N-body simulations (on general- and special-purpose supercomputers)

Jun Makino
Kobe University/RIKEN AICS/TiTech ELSI/

Nov 24, 2017, AMNH

Overview

- **History of direct N-body simulations**
- **Future directions of N-body simulation methods**
- **Summary**

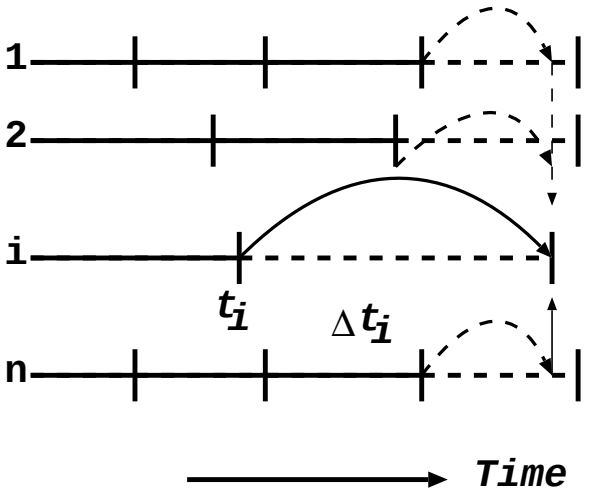
History of direct N-body simulations

- **Before 1986: Aarseth and scalar computers**
- **1986: blockstep scheme**
- **1992: 4th-order Hermite scheme**
- **1995, 2001: GRAPE-4 and GRAPE-6: without the neighbor scheme**
- **1999(?): NBODY6++ parallel neighbor scheme**
- **2006: Ninja scheme**
- **2011: $\mathrm{P^3T}$ scheme**
- **2012, 2015: NBODY6+GPU, NBODY6++ + GPU**

Before 1986: Aarseth and scalar computers

Individual timestep:

- **Each star has its own time t_i and timestep Δt_i.**
- **One with minimum $t_i + \Delta t_i$ is selected and updated, using the predicted position of other stars.**
- **Use 4-step, 4th-order predictor-corrector scheme with variable timestep.**



Neighbor scheme

- Separate total force on a star to two parts: neighbor and "regular" (the rest).
- Apply different timesteps to two parts.
- Change in the neighbor list need to be corrected at each regular step.

1986: blockstep scheme

THE VECTORIZATION OF SMALL-N INTEGRATORS

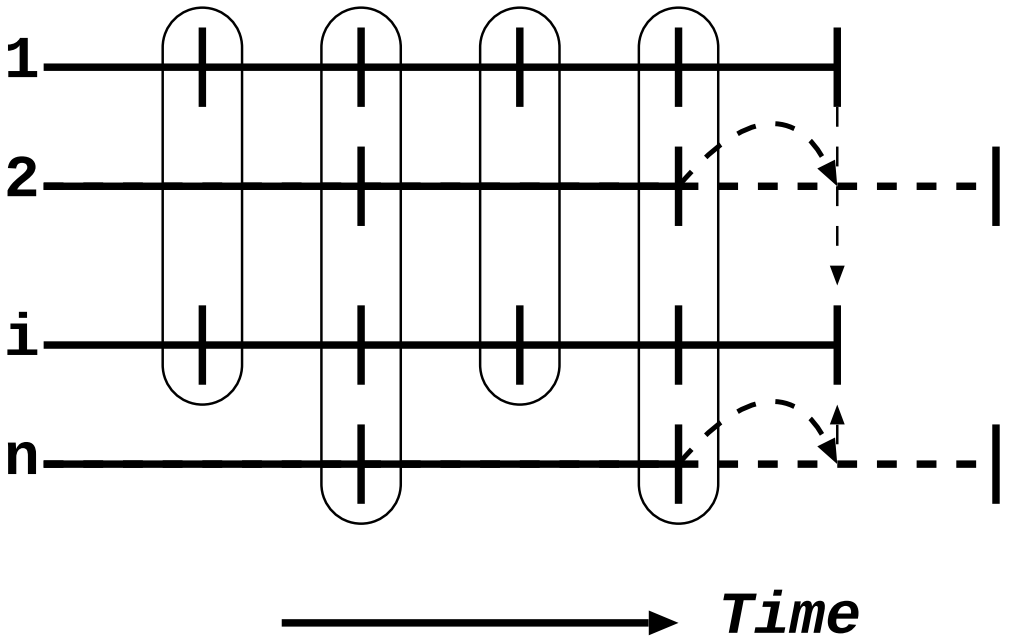
Stephen L. W. McMillan
Northwestern University
Department of Physics and Astronomy
Evanston, Illinois 60201

While it is very likely that supercomputers will greatly expand our understanding of the behavior of large self-gravitating systems, it is also probable that they will enable us to study exhaustively the dynamics of much smaller star clusters and associations. Through the simulation of large numbers of such systems (those containing less than, say, five hundred stars), one might hope eventually to put the small N-body problem on the same statistical footing as has already been achieved in the three- and four-body cases (Hills 1975, Hut and Bahcall 1983, Hut 1984, Mikkola 1984). Among the many issues that can be addressed are the timescales and modes of cluster dissolution, the formation and energy distribution of binary systems, and the detailed microphysics of individual stars' orbits and interactions.

In adapting existing codes for use on supercomputers, new, machine-dependent considerations inevitably arise. The current fastest machines derive much of their speed from their ability to operate in so-called "vector" mode, where the same operation is applied to successive elements of an array, at a rate much greater than would be attained if the calculation were done one element at a time (that is, in "scalar" mode). The details of the implementation of vector calculations vary widely from one machine to another, but, in general, the speed of an arithmetic operation can

1986: blockstep scheme

- "Quantize" the stepsize to powers of two. Update the stars with the same time in parallel.
- Much better parallel efficiency
- Reduction of the calculation cost of the predictor of other stars.

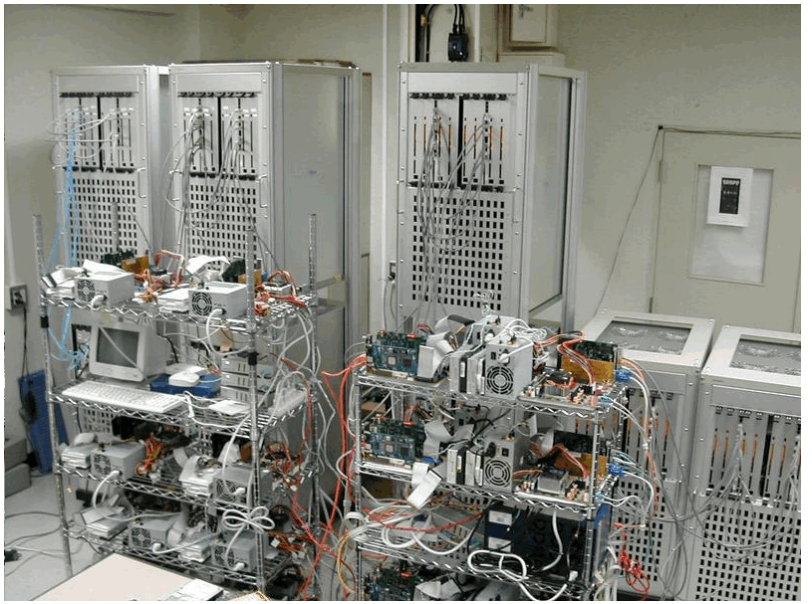


1992: 4th-order Hermite scheme

(JM and Aarseth 1992)

- Calculate "jerk" (first time derivative of acceleration) in addition to acceleration.
- Use two-point Hermite interpolation to construct 4th-order integrator. ("single step" scheme)
- Works with neighbor scheme as well.
- Several advantages
  - Easier to write
  - Longer timestep: Better efficiency on parallel/vector/special-purpose machines

1995, 2001: GRAPE-4 and GRAPE-6



- Both rely on blockstep + Hermite scheme
- We did not implement the neighbor scheme: to minimize the work of the host CPU

1999: NBODY6++ parallel neighbor scheme

(Spurzem 1999)

- Each process keeps the complete copy of the system
- After the list of stars to be integrated in the current step is determined, each process determine its share to update.
- Updated results are exchanged between processes.
- Relies on blockstep, Hermite and neighbor scheme.
- Works great for moderate number of processes.

2006: Ninja scheme

(Nitadori+ 2006)

- Parallelize in two dimensions: Force calculation on one particle is parallelized as well.
- Communication is minimized.
- Scales to thousands of cores for $N \sim 10^5$ or
- No one tried to combine this method with neighbor scheme yet...

2011: P^3T scheme

(Oshino+ 2011, Iwasawa+ 2016)

- Latest (as far as I know) approach to use treecode to integrate collisional systems.
- Previous efforts: Jernigan and Porter 1989, McMillan and Aarseth 1993, (Richardson 1993).
- Previous works tried to combine tree and individual timestep
- We gave up, and split gravitational interaction into two terms using distance-dependent switching function (same as Mercury integrator for planetary dynamics)
- Apply treecode + leapfrog to long-range term, and Hermite to short-range term.
- Seems to be quite promising.

P^3T scheme

Good news:

- Calculation cost is $O(N \log N)$ per crossing time. Thus, one N removed.
- Parallelization method has been well known and is highly efficient, on a variety of architectures
- This is because you need to optimize (mostly...) tree part only

Not so good news:

- No production-quality code (with regularization, stellar evolution, etc) yet. (I believe Long will present the state of the arts)

2012, 2015: NBODY6+GPU, NBODY6++ + GPU

(Nitadori and Aarseth 2012, Wang 2015)

- **Use GPU to the "regular" force.**
- **At present the fastest code available.**
- **The million-body problem has finally become feasible.**

Roughly speaking

Year	N	Method	Architecture
1960	30	Ind. Δt	Scalar
1970	100	Ind. Δt	Scalar
1980	300	Neighbor	Scalar
1990	3000	Neighbor	Vector
2000	30000	Ind. Δt	GRAPE
2010	100000	Neighbor	GPU
2020	3000000?	P^3T?	?

Architecture changes every 10 years.

Trends in Top500

Again roughly speaking...

Year	Speed	Arch.	Example
1990	3GF	Vector	Cray YMP
2000	5TF	Scalar MPU	ASCI Red
2010	3PF	GPU	Tianhe-1A
2020	1EF?	?	?

Architecture changes every 10 years.

Speed and Number of particles

If we compare 1990 and 2010

- **Computers became 10^6 times faster**
- **N increased only by 30 or so.**

The performance of current star cluster codes do not scale well on large HPC machines.

Designs which could reach Exaflops in 2020

- **TaihuLight (SW26010): currently Top 1.**
- **GYOUKOU (PEZY-SC2): New Top 4.**
- **Tianhe-3/4? (Chinese Accelerator?): Previous Top 1.**

GPU-like accelerator (Tianhe) or Heterogeneous many-core (other two).

N-body simulation in 2020 and beyond

- We do need combination of tree and individual timestep.
- Currently, P^3T is the only algorithm which might work.

Theoretical calculation cost scaling for P^{3}T

When core is large

- per crossing time: $N^{4/3} \log N$ ($N^{1/3}$ from timestep)
- per relaxation time $N^{7/3}$

When core is small: We need to make $O(N)$ hard binaries with triple interactions or binary-single star encounters

- Each hard binary requires constant cost, but with P^{3}T this cost might be $N \log N$
- Total cost would be $N^2 \log N$

This means that even if we reduce the global tree step down to core crossing timescale, calculation cost is still N^2.

Simulation turn-around time

When core is large:

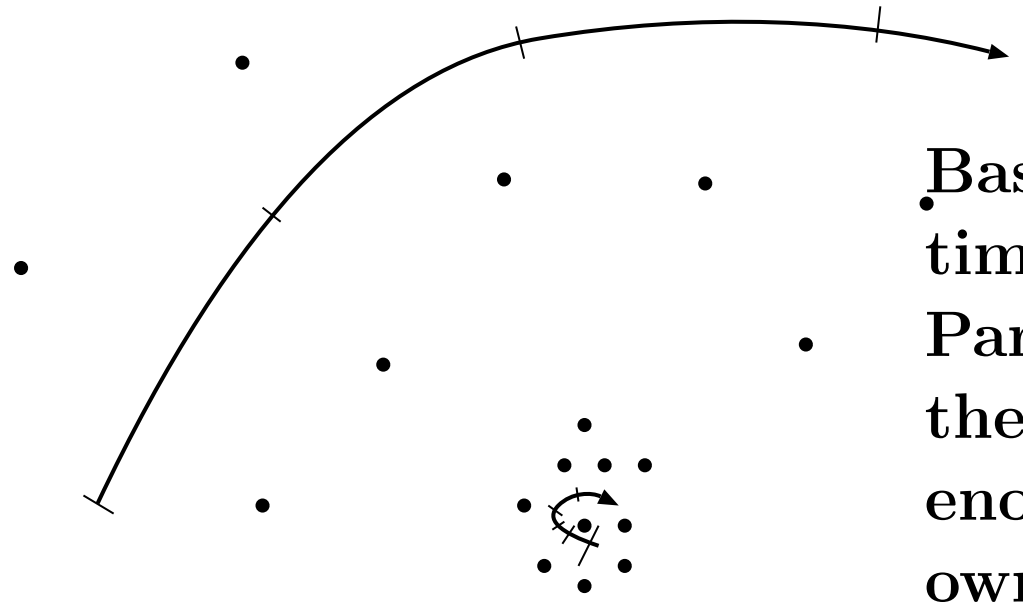
- **For N crossing times, we need around $N^{4/3}$ steps. For $N = 10^7$, 10^9 steps.**
- **If we can make one timestep 1 msec (currently difficult... 10ms is doable), we can finish one run in 1 week or so.**

Small core scaling is still difficult to predict...

Summary

- In the last three decades we have seen quite significant improvement on what we can do with direct N-body simulations.
- Currently, NBODY6(++) + GPU works great.
- To use even larger number of particle P^3T scheme with parallel treecode (on accelerators) might be necessary.
- Need cleaner treatment of small subsystems (not discussed much today)
- Long has been working on this. Stay tuned.

Problem of individual timestep/neighbor scheme



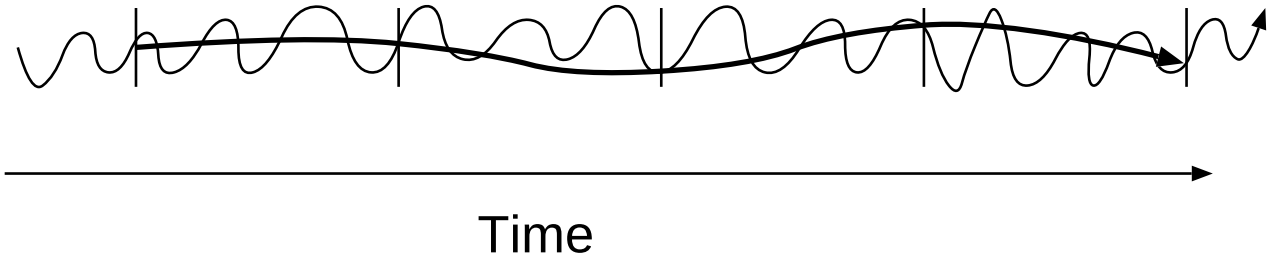
Basic idea of individual timestep:
Particles should have the timestep just enough to resolve their own orbits.

What happens to the forces from short-timescale particles to long-timescale particles?

Slide recycled from the Capri meeting

What's happening

They are integrated in a completely wrong way!

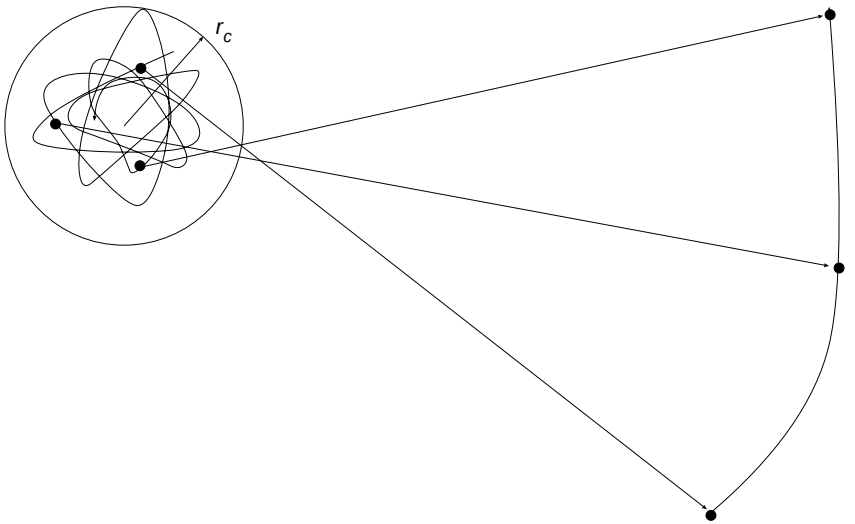


- **Forces do have rapidly changing components**
- **If the timestep is large, forces are sampled "randomly" (if the orbit is not periodic)**
- **Much more problematic with the Hermite scheme**
- **Even more problematic with the neighbor scheme**

Slide recycled from the Capri meeting

When does this happen?

- **When the orbital timescale of particles in the core becomes less than the timestep of typical particles in the cluster.**
- **Roughly speaking: If $r_c \ll r_h N^{-1/3}$**
- **Just before bounce: $r_c \sim r_h/N \ll r_h N^{-1/3}$**



Slide recycled from the Capri meeting

Does this really matter?

- This error is actually visible: The reason why the energy error increases toward the core collapse.
- Reduction of timestep helps, but only as $\Delta t^{1.5}$
- The only way to suppress this error completely is to reduce the timesteps of all particles to less than the core crossing time
- Can we just let the error grow? No. With poor energy conservation we cannot say for sure that the calculation is "correct".

Impact on the calculation cost

- Hopefully not so severe for normal star clusters
  - the fraction of time for which the core size is small is small
  - Mass spectrum makes the core size larger
- Any system with central massive BH might be problematic.

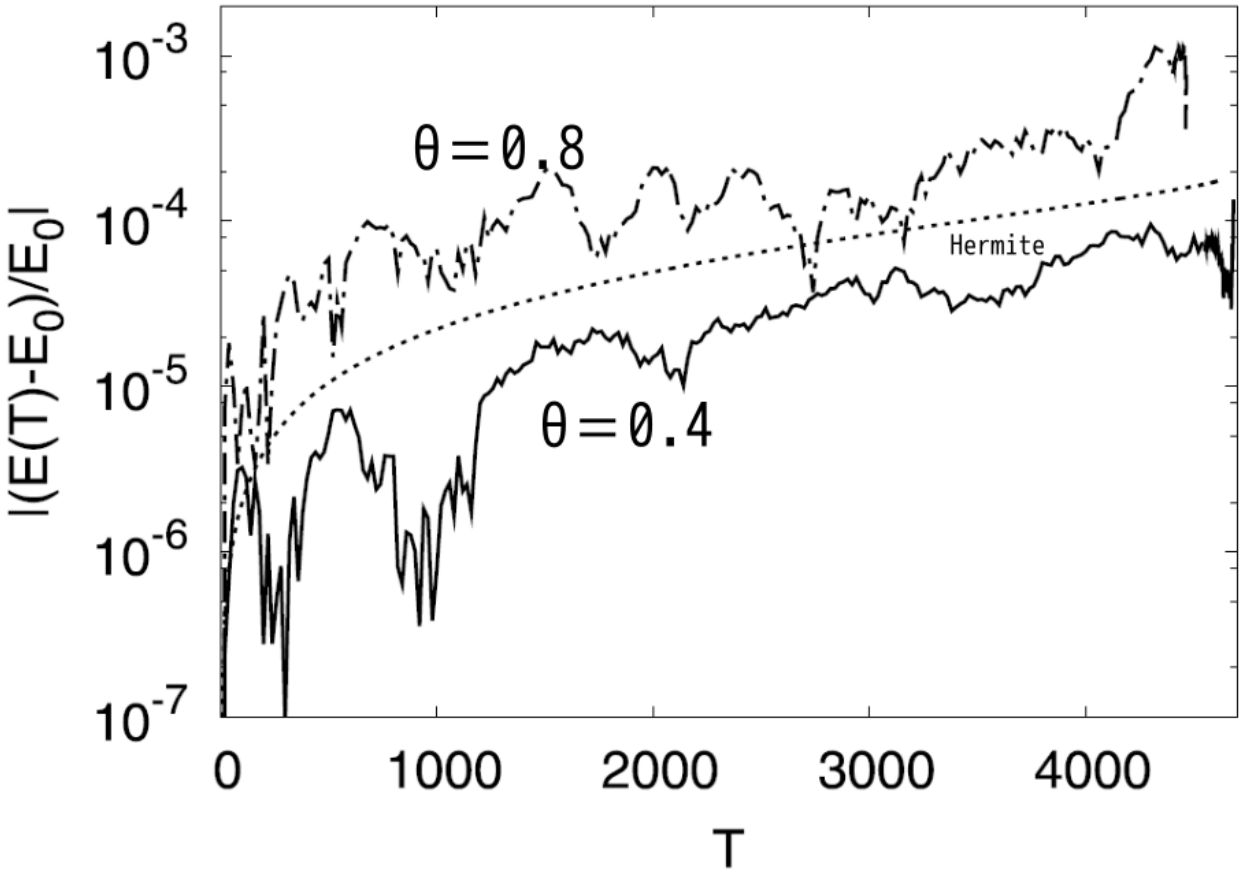
Slide recycled from the Capri meeting

"Solution"

- With P^3T scheme, it is not unpractical to integrate all stars with the timestep smaller than the core crossing time
- Almost all calculation cost is spent by the simple shared-timestep treecode
- Parallelization and the use of accelerators are pretty efficient (can be done with our FDPS package)
- Currently we are working on this...

Some good news

P^3T scheme can actually conserve energy better than Hermite (Iwasawa+ 2016)



$N = 16$k, down to core collapse.

Theoretical calculation cost scaling for P^{3}T

When core is large

- per crossing time: $N^{4/3} \log N$ ($N^{1/3}$ from timestep
- per relaxation time $N^{7/3}$

When core is small: We need to make $O(N)$ hard binaries with triple interactions or binary-single star encounters

- Each hard binary requires constant cost, but with P^{3}T this cost might be $N \log N$
- Total cost would be $N^2 \log N$

This means that even if we reduce the global tree step down to core crossing timescale, calculation cost is still N^2.

Simulation turn-around time

When core is large:

- **For N crossing times, we need around $N^{4/3}$ steps. For $N = 10^7$, 10^9 steps.**
- **If we can make one timestep 1 msec (currently difficult... 10ms is doable), we can finish one run in 1 week or so.**

Small core scaling is still difficult to predict...

FDPS

Iwasawa+2016

- Please visit: https://github.com/FDPS/FDPS
- A Framework for Developing parallel Particle Simulation code
- FDPS offers library functions for domain decomposition, particle exchange, interaction calculation using tree.
- Can be used to implement pure Nbody, SPH, or any particle simulations with two-body interactions.
- Use essentially the same algorithm as used in our treecode implementation on K computer (GreeM, Ishiyama, Nitadori and JM 2012).
- Runs efficiently on K, Xeon clusters or GPU clusters or even on TaihuLight