Corentin GIROUD - Julien PELEGRI

TP1 - Somme des n premiers entiers :

Somme des 100 000 premiers entiers pour N = 100 000 :

(N = nombre de fois que les communication sont répétées)

Nom de la méthode	Temps en seconde
La_Plus_Mauvaise	**1.05**
Intermédiaire	**792**
All_Reduce	**503**

Moyenne des temps pour 10 essais

Somme des 100 000 premiers entiers pour N = 1 :

Ici on a du prendre N = 1 sinon le calcul générait des erreurs

Nom de la méthode	Temps en micro secondes
Optimale	**563**
All_Reduce	**798**
La_Plus_Mauvaise	**108*10^1**

Moyenne des temps pour 10 essais

La méthode « all reduce » a le même ordre d'efficacité que la méthode optimale.

TP2 - Résolution de l'équation de Laplace :

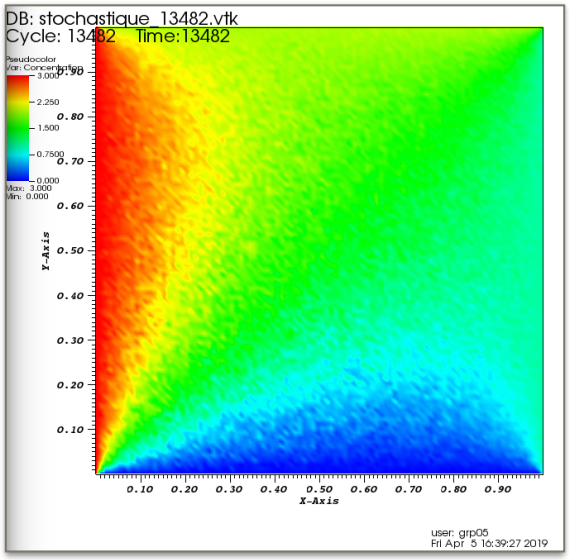
2.1 Implémentation séquentielle :

On a les résultats suivant :

```
Dimension selon x : 100
Dimension selon y : 100
Nombre de lancés par case : 100
Écriture du fichier vtk
Temps avant le vtk en sequentiel de communication pour la methode  : 26.0573 s
Temps après le vtk en sequentiel de communication pour la methode  : 26.0786 s
```

On a donc t(1) = 26.0573

On observe la figure suivant avec les conditions aux limites de l'énoncé :



Le calcul prend beaucoup de temps, la parallélisation va permettre de résoudre ce problème.

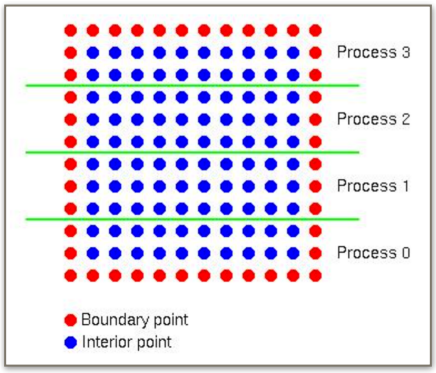
2.2 Implémentation parallèle :

Deux stratégie :

1/ Soit on donne aux n processeurs un nombre: floor(nb_tirages/n) qu'ils vont calculer chacun « séquentiellement » et ensuite envoyer au processeur 0.

2/ Soit on donne à chaque processeur une partie de la grille et dès qu'une particule entre dans son domaine il fait avancer la particule jusqu'à ce que celle-ci n'y soit plus.

Exemple de modélisation pour 4 processeurs :



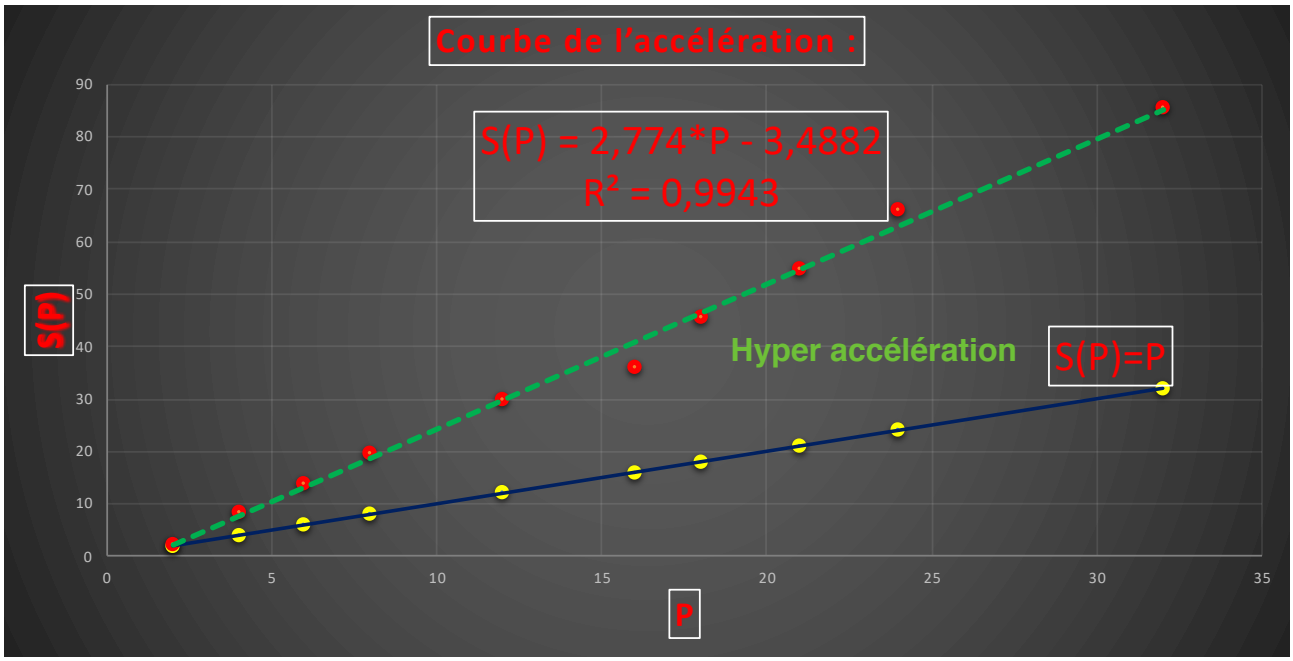
On a opté pour la première stratégie.

2.3 Analyse des résultats :

Calcul réalisé pendant les heures de TP (donc pas sur les CPU « intensive »)

2.3.1 Calcul pour une grille de taille 100*100 et pour 100 tirages :

Nombre de coeurs (proc/ threads)	Temps en secondes	Speed-Up
1 (séquentiel)	76.1	X
2 (1/2)	33.6	2.26
4 (2/2)	9.02	8.44
6 (3/2)	5.51	13.8
8 (4/2)	3.88	19.6
12 (6/2)	2.55	29.8
12 (4/3)	2.54	29.9
16 (8/2)	2.12	35.9
18 (6/3)	1.67	45.6
21 (7/3)	1.39	54.7
24 (8/3)	1.15	66.2
32 (8/4)	0.89	85.5



Corentin GIROUD - Julien PELEGRI

Remarques:

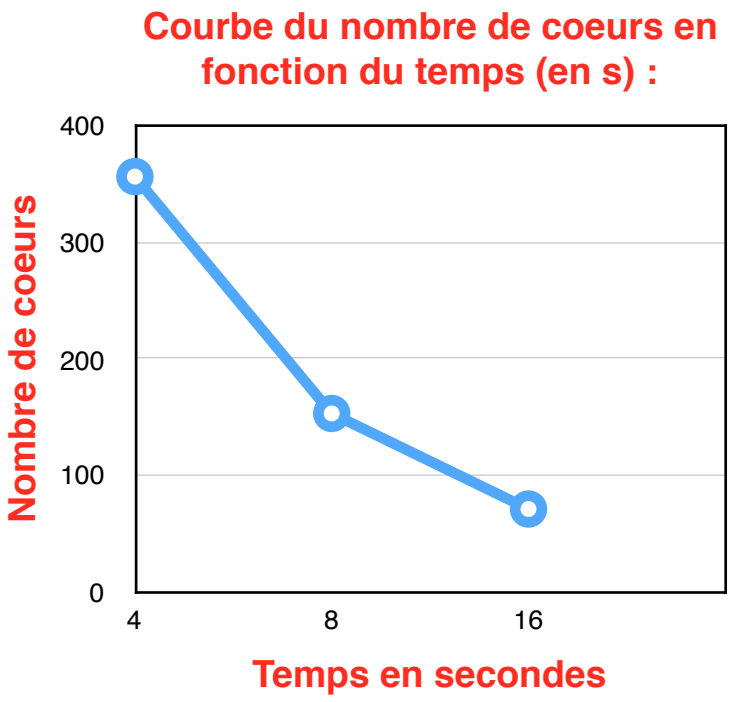
Pour le nombre de coeur on a mis dans le tableau la répartition processeur et nombre de threads même si cela revient au même c'était pour avoir plus de valeurs.

Pour les temps nous avons gardé que trois chiffres significatifs (au delà l'incertitude était trop grande).

Observation: Le calcul en parallèle est ici très efficace on est au dessus de la zone d'hyper accélération, équation de courbe : 2.774*P - 3.4882 = S(P)
De plus le coefficient de corrélation pour un modèle linaire est cohérent, on peut supposer que pour ces valeurs le modèle linéaire prévoit bien le résultat.

2.3.2 Calcul pour une grille de taille 250*250 et pour 100 tirages :

Nombre de coeurs (proc/ threads)	Temps en secondes	Speed-Up
1 (séquentiel)	X	X
2 (1/2)	>10 minutes	X
4 (2/2)	356	X
8 (4/2)	153	X
16 (8/2)	71.1	X



La tendance décroissante est cohérente, néanmoins le peu de valeurs ne rend pas exploitable les résultats.

Corentin GIROUD - Julien PELEGRI

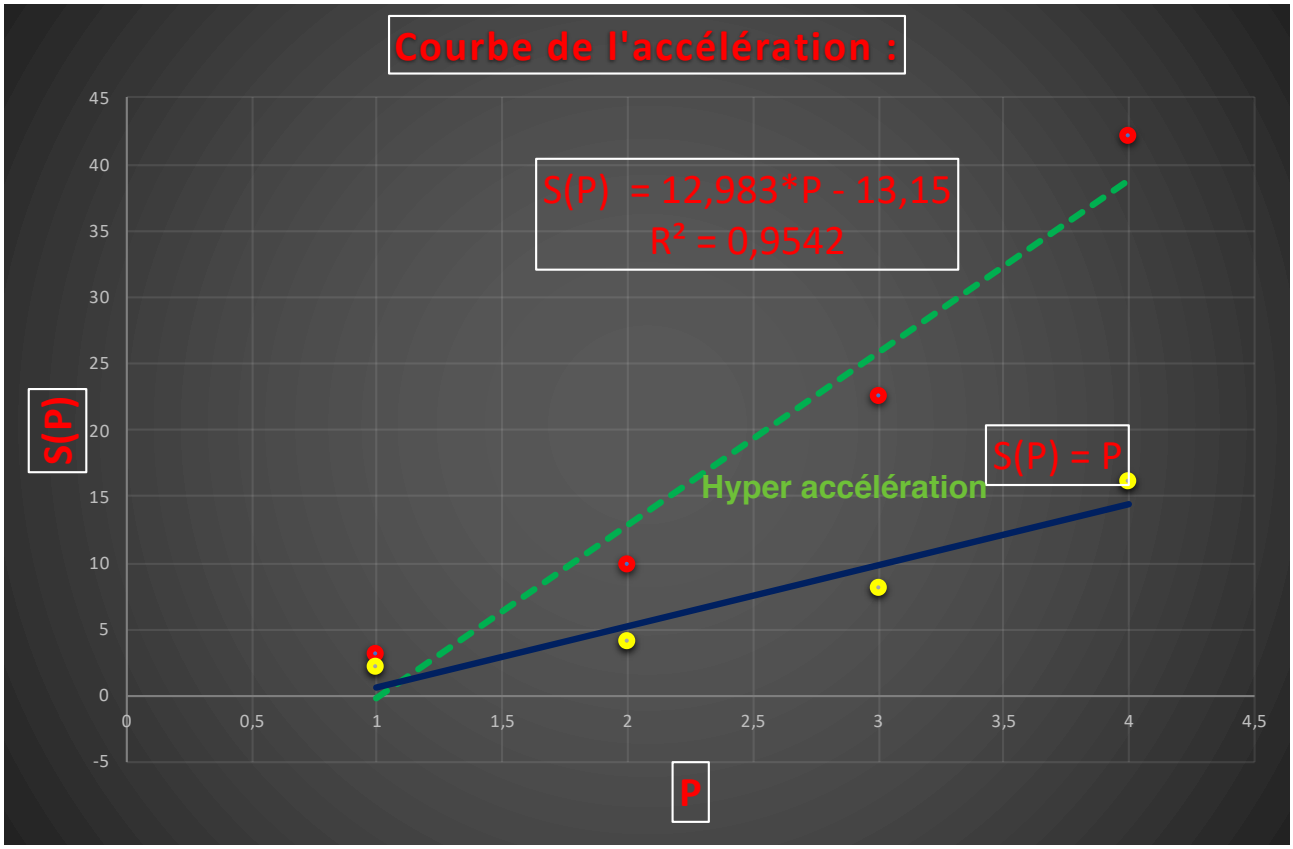
Remarques:

Ici on a pas pu calculer plus de valeurs car la réservation du cluster était finie.

On se rend compte qu'augmenter la taille de la grille augmente grandement la complexité en temps par rapport à l'augmentation du nombre de particules.

2.3.3 Calcul pour une grille de taille 100*100 et pour 250 tirages :

Nombre de coeurs (proc/threads)	Temps en secondes	Speed-Up	Efficacité de calcul E(P) = S(P)/P
1 (séquentiel)	216.7	X	X
2 (1/2)	73.0	2.97	1,5
4 (2/2)	22.2	9.76	2,4
8 (4/2)	9.62	22.5	2,8
16 (8/2)	4.45	48.7	3



Observation: Le calcul en parallèle est aussi intéressant, on a bien le résultat intuitif que la performance augmente pour un plus grand nombre de tirages car on a presque autant de communications en doublant le nombre de

particules ! On peut aussi intuiter (même si les résultats du 2.3.2 ne permettent pas de conclure que la performance de la parllélisation augmenterait plus la taille du problème est grande). Le temps des communications est « relativement » moins important plus la taille du problème augmente.

L'efficacité semble converger vers une valeur limite plus ou moins grande.

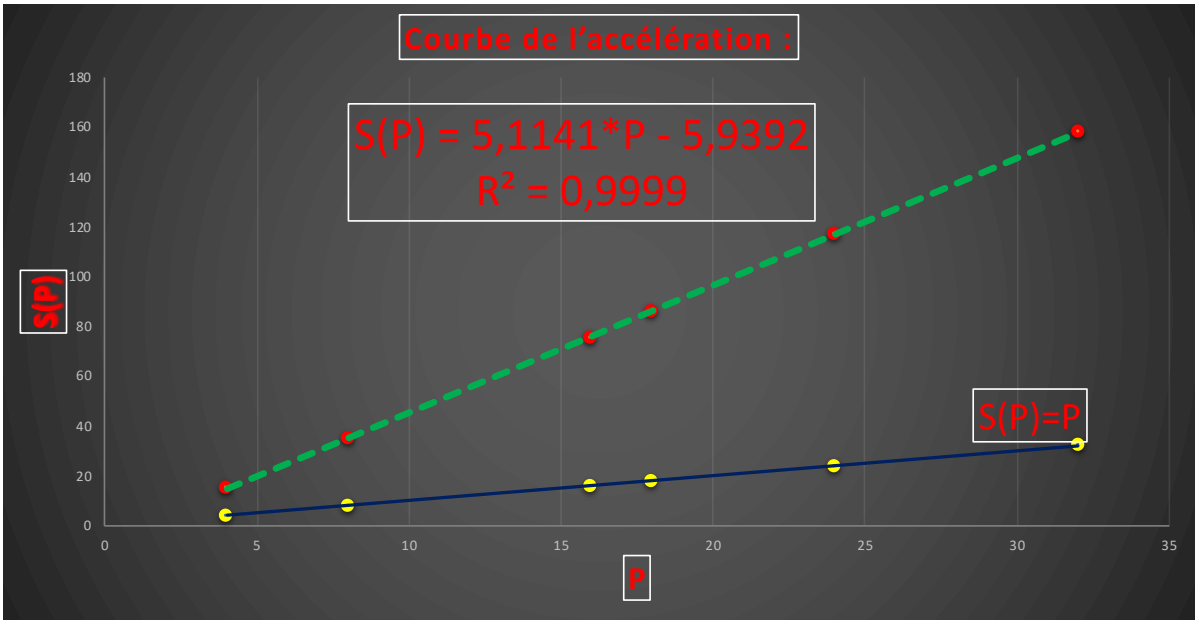
Conclusion: D'après les résultats du 2.3.1 et 2.3.3 l'algorithme parallèle est nettement plus efficace (3 à 12 fois plus rapide) (S(P) > P) pour un problème relativement trop petit (sinon le séquentiel est plus efficace) et la performance augmente avec la taille du problème.

Les résultats sont cohérents on est bien en hyper-accélération et des fois même très largement.

2.4 Résultats complémentaire sur les CPU « intensive » :

2.3.2 Calcul pour une grille de taille 250*250 et pour 100 tirages :

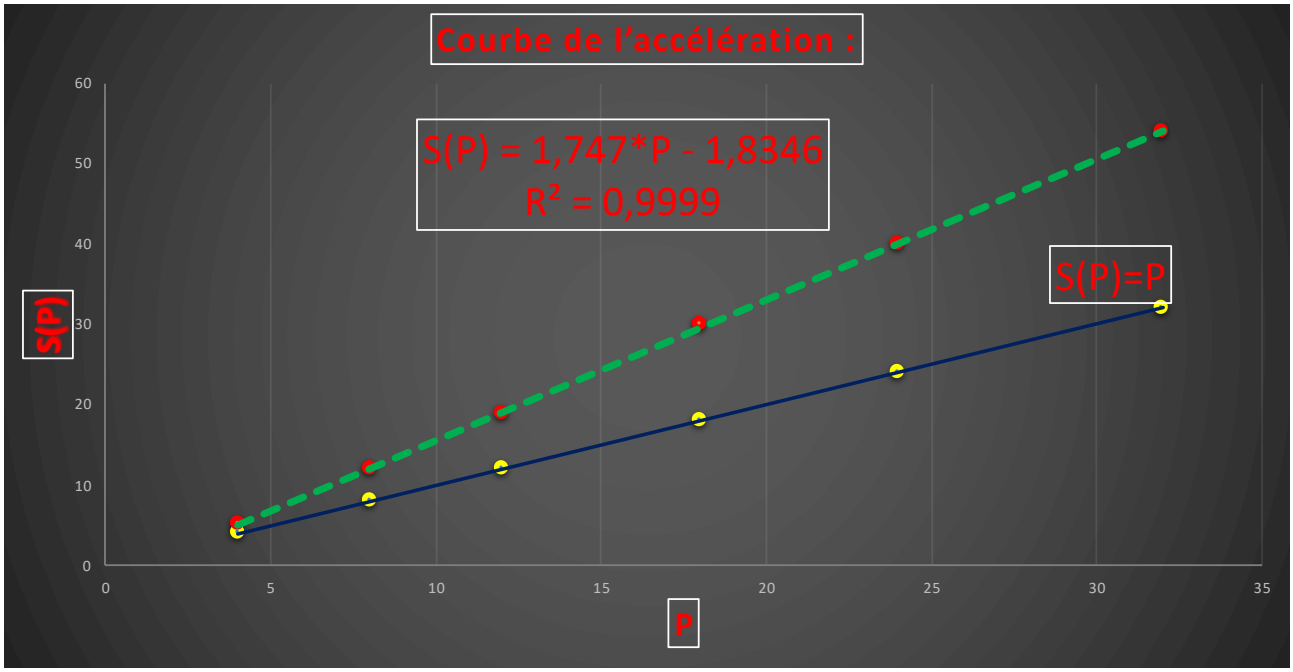
Nombre de coeurs (proc/threads)	Temps en secondes	Speed-Up	Efficacité de calcul E(P) = S(P)/P
1 (séquentiel)	5401	X	X
2 (1/2)	1118	4,8	2,4
4 (2/2)	357	15	3,8
8 (4/2)	153	35	4,4
16 (8/2)	71.6	75	4,7
18 (6/3)	62.8	86	4,8
24 (8/3)	46.5	117	4,9
32 (8/4)	34.1	158	5



Remarque: à premiere vu les résultats entre les CPU « intensive » et les autres sont similaire (intensive un peu plus performant)

2.3.2 Calcul pour une grille de taille 500*500 et pour 300 tirages :

Nombre de coeurs (proc/threads)	Temps en secondes	Speed-Up	Efficacité de calcul
1 (séquentiel)	8976*10^1	X	X
2 (1/2)	1012*10^2	0,89	0,45
4 (2/2)	1724*10^1	5,2	1,3
8 (4/2)	7409	12	1,5
12 (6/2)	4731	19	1,6
18 (6/3)	3036	30	1,7
24 (8/3)	2255	40	1,7
32 (8/4)	1666	54	1,7



Observation :

Les résultats des CPU « intensive » confirment nos conclusions, cependant on avant d'analyser les résultats on pensait que pour la grille 500*500 et 300 tirages le speed-up aurait était plus grand (courbe de coefficient directeur de 1,747 contre 12,983 pour la grille 100*100 et 250 tirages).

De plus, le calcul de l'efficacité montre bien que l'on tend rapidement vers une efficacité limite de notre programme, atteinte plus ou moins rapidement et plus ou moins grand selon la taille de la grille et du nombre de tirages. La parallélisation a ses limites.

Pour des tailles de grilles importante on peut intuiter que la deuxième stratégie (celle ou chaque processeur à une partie de la grille) est plus performante car moins de communications.