Toutes les mesures de temps étaient effectuées 3 fois sur les machines des salles de TP avec 4 cœurs. C’est la moyenne des 3 mesures qui sera indiqué dans les tableaux.

La boucle de la ligne 21 n’est pas parallélisable puisqu’on ne connait pas le nombre d’itérations de la boucle en avance (et parce qu’on peut pas paralléliser une lecture de fichier).

La boucle de la ligne 31 ne peut être parallélisée à cause de la variable *code* qui est changé d’une manière non triviale à chaque itération de la boucle (multiplication par 2, sous une certaine condition addition de 1).

La boucle de la ligne 44 ne peut pas être parallélisée à cause de la variable partagée *code* qui est utilisée dans la condition du *if* et, donc, chaque itération de la boucle à besoin de la valeur obtenue à l’itération précédente.

La boucle de la ligne 57 pourrait être parallélisée, mais les mesures de temps ont montré qu’on ne gagne pas de temps à l’exécution en le faisant (voir tableau).

La boucle de la ligne 60 était parallélisée puisque la parallélisation permet d’économiser du temps (voir tableau). Par contre, elle l’était avec un *schedule(dynamic)* puisqu’un *schedule* par défaut ne serait pas équilibré (parce que plus le compteur *i* est grand, plus le nombre d’itérations de la boucle intérieure est petit).

La boucle de la ligne 61 n’est pas parallélisée puisque c’est une boucle intérieure de la boucle précédente qui est déjà parallélisée.

On ne peut pas paralléliser la boucle de la ligne 64 puisqu’elle est une boucle intérieure de la boucle précédente, et parce qu’on ne connait pas le nombre d’itérations en avance.

La boucle de la ligne 77 ne peux pas être parallélisée parce que tout le corps de la boucle devrait être placé dans une section critique (à cause des variables partagées *max\_num\_facts* et *most\_frequent\_fact*) ce qui rendrait la parallélisation de la boucle inutile.

Il n’y a pas d’utilité de paralléliser la boucle de la ligne 93, puisque malgré le fait qu’en théorie elle peut l’être, on obtient une perte de temps (voir tableau).

La boucle de la ligne 95 est une boucle intérieure, donc, on ne la parallélise pas.

La boucle de la ligne 99 peut bien être paralléliser puisqu’on obtient un gain du temps (voir tableau).

La boucle de la ligne 100 est une boucle intérieure d’une boucle parallélisée, donc, pas de parallélisation.

La même réflexion et la même conclusion pour la boucle de la ligne 105.

On n’a pas observé de différence entre le temps séquentiel et le temps parallèle de l’exécution de la boucle de la ligne 114 (voir tableau), donc, elle peut être laissée non parallélisée.

La boucle de la ligne 128 peut être parallélisée, on obtient un gain de temps (voir tableau). Par contre, il faut faire attention à la variable *j*. Elle agit comme un compteur modulo *key\_length* qui ne pose pas de problèmes en séquentiel, mais pour avoir un bon fonctionnement en parallèle, il faut s’assurer que la variable *j* a bien la même valeur qu’elle aurait dans une exécution séquentielle. Pour cela, on la rend privée, et on distribue les itérations de boucle par « portions » de longueur *key\_length*.

|  |  |  |
| --- | --- | --- |
| **Ligne de la boucle** | **Temps en séquentiel, en secondes** | **Temps en parallèle, en secondes** |
| 57 | 0.000005 | 0.000348 |
| 60 | 0.126593 | 0.036770 |
| 93 | 0.000002 | 0.000068 |
| 99 | 0.000044 | 0.000023 |
| 114 | 0.000001 | 0.000001 |
| 128 | 0.000105 | 0.000044 |