Analyse de Performance

Barona Stephanie, Grand Maxence

November 16, 2017

1 Introduction

1.1 Environnement

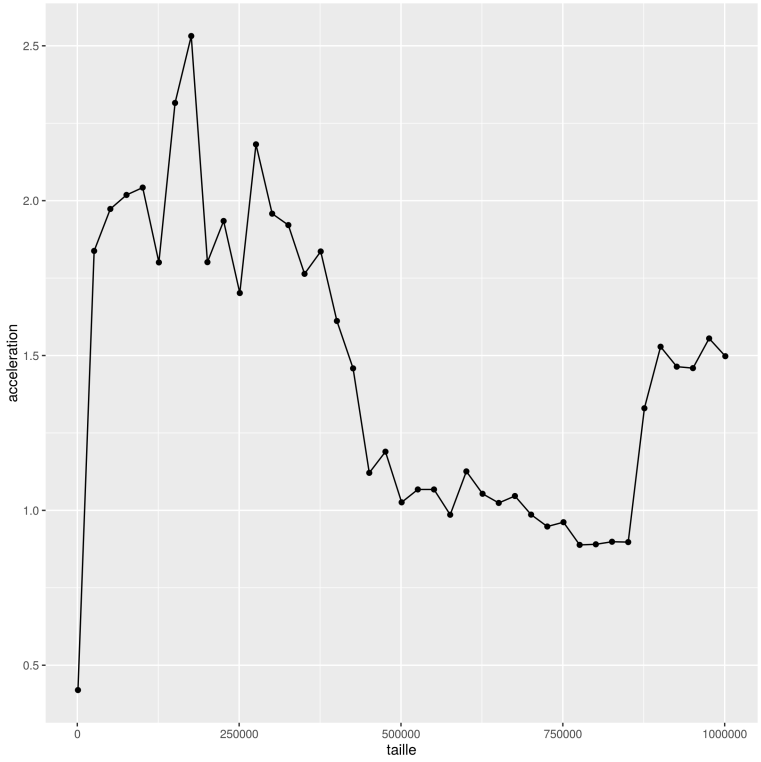
- OS : Xubuntu
- 4 processeurs Intel(R) Core(TM) i5-3470 CPU 3.20 GHz
- La machin est branché sur un réseau des problèmes au niveau du réseau peuvent poser des problèmes de performance.
- Mis à part les processus utilisés par le kernel, sur lesquels nous n'avons pas la main, nous n'avons laisser que l'exécution des programmes. Aucune application ou autre programme tourne en même temps que notre jeu de tests

1.2 Jeu de Tests et Hypothèses

Nous avons tester l'algorithme séquentiel de tri, ainsi que l'algorithme parallèle de tri. Pour l'algorithme parallèle de tri, nous avons testé avec un nombre de threads variant de 2 à 16. Chaque algorithme à été testé sur différents vecteurs, de tailles variant de 1000 à 1000000 (avec un pas de 2500). Chaque tests à été refait 30 nfois pour éliminer les incercitudes de mesures. A chaque tests nous avons calculé le temps d'exécution de l'algorithme, ainsi que le temps CPU cumulé de l'utilisateur.
Nous avons ensuite calculé l'accéleration de chaque version de l'algorithme parallèle par rapport à l'algorithme séquantiel : $a = \frac{T_{seq}}{T_{par}}$. Nous prenons l'hypothèse que l'accélération de Temps machine sera supérieur à un pour tous les algorithmes paralèlle, car nous supposons que plus il y aura de thread, plus l'exécution sera rapide. Nous prenons aussi l'hypothèse que cette accélération sera la plus forte pour 4 threads, car nous avons testé sur une machine à 4 processeurs. Pour le temps cpu de l'utilisateur, nous prenons l'hyporthèse que ce temps augmentera avec le nombre de thread, nous supposons donc que l'accélération sera inférieur à un.
Nous avons ensuite calculÍ'efficacité : $e = \frac{a}{NB_{proc}}$

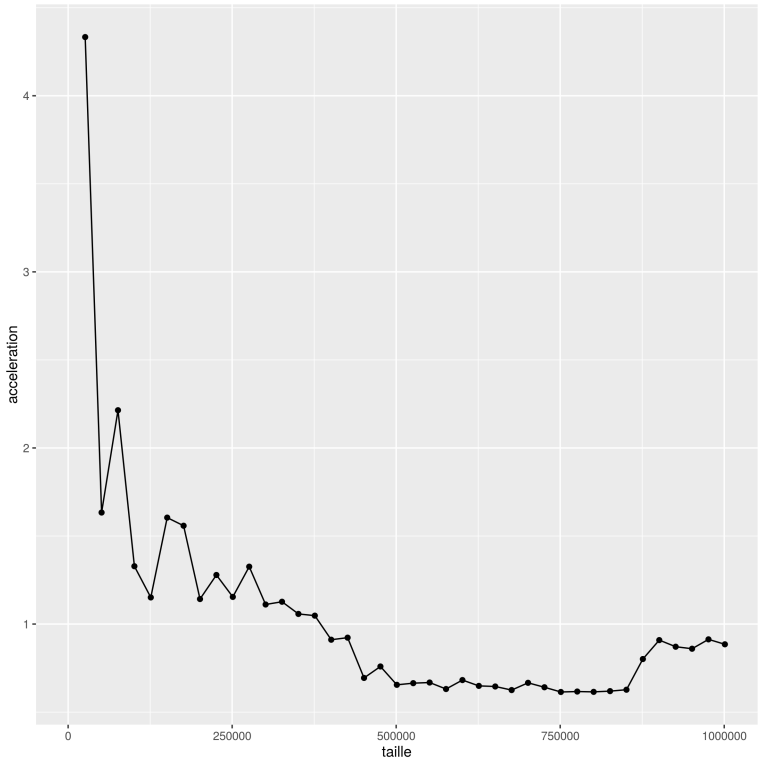
2 Avec 2 threads

2.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 200000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers un, passÍe seuil des 700000 éléments.

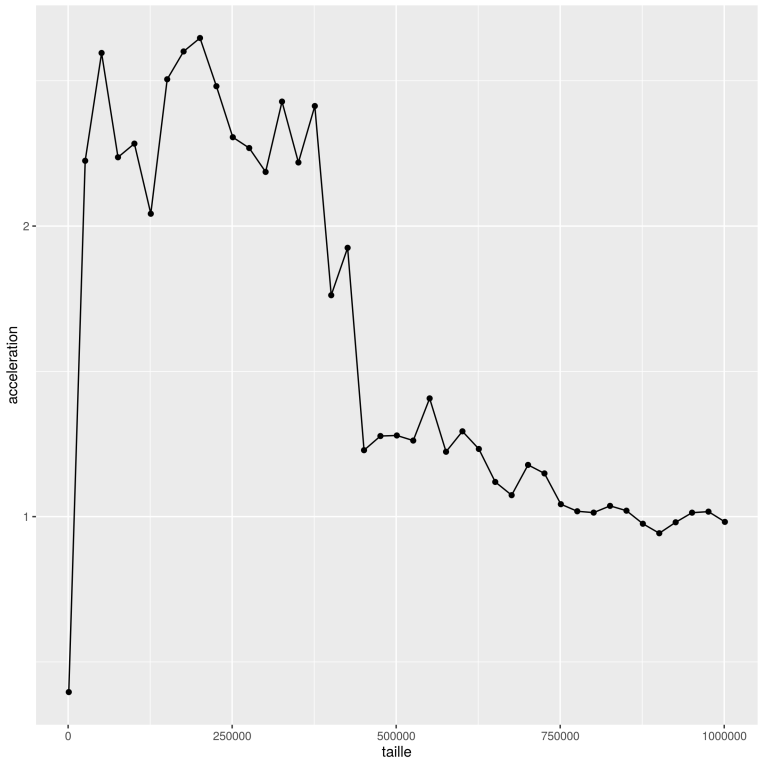
2.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 400000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 2 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.6.

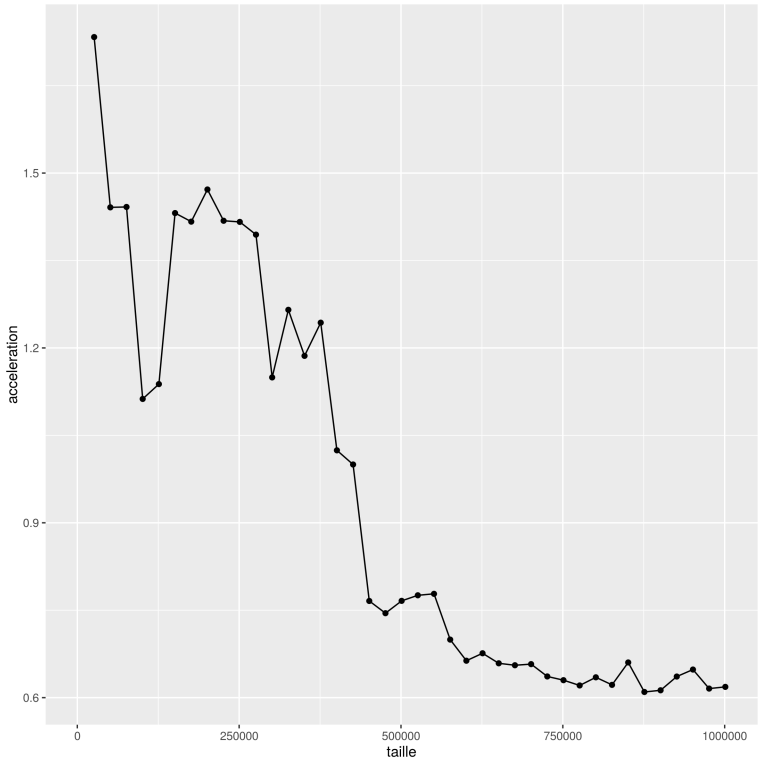
3 Avec 3 threads

3.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 200000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.5, passÍe seuil des 700000 éléments.

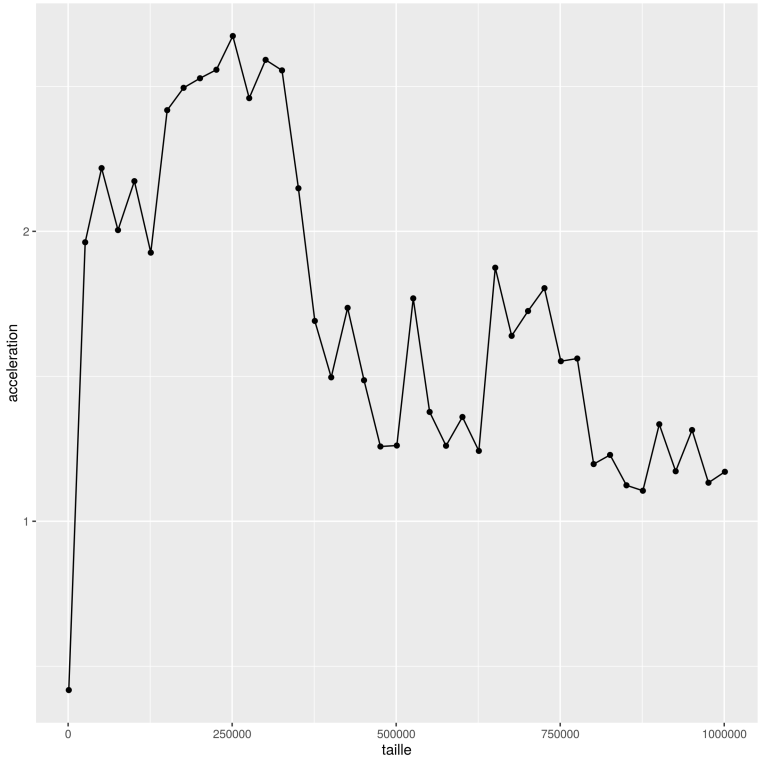
3.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 400000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 3 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.7.

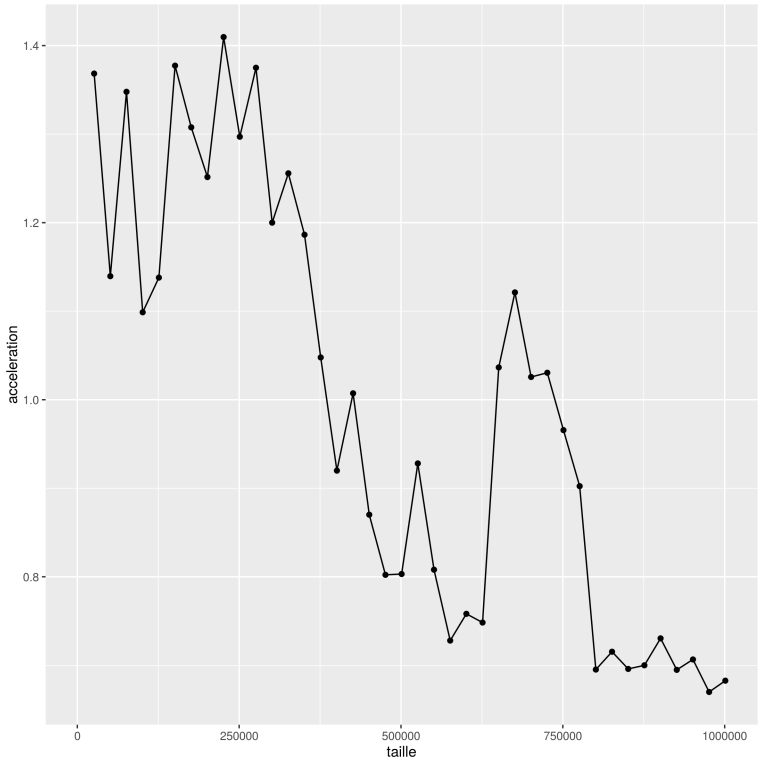
4 Avec 4 threads

4.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 200000 éléments. Passé ce seuil, le niveau de l'accélération baisse (et finit par tendre vers un, passÍe seuil des 700000 éléments).

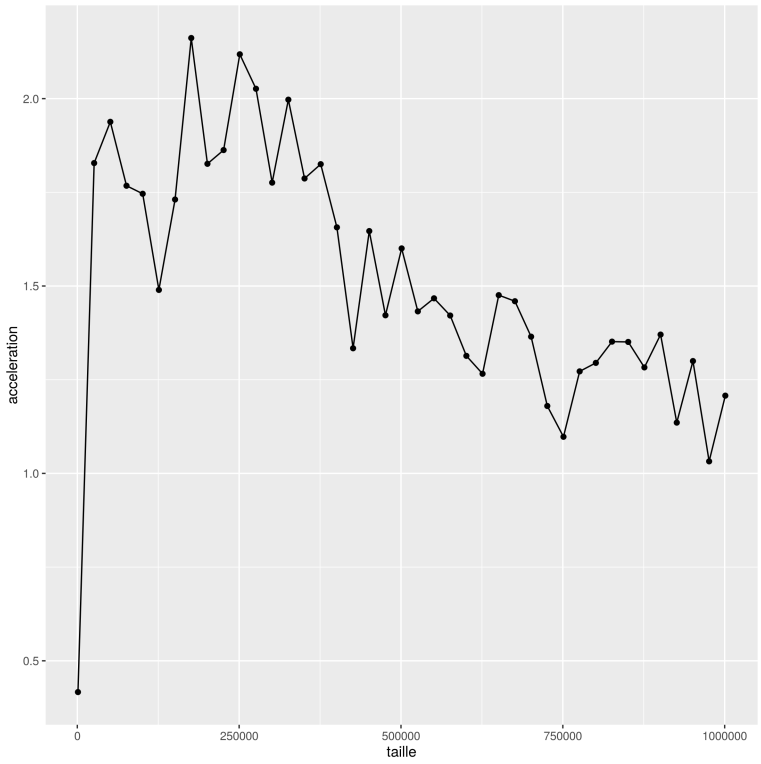
4.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 500000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 4 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.7.

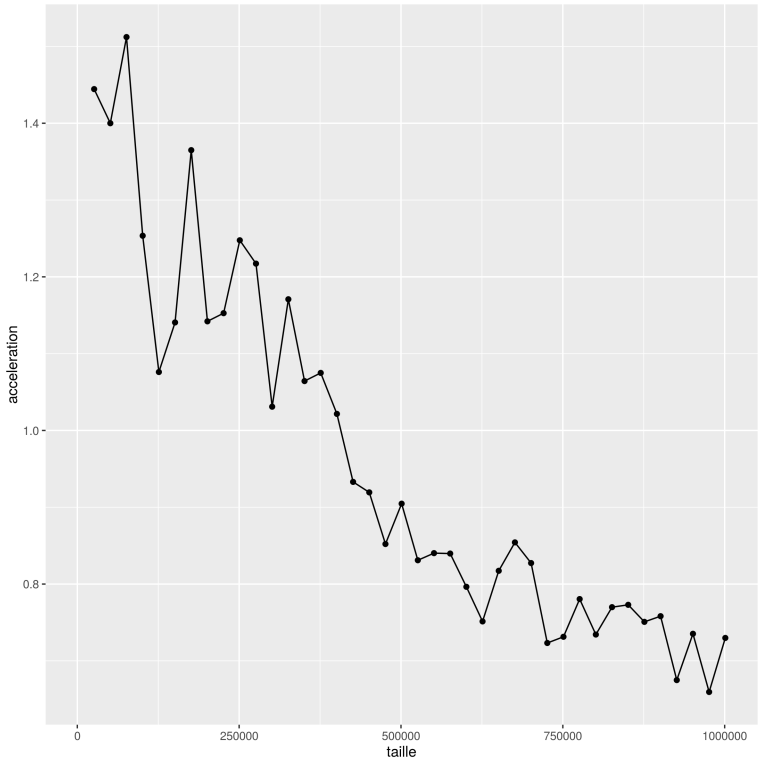
5 Avec 5 threads

5.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 200000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.3, passÍe seuil des 800000 éléments.

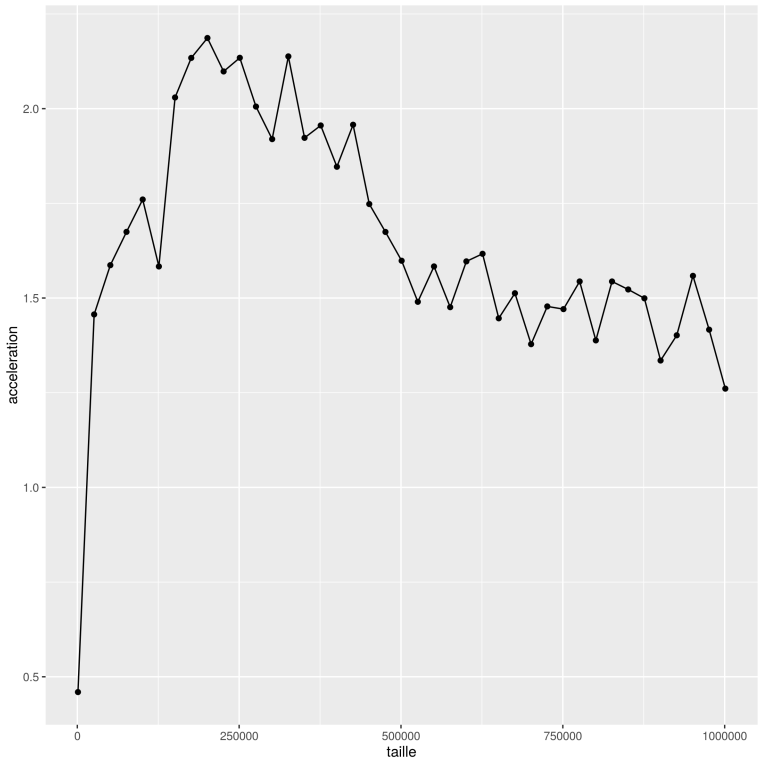
5.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 450000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 5 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.7.

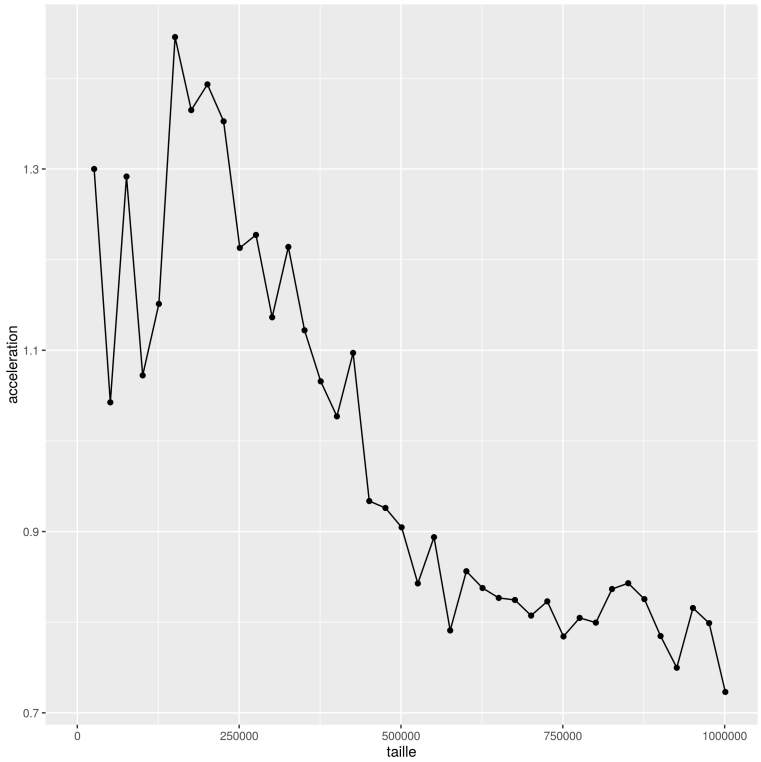
6 Avec 6 threads

6.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 200000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.5, passÍe seuil des 500000 éléments.

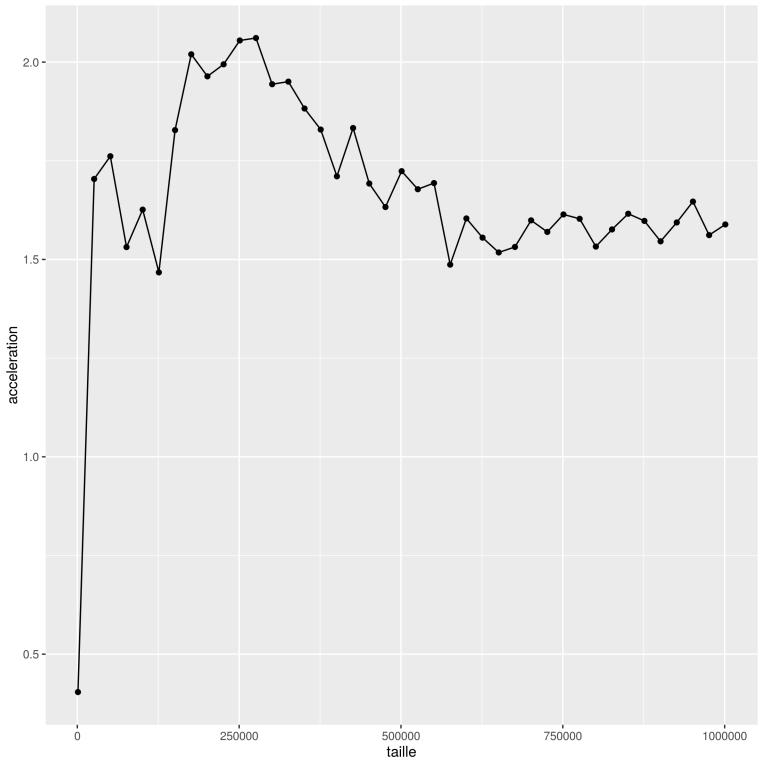
6.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 400000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 6 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.9.

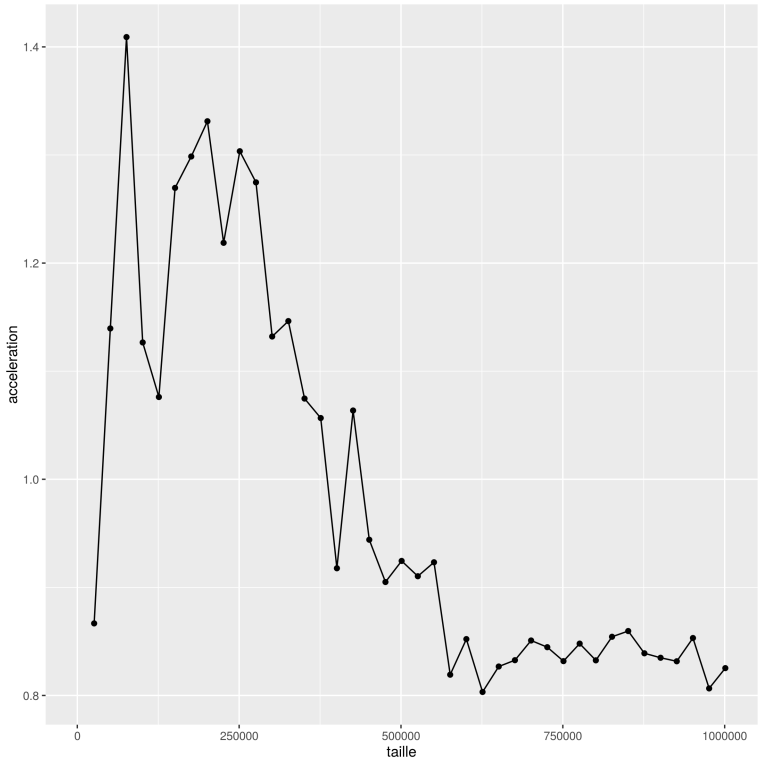
7 Avec 7 threads

7.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 200000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.5, passÍe seuil des 500000 éléments.

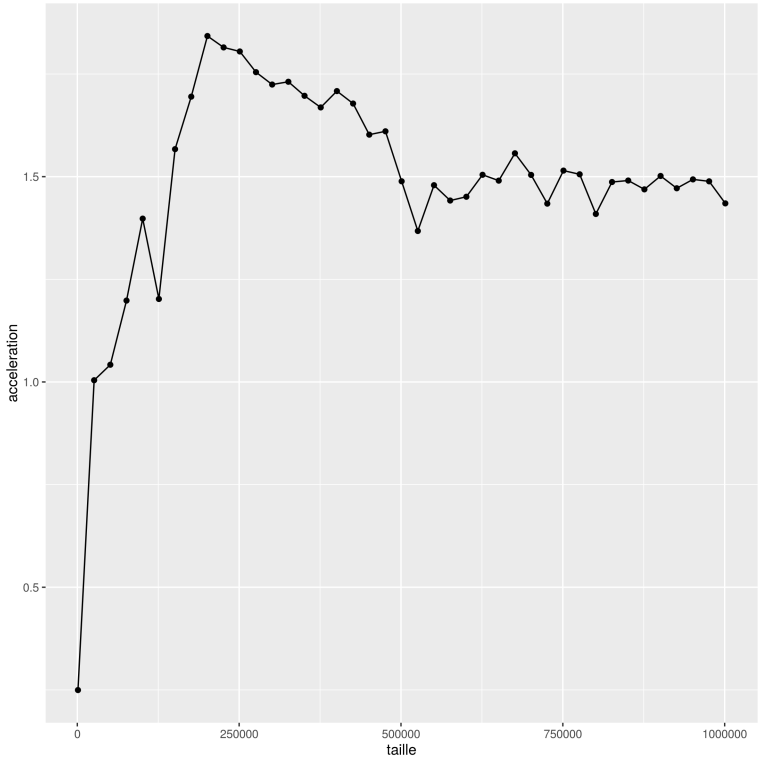
7.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 300000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 7 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.8.

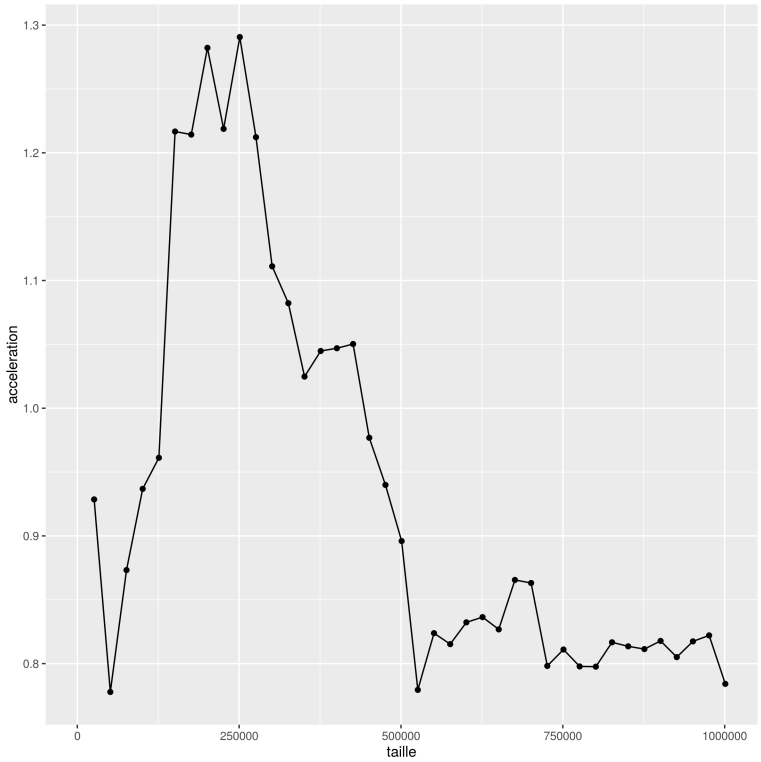
8 Avec 8 threads

8.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 200000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.4, passÍe seuil des 800000 éléments.

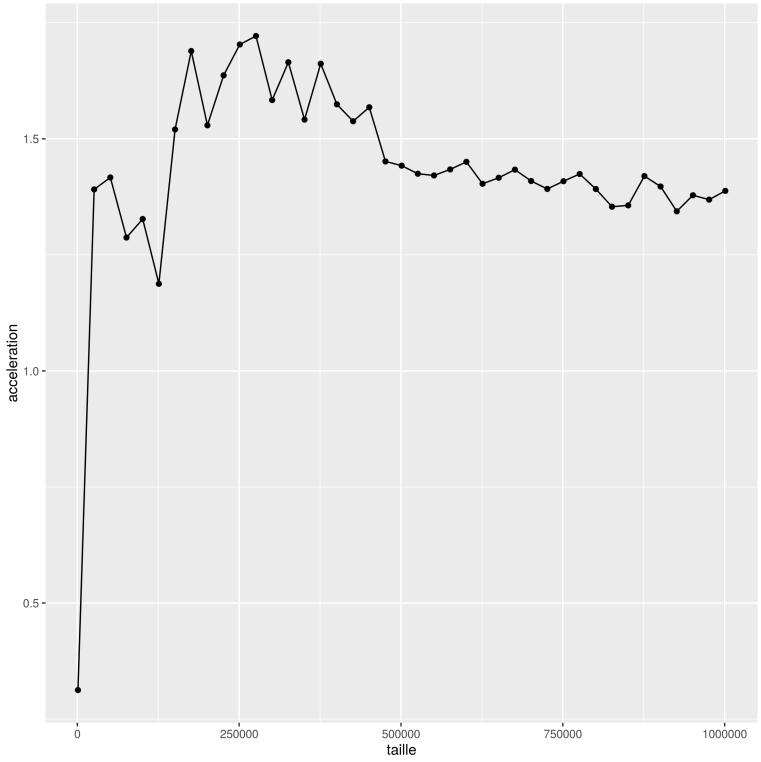
8.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 400000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 8 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.7.

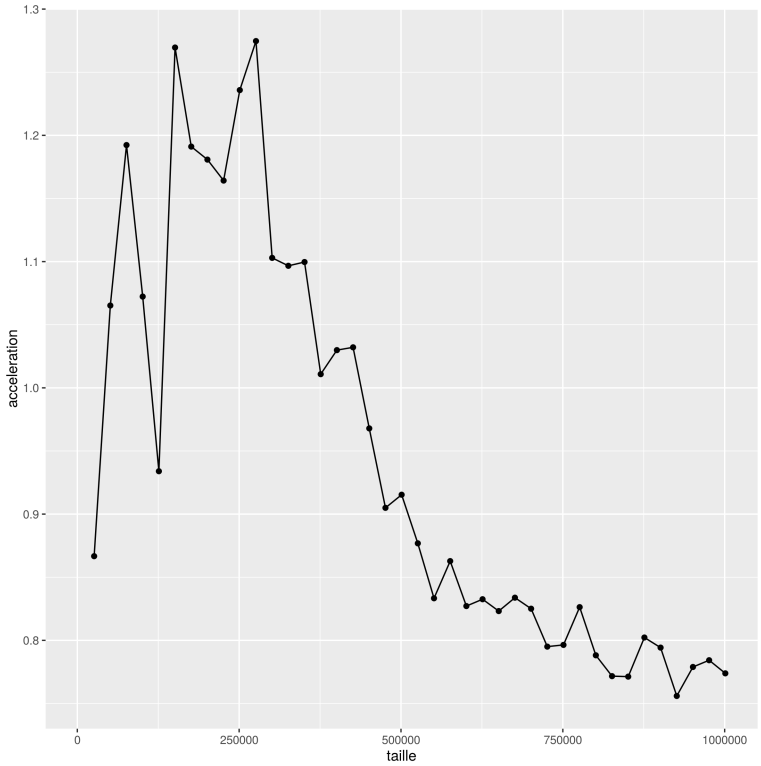
9 Avec 9 threads

9.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 400000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.4, passÍe seuil des 500000 éléments.

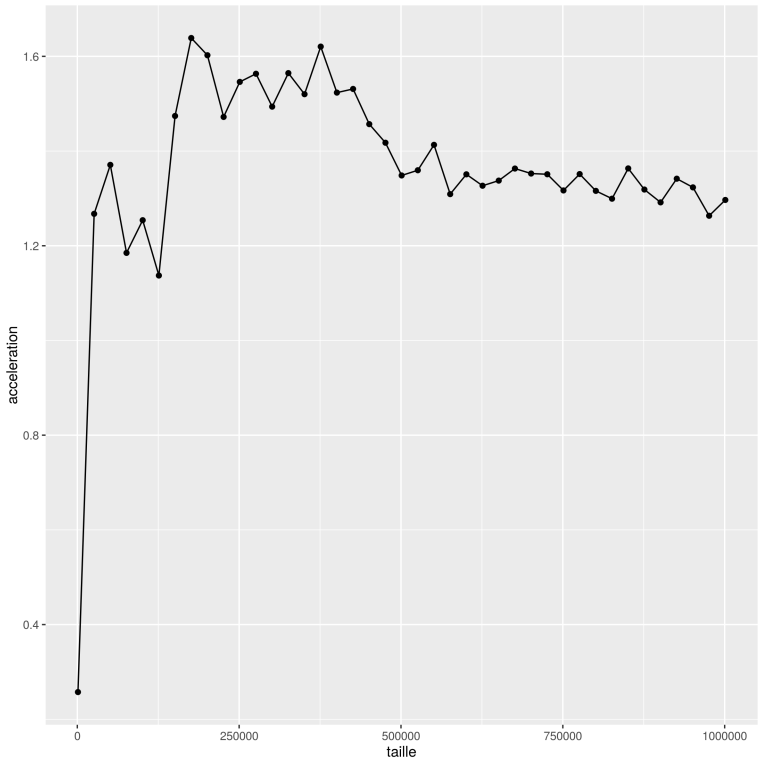
9.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 450000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 9 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.7.

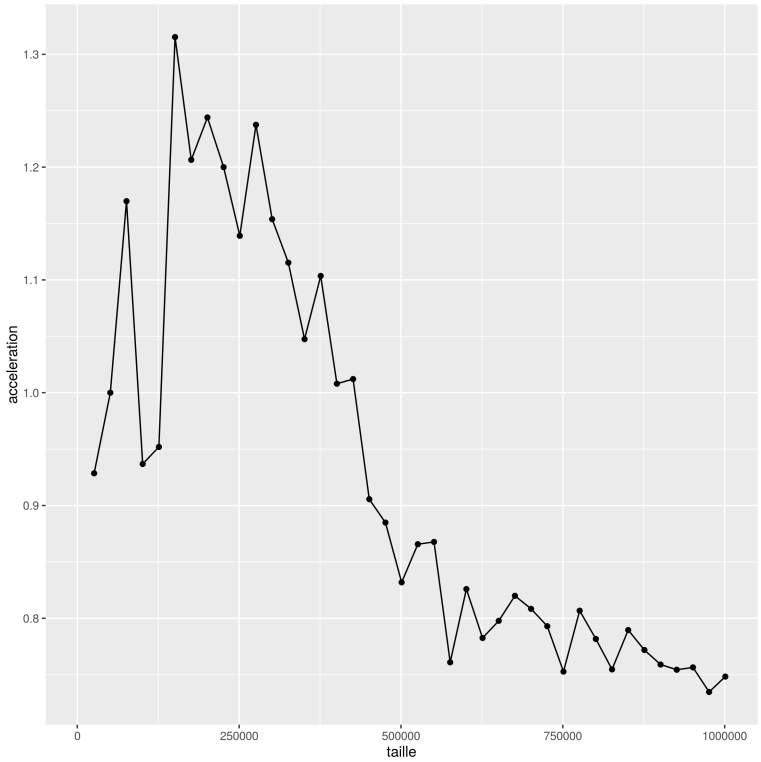
10 Avec 10 threads

10.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 250000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.3, passÍe seuil des 550000 éléments.

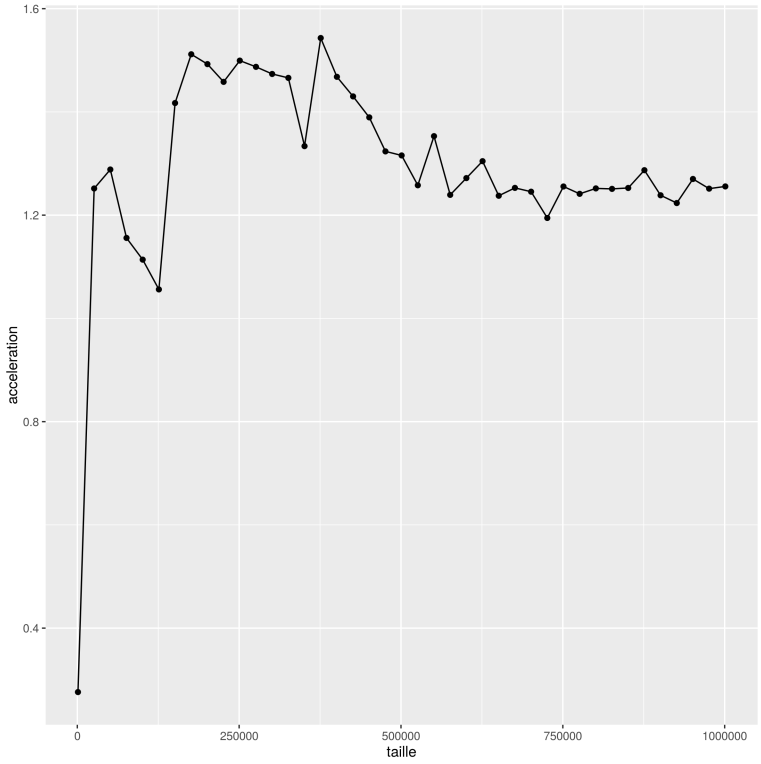
10.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 450000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 10 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.7.

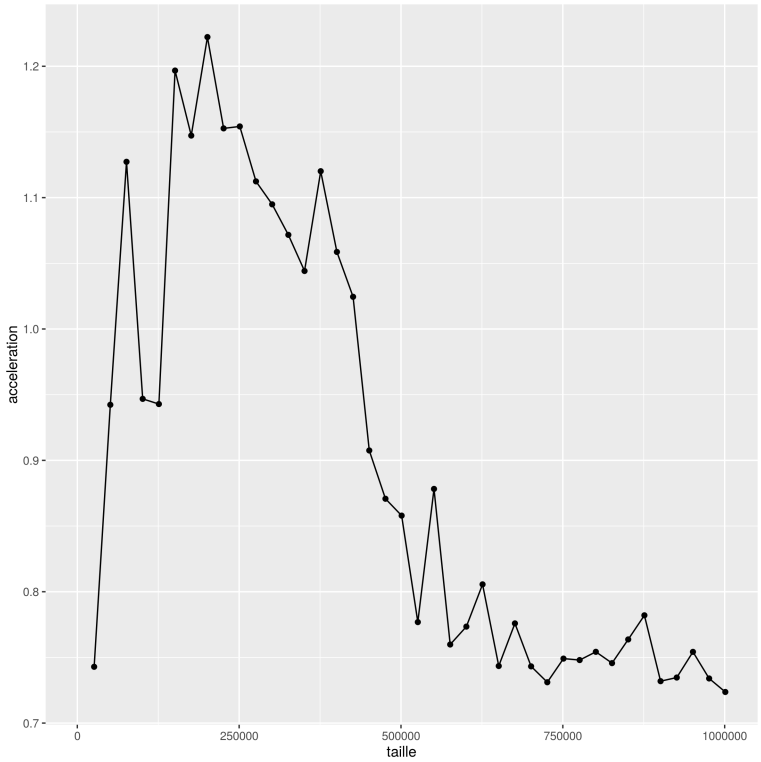
11 Avec 11 threads

11.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 450000 éléments. Passé ce seuil, le niveau de l'accélération baisse.

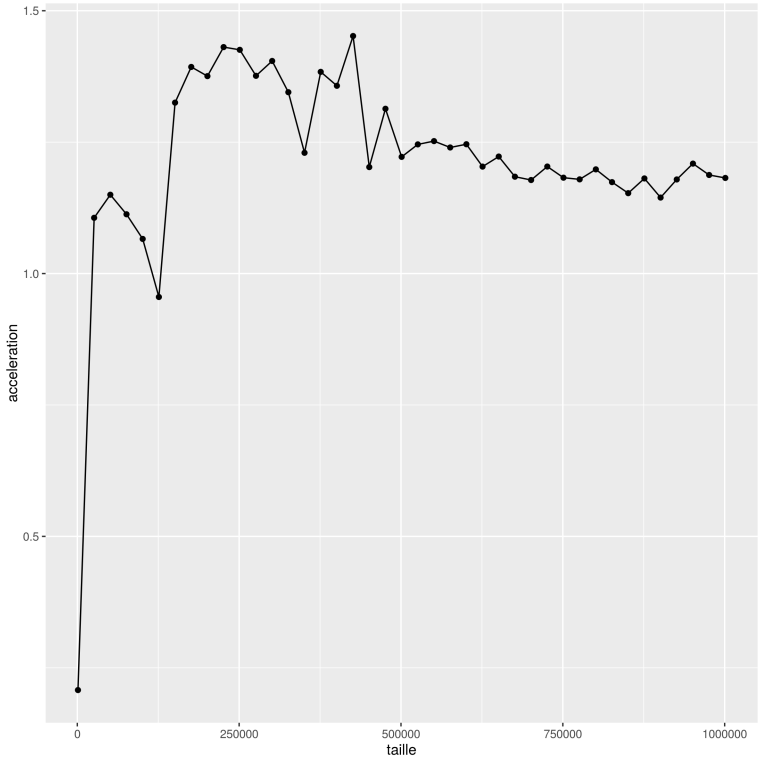
11.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 450000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 11 threads qu'en séquentiel. Cette accélération finit par tendre verss 0.7.

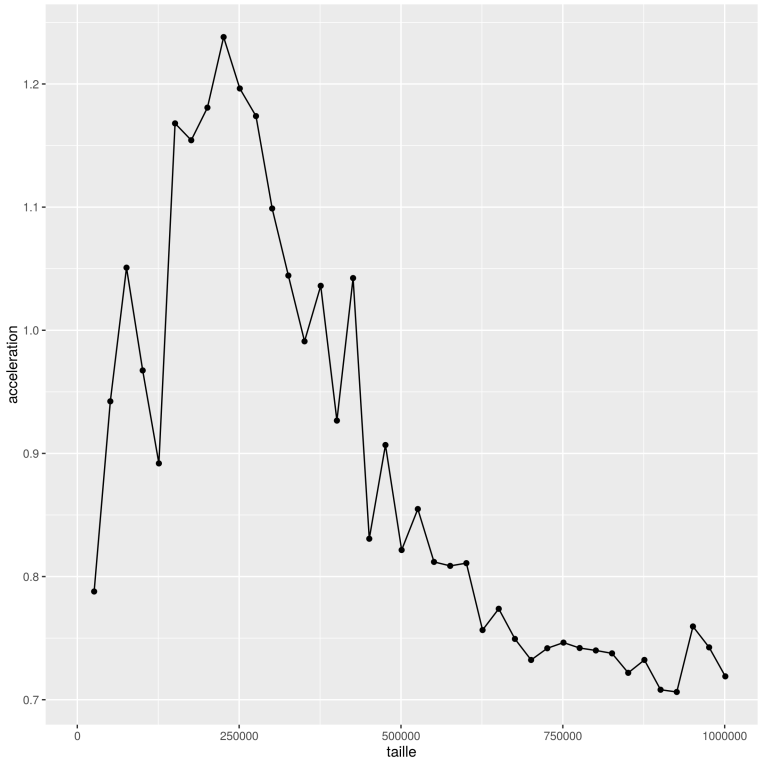
12 Avec 12 threads

12.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 400000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.2, passÍe seuil des 500000 éléments.

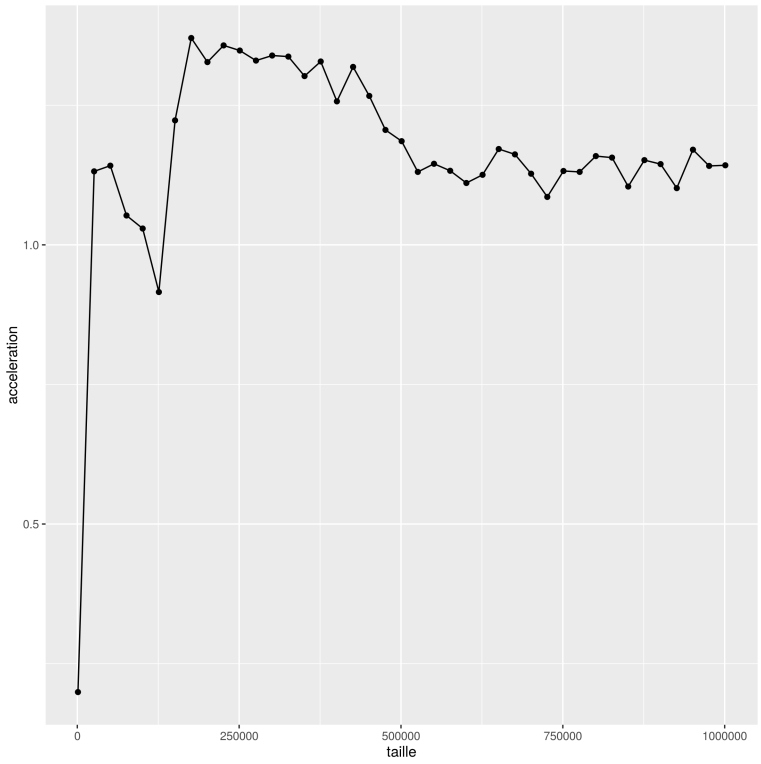
12.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 450000 et supérieur à 25000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 12 threads qu'en séquentiel. Cette accélération finit par tendre vers 0.7.

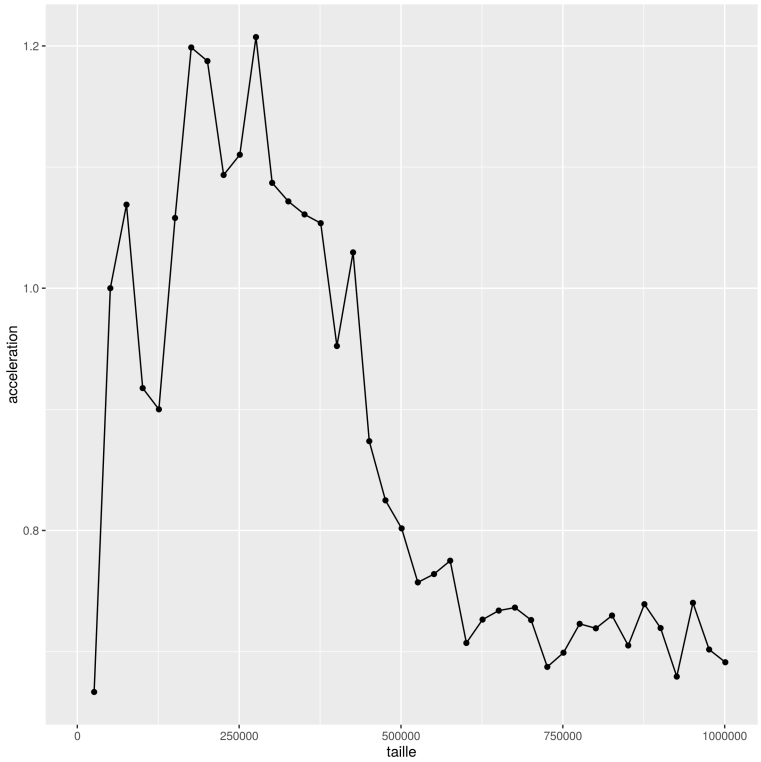
13 Avec 13 threads

13.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 400000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.2, passÍe seuil des 500000 éléments.

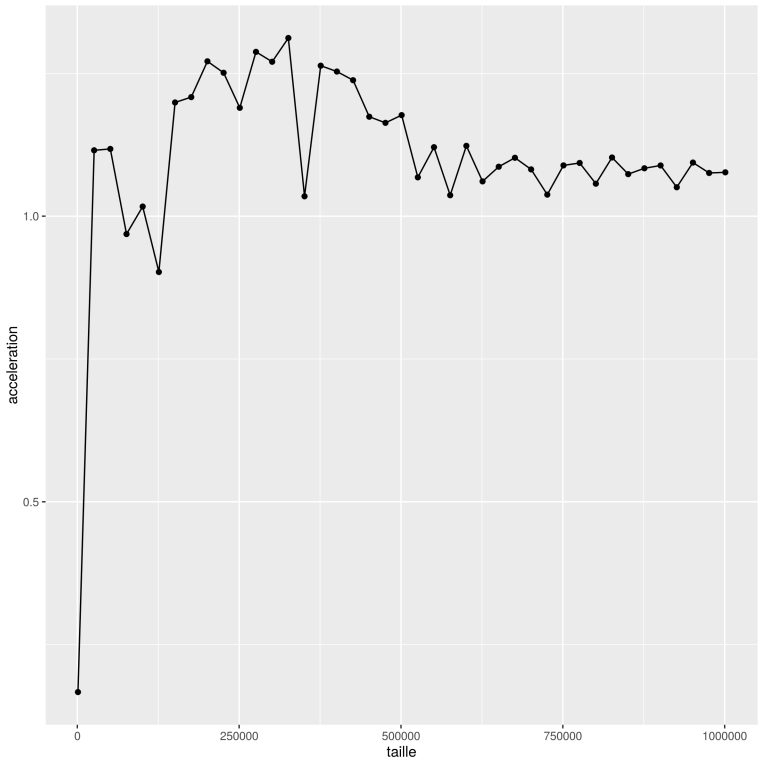
13.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 450000 et supérieur à 25000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 13 threads qu'en séquentiel. Cette accélération finit par tendre vers 0.7.

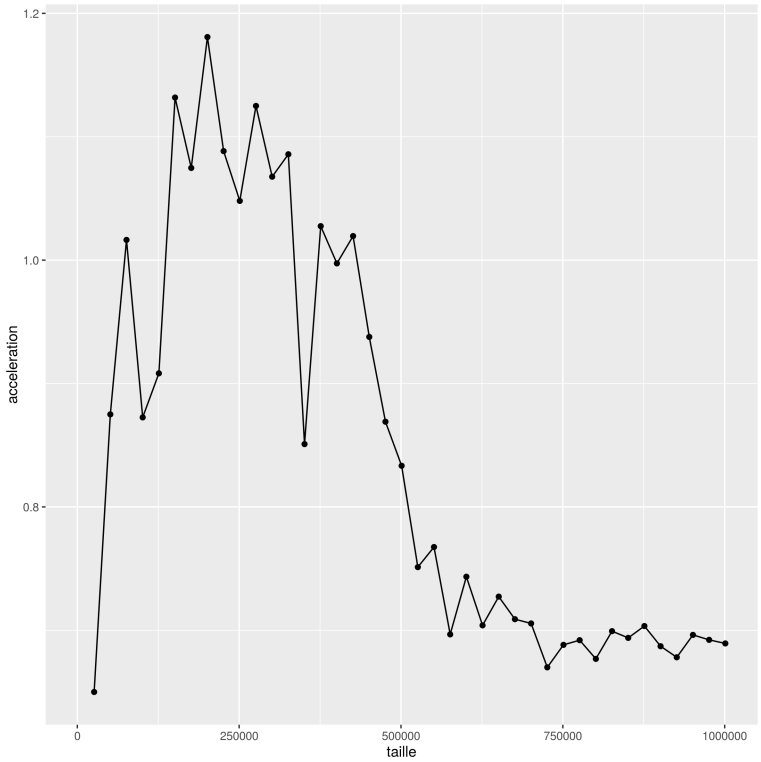
14 Avec 14 threads

14.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 400000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.2, passÍe seuil des 500000 éléments.

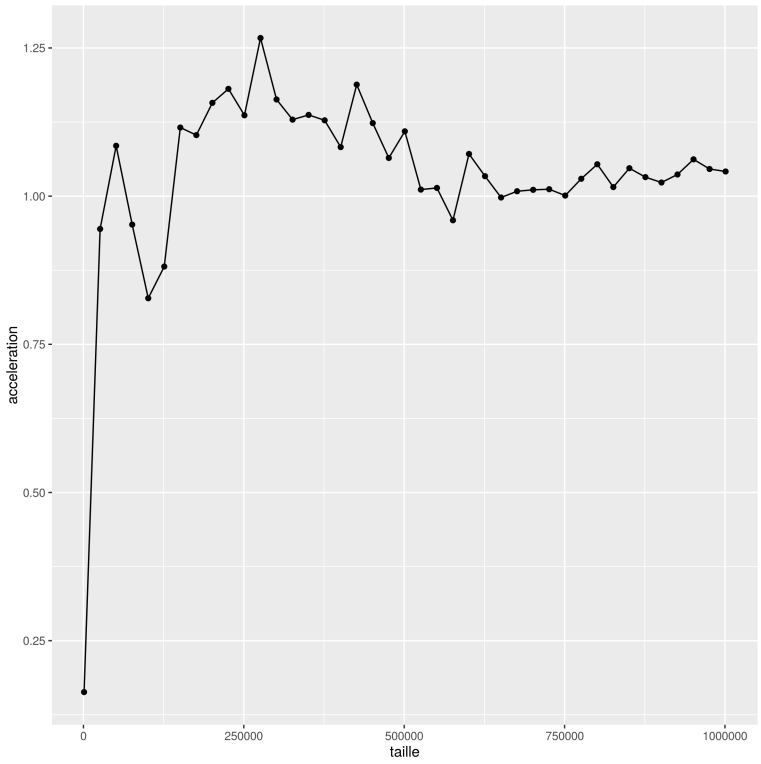
14.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 450000 et supérieur à 25000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 14 threads qu'en séquentiel. Cette accélération finit par tendre vers 0.7.

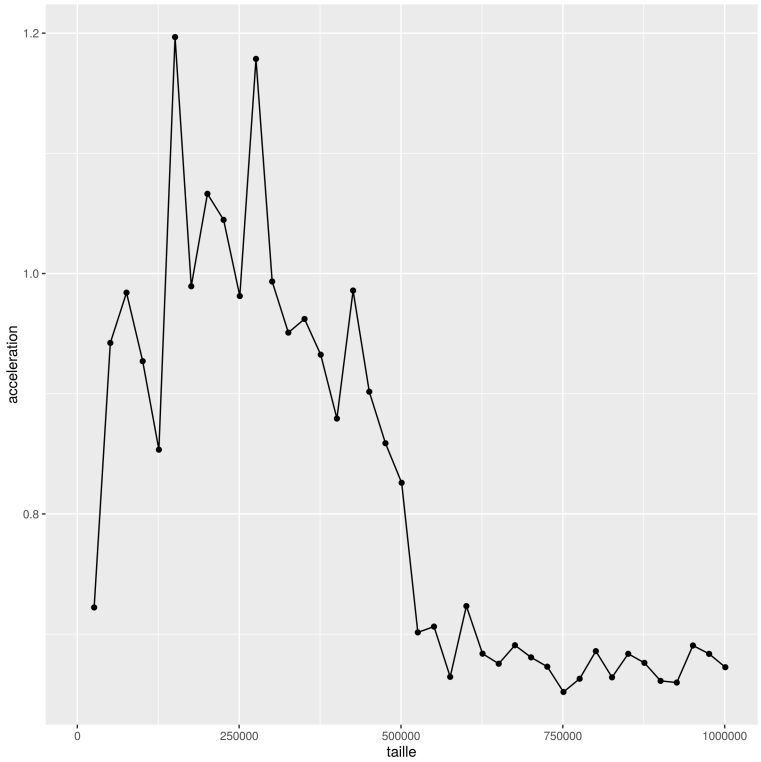
15 Avec 15 threads

15.1 Temps d'exécution



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours supérieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 400000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 1.2, passÍe seuil des 500000 éléments.

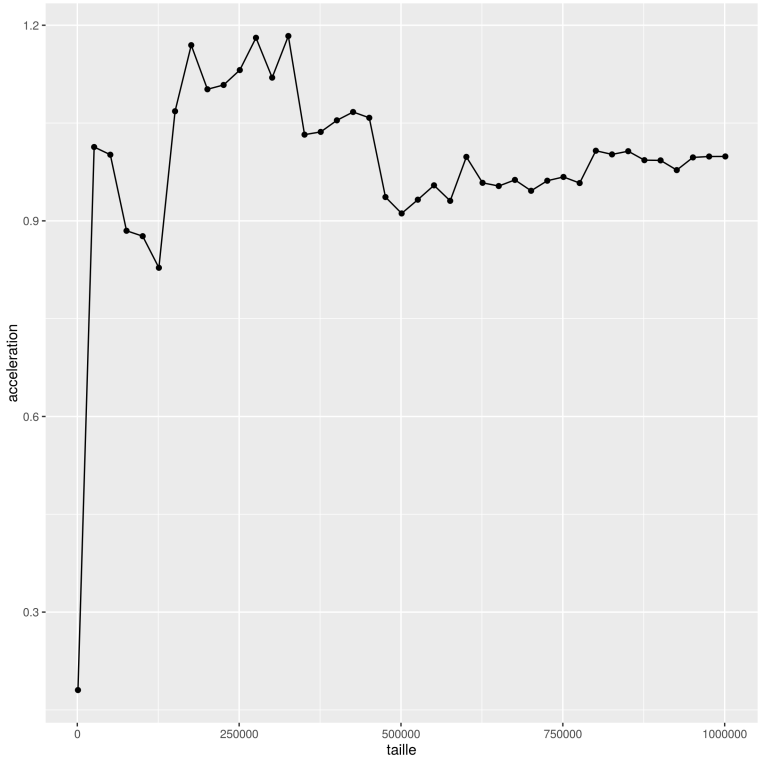
15.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs d'une taille inférieur à 450000 et supérieur à 25000, l'accélération est supérieur à 1. Néanmoins passé ce seuil, l'accélération est maintenant inférieur à un, cela signifie que le temps CPU cumulé de l'utilisateur finit par être plus élevé avec 15 threads qu'en séquentiel. Cette accélération finit par tendre vers 0.7.

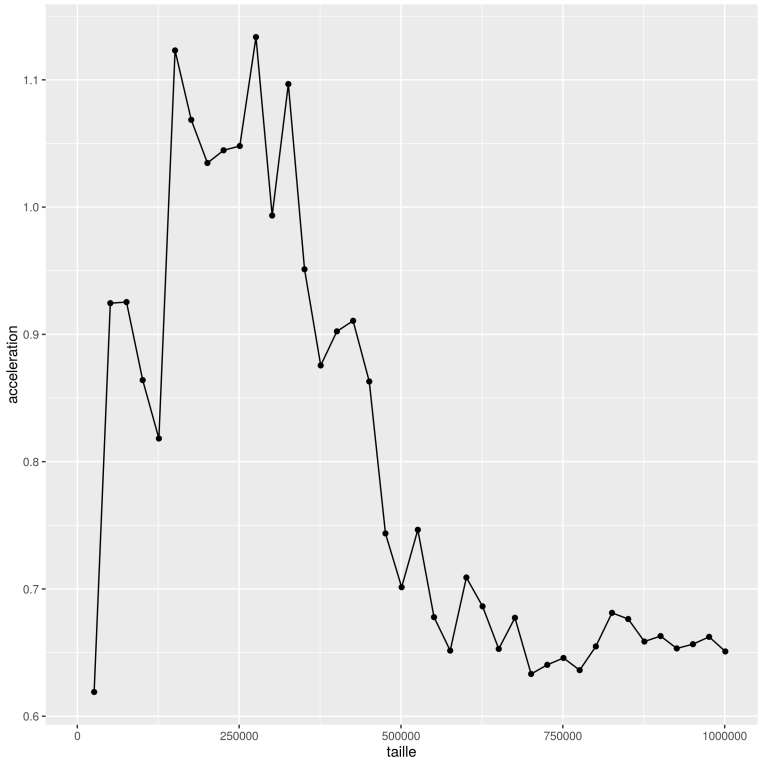
16 Avec 16 threads

16.1 Temps d'exécution



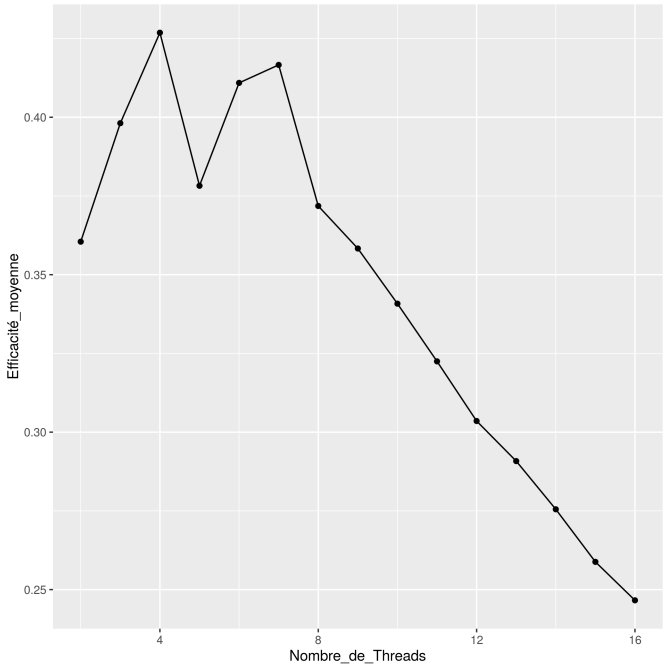
Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours inférieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 300000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 0.95, passÍe seuil des 500000 éléments.

16.2 Temps CPU cumulé de l'utilisateur



Nous observons que pour des vecteurs allant d'une taille de 1000 éléments à 1000000 d'éléments, l'accélération est toujours inférieur à 1. Nous observons aussi que l'accélération est la plus grande, pour des vecteurs d'une taille d'environ 300000 éléments. Passé ce seuil, le niveau de l'accélération baisse et finit par tendre vers 0.5, passÍe seuil des 500000 éléments.

17 Comparaison des résultats



Lorsque nous comparons l'efficacité moyenne pour chaque version de l'algorithme paralèlle, nus observons que le pic d'efficacité est atteint pour 'algorithme avec 4 threads. Nous observons, qu'à partir de 5 threads, l'efficacité baisse.