Parallélisation d'une simulation de feu de forêt

1 Présentation du projet

Ce projet a été inspiré par le rapport d'un projet fait en science de l'ingénieur de l'ENS de Paris-Saclay par Elise FOULATIER en 2021 (code séquentiel et en Python !).

Vous pourrez trouver le travail original sur :
Lien vers le projet original

Ce projet réalise donc la simulation d'un incendie, en tenant compte de la densité de végétation (supposée uniforme avant l'incendie) et de la direction et la force du vent.

La méthode utilisée ici se repose sur une probabilité de propagation du feu en fonction du vent et de la densité de végétation à partir d'une loi empirique reposant sur un modèle de probabilité au sens des moindres carrés.

2 Réalisation séquentielle

On va d'abord s'intéresser aux grandeurs utilisées pour la simulation de l'incendie.

Grandeurs utilisées

On décrit dans un premier temps la zone concernée par l'incendie, qui sera dans notre simulation assimilée à un carré de longueur l, l étant exprimée en kilomètres.

Cette zone carrée est ensuite discrétisée à l'aide de $N \times N$ cellules rectangulaires représentant chacune une case.

La simulation gère deux cartes :

- Une carte incendie décrivant pour chaque case l'intensité de l'incendie (valeur entre 0 et 255).
- Une carte densité de végétation décrivant pour chaque case la densité de végétation qu'on y trouve.

Une case est dite **saine** si elle n'est pas atteinte par le feu. Ces cases dans l'affichage seront représentées comme du vert, dont l'intensité dépendra de la **densité de végétation** (valant uniformément 255 au début de la simulation). La valeur d'une case saine dans la carte de l'incendie gérée par le programme sera nulle.

Une case est dite **brûlante** si le feu y est en train de se propager. Si l'incendie est en cours dans cette case, la valeur sur la carte d'incendie vaudra 255. Si l'incendie est en cours d'extinction sur cette case, la valeur sur la carte sera une valeur entre 127 et 1, la valeur d'une

case incendie étant divisée par deux à chaque pas de temps si l'incendie est en cours d'extinction. Sur une case brûlante, la densité de la végétation diminue de un à chaque pas de temps.

Une case est dite **brûlée** si l'incendie y est éteint et que la densité de végétation est devenue nulle.

Modélisation de l'incendie

On considère pour la simulation deux probabilités P_1 et P_2 :

- La probabilité P_1 correspond à la probabilité qu'une case saine voisine d'une case brûlante soit contaminée par le feu. Cette probabilité dépend de la vitesse du vent et de la position relative de la case saine par rapport à sa case brûlante voisine.
- La probabilité P_2 correspond à la probabilité que l'incendie ayant lieu sur une case brûlante commence à s'éteindre.

Note : Une case voisine dans cette simulation est une case adjacente se trouvant soit au Nord, à l'Ouest, au Sud ou à l'Est de la case courante.

Avant de débuter la simulation, il faudra spécifier :

- La vitesse du vent.
- Les coordonnées lexicographiques de la case d'où démarre l'incendie.

Les lois utilisées pour P_1 et P_2 sont dérivées des lois proposées dans le rapport de projet d'Elise FOULATIER en y rajoutant une gestion de la densité de la végétation à l'aide d'une loi utilisant une fonction logarithmique par rapport à cette densité.

Pour accélérer (en séquentiel) la simulation, on utilise un dictionnaire contenant les coordonnées et les intensités de chaque case **brûlante**.

La clef pour ce dictionnaire est un entier calculé par rapport aux indices lexicographiques de la case représentée à l'aide de la formule $i \times N + j$. On peut retrouver les indices lexicographiques de la case à partir de cet entier en utilisant le modulo et la division entière.

Notons enfin que le côté aléatoire a été éliminé dans le code en calculant pour chaque case une valeur pseudo-aléatoire en fonction des coordonnées lexicographiques de la case et du pas de temps actuel.

3 Travail à effectuer

Il est très conseillé de suivre les différentes étapes que nous allons décrire ici afin d'amener le projet à son terme !

Il faudra également conserver le code pour chaque étape décrit ici et rendre en fin de projet les sources pour chaque étape.

Première étape

- Regarder le nombre de cœurs physiques sur votre machine et la taille des différentes mémoires caches que vous donnerez dans votre rapport.
- Mesurer les temps moyen pris pour chaque pas de temps, puis pour l'affichage et l'avancement en temps.

- Paralléliser l'avancement en temps à l'aide d'OPENMP :
  - Récupérez dans un tableau toutes les clefs contenues dans le dictionnaire et à l'aide d'un indice, parcourez ces clefs pour l'avancement en temps.
  - Vérifiez que vous obtenez exactement toujours la même simulation.
  - Parallélisez la nouvelle boucle avec OpenMP.
- Assurez-vous que vous avez exactement la même simulation qu'en séquentiel !
- Mesurer l'accélération obtenue globalement et uniquement pour l'avancement en temps en fonction du nombre de threads utilisés. Donner les tableaux et les courbes d'accélération dans votre rapport ainsi que votre interprétation des résultats obtenus.

Deuxième étape

- Repartez de la version originale du code.
- Mettre en place l'environnement MPI dans votre code.
- Séparez l'affichage qui sera fait par le processus n°0 du calcul qui sera fait par les processus de numéro non nuls.
- Testez le code avec deux processus et mesurez le temps global moyen pris par itération en temps.
- Interprétez votre résultat par rapport au temps global mesuré sur le code d'origine.

Troisième étape

- Reprenez la parallélisation OPENMP effectuée à la première étape et utilisez-là pour paralléliser l'avancement en temps du code obtenu à la deuxième étape.
- Calculez en fonction du nombre de threads l'accélération globale et interprétez le résultat obtenu.
- Calculez en fonction du nombre de threads l'accélération de l'avancement en temps et interprétez le résultat obtenu (en fonction également de l'accélération globale).
- **BONUS** : essayez de rendre asynchrone l'affichage et l'avancement en temps. Calculez l'accélération globale obtenue ainsi que son interprétation.

NOTE : Pour que la parallélisation OpenMP fonctionne, il est possible que vous devriez rajouter avec OpenMPI l'option `--bind-to none` à `mpiexec` (ou `mpirun`) lors du lancement de votre programme.

Quatrième étape

On se propose de paralléliser l'avancement en temps à l'aide de MPI à partir du code obtenu à l'étape 2.

- Définissez un groupe de communication pour les processus impliqués dans le calcul de la simulation.
- Découpez la zone en tranche en fonction du nombre de processus impliqués dans le calcul.
- Parallélisez le calcul à partir de ce découpage en tranche et en utilisant des cellules fantômes.

- Calculez l'accélération globale ainsi que l'accélération de l'avancement en temps.
- Comparez les résultats obtenus avec les résultats obtenus dans le code écrit pour la troisième étape.
- En analysant vos résultats, suggérez des méthodes afin d'améliorer l'accélération obtenue avec MPI.

REMARQUE : Pour chaque accélération calculée, on donnera un tableau et une courbe dans le rapport final.

NOTE : Si votre PC personnel possède peu de cœurs de calcul (typiquement deux), il est conseillé de calculer les différentes accélérations sur les machines de l'ENSTA !