GRAPE-DR, GRAPE-8, and ...

Jun Makino
Center for Computational Astrophysics
and
Division Theoretical Astronomy
National Astronomical Observatory of Japan

Dec 14, 2010 CPS, Kobe

Talk structure

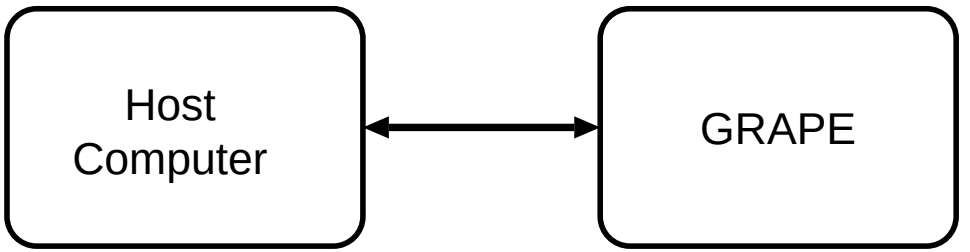
- GRAPE
- GRAPE-DR
- GRAPE-8
- What's next?

Short history of GRAPE

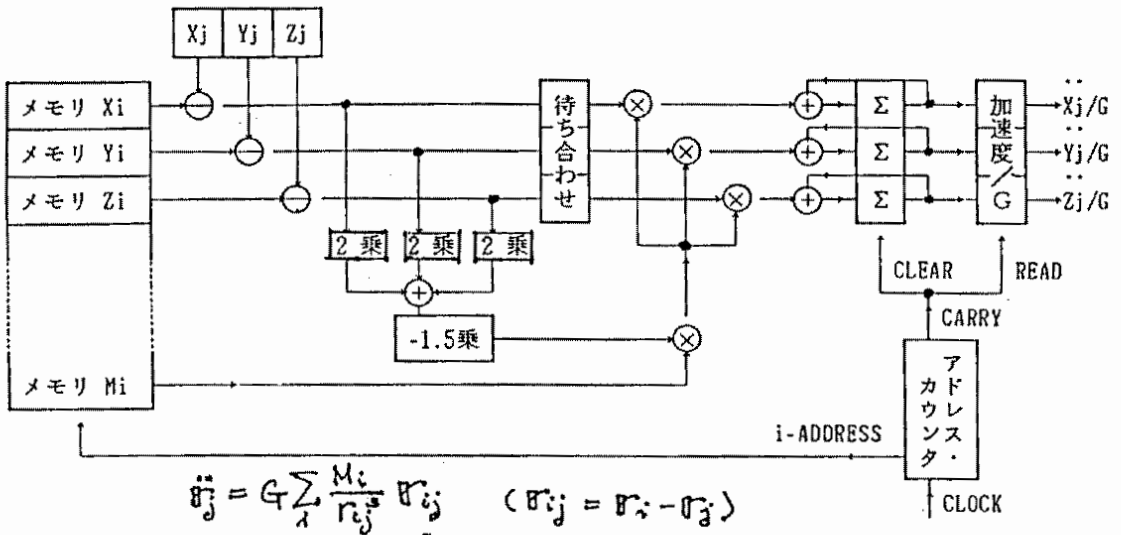
- Basic concept
- GRAPE-1 through 6
- Software Perspective

Basic concept (As of 1988)

- **With N-body simulation, almost all calculation goes to the calculation of particle-particle interaction.**
- **This is true even for schemes like Barnes-Hut treecode or FMM.**
- **A simple hardware which calculates the particle-particle interaction can accelerate overall calculation.**
- **Original Idea: Chikada (1988)**



Chikada's idea (1988)



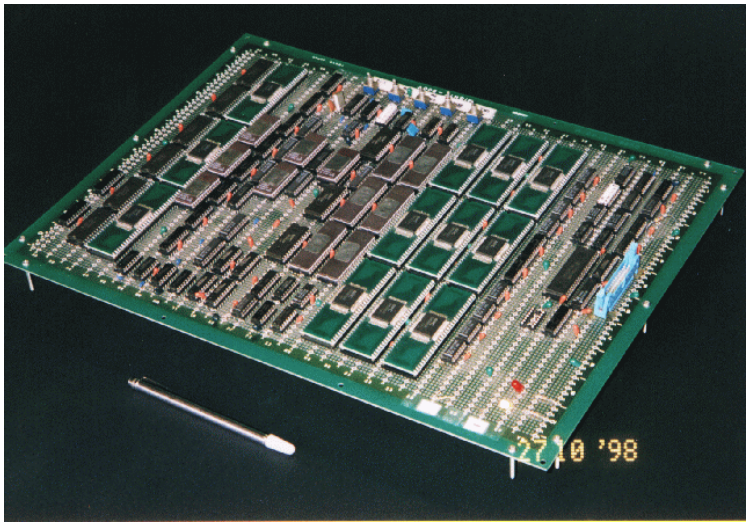
$$\ddot{\vec{r}}_j = G\sum_i \frac{M_i}{r_{ij}^3}\vec{r}_{ij} \qquad (\vec{r}_{ij} = \vec{r}_i - \vec{r}_j)$$

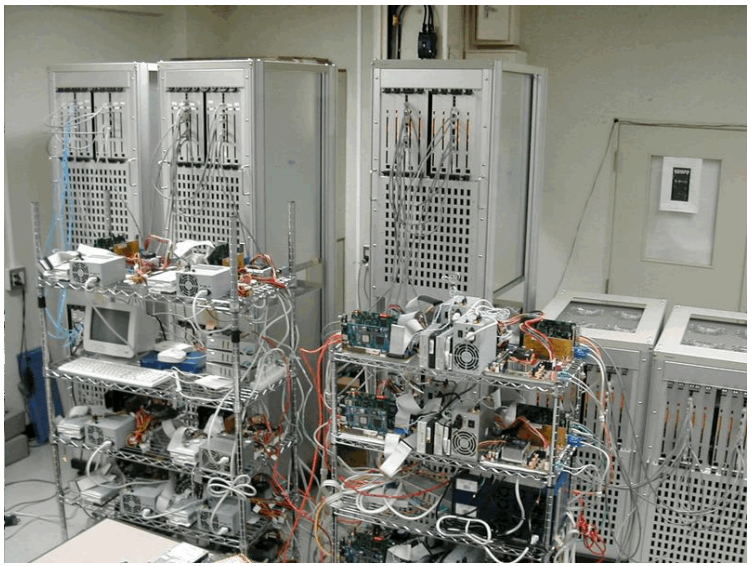
+, −, ×, 2乗は1 operation, -1.5乗は多項式近似でやるとして10operation 位に相当する.
総計24operation.
各operation の後にはレジスタがあって, 全体がpipelineになっているものとする.
「待ち合わせ」は2乗してMと掛け算する間の時間ズレを補正するためのFIFO(First-In First-Out memory).
「Σ」は足し込み用のレジスタ. N回足した後結果を右のレジスタに転送する.

図2. N体問題のj-体に働く重力加速度を計算する回路の概念図.

- Hardwired pipeline for force calculation (similar to Delft DMDP)
- Hybrid Architecture (things other than force calculation done elsewhere)

GRAPE-1 to GRAPE-6



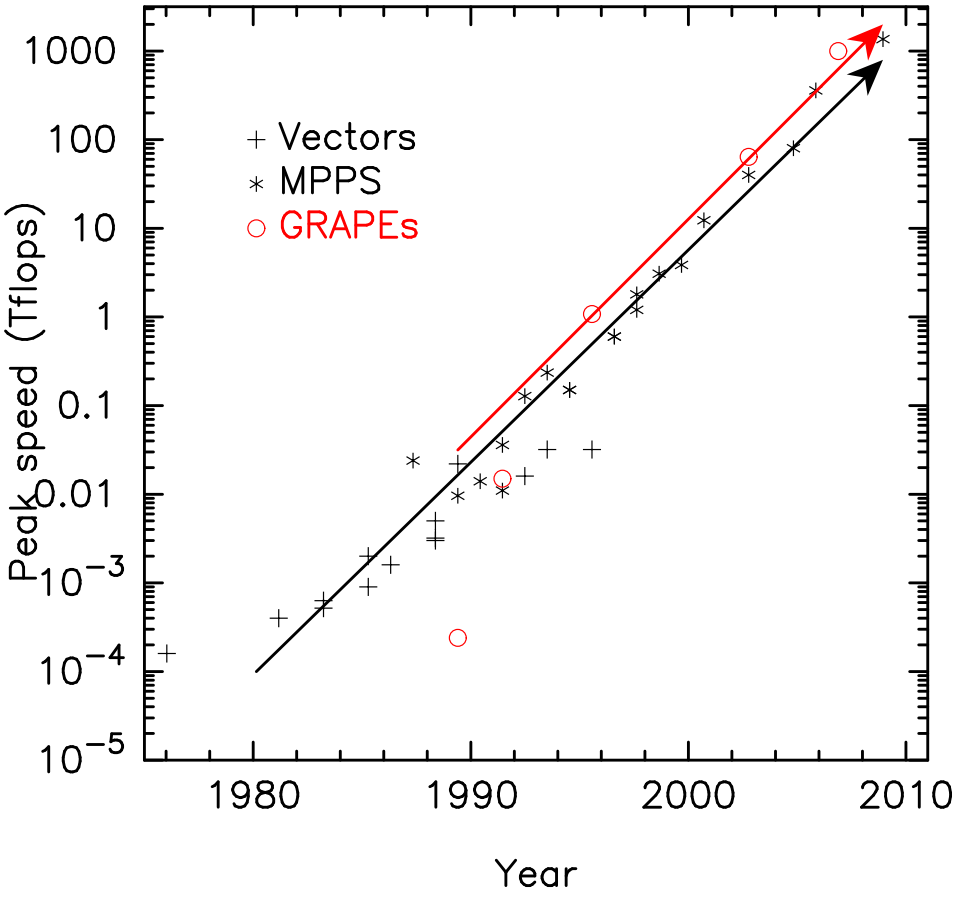


GRAPE-1: 1989, 308Mflops

GRAPE-4: 1995, 1.08Tflops

GRAPE-6: 2002, 64Tflops

Performance history



Since 1995 (GRAPE-4), GRAPE has been faster than general-purpose computers.

Development cost was around 1/100.

Software development for GRAPE

GRAPE software library provides several basic functions to use GRAPE hardware.

- **Sends particles to GRAPE board memory**
- **Sends positions to calculate the force and start calculation**
- **get the calculated force (asynchronous)**

User application programs use these functions. Algorithm modifications are necessary to reduce communication and increase the degree of parallelism

Analogy to BLAS

Level	BLAS	Calc:Comm	Gravity	Calc:Comm
0	`c=c-a*s`	1:1	$f_{ij} = f(x_i, x_j)$	1:1
1	AXPY	$N : N$	$f_i = \Sigma_j f(x_i, x_j)$	$N : N$
2	GEMV	$N^2 : N^2$	$f_i = \Sigma_j f(x_i, x_j)$ for multiple i	$N^2 : N$
3	GEMM	$N^3 : N^2$	$f_{k,i} = \Sigma_j f(x_{k,i}, x_{k,j})$ "Multiwalk"	$N^2 : N$

- Calc $\gg$Comm essential for accelerator
- Level-3 (matrix-matrix) essential for BLAS
- Level-2 like (vector-vector) enough for gravity
- Treecode and/or short-range force might need Level-3 like API.

Porting issues

- Libraries for GRAPE-4 and 6 (for example) are not compatible
- Even so, porting was not so hard. The calls to GRAPE libraries are limited to a fairly small number of places in application codes.
- Backporting the GRAPE-oriented code to CPU-only code is easy, and allows very efficient use of SIMD features.
- In principle the same for GPGPU or other accelerators.

Real-World issues with “Porting”

— Mostly on GPGPU....

- Getting something run on GPU is not difficult
- Getting a good performance number compared with non-optimized, single-core x86 performance is not so hard.

Real-World issues with "Porting" continued

- **Making it faster than 10-year-old GRAPE or highly-optimized code on x86 (using SSE/SSE2) is *VERY, VERY HARD* (you need Keigo)**
- **These are *mostly* software issues**
- **Some of the most serious ones are limitations in the architecture (lack of good reduction operation over processors etc)**

Quotes

From: Twelve Ways to Fool the Masses When Giving Performance Results on ~~Accelerators~~ Parallel Computers (D. H. Bailey, 1991)

1. Quote only 32-bit performance results, not 64-bit results.
2. Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application.
6. Compare your results against scalar, unoptimized code on ~~Xeons~~ Crays.
7. When direct run time comparisons are required, compare with an old code on an obsolete system.
8. If MFLOPS rates must be quoted, base the operation count on the parallel implementation, not on the best sequential implementation.
12. If all else fails, show pretty pictures and animated videos, and don't talk about performance.

History repeats itself — Karl Marx

"Problem" with GRAPE approach

- Chip development cost has become too high.

Year	Machine	Chip initial cost	process
1992	GRAPE-4	200K$	1μm
1997	GRAPE-6	1M$	250nm
2004	GRAPE-DR	4M$	90nm
2010?	GDR2?	$>$ 10M$	45nm?

Initial cost should be 1/4 or less of the total budget.
How we can continue?
(Riken "K" costs >1B$...)

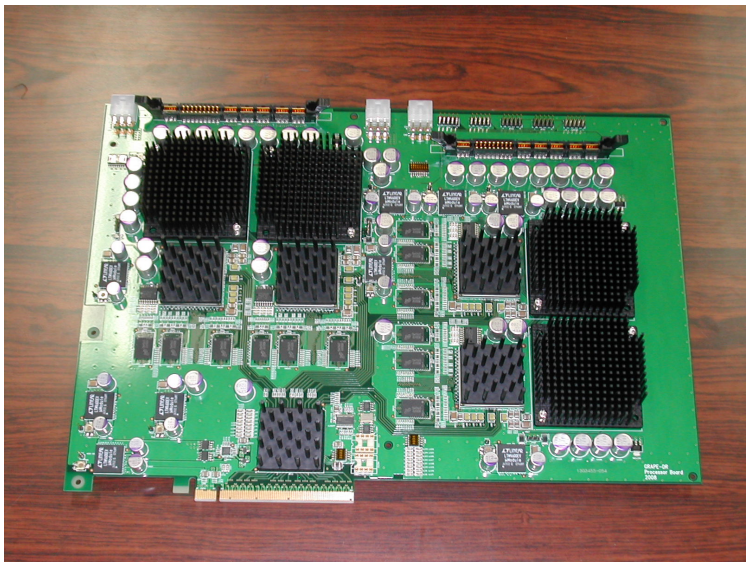
Current Generation— GRAPE-DR

- New architecture — wider application range than previous GRAPEs
- primarily to get funded
- No force pipeline. SIMD programmable processor
- "Parallel evolution" with GPUs.
- Development: FY 2004-2008

The Chip

Sample chip delivered May 2006
90nm TSMC, Worst case 65W@500MHz

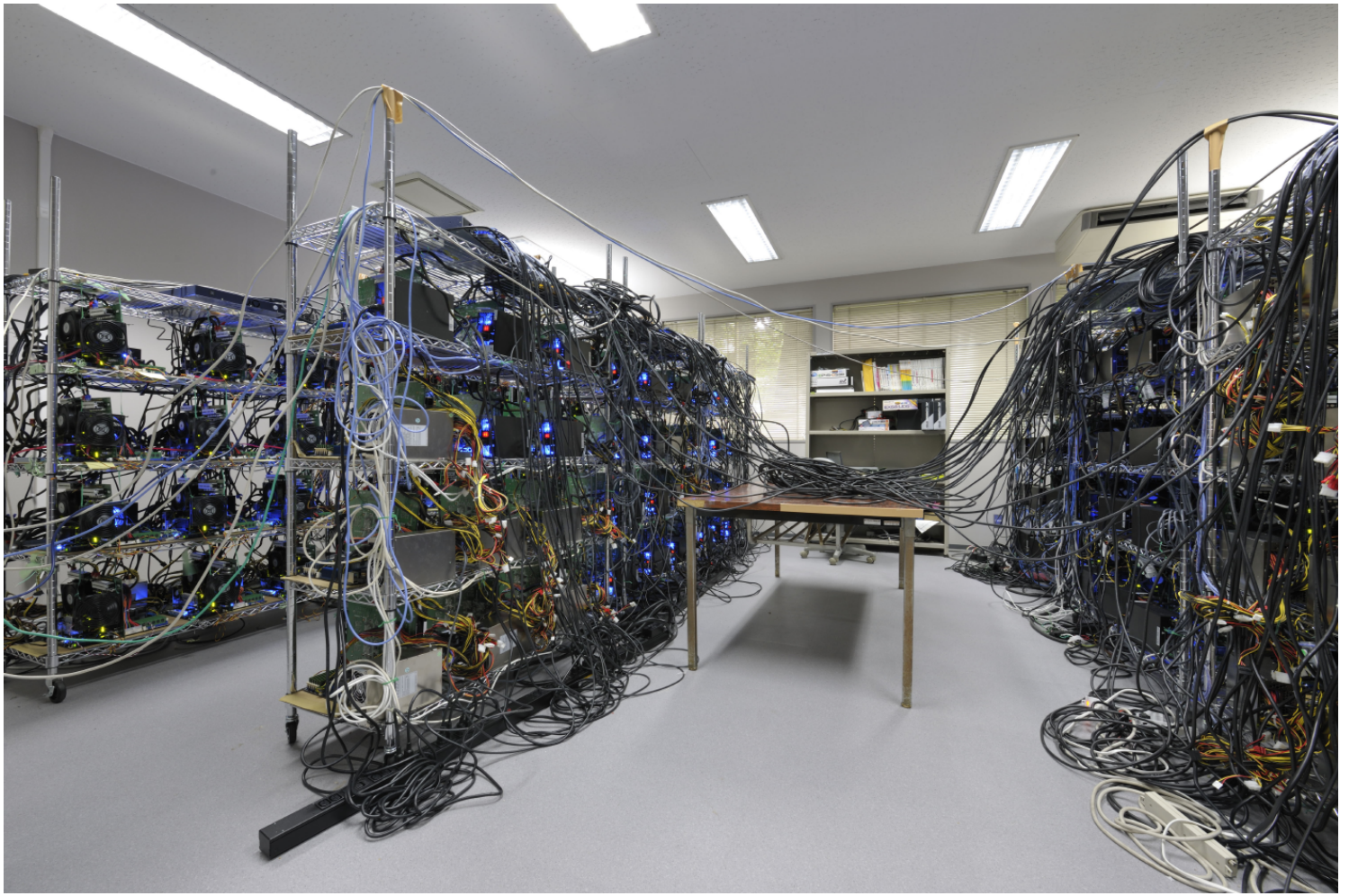
Processor board



PCIe x16 (Gen 1) interface
Altera Arria GX as DRAM controller/communication interface

- Around 200W power consumption
- Not quite running at 500MHz yet... (FPGA design not optimized yet)
- 819Gflops DP peak (400MHz clock)
- Available from K&F Computing Research (www.kfcr.jp)

GRAPE-DR cluster system



OpenMP-like compiler

Goose compiler (Kawai 2009)

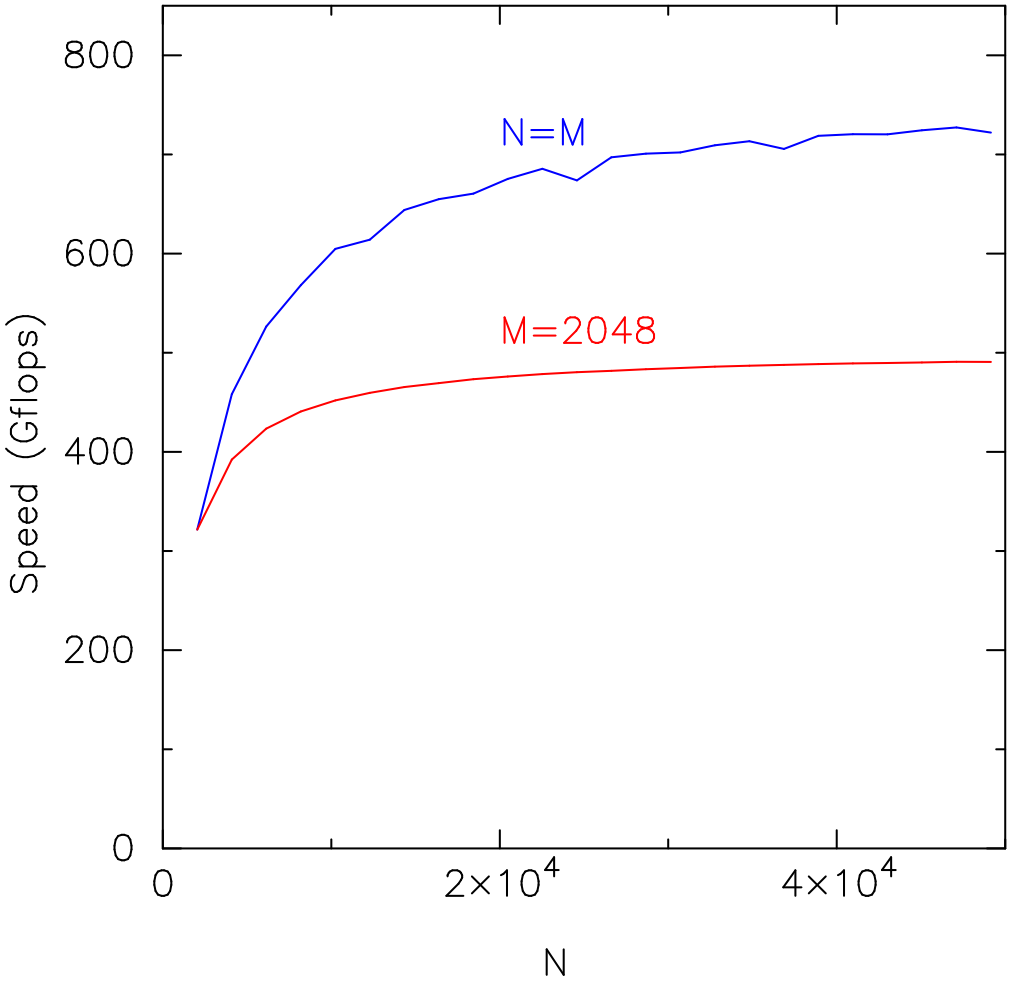
```
#pragma goose parallel for icnt(i) jcnt(j) res (a[i][0..2])
    for (i = 0; i < ni; i++) {
        for (j = 0; j < nj; j++) {
            double r2 = eps2[i];
            for (k = 0; k < 3; k++) dx[k] = x[j][k] - x[i][k];
            for (k = 0; k < 3; k++) r2 += dx[k]*dx[k];
            rinv = rsqrt(r2);
            mf = m[j]*rinv*rinv*rinv;
            for (k = 0; k < 3; k++)  a[i][k] += mf * dx[k];
        }
    }
```

Generates code for single- and double-loops
(Translates to Nakasato's language)

Performance and Tuning example

- **HPL (LU-decomposition)**
- **Gravity**

Matrix-multiplication performance



M=N, K=2048
730 Gflops
(Asymptotic: 770 Gflops)

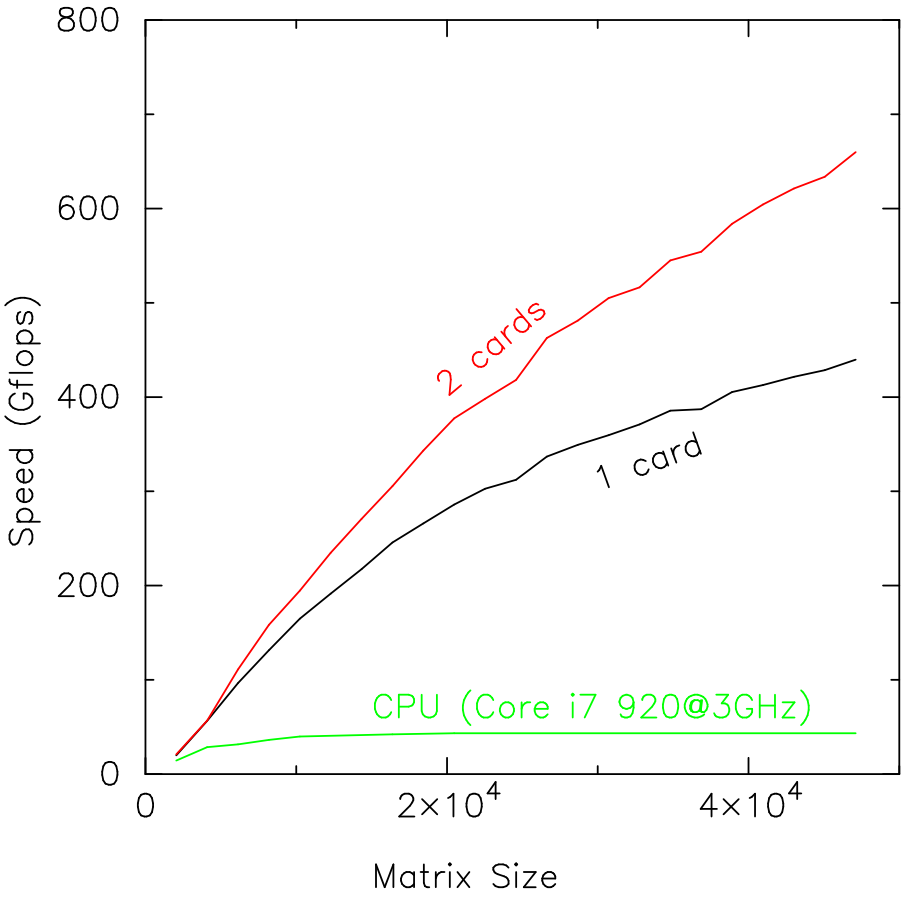
N=K=2048
490 Gflops

94% of theoretical peak

Fermi: 300Gflops, 60%

HD5870: 470Gflops, 87%

LU-decomposition performance



Speed in Gflops as function of Matrix size
A complete rewrite of HPL
430 Gflops (54% of theoretical peak) for N=50K
2x faster than HPL 1.04a for small N

LU-decomposition tuning

- Almost every know technique
  - except for the concurrent use of CPU and GDR (we use GDR for column factorization as well...)
  - right-looking form
  - TRSM converted to GEMM
  - use row-major order for fast $O(N^2)$ operations
- Several other "new" techniques
  - Transpose matrix during recursive column decomposition
  - Use recursive scheme for TRSM (calculation of L^{-1})

Little Green 500, June 2010

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	815.43	National Astronomical Observatory of Japan	GRAPE-DR accelerator Cluster, Infiniband	28.67
2	773.38	Forschungszentrum Juelich (FZJ)	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
2	773.38	Universitaet Regensburg	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
2	773.38	Universitaet Wuppertal	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
5	536.24	Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw	BladeCenter QS22 Cluster, PowerXCell 8i 4.0 Ghz, Infiniband	34.63

#1: GRAPE-DR, #2: QPACE: German QCD machine
#9: NVIDIA Fermi

HPL (parallel LU)

- **Everything done for single-node LU-decomposition**
- **Both column- and row-wise communication hidden**
- **TRSM further modified: calculate UT^{-1} instead of $T^{-1}U$**
- **More or less working, still lots of room for tuning**

Green 500, Nov 2010

Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)
1	1684.20	IBM Thomas J. Watson Research Center	NNSA/SC Blue Gene/Q Prototype	38.80
2	958.35	GSIC Center, Tokyo Institute of Technology	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows	1243.80
3	933.06	NCSA	Hybrid Cluster Core i3 2.93Ghz Dual Core, NVIDIA C2050, Infiniband	36.00
4	828.67	RIKEN Advanced Institute for Computational Science	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect	57.96
5	773.38	Forschungszentrum Juelich (FZJ)	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
5	773.38	Universitaet Regensburg	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
5	773.38	Universitaet Wuppertal	QPACE SFB TR Cluster, PowerXCell 8i, 3.2 GHz, 3D-Torus	57.54
8	740.78	Universitaet Frankfurt	Supermicro Cluster, QC Opteron 2.1 GHz, ATI Radeon GPU, Infiniband	385.00
9	677.12	Georgia Institute of Technology	HP ProLiant SL390s G7 Xeon 6C X5660 2.8Ghz, nVidia Fermi, Infiniband QDR	94.40
10	636.36	National Institute for Environmental Studies	GOSAT Research Computation Facility, nvidia	117.15

??? Where is GRAPE-DR ???

Green 500, Nov 2010

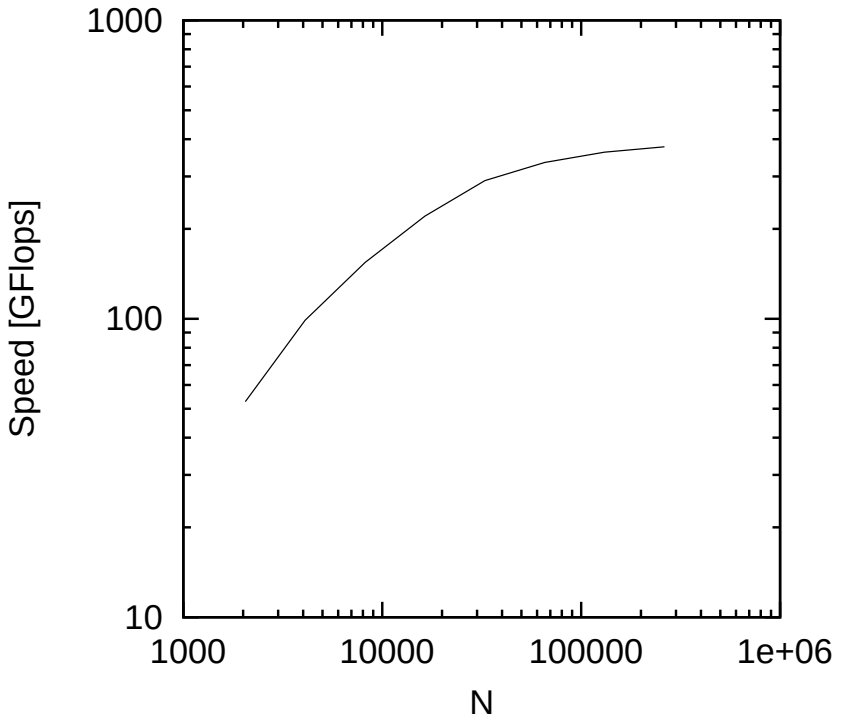
- The performance we submitted: 1474 Gflops/W
- It should be #2 in the list
- Somehow we were not listed
- Little Green 500 list is not even released yet
- Green 500 people are "working" on this (according to them)

How we achieved 80% improvement?

- Bug fix and performance improvement of software
- Replacement of power supply units (80 PLUS Gold: $\sim 90\%$ efficiency)
- Reduction of V_{core} of Intel CPU ($1.2 \rightarrow 1.05$)
- Many other small changes...

Gravity kernel performance

(Performance of individual timestep code not much different)



Assembly code (which I wrote) is not very optimized yet... Should reach at least 600 Gflops after rewrite.

GRAPE-8(9?)

Question:
Any reason to continue hardware development?

- **GPUs are fast, and getting faster**
- **FPGAs are also growing in size and speed**
- **Custom ASICs practically impossible to make**

GRAPE-8(9?)

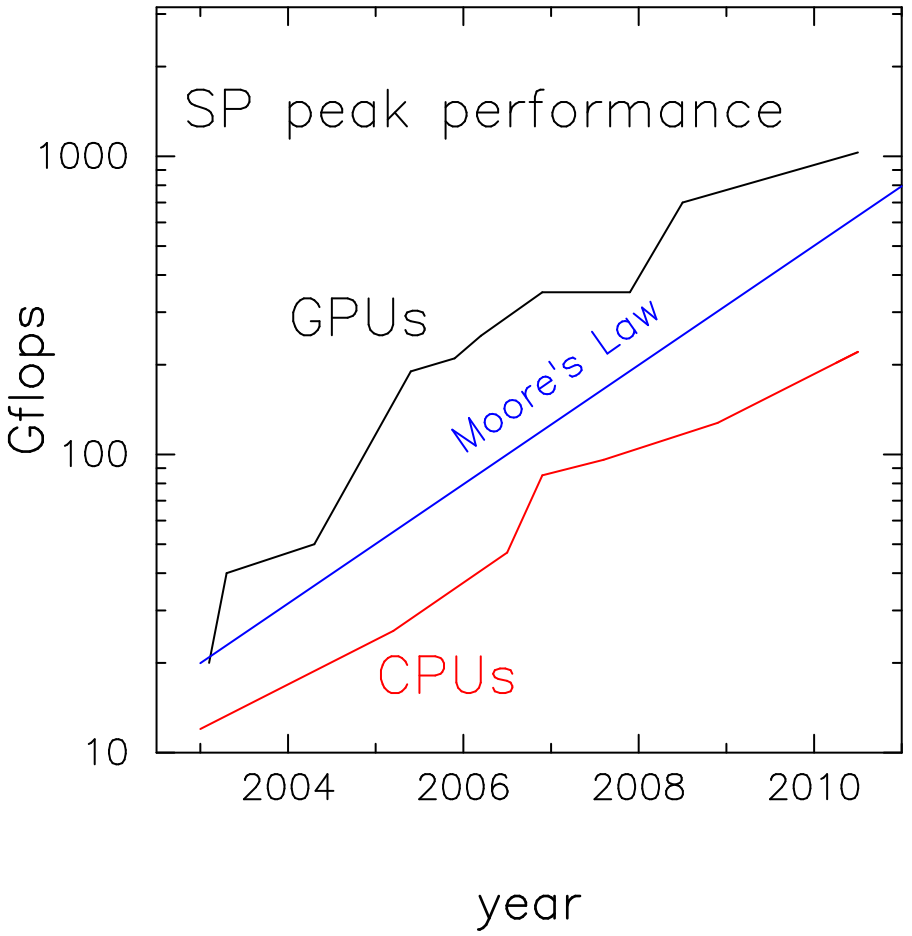
Question:
Any reason to continue hardware development?

- **GPUs are fast, and getting faster**
- **FPGAs are also growing in size and speed**
- **Custom ASICs practically impossible to make**

Answer?

- **GPU speed improvement might have slowed down**
- **FPGAs are becoming far too expensive**
- **Power consumption might become most critical**
- **Somewhat cheaper way to make custom chips**

(NVIDIA) GPU speed improvement slowing down?



Clear "slowing down" after 2006 (after G80)

Reason: shift to more general-purpose architecture

Discrete GPU market is eaten up by unified chipsets and unified CPU+GPU

But: HPC market is not large enough to support complex chip development

FPGA

"Field Programmable Gate Array"

- **"Programmable" hardware**
- **"Future of computing" for the last two decades....**
- **Telecommunication market needs: large and fast chips (very expensive)**

Structured ASIC

- **Something between FPGA and ASIC**
- **eASIC: 90nm (Fujitsu) and 45nm (Global Foundries) products.**
- **Compared to FPGA:**
  - **3x size**
  - **1/10 chip unit price**
  - **non-zero initial cost**
- **Compared to ASIC:**
  - **1/10 size and 1/2 clock speed**
  - **1/3 chip unit price**
  - **1/100 initial cost ($>$ 10M USD vs $\sim$ 100K)**

GRAPE with eASIC

- **Design finalized with eASIC Nextreme-2 chip.**
- **$\sim$ 50 pipelines ($\sim$ 500Gflops) per chip.**
- **10W/chip.**
- **50Gflops/W. (The number for total system is lower...)**

Will this be competitive?

Rule of thumb for a special-purpose computer project:

> **Price-performance goal should be more than 100 times better than that of a PC available when you start the project.**

— x 10 for 5 year development time
— x 10 for 5 year lifetime

Compared to CPU: Okay
Compared to GPU: ??? (Okay for electricity)

Will this be competitive?

Rule of thumb for a special-purpose computer project:

> **Price-performance goal should be more than 100 times better than that of a PC available when you start the project.**

— x 10 for 5 year development time
— x 10 for 5 year lifetime

Compared to CPU: Okay
Compared to GPU: ??? (Okay for electricity)

Will GPUs exist 10 years from now?

Future of HPC

Major problem: Application range of big systems have become narrow and will be even narrower

What we want to do: long-term integration.

- **10^4 years for star formation (10^7 dynamical time)**
- **10^7 years for planetary formation (10^7 dynamical time)**
- **10^{10} years for star cluster (10^6 dynamical time)**

What big machines (such as "K") let you do:
Simulation of $O(1)$ dynamical time for very large systems.

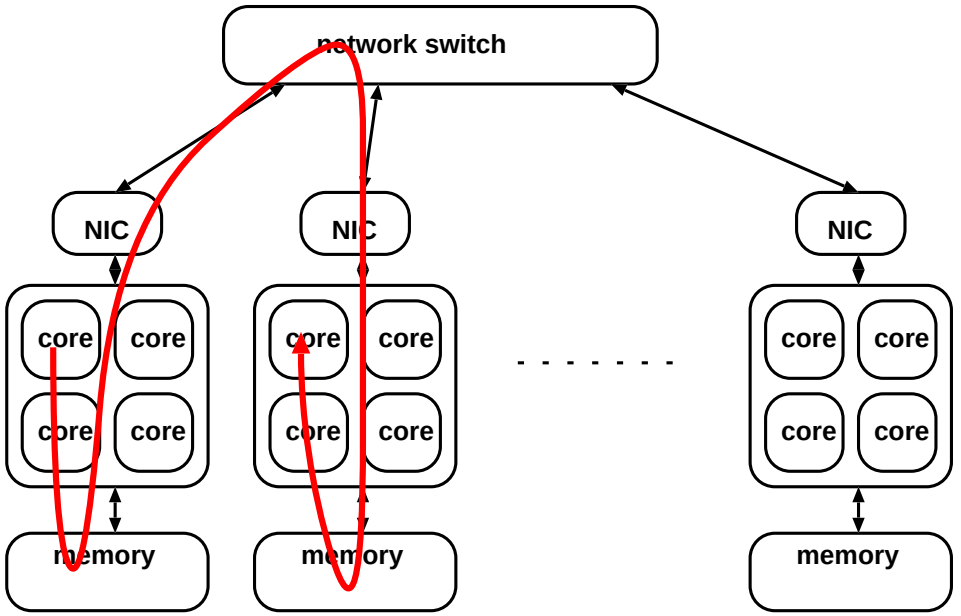
Reasons (common wisdom):

- **Performance does not scale beyond 10^4 cores, how we can use 10^6 cores?**
- **How we write programs for $> 10^6$ cores with distributed memory?**

Real issue

Communication latency: $\sim 1\mu$s or larger, 1000s of clock cycles

1. between the cores on a chip
2. between the cores and memory
3. between two nodes



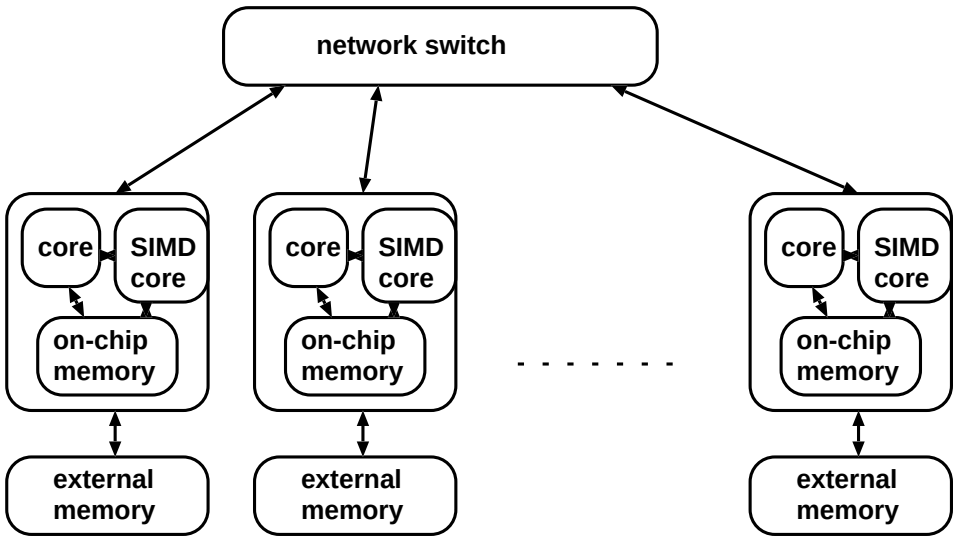
Communication latency is large simply because there are soooo many steps.

core-memory-cpu-nic-switch-nic-cpu-memory-core

What should we do?

Unfortunately, accelerator approach might not be enough.

- **integrate core, SIMD core, on-chip memory and NIC to one chip**
- **reduce communication latency to adjacent node to < 10 ns**



SIMD core: reduce the visible number of cores

On-chip memory: eliminate DRAM access overhead for small dataset

Who would make such a system?

Summary

- GRAPEs, special-purpose computer for gravitational N-body system, have been providing 10x - 100x more computational power compared to general-purpose supercomputers.
- GRAPE-DR, with programmable processors, has wider application range than traditional GRAPEs.
- DGEMM performance 730 Gflops,
  LU decomposition $>$ 450Gflops
- Achieved the best performance per W (Top 1 in June 2010 Little Green 500 list, 815Mflops/W)
- GRAPE-8(9?) will be "traditional" one, but for new algorithms
- "general purpose" HPC architecture should change soon....

Tree-Direct hybrid

BRIDGE Hamiltonian (Fujii et al 2007)

$$H = H_\alpha + H_\beta,$$

$$H_\alpha = -\sum_{i<j}^{N_G} \frac{G m_{G,i} m_{G,j}}{r_{GG,ij}} - \sum_{i=1}^{N_G} \sum_{j=1}^{N_{SC}} \frac{G m_{G,i} m_{C,j}}{r_{GS,ij}},$$

$$H_\beta = \sum_{i=1}^{N_G} \frac{p_{G,i}^2}{2m_{G,i}} + \sum_{i=1}^{N_{SC}} \frac{p_{C,i}^2}{2m_{C,i}} - \sum_{i<j}^{N_{SC}} \frac{G m_{C,i} m_{C,j}}{r_{CC,ij}},$$

Separate internal motion (or potential) of star cluster from parent galaxy (and interaction with it)

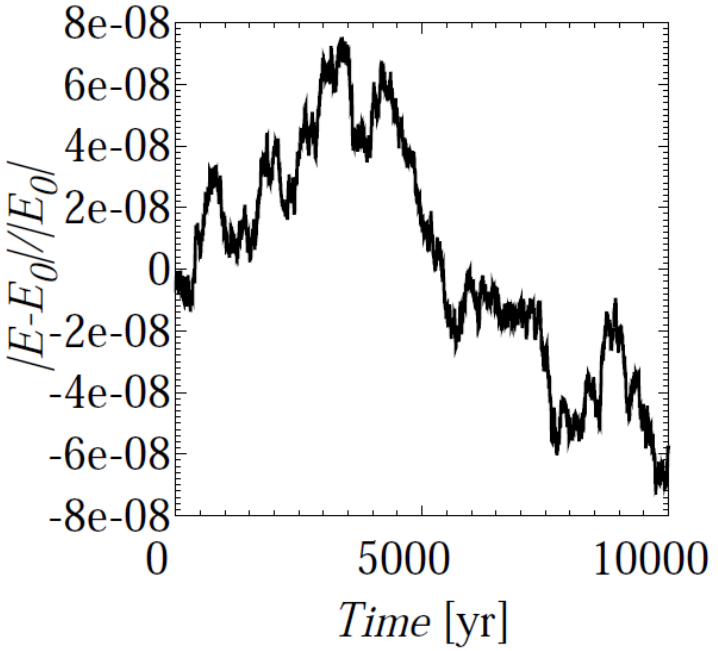
PPPT

Oshino et al (in prep)
PPPT (Particle-Particle, Particle-Tree) Hamiltonian

$$H = H_{Hard} + H_{Soft},$$
$$H_{Hard} = \sum_{i=1}^{N} \left(\frac{p_i^2}{2m_i} - \frac{G m_i m_\odot}{r_i} \right) - \sum_{i<j}^{N} \frac{G m_i m_j}{r_{ij}} W(r_{ij}),$$
$$H_{Soft} = \sum_{i<j}^{N} \frac{G m_i m_j}{r_{ij}} (1 - W(r_{ij})),$$

Separate near field and far field (cutoff could depend on particle mass)

PPPT example run

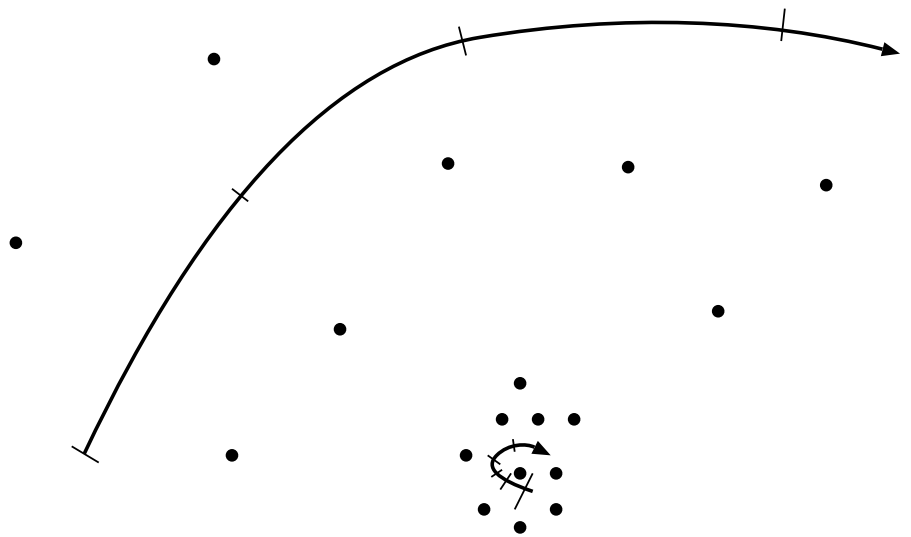


Planetesimal run (earth region 10^4 particles, $10^{-10} M_\odot$ particles)

Good enough for planet formation

Okay for star cluster?

Limit of individual timestep algorithm

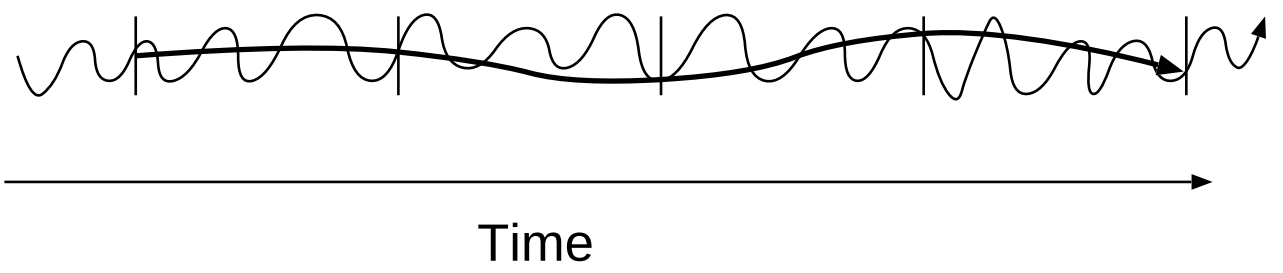


Basic idea of individual timestep:
Particles should have the timestep just enough to resolve their own orbits.

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

What's happening

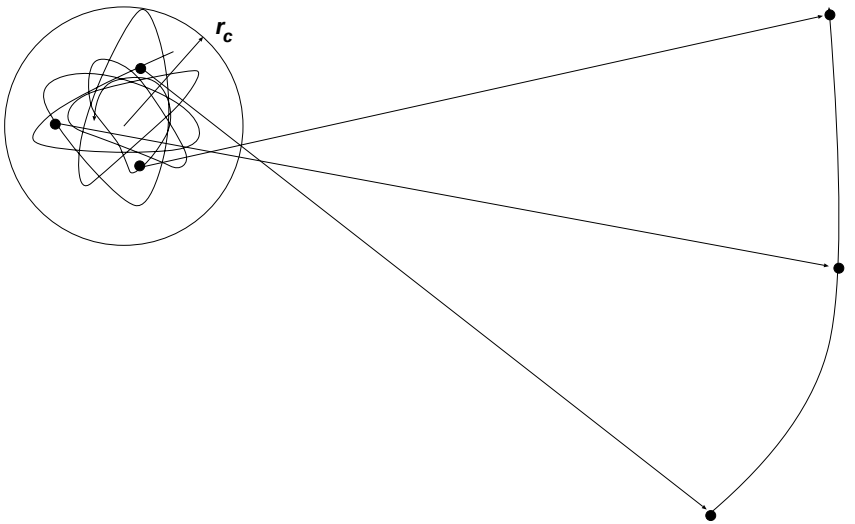
They are integrated in a completely wrong way!



- **Forces do have rapidly changing components**
- **If the timestep is large, forces are sampled "randomly" (if the orbit is not periodic)**

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: $r_c \sim r_h/N \ll r_h N^{-1/3}$



Does this really matter?

In the case of a singular isothermal cusp

- **The velocity change due to this error can be comparable to two-body relaxation (smaller by $N^{1/6}$).**
- **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**

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.

PPPT as Possible solution

- Use short enough timestep for tree part
- Accelerate tree part as much as possible
  - parallelization
  - GRAPE, GPU or whatever