Projet de Conception: Calcul paralléle et évaluation de performance: Application à un simulateur d'évolution artificielle

Jonathan Rouzaud-Cornabas (jonathan.rouzaud-cornabas@insa-lyon.fr)

1 Consignes

Nous vous rappellons que vous bénéficiez de Intel Parallel Studio XE (simple demande sur le site d'Intel).

Nous vous fournissons une machine virtuelle avec clang/llvm 3.8 et aftermath pour visualiser le code parallèle.

Il nous parait important de visualiser les performances de votre code avec Intel Advisor/VTune pour les micro optimizations. Pour l'optimisation du code parallèle, nous vous recommandons Aftermath (un guide d'utilisation est fourni sur moodle). Pour l'optimisation du code CUDA, nous vous recommandons d'utiliser NVidia profiler.

Pour la partie évaluation de performance, il est nécessaire en plus d'avoir un code fonctionnel de fournir les performances en passage à l'échelle fort et faible. Vous pouvez utiliser les timer haute précision (sous partie chrono de la STL) pour calculer les performances.

Passage à l'échelle fort En passage à l'échelle fort, on garde la taille du problème (taille des vecteurs et des matrices) et on augmente le nombre de processeurs. Vous devez essayer cela que le code soit paralléle avec OpenMP, MPI ou/et CUDA. Vous disposez de Grid'5000 pour faire vos tests sur plusieurs architectures. Pour CUDA, n'hésitez pas à nous demander d'exécuter votre code sur des carte NVidia professionnel.

Passage à l'échelle faible En passage à l'échelle faible, on augmente la taille du problème avec l'augmentation du nombre de coeurs.

Plate-forme Vous disposez des cartes graphiques du département en salle 501.208. Vous disposez aussi d'un accés à Grid'5000 (clusters avec Infiniband, GPU, plusieurs architectures avec nombre de coeurs et fréquences différentes). Vous pouvez nous demander d'exécuter votre code sur des cartes Tesla (K40). Toutes autres ressources que vous pouvez accéder sont également possible sans restriction (e.g. Amazon AWS propose des heures gratuites avec des Nvidia K80). Dans ce cas, utilisez les spot instances pour optimiser votre coût de calcul.

2 Implémentation

Une implémentation du modèle (section 4) est disponible à l'adresse suivante: `https://gforge.inria.fr/projects/pdcbioevol/` Aucune optimisation ni parallélisation n'a été faite (c'est le but du PdC).

Pour compiler le programme, vous devez avoir la bibliothéque SFML pour la partie graphique (pas nécessaire, il est possible de s'en passer). Vous devez ensuite générer les fichiers de configuration avec *cmake.*. Puis vous compiler avec *make*.

3 Déroulement du PdC

3.1 Parallélisation/optimisation du code

Optimisation Votre première tâche est de proposer des optimisations et les implémenter pour le code fourni. Pour cela, vous devez vous aider de profilers afin de trouver les parties du code consommatrice de ressources et proposer des améliorations.

Parallélisation Votre seconde tâche est de porter le code pour qu'il supporte au moins une méthode de parallélisation parmi les suivantes: OpenMP, MPI, CUDA et/ou OpenACC. Le but est de supporter 2 méthodes de parallélisation (ou au moins 2 méthodes d'exécution: paralléle en mémoire partagée, paralléle en mémoire distribué ou/et GPU). Vous avez le droit d'utiliser d'autres méthodes de parallélisation ou/et des systèmes d'exécution comme StarPU.

3.2 Rapport: Evaluation de performance

Premièrement, votre rapport doit contenir une description des différentes optimisations que vous avez réalisé avec une évaluation de performance de chacunes. Deuxièmement, votre rapport doit contenir les différentes méthodes de parallélisation que vous avez implémenter, commment vous l'avez fait et les difficultés rencontrés. Finalement, votre rapport doit contenir une évaluation de performance du passage à l'échelle faible et fort de votre code. Ce passage à l'échelle différe suivant la méthode de parallélisation utilisé:

- différents GPUs plus ou moins puissant (plusieurs nombres de CUDA cores)
- différents nombres de coeurs
- différents nombres de machines

4 Modèle

4.1 Résumé

Ce modèle permet de simuler l'évolution d'organismes sur une grille en 2 dimensions. Cette évolution a pour échelle temporelle des pas de temps globaux. Il vous est donné à titre indicatif mais on vous donne une version déjà implémenté en C++.

4.2 Organisme

Un organisme peut, à chaque pas de temps, se dupliquer, se déplacer, mourir, pomper des ressources dans l'environement et/ou en relacher. Chaque organisme est composer d'un pseudo ADN, d'un réseau métabolique ainsi que de pompes et de pseudo flagelle (pour le déplacement).

Pseudo-ADN L'ADN est composé de pseudo pair de base (BP: base pair). On peut différencier 2 grandes parties: celles qui sont dans un bout d'ARN qui seront traduites et ceux qui sont pas dans un bout d'ARN et qui sont dormantes (non traduites).

Voici la liste des BPs disponible:

- $START\_RNA$: Début du séquence ARN ce qui est contenu entre cette BP et la BP END_RNA suivante sera traduit (voir ci-dessous). Cette séquence contient également un motif (un float) qui permettra de calculer la fixation de proteines sur le bloc ARN (voir réseau de régulation). Finalement, elle contient un second float qui correspond à son taux de synthèse i.e. combien elle produira de concentration de proteines.
- END_RNA: Fin d'une séquence ARN ce qui est après ne sera pas traduit et est donc dormant.
- $START\_PROTEIN\_BLOCK$:
- $END\_PROTEIN\_BLOCK$:
- $START\_MOVE\_BLOCK$:
- END_MOVE_BLOCK:
- $START\_PUMP\_BLOCK$:
- END_PUMP_BLOCK:

Bloc de proteines Un bloc de proteines est composé de sous paires de base qui permet de modéliser une proteine. En composant ces sous blocs, on calcul la valeur (float) pour laquelle code la proteine. On peut différencier 2 types de blocs: les nombres (des floats) et les opérateurs arithmétiques (addition, produit, modulo, puissance et cosinus). La traduction d'un bloc de proteine en proteine se fait comme suit:

1. Recherche de la première sous BP contenant une valeur
2. A partir de la position trouvée en 1, recherche de l'opérateur arithmétique qui suit.
3. Recherche de la seconde sous BP contenant une valeur en partant de la position de l'étape 2.
4. Application du calcul entre les 2 valeurs suivant l'opérateur arithmétique trouvé.
5. Conservation de la valeur calculé et retour à l'étape 2.
6. Boucle jusqu'à arriver à la fin des sous BP du bloc.

La dernière valeur calculé est la valeur pour laquelle la protéine code. Suivant cette valeur (bornée entre 0 et 1), la proteine aura une fonction spécifique (en plus de pouvoir participer au réseau de régulation, voir après pour plus de détails) :

- [0.00..0.80] : elle contribue à la fitness i.e. la qualité de son adaptation à l'environement.
- [0.80..0.90] : elle contribué uniquement au réseau de régulation
- [0.90..0.95] : elle produit du poison
- [0.95..1.00] : elle produit de l'anti-poison

Bloc de pompes Un bloc de pompes est composé de 0 ou plusieurs sous paires de base qui codent chacune pour une pompe spécifique. Chaque sous paire de base est composé de 2 valeurs (float) qui codent pour le début et la fin de l'effet de la pompe. Tout comme les proteines, les valeurs des pompes sont définies sur un espace [0..1]. De plus, chaque sous pair de base contient un booléan qui permet de coder si la pompe a un effet d'entrée de protéine (in) i.e. elle prend quelque chose dans l'environement et l'injecte dans l'organisme ou le contraire (out) et donc relache dans l'environement. Finalement, chaque sous pair de base contient une troisième valeur (float) qui permet de coder pour la rapidité de la pompe (le pourcentage de concentration des proteines qui est pris par la pompe à chaque pas de temps).

Bloc de déplacement Un bloc de déplacements est composé de 0 ou plusieurs sous paires de base qui codent chacune pour un déplacement spécifique. Chaque sous pair de base code pour 2 valeurs (float). La première représente la distance maximale, i.e. le nombre de cases dans la grille, à laquelle la pseudo-flagelle permet d'accéder. La seconde représente le nombre maximum de tentative de déplacement la flagelle permet de faire à chaque pas de temps (voir simulation du cycle de vie).

4.3 Equation régulant la vie et la mort

A chaque pas de temps, un organisme peut mourir si il respecte une des régles suivantes:

- La concentration combinée de toutes ces protéines est supérieur à 10.
- La concentration combinée de toutes ces protéines est inférieur ou égale à 0.
- La quantité de poison moins la quantité d'anti poison est supérieur à 0.1.

En plus, il a un risque de mort aléatoire.

4.4 Réseau métabolique

Chaque ARN et chaque protéine contienne un motif qui permet de calculer le taux de fixation. Ce taux de fixation (de la proteine sur l'ARN) permet d'influencer la quantité (concentration) produite par chaque ARN. Ce taux de fixation peut augmenter (ou diminuer) la quantité produite. Ce calcul se fait via un taux qui correspond à une valeur dans une matrice de fixation trouvé par les 2 indexes correspondant aux 2 motifs. Puis en utilisant ce taux et la concentration des protéines, une système d'équation de Hills est résolu pour calculer le nouveau taux de production de l'ARN. En pratique, le réseau est traduit sous la forme d'un système d'équation différentielle ordinaire à résoudre à chaque pas de temps.

4.5 Environement

Les organismes sont localisés sur une grille en 2 dimensions. De plus, chaque case de la grille contient une liste de proteines et leur concentration. A chaque pas de temps, ces protéines se dégradent et se diffusent dans les cases voisines. De plus, chaque case contient un environement cible qui est utilisé pour calculer l'adaptation des organismes. Cette environement cible est un tableau qui contient des valeurs entre 0 et 0.8. L'organisme doit se rapprocher au plus de ces valeurs en combinant ces proteines.

4.6 Adaptation à l'environement: Fitness

L'adaptation à l'environement est calculé en faisant la soustraction des valeurs des protéines de l'organisme et celle de l'environement. Cela donne l'erreur métabolique. Celle-ci est ensuite utilisée pour calculer la fitness.

4.7 Processus de sélection

Quand une case est vide i.e. un organisme est mort. Une compétition (et donc une sélection) a lieu pour savoir si un organisme va se dupliquer pour combler la case vide. Pour cela, le processus de sélection regarde la fitness des organismes des cases voisines et choisi le meilleur pour remplir la case vide.

4.8 Processus de duplication (reproduction)

Quand une duplication a lieu, l'organisme sélectionne rentre dans le processus de reproduction. Pour cela, un nouvel organisme est créé. Il contient une copie du pseudo-ADN qui va subir un processus de mutation (voir les régles de mutation). Il contient également la moitié de la concentration de chaque protéine du père (le père en conservant l'autre moitié).

4.9 Régles de mutation

Quand une reproduction a lieu, un ensemble de mutation peuvent avoir lieu (ou pas) sur le pseudo ADN. Pour chaque type de mutation, un nombre aléatoire est choisi. Il correspond au nombre de mutation de ce type qui aura lieu. Voila la liste des types de mutation:

- Switch: inversion de 2 paires de base pris au hasard dans l'ADN.
- Insertion: insertion d'une pair de base à une position aléatoire dans l'ADN.
- Deletion: suppression d'une pair de base à une position aléatoire dans l'ADN.
- Duplication: duplication d'un segment de l'ADN qui est ensuite insérer à une position aléatoire dans l'ADN.
- Modification: une pair de base est modifié en une autre pair de base de manière aléatoire.

Pour la modification, si la pair de base est un bloc (de proteines, déplacements ou pompes), l'ensemble des sous pair de bases sont également modifiées. Pour l'insertion, si la pair de base est un bloc alors un nombre aléatoire de sous blocs (également aléatoire) sont également créé.

4.10 Simulation du cycle de vie

A chaque pas de temps et pour chaque case et organisme, les grandes étapes suivantes sont déroulées:

1. Traduction de l'ADN en un ensemble d'ARN
2. Traduction de chaque ARN en un ensemble de protéines.
3. Traduction de chaque ARN en un ensemble de déplacements.
4. Traduction de chaque ARN en un ensemble de pompes.
5. Activation des pompes.
6. Calcul du réseau métabolique.
7. Intégration des fonctions de Hills (résolution du système d'équation différentielle).
8. Calcul de la vie ou de la mort de l'organisme.
9. Tentative de déplacement.
10. Tentative de duplication/reproduction.
11. Dégradation des protéines se trouvant sur la case.
12. Diffusion des protéines se trouvant sur la case.
13. Passage au prochain pas de temps et retour à l'étape 5 (ou 1 si c'est le premier pas de vie de l'organisme).

4.11 Visualisations et statistiques

Une fonction de visualisation graphique de la fitness des organismes suivant leur localisation sur la grille vous est fourni. De plus, à chaque pas de temps, les statistiques suivantes sont récoltées pour le meilleur organisme ainsi que la moyenne pour l'ensemble des organismes:

- Fitness
- Erreur métabolique
- Nombre de protéines
- Nombre de protéines fitness
- Nombre de protéines réseau métabolique
- Nombre de protéines poison
- Nombre de protéines anti-poison
- Nombre de pompes
- Nombre de déplacements
- Nombre d'ARN
- Taille du réseau
- Durée de vie de l'organisme
- Nombre de mouvement (dans sa vie)
- Nombre de duplication (dans sa vie)