Simulation du problème à N-corps en astrophysique

Version du 24 février 2015

1 Présentation

1.1 Le problème à N-corps

En astrophysique, la dynamique des galaxies est simulée informatiquement en calculant, à chaque pas de temps, les interactions entre les différentes étoiles qui composent les galaxies. Dans cette simulation, dite problème à N-corps, chaque corps (ou particule) C_i est représenté en 3D par :

— une masse m_i (un flottant) ;
— un vecteur [1] position $\mathbf{P}_i = (px_i, py_i, pz_i)$ (3 flottants) ;
— un vecteur force $\mathbf{F}_i = (fx_i, fy_i, fz_i)$ (3 flottants) ;
— et un vecteur vitesse $\mathbf{V}_i = (vx_i, vy_i, vz_i)$ (3 flottants).

Le temps est découpé en intervalle régulier dt. A chaque pas de temps t, le programme doit calculer l'interaction des forces gravitationnelles pour tous les corps, deux à deux. La force gravitationnelle exercée par le corps C_j sur le corps C_i s'exprime ainsi (en omettant la constante de gravitation universelle $\mathcal{G}$) :

$$\mathbf{F}_{C_j \to C_i} = \frac{m_i.m_j}{(r_{ij}^2 + \varepsilon^2)^{\frac{3}{2}}}(\mathbf{P}_j - \mathbf{P}_i) \tag{1}$$

où r_{ij} est la distance entre les deux corps C_i et C_j, et ε est le *softening* ($\varepsilon = 0.01$ par exemple) qui permet d'éviter que le calcul des forces "explose" numériquement quand r_{ij} devient trop petit.

La résolution dite "directe" du problème à N-corps consiste à calculer pour chaque corps C_i la somme des forces dues à toutes les autres particules, soit :

$$\mathbf{F}_i = \sum_{j=1, j\neq i}^{j=n} \mathbf{F}_{C_j \to C_i} \tag{2}$$

A l'aide du principe des interactions réciproques (selon lequel $\mathbf{F}_{C_j \to C_i} = -\mathbf{F}_{C_i \to C_j}$), on peut diviser par 2 les calculs :

$$\mathbf{F}_i = \sum_{j=i+1}^{j=n} \mathbf{F}_{C_j \to C_i} \tag{3}$$

Une fois le vecteur force obtenu à l'instant t, la phase d'intégration en temps permet d'en déduire la nouvelle vitesse (à l'instant $t+1$) puis la nouvelle position (à l'instant $t+1$) de chaque corps. Pour cela, on calcule d'abord le vecteur accélération $\mathbf{A}_i$ qui vaut, d'après le principe fondamental de la dynamique :

$$\mathbf{A}_i = \mathbf{F}_i / m_i$$

Nous utilisons ensuite le schéma d'intégration en temps de type *Leapfrog* (simple et stable) suivant :

— *Kick* : $\mathbf{V}_{i,t+1/2} = \mathbf{V}_{i,t} + \mathbf{A}_{i,t}.dt/2$
— *Drift* : $\mathbf{P}_{i,t+1} = \mathbf{P}_{i,t} + \mathbf{V}_{i,t+1/2}.dt$
— Calcul des interactions (c'est-à-dire mise à jour des $\mathbf{F}_{i,t}$ et des $\mathbf{A}_{i,t}$)
— *Kick* : $\mathbf{V}_{i,t+1} = \mathbf{V}_{i,t+1/2} + \mathbf{A}_{i,t}.dt/2$

Remarque : les opérations *Kick* et *Drift* ne sont pas effectuées au premier pas de temps où seules les interactions sont calculées.

1. Les vecteurs sont notés **en gras**.

1.2 Le code NBODY_direct

Le code `NBODY_direct` implémente en séquentiel la résolution du problème à N-corps décrite ci-dessus.

Il utilise pour cela des fichiers de données au format NEMO (voir en annexe 5.2) et il offre deux façons différentes de stocker les corps en mémoire : soit en utilisant des "tableaux de structures", soit en utilisant une "structure de tableaux" (voir la macro du préprocesseur `_BODIES_SPLIT_DATA_` dans le fichier `config.h`).

Voir en annexe 5.1 pour l'utilisation de ce code.

2 Travail à effectuer

Le travail à effectuer se décompose en 2 parties.

2.1 Partie 1 : parallélisation MPI

Après avoir choisi le stockage des corps en mémoire le plus approprié, parallélisez le code `NBODY_direct` en mode multi-processus à l'aide de MPI.

Vous pourrez mettre en place les points suivants :

— anneau logique de processus,

— recouvrement des communications par le calcul,

— communications MPI persistantes.

Quelle procédure mettez-vous en place pour vérifier que le résultat de votre calcul parallèle est correct ?
On pourra se baser sur le fait que la somme des vecteurs force de tous les corps doit rester (quasiment) nulle au cours de la simulation.

Le problème à N-corps est-il bien adapté au recouvrement des communications par le calcul ? Pourquoi ?

Analysez les performances obtenues quand le nombre de corps N augmente.

Les salles à votre disposition étant équipées de PC quad-cœur on pourra notamment comparer une exécution parallèle avec un processus MPI par nœud (i.e. par PC) et une exécution parallèle avec un processus MPI par cœur (pour un même nombre total de processus MPI).

2.2 Partie 2 : parallélisation MPI+OpenMP et SIMD

2.2.1 Parallélisation MPI+OpenMP

Mettez en place une parallélisation hybride MPI+OpenMP.

Comment gérez vous les conflits possibles avec OpenMP ?

En considérant un seul nœud multicœur, la parallélisation OpenMP (1 processus avec plusieurs threads OpenMP) apporte-t-elle un gain en performance par rapport à la version "MPI pur" de la première partie (plusieurs processus MPI) ?

En considérant plusieurs nœuds multicœurs, la parallélisation hybride MPI+OpenMP apporte-t-elle un gain en performance par rapport à la version "MPI pur" de la première partie ?

Lors de ces comparaisons, on veillera bien à utiliser la même quantité de ressources matérielles entre les différentes versions parallèles.

2.2.2 Parallélisation SIMD

A l'aide d'*intrinsics* et/ou de directives OpenMP, vectorisez votre code de calcul direct du problème à N-corps.

On pourra d'abord mesurer le gain de performance offert par les unités SIMD pour un code séquentiel, avant de le mesurer pour votre meilleur code parallèle.

3 Travail à remettre

Pour chacune des deux parties, vous devrez remettre le code source, sous la forme d'une archive `tar` compressée et nommée suivant le modèle `projet_HPC_nom1_nom2.tar.gz`. L'archive ne doit contenir aucun exécutable, et les différentes versions demandées devront être localisées dans des répertoires différents. Chaque répertoire devra contenir un fichier `Makefile` : la commande `make` devra permettre de lancer la compilation, et la commande `make exec` devra lancer une exécution parallèle représentative avec des paramètres appropriés. Un fichier `Makefile` situé à la racine de votre projet devra permettre (avec la commande `make`) de lancer la compilation de chaque version.

A la fin du projet, vous devrez remettre un rapport au format `pdf` (de 5 à 10 pages, nommé suivant le modèle `rapport_HPC_nom1_nom2.pdf`) présentant vos algorithmes, vos choix d'implémentation (sans code source), vos résultats (notamment vos efficacités parallèles) et vos conclusions pour les deux parties. L'analyse du comportement de vos programmes sera particulièrement appréciée. Les machines n'étant pas strictement identiques d'une salle à l'autre, on précisera dans le rapport la salle utilisée pour les tests de performance.

4 Quelques précisions importantes

— Le projet est à réaliser par binôme.
— Vous **devez** lire le document intitulé « Projet HPC : conditions d'utilisation d'OpenMPI ». Vous veillerez notamment à n'utiliser qu'une salle à la fois pour vos tests, et vous n'oublierez pas de tuer **tous** vos processus sur **toutes** les machines utilisées à la fin de vos tests.
— Attention aux quotas sur les disques. N'oubliez pas que vous pouvez utiliser le répertoire `/tmp` pour vos fichiers temporaires afin d'éviter de surcharge le système de fichiers (voir aussi section 5.1).
— Le code de la première partie («Parallélisation MPI»), accompagné des slides de votre soutenance (prévoir une soutenance de 10 à 15 minutes, suivie de 5 minutes de questions), est à remettre au plus tard le dimanche 29 / 03 / 2015 à 23h59 (heure locale). Les soutenances de présentation de la première partie auront lieu lors de la séance de TDTP du mardi 31 / 03 / 2015.
Le code de la seconde partie («Parallélisation MPI+OpenMP et SIMD») et le rapport final sont à remettre au plus tard le dimanche 10 / 05 / 2015 à 23h59 (heure locale).
— Les remises se feront par courriel à : `pierre.fortin@lip6.fr`
— En cas d'imprévu ou de problème technique commun, n'hésitez pas à nous contacter pour que nous puissions vous proposer une solution ou une alternative.

5 Annexes

5.1 Utilisation du code NBODY_direct

Le code NBODY_direct est situé sur les machines de la PPTI dans le répertoire :
`/users/Enseignants/fortin/Public/HPC_fev2015/Projet/NBODY_direct`

Dans le répertoire `NBODY_direct`, vous pouvez lancer :

```
$ ./bin/NBODY_direct
```

pour avoir la liste des options disponibles.

Pour simuler par exemple la distribution `baredisk_2048-plummer_2048.nemo`, du temps 0.0 (valeur fixée dans le fichier ou par défaut) jusqu'au temps 10.0, par $dt = 0.1$, avec un softening $\varepsilon = 0.01$, et avec un affichage de la somme des pas des vecteurs force à chaque pas de temps, il faut lancer :

```
$ ./bin/NBODY_direct --in=../data/baredisk_2048-plummer_2048.nemo
                     --tend=10.0 --dt=0.1 --soft=0.01 --sum
```


Pour la sauvegarde d'un fichier NEMO à chaque pas de temps, utilisez l'option `--save` après avoir au préalable remplacer LOGINS pas vos logins dans la macro `RESULTS_DIR` définie dans le fichier `src/main_direct_method.c`. Les fichiers seront sauvegardés dans `/tmp/NBODY_direct_results_<LOGINS>`

La commande `make doc` génère une documentation au format `html` à partir du code source avec l'outil `doxygen` (ouvrir ensuite le fichier `doc/html/index.html` avec un navigateur).

5.2 NEMO

NEMO est un ensemble de logiciels libres pour le problème à N-corps (en astrophysique principalement) : voir `http://bima.astro.umd.edu/nemo/`

Ne recopiez pas le répertoire `/users/Enseignants/fortin/Public/HPC_fev2015/Projet/NEMO` sur votre compte mais utilisez directement celui-ci.

Nous utiliserons le format de données propre à NEMO pour le stockage des données dans les fichiers (des *snapshosts* pour NEMO). C'est un format binaire : des programmes comme `tsf` ou `snapprint` permettent d'en consulter le contenu (pour la documentation sur les programmes NEMO, voir : `http://bima.astro.umd.edu/nemo/man_html/index1.html`).

Remarque : NEMO utilise généralement le shell `tcsh`. Si certaines commandes ou scripts ne fonctionne pas sous `bash`, utilisez `tcsh` et/ou positionnez la variable d'environnement NEMO à `/users/Enseignants/fortin/Public/HPC_fev2015/Projet/NEMO/nemo_cvs`

Les fichiers disponibles sont localisés dans le répertoire : `/users/Enseignants/fortin/Public/HPC_fev2015/Projet/data`

Vous y trouverez diverses distributions avec différents nombres de particules :

- *cube* : particules uniformément réparties dans un cube ;
- *hom* : particules uniformément réparties dans une sphère homogène ;
- *plummer* : modèle de Plummer pour une galaxie (voir `http://en.wikipedia.org/wiki/Plummer_model`);
- *baredisk* : distribution sous forme de disque ;
- *baredisk-plummer* : fusion d'un *plummer* et d'un *baredisk*.

La visualisation de ces fichiers (en 3D, utilisez la souris !) pourra se faire avec le logiciel `glnemo2` (voir `http://projets.lam.fr/projects/glnemo2`). Par exemple :

```
$ ./NEMO/nemo_cvs/bin/glnemo2 ./data/baredisk_32768-plummer_32768.nemo
```

Ceci est aussi valable pour les fichiers de sortie.

Si vous souhaitez recompiler NEMO chez vous sur votre machine personnelle, lancez : `./configure --enable-single` puis dans un shell `tcsh` suivez les instructions données par : `make install`

`glnemo2` est installé avec `NEMO` : pour les dépendances particulières de `glnemo2`, voir le site web.