TP Parallélisme

Compte-Rendu 10/12/2008

Gravitation universelle

Encadrant :

François PELLEGRINI

Binome { Seifeddine HABASSI
Mohamed Amine EL AFRIT }

Table des matières

Introduction

Le problème à N-corps est un problème mathématique fondamental pour l'astronomie classique. Il s'agit de résoudre les équations du mouvements de Newton pour N corps interagissant gravitationnellement connaissant leurs masses, leurs positions et leurs vitesses initiales.

On souhaite simuler l'action mutuelle des forces gravitationnelles agissant sur un ensemble de masses ponctuelles situées dans un plan infini.

Chapitre 1

Conception

On va présenter ici les choix de conception. Les détails spécifiques seront détaillés dans le chapitre suivant.

Le projet consiste à simuler les interactions gravitationnelles entre un ensemble de particules. Ces particules sont distribuées sur un plan supposé infini. La masse est le seule invariant de toute particule.

1.1 Calcul de la force de gravitation

Le début de la simulation est supposé à l'instant 0. Le temps est simulé par une succession de pas séparés d'un délai dt très petit et qui n'est pas forcément constant au cours du temps. À chaque pas, toute particule va subir l'effet de la gravitation causé par toutes les autres particules pendant le délai dt. Cet effet est calculé pour un pas de temps donné. Le calcul des forces de gravitation est basé sur les formules de physique génrale. Pour deux particules a et b de masses m_1 et m_2, distantes de d_{12}, la force appliquée par a sur b est : $F_{21} = G * m_1 * m_2/d^3u$ où u est le vecteur unitaire de direction ab. La force finale F_a appliquée sur a est la somme de toutes les force F_{j1} pour j allant jusqu'au nombre total de particules.

Les nouvelles vitesses et positions peuvent se déduire avec les formules suivantes :
$a_a(t + dt) = F_a/m$
$v_a(t + dt) = v_a(t) + dt * a_a(t)$
$p_a(t + dt) = p_a(t) + dt * v_a(t) + dt^2/2 * a_a(t)$

Pour calculer sa position suivante, une masse a donc besoin de connaître les masses et les positions de toutes les autres particules. Dans le cas où les données sur les particules sont distribuées sur plusieurs processus, le calcul de la force à l'instant t va nécessiter des échanges de données entre les processus (voir chapitre 2). La simulation doit aussi assurer que les particules passent au même instant de l'instant t à l'instant $t + dt$ ce qui implique une synchronisation entre les particules au début de chaque pas de temps.

D'autre part, la norme de la force appliquée par une particule sur une autre croit au fur et à mesure que les deux particules se rapprochent (la norme est en $1/d_3$). Cette force risque donc de diverger et donc amener à de très grandes vitesses. La section suivante détaille ce problme.

1.2 Distance minimale entre deux particules

Bien que dans la vie réelle le temps est continue, la simulation ne peut simuler le temps qu'avec une vision discontinue. En effet, la simulation maîtrise les positions d'une particule aux instants t et $t + dt$ mais n'a pas d'informations sur les positions "intermdiaires" entre les deux instants. Une très grande vitesse peut induire un déplacement très grand entre t et $t + dt$ et donc négliger les effets auxquels serai exposée la particule entre ces deux instants. Un exemple de scénario est qu'une particule avec une très grande vitesse peut dépasser une autre masse sur la direction de sa vitesse sans la toucher.

Pour remdier ce problme, le sujet suggère d'imposer une contrainte sur le déplacement des masses : une particule ne peut se rapprocher de plus de 10% de la masse la plus proche à l'instant courant. Cela impose, une décision collective entre toutes les particules pour la valeur suivante du pas de temps dt.

Le calcul de la position de la particule la plus proche peut se faire en parallèle du calcul de la force de gravitation puisque ce calcul demande le parcourt des positions de toutes les masses du plan. Si la valeur courante de dt va trop rapprocher la particule de un de ses voisins, il recommence le calcul avec une valeur plus petite de dt. Avant le passage au pas suivant, toutes les particules se mettent d'accord sur la valeur la plus petite de dt parmi celles choisies. L'implmentation de ce mécanisme sera dtaillé dans le chapitre 2.

D'autre part, le temps de la simulation ne pouvant pas être infini. Nous avons défini des conditions d'arrêt. Cela va être discuté dans la section suivante.

1.3 Conditions de terminaison

Deux paramètres dans la simulation permettent de décrire l'avancement dans le temps :

- Le pas *pas* définit le nombre d'itérations.
- le temps t incrémenté après chaque itération par la valeur courante de dt.

Nous allons donc utiliser deux conditions de terminaison. Il suffit que l'une soit vrifiée pour que la simulation s'arrête.

- Un nombre de pas maximal.
- Un dt très petit.

Chapitre 2

Réalisation

2.1 Structures des données

Chaque particule est définie par trois paramètres, à savoir, sa masse, sa position et sa vitesse. Donc on a opté pour la structure suivante :

```
typedef struct masse_{
  double m ; /* La masse */
  vector p ; /* Le vecteur position */
}masse;
```

Un vecteur est défini par une structure de deux réels définissant ses coordonnées dans le plan.

La structure masse est définie dans le fichier *masse.h*. Le fichier *masse.c* implémente différents accesseurs pour cette structure. La structure *vector* est définie dans le module *plan.h*. Le fichier *plan.c* implémente des opérations divers sur cette structure (norme, calcul de distance, somme...).

2.2 Distribution des particules et échanges de données

Pour simplifier la distribution dans le programme, on va faire l'hypoèthse que tous les processeurs hébergent le même nombre de particules. Donc si on a N particules et P processeurs, chacun va traiter $N_{loc} = N/P$ particules.

Calculer la somme des forces gravitationnelles nécessite un parcours de toutes les masses par chaque processus. Pour ce faire, les processus doivent échanger les informations concernant les particules "locales".

REMARQUE :
Dans notre structure masse, on n'a pas mis de champs concernant la vitesse de la particule. En effet, lors de l'échange des données entre deux processus, ceux-ci ne s'envoient que la masse et la position.

Le sujet suggere une topologie en anneau logique. Les buffers des masses sont échangs dans cet anneau dans le sens des aiguilles d'une montre. Les forces gravitationnelles sont accumulées dans un tableau local de vecteurs de forces. Pour optimiser les communications, un nouveau type MPI a été défini pour envoyer ou

recevoir un tableau contigu de $3 * N/P$ réels (pour chaque particule locale on a un réel pour la masse et deux pour la position).

```
MPI_Type_contiguous(3*N_loc, MPI_DOUBLE, &mpi_masse) ;
```

Pour tous les pas, les schémas decommunication sont identiques ce qui suèggre l'utilisation des communications persistentes. L'algorithme génrale est le suivant :

Algorithme Schémas des communication
Poster les envois.
Poster les rceptions.
Calculer des contributions des masses reçus l'échange précdent sur les masses locales.
Attendre la fin des communications.
Fin

Lors de chaque pas, un processus calcule les nouvelles positions et vitesses pour chacune de ses masses locales. Donc, on a besoin de synchroniser les diffrents processus pour que toutes les particules soient à tout moment au meme pas. Cela peut être garantit par les communications collectives introduites dans la section qui suit.

2.3 La valeur de dt

On a vu dans la section 1.2 du chapitre précdent, que la mise à jour du pas de temps peut être nécessaire avant le passage au pas suivant. La décision de la nouvelle valeur de dt doit être fait de facon collective pour synchroniser les particules : il faut qu'à chaque pas de la simulation, tous les processeurs aient la même valeur du temps t. Cette décision est faite avec la routine MPI collective MPI_Reduce en utilisant l'opérateur MPI_MIN. Ainsi, la valeur minimale de tous les dt locaux des processus est récuprée sur le processus 0 qui la diffuse avec un MPI_Bcast.

2.4 Visualisation des résultats

La visualisation des résultats est l'un des plus importants aspects de la programmation parallle. Notre programme offre une méthode pour exporter les résultats dans des fichiers textes susceptibles d'être transformés en graphe l'aide de l'outil *Gnuplot*.
Cette représentation offre la possibilité de voir la trajectoire des particules. En effet, l'application crée un fichier par masse indiquant les différentes positions de la particule ainsi que la fréquence de pas et la vitesse courante.

Un exemple de visualisation est donné au chapitre 4

2.5 Importation de fichiers de masses

L'application offre aussi la possibilité d'importer des fichier de masses. Ceci permet d'effectuer des tests personnalisés. Un fichier de masse est formés par une ligne par particule. Chaque ligne donne la masse, la position et la vitesse de la particule. Un exemple de fichier de masse (univers.m) est donnée dans l'archive (dans le repertoire *run*).

```
# fichier de 3 particules
# m   px   py  vx   py
1000000000  0    0    -10  -10
1000000000  3    0    10   10
10    3   1    -10   -10
10  0    -1    10   10
```

Chapitre 3

Mini-manuel d'utilisation

3.1 Configuration

Le fichier *gravitation.h* définit les constantes les plus importantes :

```
/* Constantes */
#define PAS_MAX 50000 /*  Nombre de pas maximal */
/* Parametres de configuration */
int N=10 ; /* Nombre des particules */
int Pas_f=100 ; /* Frequence de sauvegarde*/
```

3.2 Lancement de l'application :

- Lancement à l'aide du script suivant :
  ./run.sh nbProcessus nbParticules fichier_configraution
- Lancement avec le nombre de particules par défaut :
  ./gravitation
- Fixer le nombre de particules :
  ./gravitation nbParticules
- Donner les paramètres des particules dans un fichier :
  ./gravitation nbParticules fichier

Chapitre 4

Tests et résultats

Fichier de particules : *univers.m*

Test avec 4 masses

4 particules : une avec deux très grande masse et deux particules sur les axes.

```
1000000000  0    0    -10  -10
1000000000  3    0    10   10
10    3   1    -10   -10
10  0    -1    10   10
```

Commandes à exécuter : (deux processus)

```
$> make
$> cd run
$> ./test1.sh
```

Les résultats sont sur la figure 4.1.

Test avec 4 masses

4 particules : une avec deux très grande masse. et deux particules sur les axes.

```
1000000000  -2    0    0  0
1000000000  2    0    0   0
10  0    1    10   0
10  1    0    -10  0
10  -1   0    10   0
10  0    -1   -10  0
```

Commandes à exécuter : (deux processus)

```
$> make
$> cd run
$> ./test2.sh
```

Les résultats sont sur la figure 4.2.

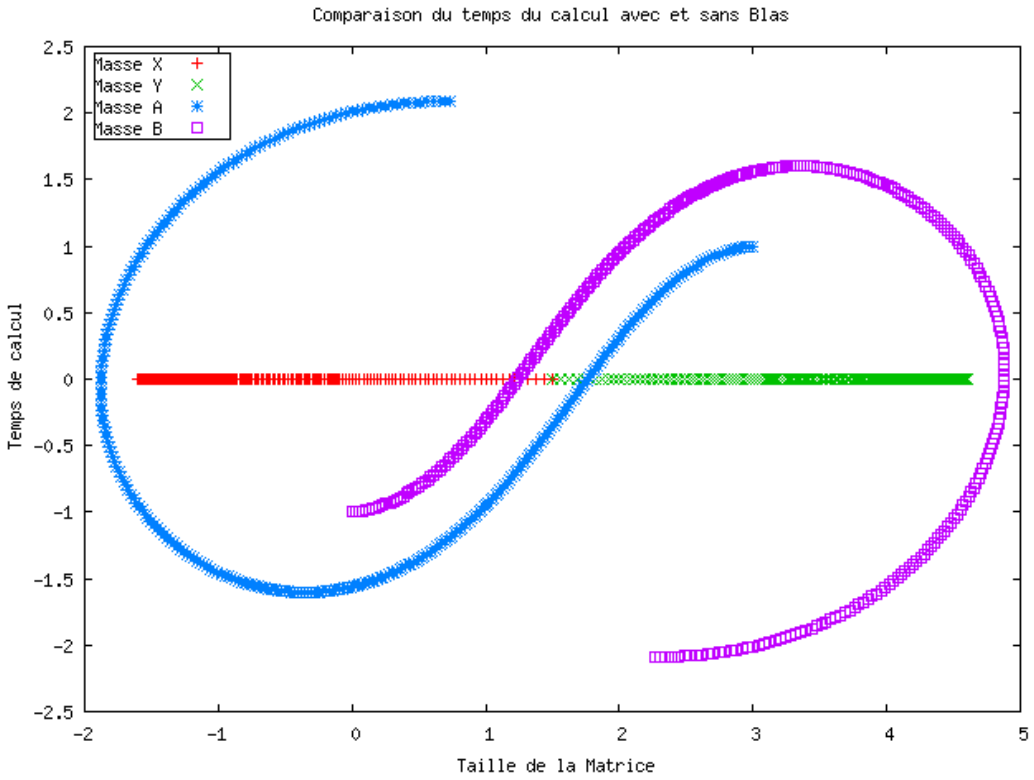

FIG. 4.1 – TEST 4 masses, 2 processeurs

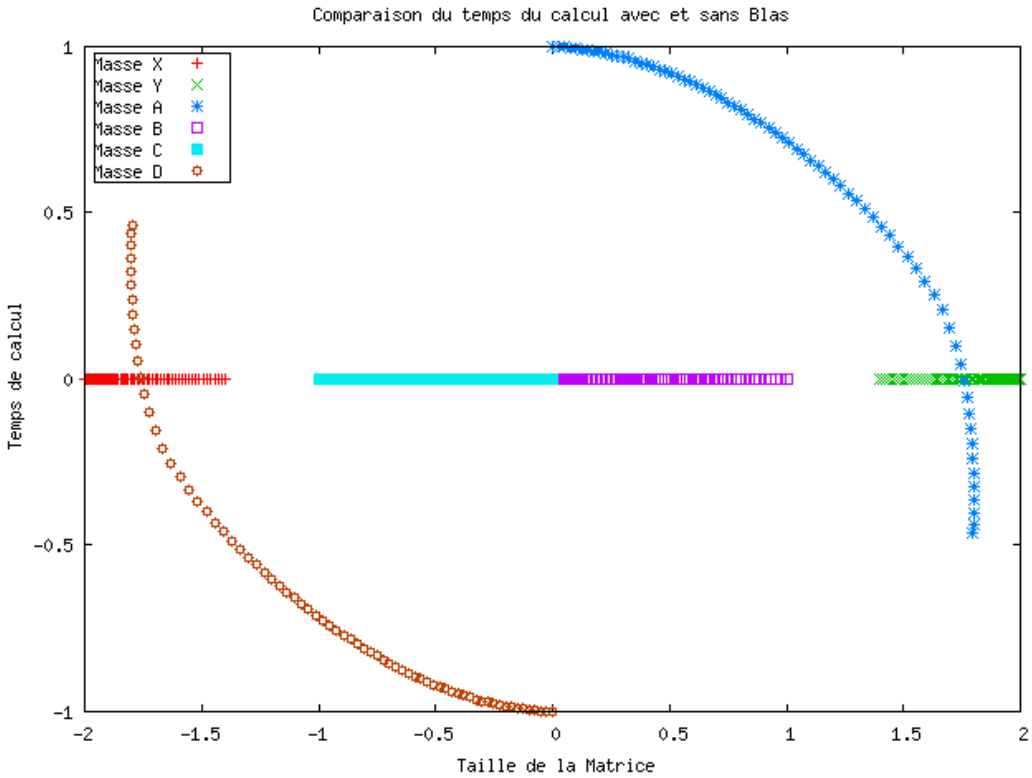

FIG. 4.2 – TEST 6 masses, 2 processeurs

Conclusion

Ce projet nous a permis de voir plus de détails et de maitriser plus de techniques dans la programmation parallèle avec MPI. Celà été à travers la résolution d'un problème réel de simulation physique qui nécessite une grande capacité de calcul.