Astrophysical N-body/SPH TREE-GRAPE/GPU implementations.

Peter Berczik

Astronomisches Rechen-Institut (ARI),
Zentrum für Astronomie Univ. Heidelberg, Germany

berczik@ari.uni-heidelberg.de

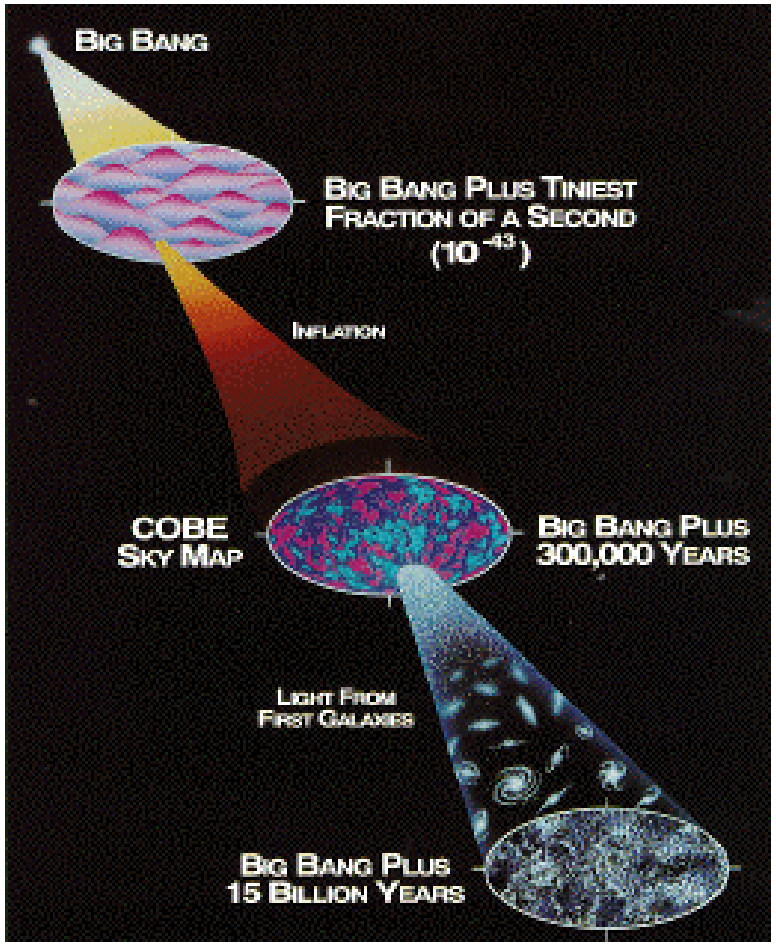
N-body 2008, Turku, Finland, 10 – 14 August 2008

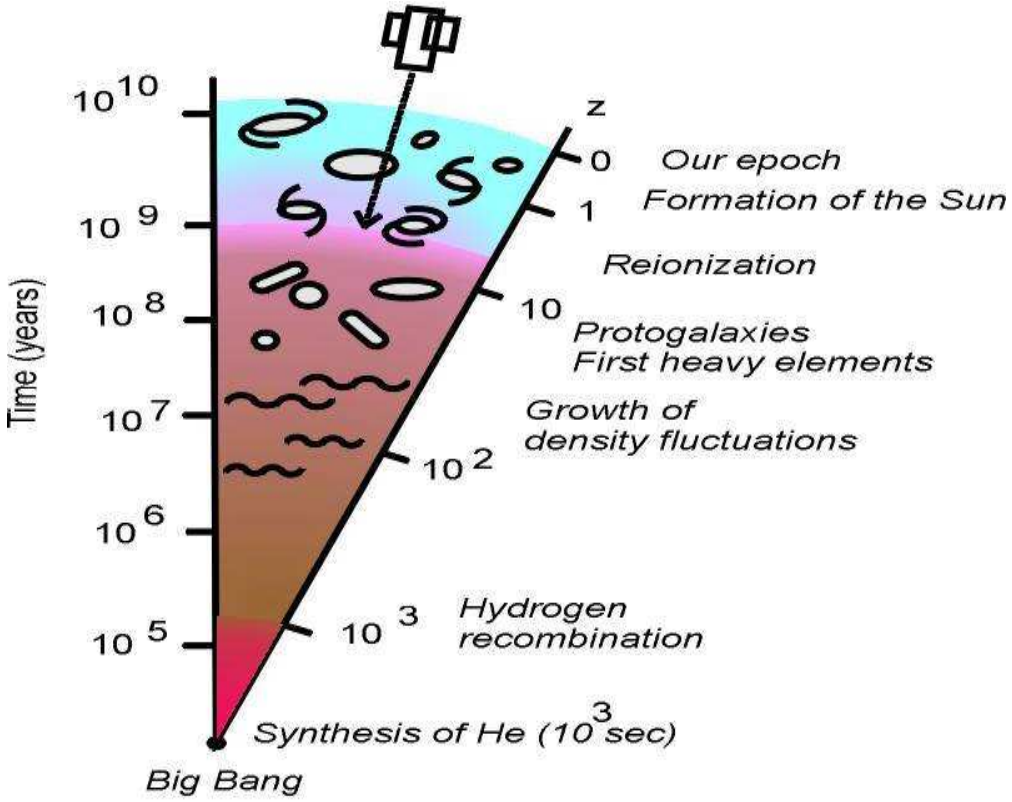
Collaborators & Grants

Naohito Nakasato	Univ. of Aizu, Japan
Keigo Nitadori	Tokyo Univ., Japan
Ingo Berentzen & Rainer Spurzem	Univ. Heidelberg
G. Marcus, G. Lienhart, A. Kugel, R. Maenner	Univ. Heidelberg
M. Wetzstein, T. Naab, A. Burkert	Univ. Munich

- **DFG SFB: No. 439/B11: 2005 - 2008**
- **Volkswagen/Baden-Württemberg, GRACE project: 2005 - 2008**

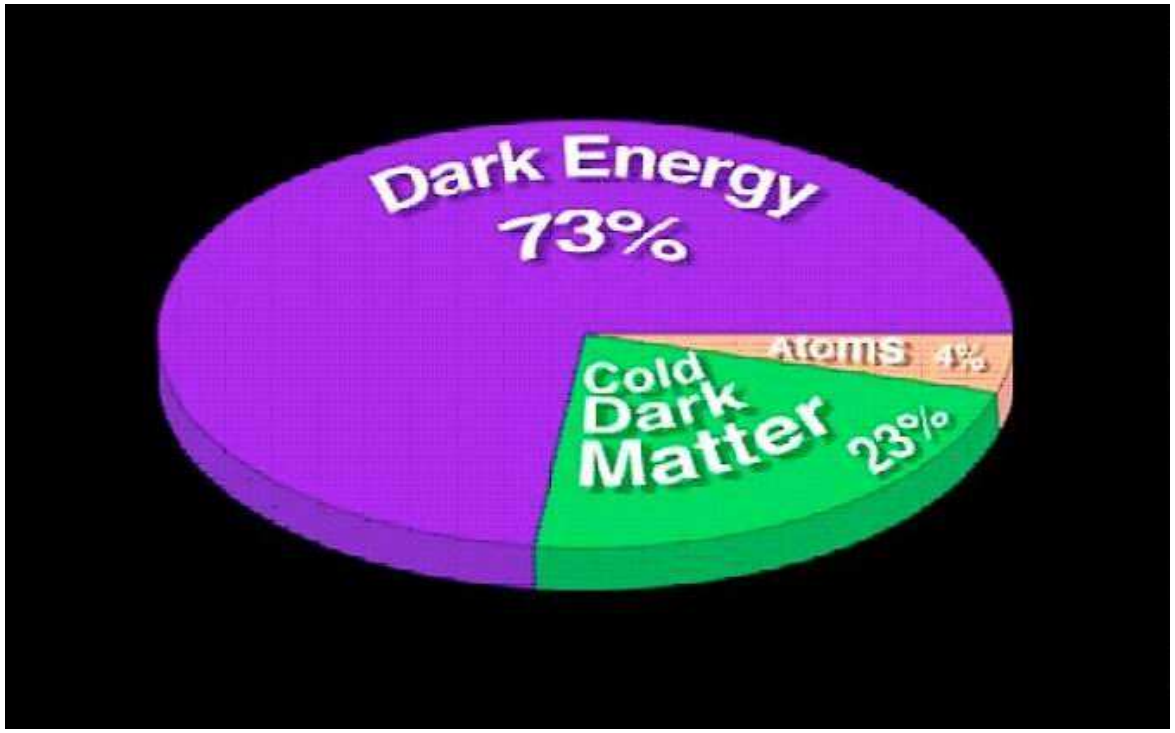
Formation of the Universe





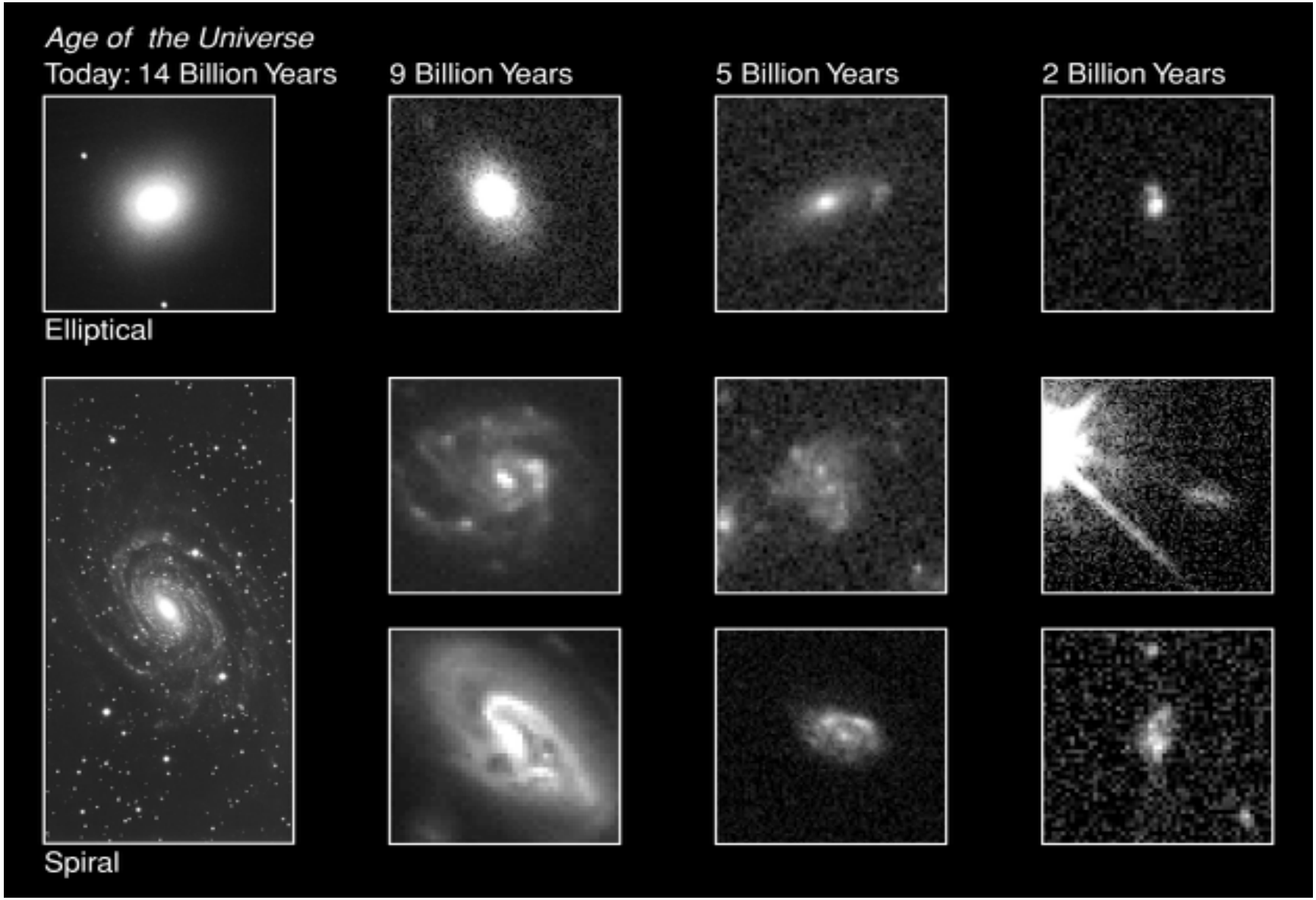
History of the Universe

Formation of the Universe

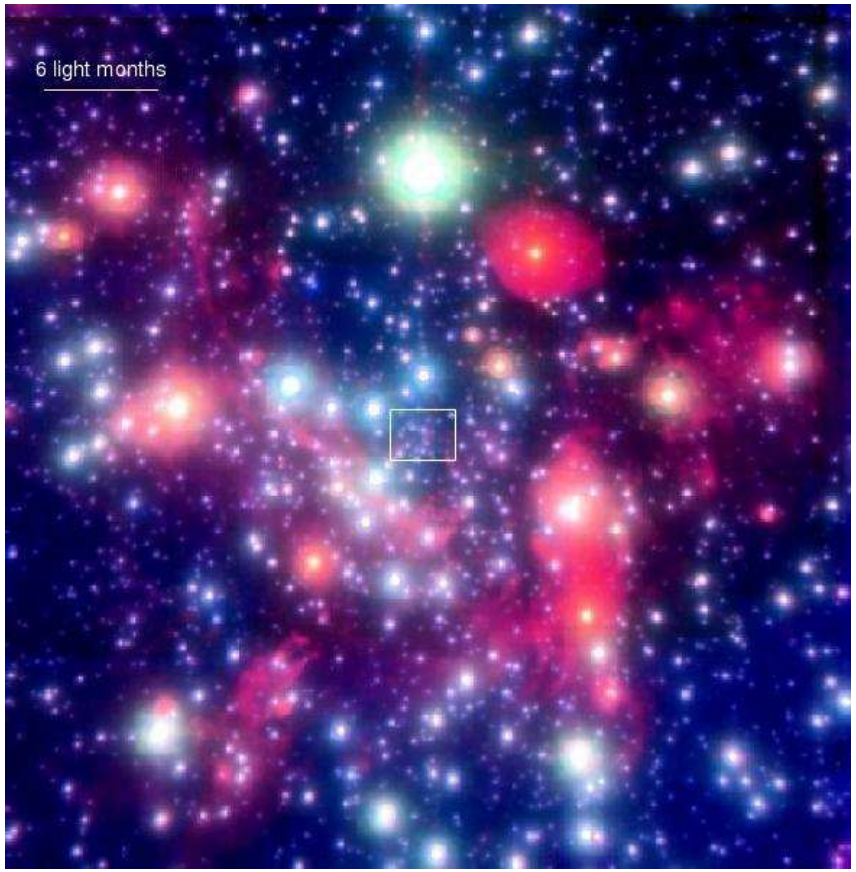


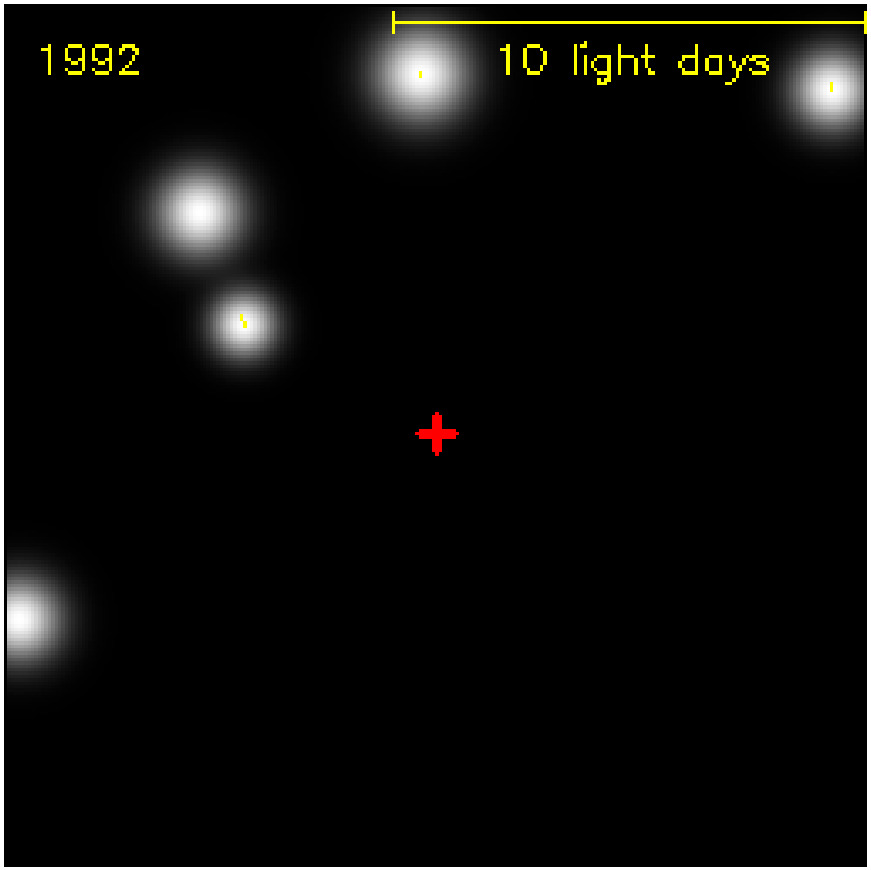
Darth Vader says:
Don't underestimate the power
of the dark side...

Observations

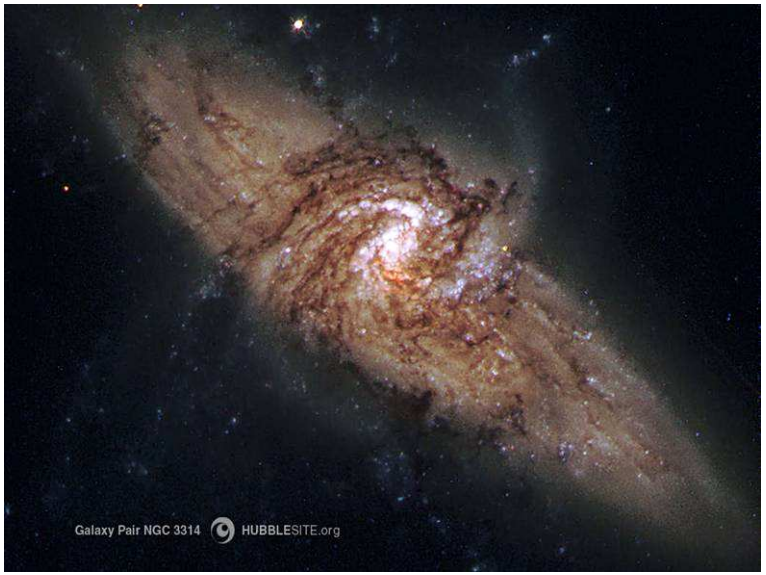


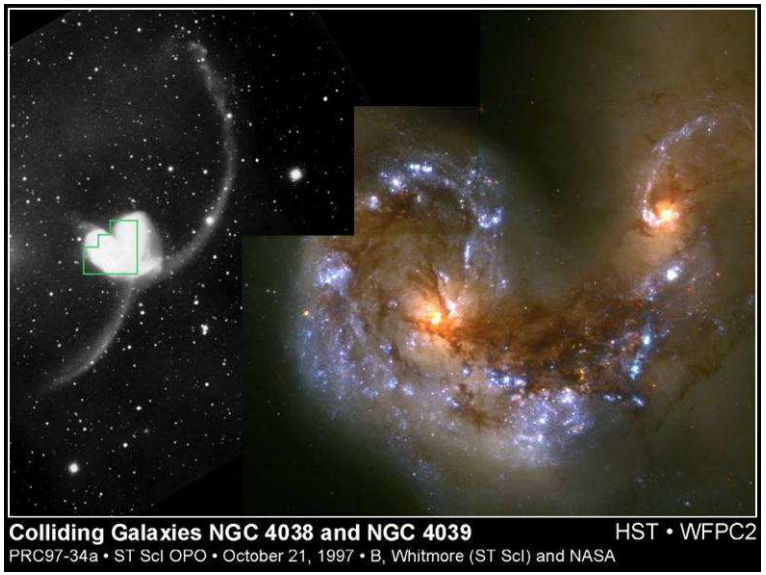
BH's in galaxies (MW - Sgr A*)





Galaxy Collisions

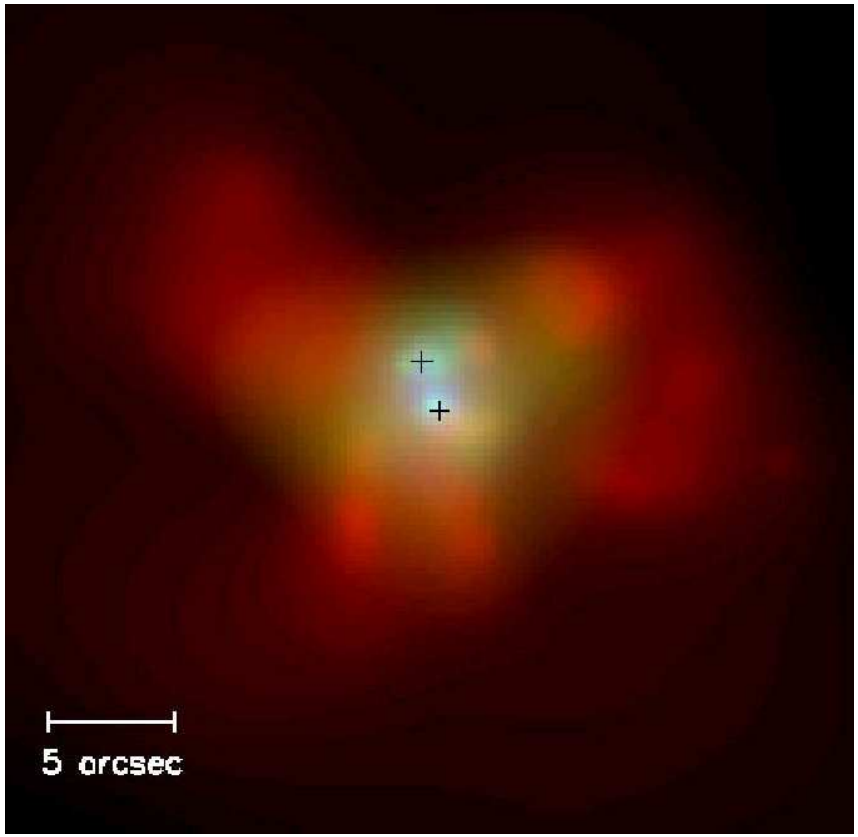




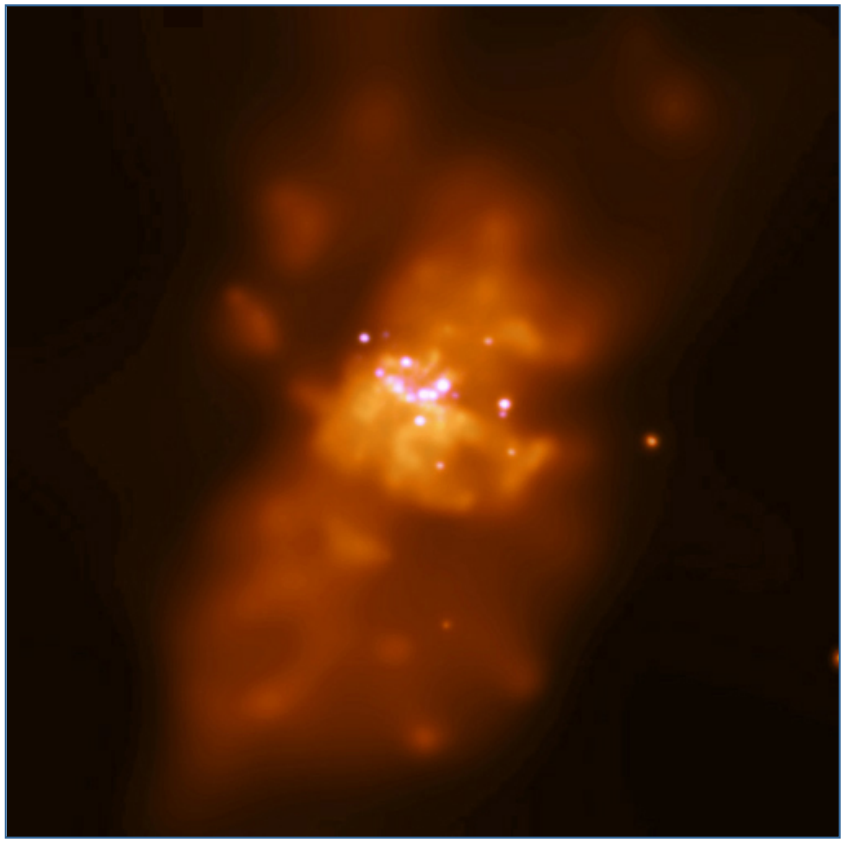
Galaxy Collisions ≈ BH's collisions

Multiple Massive Black Holes
NGC6240
strong ongoing merger…
Komossa et al. 2002

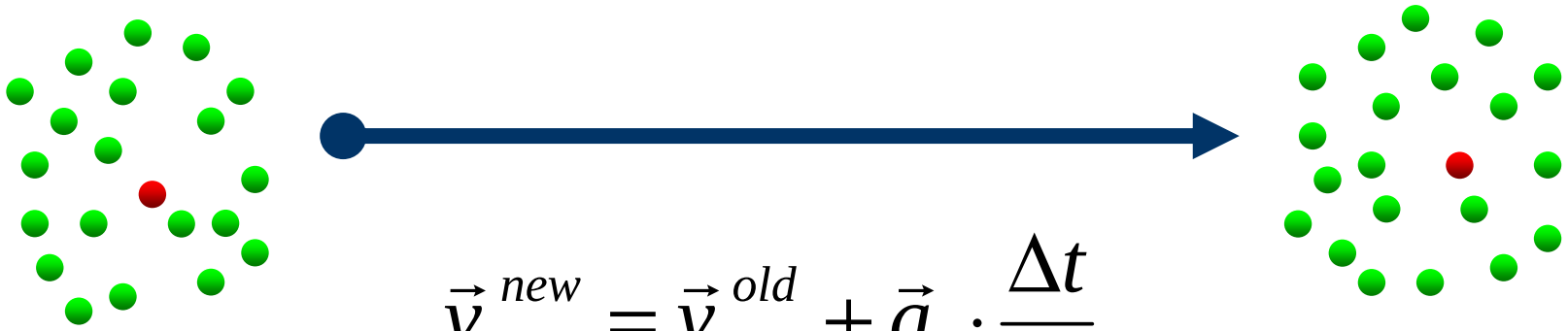
Two AGN in each of Nuclei separation
~1kpc Chandra X-Ray



M82: The bright spots in the center are supernova remnants and X-ray binaries. The luminosity of the X-ray binaries suggests that most contain a black hole. A close encounter with a large galaxy, M81, in the last 100 Myr is thought to be the cause of the starburst activity. Ebisuzaki et al. 2002



Basic idea of any N-body code



$$\vec{v}_i^{\,new} = \vec{v}_i^{\,old} + \vec{a}_i \cdot \frac{\Delta t}{2}$$

$$\vec{r}_i^{\,new} = \vec{r}_i^{\,old} + \vec{v}_i^{\,new} \cdot \Delta t$$

t^{old}

$t^{new} = t^{old} + \Delta t$

$$\vec{a}_i = -\sum_{j=1;\ j\neq i}^{N} \frac{G \cdot m_j}{(r_{ij}^2 + \varepsilon^2)^{3/2}} \vec{r}_{ij}$$

$\vec{r}_i^{\,old}$

$\vec{r}_i^{\,new}$

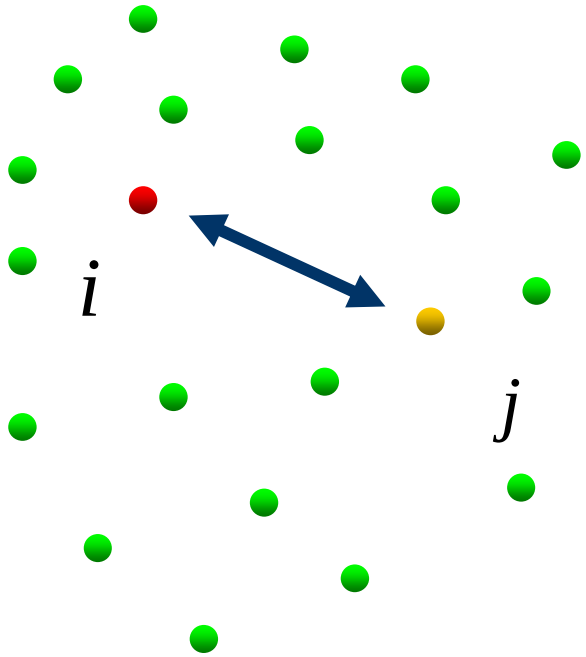
$\vec{v}_i^{\,old}$

$\vec{v}_i^{\,new}$

$$\vec{v}_i^{\,new} = \vec{v}_i^{\,new} + \vec{a}_i \cdot \frac{\Delta t}{2}$$

$$\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$$

Basic idea of any N-body code

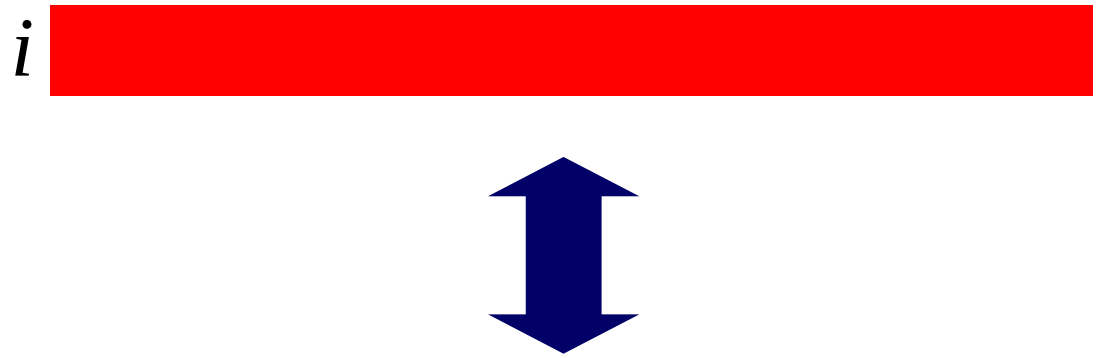


m_j

i

j

N



i

j

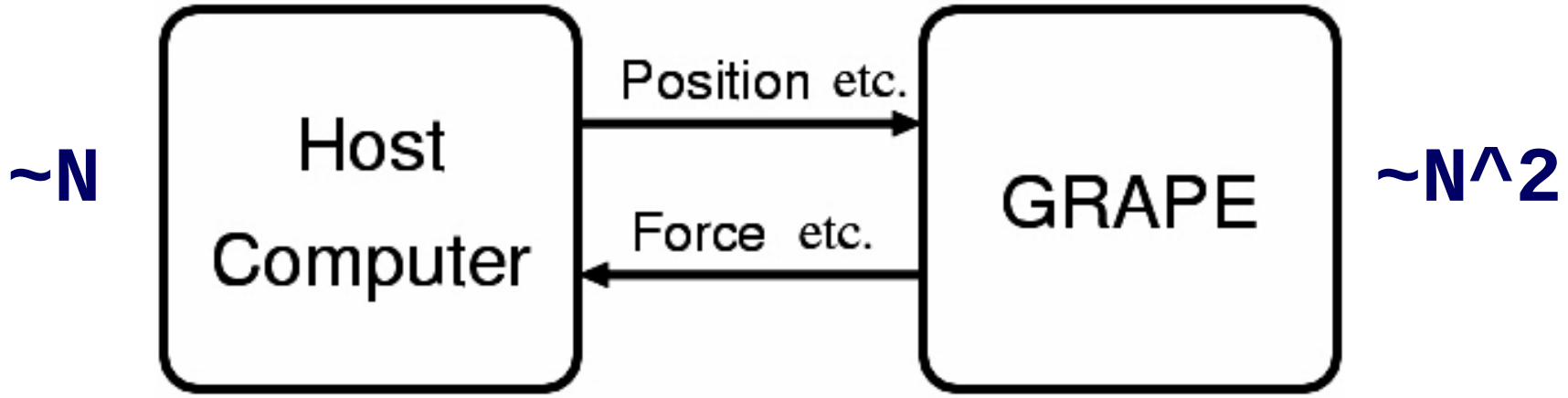
$\vec{r}_i$

t_i

$\vec{v}_i$

N

Basic idea of any GRAPE N-body code



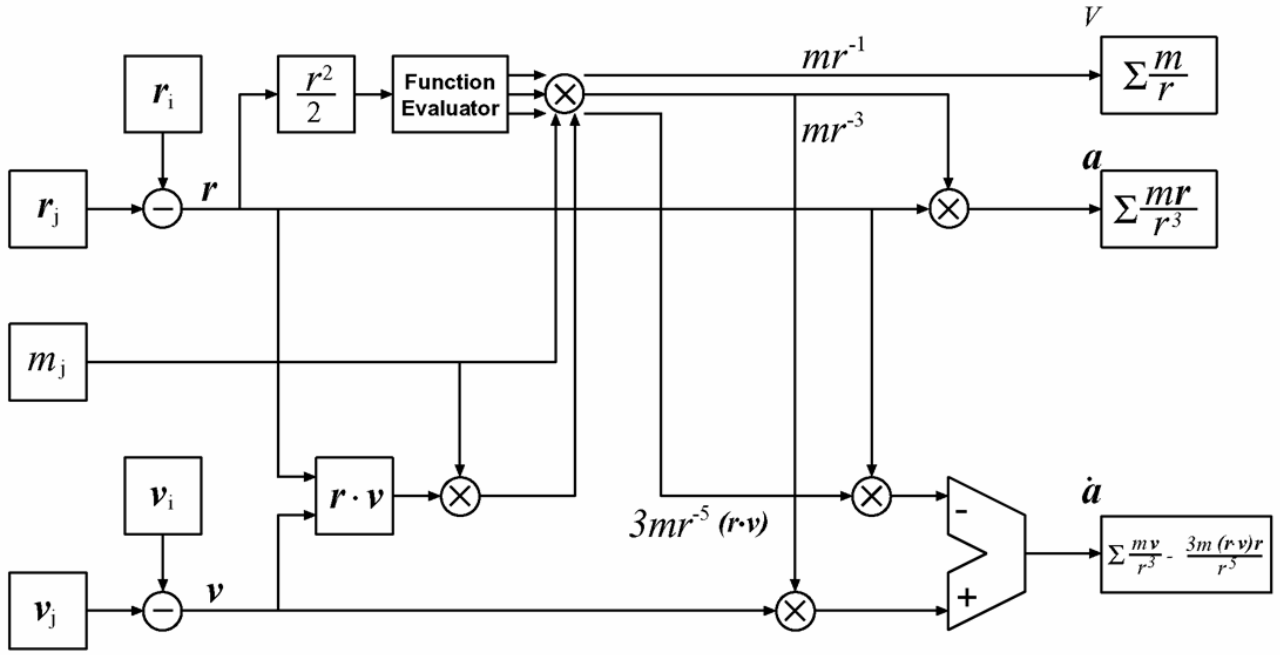
$$\vec{a}_i = \sum_{j=1;\ j\neq i}^{N} \vec{f}_{ij} \qquad \vec{f}_{ij} = -\frac{G \cdot m_j}{\left(r_{ij}^2 + \varepsilon^2\right)^{3/2}} \vec{r}_{ij}$$

GRAPE = GRAvity PipE – more detail…

$m_i; \vec{r}_i; \vec{v}_i; t_i$

48 pipelines

$m_j; \vec{r}_j; \vec{v}_j; t_j$



$\phi_i; \vec{a}_i; \dot{\vec{a}}_i$

Commerce GRAPE6a boards

http://www.metrix.co.jp

Profile

Hamamatsu Metrix
浜松メトリックス株式会社

Micro Grape 6Af

G6TM01

▲Rear side

▲Front Side

Grape 6 BL4

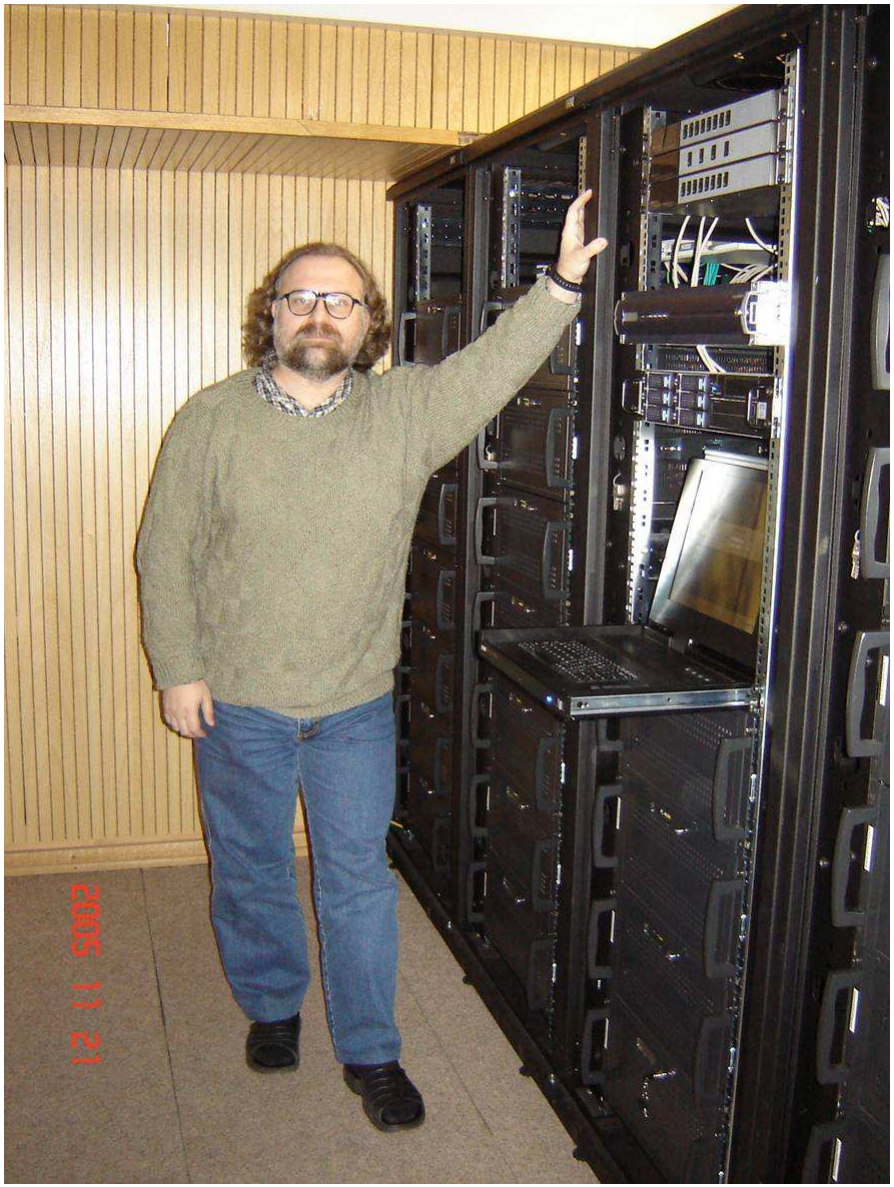
The most suitable for a Cluster system

▲Grape6 BL4 with Heatsink

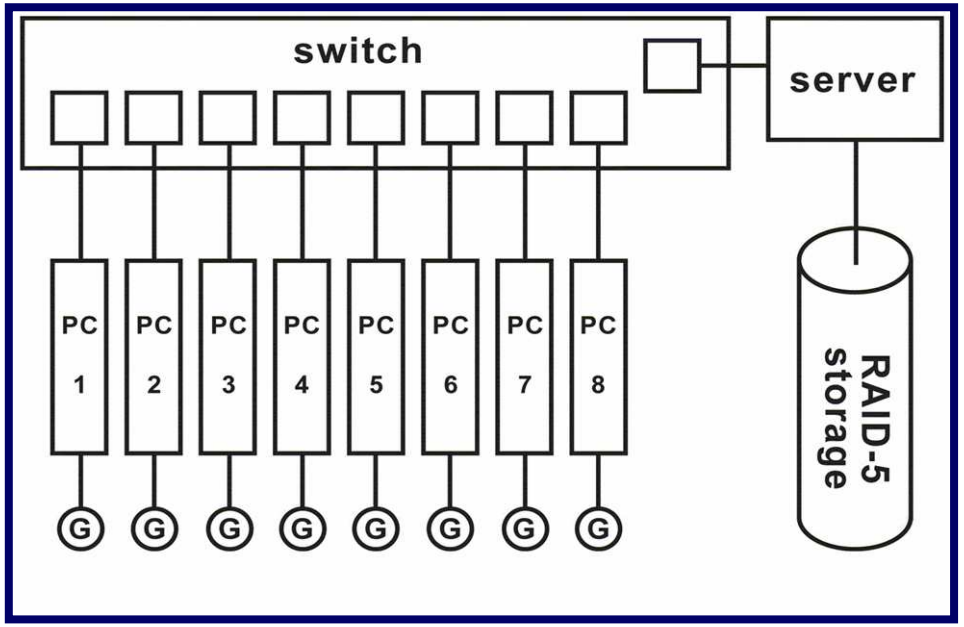
▲Front side

G6TM-01 assembly with G6TM-01MB module

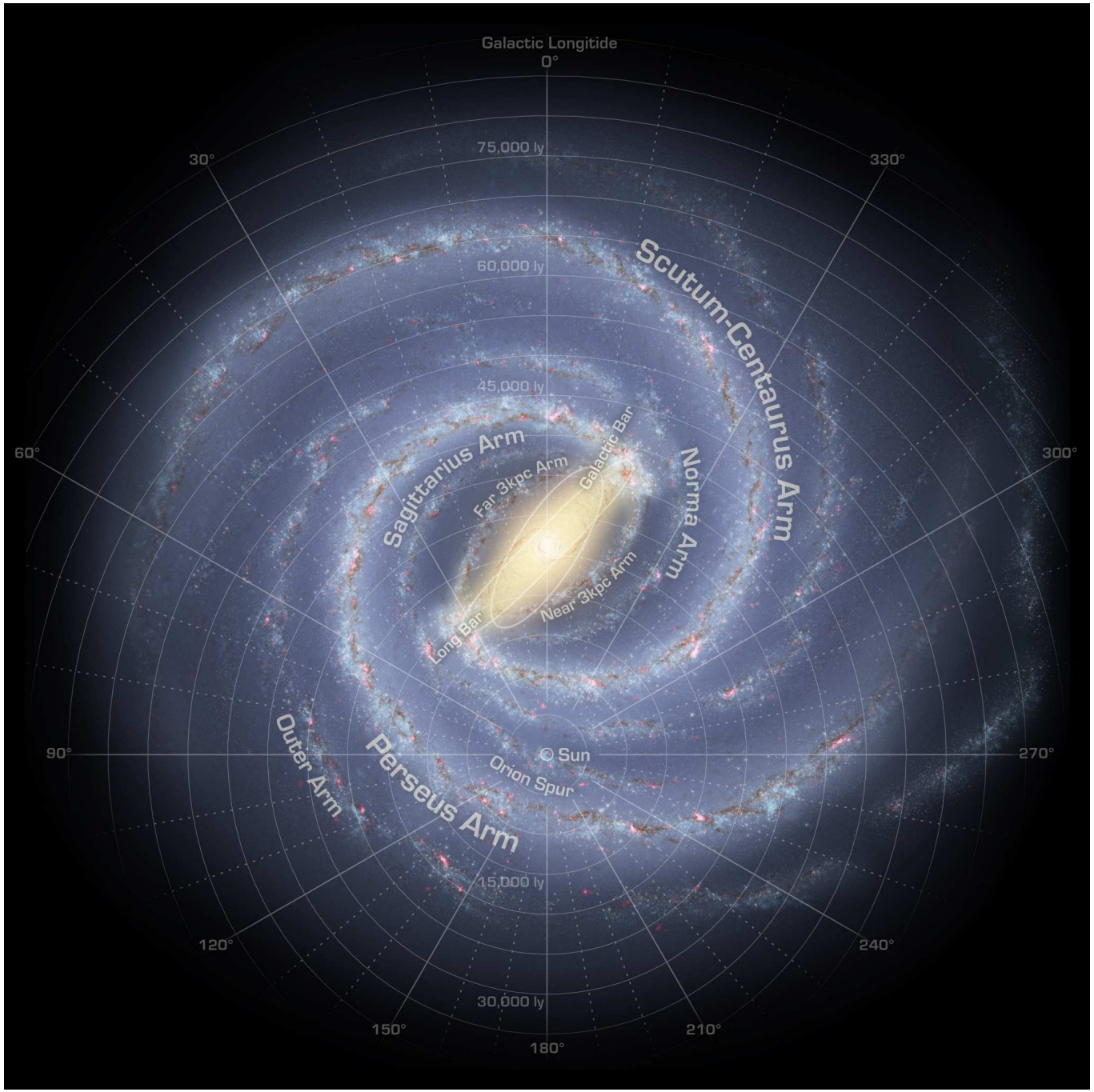
RIT & ARI 32 node GRAPE6a clusters



MAO 8+1 node GRAPE6 blx64 cluster



- 9 x2 dual-core Xeon 2.0 GHz
- 9 GRAPE6 blx64
- 5 TB RAID
- Infiniband switch (2x10 Gb/s)
- Speed: ~1 Tflops
- N up to 2M
- Cost: ~100k EUR
- Funding: NASU

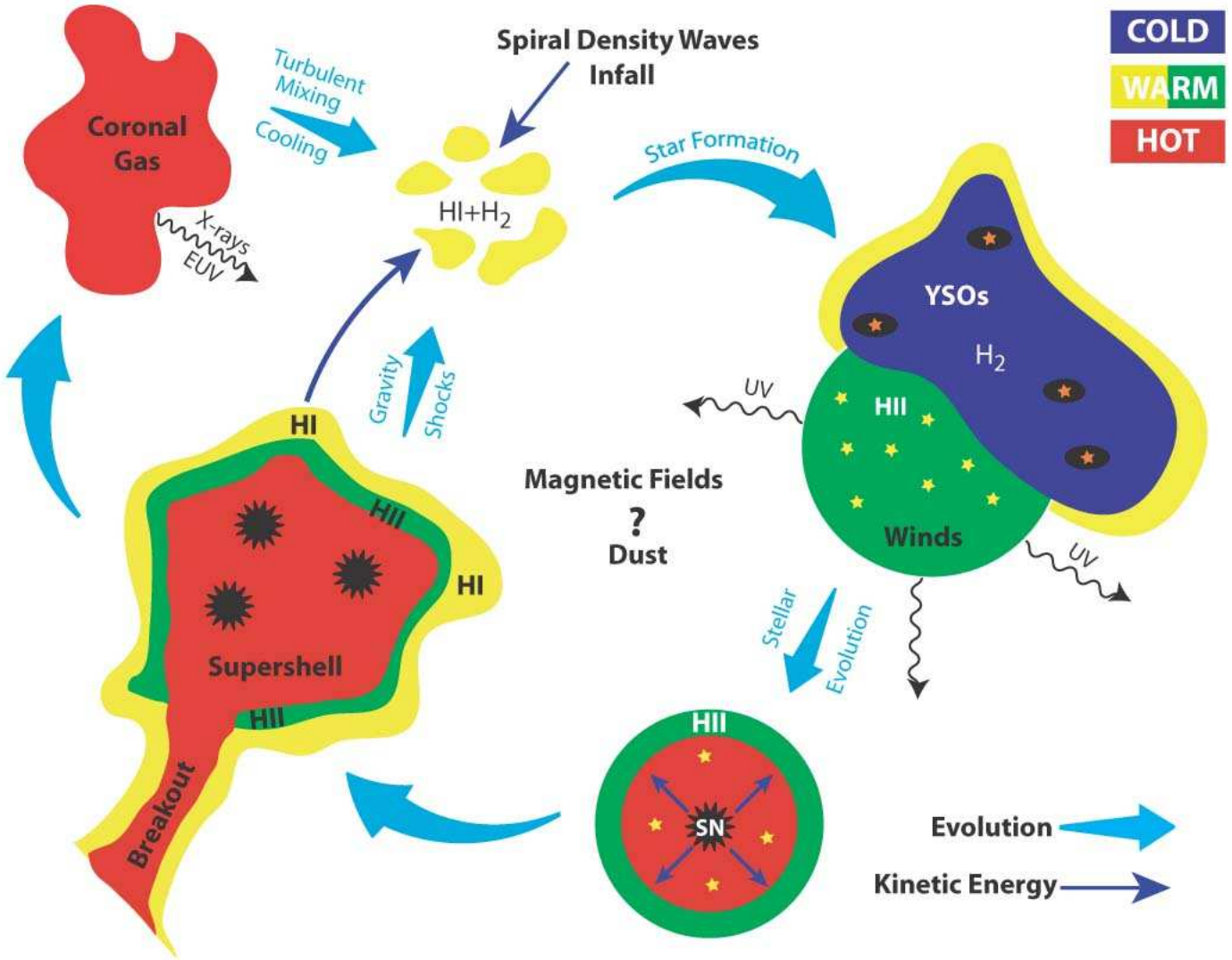


~80% stars

~20% gas

Highly multi-phase

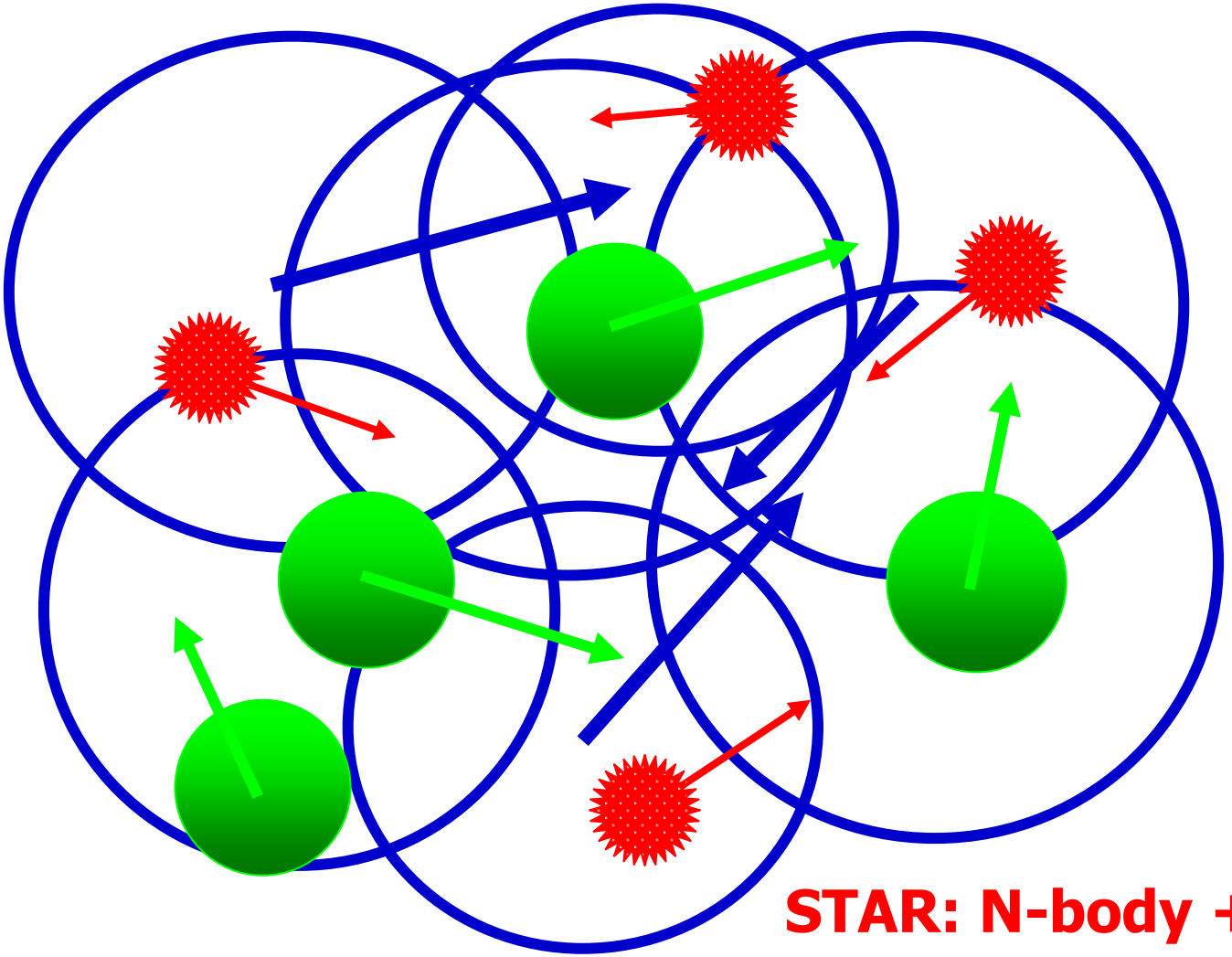
ISM "Ecology"



Tumlinson, 2004: astro-ph/0411249

Multi-Phase SPH code

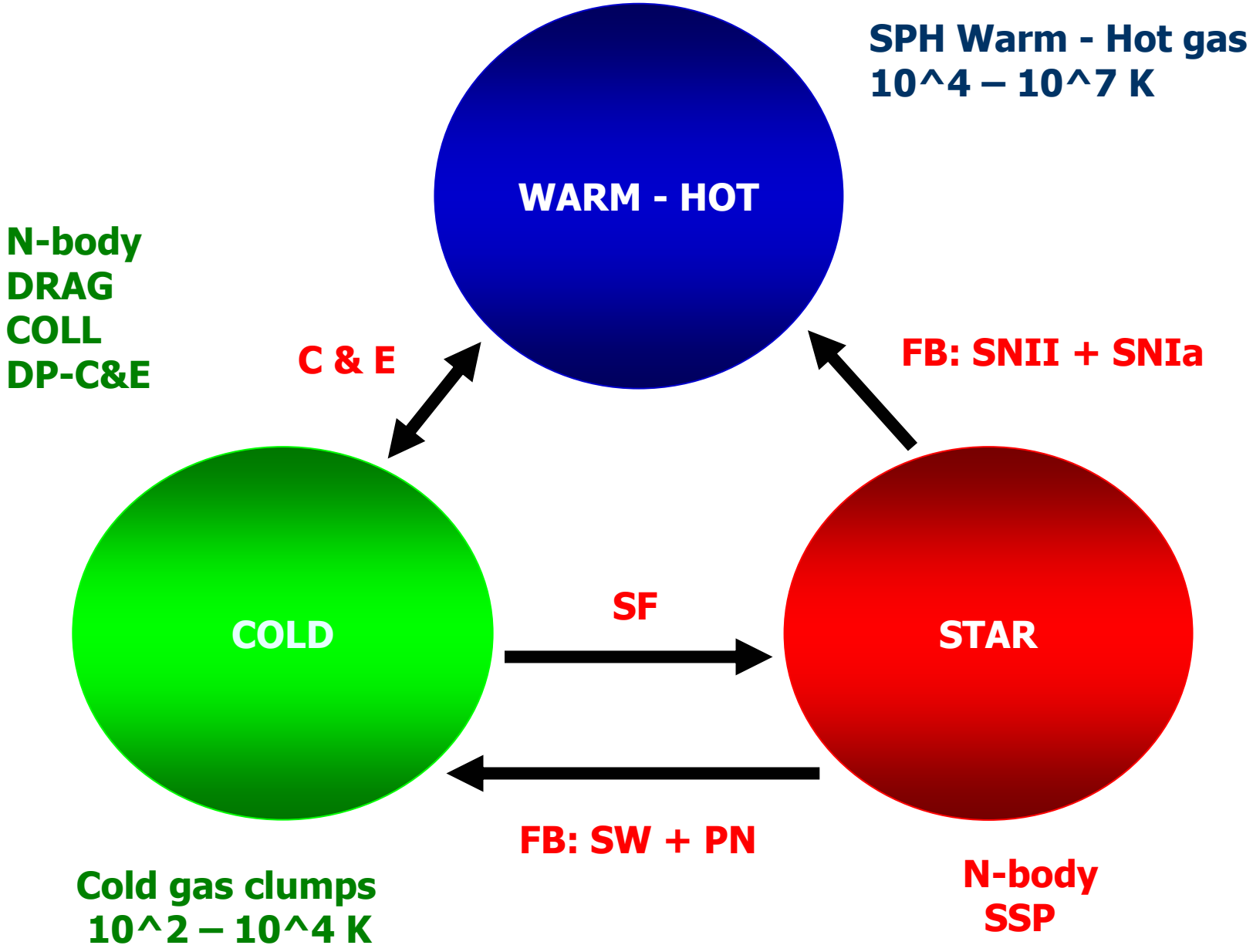
WARM - HOT: SPH



STAR: N-body + SSP

COLD: N-body + Viscosity

Multi-Phase SPH code



Basic Equations

Monaghan, 1977; Lucy, 1977

$$\rho_i \equiv \sum_{j=1}^{N} m_j \cdot W_{ij}$$

$$\frac{d\vec{v}_i}{dt} = -\sum_{j=1}^{N} m_j \cdot \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \tilde{\Pi}_{ij} \right) \cdot \vec{\nabla}_i W_{ij} - \vec{\nabla}_i \Phi_i - \vec{\nabla}_i \Phi_i^{ext}$$

$$\frac{du_i}{dt} = \frac{1}{2} \sum_{j=1}^{N} m_j \cdot \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \tilde{\Pi}_{ij} \right) \cdot \left(\vec{v}_i - \vec{v}_j \right) \cdot \vec{\nabla}_i W_{ij} + \frac{\Gamma_i - \Lambda_i}{\rho_i}$$

$$P_i = (\gamma - 1) \cdot \rho_i \cdot u_i$$

Basic Equations

Monaghan & Gingold, 1983

$$\Pi_{ij} = \begin{cases} \left[-\alpha \cdot c_{ij} \cdot \mu_{ij} + \beta \cdot \mu_{ij}^2 \right] / \rho_{ij} & \text{if } (\vec{r}_{ij} \cdot \vec{v}_{ij}) < 0 \\ 0 & \text{else} \end{cases}$$

$$\mu_{ij} = \frac{h_{ij} \cdot (\vec{v}_i - \vec{v}_j) \cdot (\vec{r}_i - \vec{r}_j)}{\left| \vec{r}_i - \vec{r}_j \right|^2 + \varepsilon \cdot h_{ij}^2}$$

$$\alpha = 1 \qquad \beta = 2 \qquad \varepsilon = 0.01$$

$$\rho_{ij} = \frac{1}{2}\left(\rho_i + \rho_j\right) \quad h_{ij} = \frac{1}{2}\left(h_i + h_j\right) \quad c_{ij} = \frac{1}{2}\left(c_i + c_j\right)$$

Basic Equations

Monaghan & Lattanzio, 1985 **Hernquist & Katz, 1989**

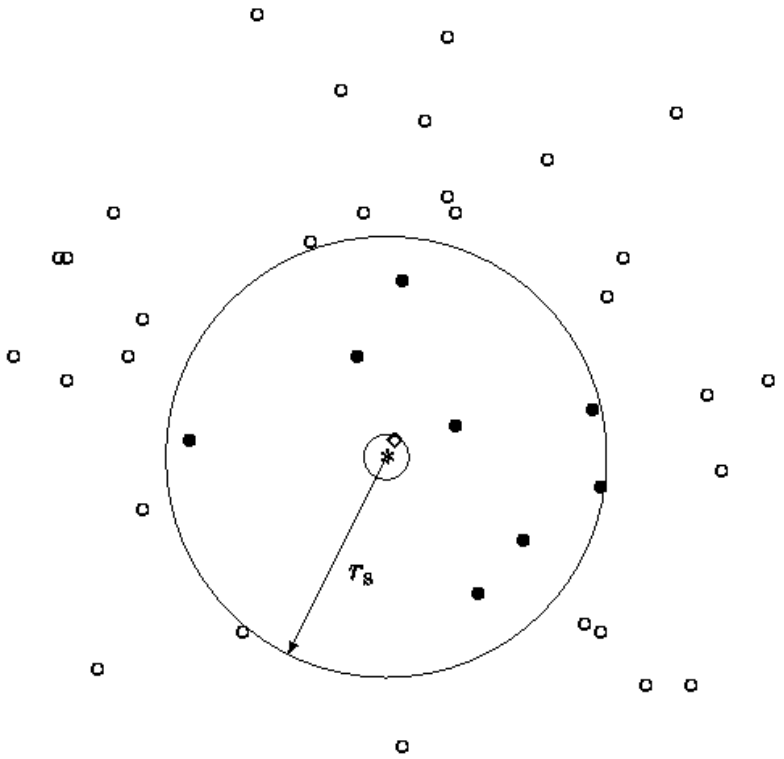
$$W(r;h)=\frac{1}{\pi\cdot h^3}\cdot\begin{cases}1-\frac{3}{2}(r/h)^2+\frac{3}{4}(r/h)^3 & 0\le r/h<1\\ \frac{1}{4}(2-r/h)^3 & 1\le r/h<2\\ 0 & 2\le r/h\end{cases}$$

$$W_{ij}=W\left(\left|\vec{r}_i-\vec{r}_j\right|;h_{ij}\right)$$

Balsara, 1995; Steinmetz, 1996

$$\tilde{\Pi}_{ij}=\frac{1}{2}\left(f_i+f_j\right)\cdot\Pi_{ij}\qquad f_i=\frac{\left|\left(\vec{\nabla}\cdot\vec{v}\right)_i\right|}{\left|\left(\vec{\nabla}\cdot\vec{v}\right)_i\right|+\left|\left(\vec{\nabla}\times\vec{v}\right)_i\right|+\varepsilon\cdot c_i/h_i}$$

Define smoothing length



$$\left|r_{ij}\right| \le 2 \cdot \max\left(h_i; h_j\right)$$

Inside "2•h" N_B = const =50

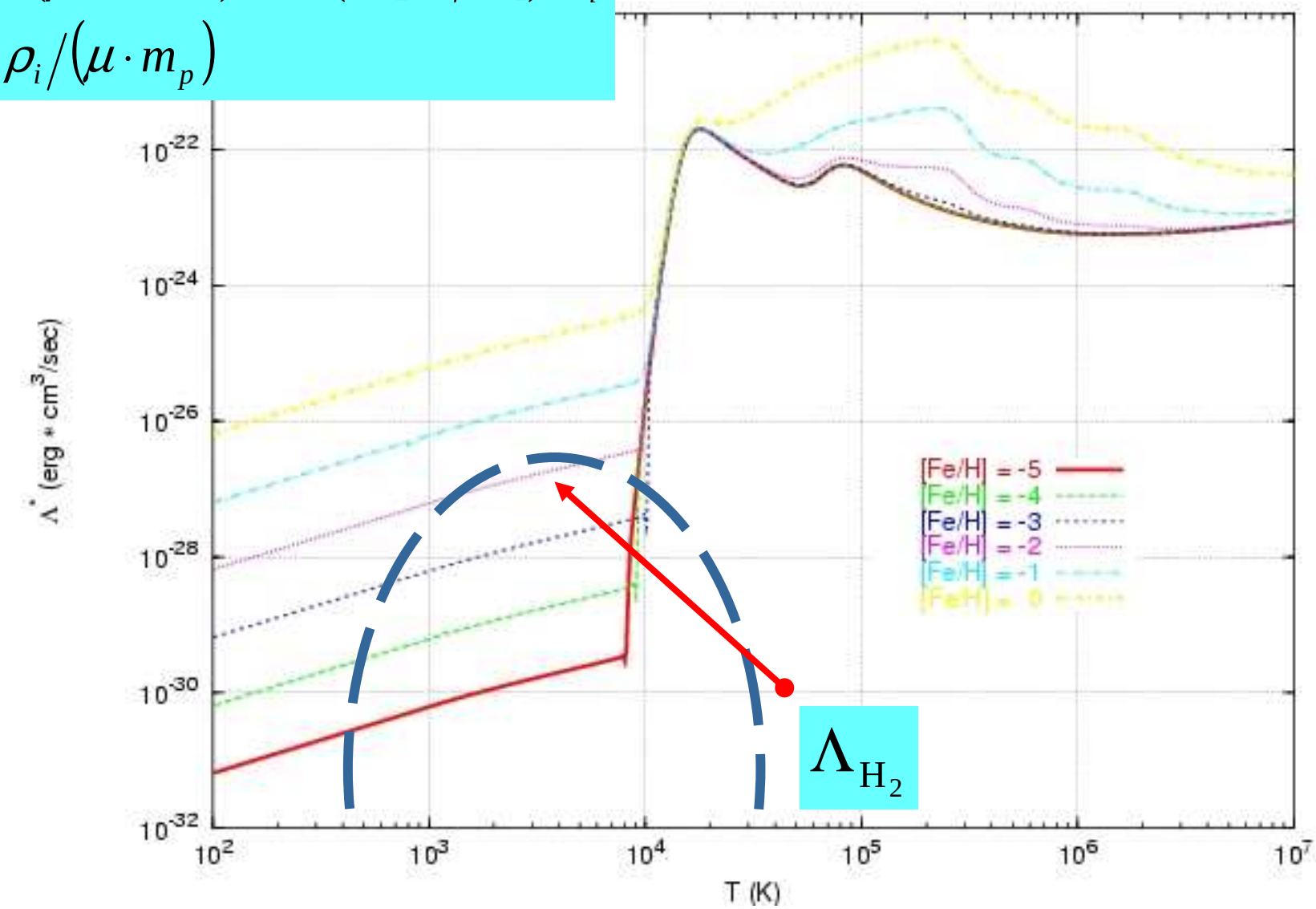
Using the ANN with kdtree

http://www.cs.umd.edu/~mount/ANN/

Delgarno & McCray, 1972; Sutherland & Dopita, 1993

$$\Lambda \equiv \Lambda(\rho, u, \mathrm{Z}, \ldots) \cong \Lambda^{*}(\mathrm{T}, [\mathrm{Fe/H}]) \cdot n_i^2$$

$$n_i^2 = \rho_i / (\mu \cdot m_p)$$



Integrator

Predictor step:

$$\begin{cases} \vec{v}_i^{\,p} = \vec{v}_i^{\,n} + \vec{a}_i^{\,n} \cdot \Delta t \\ \vec{r}_i^{\,p} = \vec{r}_i^{\,n} + (\vec{v}_i^{\,n} + \vec{v}_i^{\,p}) \cdot \dfrac{\Delta t}{2} \\ u_i^{\,p} = u_i^{\,n} + \dot{u}_i^{\,n} \cdot \Delta t \end{cases}$$

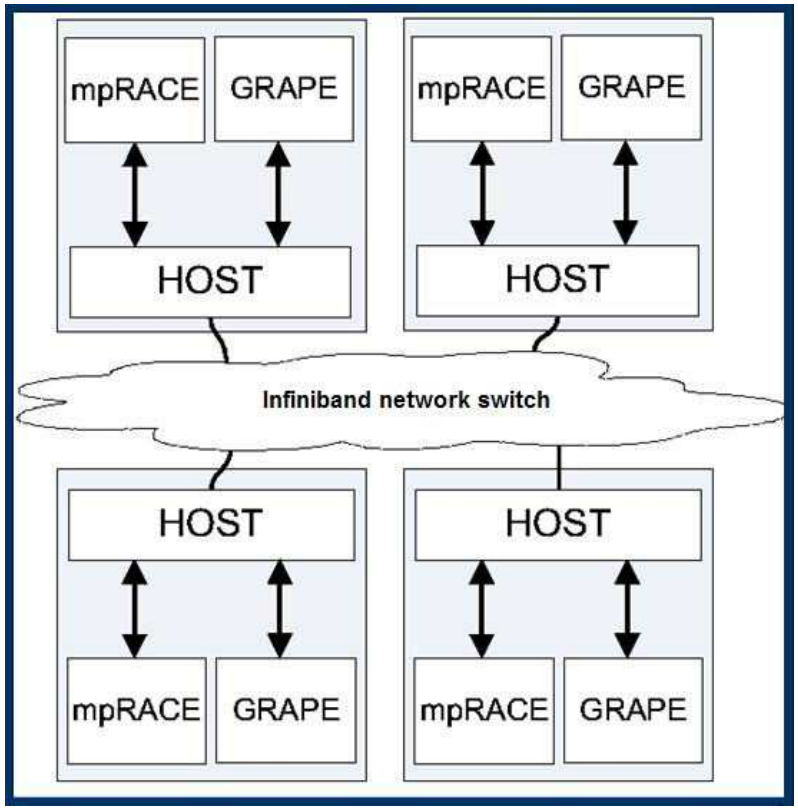
Corrector step:

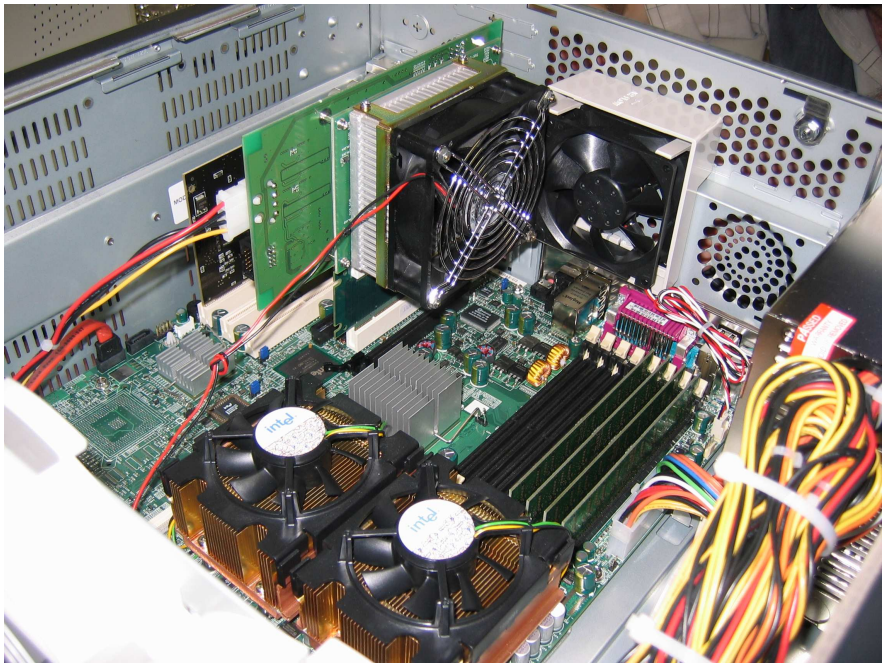
$$\begin{cases} \vec{v}_i^{\,n+1} = \vec{v}_i^{\,n} + (\vec{a}_i^{\,n} + \vec{a}_i^{\,p}) \cdot \dfrac{\Delta t}{2} \\ \vec{r}_i^{\,n+1} = \vec{r}_i^{\,n} + (\vec{v}_i^{\,n} + \vec{v}_i^{\,n+1}) \cdot \dfrac{\Delta t}{2} \\ u_i^{\,n+1} = u_i^{\,n} + (\dot{u}_i^{\,n} + \dot{u}_i^{\,p}) \cdot \dfrac{\Delta t}{2} \end{cases}$$

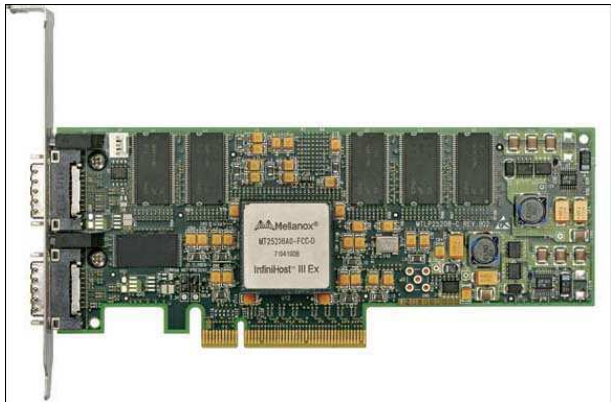
Time step:

$$\Delta t = 0.1 \cdot \min\left(\sqrt{\frac{2 \cdot h_i}{|\vec{a}_i|}}; \frac{h_i}{|\vec{v}_i|}; \frac{h_i}{c_i}; \frac{u_i}{\dot{u}_i} \right)$$

ARI 32 node GRAPE6a cluster





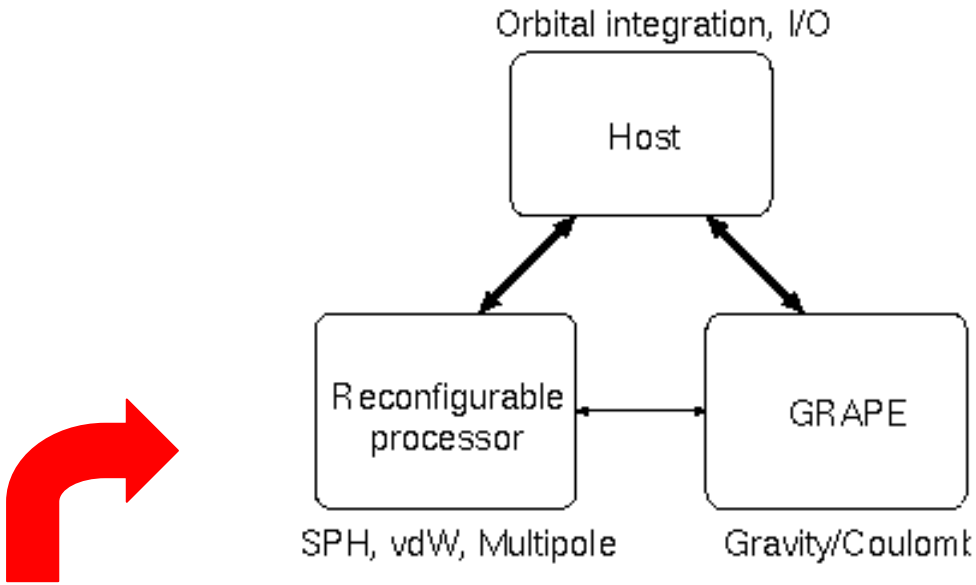


32x2 64 bit-Xeon P4, 3.2 GHz (~2 Gfps)
32 GRAPE6a (~120 Gfps)
32 FPGA-MPRACE (~20 Gfps)
3.5 TB RAID5 disk system
Infiniband, dual port network (~20 Gb/s)

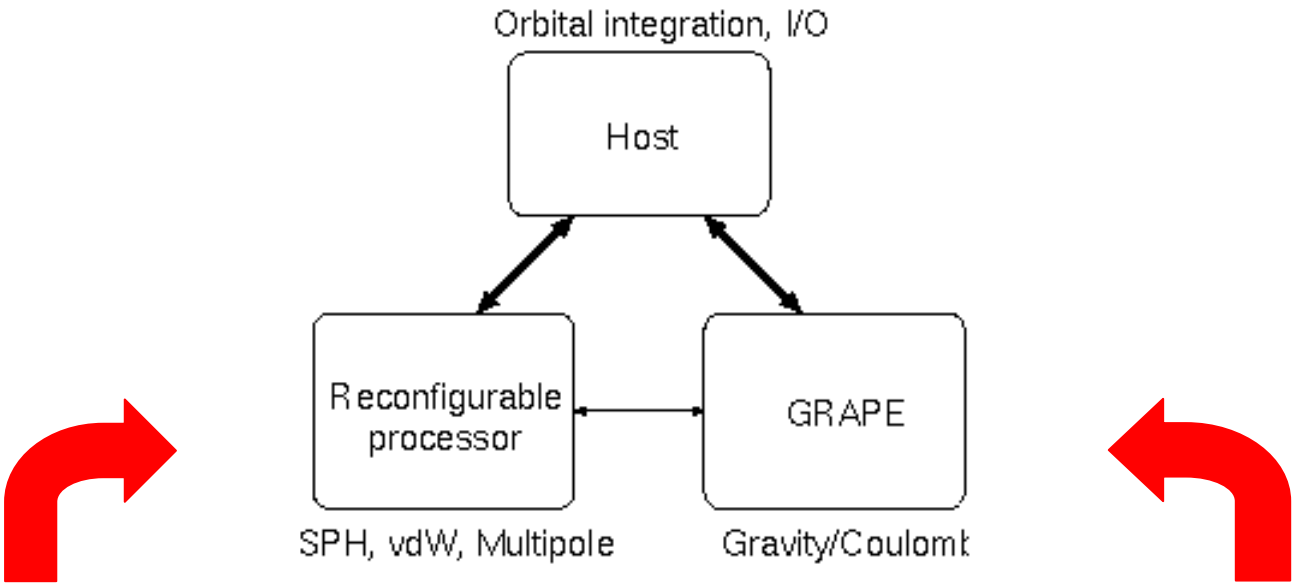
Summary speed: ~4 Tfps
N (direct summation) up to 4M

Volkswagen/Baden-Württemberg ~400k EUR

GRACE=GRAPE + MPRACE

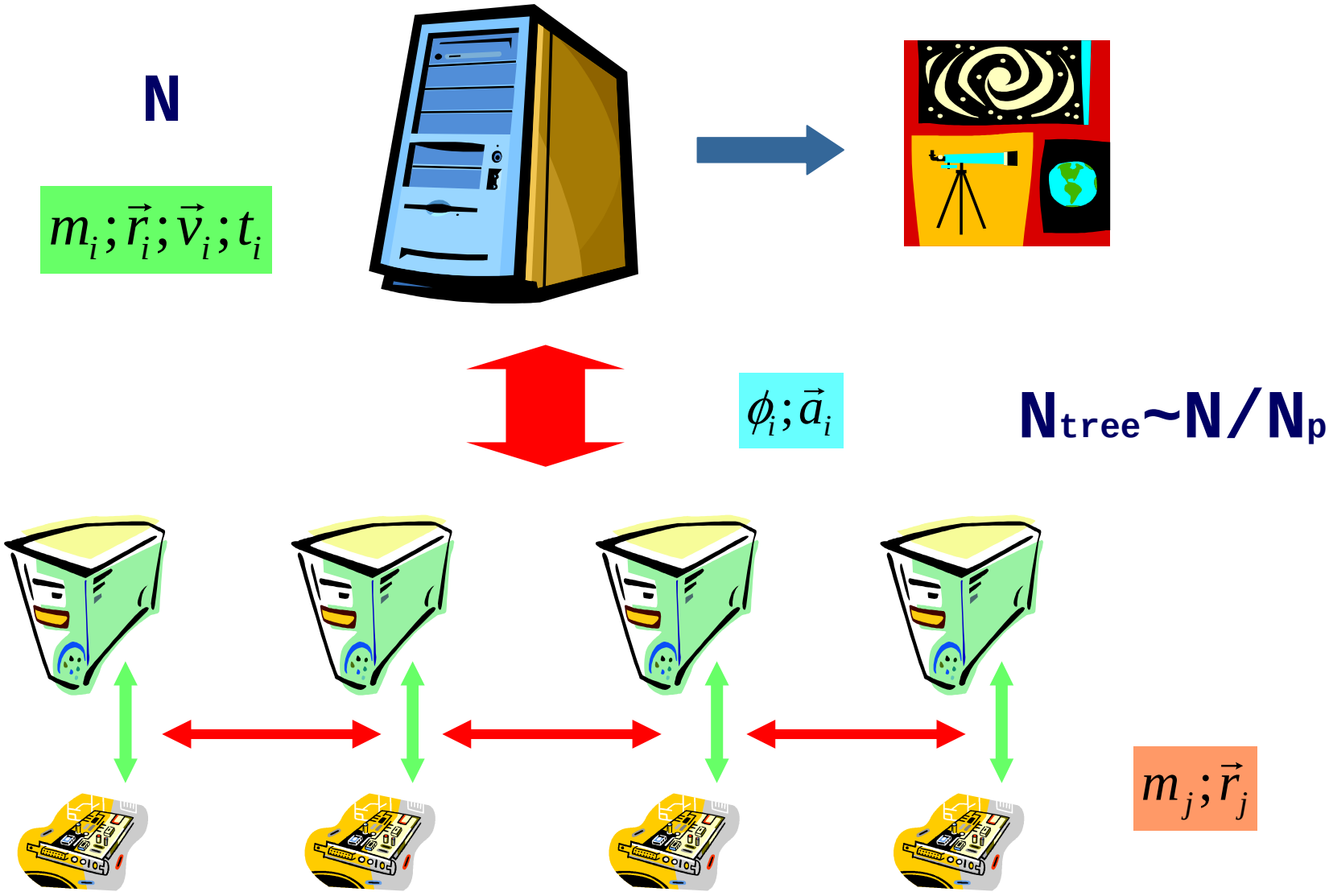


GPU...


Orbital integration, I/O
Host
Reconfigurable processor
GRAPE
SPH, vdW, Multipole
Gravity/Coulomb

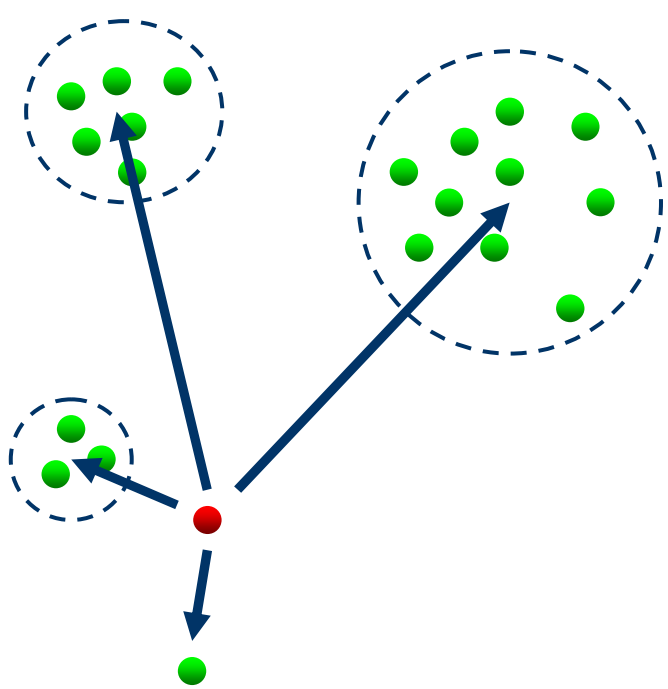

EVGA
GeForce GTX 280
EVGA

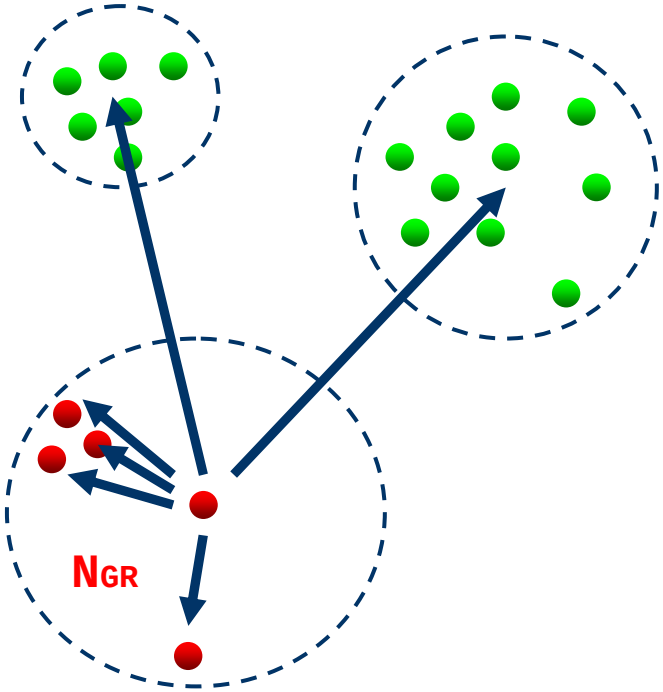
Parallel TREE gravity on the cluster



Parallel TREE gravity on the cluster

Jun Makino (pC++): TREE+GRAPE code



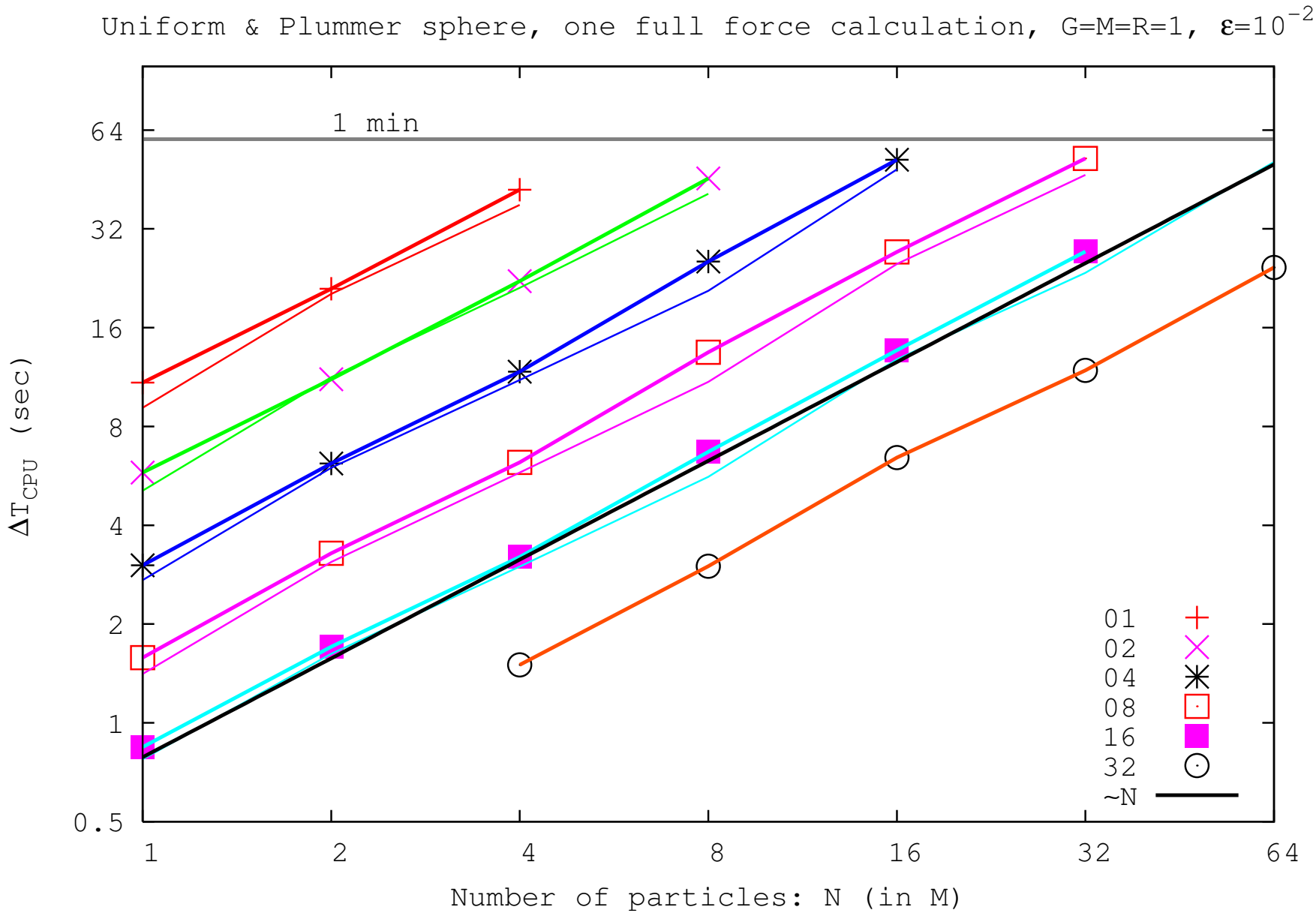


Makino, PASJ, 43, 621 (1991)

Inter. list on host ~N
Inter. list length -> short…

Makino, PASJ, 56, 521 (2004)
Fukushige, Makino & Kawai, PASJ, 57, 1009 (2005)
One interaction list is shared among NGR particles!
Inter. list on host ~N/NGR
Inter. list length -> larger…

Parallel TREE gravity on the cluster



Parallel TREE gravity on the cluster

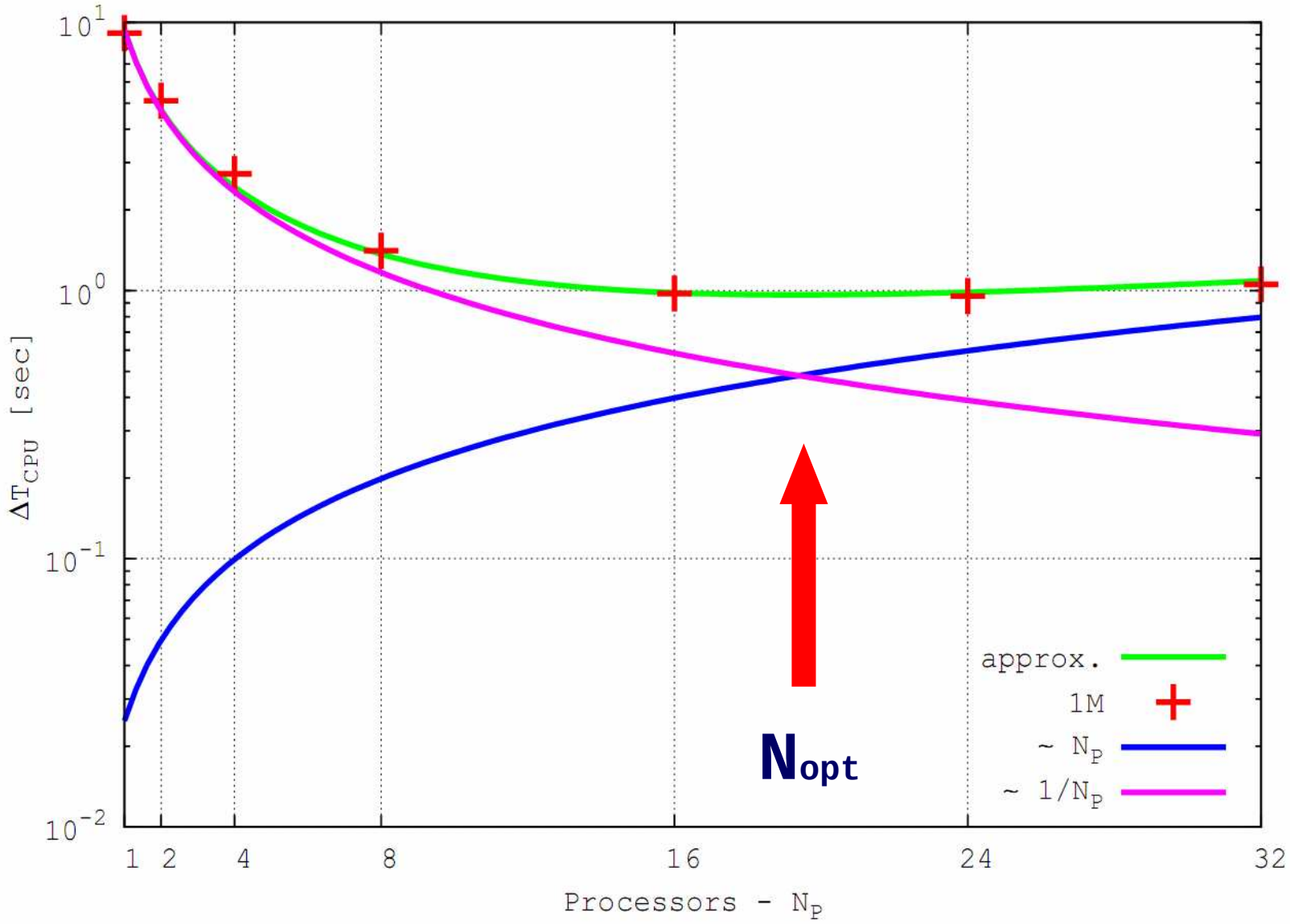
ε=0.001; θ=0.75; 10<N_{GR}<3500

$$\Delta T_{total} = \Delta T_{constr} + \Delta T_{calc} + \Delta T_{comm}$$

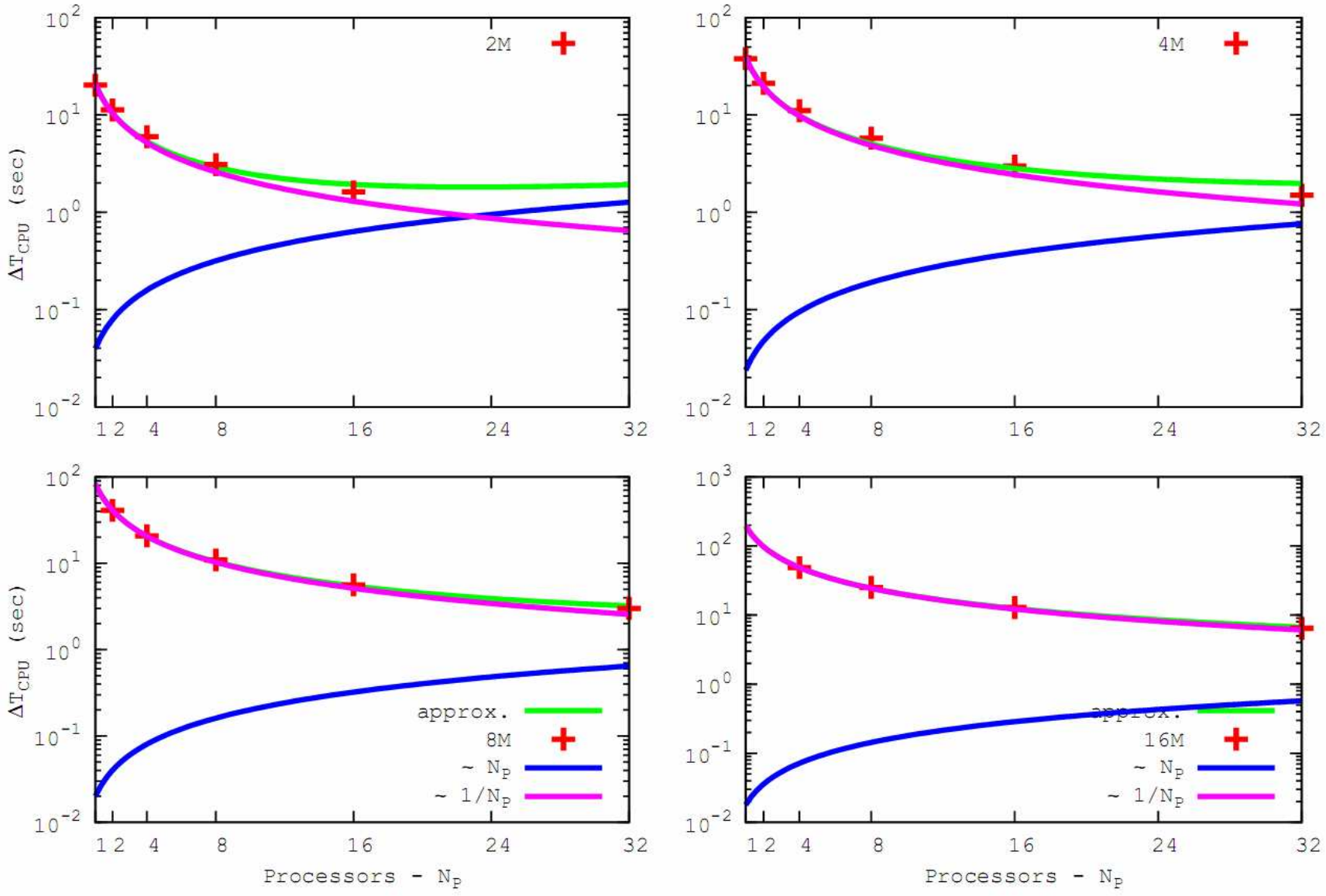
$$\Delta T_{constr} \propto \frac{N}{N_{proc}} \qquad \Delta T_{calc} \propto \frac{N \cdot \log(N)}{N_{proc}}$$

$$\Delta T_{comm} \propto N \cdot N_{proc}$$

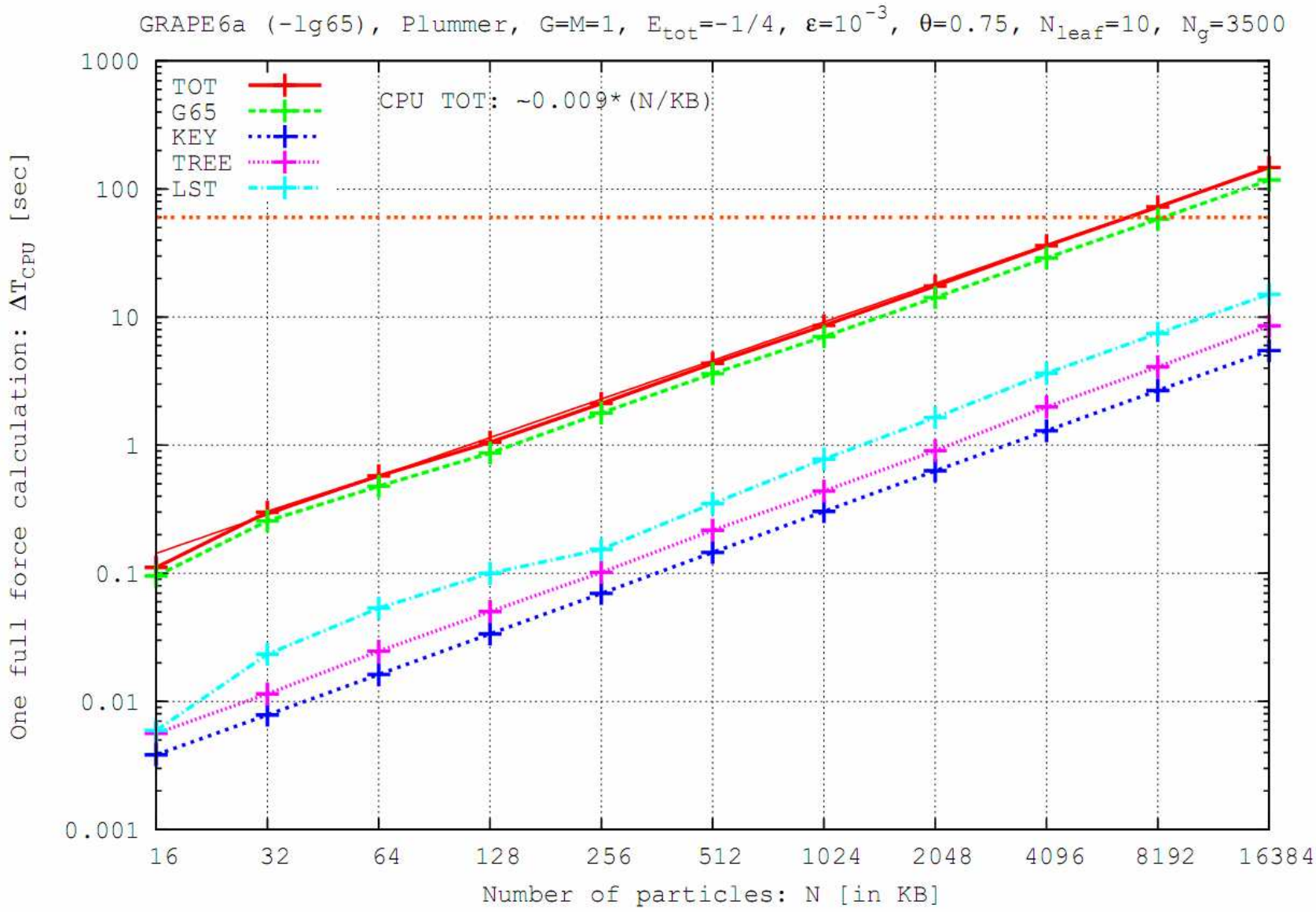
Parallel TREE gravity on the cluster



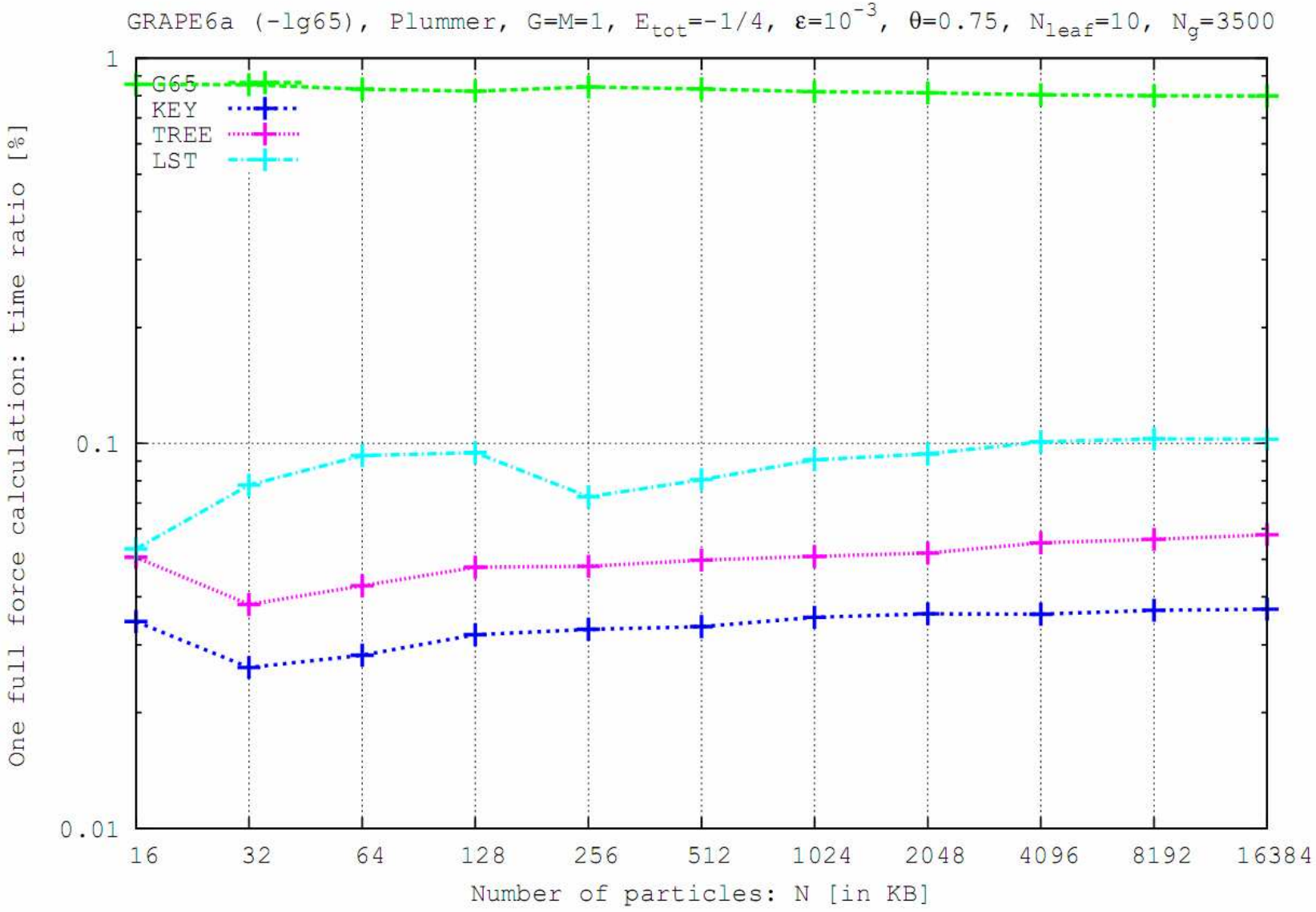
Parallel TREE gravity on the cluster



Parallel TREE gravity on the cluster



Parallel TREE gravity on the cluster



SPH - test

Adiabatic collapse of a cold gas sphere.

Evrard, 1988
Steinmetz & Muller, 1993
Carraro et al., 1998
Springel et al., 2001

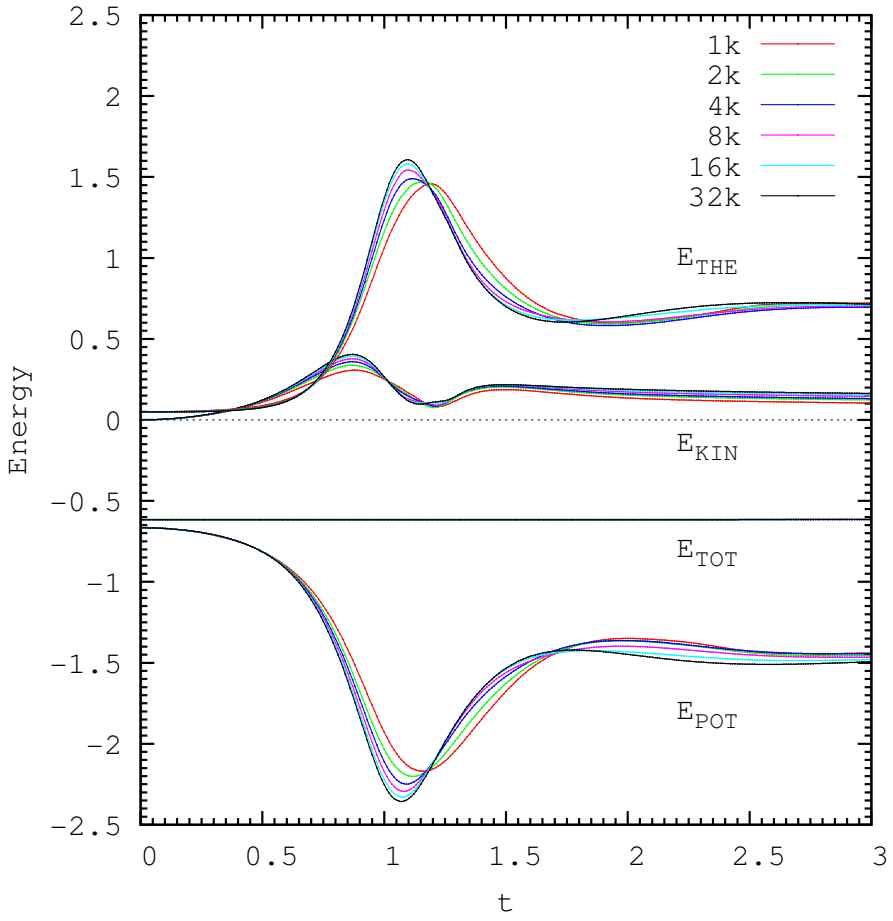
$$\rho = \frac{M}{2 \cdot \pi \cdot R^2} \cdot \frac{1}{r}$$

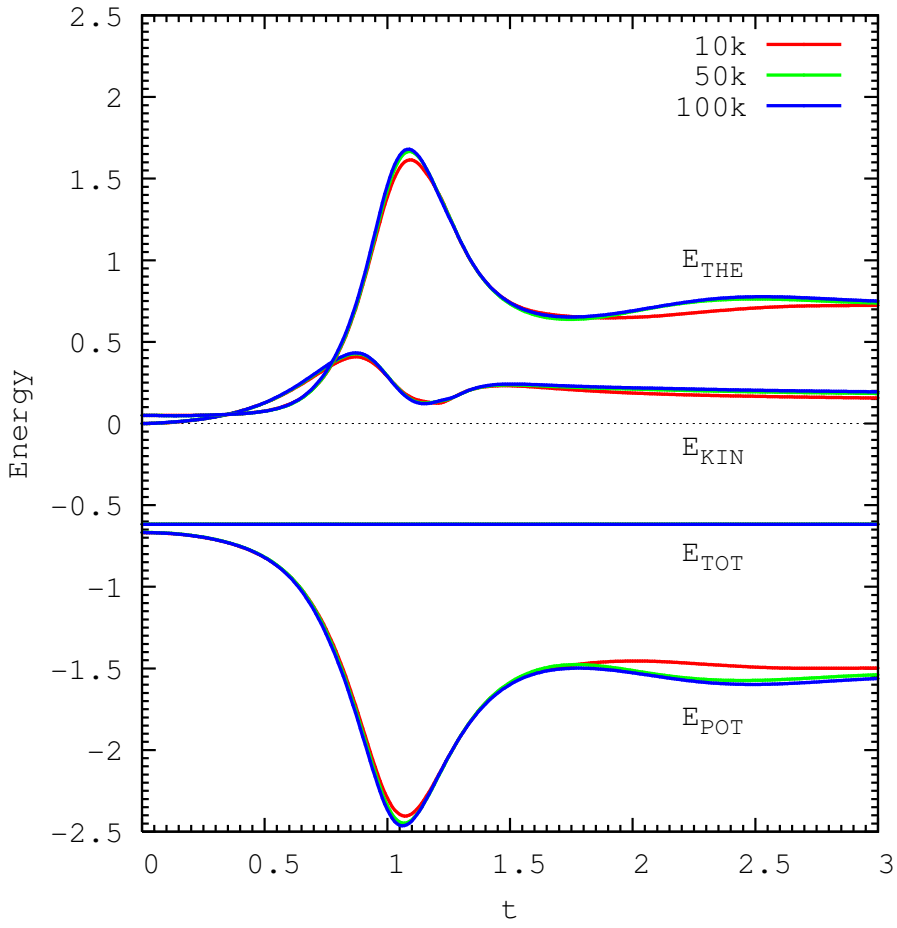
$$E_G = -\frac{2}{3} \cdot \frac{G \cdot M^2}{R}$$

$$G = M = R = 1$$

$$u = 0.05 \cdot \frac{G \cdot M}{R}$$

SPH - test

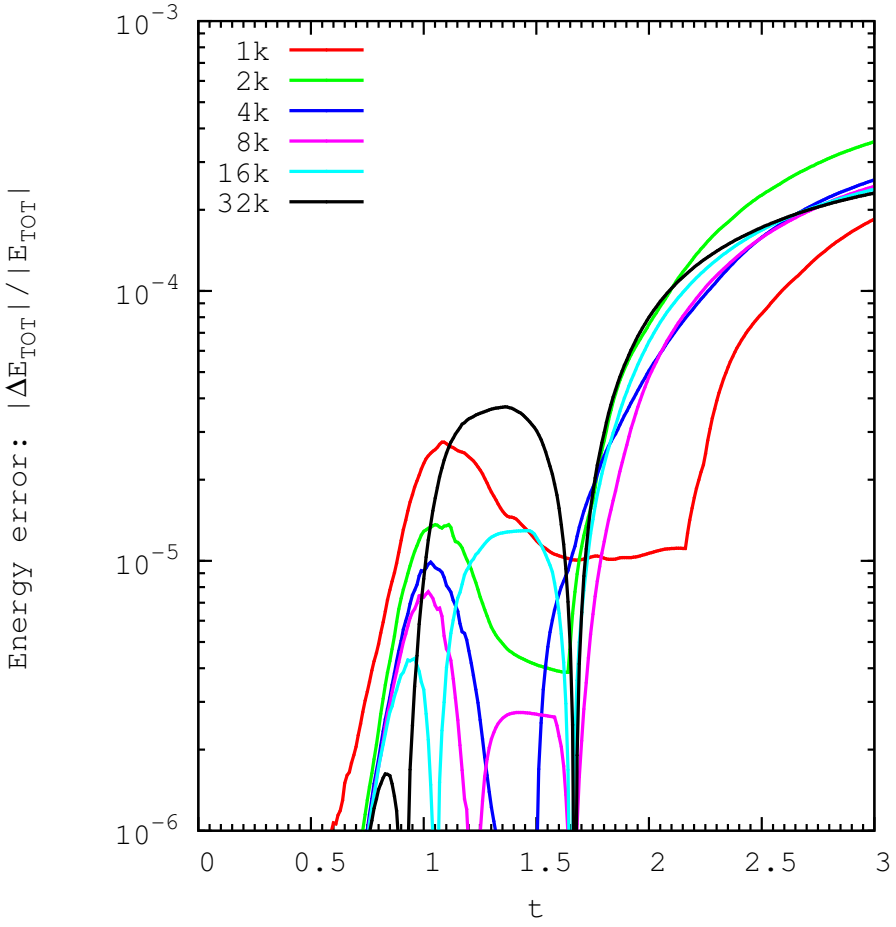


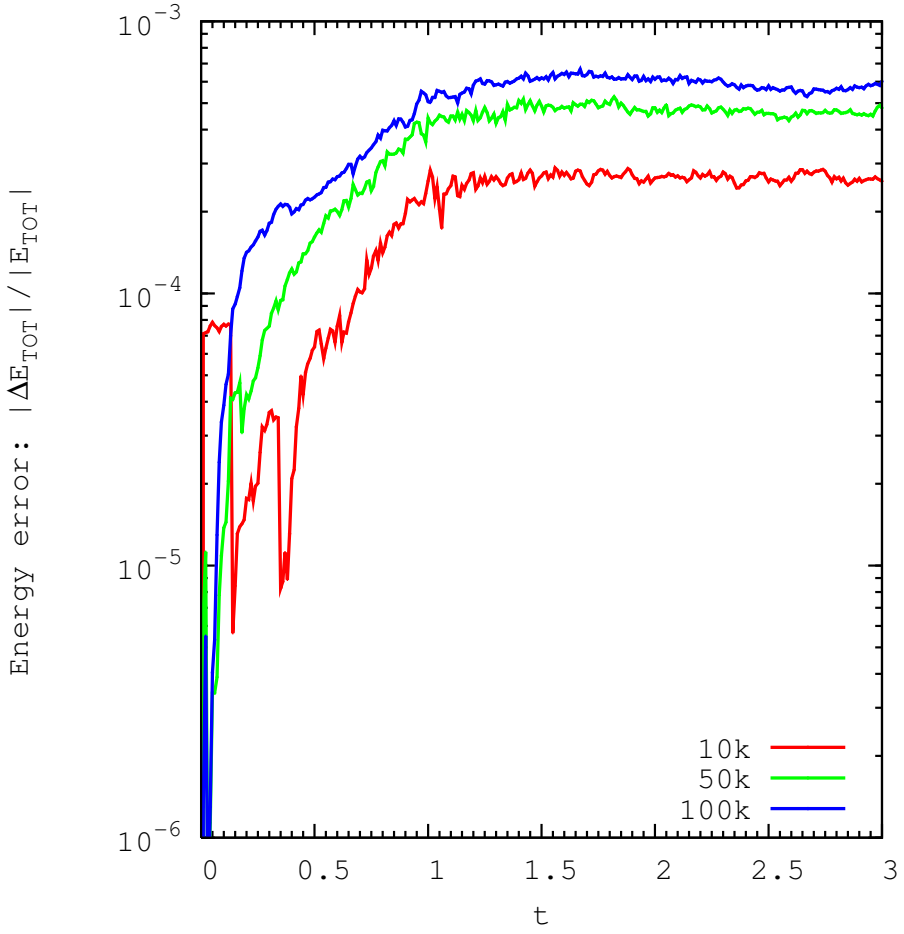


Berczik ($N_{CPU}=1$)

Nakasato ($N_{CPU}=4$)

SPH - test



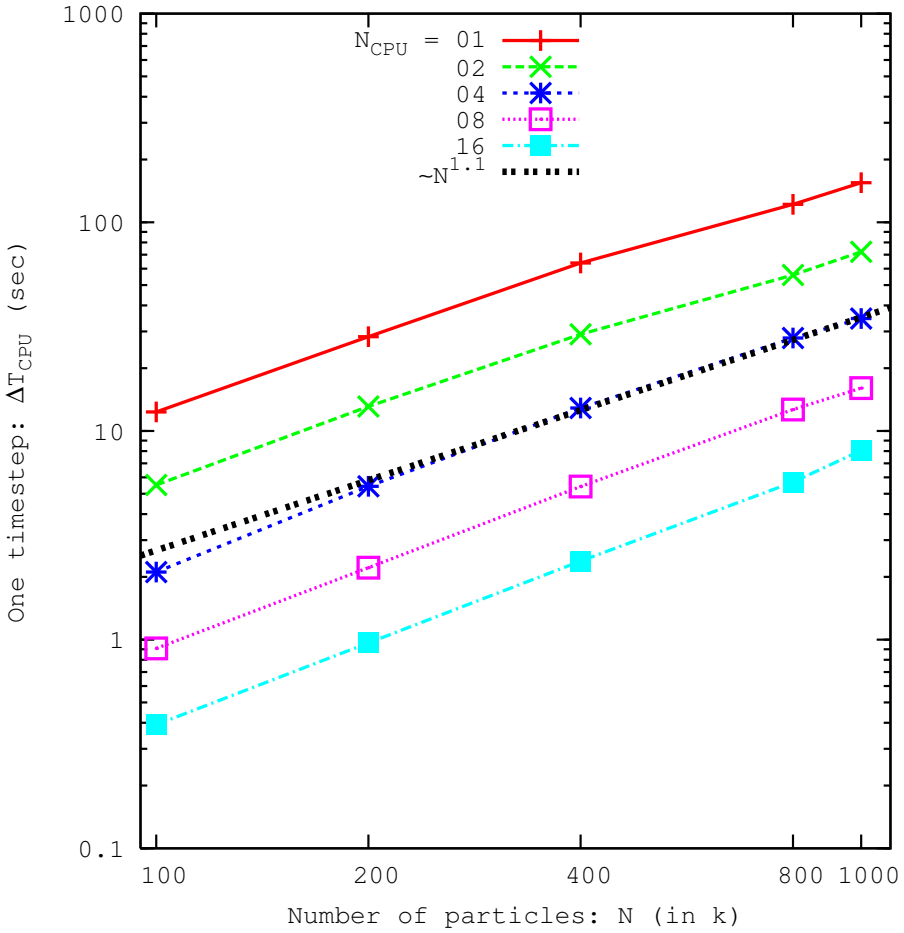


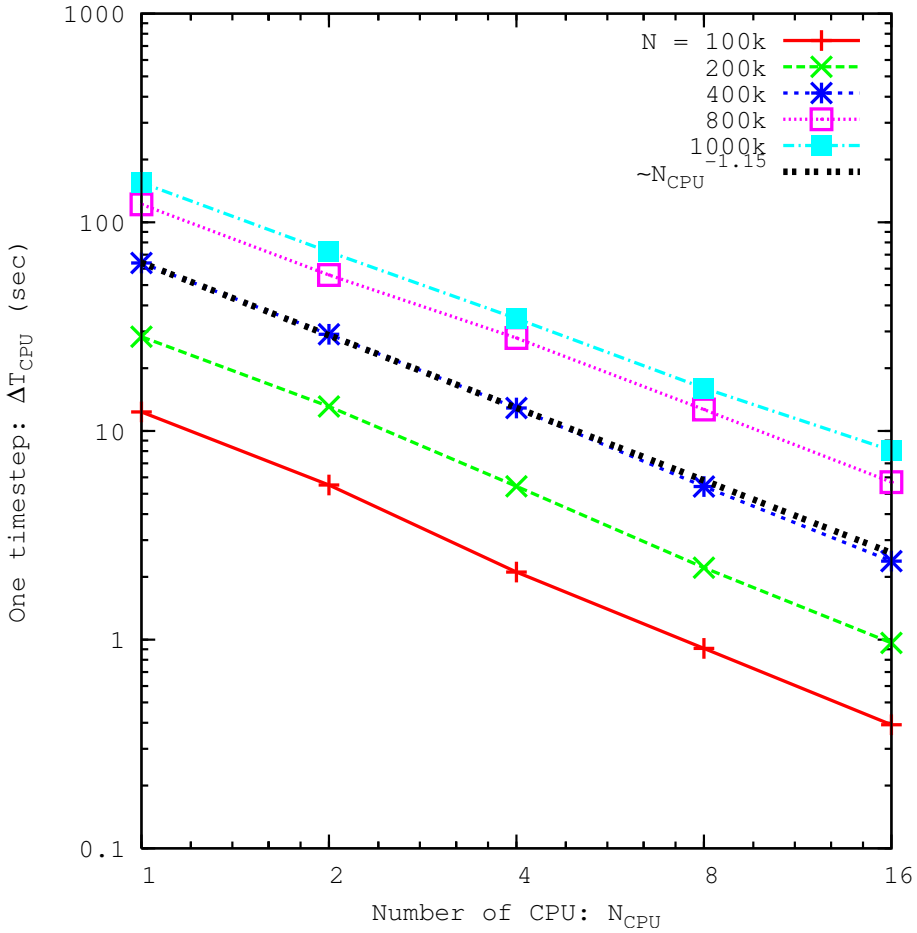
Berczik (N_{CPU}=1)

Nakasato (N_{CPU}=4)

Scaling results

GRAPE + SPH code: One timestep integration





GPU Hardware

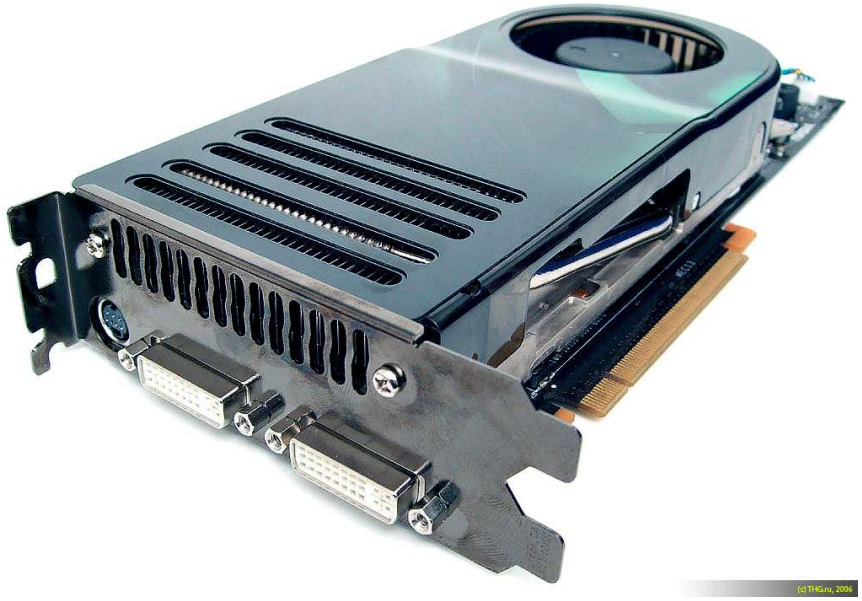
http://www.nvidia.com

http://gpgpu.org

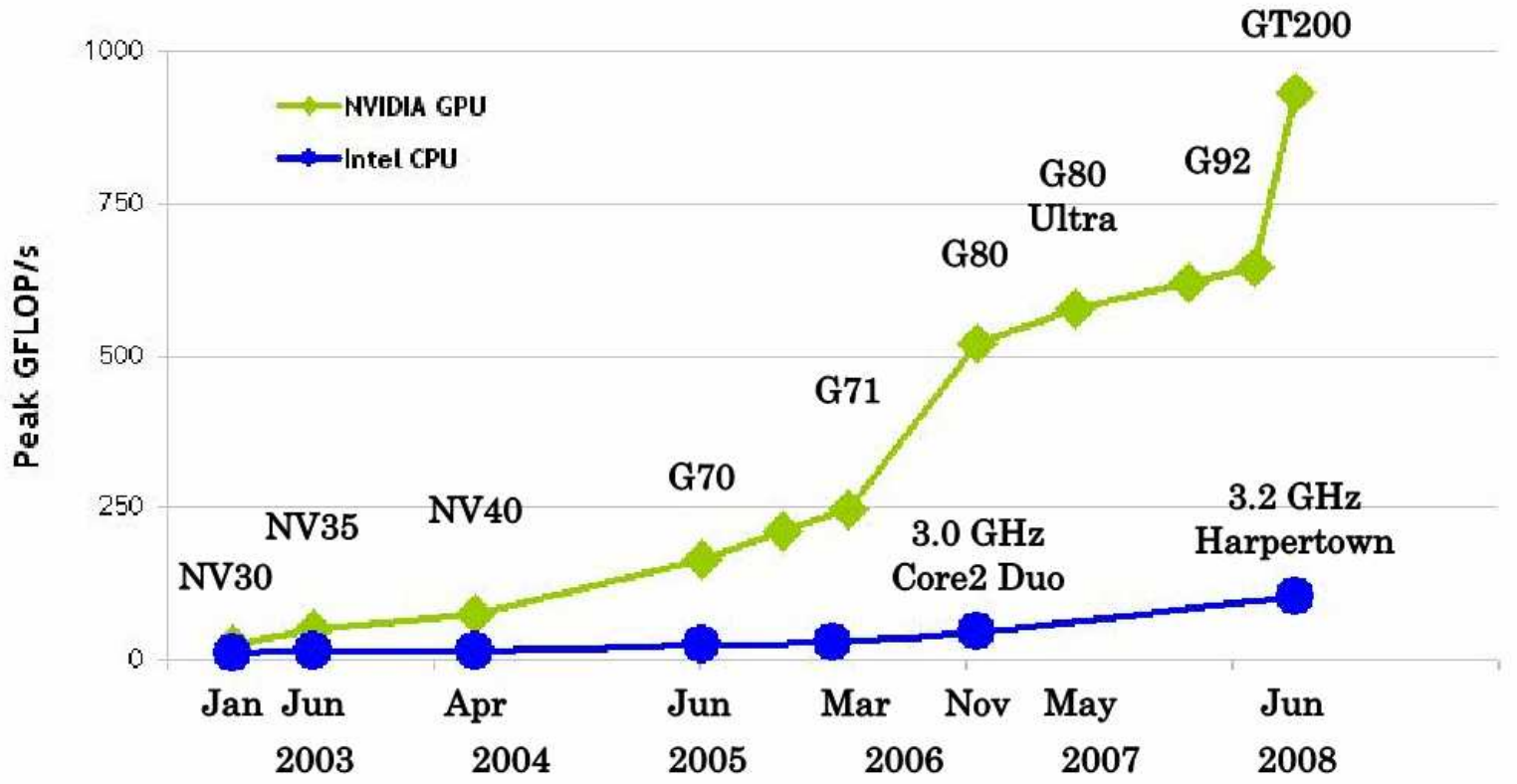
2007…
GeForce 8800 GTX, 128 Stream Proc., 768 MB
GeForce 8800 GTS, 128 Stream Proc., 512 MB
GeForce 8800 GT, 112 Stream Proc., 512 MB

2008…
GeForce 9800 GTX, 128 Stream Proc., 512 MB
GeForce 9800 GX2, 256 Stream Proc., 1 GB
GeForce 9800 GT, 64 Stream Proc., 512 MB

GeForce GTX 280, 240 SP + 30 fp64, 1 GB
GeForce GTX 260, 192 SP + 24 fp64, 890 MB

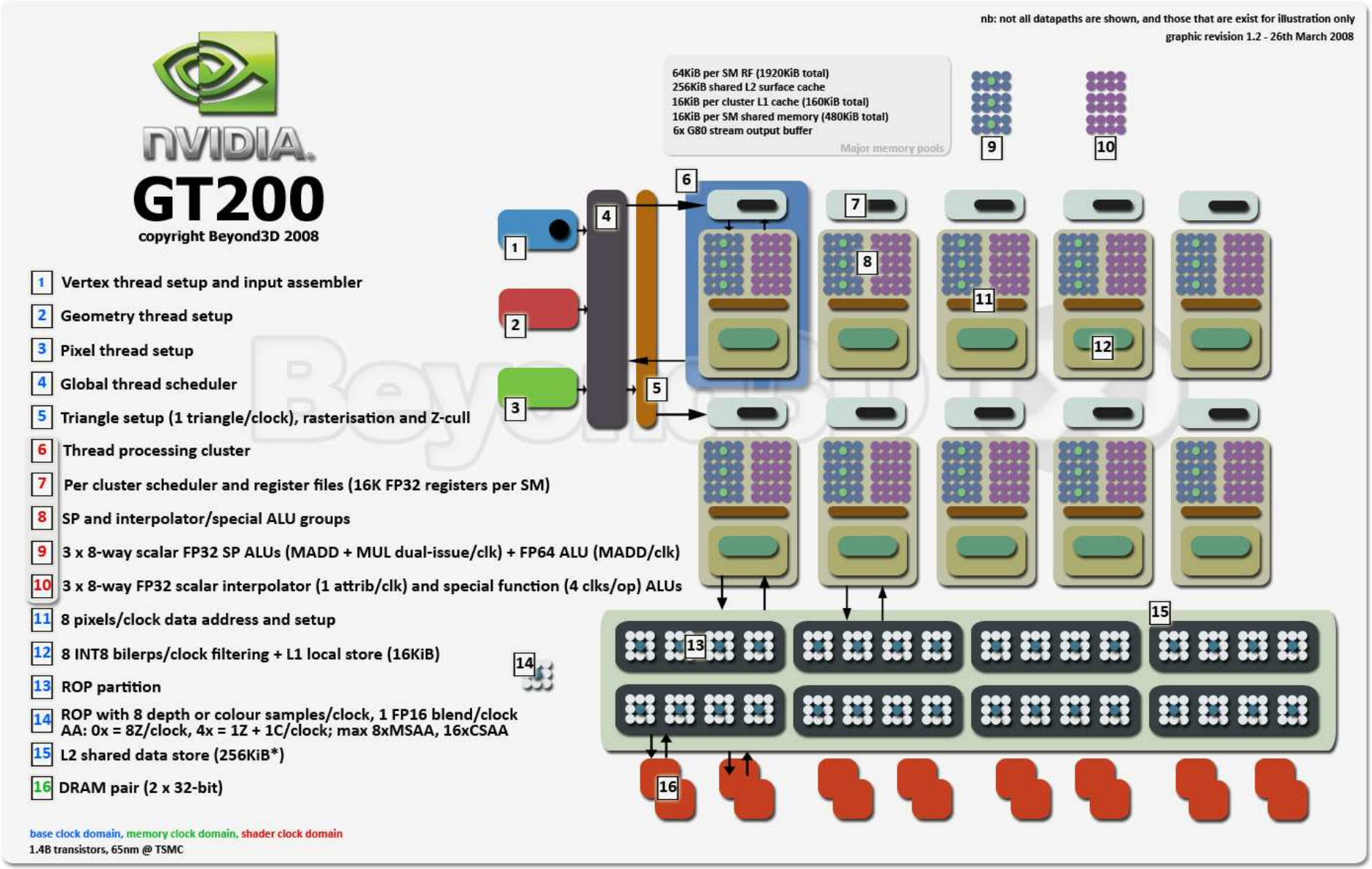


CPU vs. GPU speedup timeline



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

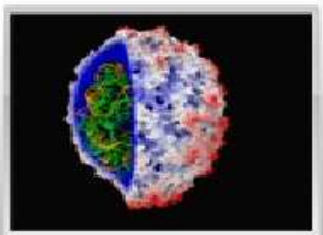
Hardware


nb: not all datapaths are shown, and those that are exist for illustration only
graphic revision 1.2 - 26th March 2008
NVIDIA
GT200
copyright Beyond3D 2008
64KiB per SM RF (1920KiB total)
256KiB shared L2 surface cache
16KiB per cluster L1 cache (160KiB total)
16KiB per SM shared memory (480KiB total)
6x G80 stream output buffer
Major memory pools
1 Vertex thread setup and input assembler
2 Geometry thread setup
3 Pixel thread setup
4 Global thread scheduler
5 Triangle setup (1 triangle/clock), rasterisation and Z-cull
6 Thread processing cluster
7 Per cluster scheduler and register files (16K FP32 registers per SM)
8 SP and interpolator/special ALU groups
9 3 x 8-way scalar FP32 SP ALUs (MADD + MUL dual-issue/clk) + FP64 ALU (MADD/clk)
10 3 x 8-way FP32 scalar interpolator (1 attrib/clk) and special function (4 clks/op) ALUs
11 8 pixels/clock data address and setup
12 8 INT8 bilerps/clock filtering + L1 local store (16KiB)
13 ROP partition
14 ROP with 8 depth or colour samples/clock, 1 FP16 blend/clock
AA: 0x = 8Z/clock, 4x = 1Z + 1C/clock; max 8xMSAA, 16xCSAA
15 L2 shared data store (256KiB*)
16 DRAM pair (2 x 32-bit)
base clock domain, memory clock domain, shader clock domain
1.4B transistors, 65nm @ TSMC

Speedups using GPU vs. CPU

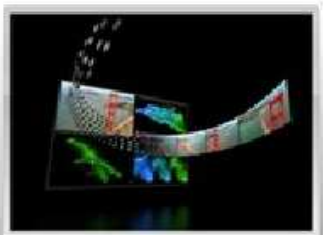
146X

Interactive visualization of volumetric white matter connectivity[1]



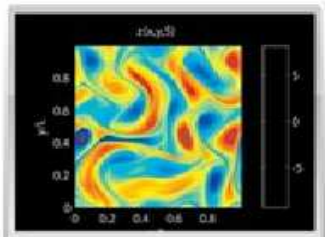
36X

Ionic placement for molecular dynamics simulation on GPU[2]



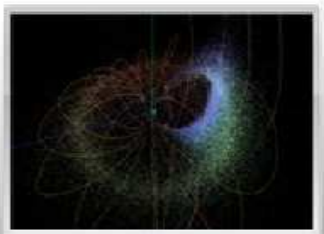
18X

Transcoding HD video stream to H.264 for portable video[3]



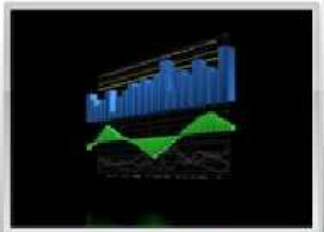
17X

Simulation in Matlab using .mex file CUDA function[4]



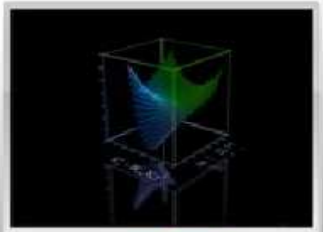
100X

Astrophysics N-body simulation[5]



149X

Financial simulation of LIBOR model with swaptions[6]



47X

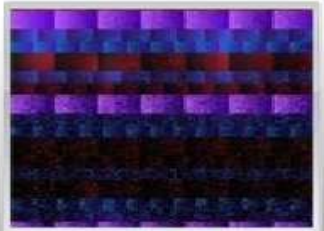
GLAME@lab: M-script API for linear Algebra operations on GPU[7]

20X

Ultrasound medical imaging for cancer diagnostics[8]

24X

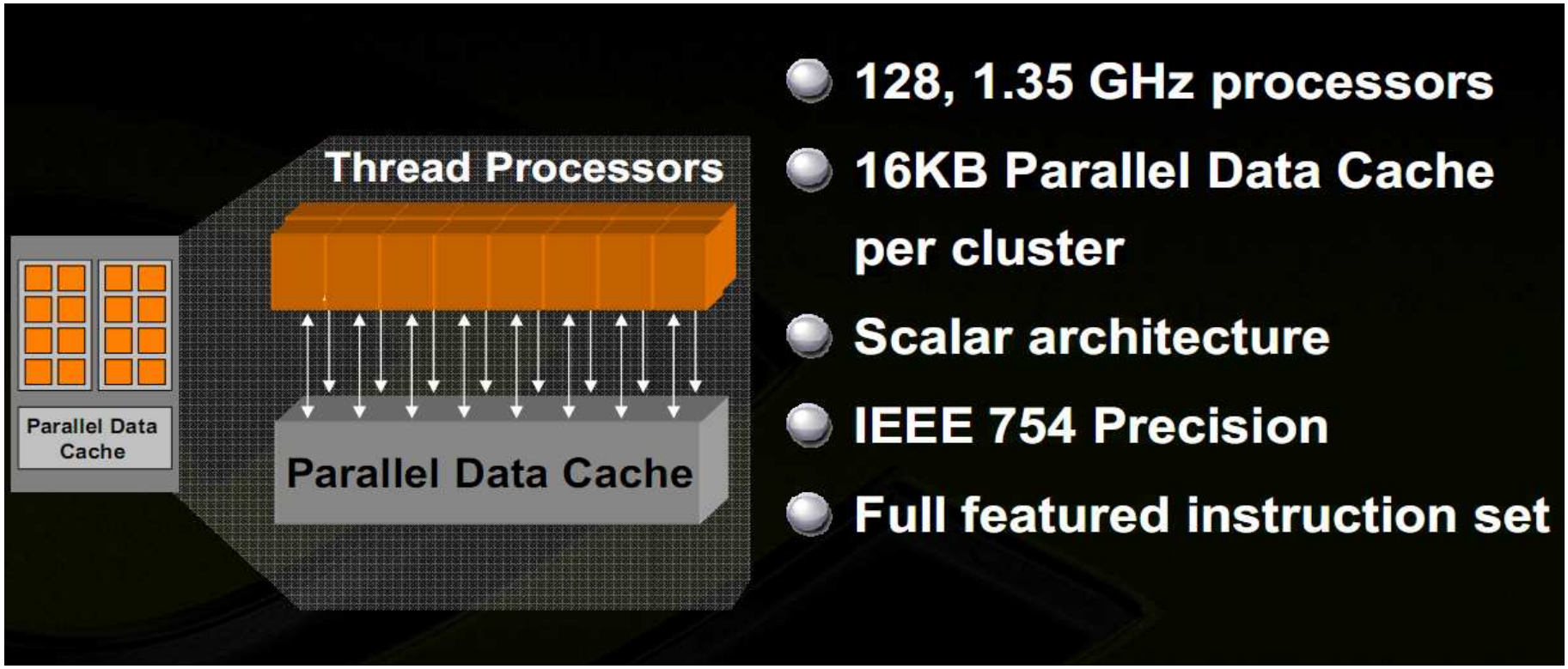
Highly optimized object oriented molecular dynamics[9]



30X

Cmatch exact string matching - find similar proteins & gene sequences[10]

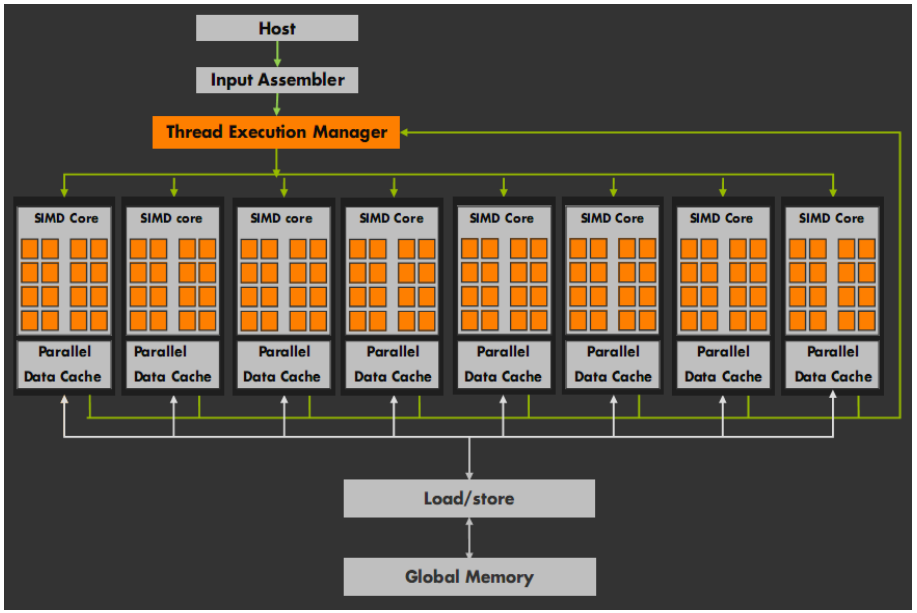
Hardware

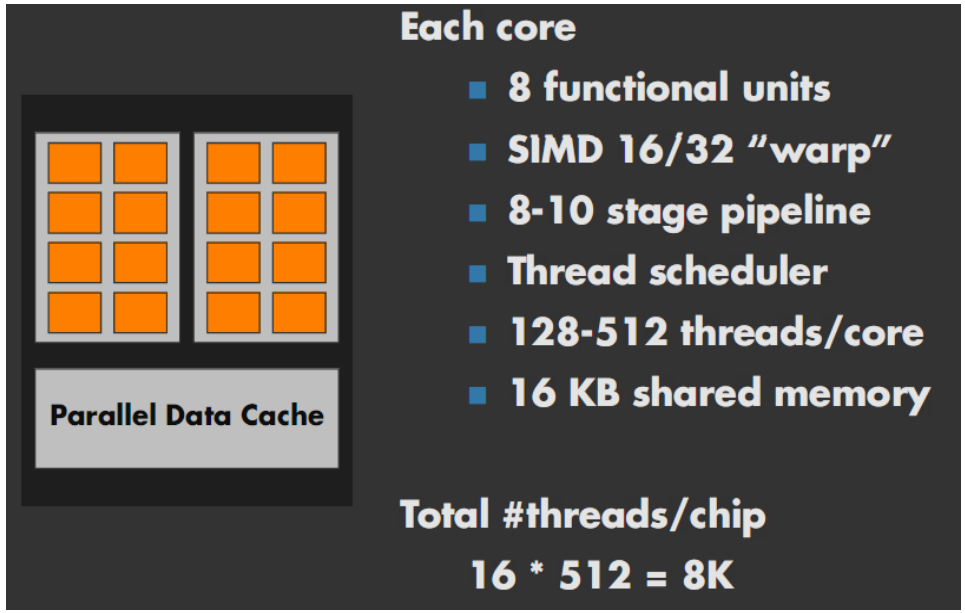


GeForce 8800 GTX:

575 MHz * 128 processors * 2 flop/inst * 2 inst/clock = 333 Gflops

Hardware

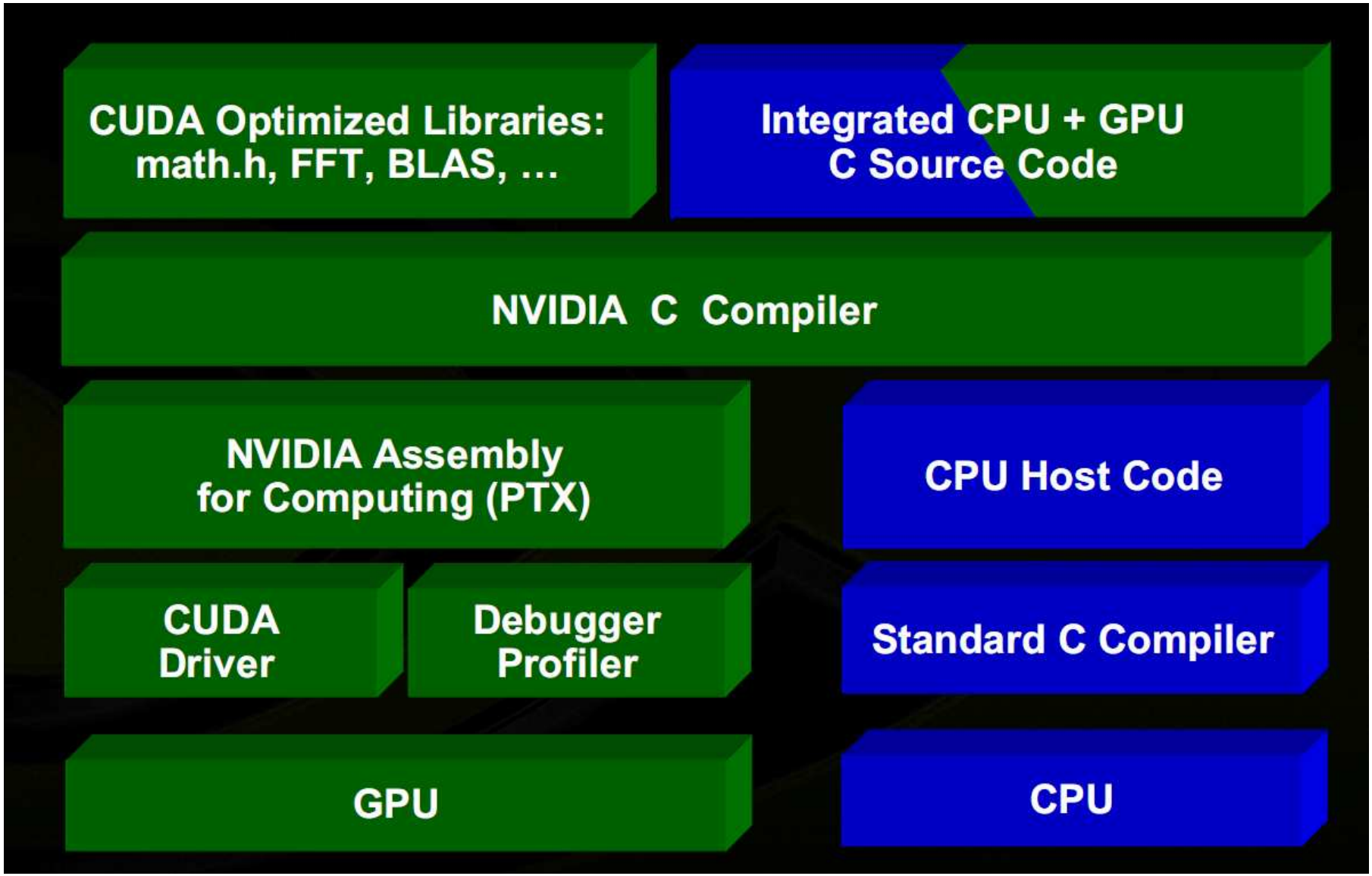




GeForce 8800 GTX:

575 MHz * 128 processors * 2 flop/inst * 2 inst/clock = 333 Gflops

CUDA


CUDA Optimized Libraries:
math.h, FFT, BLAS, …
Integrated CPU + GPU
C Source Code
NVIDIA C Compiler
NVIDIA Assembly
for Computing (PTX)
CPU Host Code
CUDA
Driver
Debugger
Profiler
Standard C Compiler
GPU
CPU

Simple CUDA example

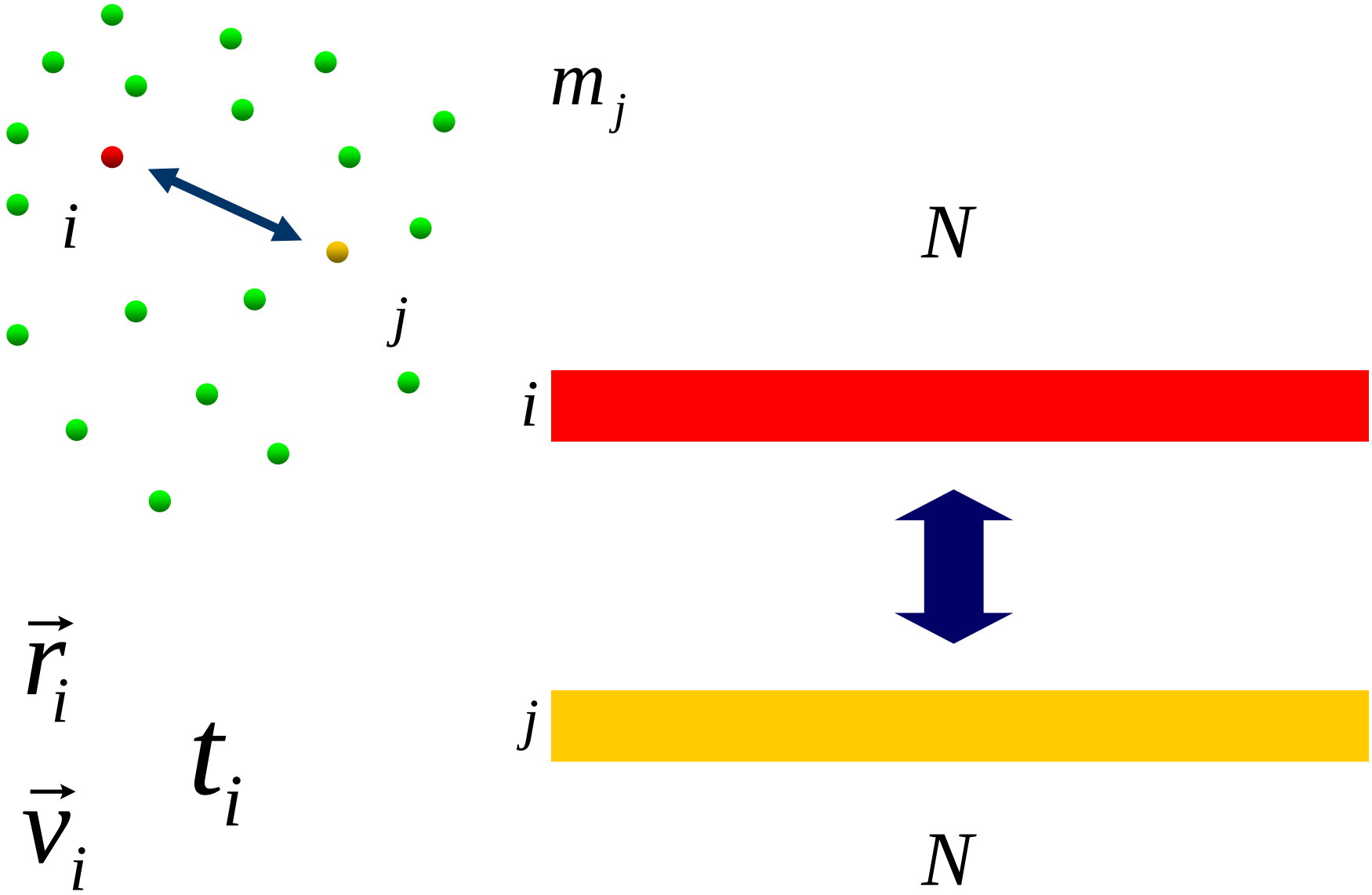
CPU C program

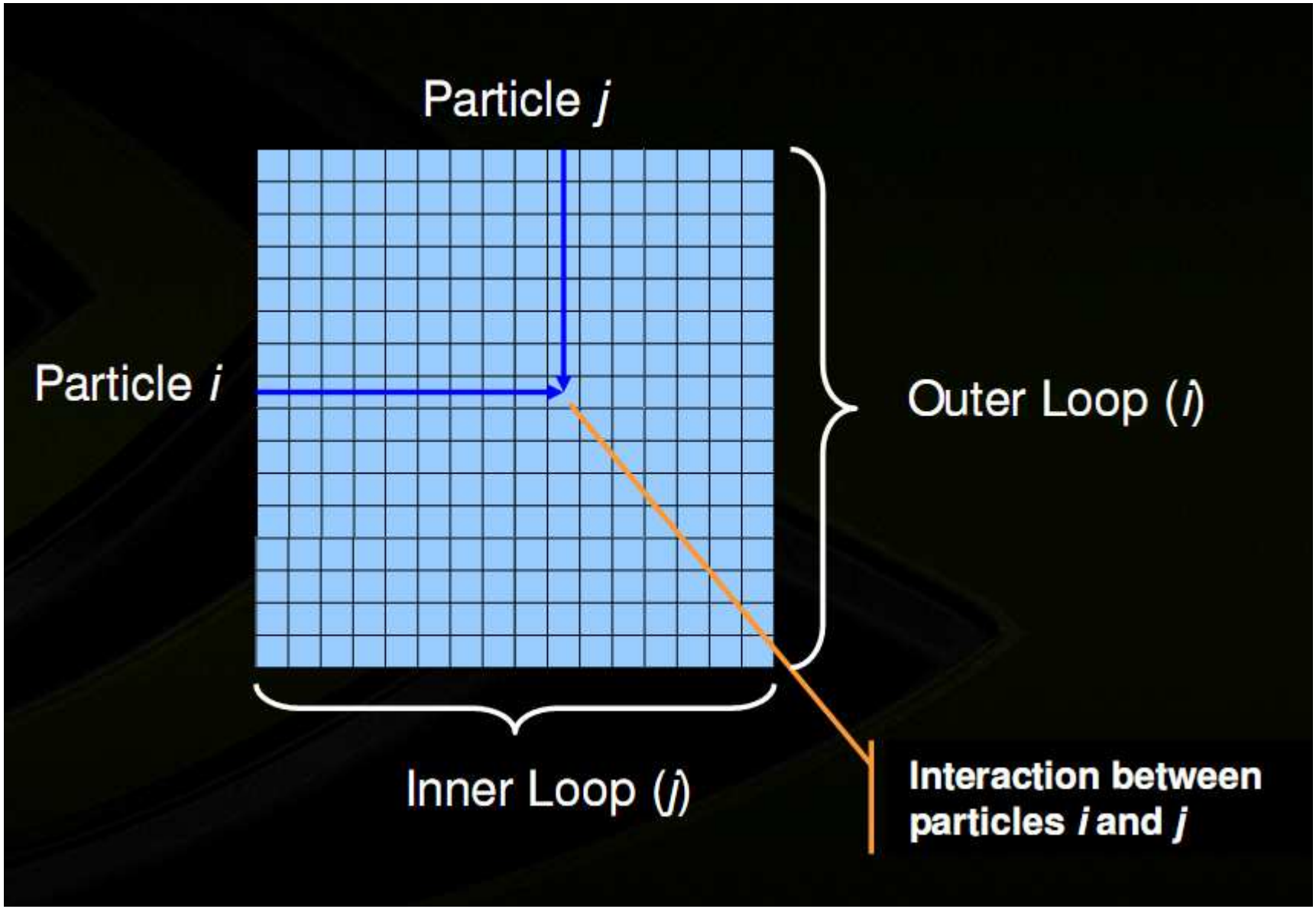
```
void addMatrix(float *a, float *b,
               float *c, int N)
{
  int i, j, index;
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      index = i + j * N;
      c[index]=a[index] + b[index];
    }
  }
}

void main()
{
  .....
  addMatrix(a, b, c, N);
}
```

CUDA C program

```
__global__ void addMatrix(float *a,float *b,
                          float *c, int N)
{
  int i=blockIdx.x*blockDim.x+threadIdx.x;
  int j=blockIdx.y*blockDim.y+threadIdx.y;
  int index = i + j * N;
  if ( i < N && j < N)
    c[index]= a[index] + b[index];
}

void main()
{
  ..... // allocate & transfer data to GPU
  dim3 dimBlk (blocksize, blocksize);
  dim3 dimGrd (N/dimBlk.x, N/dimBlk.y);
  addMatrix<<<dimGrd,dimBlk>>>(a, b, c,N);
}
```

Basic idea of any N-body code

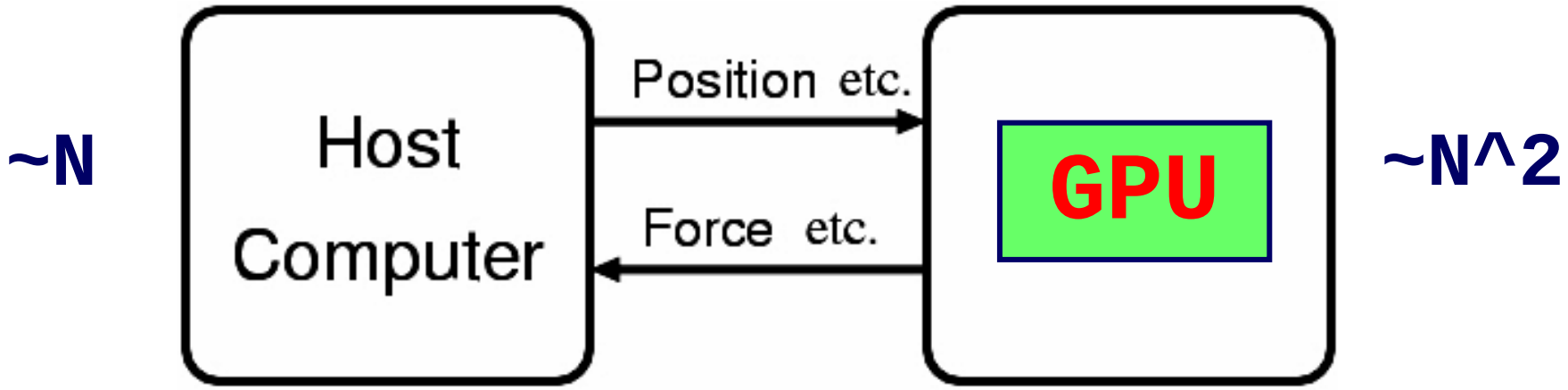



Particle *j*
Particle *i*
Outer Loop (*i*)
Inner Loop (*j*)
Interaction between particles *i* and *j*

```
__device__ float3
bodyBodyInteraction(float3 ai, float4 bi, float4 bj) {
    float3 r;
    r.x = bi.x - bj.x;              // r_ij [3 FLOPS]
    r.y = bi.y - bj.y;
    r.z = bi.z - bj.z;

    // distSqr = dot(r_ij, r_ij) + EPS^2   [6 FLOPS]
    float distSqr = r.x * r.x + r.y * r.y + r.z * r.z;
    distSqr += softeningSquared;

    // invDistCube =1/distSqr^(3/2)        [4 FLOPS (2 mul, 1 sqrt, 1 inv)]
    float distSixth = distSqr * distSqr * distSqr;
    float invDistCube = 1.0f / sqrtf(distSixth);

    float s = bj.w * invDistCube;   // s = m_j * invDistCube [1 FLOP]

    ai.x += r.x * s;                // a_i =  a_i + s * r_ij [6 FLOPS]
    ai.y += r.y * s;
    ai.z += r.z * s;
    return ai;
}
```

Total: 20 FLOPS

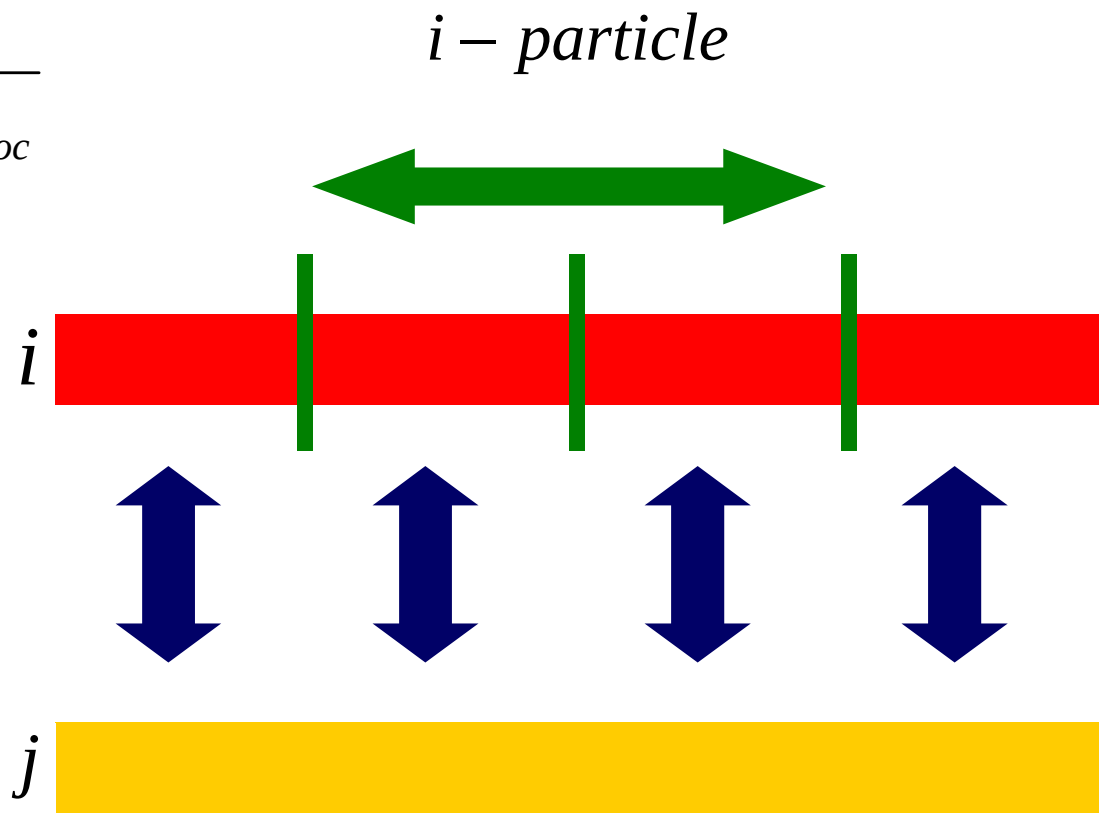
Basic idea of GRAPE/GPU N-body code



$$\vec{a}_i = \sum_{j=1;\ j\neq i}^{N} \vec{f}_{ij} \qquad \vec{f}_{ij} = -\frac{G \cdot m_j}{(r_{ij}^2 + \varepsilon^2)^{3/2}} \vec{r}_{ij}$$

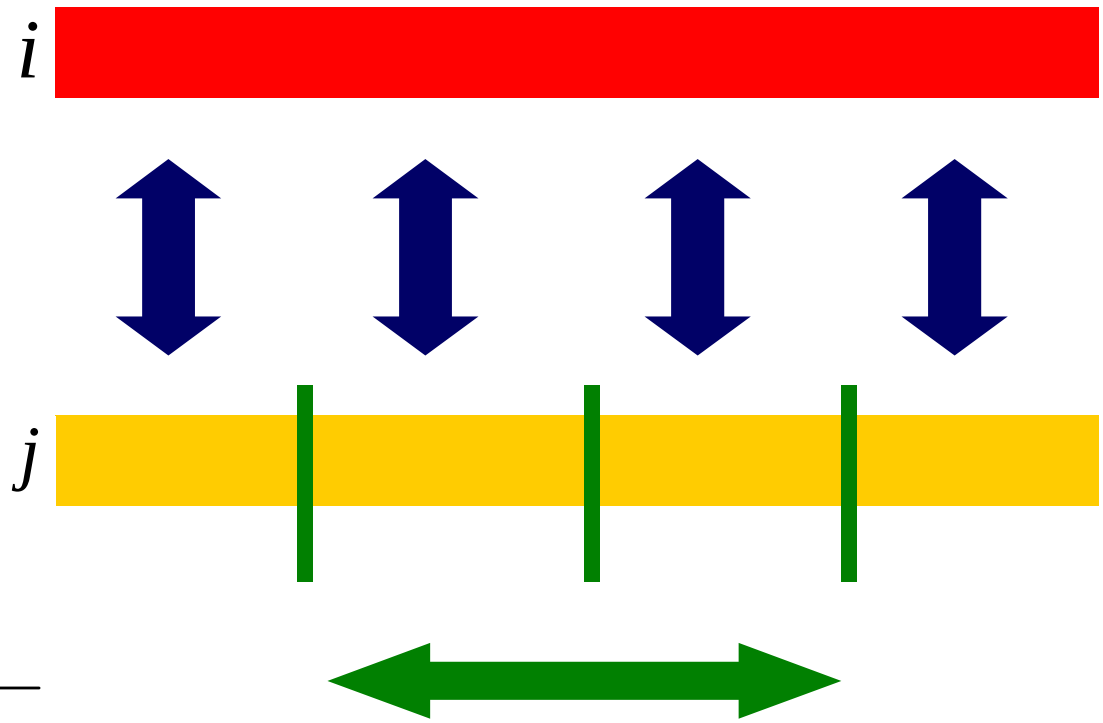
Basic idea of any parallel N-body code

$$N_{loc} = \frac{N}{N_{proc}}$$



Basic idea of any parallel N-body code

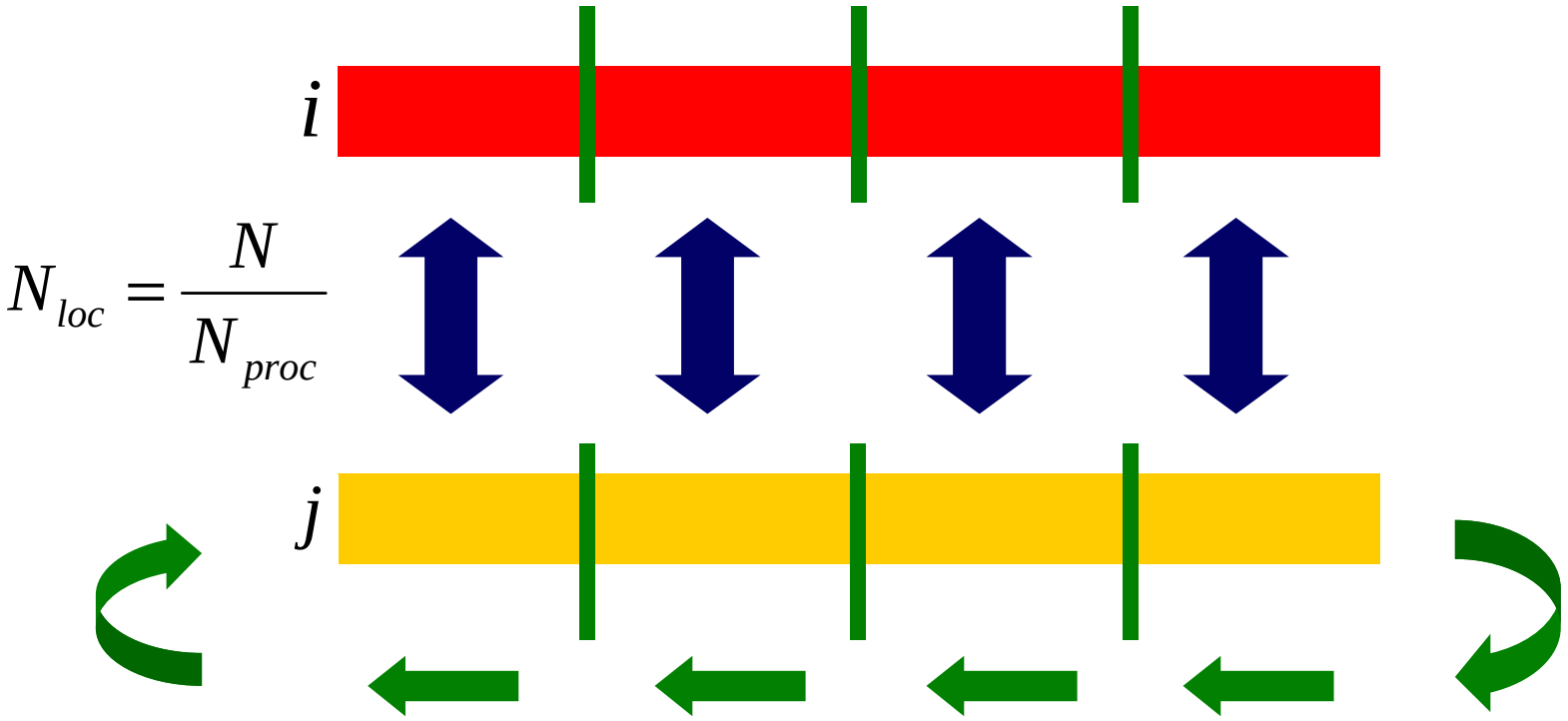
$j - particle$



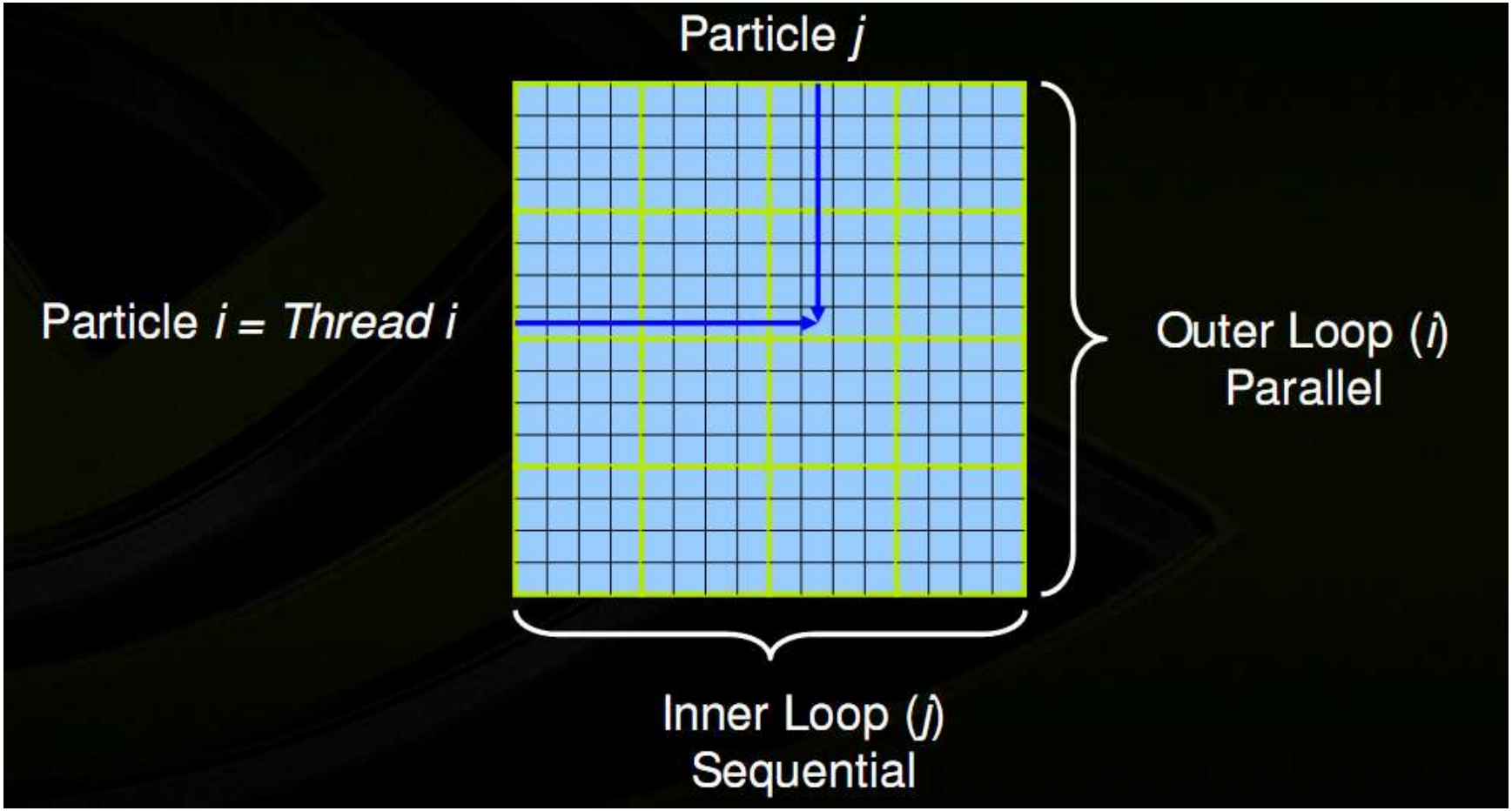
$$N_{loc} = \frac{N}{N_{proc}}$$

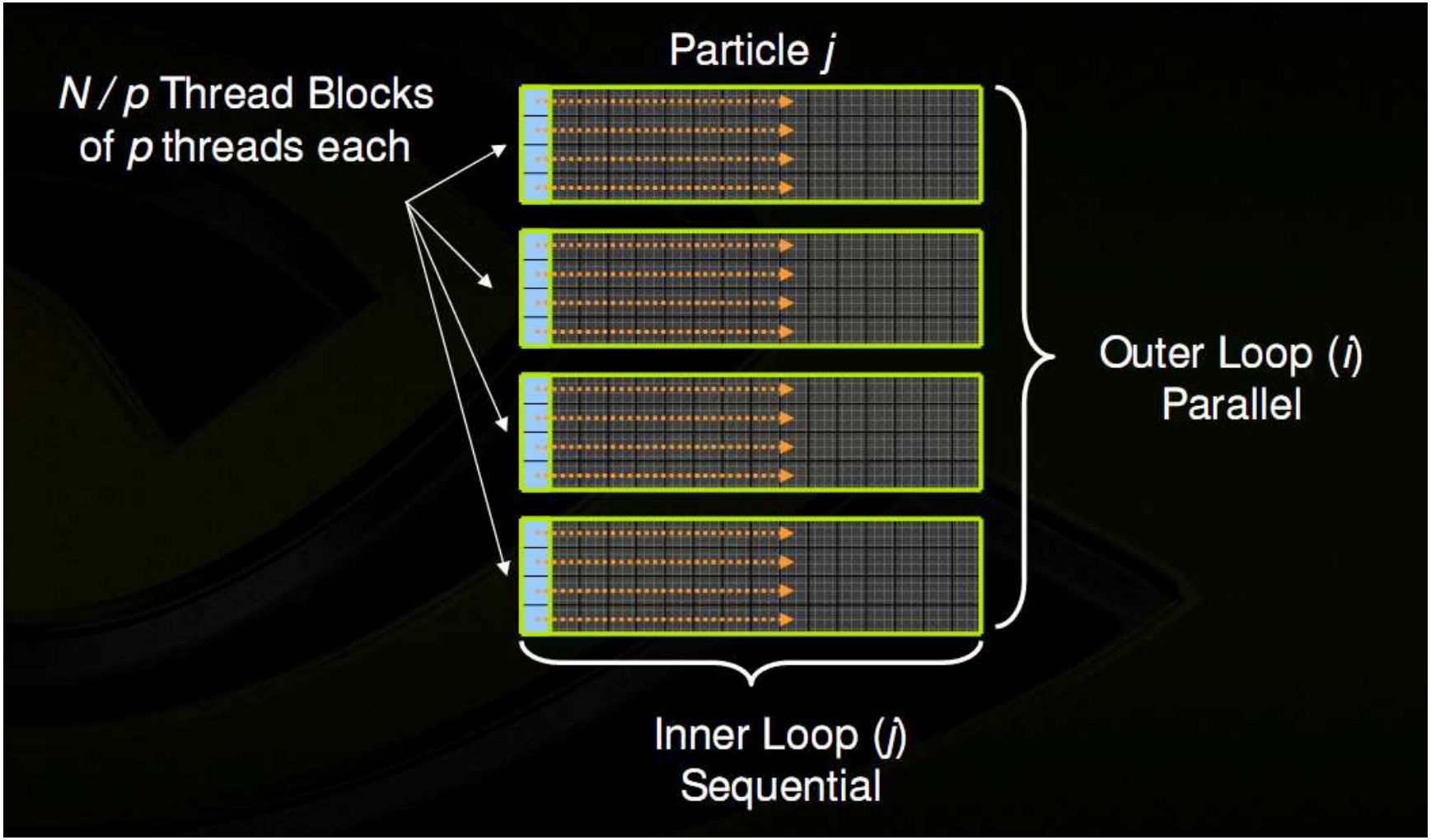
Basic idea of any parallel N-body code

$i, j - particle$



Some communication scheme...

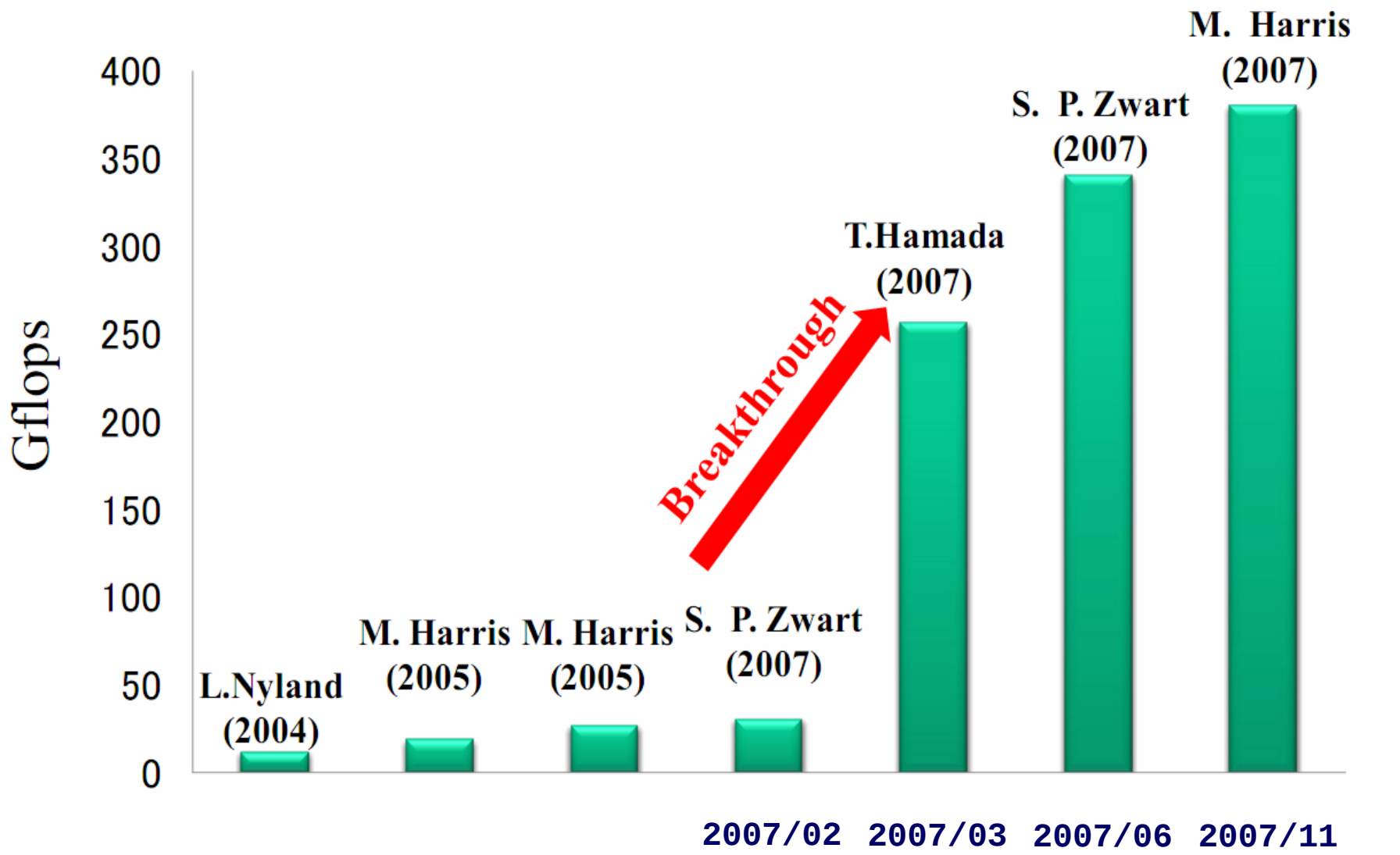

Particle *j*
Particle *i* = *Thread i*
Outer Loop (*i*)
Parallel
Inner Loop (*j*)
Sequential


Particle *j*
N / *p* Thread Blocks
of *p* threads each
Outer Loop (*i*)
Parallel
Inner Loop (*j*)
Sequential

```
forall bodies i in parallel {
  accel = 0;
  pos = position[i]
  foreach tile q {
    forall threads p in thread block in parallel {
      shared[p] = position[q*tile_size + p]
    }
    synchronize threads in block
    foreach body j in tile q {
      accel +=
        computeAcceleration(pos, position[j])
    }
    synchronize threads in block
  }
}
```

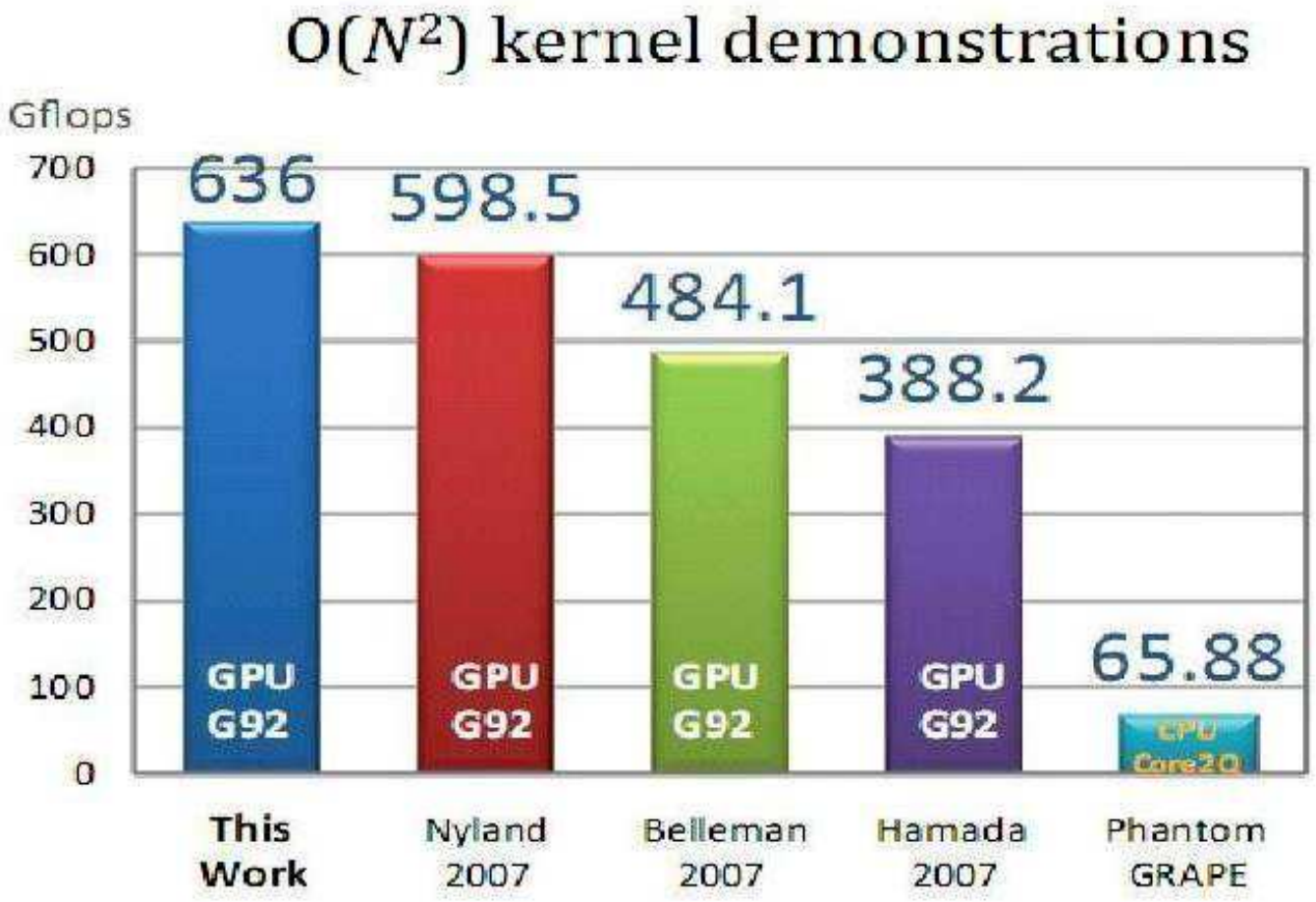
GPU N-body speedup timeline

All on a same GPU: 8800 GTX (G80)



GPU N-body gravity

Hamada et al. 2008: Direct GPU code



GRAPE6a

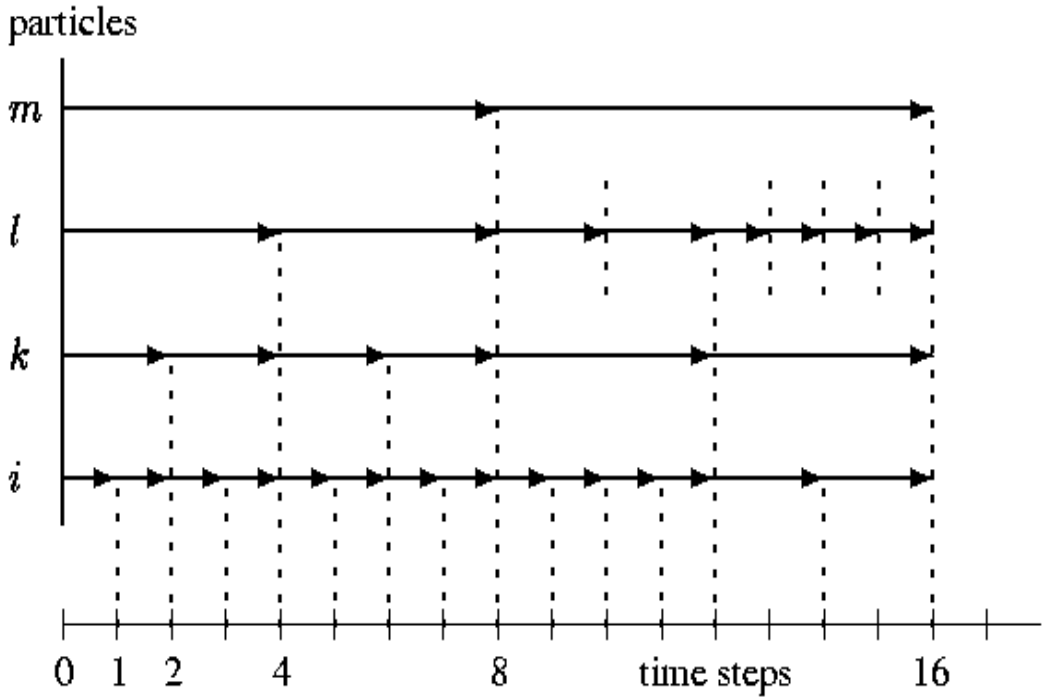
~100

Results for 8800 GTS 512MB (G92)

Our own φGRAPE/GPU N-body code

Harfst et al, NewA, 12, 357 (2007) [astro-ph/0608125]

Hierarchical Individual Block Time Steps



$$\Delta t = \sqrt{\eta \frac{|\vec{a}||\vec{a}^{(2)}| + |\dot{\vec{a}}|^2}{|\dot{\vec{a}}||\vec{a}^{(3)}| + |\vec{a}^{(2)}|^2}} \ .$$

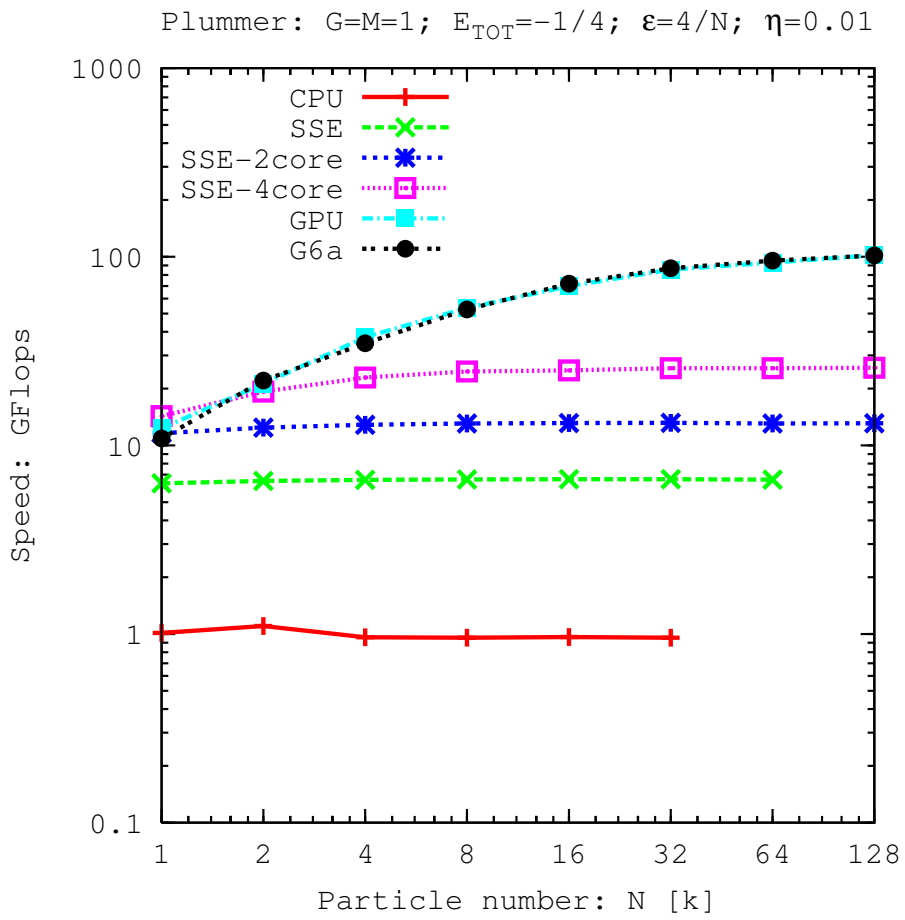
4th order Hermite scheme

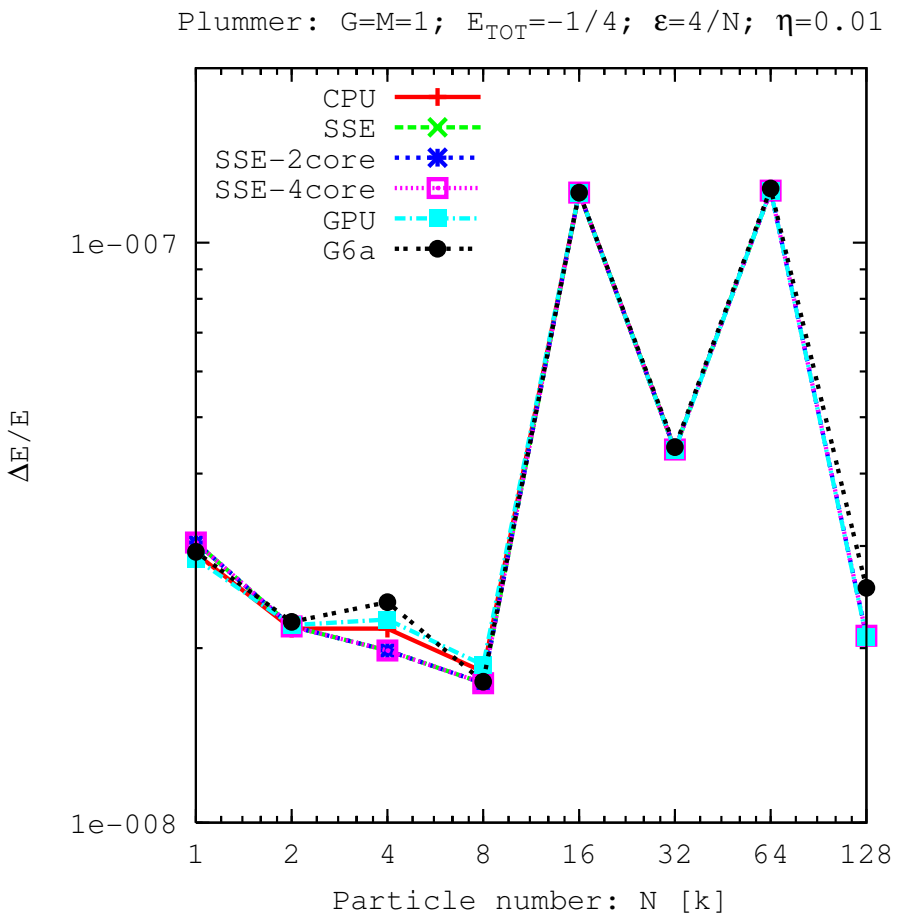
$$\frac{d^2\vec{r}_i}{dt^2} = \vec{a}_i$$

ftp://ftp.ari.uni-heidelberg.de/pub/staff/berczik/phi-GRAPE/

GPU Hermite 4th results

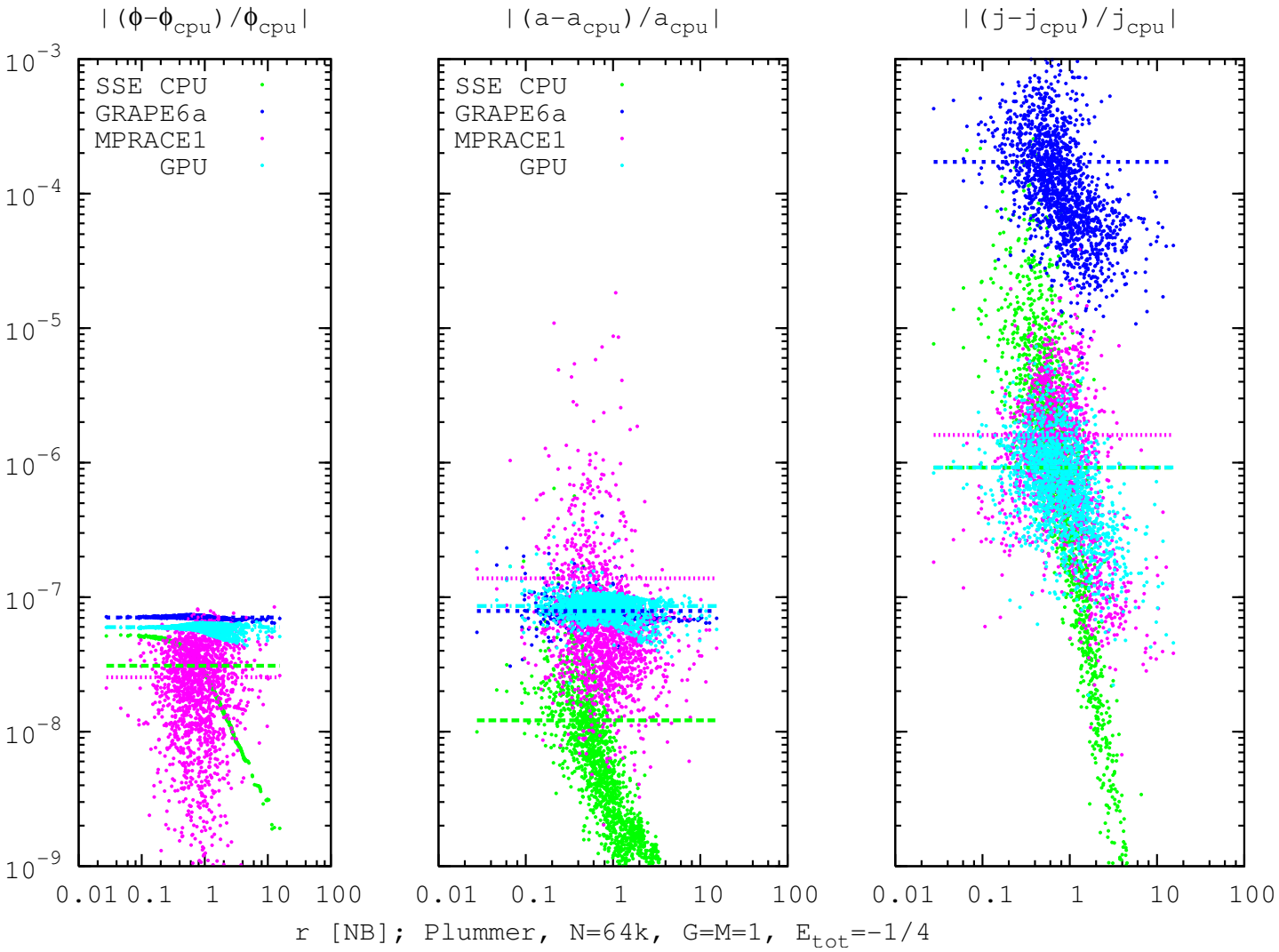
Nitadori, Berczik et al. 2007.11



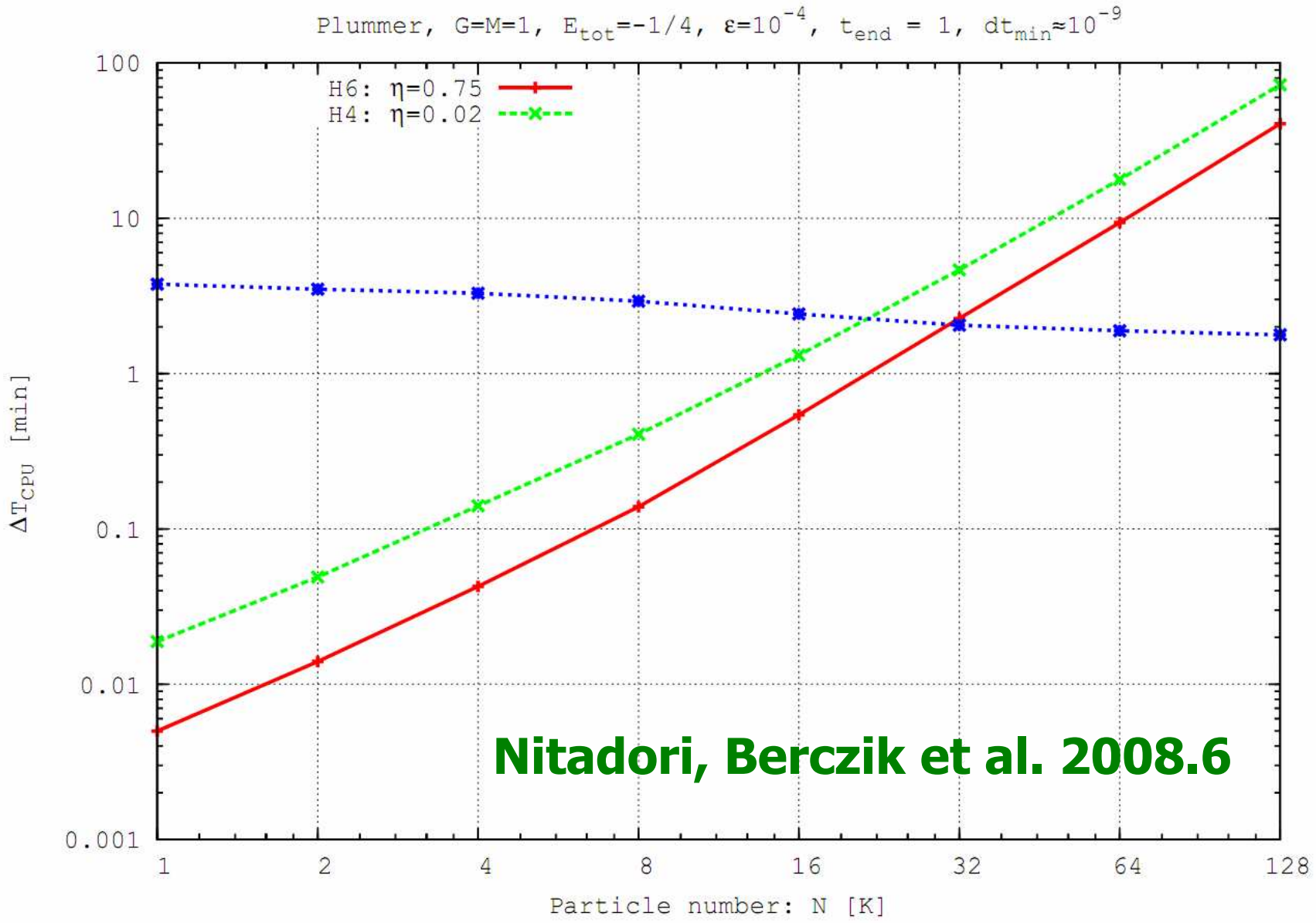


GPU Hermite 4th results

Nitadori, Berczik et al. 2007.11

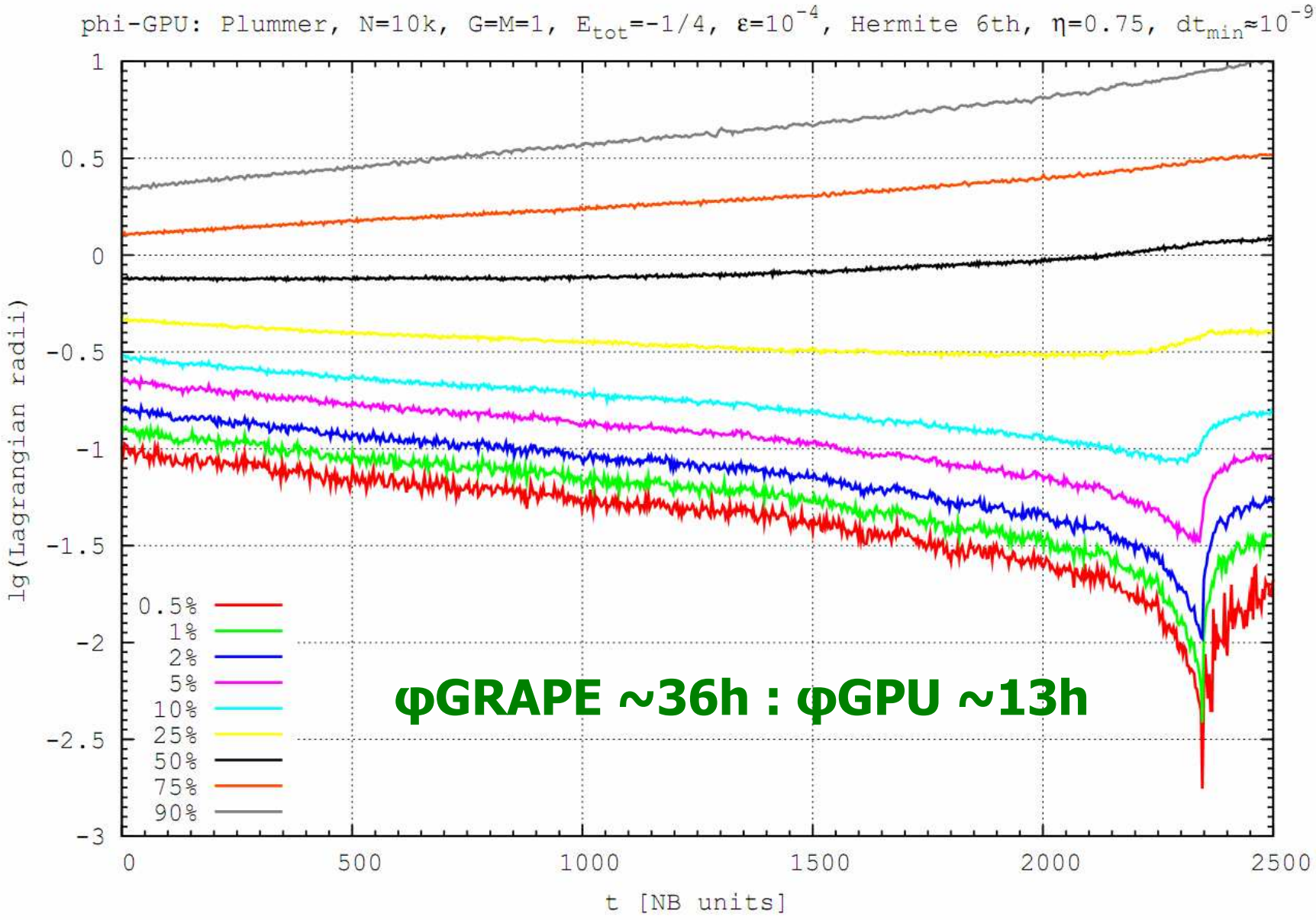


Hermite 4th vs. 6th results

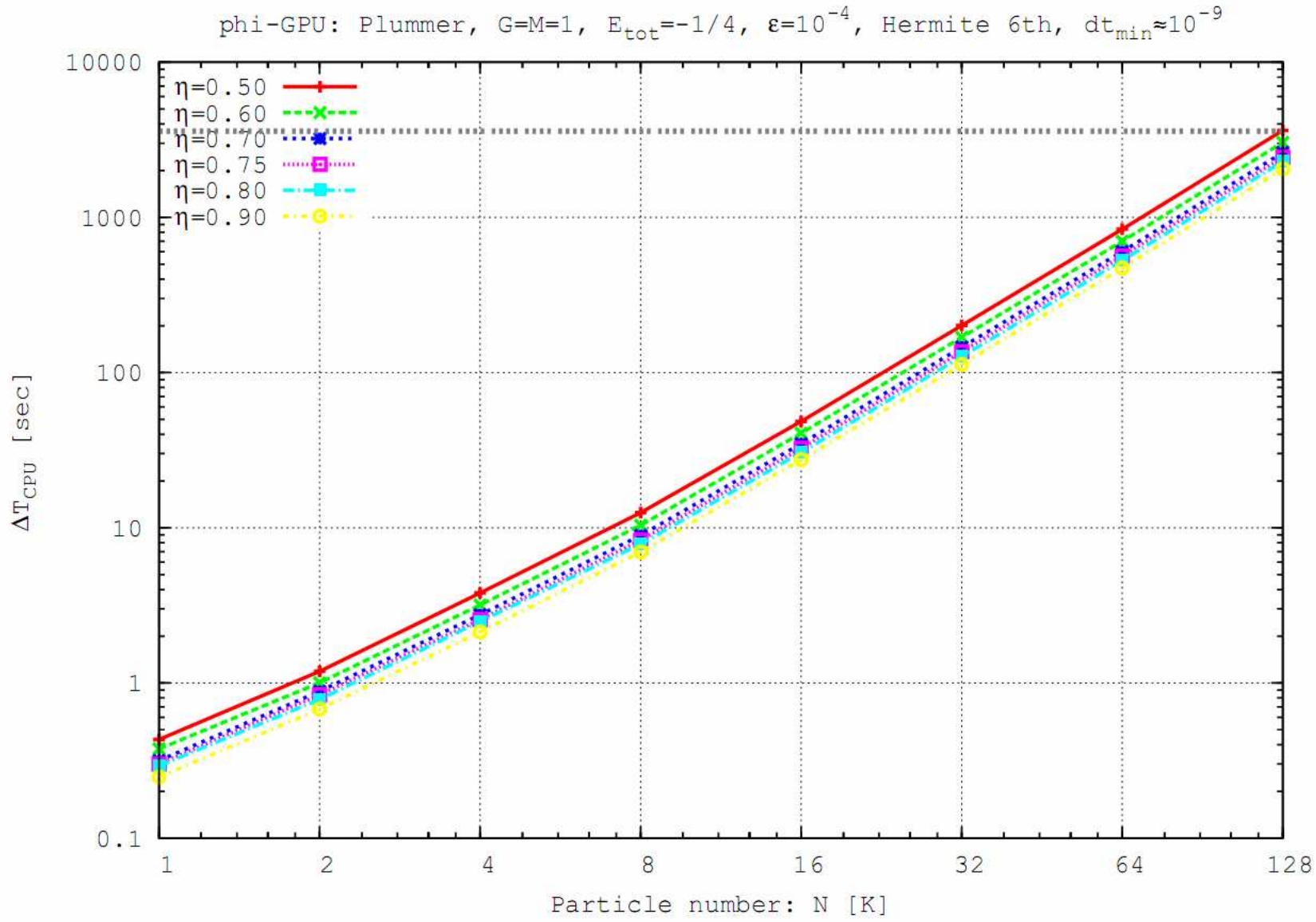


Nitadori, Berczik et al. 2008.6

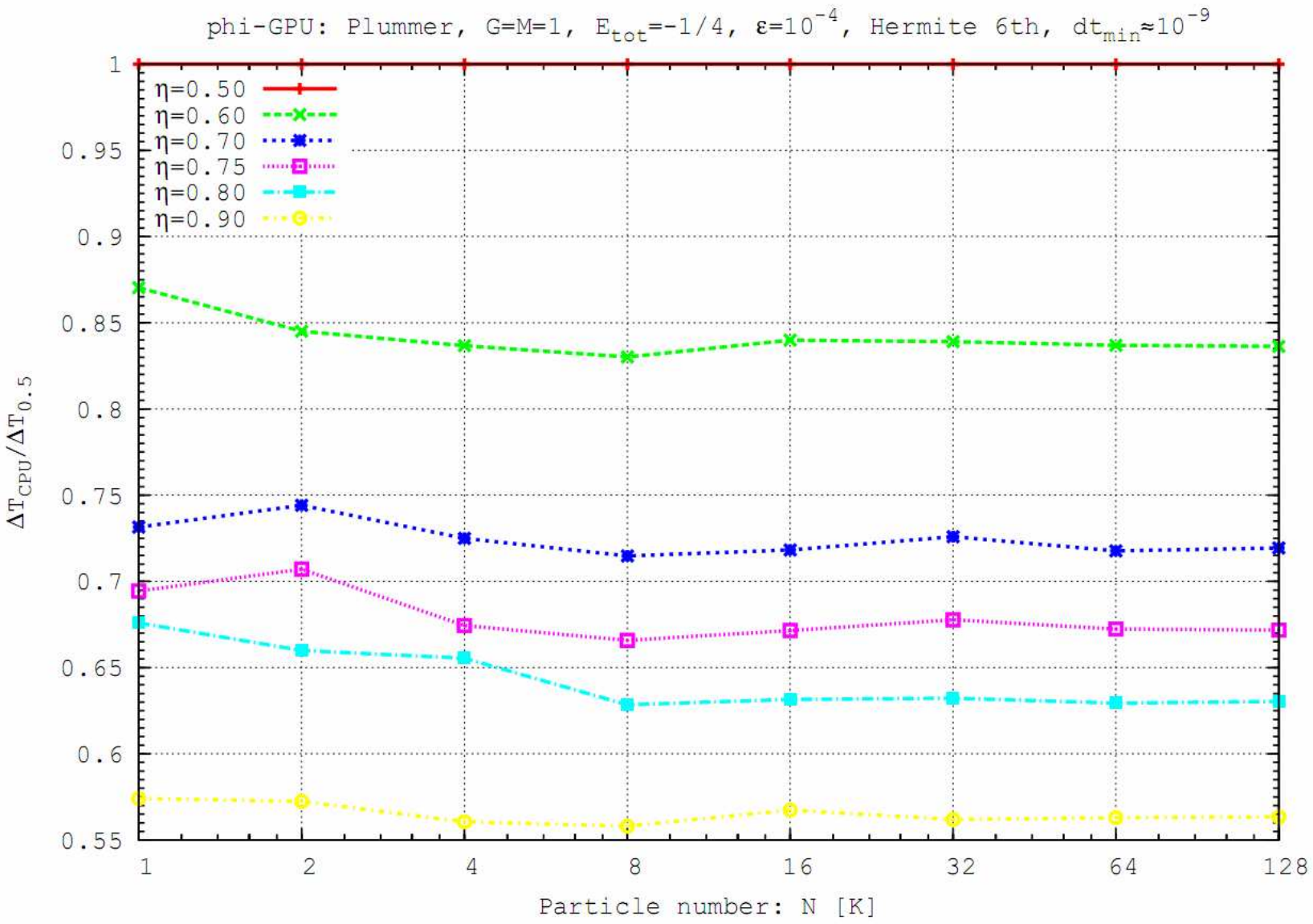
φGPU Hermite 6th results



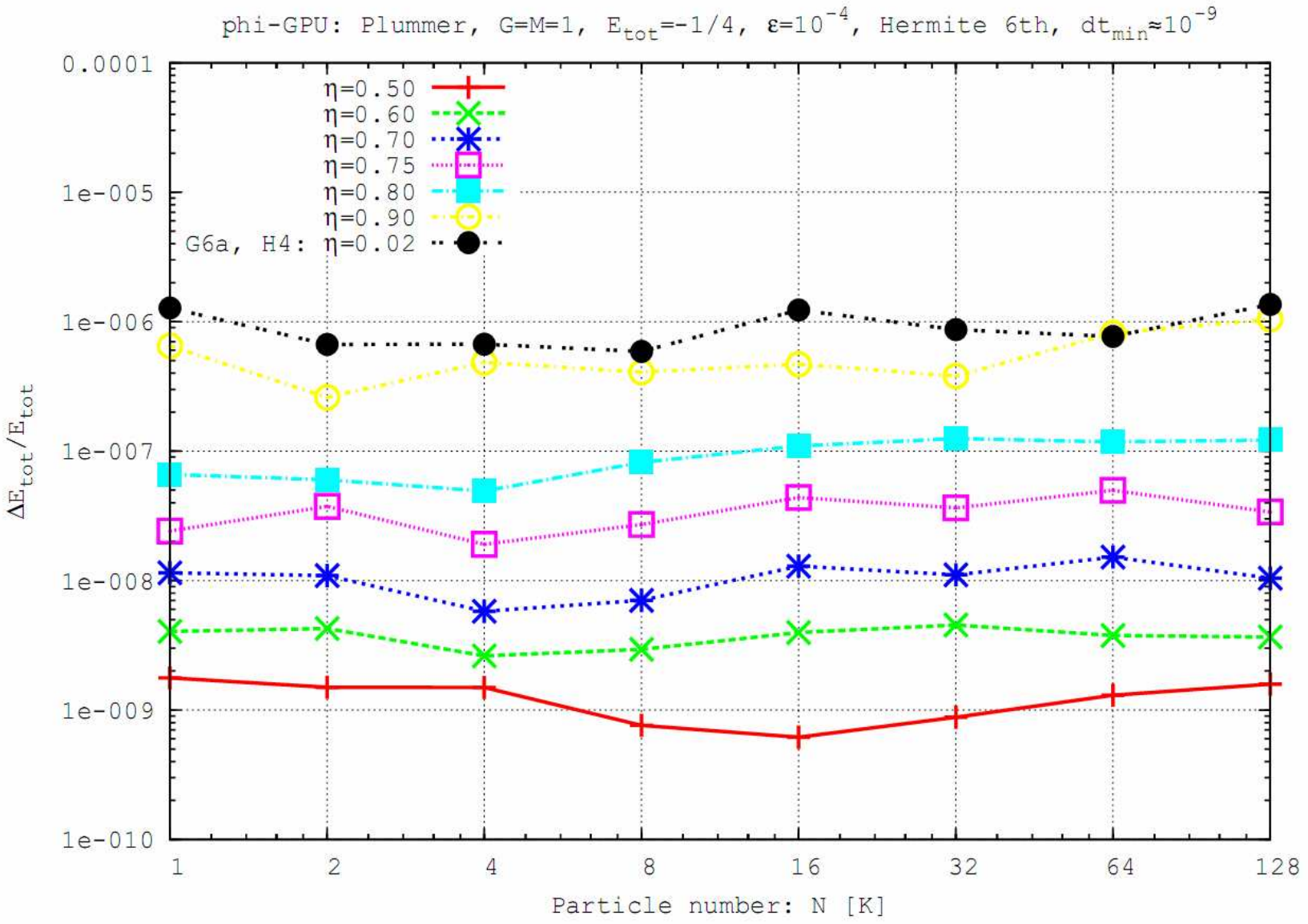
φGPU Hermite 6th results



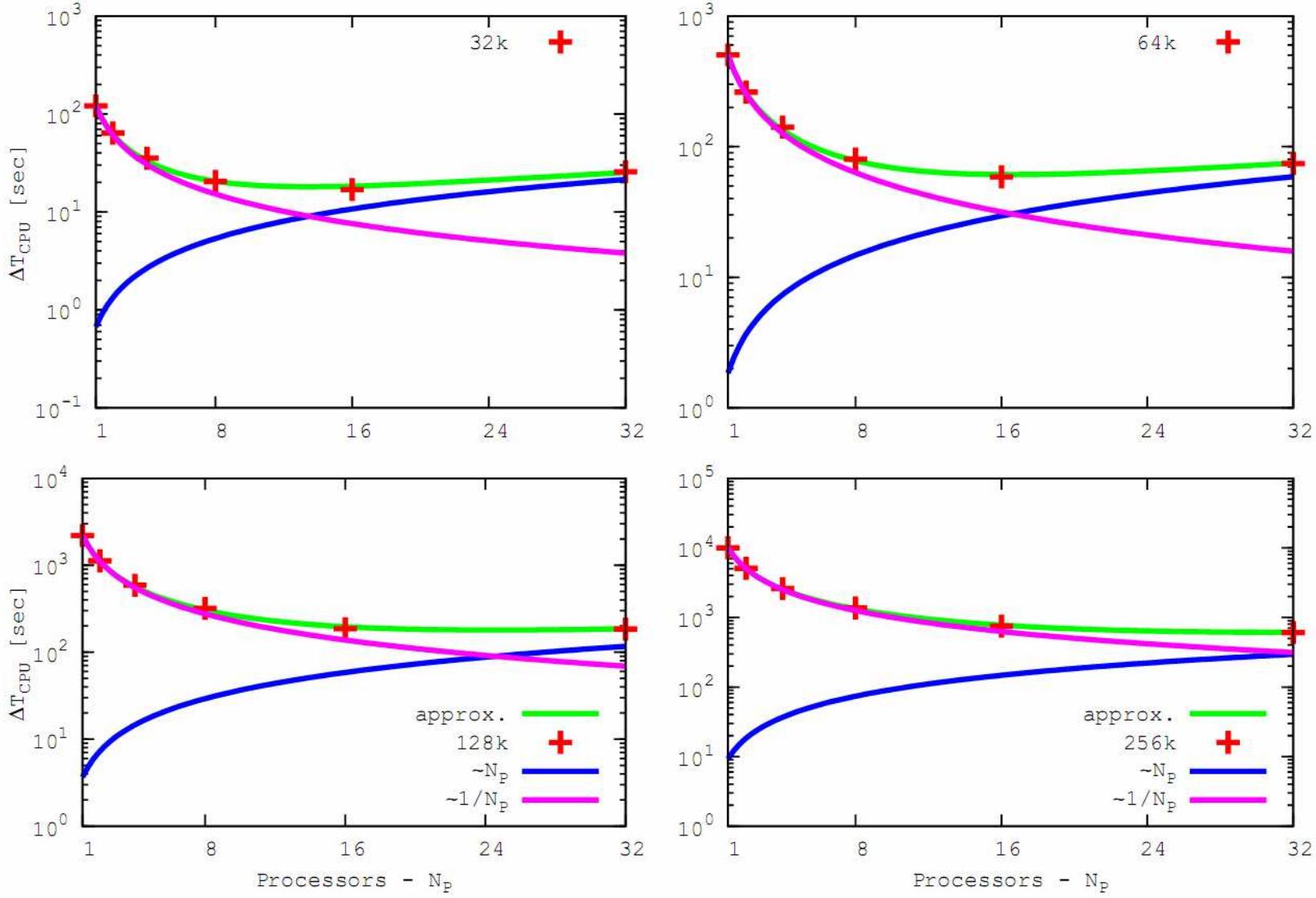
φGPU Hermite 6th results



φGPU Hermite 6th results

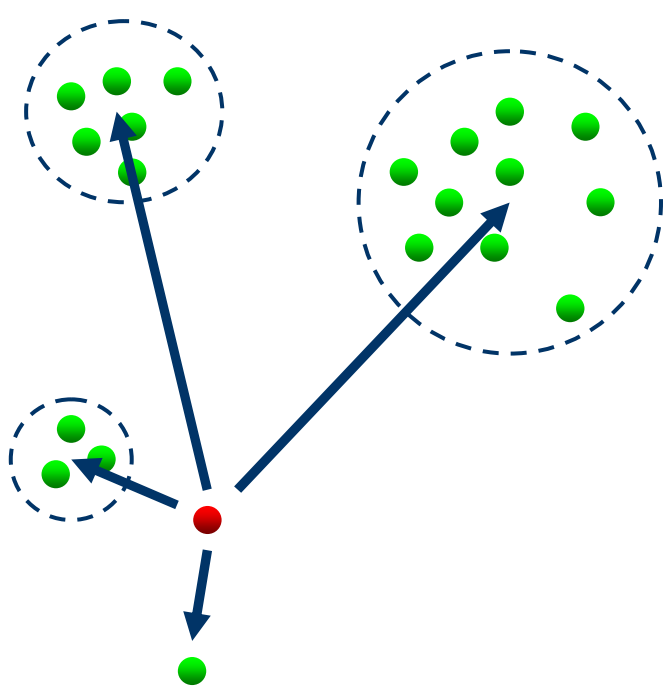


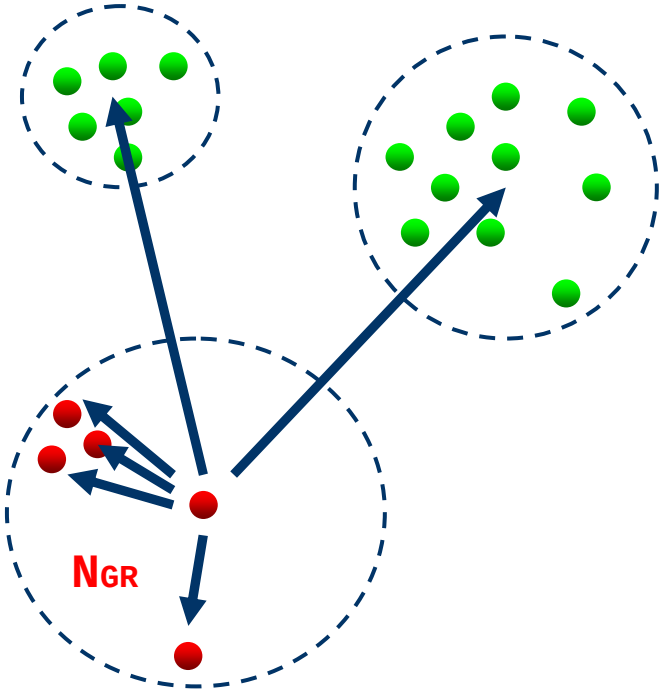
φGPU Hermite 6th results



Parallel TREE GPU gravity

Jun Makino (pC++): TREE+GRAPE/GPU code





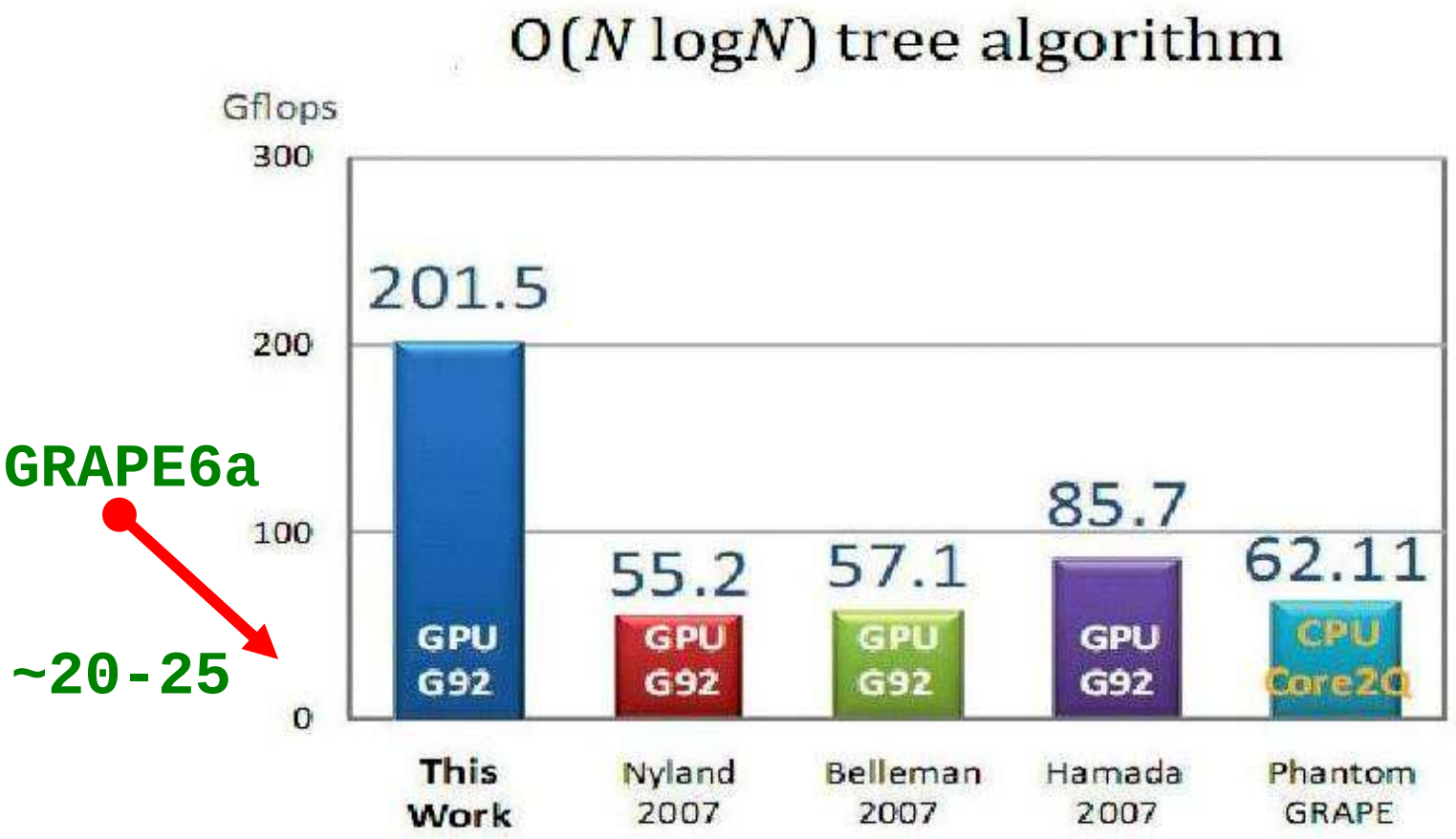
Makino, PASJ, 43, 621 (1991)

Inter. list on host ~N
Inter. list length -> short…

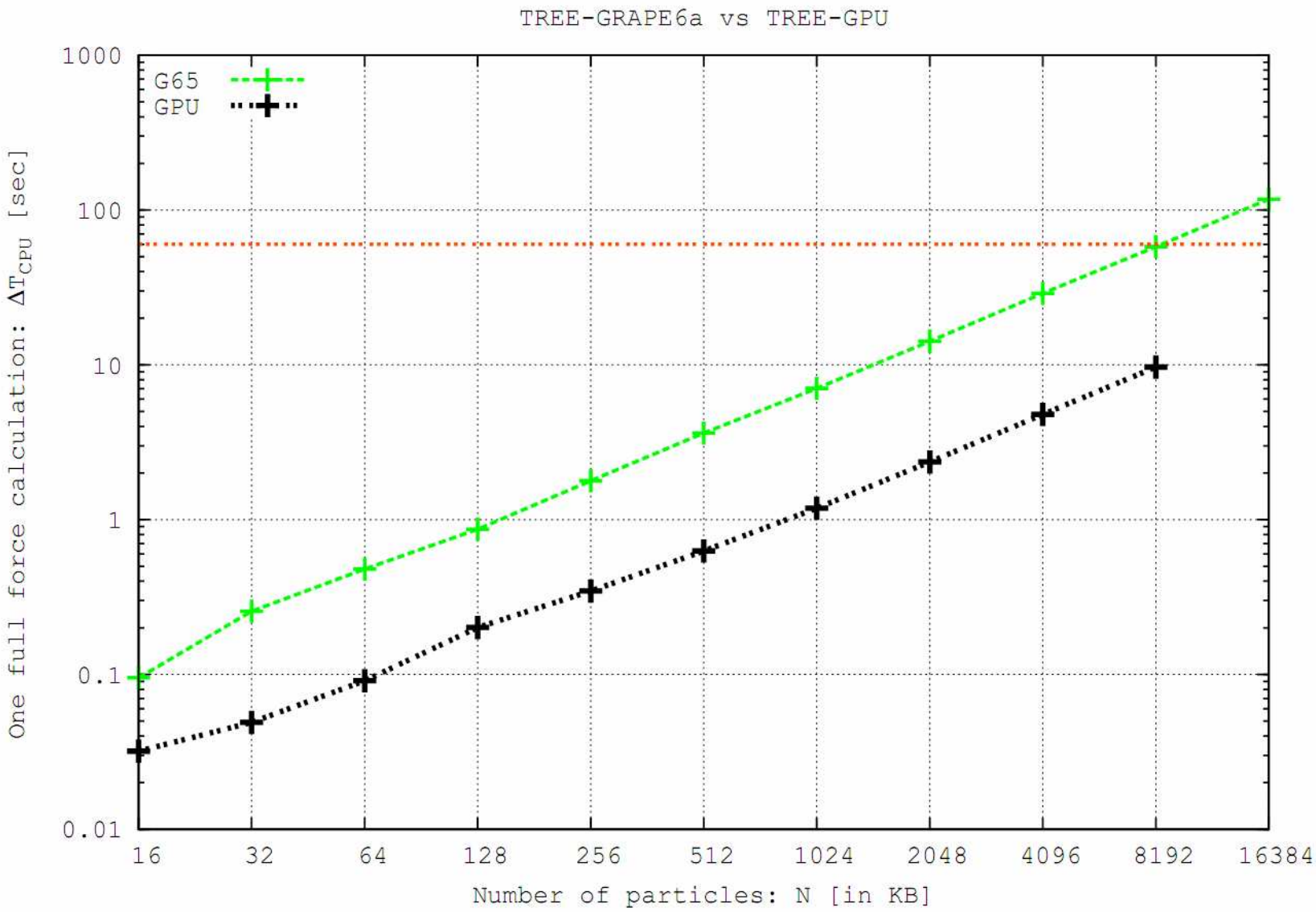
Makino, PASJ, 56, 521 (2004)
Fukushige, Makino & Kawai, PASJ, 57, 1009 (2005)
One interaction list is shared among NGR particles!
Inter. list on host ~N/NGR
Inter. list length -> larger…

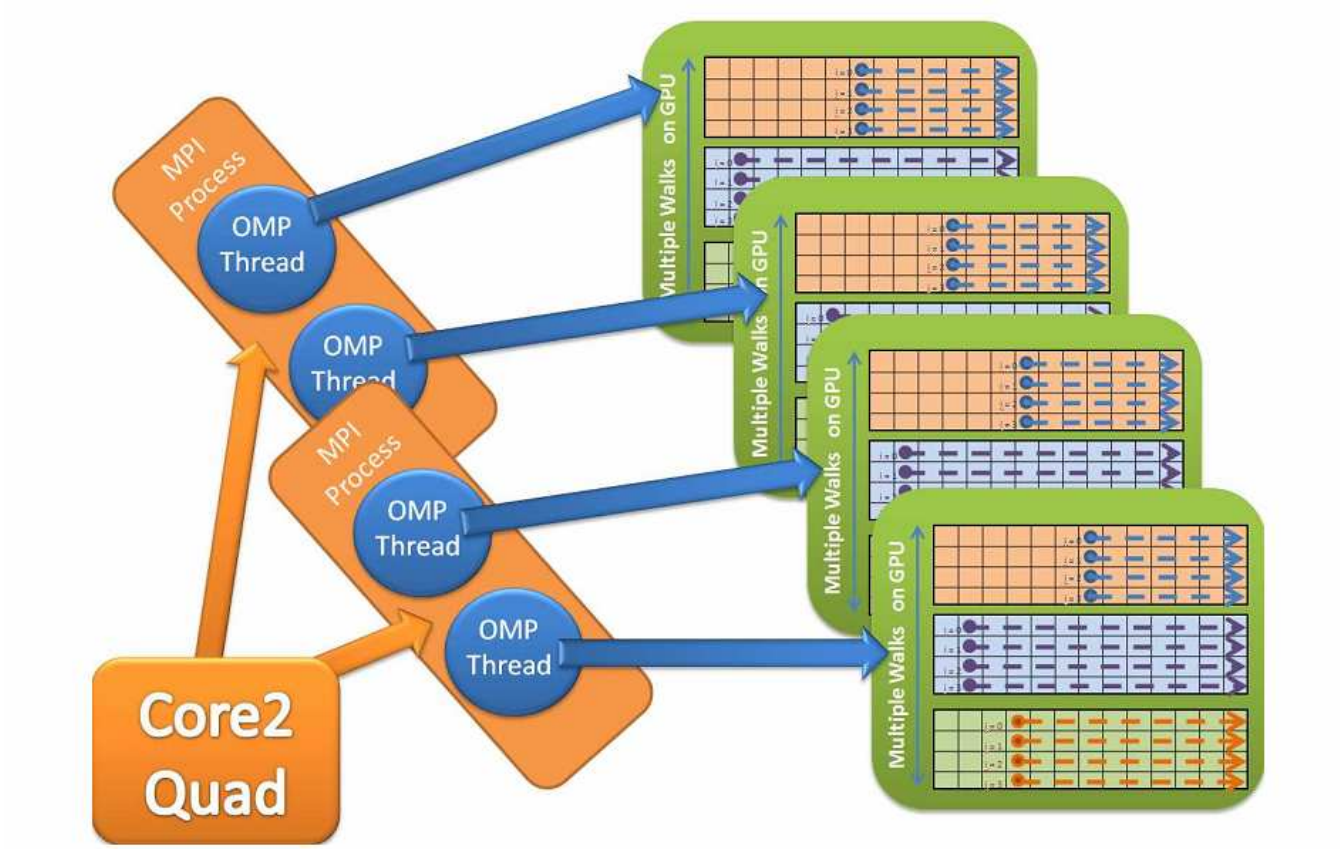
Parallel TREE GPU gravity

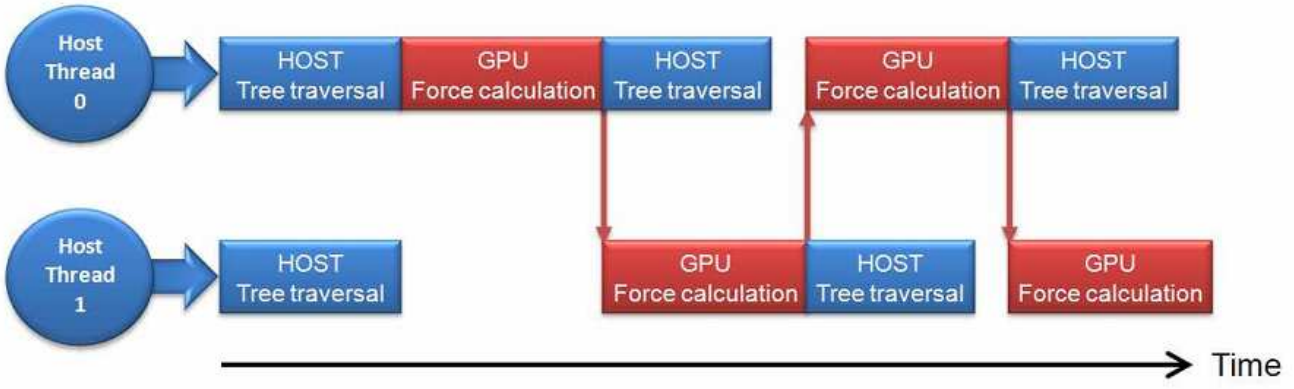
Hamada et al. 2008: TREE+GRAPE/GPU code



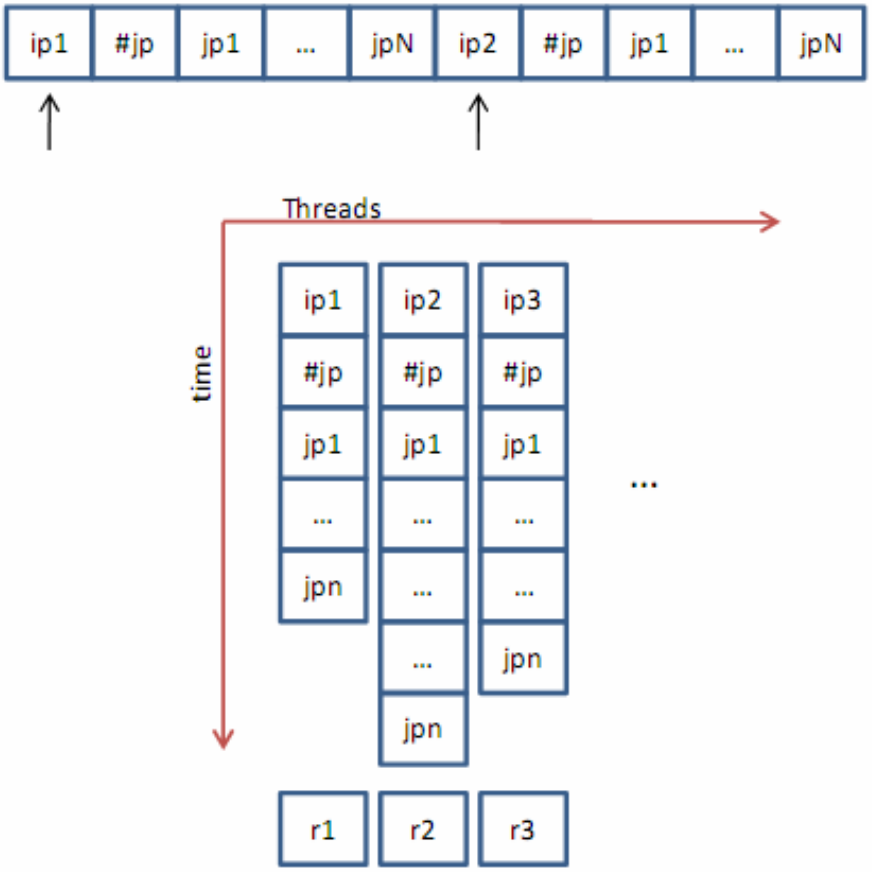
Parallel TREE GPU gravity

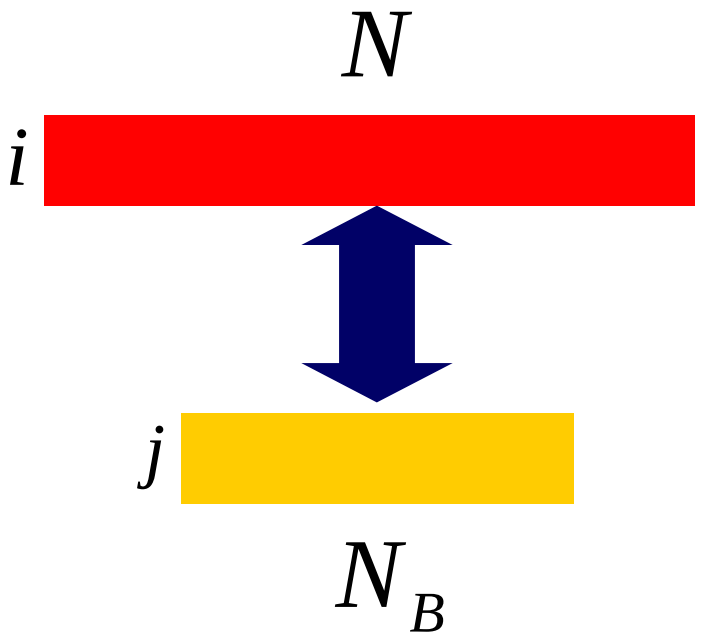


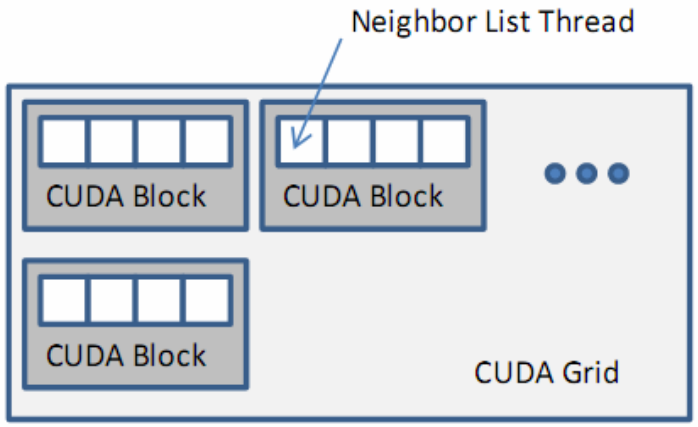

MPI Process
OMP Thread
OMP Thread
MPI Process
OMP Thread
OMP Thread
Core2 Quad
Multiple Walks on GPU
Multiple Walks on GPU
Multiple Walks on GPU
Multiple Walks on GPU

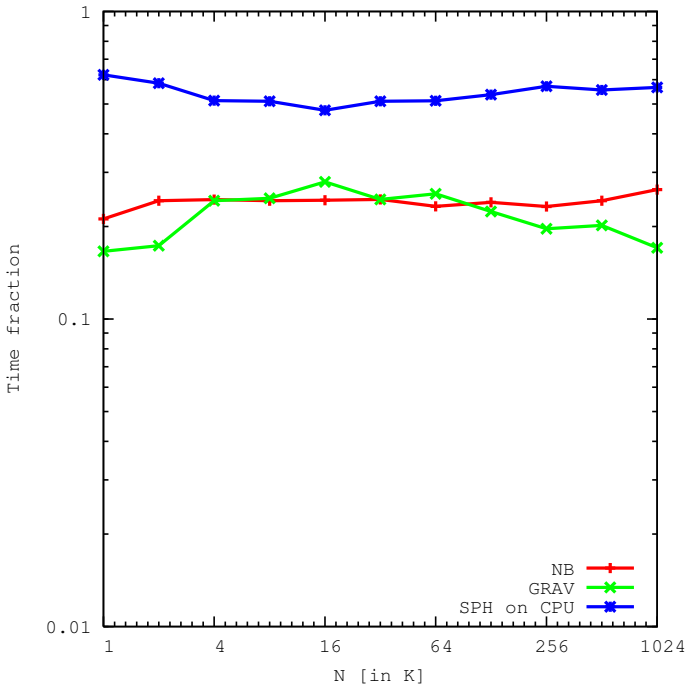

Host Thread 0
HOST Tree traversal
GPU Force calculation
HOST Tree traversal
GPU Force calculation
HOST Tree traversal
Host Thread 1
HOST Tree traversal
GPU Force calculation
HOST Tree traversal
GPU Force calculation
Time

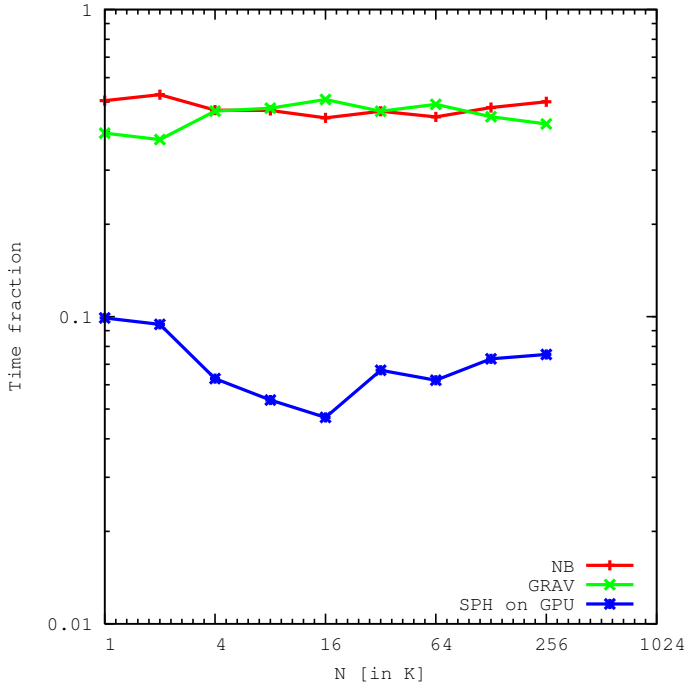
Simple GPU SPH code



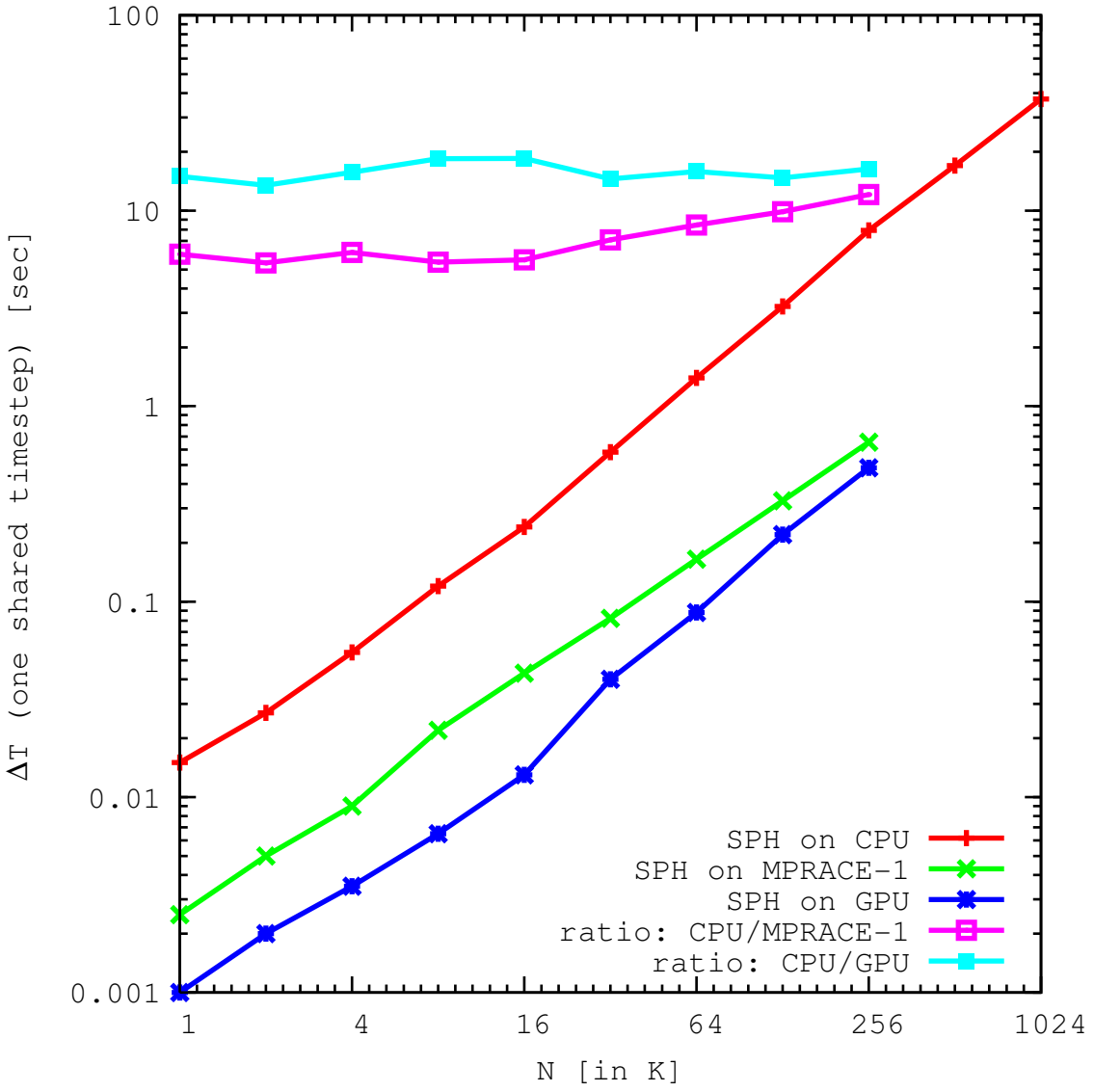








SPH speedup with GPU



SPH astrophysical results

TREE-GRAPE SPH (on 4 nodes)

M = 2000 $M_\odot$ R = 3 pc fully -> H_2

Isothermal evolution.

Initial density distr. ~1/r

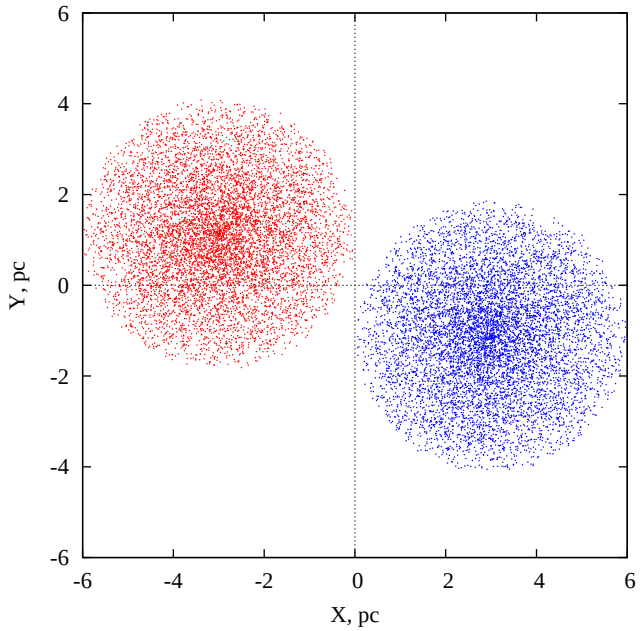
T = 20 K (c_sound = 0.3 km/sec)

V_merge = 5 km/sec

Calculation time 3*t_ff = 6 Myr

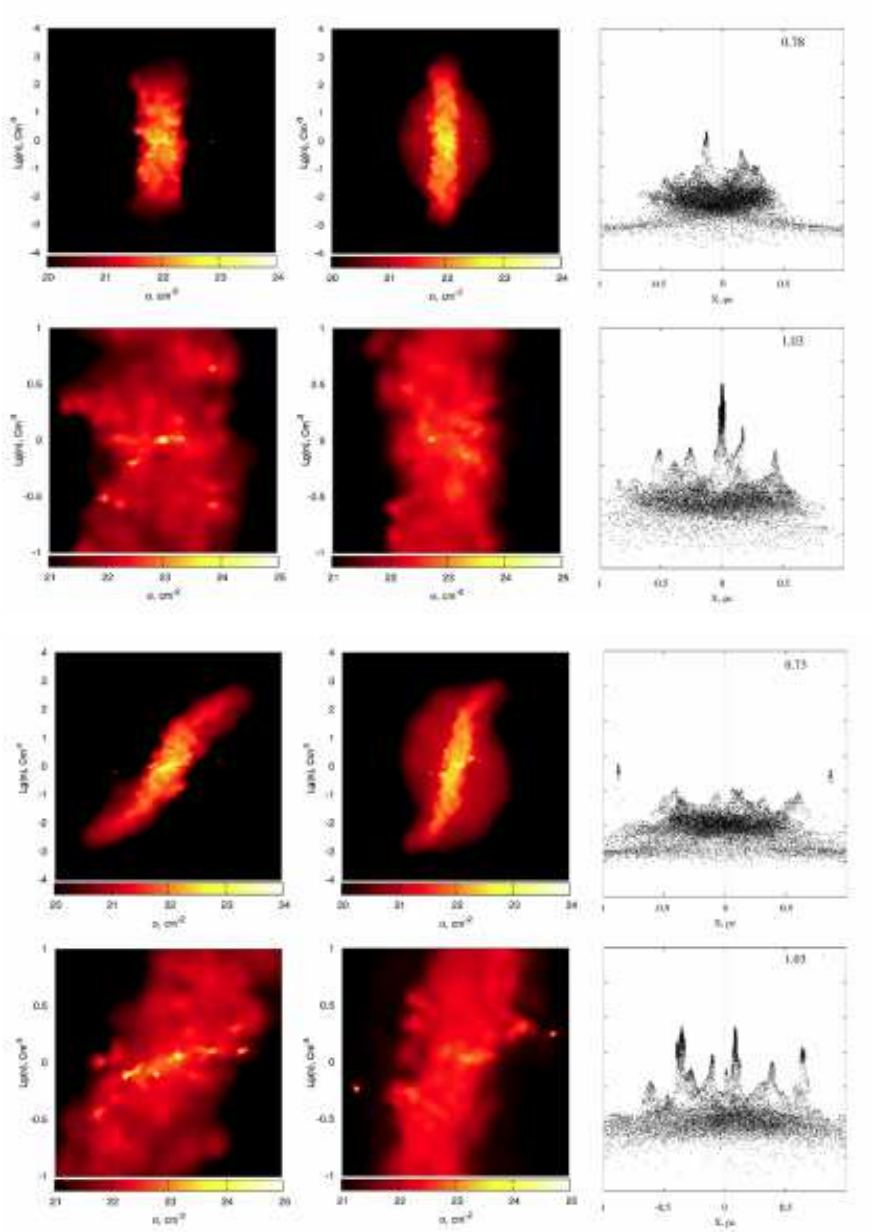
Resolution is h_min = 1e-4 pc

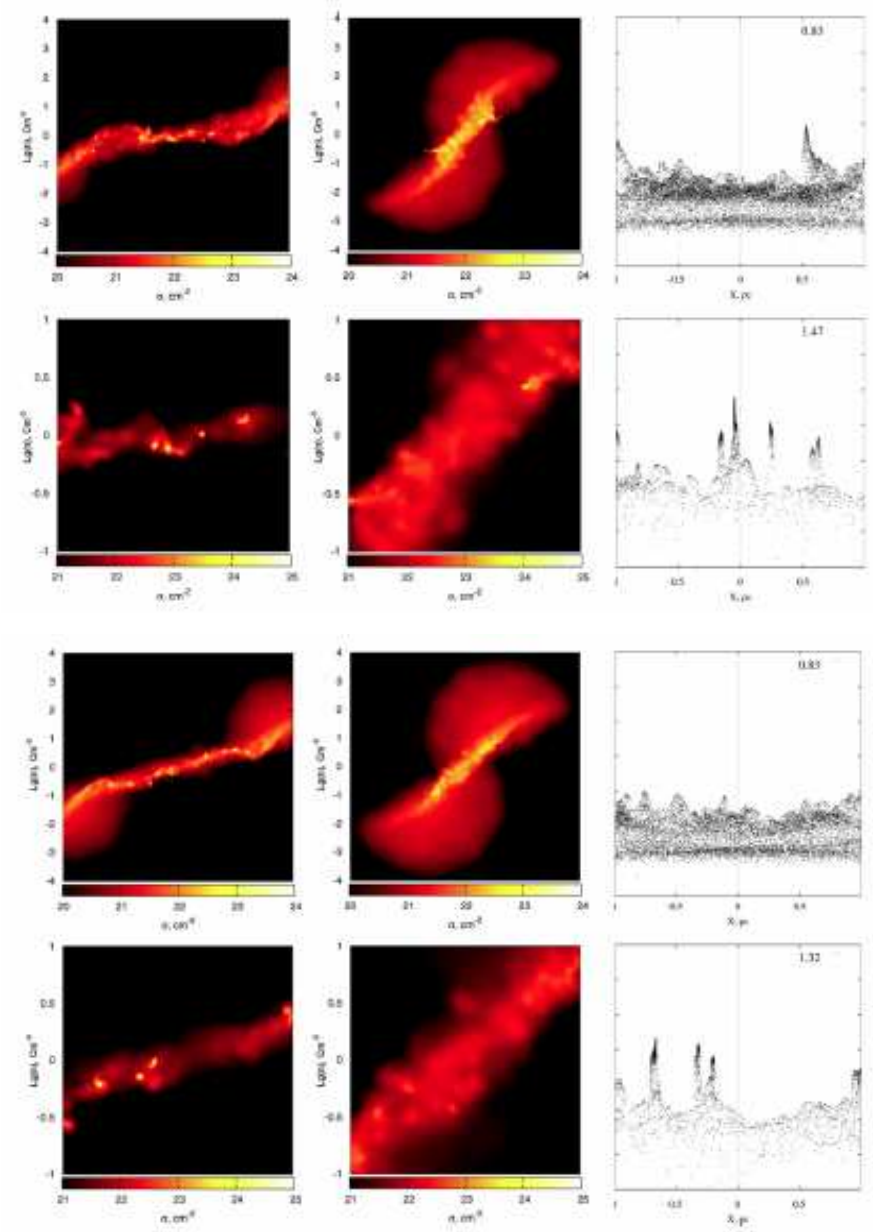
SPH: GPU/CPU speedup ~20



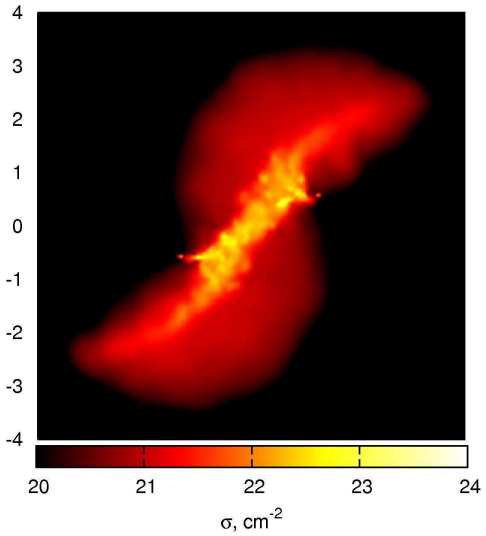
	N =	DT_CPU =
	2x4k	52 min
	2x8k	1.74 hours
	2x16k	3.5 hours
	2x32k	6.9 hours
	2x64k	14 hours
*	2x128k	28 hours
	2x256k	55 hours
	2x512k	111 hours

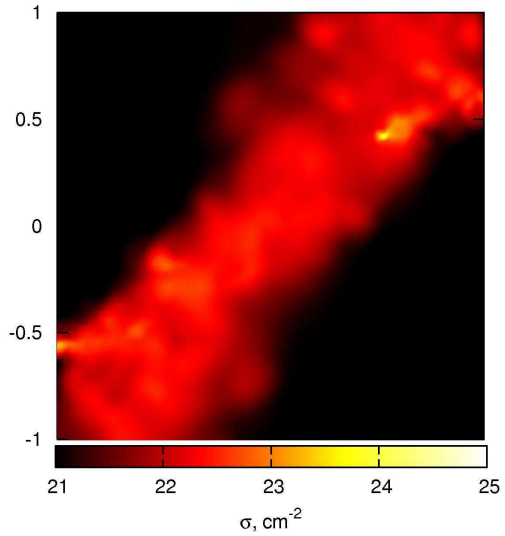
SPH astrophysical results

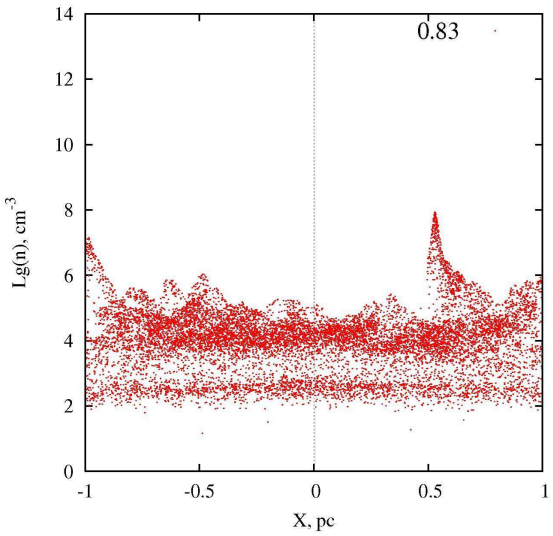


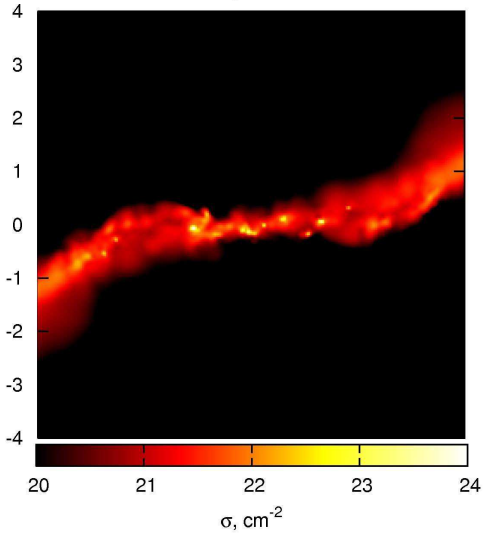


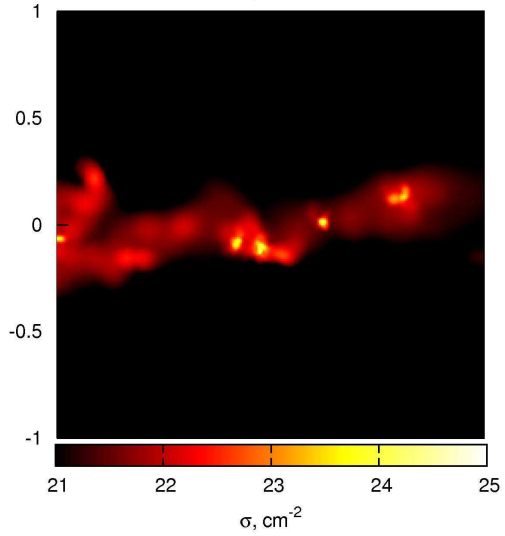
SPH astrophysical results

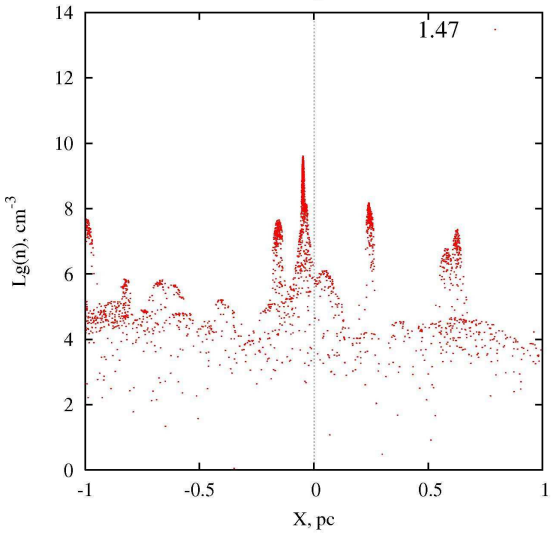




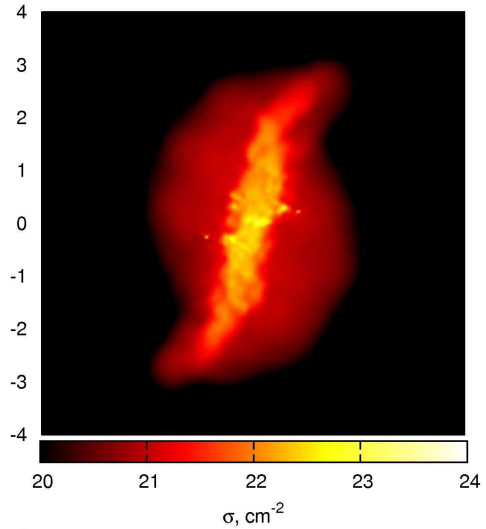


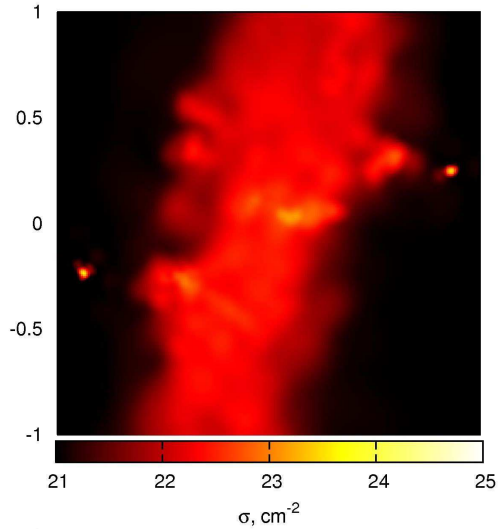


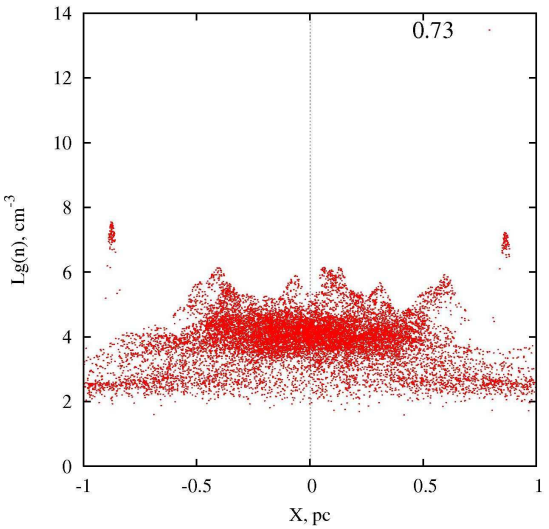


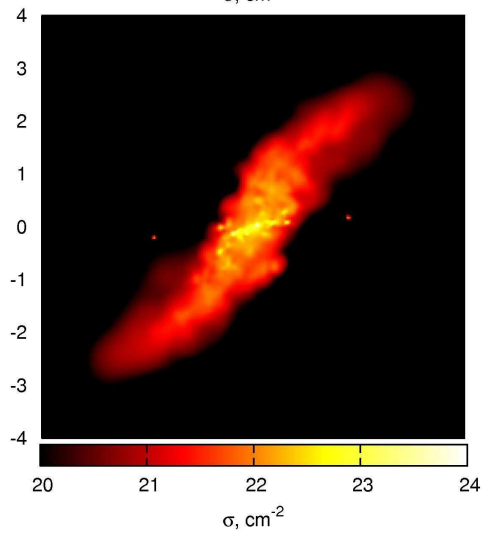


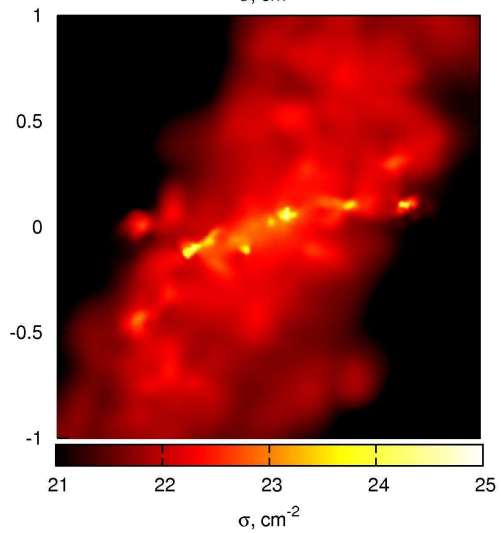
SPH astrophysical results

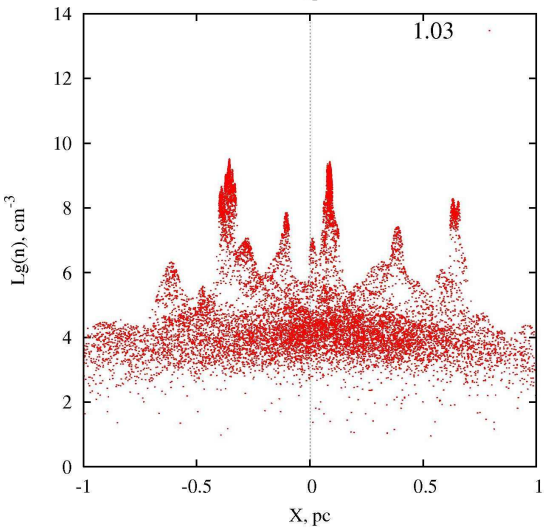




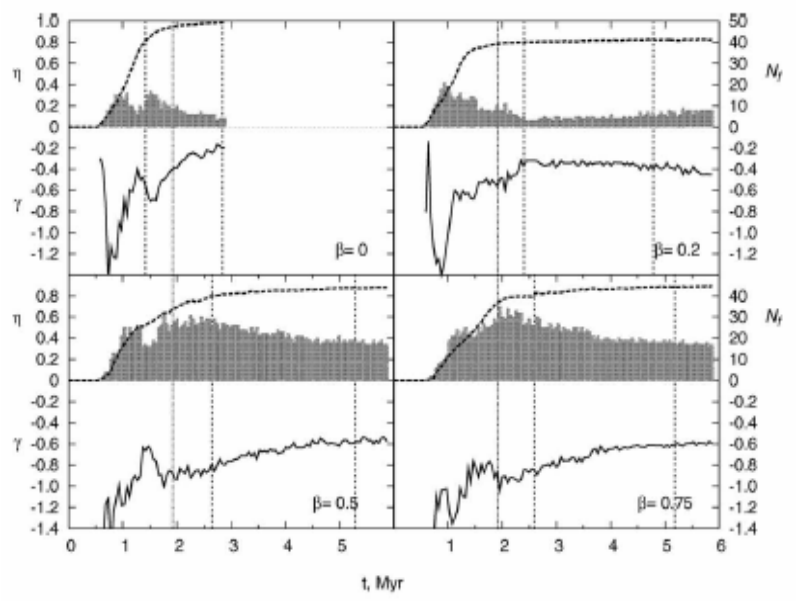


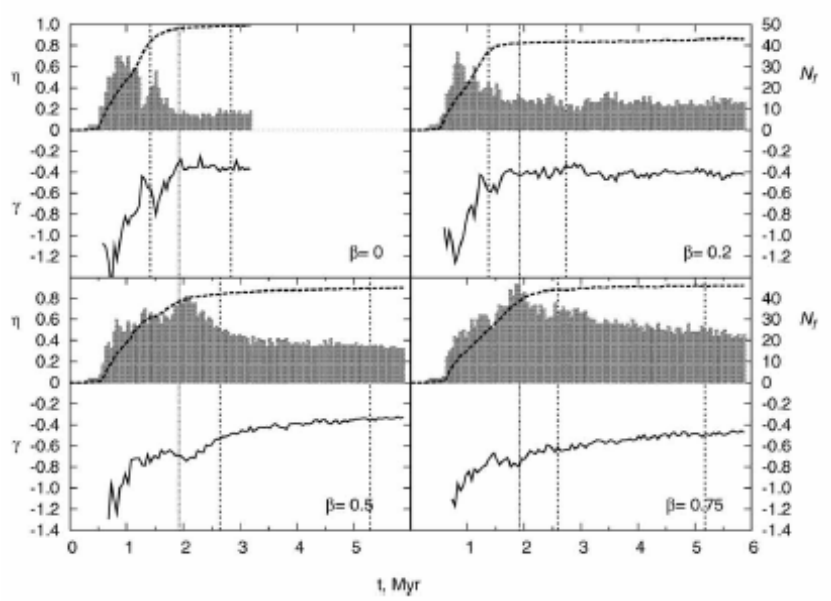


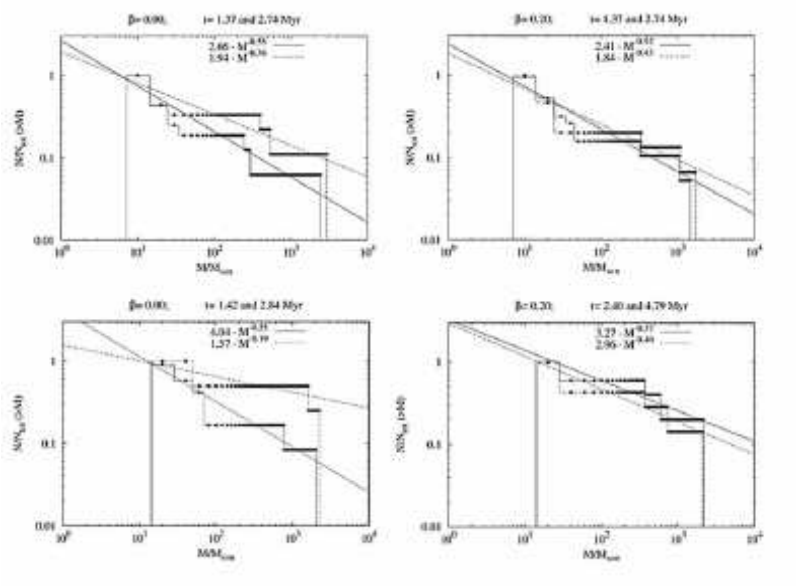


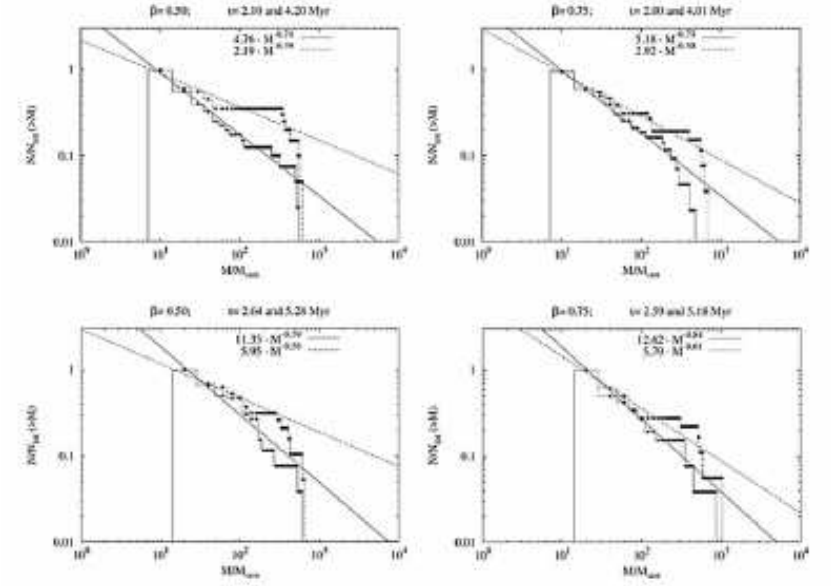


SPH astrophysical results









Plummer Initial Condition

$$M(r) = M_o \cdot \frac{1}{(1+(r_o/r)^2)^{3/2}}$$

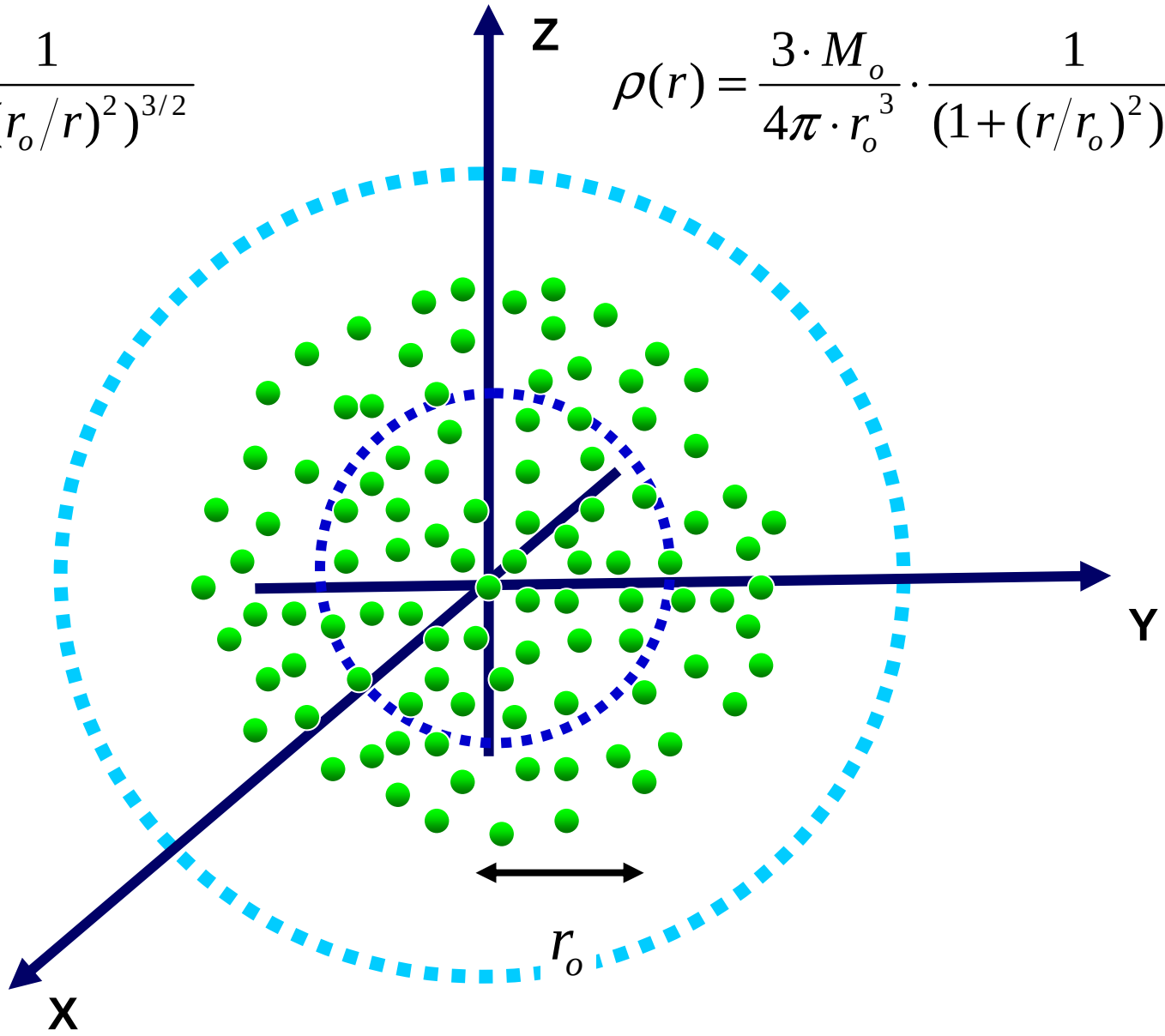
$$\rho(r) = \frac{3 \cdot M_o}{4\pi \cdot r_o^{\,3}} \cdot \frac{1}{(1+(r/r_o)^2)^{5/2}}$$

$$r_{HM} \approx 0.769$$

$$G = M = 1$$

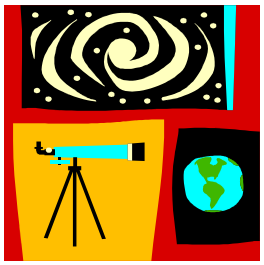
$$E_{TOT} = -\frac{1}{4}$$

$$r_o = \frac{3\pi}{16} \approx 0.6$$



Parallel code on GRAPE cluster

N



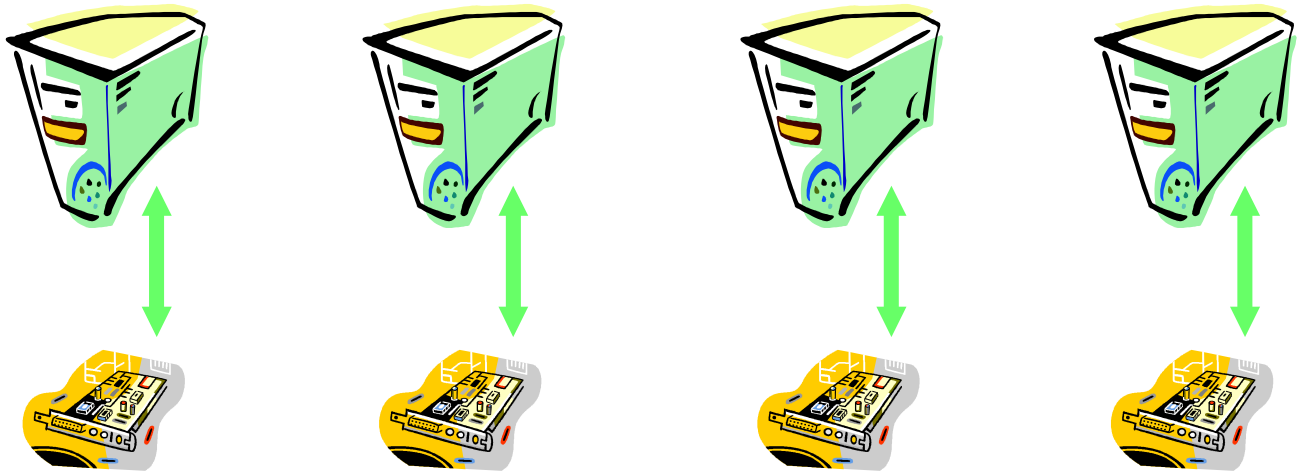
MPI_Bcast

MPI_Reduce

N_{act}

$m_i; \vec{r}_i; \vec{v}_i; t_i$

$\phi_i; \vec{a}_i; \dot{\vec{a}}_i$

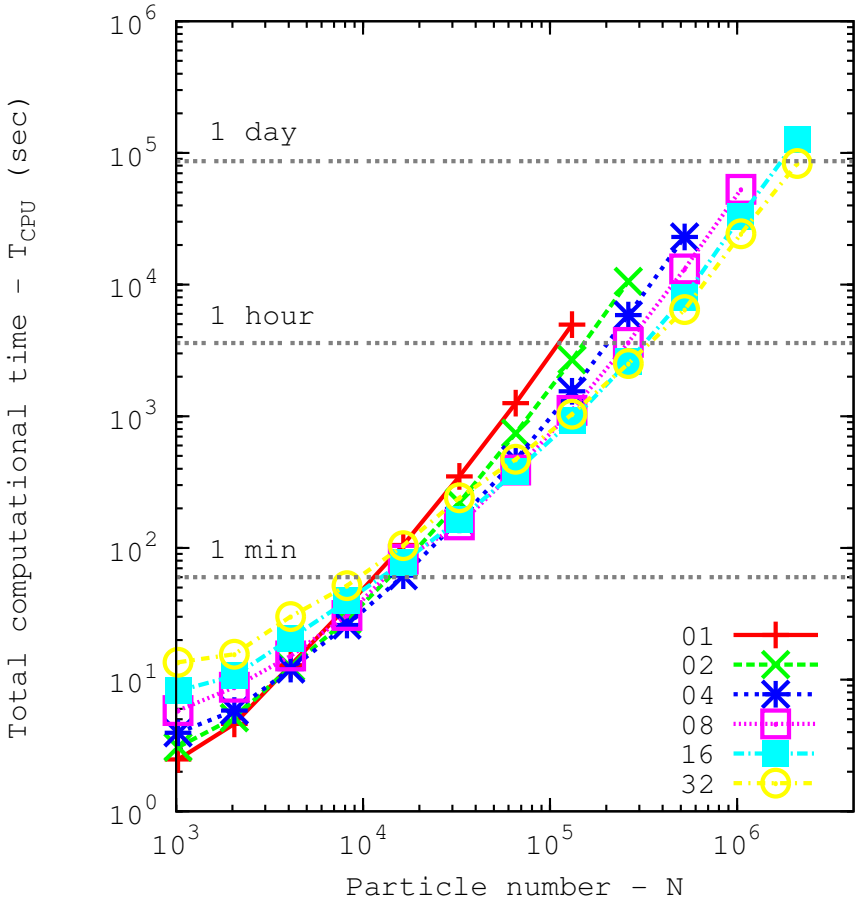


MPI_Scatter

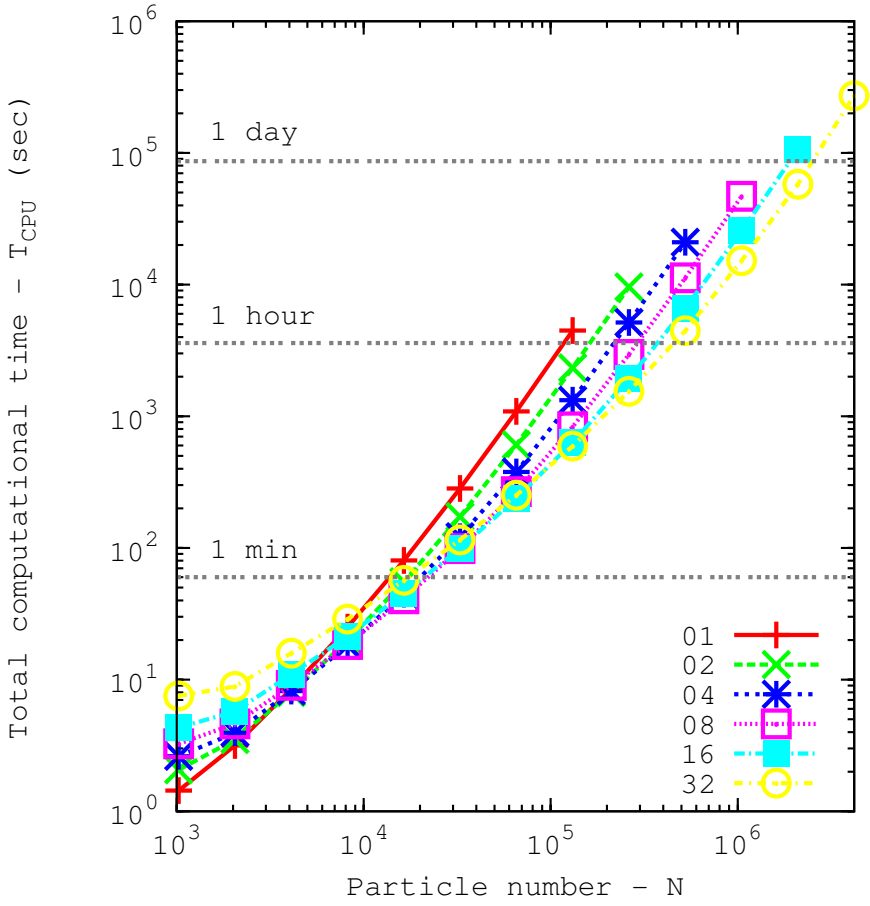
N/N_p

$m_j; \vec{r}_j; \vec{v}_j; t_j$

Parallel PP on GRAPE clusters

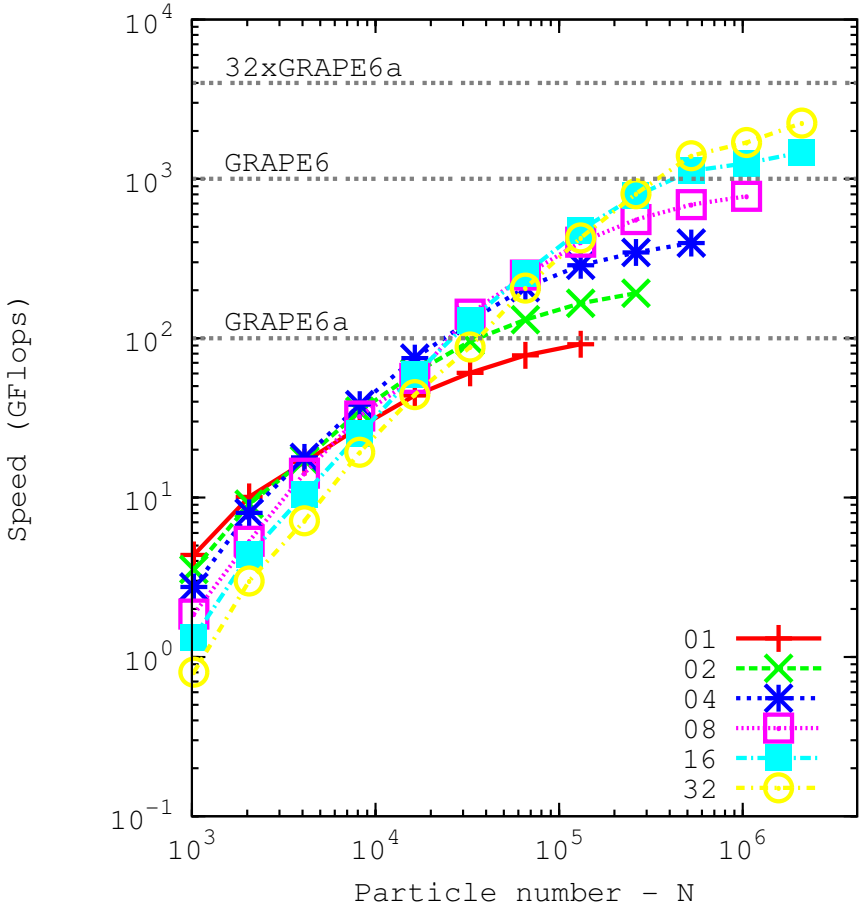


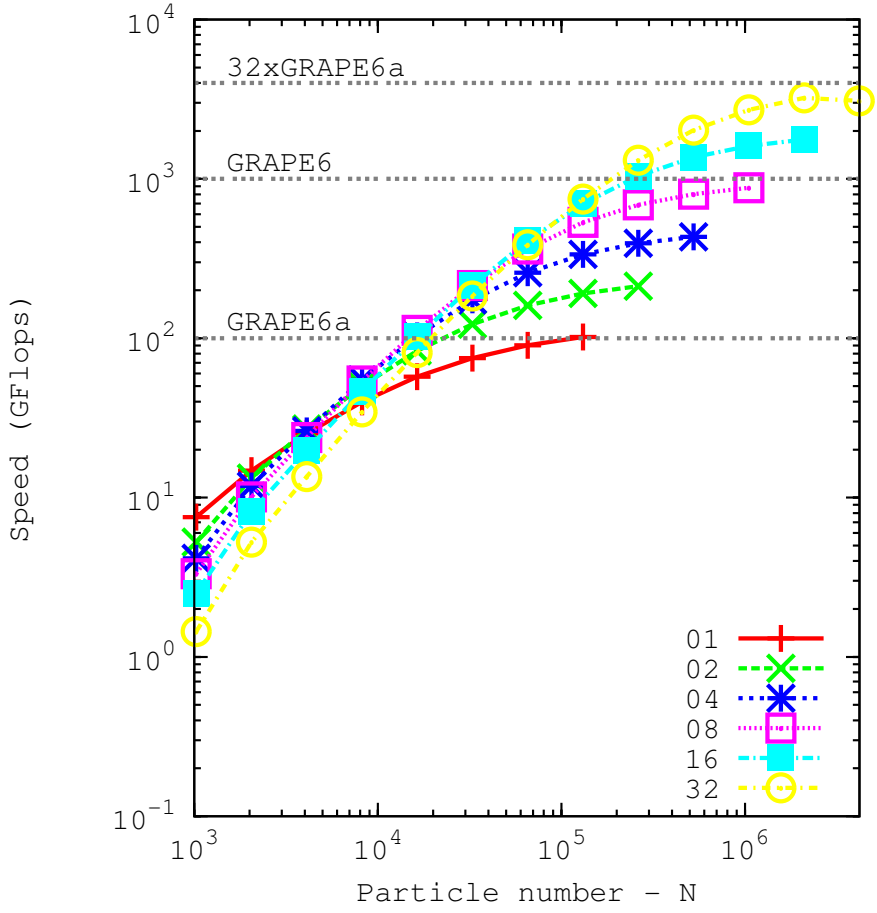
RIT



ARI

Parallel PP on GRAPE clusters





RIT: ~2.2 TFlops **ARI: ~3.2 TFlops**

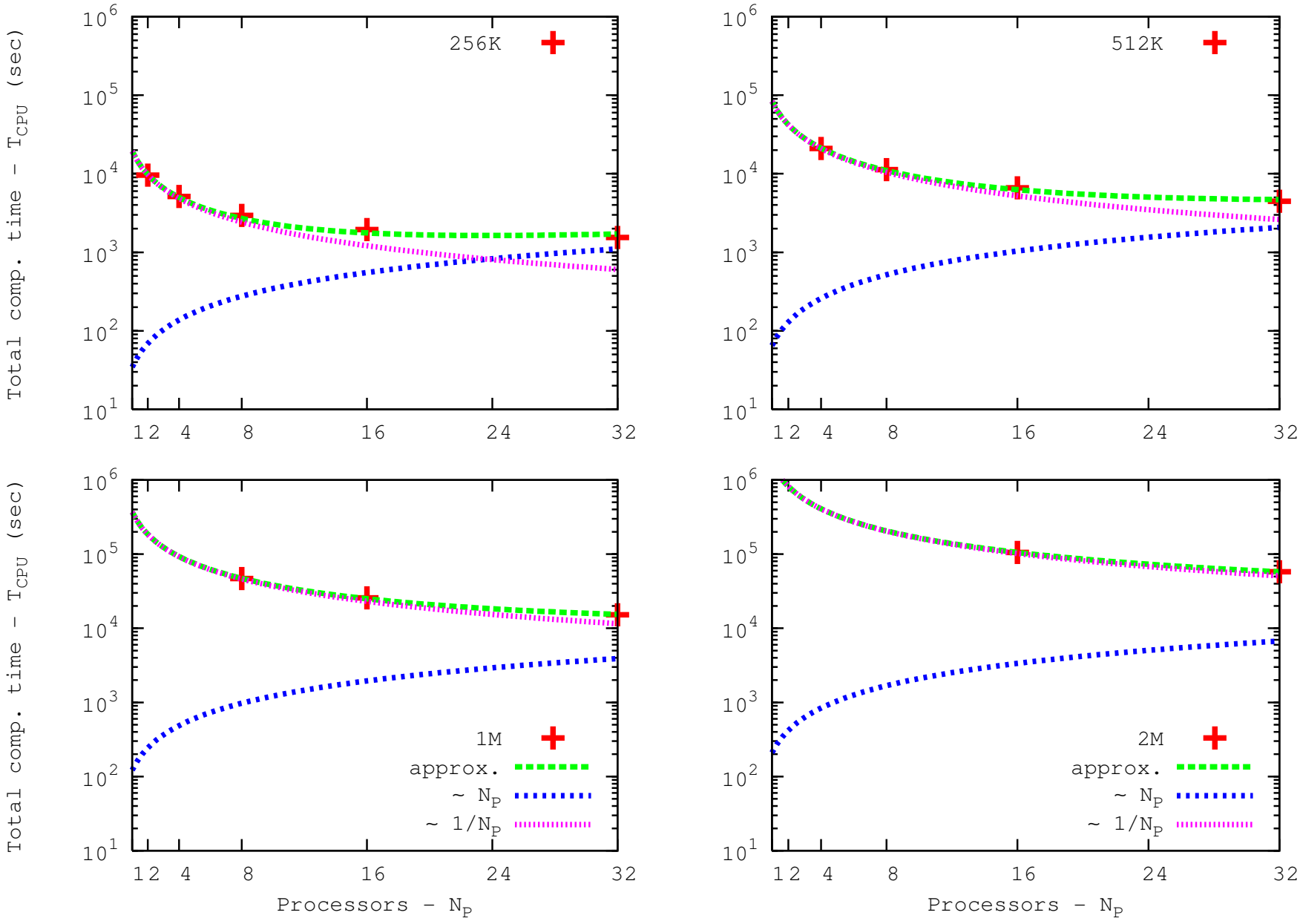
Parallel code on GRAPE cluster

$$\Delta T_{total} = \Delta T_{comm} + \Delta T_{calc}$$

$$\Delta T_{comm} \propto N_{act} \cdot N_{proc} \qquad \Delta T_{calc} \propto N_{act} \cdot \frac{N_{tot}}{N_{proc}}$$

$$N_{act} \propto N_{tot}^{2/3} \qquad \varepsilon = \frac{T_1}{N_{proc} \cdot T_{N_{proc}}}$$

Parallel code on GRAPE cluster



Parallel code on GRAPE cluster