Rapport – Travaux Pratique 11

Sommaire

Exercice 1

La fonction `MPI_Comm_split` sert à créer des sous-communications à partir d'une communication plus général. Cette création se gère avec un système de couleurs et de clefs.

On utilise cette fonction pour faire des communications sur tore. Les communications sont donc en deux dimensions, par ligne et par colonne.

Exercice 2

Dans cet exercice nous transposons une matrice par bloc avec une communication sur tore entre les processeurs.

Exercice 3

Nombre de processeurs	Taille de la matrice	Temps d'exécution (ms)
4	(8, 8)	0.1
4	(32, 32)	1.86
4	(128, 128)	24.24
4	(256, 256)	169.53
4	(512, 512)	1 243.12
16	(8, 8)	199.9
16	(32, 32)	199.9
16	(128, 128)	230.0
16	(256, 256)	389.9
16	(512, 512)	980.0
64	(8, 8)	1 530.4
64	(32, 32)	1 519.9
64	(128, 128)	1 670.0
64	(256, 256)	1 789.9
64	(512, 512)	2 849.9

Pour de petites matrices il est plus intéressant d'utiliser peu de processeur, tandis que pour de grandes matrices, 16 processeurs semble approprié.

Exercice 4

Nombre de processeurs	Taille de la matrice	Temps d'exécution (ms)
4	(8, 8)	0.05
4	(32, 32)	4.0
4	(128, 128)	29.2
4	(256, 256)	…
4	(512, 512)	…
16	(8, 8)	209.5
16	(32, 32)	209.6
16	(128, 128)	209.8
16	(256, 256)	319.7
16	(512, 512)	…
64	(8, 8)	1 388.2
64	(32, 32)	1 329.9
64	(128, 128)	1 609.7
64	(256, 256)	1 836.1
64	(512, 512)	2 124.2

L'algorithme de Cannon est plus rapide avec 4 et 16 processeurs que le l'algorithme de base.

Exercice 5

Nombre de processeurs	Taille de la matrice	Temps d'exécution (ms)
4	(8, 8)	0.6
4	(32, 32)	1.6
4	(128, 128)	28.9
4	(256, 256)	…
4	(512, 512)	…
16	(8, 8)	157.1
16	(32, 32)	126.9
16	(128, 128)	189.8
16	(256, 256)	372.3
16	(512, 512)	…
64	(8, 8)	1 208.0
64	(32, 32)	1 299.5
64	(128, 128)	1 341.4
64	(256, 256)	2 555.8
64	(512, 512)	5 447.6

L'algorithme de Fox est encore légèrement que plus rapide que Cannon.

Exercice 6

Nombre de processeurs	Taille de la matrice	Temps d'exécution (ms)
4	(8, 8)	0.1
4	(32, 32)	0.3
4	(128, 128)	4.5
4	(256, 256)	…
4	(512, 512)	…
16	(8, 8)	309.5
16	(32, 32)	283.7
16	(128, 128)	309.9
16	(256, 256)	349.6
16	(512, 512)	…
64	(8, 8)	3 628.4
64	(32, 32)	3 417.2
64	(128, 128)	3 948.1
64	(256, 256)	4 168.3
64	(512, 512)	3 407.7

Je n'ai pas réussi à faire marcher l'algorithme de Snyder mais dans ma version il est plus rapide que les autres algorithme avec 4 processeurs.