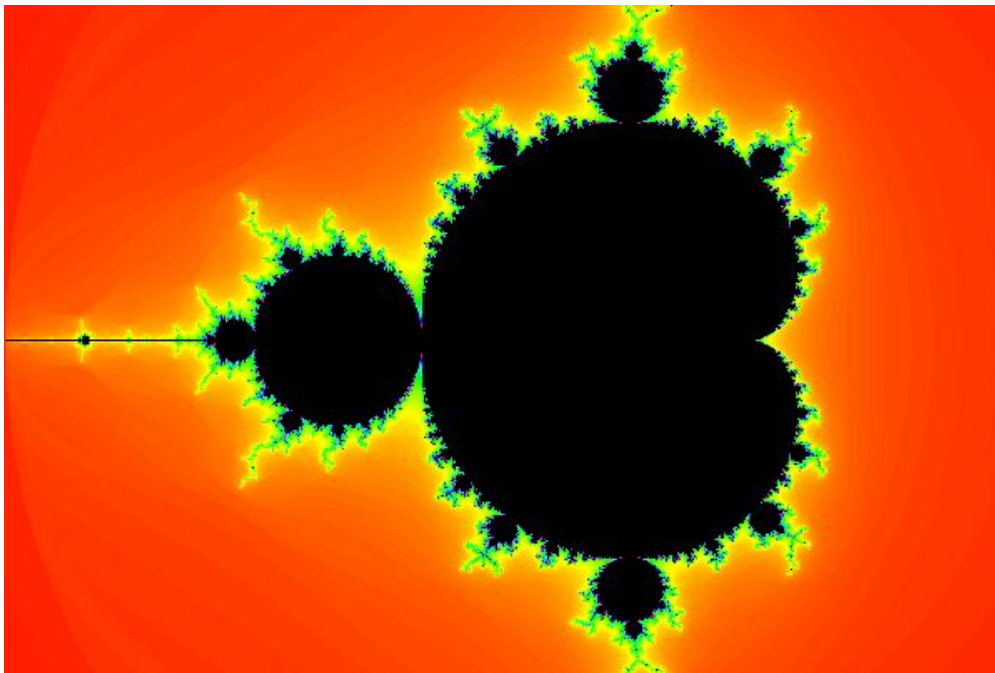


# Mandelbrot

## TP - C++

---



BOUCHAIN Loïc

DEZAT Valentin

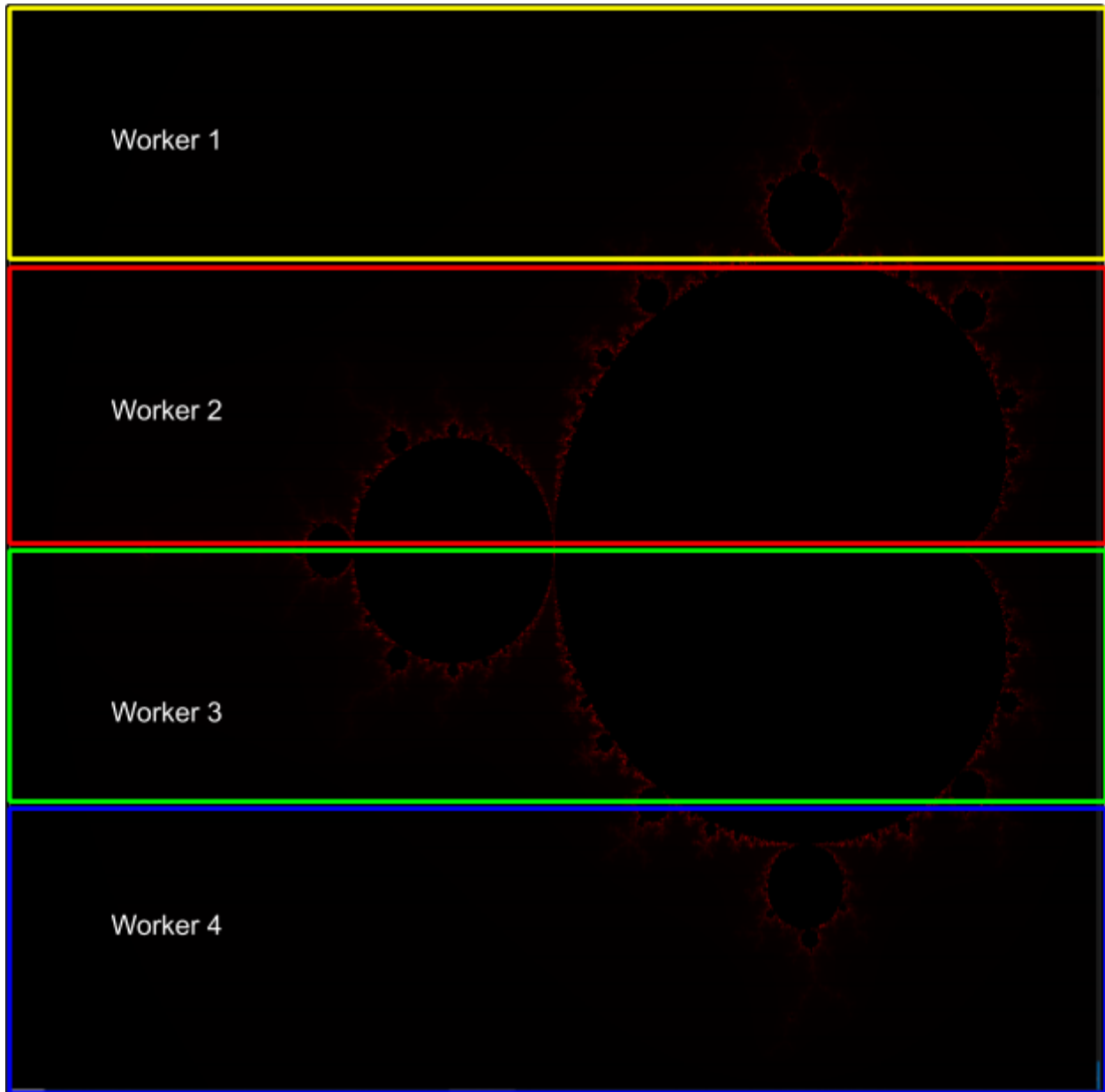
B3 Ynov 2019-2020


Les threads sont-ils aussi rapides les uns que les autres? Le travail est-il bien réparti?

Non, le travail n'est pas bien réparti, en effet les workers du milieu ont un temps de traitement plus long que les workers des extrémités.

A quelle amélioration du temps de calcul avez-vous abouti? Est-il grossièrement divisé par 4? Pourquoi d'après vous?

Le temps de calcul n'est pas bien réparti entre les 4 workers :





Cela peut s'expliquer par la charge de calcul de chaque worker, grâce à l'image on peut voir globalement les lignes affectées pour chaque worker, ainsi on peut voir que la charge de calcul est beaucoup plus importante pour les workers 2 et 3 que pour les workers 1 et 4 du fait du nombre de lignes à afficher.

Quelle méthode avez-vous choisie pour améliorer la performance et mieux répartir le travail? Pourquoi? Dans les grandes lignes, comment l'avez vous implémentée?

Certains workers en fonction de l'emplacement de leur tranche peuvent avoir plus ou moins de travail, en effet les tranches sur les extrémités du mandelbrot auront moins de lignes à modifier et donc moins de calculs.

L'objectif est donc de mieux répartir les tranches de calculs pour chaque worker. Nous avons donc commencé par définir le travail d'un worker en 2 tranches de calculs au lieu d'une.

Un mandelbrot étant symétrique, nous avons positionné les tranches différemment sur chaque moitié du mandelbrot afin de mieux dispatcher le travail, ainsi un worker ayant une tranche à l'extrémité du mandelbrot en aura une autre au milieu.



Voici un exemple de notre méthode avec 4 workers.

Gains ressentis :

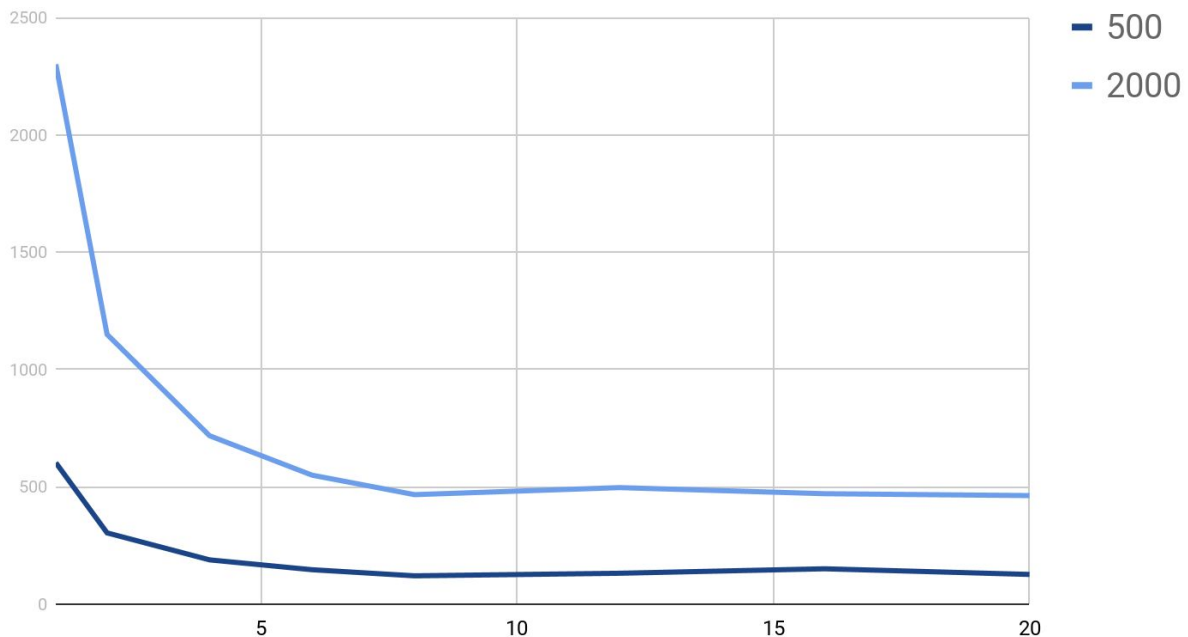
Le gain de performance est notable, avec 4 workers et sur un ordinateur intel core i5

Méthode	Worker 1 (en ms)	Worker 2(en ms)	Worker 3(en ms)	Worker 4(en ms)	Total (temps de traitement)
Ancienne	51	543	541	51	543
Notre méthode	345	275	281	373	373

Ainsi on peut observer un temps de calcul mieux dispatché et un gain total de 133 ms soit un gain de 69% de performance.

Comment se comporte la courbe? Quelles sont les tendances qui se dessinent indépendamment du nombre d'itérations maximales? D'après vous, comment se justifient ces tendances?

Temps de traitement en fonction du nombre de workers et du nombre d'itérations



On peut voir que dans les 2 cas, le temps de traitement est décroissant jusqu'au 7ème worker. A partir du 7ème worker le temps stagne et ne décroît plus.

Cela peut s'expliquer par le fait que la charge de travail est déjà bien répartie et que les temps de traitement de chaque worker ne peuvent être baissés.