Travaux Pratique 9

Contexte : Les données indiquées dans ce rapport ont été obtenues avec un Processeur 6 cœurs et 12 processeurs logiques.

Exercice 1 :

Pour cet exercice, il n'y a pas grand-chose à faire si ce n'est compiler et exécuter. Après plusieurs exécutions, on peut voir que les processeurs gérés par MPI fonctionnent en parallèle comme les threads, de manière aléatoire au niveau de l'ordre d'exécution de ceux-ci.

Exercice 2 :

Pour cet exercice, j'ai bien entendu recopié les fonctions fournies, mais j'ai aussi ajouté un petit affichage pour voir l'exécution du programme. Cela permet de vérifier que la communication est bien synchrone, par exemple. Il faut aussi ajouter les primitives d'initialisation de MPI. Comme la classe Initializer est fournie, on appelle les méthodes init et close de cette classe plutôt que d'utiliser les méthodes MPI pure MPI_Init et MPI_Finalize.

Exercice 3 :

On réécrit le code des diffusions vues en cours.

Dans le cas de la diffusion naïve, on a trois cas très simples :

- Si on est le processeur émetteur, alors on envoie les données.
- Si on le processeur juste avant l'émetteur, alors on ne fait que recevoir les données car on est en bout de chaînes.
- Sinon on reçoit puis on envoie les données.

Pour la diffusion en bitoduc, l'algo est un petit peu plus complexe :

- Si on est le processeur émetteur, alors on envoie toutes les données paquet par paquet dans une boucle.
- Si on le processeur juste avant l'émetteur, alors on ne fait que recevoir les données paquet par paquet dans une boucle, car on est en bout de chaînes.
- Sinon on reçoit un paquet de données de manière synchrone, puis, dans une boucle, on envoie les données de manière asynchrone et on reçoit de manière synchrone (ou pas, pour ce cas, ça n'a pas vraiment d'importante). Il faut attendre que l'envoi se soit effectué avant d'itérer dans la boucle. En fin de boucle, on fait un dernier envoi de données.

Etude des temps en ms :

Diffusion naïve :

Size	T4	T8	T12	T16	T20
1048576	2	5	10	18	26
33554432	70	180	260	360	550

Diffusion en pipeline :

Size	T4	T8	T12	T16	T20
1048576	17	25	30	60	70
33554432	180	250	340	560	750

On remarque que la diffusion en pipeline est toujours plus lente (et ce même pour des tailles plus grandes de données). En principe, si le message est assez gros la diffusion en pipeline devrait être plus intéressante. Donc le message n'est pas assez gros. (ou le problème vient de mon code, mais ça ne me parait pas être ça, car c'est le code donné dans le cours).