A modular, GPU-based, direct-summation $N-$body integrator

Cristián Maureira
Dr. Pau Amaro-Seoane

Albert Einstein Institute

November 22, 2013

Introduction

Motivation

- Dynamical evolution of a dense stellar systems. (N-body Problem)
- Newtonian systems compounded by more than two stars, needs numerical approaches.
- Evolution of the High Performance Computing (HPC).

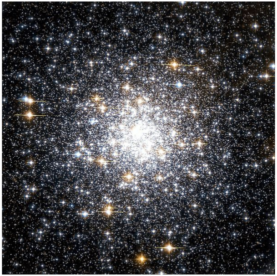


Figure : Globular cluster "Messier 69" in the constellation Sagittarius.

Introduction

N-body algorithms classification

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].

Introduction

The computational challenge

- The N-body codes evolution is related to the available hardware in our time.
- The algorithms with a complexity of $O(N^2)$ 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 [7, 8, 9, 10].
- The literature overview reveals a strong interest on porting the existing codes to the GPU architecture, like e.g. the work of [11, 12, 13] on single nodes or using large clusters [14, 15, 16].

The N-body problem

Definition

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

$$\ddot{\boldsymbol{r}}_i = -G \sum_{\substack{j=1 \\ j \neq i}}^{N} m_j \frac{(\boldsymbol{r}_i - \boldsymbol{r}_j)}{|\boldsymbol{r}_i - \boldsymbol{r}_j|^3}, \quad (1)$$

where G is the gravitational constant ($6.67384 \times 10^{-11} m^3 kg^{-1} s^{-2}$), m_j is the mass of the jth particle and $\boldsymbol{r}_j$ the position in *Cartesian* coordinates.

Note

We denote vectors by bold fonts.

The N-body problem

Checking the system evolution

- The initial condition are usually the masses, position and velocity.
- Chaotic nature, the evolution of this systems will depend of the initial parameters.
- The often invariant to check the integration of the system, is the system's energy,

$$E = \frac{1}{2}\sum_{i=1}^{N} m_i \boldsymbol{v}_i^2 - \sum_{i=1}^{N}\sum_{j>i}^{N} \frac{G m_i m_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|}, \tag{2}$$

where $\boldsymbol{v}_i$ is the velocity of the particle i.

The N-body problem

Particle's time steps

- The real scenario, individual time steps.
  - Hard scenario for parallel computing.
- Forming groups of particles, block time steps scheme [17].
  - This time step scheme is popular among N-body code, like Starlab [18, 19], Aarseth N-body codes [3, 5, 15], ϕGRAPE [20], which gives us the possibility to check our algorithm behavior.

GPU Computing

Introduction

- *“Using a GPU (Graphic Processing Unit) together with a CPU to accelerate scientific calculation operations or general purpose calculation”*

Figure : NVIDIA® GTX Titan

GPU Computing

Features

- 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 Computing

Architecture

Task parallelism

Each processor perform a different task.

Data parallelism

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

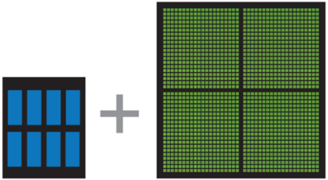


Figure : GPU and CPU core scheme

GPU Computing

Functionality

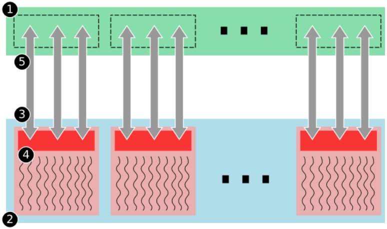


Figure : CUDA Programming strategy

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,

Implementation

Integrators details

- The code structure,
- Integration scheme,
- Parallelization scheme.

Implementation

The code

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

Implementation

The code

For the development of GRAVIDY , we have followed the next steps:

- Serial implementation,
- Profiling and performance assessment,
- Parallelization of the hot-spots,
- Optimization,

Note

The same as CUDA C Best Practices [21], called the APOD cycle, (Assess, Parallelize, Optimize and Deploy).

Implementation

The code

- OOP and SoA.
- Double-precision, importance of accuracy.
  - on GPU, this reach the half of the theoretical maximum performance peak.
  - NVIDIA Tesla C2050/M2050, single precision peak in GFLOPs is 1030.46, and only 515.2 with double precision.

Note

A posible future upgrade is to use mixed-precision [22], or pseudo double-precision [23] to achieve a better performance in our code.

Prediction-correction method

Introduction

In a *N*-body system

The force acting on each particle varies smoothly.

- Interpolation for the time interval extension (prediction).
- Reduce the amount of the force calculation.
- By finite differences it is possible to get the higher order derivatives of the force, which are used to add precision (correction).

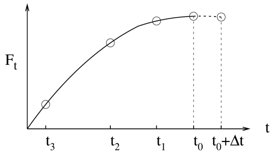


Figure : Polynomial fit of the gravitational force in function of time.

Prediction-correction method

Block time steps

- Reduce the amount of predictions,
- Enhance the parallelism,
- Based on an adaptive system.

$$\Delta t_i = \sqrt{\eta \frac{|\boldsymbol{a}_i||\boldsymbol{a}_i^{(2)}| + |\boldsymbol{a}_i^{(1)}|^2}{|\boldsymbol{a}_i^{(1)}||\boldsymbol{a}_i^{(3)}| + |\boldsymbol{a}_i^{(2)}|^2}}, \qquad (3)$$

Block definition

$2^n \Delta t_s \leq \Delta t_i < 2^{n+1} \Delta t_s$

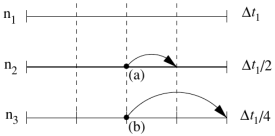


Figure : Time step hierarchical levels.

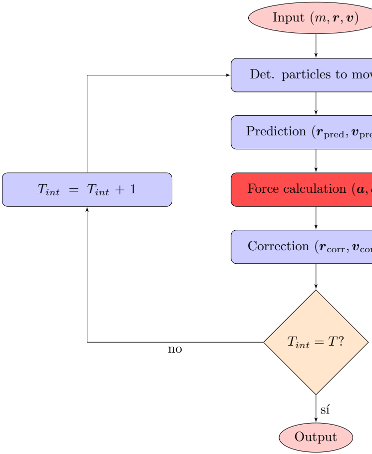
Prediction-correction method

Hermite scheme

Prediction

$$\boldsymbol{r}_{i,pred} = \boldsymbol{r}_{i,0} + \boldsymbol{v}_{i,0}\Delta t_i + \frac{1}{2!}\boldsymbol{a}_{i,0}\Delta t_i^2 + \frac{1}{3!}\dot{\boldsymbol{a}}_{i,0}\Delta t_i^3 \quad (4)$$

$$\boldsymbol{v}_{i,pred} = \boldsymbol{v}_{i,0} + \boldsymbol{a}_{i,0}\Delta t_i + \frac{1}{2!}\dot{\boldsymbol{a}}_{i,0}\Delta t_i^2 \quad (5)$$



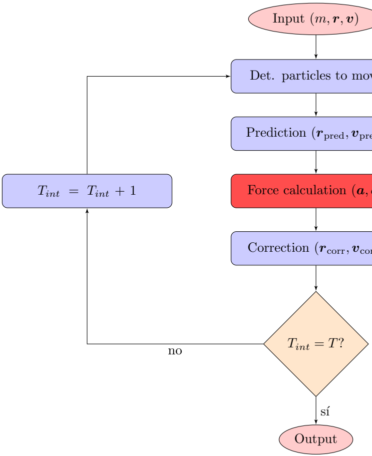
Prediction-correction method

Hermite scheme

Force calculation

$$\boldsymbol{a}_{i,1} = \sum_{\substack{j=0 \\ j \neq i}}^{N} G m_j \frac{\boldsymbol{r}_{ij}}{(r_{ij}^2 + \epsilon^2)^{\frac{3}{2}}}, \tag{6}$$

$$\dot{\boldsymbol{a}}_{i,1} = \sum_{\substack{j=0 \\ j \neq i}}^{N} G m_j \left[\frac{\boldsymbol{v}_{ij}}{(r_{ij}^2 + \epsilon^2)^{\frac{3}{2}}} - \frac{3(\boldsymbol{v}_{ij} \cdot \boldsymbol{r}_{ij})\boldsymbol{r}_i}{(r_{ij}^2 + \epsilon^2)^{\frac{5}{2}}} \right], \tag{7}$$



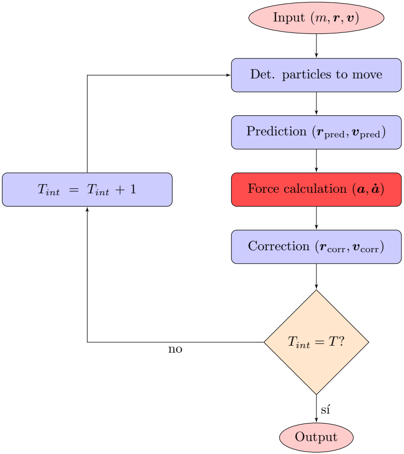
Prediction-correction method

Hermite scheme

Correction

$$\boldsymbol{r}_{i,1} = \boldsymbol{r}_{i,pred} + \frac{1}{24}\Delta t_i^4 \boldsymbol{a}_{i,0}^{(2)} + \frac{1}{120}\Delta t_i^5 \boldsymbol{a}_{i,0}^{(3)} \tag{8}$$

$$\boldsymbol{v}_{i,1} = \boldsymbol{v}_{i,pred} + \frac{1}{4}\Delta t_i^3 \boldsymbol{a}_{i,0}^{(2)} + \frac{1}{24}\Delta t_i^4 \boldsymbol{a}_{i,0}^{(3)} \tag{9}$$



Parallelization scheme

Force calculation

for i in $N_{act}[\,]$

  for j in $N[\,]$

    force_calculation(i,j)

Figure : Force calculation code illustration. The process itself belongs to a second level `for` loop. Thanks to the integrator scheme, we have a reduction of the step complexity from $O(N^2)$ to $O(N_{act}N)$

Parallelization scheme

Calculation relations

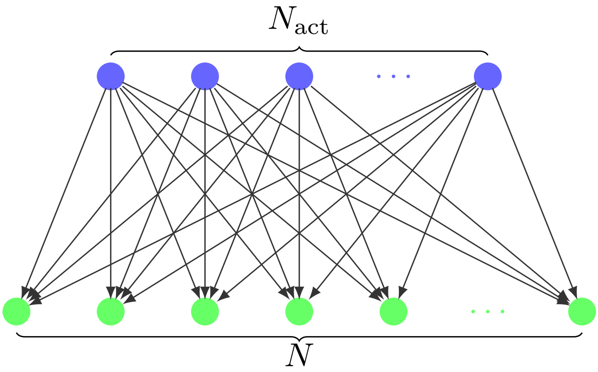


Figure : Relation between the particles which will be updated in a certain integration time (N_{act}) and the whole set of particles (N). The relation between the active particles and the others is $N_{\mathrm{act}} << N$

Parallelization scheme

Threads task

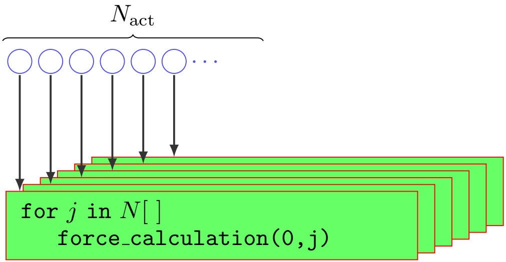


Figure : For each $i-$particle in N_{act} a `for` will be through the whole set of particles and perform the calculation of the gravitational interaction between i and N particles. The scheme could be know as $i-$parallelization, since it assumes that each thread will be one N_{act} particle.

Parallelization scheme

Tiles scheme

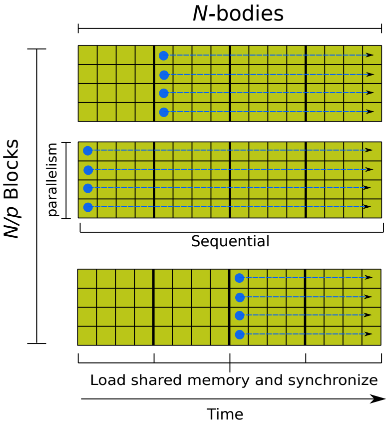


Figure : Grid configuration using the *tiles* approach (This figure is based on [24])

Parallelization scheme

$j-$parallelization scheme

Our configuration is based in the idea presented in [15],

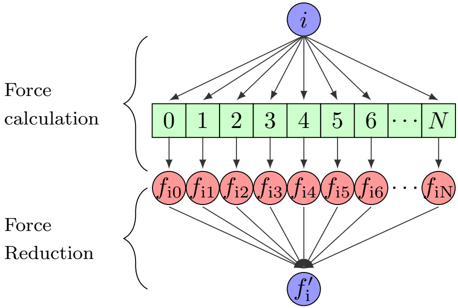


Figure : Parallelization scheme to split the $j-$loop instead of the $i-$loop. In this case, we have two sections, the first is to calculate the force interactions of the $i-$particle with the whole system but by different threads. Then a reduction (sum) is necessary to get the new value for the $i-$particle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

Experiments

Computational specs

UTFSM Cluster, GPU node. (Chile)

CPU	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz (24 cores)
GPU	Tesla M2050 @ 575 Mhz (448 cores).
RAM	24 GB
OS	Scientific Linux release 6.4

Table : Hardware and Software settings.

Experiments

Units and systems

- Equal-mass Plummer sphere [25].
- Standard N-body units for the calculations and resulting output of the code [26, 27],

N-body Units

- Total mass, $\sum_{\mathrm{i=0}}^{N} m_i = 1$.
- Gravitational constant, $G = 1$.
- Total energy, $E_{\mathrm{tot}} = K + U = -0.25$,

Experiments

η and the energy conservation

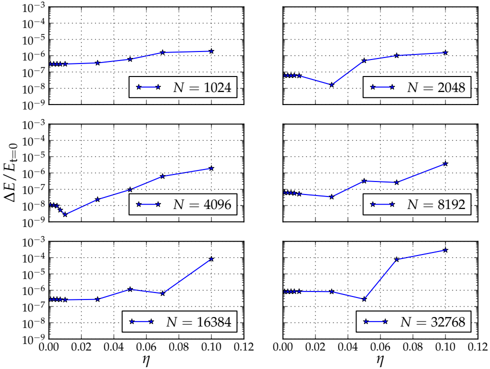


Figure : Cumulative energy error up to $t = 1$ NBU as a function of η. All the plots represent Plummer spheres with different amount of particles.

Experiments

η and the clock time

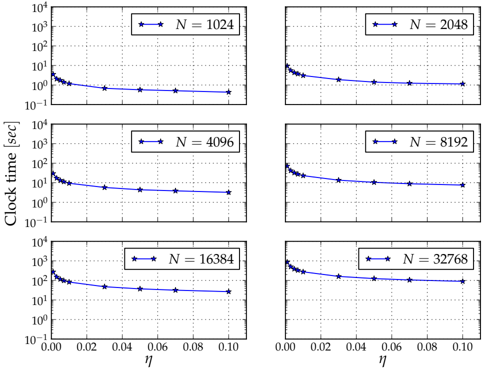


Figure : Clock time up to $t = 1$ NBU as a function of η. All the plots represent Plummer spheres with different amount of particles.

Experiments

Integrator scaling

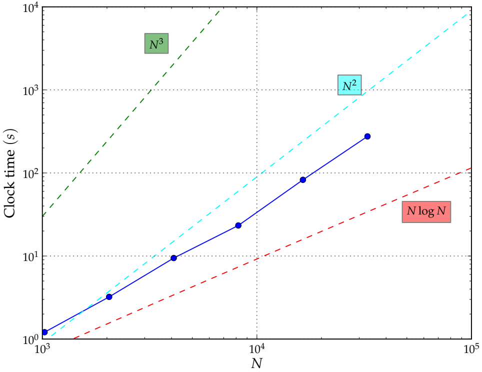


Figure : Clock time of integration up to $t = 1$ NBU using $\eta = 0.01$ and $\epsilon = 10^{-4}$ using different amount of particles.

Experiments

Clock time comparison

N	CPU (single thread)	CPU + OpenMP (many threads)	CPU + GPU (mixed approach)	GPU (multi threads)
1k	12.98 [s]	8.19 [s]	3.57 [s]	1.21 [s]
2k	61.32 [s]	34.94 [s]	13.42 [s]	3.22 [s]
4k	282.98 [s]	162.64 [s]	54.28 [s]	9.45 [s]
8k	1227.40 [s]	682.56 [s]	208.91 [s]	23.31 [s]
16k	5542.35 [s]	3227.91 [s]	904.82 [s]	82.63 [s]
32k	26383.71 [s]	15076.40 [s]	3722.92 [s]	275.53 [s]

Table : Clock time foreach integrator version.

Experiments

Clock time comparison

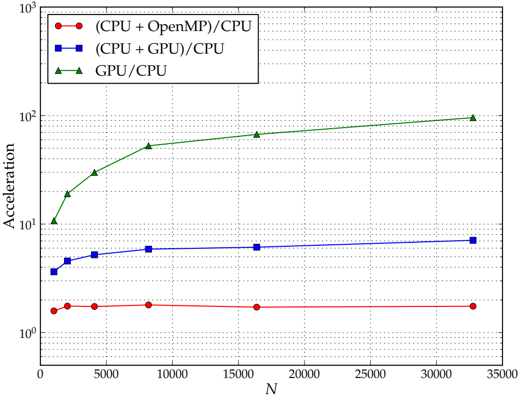


Figure : Acceleration between the implementations described in Table 2

Experiments

Integrator Performance

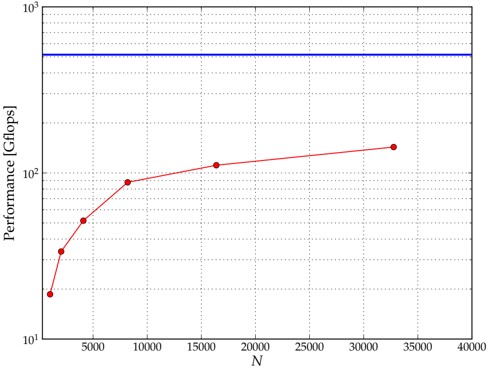


Figure : GPU gravitational interactions performance in GFLOPS for different amount of particles.

Experiments

Lagrange radii

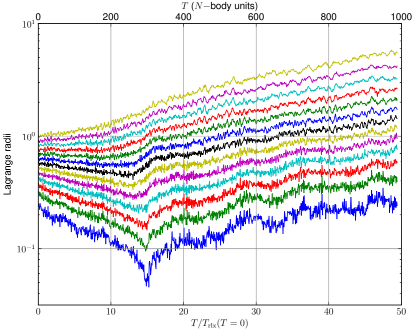


Figure : Lagrange radii of an N-body system with 1024 particles. The radii distribution are $5\%, 10\%, 15\%, \ldots, 65\%$ of the total mass. The core collapse is reached at $T_{\rm cc} \approx 15\,T_{\rm rh}$. The half-mass relaxation time is $T_{\rm rh} = 20.24[nbu]$

Experiments

Core radius and density evolution

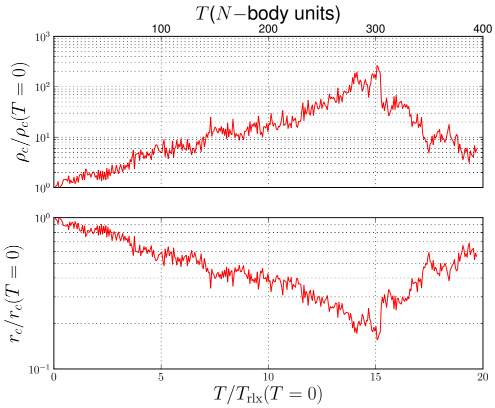


Figure : Evolution of the core radius and core density until core collapse in a system with $N = 1024$

Experiments

Long term integration

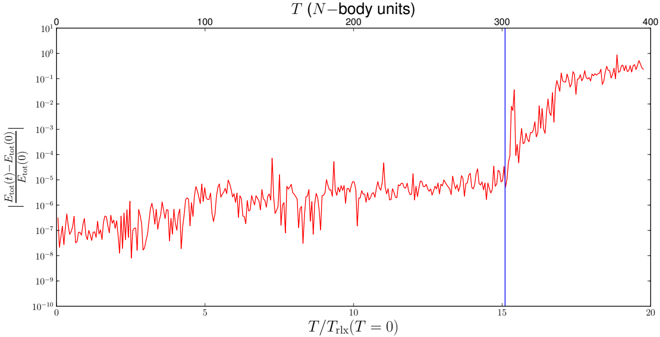


Figure : Energy conservation in a long time integration of a system with $N = 1024$

Challenges and future work

We presented a first version of our new N-body code, written purely in C/C++ and CUDA, called GRAVIDY .

- The current version of our code,
  - evolves a globular cluster, using and Hermite 4th order integration scheme, using block time steps.
  - has a suitable in the energy conservation, reaching errors around $\approx 10^{-9}$ and $\approx 10^{-7}$-

Challenges and future work

Features

- Iterative and incremental development (APOD cycle),
- Modular code,
- Documentation,
- Hermite 4th order integrator scheme,
- Block time steps,
- GPU computing to improve the hot-spots,

challenges and future work

the next steps (1/2)

- Continue the research on N-body integrators, like higher order Hermite schemes.
- More physical phenomena treatments,
  - Neighbors treatment, to handle the binary formation, This has been proved by using a time-symmetric integration scheme for the binary system, which is described in the work of [28].
  - Handle semi-Keplerian systems, N-body systems with a central massive particle [29], like a massive black hole (MBH) in the galactic center, or a star in a protoplanetary system, etc.
  - Post-Newtonian terms into the acceleration and its derivatives.

challenges and future work

the next steps (2/2)

- Porting the code towards large GPU clusters, multi-GPU environment.
- Studying new computational improvements,
  - Numerical precision,
  - Mixed-parallel schemes,
  - Different architectures,
- Use GPU Computing to solve additional Astrophysical problems,
  - Smoothed-particle hydrodynamics (SPH).

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," *MNRAS*, vol. 321, pp. 199–226, Feb. 2001.

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," *PASJ*, 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," *New Astronomy*, 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," *New Astronomy*, 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," *New Astronomy*, 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.
- K. Nitadori and S. J. Aarseth, "Accelerating NBODY6 with graphics processing units," *MNRAS*, vol. 424, pp. 545–552, July 2012.
- 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.
- 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.

S. Harfst, A. Gualandris, D. Merritt, and S. Mikkola, "A hybrid N-body code incorporating algorithmic regularization and post-Newtonian forces," *MNRAS*, vol. 389, pp. 2–12, Sept. 2008.

NVIDIA, "Cuda c best practices guide." `http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html`, 2012.

Accessed: 2012-07-04.

S. J. Aarseth, "Direct methods for N-body simulations.," in *Multiple time scales, p. 377 - 418* (J. U. Brackbill and B. I. Cohen, eds.), pp. 377–418, 1985.

K. Nitadori, *New approaches to high-performance N-body simulations with high-order integrator, new parallel algorithm, and efficient use of SIMD hardware*.
PhD thesis, University of Tokyo, 2009.

H. Nguyen, *Gpu gems 3*.
Addison-Wesley Professional, first ed., 2007.

H. C. Plummer, "On the problem of distribution in globular star clusters," vol. 71, pp. 460–470, Mar. 1911.

M. H. Hénon, "The Monte Carlo Method (Papers appear in the Proceedings of IAU Colloquium No. 10 Gravitational N-Body Problem (ed. by Myron Lecar), R. Reidel Publ. Co. , Dordrecht-Holland.)," vol. 14, pp. 151–167, Nov. 1971.

D. C. Heggie and R. D. Mathieu, "Standardised Units and Time Scales," in *The Use of Supercomputers in Stellar Dynamics* (P. Hut and S. L. W. McMillan, eds.), vol. 267 of *Lecture Notes in Physics, Berlin Springer Verlag*, p. 233, 1986.

S. Konstantinidis and K. D. Kokkotas, "MYRIAD: a new N-body code for simulations of star clusters," vol. 522, p. A70, Nov. 2010.

U. Löckmann, *Stellar Dynamics in the Vicinity of Super-massive Black Holes*. PhD thesis, Rheinischen Friedrich-Wilhelms-Universität Bonn, 2009.

A modular, GPU-based, direct-summation $N-$body integrator

Cristián Maureira
Dr. Pau Amaro-Seoane

Albert Einstein Institute

November 22, 2013