MVCOMP1:

MODIFIED MIDPOINT, BULIRSCH - STOER, BLOCK TIME STEPS

Michela Mapelli

High-order methods. Modified mid-point method

Let's refresh the mid-point method:

$$x_0 = x(t)$$

$$x_{1/2} = x_0 + \frac{h}{2} f(x_0, t) \quad \text{Euler half step}$$

$$x_1 = x_0 + h f\left(x_{1/2}, t + \frac{h}{2}\right)$$

$$x_{3/2} = x_1 + \frac{h}{2} f(x_1, t + h)$$

$$x_2 = x_1 + h f\left(x_{3/2}, t + \frac{3}{2}h\right)$$

High-order methods. Modified mid-point method

Let's refresh the mid-point method:

$$x_0 = x(t)$$

$$x_{1/2} = x_0 + \frac{h}{2} f(x_0, t) \quad \text{Euler half step}$$

$$x_1 = x_0 + h f\left(x_{1/2}, t + \frac{h}{2}\right)$$

$$x_{3/2} = x_{1/2} + h f(x_1, t + h)$$

$$x_2 = x_1 + h f\left(x_{3/2}, t + \frac{3}{2}h\right)$$

High-order methods. Modified mid-point method

Let's refresh the mid-point method:

$$x_0 = x(t)$$

$$y_1 = x_0 + \frac{h}{2} f(x_0, t)$$

$$x_1 = x_0 + \frac{1}{2} h f\left(y_1, t + \frac{h}{2}\right)$$

$$y_2 = y_1 + h f(x_1, t + h)$$

$$x_2 = x_1 + h f\left(y_2, t + \frac{3}{2} h\right)$$

Where we have introduced y1, y2, … instead of x1/2, x3/2, ...

High-order methods. Modified mid-point method

Generalising:

$$y_{m+1} = y_m + h\, f\,(x_m, t + h)$$

$$x_{m+1} = x_m + h\, f\left(y_{m+1}, t + \left(m + \frac{1}{2}\right)\, h\right)$$

We have built a sort of leapfrog

If *t*+*H* is the end of the integration, the last two points are:

$$y_n = x\left(t + H - \frac{h}{2}\right)$$

$$x_n = x(t + H)$$

But I can also calculate the last step as

$$x(t + H - h/2) = x(t + H) - \frac{h}{2} f(x_n, t + H)$$

$$x(t + H) = x(t + H - h/2) + \frac{h}{2} f(x_n, t + H)$$

$$\rightarrow x(t + H) = y_n + \frac{h}{2} f(x_n, t + H)$$

But then I can take the average:

$$x(t + H) = \frac{1}{2}\left[x_n + y_n + \frac{h}{2} f(x_n, t + H)\right]$$

High-order methods. Modified mid-point method

William GRAGG in 1965 demonstrated that the last step

$$x(t+H) = \frac{1}{2}\left[x_n + y_n + \frac{h}{2} f(x_n, t+H)\right]$$

Completely cancels out the *h*^2 error introduced by the first Euler half-step

NOW WHAT ABOUT THE REST OF THE ERRORS?

Let's write again the eqs of the MMP method

$$x\left(t+h\right) = x\left(t\right) + h\, f\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{h}{2}\right)$$

$$x\left(t+\tfrac{3}{2}h\right) = x\left(t+\tfrac{h}{2}\right) + h\, f\left(x\left(t+h\right), t+h\right)$$

High-order methods. Modified mid-point method

Let's write them backward in time for *-h*

$$x(t-h) = x(t) - h f\left(x\left(t-\frac{h}{2}\right), t-\frac{h}{2}\right)$$

$$x\left(t-\frac{3}{2}h\right) = x\left(t-\frac{h}{2}\right) - h f\left(x(t-h), t-h\right)$$

And then let's substitute t → t+3/2 h in the backward formula

$$x\left(t+\frac{h}{2}\right) = x\left(t+\frac{3h}{2}\right) - h f\left(x(t+h), t+h\right)$$

$$x(t) = x(t+h) - h f\left(x\left(t+\frac{h}{2}\right), t+\frac{h}{2}\right)$$

They are exactly the same as the general formula

$$x(t+h) = x(t) + h f\left(x\left(t+\frac{1}{2}h\right), t+\frac{h}{2}\right)$$

$$x\left(t+\frac{3}{2}h\right) = x\left(t+\frac{h}{2}\right) + h f\left(x(t+h), t+h\right)$$

→ ***The MMP method is time-reversal symmetric***

High-order methods. Time-reversal symmetric

The change in the solution backwards is the exact REVERSE
of the change forward

→ ***the backward error must be equal to minus the forward one***

$$\epsilon(-h) = -\epsilon(h)$$

EPSILON is an ODD function: it is anti-symmetric about origin and
Its Taylor expansion in powers of h contains only ODD terms

$$\epsilon(h) = c_3\, h^3 + c_5\, h^5 + \ldots$$

Even if the total error contains only even terms:

$$E(h) = \epsilon(h)\,\frac{\Delta t}{h} \propto h^2$$

High-order methods. Bulirsch – Stoer method

The Bulirsch-Stoer method exploits the advantages of the modified midpoint method to reach an **accuracy as large as we want, at least in principle**

Let's consider, for simplicity, a first-order ODE in a single variable: dx/dt = f(x, t).

We want to integrate it from t to $t + h_1$.

We start calculating the solution

i) with the **modified midpoint method**,

ii) in just one single time-step of size h_1 equal to the entire range.

This yields a very approximate solution $x(t + h_1) = R_{1,1}$

Then, we split the time step in two: $h_2 = h_1 / 2$

and we repeat the calculation, getting a solution $x(t + h_2) = R_{2,1}$

Since the error on the modified midpoint method is always and even function of the step-size, we have that

(1) $$x(t + h_1) = R_{2,1} + c_1 h_2^2 + \mathcal{O}(h_2^4)$$

(2) $$x(t + h_1) = R_{1,1} + c_1 h_1^2 + \mathcal{O}(h_1^4) = R_{1,1} + 4 c_1 h_2^2 + \mathcal{O}(h_2^4)$$

where c_1 = unknown constant,

$R_{1,1}$ and $R_{2,1}$ are the solutions with the modified midpoint method

High-order methods. Bulirsch – Stoer method

We use (1) and (2) to calculate c_1: $c_1\, h_2^2 = \frac{1}{3}(R_{2,1} - R_{1,1})$

Substituting in (1): $x(t + h_1) = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1}) + \mathcal{O}(h_2^4)$
where we got rid of the terms in h^2

Let's call the new solution: $R_{2,2} = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1})$

We can take this approach further. Let us now consider $h_3 = 1/3\ h_1$.
The solution of the modified midpoint will be

$$x(t + h_1) = R_{3,1} + c_1\, h_3^2 + \mathcal{O}(h_3^4)$$
$$x(t + h_1) = R_{2,1} + c_1\, h_2^2 + \mathcal{O}(h_2^4)$$
$$= R_{2,1} + c_1\, \frac{9}{4}\, h_3^2$$

$$\rightarrow c_1\, h_3^2 = \frac{4}{5}\,(R_{3,1} - R_{2,1})$$

$$\rightarrow x(t + h_1) = R_{3,1} + \frac{4}{5}\,(R_{3,1} - R_{2,1}) + \mathcal{O}(h_3^4)$$

Let's define:

$$R_{3,2} \equiv R_{3,1} + \frac{4}{5}(R_{3,1} - R_{2,1})$$

High-order methods. Bulirsch – Stoer method

Let's go to the higher order

$$x(t+H) = R_{22} + O(h_2^4)$$

$$x(t+H) = R_{32} + O(h_3^4)$$

Thus I can write

$$x(t+H) = R_{22} + c_2 h_2^4 + O(h_2^6)$$

$$x(t+H) = R_{32} + c_2 h_3^4 + O(h_3^6)$$

But then:

$$x(t+H) = R_{22} + c_2 \left(\frac{3}{2}\right)^4 h_3^4 + O(h_2^6)$$

$$R_{32} + c_2 h_3^4 = R_{22} + c_2 \frac{81}{16} h_3^4$$

$$\Rightarrow \quad c_2 h_3^4 = \frac{16}{65}(R_{32} - R_{22})$$

$$x(t+h) = R_{32} + \frac{16}{65}(R_{32} - R_{22}) + O(h_3^6)$$

Which allows me to get rid of the *h*^4 errors:

defining $R_{33} = R_{32} + \frac{16}{65}(R_{32} - R_{22})$

High-order methods. Bulirsch – Stoer method

Generalizing:

① $x(t+H) = R_{n,m} + C_m h_n^{2m} + O(h_n^{2m+2})$

The estimate of $R_{n-1,m}$ corresponds to

② $x(t+H) = R_{n-1,m} + C_m h_{n-1}^{2m} + O(h_{n-1}^{2m+2})$

with $h_n = \frac{H}{n}$ $\quad h_{n-1} = \frac{H}{n-1}$

③ Substituting ③ into ②

$\Rightarrow$ $h_n\, x(t+H) = R_{n-1,m} + C_m \left(\frac{n}{n-1}\right)^{2m} h_n^{2m} + O(\quad)$

Equation with ①

High-order methods. Bulirsch – Stoer method

Substituting (3) into (2)

$$x(t+H) = R_{n-1,m} + C_m \left(\frac{n}{n-1}\right)^{2m} h_n^{2m} + \mathcal{O}(\,)$$

Equating with (1)

$$R_{n,m} + C_m h_n^{2m} + \mathcal{O}() = R_{n-1,m} + C_m \left(\frac{n}{n-1}\right)^{2m} h_n^{2m} + \mathcal{O}()$$

$$\Rightarrow C_m h_n^{2m} = \frac{R_{n,m} - R_{n-1,m}}{[n/(n-1)]^{2m} - 1}$$

High-order methods. Bulirsch – Stoer method

Using $c_m h_n^{2m} = \dots$

into eq (1) gives an estimate

of $x(t+H)$ that is 2 orders

of magnitude in h more accurate

$$x(t+H) = R_{n,m+1} + \mathcal{O}\left(h_n^{2m+2}\right)$$

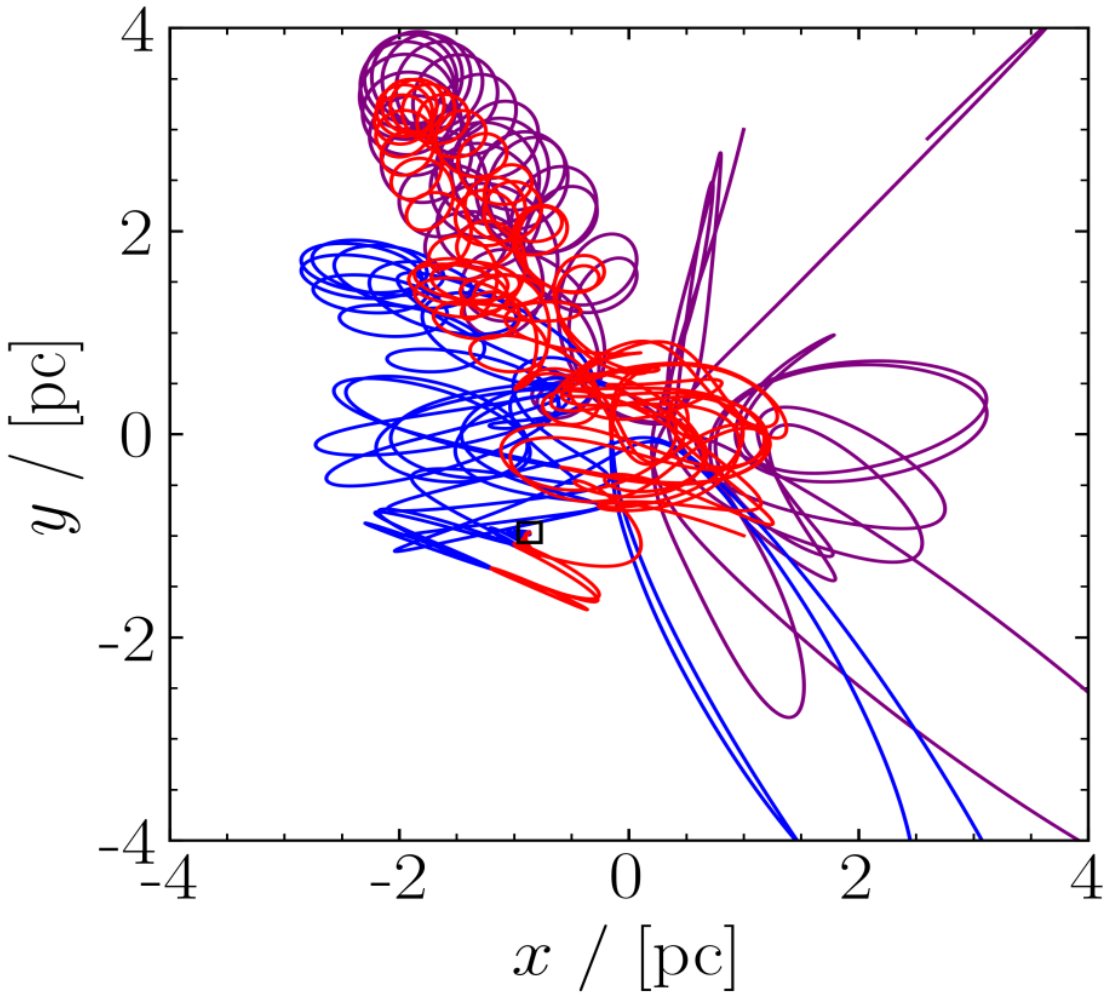
with

$$R_{n,m+1} = R_{n,m} + \underbrace{\frac{R_{n,m} - R_{n-1,m}}{\left[n/(n-1)\right]^{2m} - 1}}_{c_m h_n^{2m}}$$

High-order methods. Bulirsch – Stoer method

Used by code **BRUTUS** (Boekholt et al. 2018)
to reach arbitrarily large precision

https://comp-astrophys-cosmol.springeropen.com/articles/10.1186/s40668-014-0005-3



Adaptive time-step. Block time-step

1. Initial time-step calculated as $\Delta t_i = \eta \frac{a_i}{\dot{j}_i}$
   for a particle i

   η = 0.01 – 0.02 is good choice

2. The new system time is set as t + min (Δt_i)
   All particles with time-step = min (Δt_i) are called ACTIVE PARTICLES

3. Positions and velocities are PREDICTED for ALL PARTICLES

4. Acceleration and jerk are calculated ONLY for ACTIVE PARTICLES

5. Positions and velocities are CORRECTED ONLY for active particles
   (for the other particles predicted values are fine)

After force calculation, new timesteps evaluated as 1. and everything is repeated

BUT a different t_i for each particles is VERY EXPENSIVE (or system loses coherence)

Adaptive time-step. Block time-step

BLOCK TIME STEP SCHEME consists in grouping particles by replacing their individual time steps Δt_i with a

BLOCK TIME STEP $\Delta t_{i,b} = (1/2)^n$

where *n* is chosen according to

$$\left(\frac{1}{2}\right)^n \leq \Delta t_i < \left(\frac{1}{2}\right)^{n-1}$$

This imposes that $t/\Delta t_{i,b}$ be an integer → good for synchronizing the particles at some time

Often it is set a minimum $\Delta t_{min} = (1/2)^{23}$

Special-purpose hardware. Grape and GPUs

GRAPE (see http://www.ids.ias.edu/~piet/act/comp/hardware/index.html)

GRAvity PipE: a hardware implementation of Newtonian pair-wise force calculations between particles in a self-gravitating N-body system

HIGHLY SPECIALIZED HARDWARE

SORT of GRAVITY ACCELERATOR

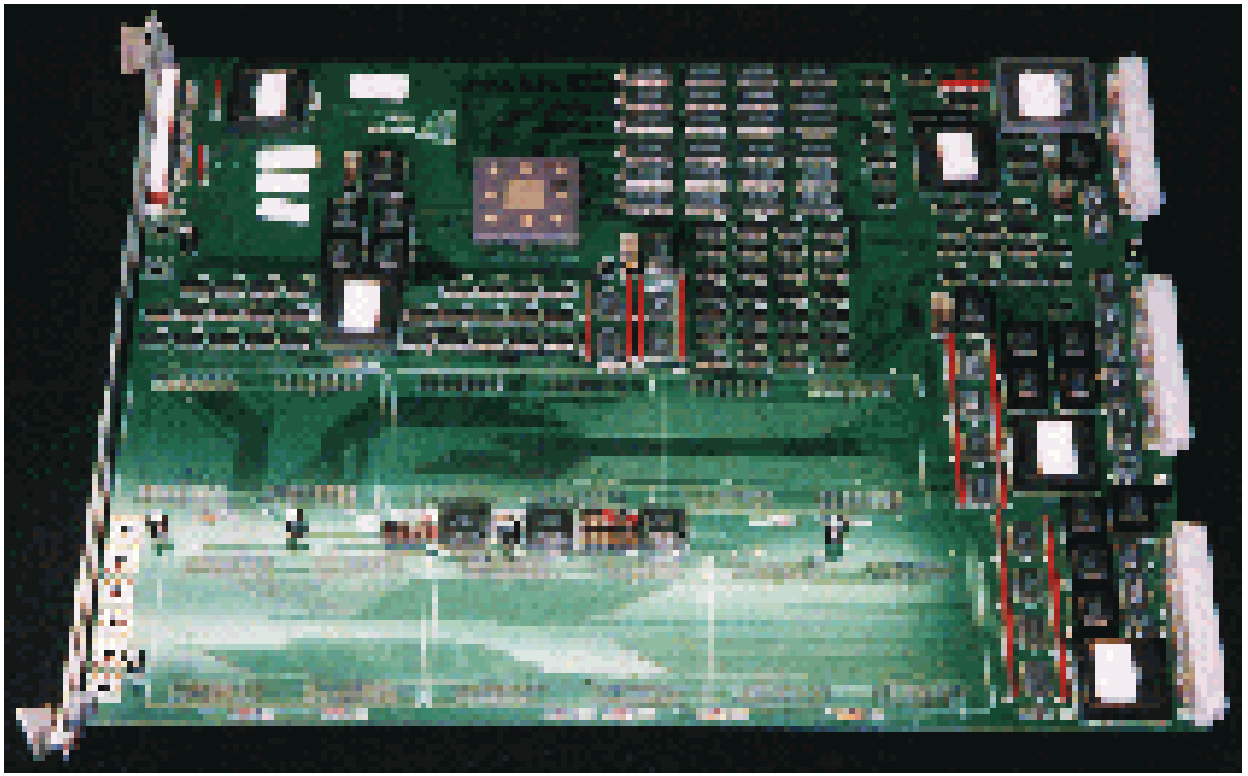
as a

GRAPHICS CARD is a GRAPHICS ACCELERATOR

– – – – – – – – – – – – –

Predictor/corrector on PC

Acceleration and jerk calculation on GRAPE



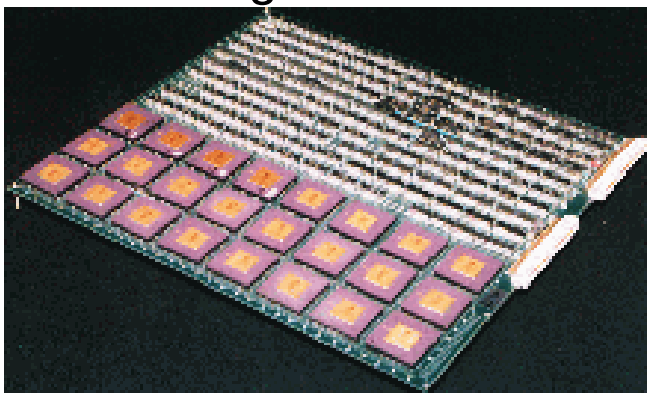
Special-purpose hardware. Grape and GPUs

FLOPS: floating point operations per second

History:

1989: GRAPE project starts at Tokyo university (Daiichiro Sugimoto and then Junichiro Makino)

GRAPE-1 at 240 Mflops at single precision



1990: GRAPE-2 at 40 Mflops at double pr.

1991: GRAPE-3 at 15 Gflops at single pr.

(first one with specialized gravity chips rather than commercial chips)

→ Comparison: A11 chip mounted on Iphone reaches ~100 Gflops

1995: GRAPE-4 at double pr.

4-cabinet GRAPE-4 computer reaches 1Tflop

!!! 1st computer which reached 1Tflop !!!

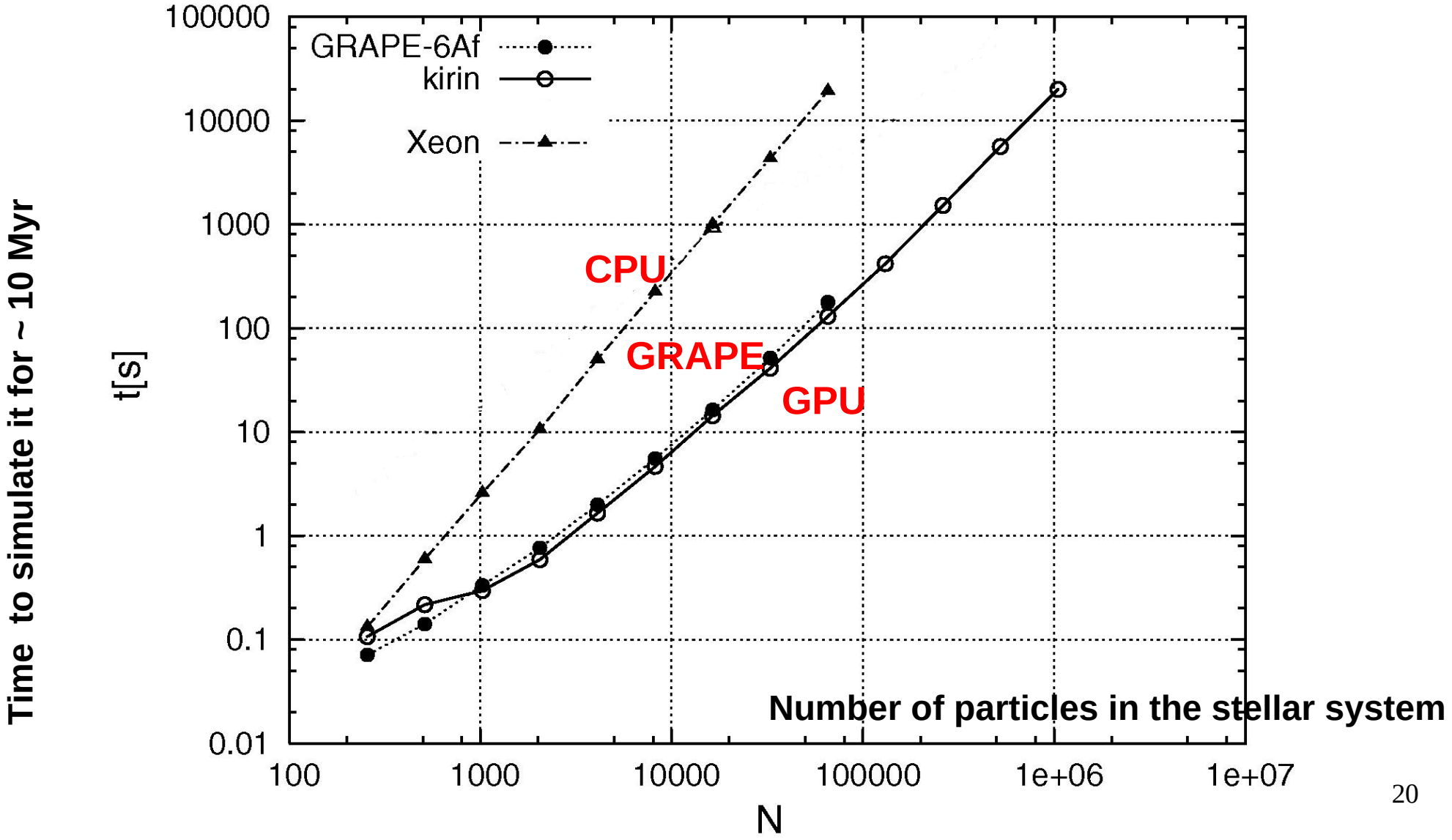
2001: GRAPE-6 at double pr.

A single GRAPE-6 boards runs at 1 Tflop
A 4-cabinet (with 8 GRAPE-6 boards each) at 32 Tflop

GRAPE-8 was in project but.....

Special-purpose hardware. Grape and GPUs

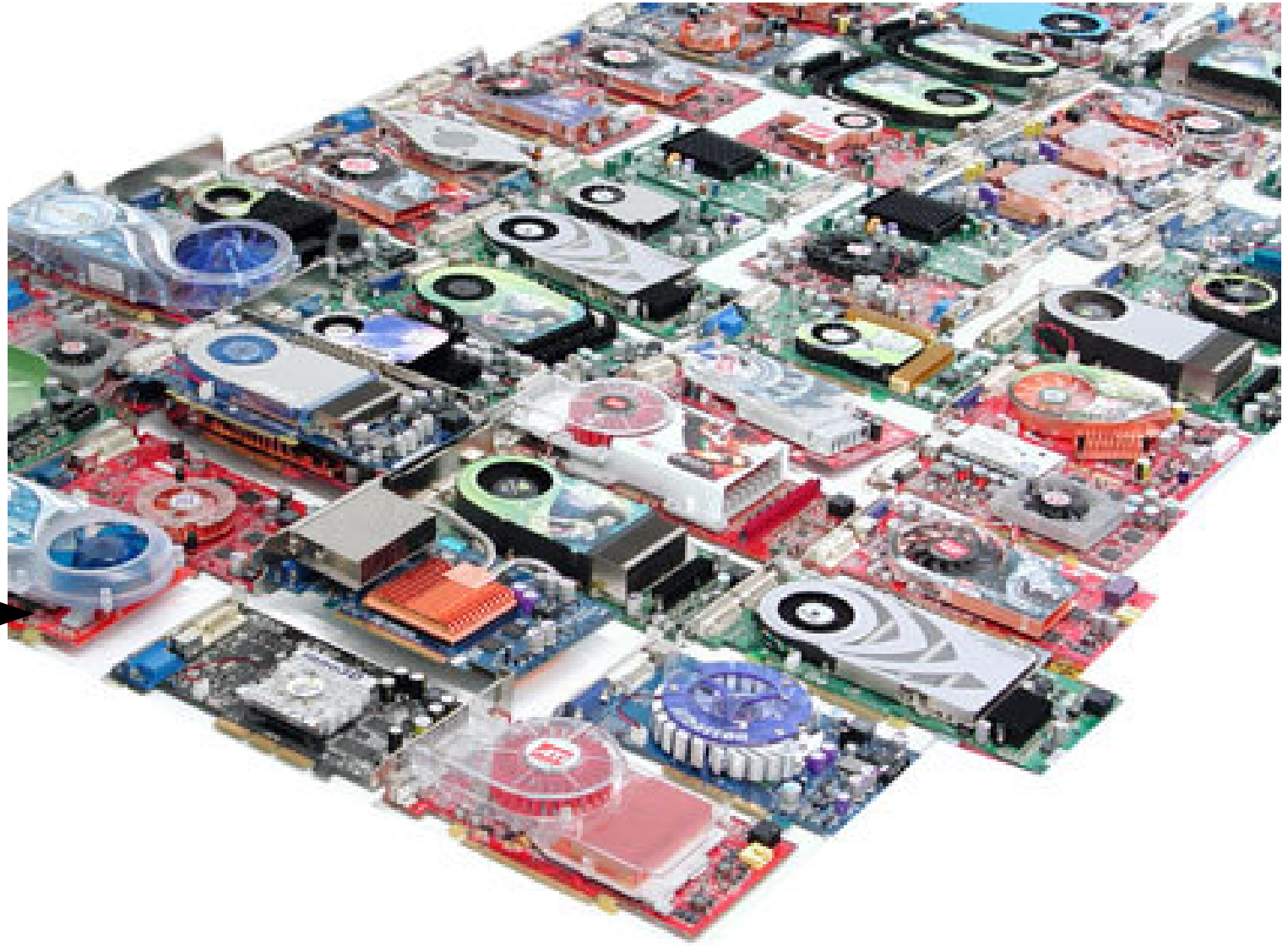
In 2004-2008, researchers found that GPUs are at least as fast as GRAPES for direct N-body codes (Portegies Zwart et al. 2007; **Belleman et al. 2008**; Gaburov et al. 2009)



Special-purpose hardware. Grape and GPUs

Wikipedia's definition of a GPU: specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display

Mostly graphics accelerator of the VIDEO CARD, but in some PC are in the MOTHERBOARD

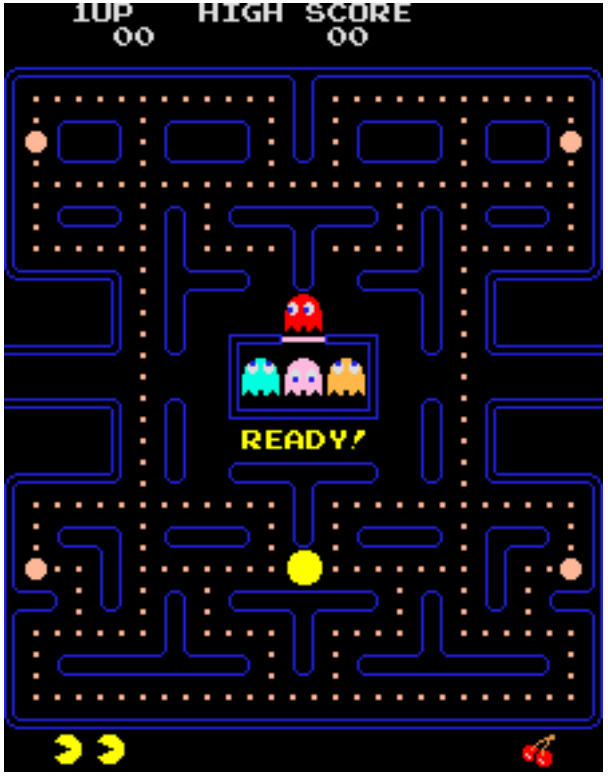


VIDEO CARDS WITH GPUS

Special-purpose hardware. Grape and GPUs

Born for applications that need FAST and HEAVY GRAPHICS: VIDEO GAMES

BEFORE GPU



AFTER GPU

Special-purpose hardware. Grape and GPUs

In ~2004 GPUS WERE FOUND TO BE USEFUL FOR CALCULATIONS:

- first N-body simulations (2nd order) by Nyland et al. (2004)
- first GPU implementation of **Hermite** scheme by **Portegies Zwart et al. (2007)**
- **molecular dynamics** on GPU (Anderson et al. 2008; van Meel et al. 2008)
- Kepler's equation (Ford 2009)
- many more N-body:

  Cunbody (Hamada & Iitaka 2007),

  kirin (Belleman et al. 2008),

  Yebisu (Nitadori & Makino 2008; Nitadori 2009),

  HiGPUs (Capuzzo-Dolcetta et al. 2013),

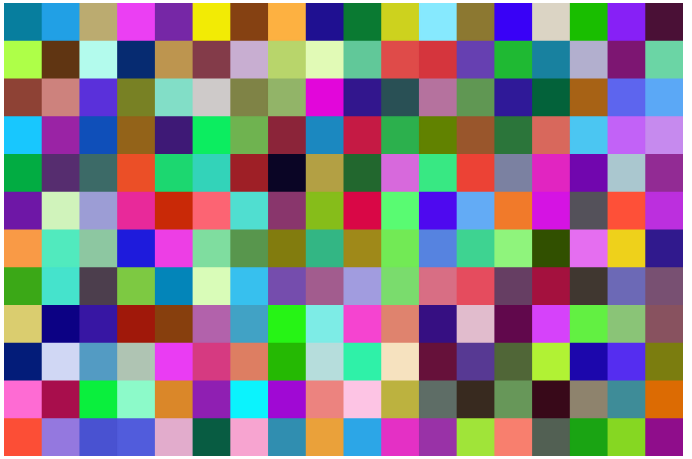
  Sapporo (Gaburov et al. 2009, Bedorf et al. 2015),

  NBODY6++GPU (Wang+ 2015, https://github.com/nbodyx/Nbody6ppGPU)

  PeTar (Wang et al. 2020, https://github.com/lwang-astro/PeTar)

WHY?

Special-purpose hardware. Grape and GPUs



SIMPLE IDEA:

coloured pixel represented by 4 numbers

(R, G, B and transparency)

each pixel does not need information about other pixels (near or far)

→ **when an image must be changed each single pixel can be updated INDEPENDENTLY of the others and SIMULTANEOUSLY to the others**

→ **GPUs are optimized to perform MANY SMALL OPERATIONS (change a single pixel) SIMULTANEOUSLY i.e. MASSIVELY PARALLEL**

THIS IS THE CONCEPT OF **SIMD** TECHNIQUE:

SINGLE INSTRUCTION MULTIPLE DATA

GPUS are composed of many small threads, each able to perform a small instruction, which is the same for all threads but applied on different data

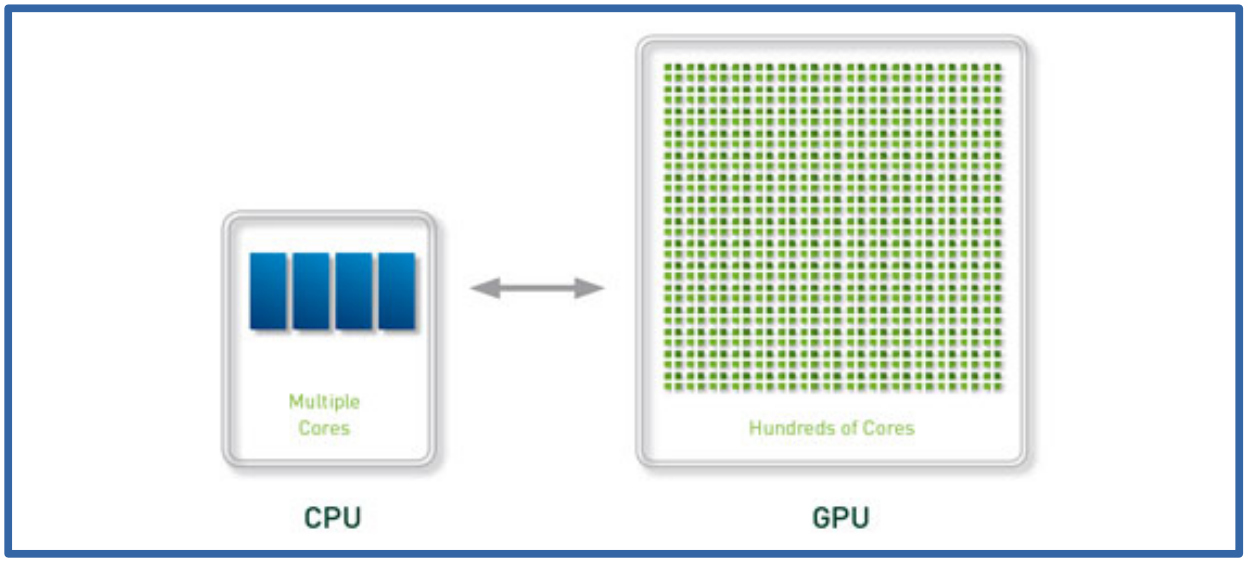
→ NVIDIA calls it **SIMT**= single instruction multiple **THREAD**

Special-purpose hardware. Grape and GPUs

SIMD/SIMT TECHNIQUE: SINGLE INSTRUCTION MULTIPLE DATA/THREADS

many processing units perform the same series of operations

on different sub-samples of data



Even current CPUs are multiple CORES (i.e. can be multi-threading)

but the number of independent cores in GPUs is ~100 times larger!

1M $ QUESTION: WHY IS THIS PARTICULARLY GOOD FOR DIRECT N-BODY CODES?

Special-purpose hardware. Grape and GPUs

SIMD TECHNIQUE: SINGLE INSTRUCTION MULTIPLE DATA

WHY IS THIS PARTICULARLY GOOD FOR DIRECT N-BODY CODES?

BECAUSE THEY DO A SINGLE OPERATION

(acceleration and jerk calculation)

on MANY PAIRS of PARTICLES

$$\vec{a}_i = G \sum_{j \neq i} \frac{M_j}{r_{ji}^3} \vec{r_{ij}}$$

EACH INTERPARTICLE FORCE BETWEEN A PAIR IS INDEPENDENT OF THE OTHER PAIRS!!

SINGLE INSTRUCTION: ACCELERATION CALCULATION

MULTIPLE DATA: ~ N^2 PAIRS of STARS

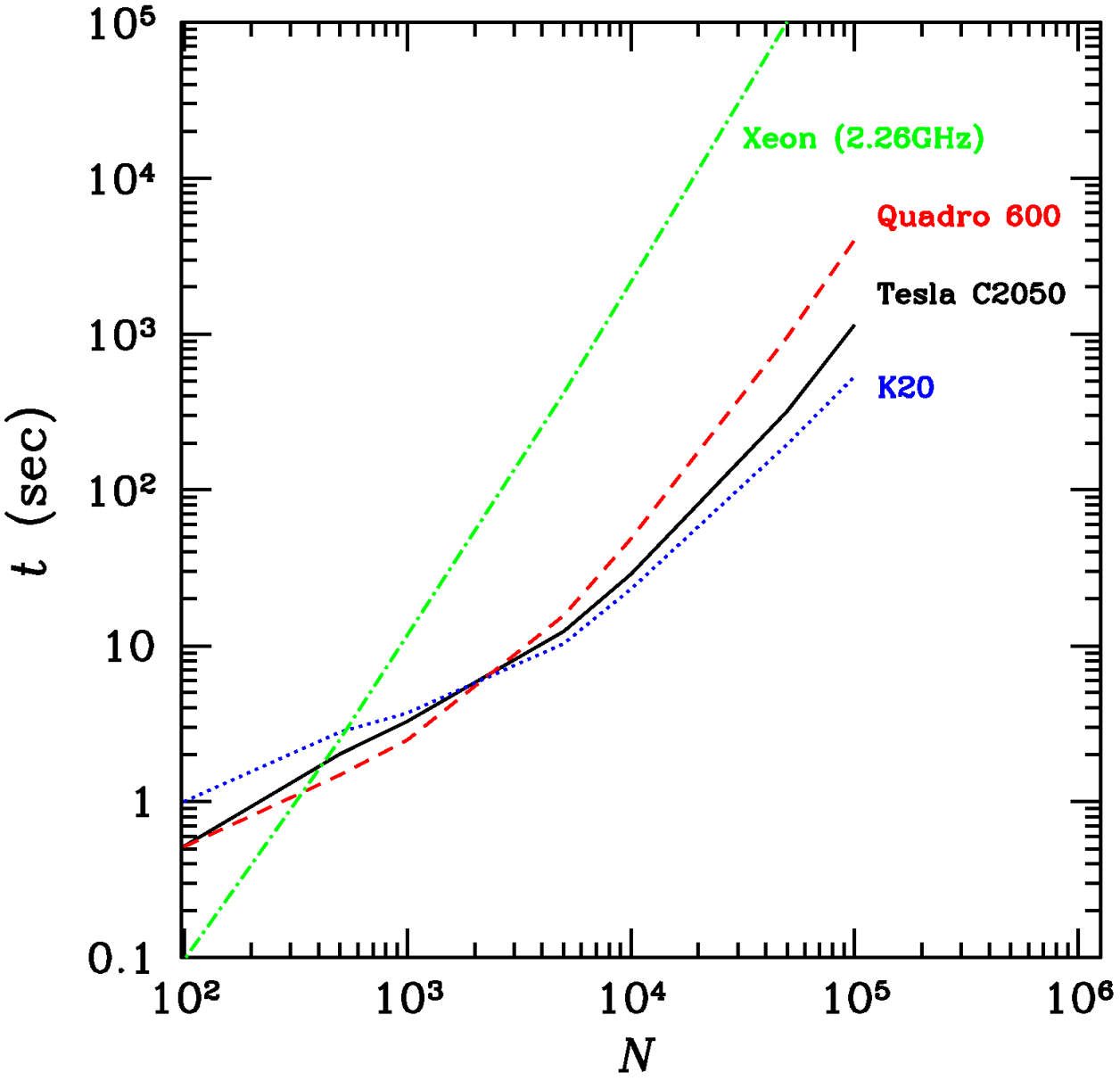
Special-purpose hardware. Grape and GPUs

SIMPLE PERFORMANCE TEST

for a star cluster with N particles:

when N large the number of cores and the possibility to distribute calculations wins

over the power of the single core



YOU CAN RUN YOUR OWN TESTS @ HOME!

References

- Gravitational N-Body simulations, Tools and Algorithms,
Sverre J. Aarseth, ed Cambridge