

Projet CHPS0801 : Modèles de programmation parallèle

Yannis Ondars
Pierre-Antoine Raclius

Mai 2023

Table des matières

1	Introduction	3
	Algorithme de Gauss-Seidel	3
	Objectifs et Solutions	3
2	Algorithme de Gauss-Seidel et OpenMP	4
	Algorithme Séquentiel	4
	Algorithme Parallèle	4
	Première version "naïve"	5
	Seconde version "Threads + Tâches"	5
	Troisième version : Taches plus conséquentes	5

1 Introduction

Algorithme de Gauss-Seidel

//Explication



FIGURE 1 – Illustration du problème de Langford

Objectifs et Solutions

2 Algorithme de Gauss-Seidel et OpenMP

Algorithme Séquentiel

Pour réaliser l'algorithme de Gauss-Seidel, le programme va être divisé en 3 étapes. Le format portrait n'étant pas totalement au point, nous prendrons le cas d'une image en format paysage.

En premier lieu, l'algorithme séquentiel va parcourir les diagonales se situant avant le coin inférieur gauche de l'image (En bleu)

Ensuite, les diagonales qui se situent entre les coins inférieurs gauches et supérieurs droits, donc les plus grandes diagonales de l'image, vont être traitées (En vert)

Et enfin, les dernières diagonales, allant du coin supérieur droit au bout de l'image (En Rouge) Par ailleurs, ce système de répartition des tâches va être réutilisé pour la version parallélisée avec OpenMP du projet.

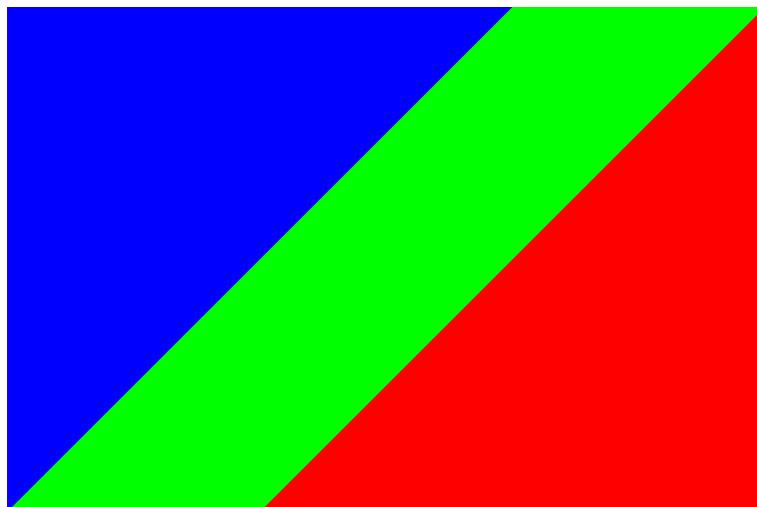


FIGURE 2 – Illustration des 3 étapes de traitement de l'image

Pour les tests, nous allons utiliser une image au format portrait d'une taille de 4000*3000 pixels. Les résultats de l'algorithme seront enregistrés dans le dossier res du projet.



FIGURE 3 – Avant le Filtre



FIGURE 4 – Après le filtre

Algorithme Parallèle

Première version "naïve"

En guise de première version que l'on pourrait dire "naïve", nous avons décidés, en raison de l'impossibilité de parallélisé les diagonales entre elles (vu qu'elles sont inter-dépendantes), d'affecter une tache à un pixel. Étant donné le nombre de taches créées (égal au nombre de pixels sur une diagonale), les performances sont assez faibles comparé à l'algorithme séquentiel.

Seconde version "Threads + Tâches"

Ensuite, en guise de seconde version, nous avons partagé chaque diagonale entre n threads, puis chacun de ces threads créé une tache par pixel présent sur sa partie de diagonale à traiter.

Troisième version : Taches plus conséquentes

Enfin, pour la troisième version de la parallélisation, nous avons opté pour créer des taches plus conséquentes : Cette fois si les diagonales seront divisées en un nombre de taches prédéfinies. Et chacune de ces taches devra donc gérer $\text{nbPixel}/\text{nbcTaches}$ pixels.

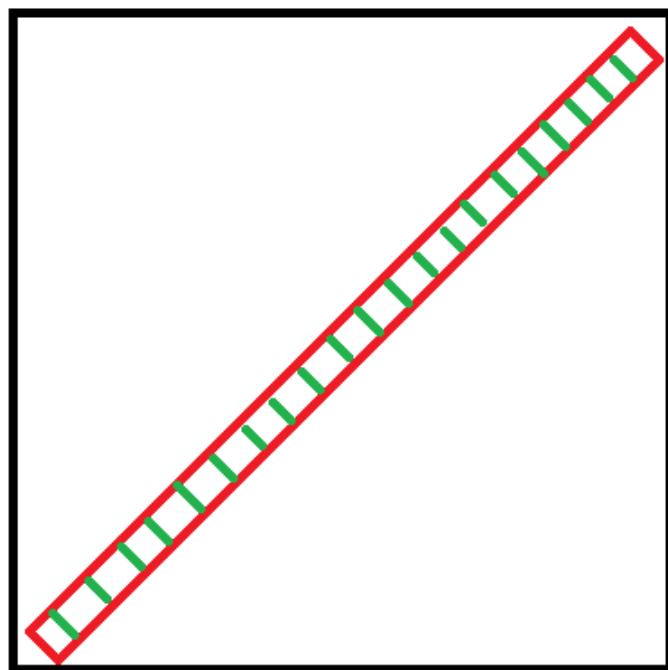


FIGURE 5 – Illustration du découpage d'une diagonale