Projet de OS202 : Parallélisation d'une modélisation de tourbillons sur un tore

Walid Outtaleb – Lorenzo Perrier de La Bâthie

Parallélisation en mémoire partagée

La boucle qui prend du temps et le calcul :

```
cloud = Numeric::solve_RK4_movable_vortices(dt, grid, vortices, cloud);
```

On remarque que cette fonction est composée de trois boucles for.

La première est assez gourmande, surtout si on a un grand nombre de points. On peut donc la paralléliser avec un #pragma omp parallel for, ce qui permet de speed up cette fonction de 15 FPS à 40 FPS. Néanmoins, comme il s'agit surtout des opérations et assignations, ces boucles ne sont pas plus optimisables.

La deuxième et troisième boucles s'exécutent déjà quasiment instantanément donc rien ne sert de la paralléliser.

Enfin, il y a une dernière opération :

```
t_velocity.updateVelocityField(t_vortices);
```

Celle-ci s'exécute à environ à 150FPS mais on peut tout de même la paralléliser pour gagner de légères performances car c'est une double boucle for.

Sachant que l'on n'a pas pu tester l'algorithme sur plusieurs machines, on n'a pas paralléliser avec OMP l'affichage (même si on a remarqué la présence de certaines boucles for potentiellement parallélisables).

En conclusion, la parallélisation en mémoire partagée a permis de speed up le process (sans compter MPI) d'une moyenne de 16 FPS à 30 FPS environ. Soit une augmentation de 90% de performances.

Réflexions sur l'approche mixte Eulérienne–Lagrangienne

Afin de paralléliser le champs de vitesse, on découpe la grille en sous domaines, chacun géré par un processus MPI. Tous les processus ont accès à tous les vortex : cela permet le calcul de la vitesse en chaque point et donc l'actualisation en chacun des sous domaines de la grille sans avoir à se soucier de potentiels effets de bords qui rajouteraient davantage de communications entre les processus. Faire cela est d'autant plus avantageux que le calcul de la position des vortex est en temps constant.

Les nouvelles valeurs des vitesses dans chaque sous domaine sont gather dans le processus de rang 0 qui les affiche.

- Dans le cas d'un maillage de très grande taille : Soit on garde le même nombre de process et le temps de calcul augmente, soit on augment le nombre de processus (on peut connecter de

nouvelles machines par exemple), auquel cas le coût des communications risque de devenir important (notamment le gather du processus 0).

- Etant donné que chaque processus a un nombre fixé de particules à gérer, on évite les effets de bords et le surcroit des communications correspondantes qui seraient nécessairement apparues si on avait fait une séparation en sous domaine. Dans le cas d'un très grand nombre de particules, on se retrouve dans la même situation que précédemment.
- Dans le cas d'un maillage de très grande dimension et un très grand nombre de particules, la parallélisation eulérienne-lagrangienne peut devenir très complexe et nécessiter des solutions de compromis pour garantir des performances optimales.