TD1 - OpenMP Manipulation Matrice Vecteur

Ce TD a consisté en l'optimisation de programmes de calcul sur des vecteurs et matrices à l'aide d'OpenMP. Voici les résultats des temps d'exécution de ces programmes et l'évaluation de performance résultante.

Les valeurs données sont les moyennes des temps sur 10 essais.

L'ordinateur utilisé pour l'évaluation de performances est un des nœuds « dahu » du site de Grenoble de Grid'5000 :
CPU : 2x Intel Xeon Gold 6130 à 16 cœurs/CPU, 192 Gio de RAM, Stockage : 240 Go SSD + 480 Go SSD + 4.0 To HDD, Réseau : 10 Gbps + 100 Gbps Omni-Path.

Dans les données de passage à l'échelle fort, la valeur avec 0 cœur correspond en réalité à une compilation sans l'option « -fopenmp », donc non parallélisée et s'exécutant sur un seul cœur.

La totalité de la réalisation du TD peut être retrouvée sur le GitHub public : https://github.com/Pyhro36/TD-1-OpenMP. La branche « master » a été la branche de travail. Les branches « question21, question22, question23 ... » sont les branches de rendu décrites ci-dessous. Les fichiers et méthodes ayant servis à l'évaluation des performances sont décrits dans le fichier « README.md » dans les branches correspondantes.

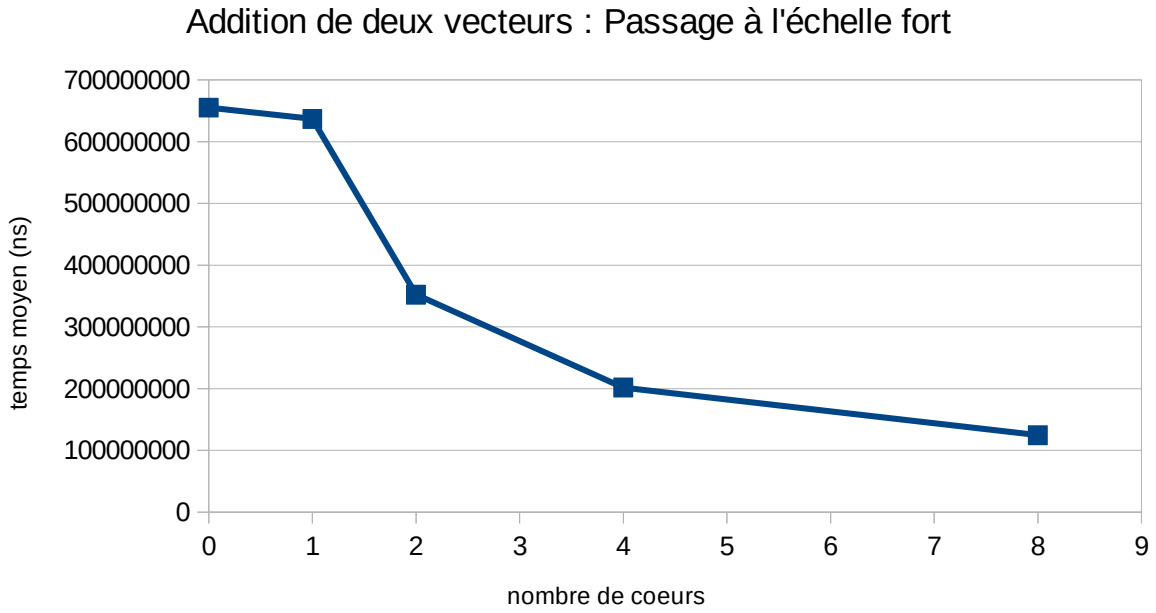
2.1 Vecteur-vecteur

Le code chaque question de cette partie est contenu sur une branche correspondante dans le GitHub : « question2X » pour la question 2.X (question 2.1 : question21, question 2.4 : question24, question 2.10 : question210…). La question 2.8 n'a pas de branche car elle n'a pas de signification. Les fichiers et méthodes concernés sont décrits dans le fichier « README.md », à la partie « 2.1 Vecteur-Vecteur ». Il suffit donc de faire un checkout de chaque branche pour accéder au code, dans le fichier « vecteur-vecteur/main_vect.cpp », les fonctions étant décrites dans « vecteur-vecteur/main_vect.h ».

Addition de deux vecteurs

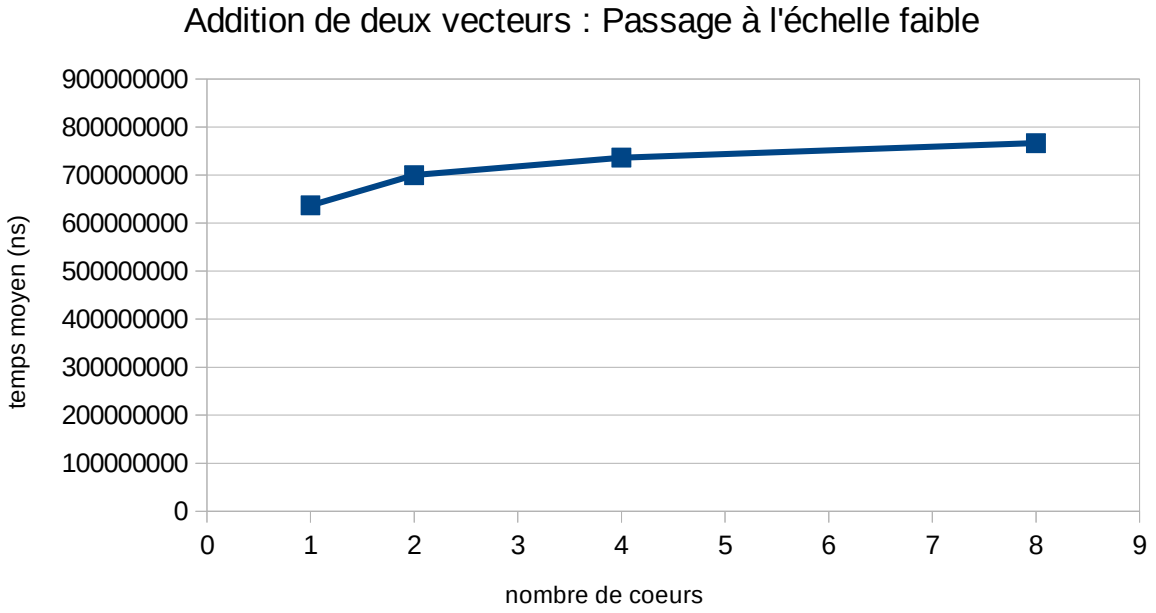
Passage à l'échelle fort : 100 000 000 termes

nombre de coeurs	0	1	2	4	8
temps moyen (ns)	655058308,4	636827934,6	352213094,4	201747465	124595296,5



Passage à l'échelle faible

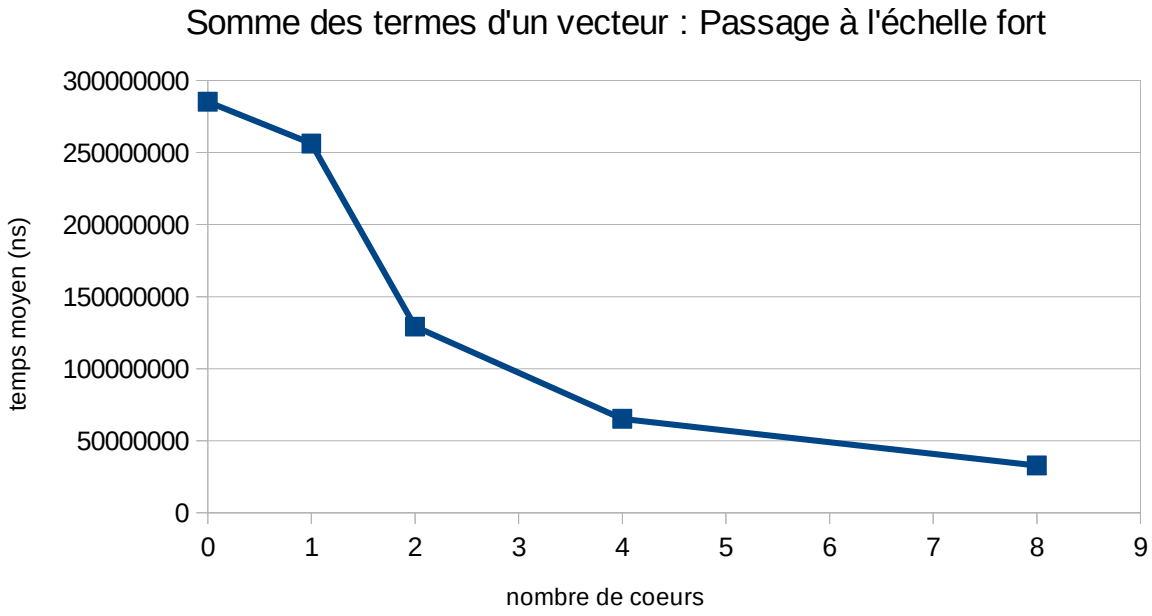
nombre de termes	100000000	200000000	400000000	800000000
nombre de coeurs	1	2	4	8
temps moyen (ns)	636827935	700005867,7	736360163,5	766518690,5



Somme des termes d'un vecteur

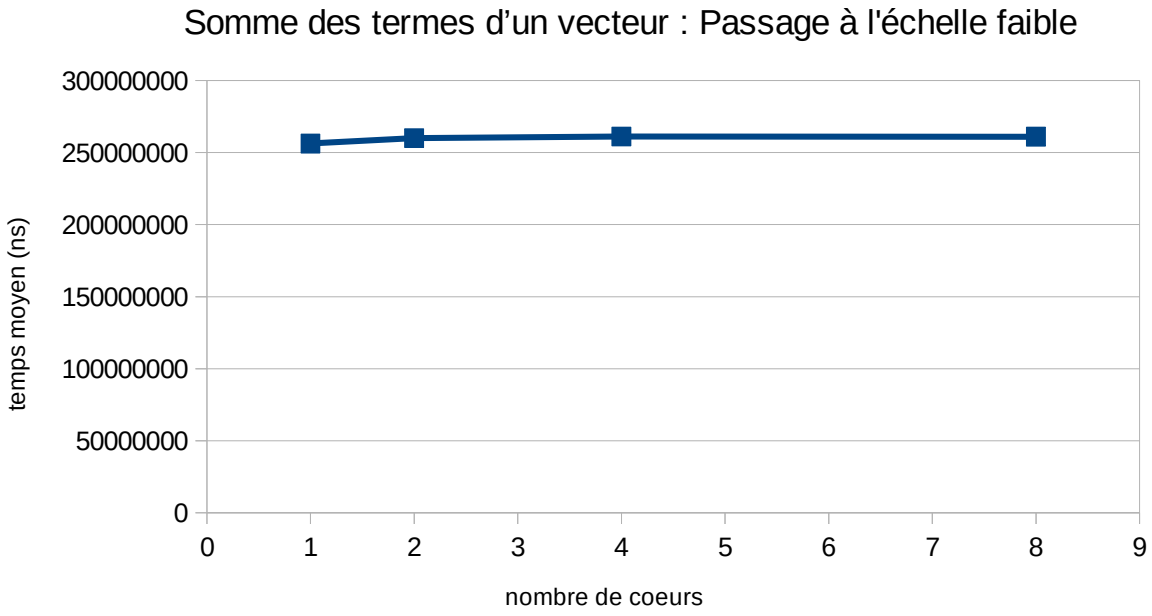
Passage à l'échelle fort : 100 000 000 termes

nombre de coeurs	0	1	2	4	8
temps moyen (ns)	285235055	256286726,4	129158747,7	65162311,5	32824434,8



Passage à l'échelle faible

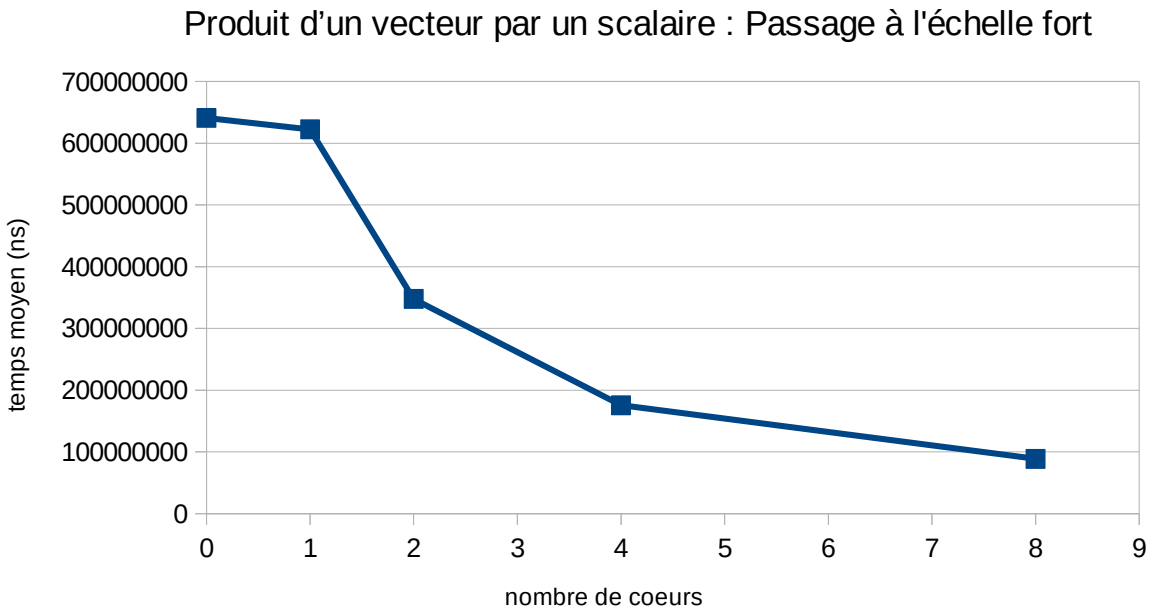
nombre de termes	100000000	200000000	400000000	800000000
nombre de coeurs	1	2	4	8
temps moyen (ns)	256286726	259985866,6	261111140,3	260976126,3



Produit d'un vecteur par un scalaire

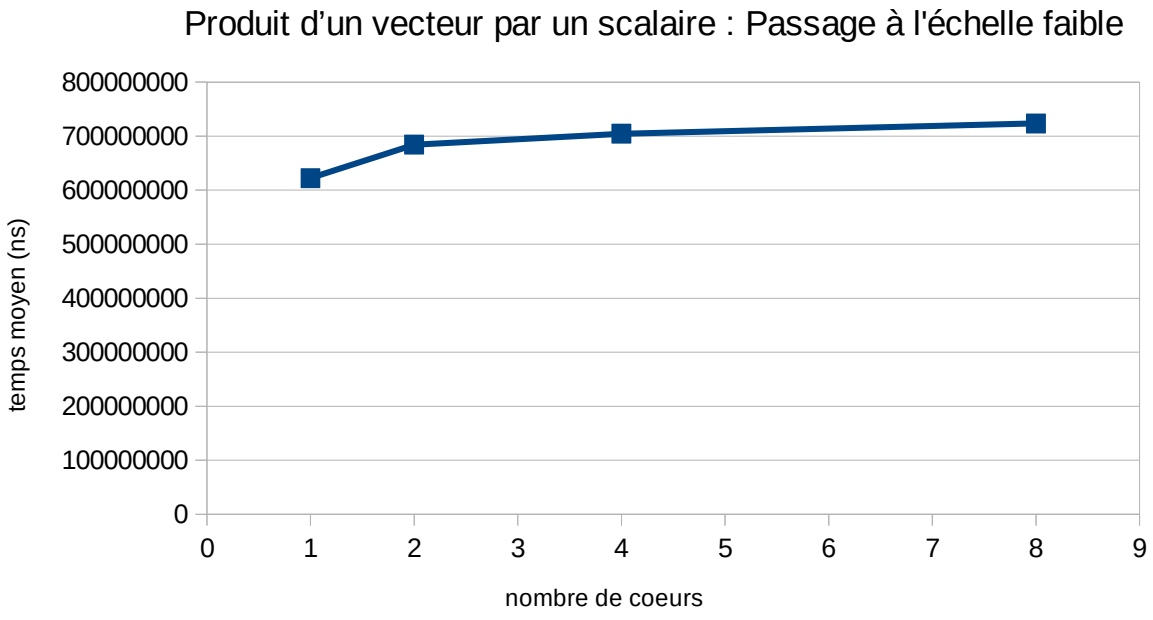
Passage à l'échelle fort : 100 000 000 termes

nombre de coeurs	0	1	2	4	8
temps moyen (ns)	640870684	622340529,2	347700285,3	175592209,4	88787369,8



Passage à l'échelle faible

nombre de termes	100000000	200000000	400000000	800000000
nombre de coeurs	1	2	4	8
temps moyen (ns)	622340529	684203538,1	704566602,1	723410431,1



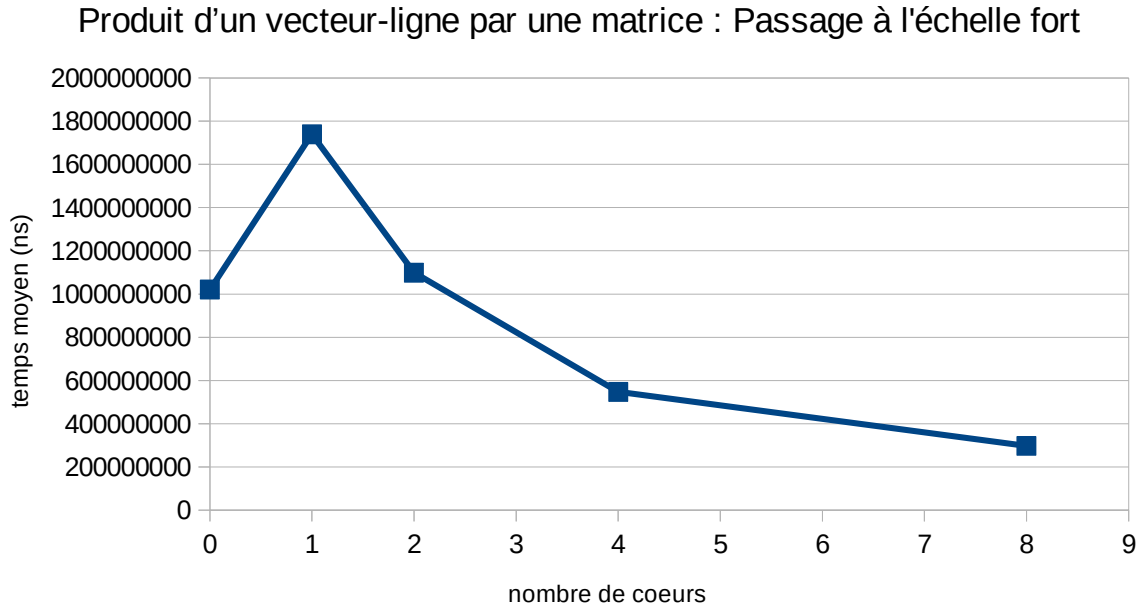
2.2 Matrice-Vecteur

Le code de la question 2.11 de cette partie est contenu sur la branche « question211 » dans le GitHub . Les fichiers et méthodes concernés sont décrits dans le fichier « README.md » à la partie « 2.2 Matrice-Vecteur ». Il suffit donc de faire un checkout de cette branche pour accéder au code, dans le fichier « matrice-vecteur/main_mat_vect.cpp », les fonctions étant décrites dans « matrice-vecteur/main_mat_vect.h ».

Produit d’un vecteur-ligne par une matrice

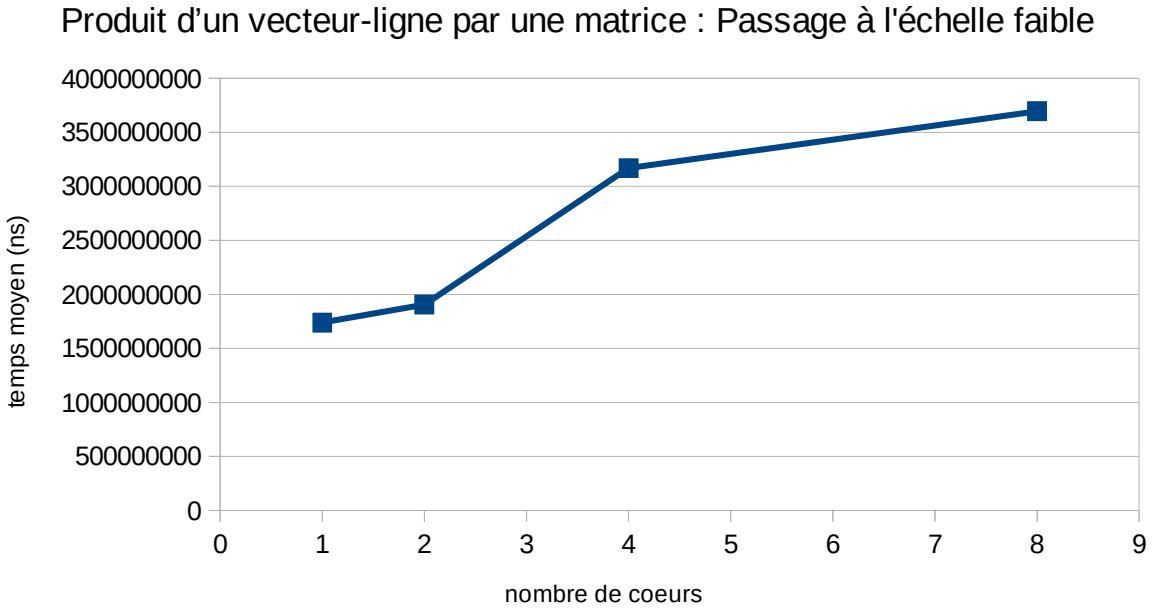
Passage à l’échelle fort : matrices de 100 sur 1 000 000

nombre de coeurs	0	1	2	4	8
temps moyen (ns)	1020477666,9	1739215803,5	1099060638,9	547479889,1	297177321



Passage à l'échelle faible

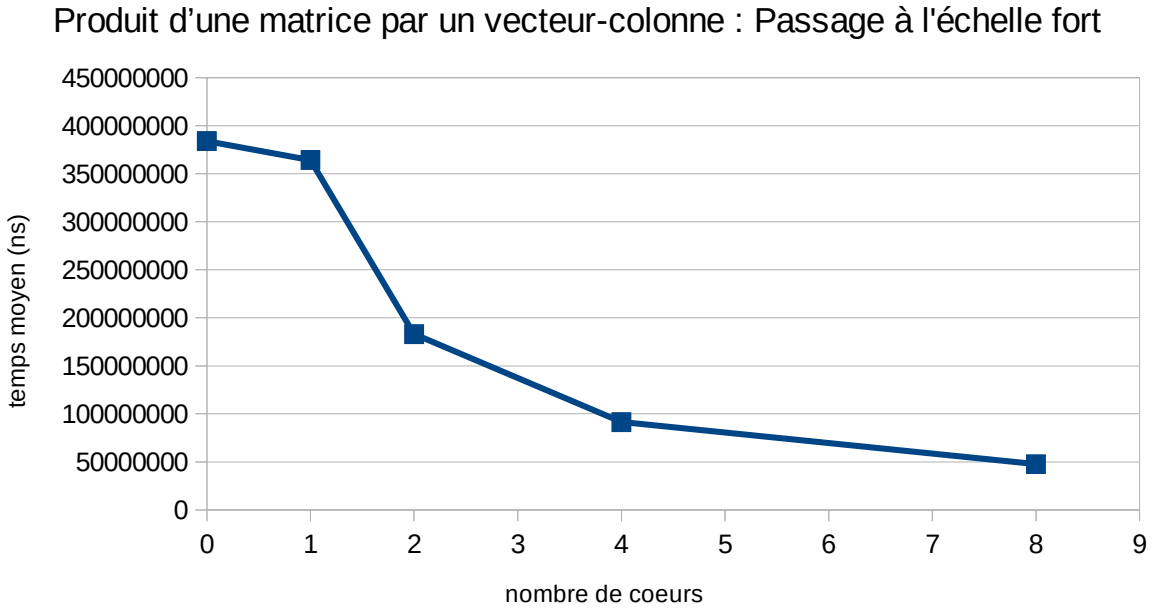
taille de la matrice	100x1000000	200x1000000	400x1000000	800x1000000
nombre de coeurs	1	2	4	8
temps moyen (ns)	1739215803,5	1905342520,7	3169045794,3	3695605864



Produit d'une matrice par un vecteur-colonne

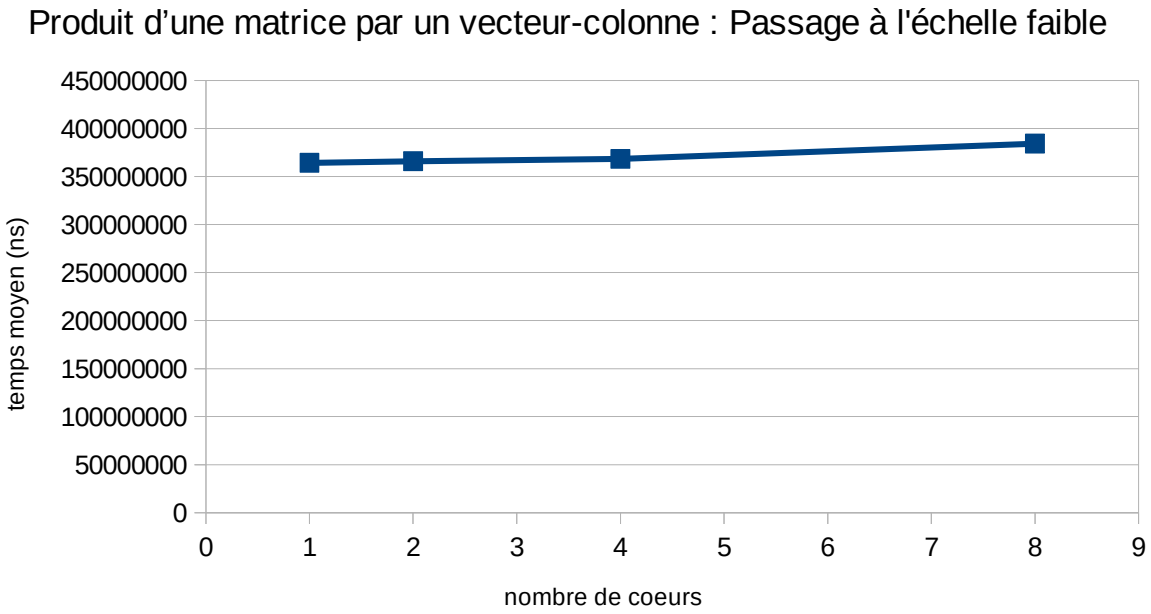
Passage à l'échelle fort : matrices de 100 sur 1 000 000

nombre de coeurs	0	1	2	4	8
temps moyen (ns)	383935514,2	364295872	183125504	91481405,4	47675467,2



Passage à l'échelle faible

taille de la matrice	100x1000000	200x1000000	400x1000000	800x1000000
nombre de coeurs	1	2	4	8
temps moyen (ns)	364295872	365923789,4	368364892,1	384285793,9



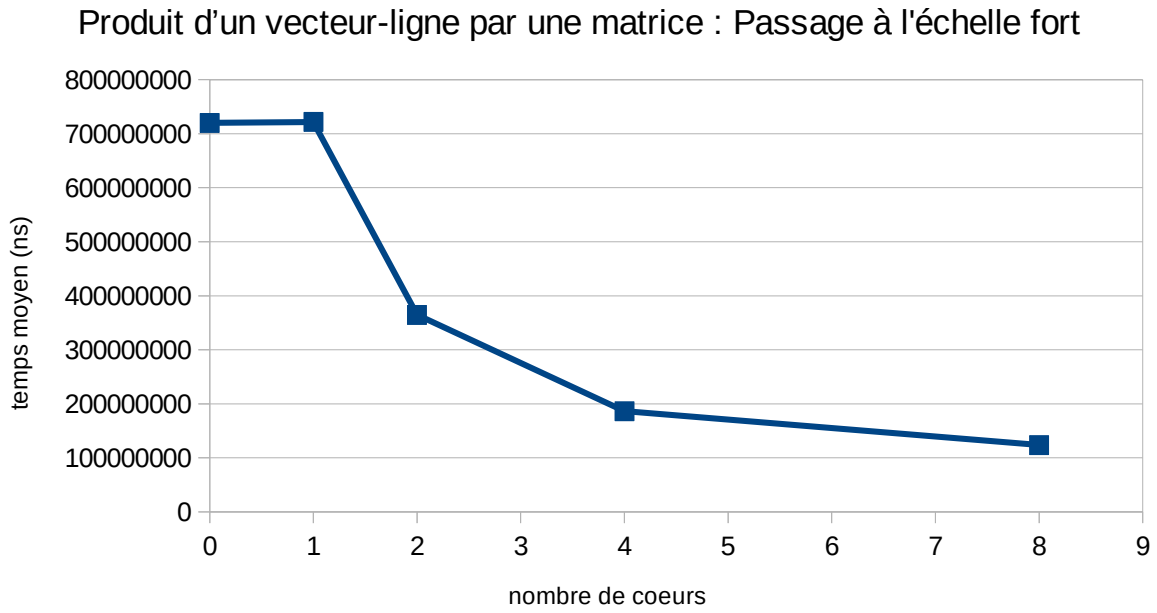
2.3 Matrice-Matrice

Le code de la question 2.12 de cette partie est contenu sur la branche « question212 » dans le GitHub . Les fichiers et méthodes concernés sont décrits dans le fichier « README.md » à la partie « 2.3 Matrice-Matrice ». Il suffit donc de faire un checkout de cette branche pour accéder au code, dans le fichier « matrice-matrice/main_mat.cpp », les fonctions étant décrites dans « matrice-matrice/main_mat.h ». Cette branche contient ainsi le résultat final du TD mais également ce présent rapport.

Produit d’une matrice par une matrice

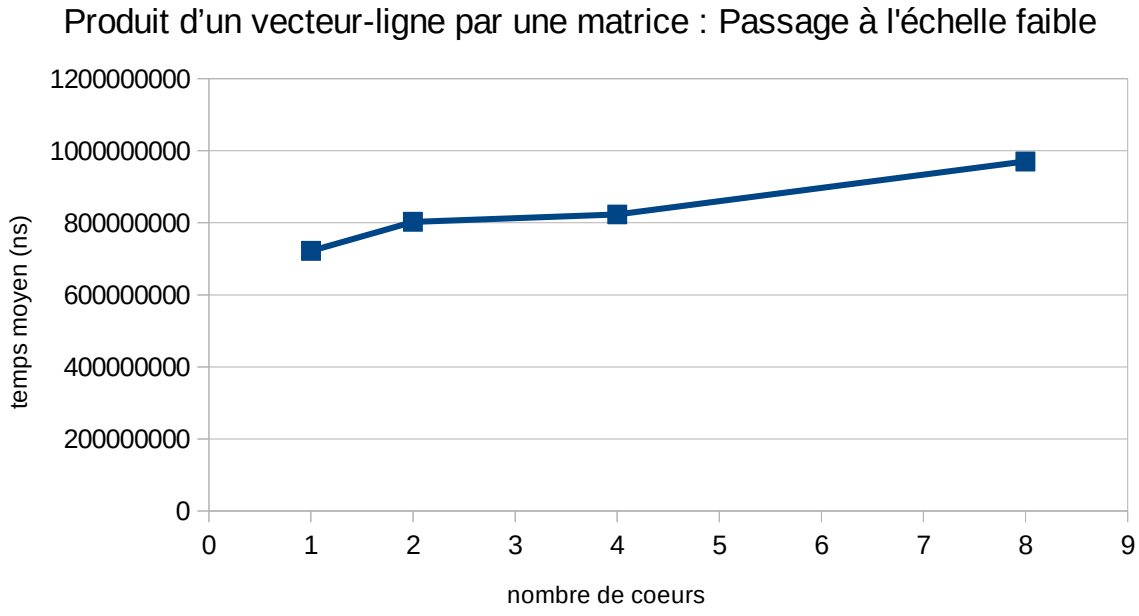
Passage à l’échelle fort : matrices de 200 sur 1000 et 1000 sur 500

nombre de coeurs	0	1	2	4	8
temps moyen (ns)	719990905,3	721745193,9	364554976,5	186374337,4	123564089,3



Passage à l‘échelle faible

taille des matrices	200x1000 et 1000x500	200x1000 et 1000x1000	200x1000 et 1000x2000	200x1000 et 1000x4000
nombre de coeurs	1	2	4	8
temps moyen (ns)	721745193,9	802629454,3	823120587,2	970114067,2



Conclusions

Globalement, on constate que la compilation avec OpenMP permet une amélioration des performances et même parfois lorsqu'on utilise qu'un seul cœur par rapport à une compilation normale non parallélisée. Cela peut s'expliquer par le fait que la compilation OpenMP intègre des optimisations de code à la compilation.

Dans le cas des passages à l'échelle forts, on constate une amélioration des performances qui suit presque le nombre de cœurs utilisés, la vitesse de calcul double presque avec le nombre de cœurs. Le fait est qu'avec l'augmentation du nombre de cœurs, il y a un écart de plus en plus important au résultat attendu si le passage l'échelle fort était parfait (division par 2 du temps de calcul). Il est dû au temps de calcul nécessaire à la création et la destruction des threads OpenMP, augmentant avec le nombre de cœurs.

Avec le passage à l'échelle faible, On peut noter certes une augmentation du temps de calcul par rapport à la taille des données et au nombre de cœurs, dû au temps de synchronisation, de création et de destruction des threads OpenMP, mais que cette augmentation tend à diminuer avec le passage à l'échelle, ce qui sous-tend que OpenMP est plutôt bien adapté au passage à l'échelle faible.

Dans le cas de la multiplication d’une matrice par un vecteur, on constate que le passage à l’échelle faible de la multiplication d’un vecteur-ligne par une matrice est beaucoup moins efficace que celui pour la multiplication d’une matrice par un vecteur-colonne. Cependant il reste toujours plus performant sur des grandeurs de vecteur basses. Ceci peut s’expliquer par le fait la réduction OpenMP utilisée pour parcourir le vecteur est coûteuse à mettre en place. De ce fait, elle augmente fortement le temps d’exécution lorsqu’elle effectuée de nombreuses fois.