qui est très différent en fonction du nombre de noeux utilisés. Nous avons mis du temps à identifier ce problème, et nous avons d'abord pensé, que les problèmes venaient de l'écriture des résultats, du temps de calcul dans un fichier commun à toutes les simulations.

Nous avons d'abord voulu ne retenir que le temps de calcul du noeud de rang 0, mais nous avons remarqué que le résultat n'était pas toujours retourné. Nous

avons donc choisit de permettre à tout les noeux d'écrire leur résultat dans le fichier commun afin d'être sûrs d'avoir au moins un temps de calcul. Ceci nous a permis de vérifier que les temps de calcul était très proche d'un noeud à l'autre pour un nombre donné de processeurs utilisés.

2 Speed-up et Efficacité

Le programme a mis les temps suivants (sans compter l'écriture) à faire les calculs :

Exécution	Temps d'exécution(s)	Speed up(10^{-3})	Efficacité(10^{-3})
Séquentiel	1.082e+03	-	-
1 coeur	2.879105e+03	56	56
2 coeurs	2.948210e+03	54	27
4 coeurs	5.480782e+02	291	73
8 coeurs	3.138817e+02	510	54
16 coeurs	1.729616e+02	925	58
32 coeurs	9.904873e+01	1061	50

Nous avons ensuite tracé les courbes du speed-up et de l'éfficacité :

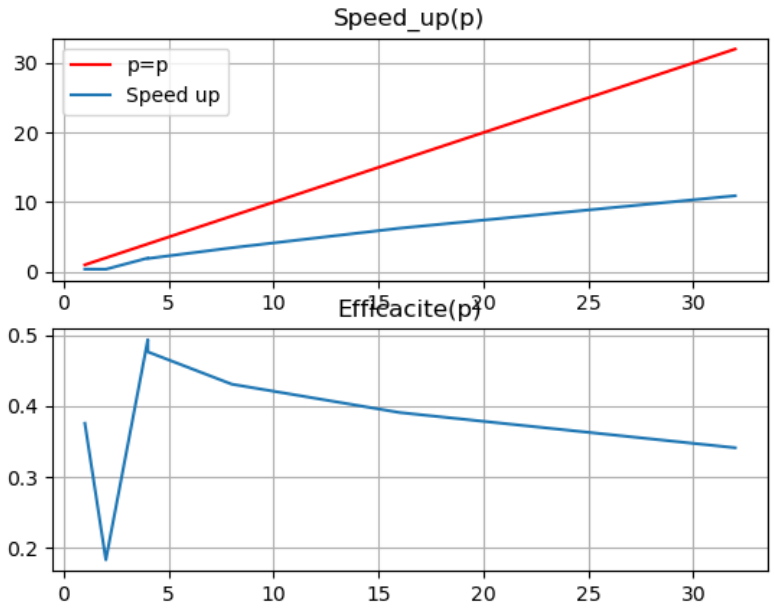

FIGURE 1: Speed up et Efficacité en fonction du nombre de processeurs

Nous avons bien la courbe de Speed up qui est en dessous de la courbe $y = p$, même si celle-ci ne se confond pas malheureusement avec celle-ci.

3 Résultat du problème

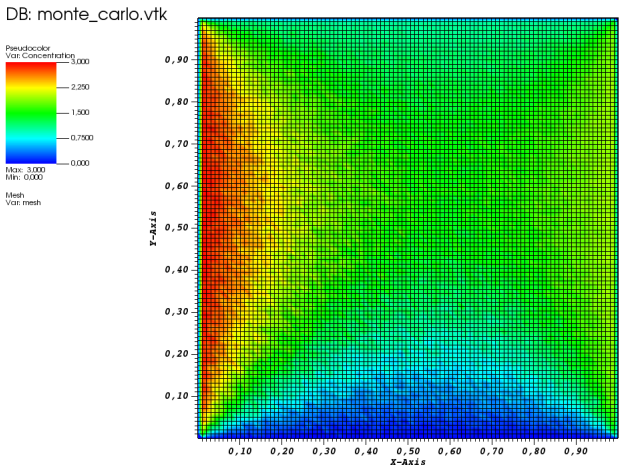

FIGURE 2: Affichage avec grille