

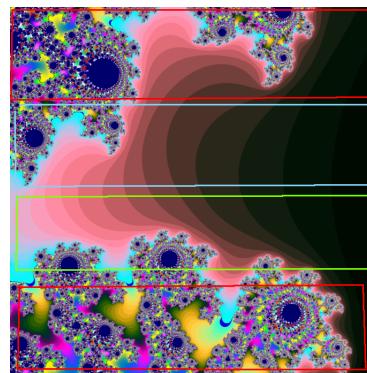
HPC - TME 2

Diez Marie

1 Parrallélisation

1.1 Bloc par Bloc

Question 1. Pour parralléliser le programme séquentiel du calcule de l'ensemble de mandelbrot on peux commencer par dire aux processeurs de gérer bloc par bloc les lignes.



Question 2.

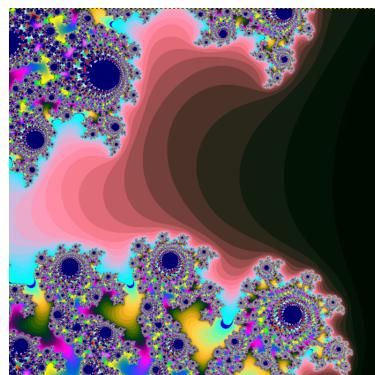
Voici mon code : https://github.com/MarieDiez/HPC_multiThreading/blob/main/mandel_paraBloc.c

Il faut faire attention à un point particulier : $N \% P! = 0$, dans ce cas là il faut dire au dernier processeur de faire le reste de lignes en plus $nb_lines+ = h \% size$ avec h le nombre de lignes et $size$ le nombre de processeurs, il faut aussi indiquer au processus 0 qui récupère les lignes que le dernier processus va lui en envoyer plus.

Remarque : J'ai envoyé ligne par ligne les blocs au processus 0, il pourrait être avantageux de limiter les communications et d'envoyer directement le bloc au processus. J'ai mise cette étape en place dans la version d'équilibrage de charge dynamique avec le modèle patron/ouvrier.

Question 3. Résultat et Temps d'execusion

Voici l'image obtenu:



Temps d'exécution :

Soit p , le nombre de processeurs.

- $p = 1$ (Version séquentielle) :

Temps total de calcul : 8.77915 sec

- $p = 3$:

Processeurs 1 : Temps total de calcul : 5.04614 sec

Processeurs 2 : Temps total de calcul : 7.60301 sec

Temps total de calcul : 7.60303 sec

- $p = 8$:

Processeurs 3 : Temps total de calcul : 2.4431 sec

Processeurs 4 : Temps total de calcul : 2.46956 sec

Processeurs 2 : Temps total de calcul : 2.71493 sec

Processeurs 1 : Temps total de calcul : 3.06797 sec

Processeurs 5 : Temps total de calcul : 3.1604 sec

Processeurs 6 : Temps total de calcul : 3.52707 sec

Processeurs 7 : Temps total de calcul : 3.75633 sec

Temps total de calcul : 3.75637 sec

On remarque que les temps de calculs ne sont pas parfaitement réparties.

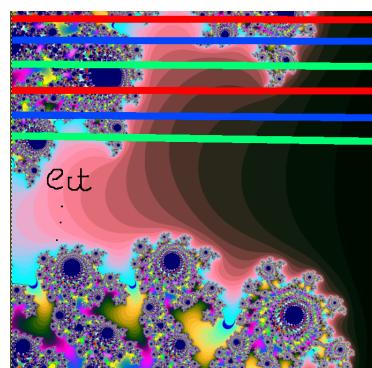
Question 4.1 Cette méthode de parallélisation fonctionne, mais n'est pas parfaite, en effet si dans l'image à calculer il y a une portion qui demande beaucoup plus de calculs qu'une autre, alors le processeur qui sera chargé de cette portion sera plus long, et les autres resteront alors inactifs. Nous devons alors penser à un mettre en place un équilibrage des charges.

2 Equilibrage des charges

2.1 Statistiques

2.1.1 Ligne par Ligne

Dans un premier temps, on peut mettre en place un système où chaque processeur calcule chacun k lignes comme au paravant, cependant les lignes ne doivent pas être contiguës, en effet cela va permettre d'améliorer la charge de travail étant donnée qu'on aura moins de chance de tomber sur une portion entière plus longue à calculer. Pour cela j'ai mis en place un système où chaque processeur dans l'ordre les lignes



Voici mon code :

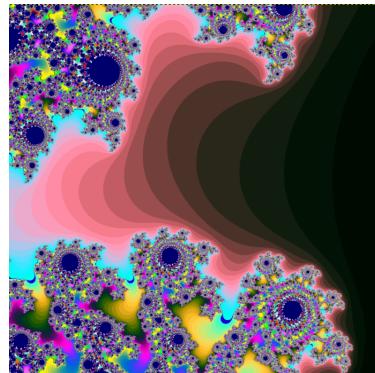
https://github.com/MarieDiez/HPC_multiThreading/blob/main/mandel_paraLigne.c

Il faut faire attention à un point particulier comme précédemment : $N \% P! = 0$, dans ce cas j'ai dis au processus 0 de finir le traitement, ce qui correspond le reste de lignes $nb_lines+ = h \% size$ avec h le nombre de ligne et $size$ le nombre de processeurs. Il traitera ces lignes séquentiellement.

Remarque : J'ai envoyé ligne par ligne les blocs au processus 0, il pourrait être avantageux tout comme précédemment de limiter les communications et d'envoyer directement le bloc au processus, dans ce cas le traitement s'avère plus délicat pour le réglage des indices.

Résultat et Temps d'exécution

Voici l'image obtenu:



Temps d'exécution :

Soit p , le nombre de processeurs.

- $p = 3$:

Processeurs 1 : Temps total de calcul : 5.97234 sec
Processeurs 2 : Temps total de calcul : 5.97234 sec
Temps total de calcul : 5.97237 sec

- $p = 8$:

Processeurs 1 : Temps total de calcul : 2.62024 sec
Processeurs 2 : Temps total de calcul : 2.62599 sec
Processeurs 3 : Temps total de calcul : 2.62743 sec
Processeurs 4 : Temps total de calcul : 2.62738 sec
Processeurs 5 : Temps total de calcul : 2.62693 sec
Processeurs 6 : Temps total de calcul : 2.62804 sec
Processeurs 7 : Temps total de calcul : 2.62806 sec
Temps total de calcul : 2.62811 sec

On remarque que la charge de travail est beaucoup mieux répartie.

Question 4. Cependant il serait encore possible qu'une ligne demande beaucoup plus de travail qu'une autre, l'équilibrage reste alors imparfait. On va alors mettre en place un équilibrage des charges dynamiques.

2.2 Dynamique

2.2.1 Modèle Patron / Ouvrier (Maître/Escalve)

Question 5. Le modèle Patron / Ouvrier permet de faire un équilibrage des charges dynamiquement, voici une petite histoire expliquant le principe de fonctionnement:

(Patron) - Bonjouurs petits ouvriers, je vais vous affetez votre travaille, revenez vers moi quand vous aurez terminé. Je vous donnez chacun **k** lignes à traiter.

(Ouvrier 1) - Bonjour patron, j'ai terminé mon travail, voici la portion de l'image que j'ai calculé.

(Patron) - Merci, je vais l'ajouter à mon image globale. Puisque tu as fini et qu'il reste du travail à faire voici les **k** lignes suivantes.

(Ouvrier 1) - Ok, je m'y mets.

(Ouvrier 2) - Bonjour patron, j'ai terminé mon travail, voici la portion de l'image que j'ai calculé.

(Patron) - Merci, je vais l'ajouter à mon image globale. Puisque tu as fini et qu'il reste du travail à faire voici les **k** lignes suivantes.

Et ainsi de suite jusqu'à ce qu'il n'y ai plus de travail à faire.

De cette manière aucun processus ne restent trop longtemps inactifs. Il faut cependant faire attention à plusieurs points :

- Choisir une quantité de travail / un nombre de ligne **k** adéquat, il ne faut pas trop petit sinon les temps de communications seront trop important par rapport aux temps de travaux, ni trop grand pour pouvoir utiliser tous les processeurs et qu'ils calculent en parallèle efficacement.
- Si il reste moins de ligne à traiter que ce qui était prévu, c'est-à-dire si $N\%P! = 0$ il faut donner le reste au dernier processus et non ce qui était prévu initialement.

Voici mon code :

https://github.com/MarieDiez/HPC_multiThreading/blob/main/mandel_para.c

Temps d'exécution :

Soit p , le nombre de processeurs.

- $p = 3$:

Processeurs 1 : Temps total de calcul : 4.73623 sec
Processeurs 2 : Temps total de calcul : 4.73625 sec
Fin du maître : 4.73611 sec

- $p = 8$:

Processeurs 1 : Temps total de calcul : 1.54045 sec
Processeurs 2 : Temps total de calcul : 1.54021 sec
Processeurs 3 : Temps total de calcul : 1.54041 sec
Processeurs 4 : Temps total de calcul : 1.54048 sec
Processeurs 5 : Temps total de calcul : 1.54039 sec
Processeurs 6 : Temps total de calcul : 1.54049 sec
Processeurs 7 : Temps total de calcul : 1.54038 sec
Fin du maître : 1.54043 sec

On remarque que pour peu de processeurs, il n'est pas intéressant de passer par le modèle Patron/Ouvrier il y a peu besoin d'équilibré la charge de travail vu qu'il y a peu de processeurs, le cout de mise en place est alors trop élevé par rapport au temps de calcule. Cependant pour un nombre de processeurs plus élevé il est très intéressant de passer par ce modèle, les temps de calcule sont très faibles, en effet il n'y a pas d'attente passive des processus comme dans les derniers modèle vu précédemment et de plus la charge de travail est très bien équilibrée.

Recherche des paramètres optimaux :

Pour $p = 12$ processeurs je trouve un temps minimale pour environ $k=25$ lignes par processeurs, en effet la valeur n'est pas trop faible ce qui n'indue pas de cout de communication trop important, ni trop élevé pour profiter au maximum de la parrallélisation.

Résultats Voici différents résultats obtenus :

