UNIVERSITÉ DE STRASBOURG

Master RISE
Parallélisme
2017

Projet : MPI et le jeu de la vie

1 Avant propos

Ce projet est à rendre sur la plateforme Moodle en utilisant le dépôt prévu pour cela. Vous rendrez une unique archive (au format .tar.gz ou .zip obligatoirement) dont le nom aura obligatoirement la forme `NOM_Prenom.extension`. Toutes fichiers qui constituent votre reponse commence avec un rappel de sujet sur lequel il répond et vous identifie avec votre nom.

L'archive devra contenir un fichier `rapport.txt` (ou .doc), où vous donnerez les réponses aux différentes questions et toute information que vous jugerez utile pour décrire votre projet, ainsi que les fichiers sources fournies et demandés incluant le fichier `Makefile` modifié pour les compiler. Vous veillerez à commenter soigneusement vos codes et à respecter le nommage des fichiers demandé dans le sujet. Enfin, respectez au mieux la progression proposée dans ce sujet de manière à répondre au mieux aux questions.

2 Le jeu de la vie

Le "jeu de la vie" se déroule sur un tableau de cases occupées par des cellules, qui évolue au cours du temps : à chaque itération, un nouveau tableau est calculé selon des règles très simples. La notion de voisinage utilisée est le 8-voisinage.

— une cellule qui a 4 voisins ou plus meurt d'étouffement,
— une cellule qui a 1 ou 0 voisin meurt d'isolement,
— une case vide qui a exactement 3 cellules voisines voit la naissance d'une nouvelle cellule.

Voir : `http://fr.wikipedia.org/wiki/Jeu_de_la_vie`

On vous procure les fichiers suivants :

— `functions.h` et `functions.c` qui implémantent des outils pour gérer un jeu de la vie.
— `graph.h` et `graph.c` qui implémantent des fonctions d'affichage.
— `gvie.c` qui réalise le jeu en version graphique. En compilant ce programme et le lançant vous pouvez visualiser le jeu et vous rendre compte qu'après un certain temps la configuration initiale se transforme dans une configuration qui revient après relativement peu d'étappes. Le jeu se repète alors de façon cyclique.
— `gvie_cycle.c` qui réalise le jeu en version sans graphique, mais qui par contre détecte quand le jeu tourne dans un cycle stationnaire, et qui peut nous fournir des information sur la longuer du cycle.

Récupérez les programmes source et lisez et comprenez le code.

2.1 Mise en place d'une version parallèle

Dans tous ce qui suit, vous pouvez supposer que le nombre de procès MPI divise le nombre de lignes de votre tableau de cellules, et que tous les procès peuvent tenir un copie du tableau qui leur est propre et qu'il peuvent tous effectuer l'initialisation de tous se tableau. La parallélisation se portera toujours sur la version `gvie_cycle.c`.

Tâche 1 : Implémentez une première (incorrecte) version parallélisée mais basique nommée `gvie_cycle_mpi1.c`. Dans cette version chaque procès MPI s'occupe d'une rangé de lignes pour calculer le jeu. Après chaque itération chaque procès détecte si sa partie du jeu est devenu cyclique, et le programme se termine par consensus, quand tous les procès ont détecté cette propriété.

Par contre cette version ne fera pas d'autre communication entre itérations. Comme pour calculer les valeurs correctes sur les frontières entre procès on aurai besoin des valeurs actuels des voisins, cette version n'est pas correcte.

2.2 Cycles

Question 1 (répondre dans votre fichier `rapport.txt`) : Prouvez que toute configuration du jeu de la vie qui se joue sur un plan borné, finit dans une configuration cyclique.

Question 2 (répondre dans votre fichier `rapport.txt`) : Si n procès MPI ont trouvé des cycles finaux de longueur $k_0, k_1, \ldots, k_{n-1}$ pour leurs partie du jeu respective, prouvez que la longuer du cycle global est ppcm$(k_0, k_1, \ldots, k_{n-1})$ où ppcm est le "plus petit commun multiple" des n valeurs.

2.3 PPCM

Tâche 2 : Implémentez une deuxième (toujours incorrecte) version parallélisée nommée `gvie_cycle_mpi2.c`. Cette version suit la même stratégie que la précédente, mais pour la détection de la terminaison chaque procès communique sa longueur local de cycle. Une fois le procès 0 détecte la terminaison de tous les procès, il calcul et imprime la valeur pour le cycle global.

Tâche 3 : Implémentez une troisième (toujours incorrecte) version parallélisée nommée `gvie_cycle_mpi3.c`. Pour cette version vous implémentez un opérateur de réduction MPI pour le ppcm et vous utilisez ce opérateur pour détecter la terminaison et la longueur du cycle d'un coup.

2.4 Version correcte

Tâche 4 : Implémentez une quatrième version parallélisée nommée `gvie_cycle_mpi4.c` qui cette fois-ci doit être correcte, c'est à dire pour laquelle les configurations globales parcourues sont toujours exactement les mêmes que pour le programme séquentiel fournie. Pour arriver à cela, utilisez les communications point-à-point appropries pour transférer les informations des lignes voisins (*frontières*) tant que nécessaire.

2.5 Version optimisée

Tâche 5 : Implémentez une cinqième version parallélisée nommée `gvie_cycle_mpi5.c` qui cette fois-ci doit être optimisée. En particulier, elle doit initier des communications non-bloquants des frontières avant la phase majeure du calcul et être capable de commencer une partie du calcul avant que la communication se termine. Une fois cette partie majeure de calcul terminé, elle doit pouvoir attendre la fin des communications et pouvoir effectuer les calculs restants.

2.6 Benchmarking

Tâche 6 : Testez vos cinq programmes avec des tailles (au moins 5) de problèmes différentes et des nombres (au moins 1, 2, 4, et 8) de procès MPI différents. Le but de cette partie est de montrer l'efficacité (ou pas) des différents versions parallèles. Testez sur une machine multi-cœur et, si l'installation vous le permet, en utilisant plusieurs PC de la salle.

Décrivez nous la ou les plates-formes de test (nombre et type de CPU et de cœurs, taille de mémoire etc). Documentez vos tests avec un tableau qui contient pour chacun la moyenne d'au moins 5 run de testes avec même caractéristiques et calculez aussi l'écart-type. Produisez un graphique qui montre l'évolution de vos exécutions en fonction du nombre de procès MPI.