# Clustering : calcul matriciel et vectoriel – quels sont les bénéfices de la programmation concurrentielle?

Par Kevin Mwanangwa, Laurier Lavoie-Giasson et Chris David

## Introduction : quel a été le but des observations?

L’orientation prise lors de cette expérimentation a été orientée davantage vers le temps de calcul que vers la précision des résultats. Bien que la précision des calculs a été vérifiée, davantage de temps a été consacré à comparer les temps que les résultats.

## Questions posées

L’expérimentation a permis de répondre à plusieurs interrogations concernant les différences de temps de calcul en utilisant la programmation en parallèle. Est-il très long de démarrer des threads pour calculer? Est-il simple de le faire? Aussi, avant de commencer à réécrire le programme en sa version en parallèle, la question de l’écriture dans la même zone mémoire s’est posée. Est-ce que deux threads qui tentent d’écrire ou de faire des opérations avec la même valeur en mémoire vont générer une erreur?

## Premier blocage : le GIL (*Global Interpreter Lock*)

Le premier obstacle à la programmation concurrentielle rencontré est le *Global Interpreter Lock*. Qu’est-ce que le *Global Interpreter Lock*?

«In CPython, the global interpreter lock, or GIL, is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes at once. This lock is necessary mainly because CPython's memory management is not thread-safe. (However, since the GIL exists, other features have grown to depend on the guarantees that it enforces.) »

Python.org

En effet, cela rend théoriquement la programmation concurrentielle impossible en CPython. Il aurait été très facile d’abandonner, mais