Rapport d'Energie

Poitelea Mihai

December 2023

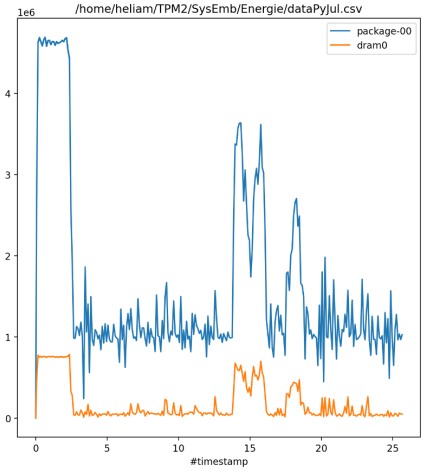
1 Résultats

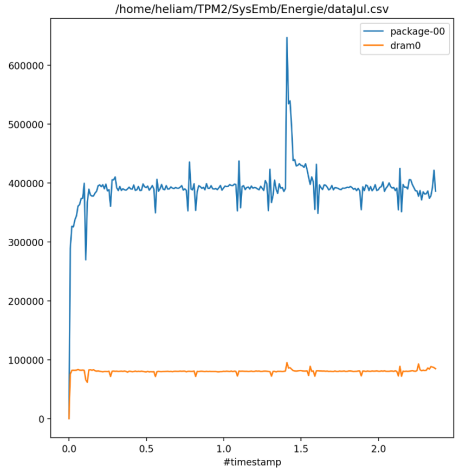
Langage	taille	iter	Ligne commande	Ligne compilation	Fréq	Temps	Energie	Puissance
python	900	100	python3 julia.py	--	3.9GHz	62.41s	712	11.41
C++	900	100	./julia	g++ -std=c++20 -Wall -Wextra -pedantic julia.cpp -o julia -lSDL2main -lSDL2	3.9GHz	0.6	2633	4362
C++	10000	1000	./julia	g++ -std=c++20 -Wall -Wextra -pedantic julia.cpp -o julia -lSDL2main -lSDL2	3.9GHz	211.46	1787	8.5

2 Analyse

Le premier essai que nous avons réalisé est en python. Le temps était un peu trop élevé et la consommation assez instable.
Nous avons très vite cherché un autre langage qui serait plus rapide et si possible qui consommerait moins.
Après avoir fait des essais sur différents langages(C, julia,...), nous avons donc choisi d'utiliser C++ qui est beaucoup plus rapide que python et car nous sommes plus familiarisés à C++ qu'a d'autres potentiels langages.

Voici les deux graphes représentant la consommation et le temps d'éxecution pour python à gauche et C++ à droite, avec une taille de 900 et pour 100 itérations:





On peut voir un énorme pic de consommation avec python au début de l'éxecution, elle est surement due à la création de notre tableau au début du programme avec le np.zeros.
Une manière d'optimiser notre code aurait pu être de ne pas allouer tout notre tableau au début du programme et de faire une allocation dynamique.
En C++ on peut voir que la consommation est plus régulière hormis un pic vers le milieu de l'éxecution, et que celle si est beaucoup plus rapide.
Avec la version taille 10000 et 1000 itération, le temps est bien sur grandement augmenté mais la consommation elle est grandement diminuée. Je pense que cela se fait au moment de la compilation et que la Puissance est ajustée pour économiser de l'énergie.

3 Méthodes d'expérimentation

Afin d'experimenter avec ce programme, nous avons d'abord essayé plusieurs langages pour trouvé le meilleur pour notre cas d'utilisation.
Dans un deuxieme temps, nous avons effectué différents tests avec des tailles plus ou moins grandes, plus ou moins d'itérations, dans le but de faire varier la taille des boucles de notre programme.

4 Pour aller plus loin

Nous pourrions essayer d'optimiser encore plus notre programme en étudiant la variation de fréquence de notre processeur, pour essayer de moins consommer, ce qui pourrait être intéressant dans le cas de notre test C++ petite taille.
Nous pourrions aussi essayer d'optimiser nos boucles qui prennent beaucoup de temps et d'énergie, en essayant nottament de véctoriser nos boucle grâce à SIMD.