HPC for Dense Stellar Systems

Cristián Maureira-Fredes

Albert Einstein Institute

October 19, 2017

Contents

Introduction

- The $N-$body problem
- Computational aspects
  - CPU approach
  - GPU Computing
  - Programming strategy

Experiments

Projects

Future work

The $N-$body problem

Definition (1/2)

Predicting motion of a group or celestial objects that interact each other gravitationally.

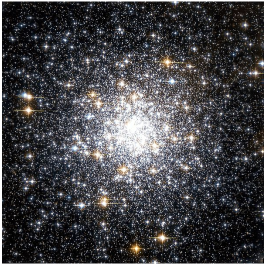


Figure: M69 in Sagittarius.

The $N-$body problem
Definition (2/2)

Purely dynamic problem, in which the bodies orbital evolution is determined exclusive by the gravitational interaction,

$$\ddot{\vec{r}}_i = -G \sum_{\substack{j=1 \\ j \neq i}}^{N} m_j \, \frac{\vec{r}_{ij}}{|\vec{r}_{ij}|^3}, \tag{1}$$

where G is the gravitational constant, m_j is the mass of the $j-$th particle and $\vec{r}_{ij} = (\vec{r}_i - \vec{r}_j)$ the position in *Cartesian* coordinates.

The $N-$body problem

Checking the system evolution

- The initial condition are usually the masses, position and velocity. (Different distributions and shapes: Plummer, King, Dehnen, etc)
- Chaotic nature, the evolution of the system is highly sensitive to the initial conditions.
- The often invariant to check the integration of the system, is the system's energy,

$$E = K + U \tag{2}$$

and sometimes the angular momentum,

$$\vec{L} = \vec{r} \times \vec{p} \tag{3}$$

Collision-less A star just sees the background potential of the rest of the stellar system. A model of this situation is the Barnes-Hut Treecode with a complexity $O(N \log N)$ [1] or the fast multipole method with $O(N)$ [2].

Collisional ("direct-summation") One star integrates all gravitational forces for all stars. This typically scale as $O(N^2)$. A well-known example is the family of algorithm of Aarseth the direct-summation NBODY integrator [3, 4, 5] or KIRA code [6].

The $N-$body problem

Moving the particles (Timesteps) (1/3)

- Individual timesteps allows an accurate treatment of the evolution (handle close encounters)
- An Predictor-Corrector integration scheme:
  1. Select a particle i which has the minimum $\Delta t_i + t_i$ (where Δt_i and t_i are its own timestep and current time)
  2. Calculate gravitational interactions of this particle. (Predict all the other particles to the current time)
  3. Integrate the particle i to its new time $t_{new} = \Delta t_i + t_i$. (Correcting its information using higher derivatives elements)
  4. Get the new timestep.
- Difficult scenario for parallel computing.

The $N-$body problem

Moving the particles (Timesteps) (2/3)

- Block timesteps offers a restriction for the particles timesteps, to be a power of two [7].
- The integration scheme will change from "Select a particle", for "Select a group of particles".
- This time step scheme is popular among $N-$body code, like Starlab [8, 9], Aarseth $N-$body codes [3, 5, 10], ϕGRAPE [11].

The N−body problem

Moving the particles (Timesteps) (3/3)

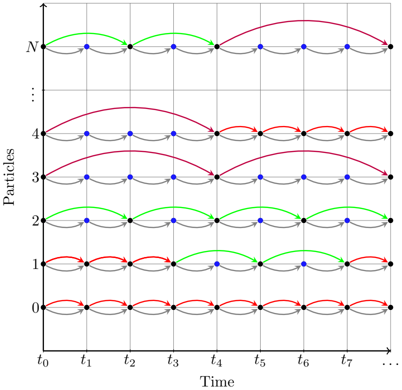


Figure: Block time steps illustration. The different blocks are represented by different colors. Each particle is predicted (not move) at every time t (gray arrows), even if it's not their block time-step (blue circles). The particles will be updated (moved) only in their block time-step (black circles).

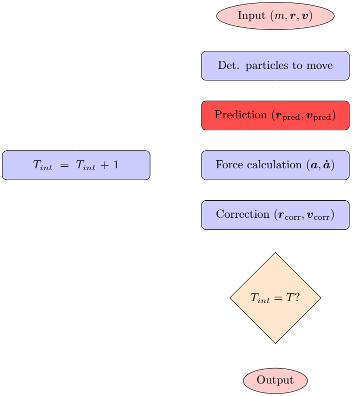
The $N-$body problem

Hermite 4th order (Predictor-Corrector)

Prediction

$$\vec{r}_{i,pred} = \vec{r}_{i,0} + \vec{v}_{i,0}\Delta t_i + \vec{a}_{i,0}\frac{\Delta t_i^2}{2!} + \dot{\vec{a}}_{i,0}\frac{\Delta t_i^3}{3!}$$

$$\vec{v}_{i,pred} = \vec{v}_{i,0} + \vec{a}_{i,0}\Delta t_i + \dot{\vec{a}}_{i,0}\frac{\Delta t_i^2}{2!}$$



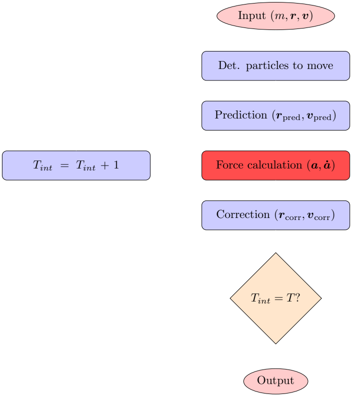
The $N-$body problem

Hermite 4th order (Predictor-Corrector)

Force calculation

$$\vec{a}_{i,1} = \sum_{\substack{j=0\\ j\neq i}}^{N} Gm_j \frac{\vec{r}_{ij}}{(r_{ij}^2+\epsilon^2)^{\frac{3}{2}}},$$

$$\dot{\vec{a}}_{i,1} = \sum_{\substack{j=0\\ j\neq i}}^{N} Gm_j \left[\frac{\vec{v}_{ij}}{(r_{ij}^2+\epsilon^2)^{\frac{3}{2}}} - \frac{3(\vec{v}_{ij}\cdot\vec{r}_{ij})\vec{r}_i}{(r_{ij}^2+\epsilon^2)^{\frac{5}{2}}}\right],$$



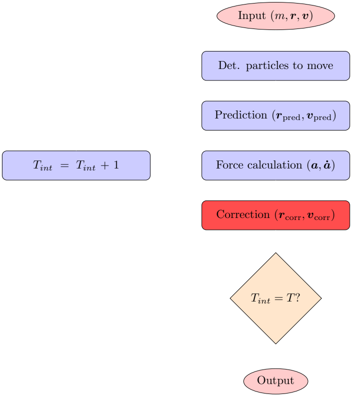
The $N-$body problem

Hermite 4th order (Predictor-Corrector)

Correction

$$\vec{r}_{i,1} = \vec{r}_{i,pred} + \frac{1}{24}\Delta t_i^4 \vec{a}_{i,0}^{(2)} + \frac{1}{120}\Delta t_i^5 \vec{a}_{i,0}^{(3)}$$

$$\vec{v}_{i,1} = \vec{v}_{i,pred} + \frac{1}{4}\Delta t_i^3 \vec{a}_{i,0}^{(2)} + \frac{1}{24}\Delta t_i^4 \vec{a}_{i,0}^{(3)}$$



Computational Aspects

The computational challenge

- The $N-$body codes evolution is related to the available hardware in our time.
- The algorithms with a complexity of $O(N^2)$ or $O(N^3)$ require supercomputers.
  - e.g beowulf clusters, which require a parallelization of the code (NBODY6++ developed by Spurzem et al. [4]).
  - Special-purpose hardware, like the GRAPE (short for GRAvity PipE system [12, 13, 14, 15].
- The literature overview reveals a strong interest on porting the existing codes to the GPU architecture, like e.g. the work of [16, 17, 18] on single nodes or using large clusters [19, 10, 20].

Computational Aspects

Parallel CPU implementation

- Single-core implementation to perform a profiling (`gprof`).
  - Gravitational interaction is the bottleneck.
  - Usually $N_{act} << N$.
- Many-core with OpenMP.
  - `#pragma omp parallel for`
- Many-core with MPI (two implementations)
  - `MPI_Allreduce, MPI_Bcast`

Computational Aspects

Parallel CPU implementation

Listing 1: Reduce every $Nact$ particle

```
for (int i = 0; i < Nact; i++)
{
    ...
    for (int j = 0; j < N; j++)
    {
        gravitational_interaction(...);
    }
    ...
    MPI_Allreduce(...);
}
```

Listing 2: Reduce all $Nact$ particle

```
for (int i = 0; i < Nact; i++)
{
    ...
    for (int j = 0; j < N; j++)
    {
        gravitational_interaction(...);
    }
    ...
}
MPI_Allreduce(...);
```

Computational Aspects

GPU Computing

"Using a GPU (Graphic Processing Unit) together with a CPU to accelerate scientific calculation operations or general purpose calculation"

Figure: NVIDIA® GTX Titan

- CPU,
  - Designed to have a good performance in parallel and non-parallel scenarios.
  - Minimizes the latency experimented by a thread (large cache memory)
- GPU,
  - Designed to perform highly parallel work.
  - Maximizes the throughput of all the threads.

Performance

Capacity of perform individual instructions in a certain time.

Latency

Measure of time delay experienced in a system.

Throughput

Capacity of perform a whole task in a certain time.

GPU Architecture

Computational Aspects

Task parallelism

Each processor perform a different task.

Data parallelism

Each processor perform the same task, but not on the same data set.

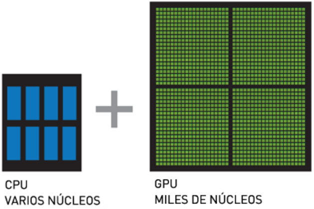


Figure: GPU and CPU core scheme

Computational Aspects

Programming strategy

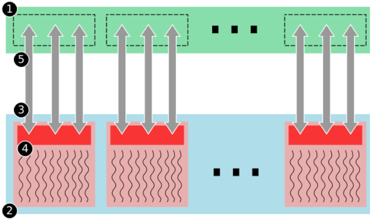


Figure: CUDA Programming strategy

1. CPU memory allocation,
2. GPU memory allocation,
3. Data copying, CPU $\rightarrow$ GPU,
4. Task execution on the data,
5. Data copying, GPU $\rightarrow$ CPU,

Parallelization scheme

$j-$parallelization scheme

Our configuration is based in the idea presented in [10],

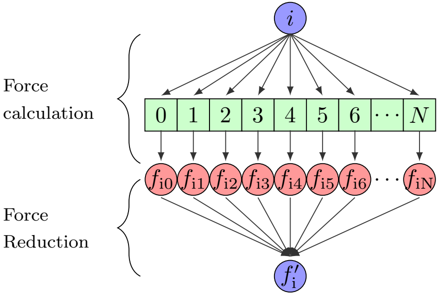
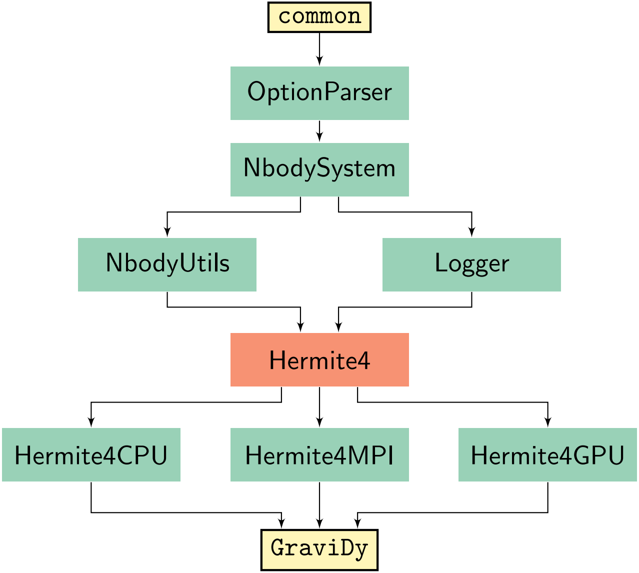


Figure: Parallelization scheme to split the $j-$loop instead of the $i-$loop. In this case, we have two sections, the first is to calculate the force interactions of the $i-$particle with the whole system but by different threads. Then a reduction (sum) is necessary to get the new value for the $i-$particle force.

Implementation

Class diagram



Experiments

Details of the hardware

CPU	Intel(R) Xeon(R) CPU E5-4650 0 @ 2.70GHz
GPU	Tesla M2050 @ 575 Mhz (448 cores).
RAM	24 GB
OS	Scientific Linux release 6.4

Experiments

Integrator scaling

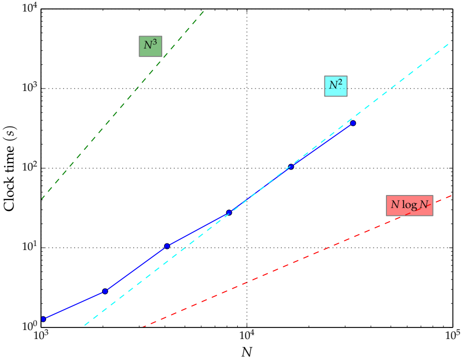


Figure: Clock time of integration from $t = 1$ to $t = 2$ NBU using $\eta = 0.01$ and $\epsilon = 10^{-4}$ using different amount of particles.

Experiments

Clock time comparison

N	CPU	OpenMP	CPU + GPU	MPI-1	MPI-2	GPU
1k	12	8	3	6	2	1
2k	61	34	13	14	7	3
4k	282	162	54	51	27	9
8k	1227	682	208	105	64	23
16k	5542	3227	904	364	317	82
32k	26383	15076	3722	1247	1145	275

Table: Clock time foreach integrator version (in sec).

Experiments

Clock time comparison

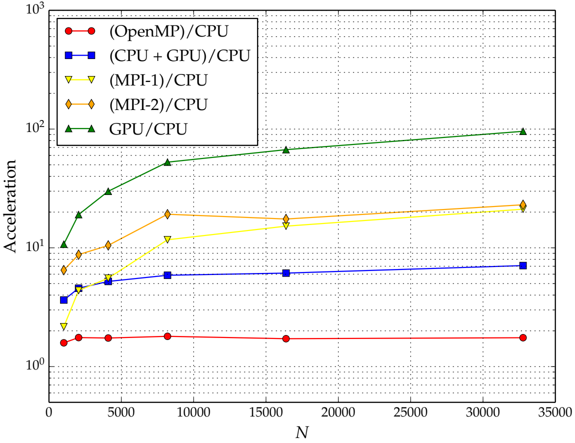


Figure: Acceleration between the implementations described in Table 1

Experiments

Integrator Performance

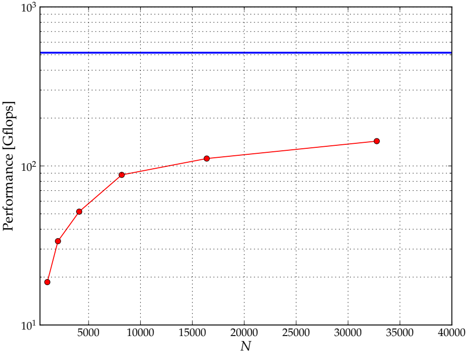


Figure: GPU gravitational interactions performance in GFLOPS for different amount of particles.

Projects
Software

The current version of our new $N-$body code, written in C/C++ and CUDA, called GRAVIDY .

- The current version of our code,
  - Using Hermite 4th order integration scheme.
  - Block timesteps for helping parallelism.
  - Suitable in the energy conservation, reaching errors around $\approx 10^{-9}$ and $\approx 10^{-7}$-
  - OO.
  - Documentation (Doxygen).

Projects

Software development

- Main software
  - Main goal in the development, legibility.
    - Easy to read, modify and understand,
  - Balance between optimization and maintainability.
- Utility scripts.

Projects

Milestones

- Programming
- Physics

Projects

Milestones

- Programming
  - Multi-GPU environment, large clusters (MPI).
- Physics

Projects

Milestones

- Programming
  - Multi-GPU environment, large clusters (MPI).
  - Assess, Parallelize, Optimize, Deploy(APOD) design cycle.
- Physics

Projects

Milestones

- Programming
  - Multi-GPU environment, large clusters (MPI).
  - Assess, Parallelize, Optimize, Deploy(APOD) design cycle.
  - Analysis of FP precision.
- Physics

Projects

Milestones

- Programming
  - Multi-GPU environment, large clusters (MPI).
  - Assess, Parallelize, Optimize, Deploy(APOD) design cycle.
  - Analysis of FP precision.
- Physics
  - Semi-Keplerian systems
    - Globular cluster + BH.

Projects

Milestones

- Programming
  - Multi-GPU environment, large clusters (MPI).
  - Assess, Parallelize, Optimize, Deploy(APOD) design cycle.
  - Analysis of FP precision.
- Physics
  - Semi-Keplerian systems
    - Globular cluster + BH.
  - Neighbors treatment.
    - Near particles are more important than the rest of the system.

Projects

Milestones

- Programming
  - Multi-GPU environment, large clusters (MPI).
  - Assess, Parallelize, Optimize, Deploy(APOD) design cycle.
  - Analysis of FP precision.
- Physics
  - Semi-Keplerian systems
    - Globular cluster + BH.
  - Neighbors treatment.
    - Near particles are more important than the rest of the system.
  - Regularisations (KS, Chain, ...)
    - Remove the softening parameter ϵ^2.

Projects

Milestones

- Programming
  - Multi-GPU environment, large clusters (MPI).
  - Assess, Parallelize, Optimize, Deploy(APOD) design cycle.
  - Analysis of FP precision.
- Physics
  - Semi-Keplerian systems
    - Globular cluster + BH.
  - Neighbors treatment.
    - Near particles are more important than the rest of the system.
  - Regularisations (KS, Chain, ...)
    - Remove the softening parameter ϵ^2.
  - Higher order integration schemes.
    - Hermite 6th order (Faster, and more accurate).

Projects

Milestones

- Programming
  - Multi-GPU environment, large clusters (MPI).
  - Assess, Parallelize, Optimize, Deploy(APOD) design cycle.
  - Analysis of FP precision.
- Physics
  - Semi-Keplerian systems
    - Globular cluster + BH.
  - Neighbors treatment.
    - Near particles are more important than the rest of the system.
  - Regularisations (KS, Chain, ...)
    - Remove the softening parameter ϵ^2.
  - Higher order integration schemes.
    - Hermite 6th order (Faster, and more accurate).
  - Smoothed-particle hydrodynamics (SPH).
    - Different treatment for particle systems.

Projects

Milestones

- Programming
  - Multi-GPU environment, large clusters (MPI).
  - Assess, Parallelize, Optimize, Deploy(APOD) design cycle.
  - Analysis of FP precision.
- Physics
  - Semi-Keplerian systems
    - Globular cluster + BH.
  - Neighbors treatment.
    - Near particles are more important than the rest of the system.
  - Regularisations (KS, Chain, ...)
    - Remove the softening parameter ϵ^2.
  - Higher order integration schemes.
    - Hermite 6th order (Faster, and more accurate).
  - Smoothed-particle hydrodynamics (SPH).
    - Different treatment for particle systems.
  - etc.

HPC for Dense Stellar Systems

Cristián Maureira-Fredes

Albert Einstein Institute

October 19, 2017

J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," vol. 324, pp. 446–449, Dec. 1986.

L. Greendard, *The rapid evaluation of potential fields in particles systems*. PhD thesis, Yale University, New Haven, CT, 1987.

S. J. Aarseth, "From NBODY1 to NBODY6: The Growth of an Industry," *The Publications of the Astronomical Society of the Pacific*, vol. 111, pp. 1333–1346, Nov. 1999.

R. Spurzem, "Direct N-body Simulations," *Journal of Computational and Applied Mathematics*, vol. 109, pp. 407–432, Sept. 1999.

S. J. Aarseth, *Gravitational N-Body Simulations*. ISBN 0521432723. Cambridge, UK: Cambridge University Press, November 2003., Nov. 2003.

S. F. Portegies Zwart, S. L. W. McMillan, P. Hut, and J. Makino, "Star cluster ecology - IV. Dissection of an open star cluster: photometry," vol. 321, pp. 199–226, Feb. 2001.

W. H. Press, "Techniques and tricks for n-body computation," in *The Use of Supercomputers in Stellar Dynamics* (P. Hut and S. L. W. McMillan, eds.), p. 184, Springer-Verlag, 1986.

- S. F. Portegies Zwart, S. L. W. McMillan, P. Hut, and J. Makino, "Star cluster ecology - IV. Dissection of an open star cluster: photometry," vol. 321, pp. 199–226, Feb. 2001.
- P. Hut, "The Starlab Environment for Dense Stellar Systems," in *Astrophysical Supercomputing using Particle Simulations* (J. Makino and P. Hut, eds.), vol. 208 of *IAU Symposium*, p. 331, 2003.
- K. Nitadori and S. J. Aarseth, "Accelerating NBODY6 with graphics processing units," vol. 424, pp. 545–552, July 2012.
- S. Harfst, A. Gualandris, D. Merritt, and S. Mikkola, "A hybrid N-body code incorporating algorithmic regularization and post-Newtonian forces," vol. 389, pp. 2–12, Sept. 2008.
- M. Taiji, J. Makino, T. Fukushige, T. Ebisuzaki, and D. Sugimoto, "Grape-4: A teraflops machine for n-body simulations," in *IAU Symp. 174: Dynamical Evolution of Star Clusters: Confrontation of Theory and Observations* (P. Hut and J. Makino, eds.), p. 141, 1996.

J. Makino and M. Taiji, *Scientific simulations with special-purpose computers : The GRAPE systems*.

Scientific simulations with special-purpose computers : The GRAPE systems /by Junichiro Makino & Makoto Taiji. Chichester ; Toronto : John Wiley & Sons, c1998., 1998.

J. Makino, "Grape-6," *Highlights in Astronomy*, vol. 11, p. 597, 1998.

T. Fukushige, J. Makino, and A. Kawai, "GRAPE-6A: A Single-Card GRAPE-6 for Parallel PC-GRAPE Cluster Systems," vol. 57, pp. 1009–1021, Dec. 2005.

S. F. Portegies Zwart, R. G. Belleman, and P. M. Geldof, "High-performance direct gravitational N-body simulations on graphics processing units," vol. 12, pp. 641–650, Nov. 2007.

T. Hamada and T. Iitaka, "The Chamomile Scheme: An Optimized Algorithm for N-body simulations on Programmable Graphics Processing Units," Mar. 2007.

R. G. Belleman, J. Bédorf, and S. F. Portegies Zwart, "High performance direct gravitational N-body simulations on graphics processing units II: An implementation in CUDA," vol. 13, pp. 103–112, Feb. 2008.

- P. Berczik, K. Nitadori, S. Zhong, R. Spurzem, T. Hamada, X. Wang, I. Berentzen, A. Veles, and W. Ge, "High performance massively parallel direct n-body simulations on large gpu clusters," 2011.
- R. Capuzzo-Dolcetta, M. Spera, and D. Punzo, "A fully parallel, high precision, N-body code running on hybrid computing platforms," *Journal of Computational Physics*, vol. 236, pp. 580–593, Mar. 2013.