BE SXS HPC

ACCOU Martin, LALONNIER Lucas

Q 1.1

OK.

Q 1.2

Pour notre exemple, on a :

- Temps d'initialisation : 0.015s;
- Temps d'assignation : 9.256s;
- Temps de calcul : 0.943s.

La tâche la plus longue est l'assignation, suivie du calcul et enfin de l'initialisation.

Il est normal que l'initialisation soit clairement la tâche la plus rapide, puisque on choisit simplement n positions de centroïde aléatoirement.

Il est normal que l'assignation soit beaucoup plus coûteuse que le calcul des points moyens : pour chaque point, on doit calculer la distance à chacun des autres points, alors que le calcul des points moyens est de l'ordre du nombre de points.

Q 1.3

Notons d'abord qu'il est inutile de paralléliser l'initialisation, même s'il est théoriquement possible de le faire.

L'assignation se parallélise facilement car l'assignation d'un point i est indépendante de l'affectation d'un point j tel que i est différent de j.

A priori, la parallélisation du calcul des points moyens est possible car elle repose sur un calcul de barycentres qui peut se paralléliser par réduction.

Q 2.1

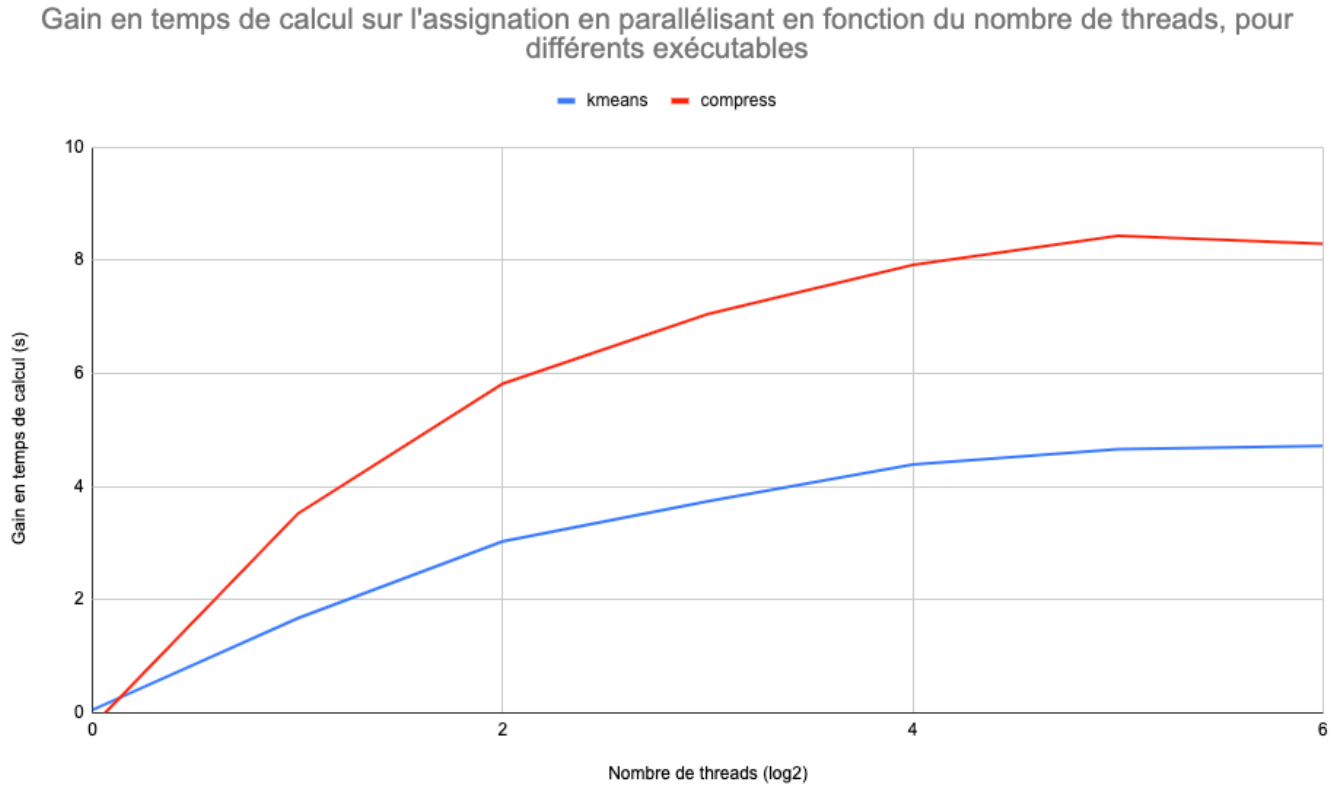
On peut d'abord remarquer que la boucle intéressante à paralléliser est la boucle sur le nombre de points, car c'est celle qui comporte le plus d'itérations.

L'utilisation de la clause *pragma omp for* ne nous fournit pas des résultats cohérents au premier abord.

Il faut en fait rajouter une clause *pragma omp atomic write* qui permet de gérer proprement l'accès à la variable externe *false*.

Q 2.2

En mesurant le gain de temps par rapport au cas où on n'emploie pas la parallélisation, on obtient le graphe suivant :



Comme on pouvait s'y attendre, le gain en temps est bien croissant en fonction du nombre de threads utilisés.

Cette tendance est moins vraie à mesure que le nombre de threads augmente, car le gain obtenu par la parallélisation est compensé négativement par overhead.

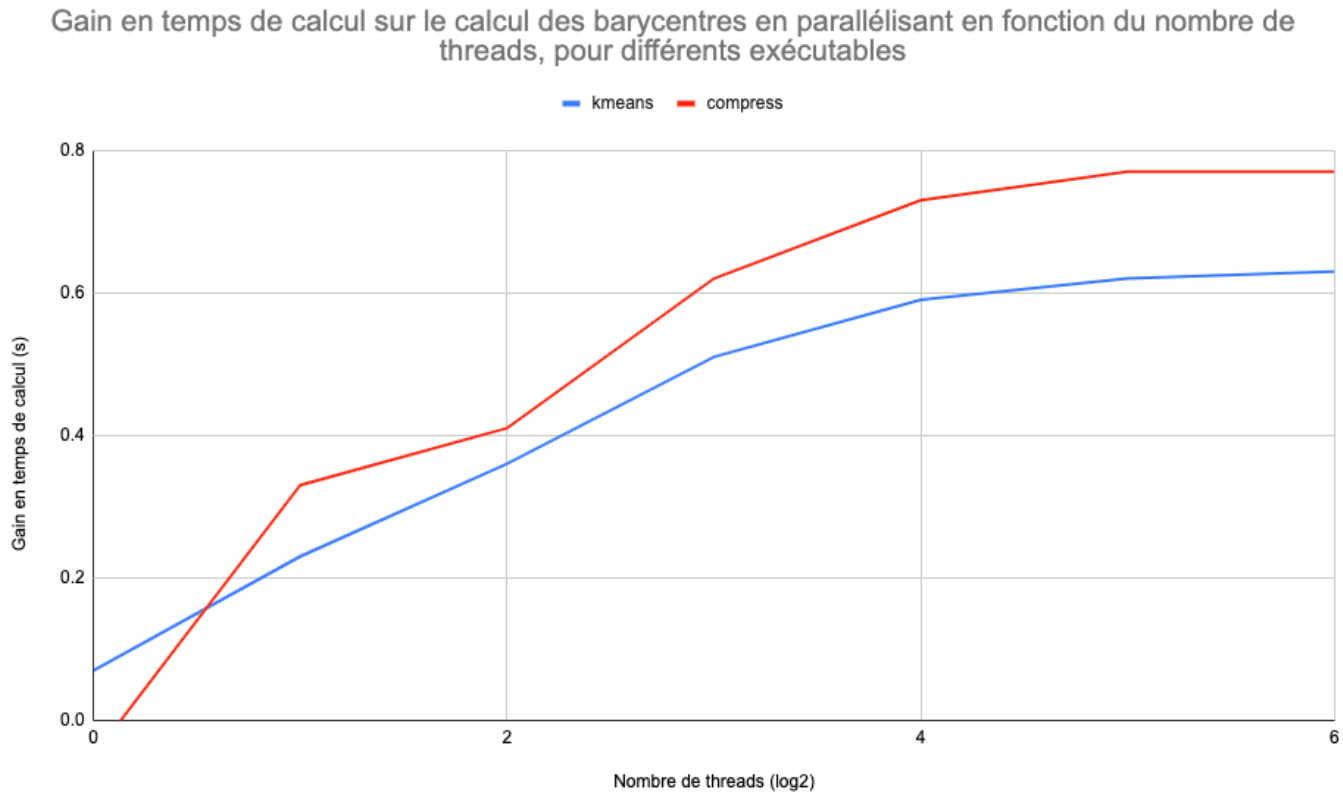
Q 3.1

Notons encore une fois que la boucle intéressante à paralléliser ici est celle qui porte sur n.

La difficulté réside ici dans la réduction appliquée à des vecteurs, et il s'avère qu'il existe un enchaînement de clauses OpenMP qui permet de solutionner ce problème, au lieu de recoder à la main la réduction.

Q 3.2

En mesurant le gain de temps par rapport au cas où on n'emploie pas la parallélisation, on obtient le graphe suivant :



Les mêmes conclusions peuvent être tirées.

On peut se réjouir de l'excellente amélioration globale des résultats, permise par la parallélisation de l'affectation des points ainsi que du calcul des barycentres.