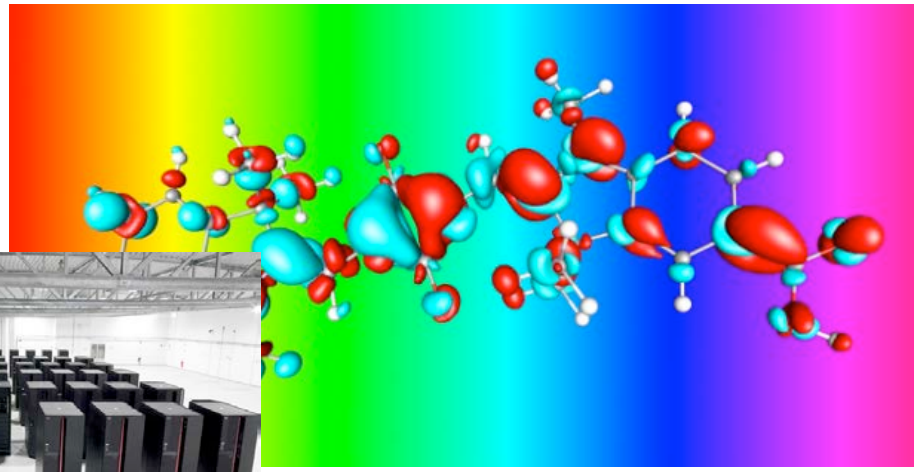
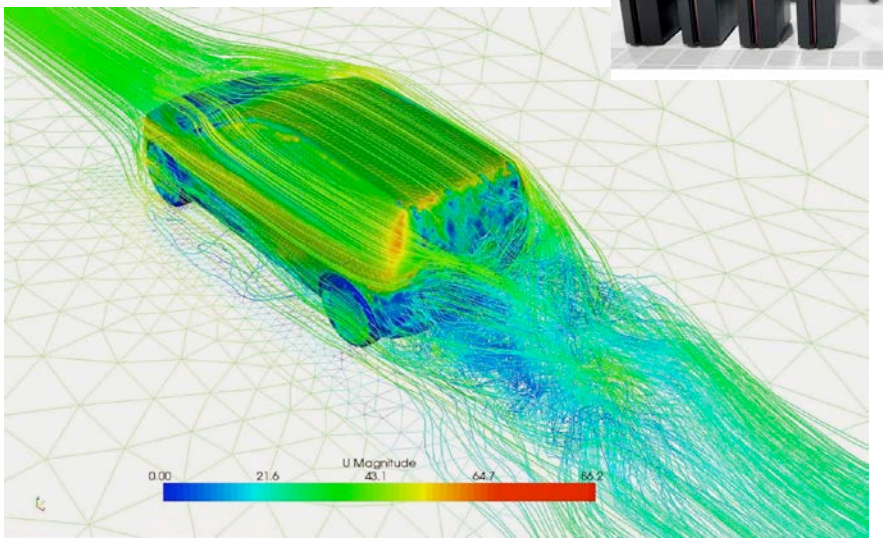
COSMOLOGICAL SIMULATIONS @ OATS

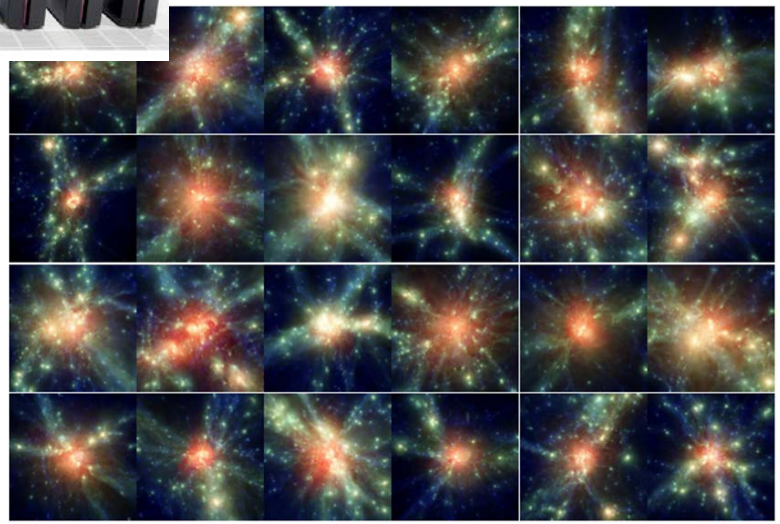
G. Murante S. Borgani G. Taffoni

Why HPC

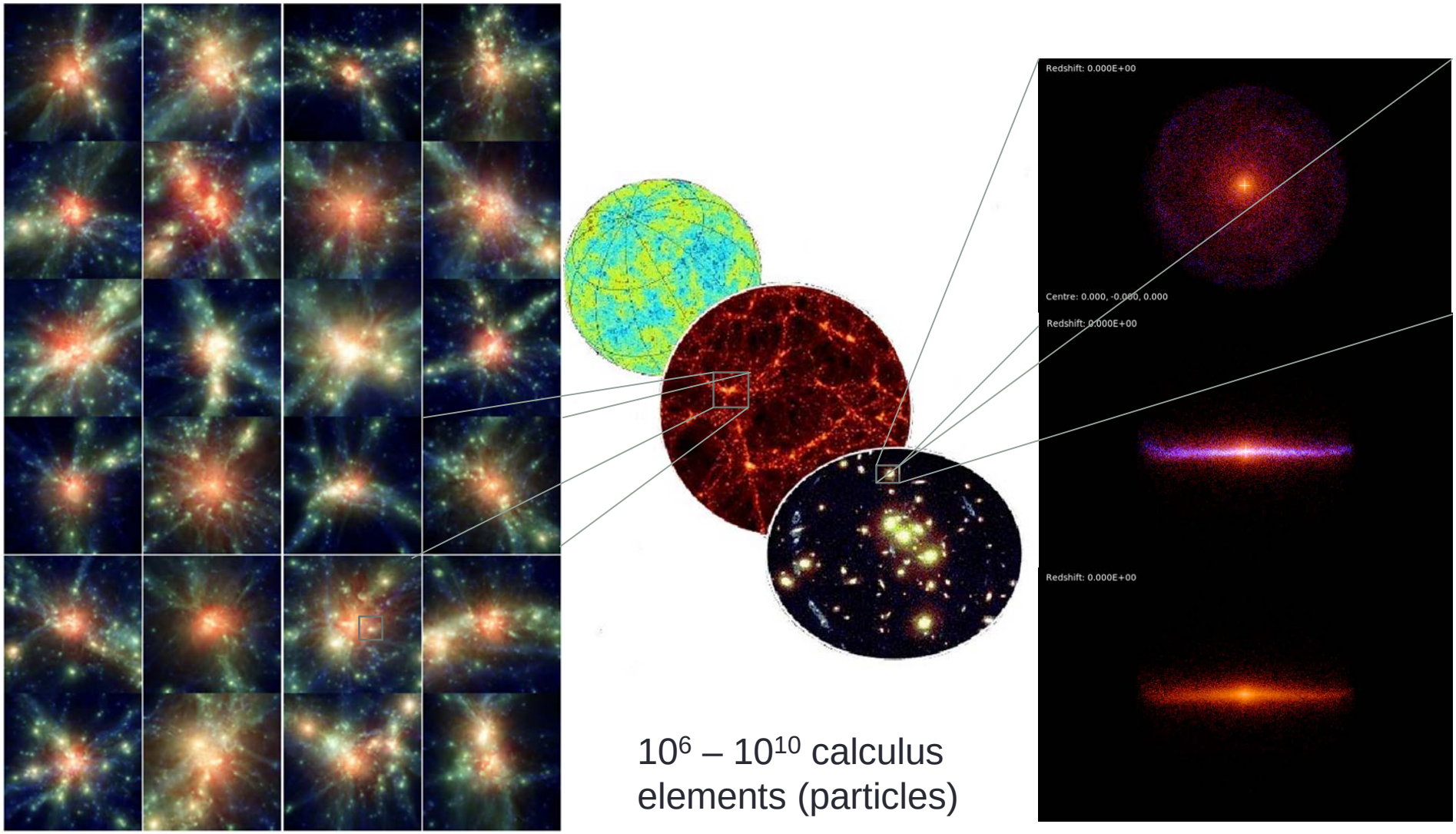
$$r : \rho\left(\frac{\partial u_r}{\partial t} + u_r\frac{\partial u_r}{\partial r} + \frac{u_\phi}{r}\frac{\partial u_r}{\partial \phi} + u_z\frac{\partial u_r}{\partial z} - \frac{u_\phi^2}{r}\right) =$$
$$-\frac{\partial p}{\partial r} + \mu\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u_r}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u_r}{\partial \phi^2} + \frac{\partial^2 u_r}{\partial z^2} - \frac{u_r}{r^2} - \frac{2}{r^2}\frac{\partial u_\phi}{\partial \phi}\right] + \rho g_r$$
$$\phi : \rho\left(\frac{\partial u_\phi}{\partial t} + u_r\frac{\partial u_\phi}{\partial r} + \frac{u_\phi}{r}\frac{\partial u_\phi}{\partial \phi} + u_z\frac{\partial u_\phi}{\partial z} + \frac{u_r u_\phi}{r}\right) =$$
$$-\frac{1}{r}\frac{\partial p}{\partial \phi} + \mu\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u_\phi}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u_\phi}{\partial \phi^2} + \frac{\partial^2 u_\phi}{\partial z^2} +$$







Some example **result**



10^6 – 10^{10} calculus elements (particles)

Algorithms

- **GRAVITY** – long-range, all-to-all calculus elements communication needed (in principle)
- **HYDRODYNAMICS** – short-range, but a small number of calculus elements needs many time steps
- **ASTROPHYSICAL PROCESSES** – (radiative cooling, stqar formaton, black holes evolution, energy exchanges between BH/stars and gas) partially subgrid: the exchange part needs communications

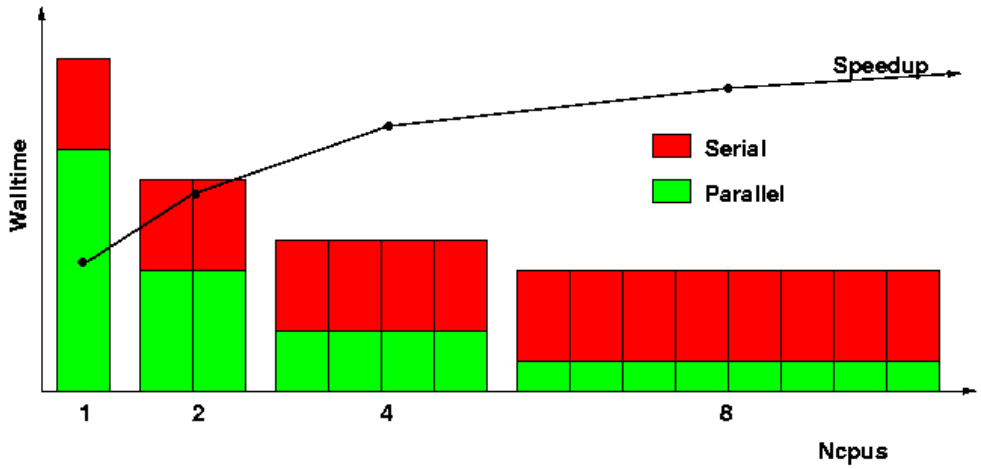
- **CODE** used by our group: **GADGET3** (V. Springel, K. Dolag et al).
- Our group has *access to the international repository*, and is among the **code developers**
- Our group often was a **beta-tester** for supercomputers installed at CINECA, since 2003

HPC computing time

- Most of our CPU time obtained with competitive grants at CINECA (INAF-CINECA convention, ISCRA) and CASPUR
- Two PRACE projects with local PI (developement)
- Involved in several Class-A PRACE projects
- A DECI project under review

Portability, scalability…

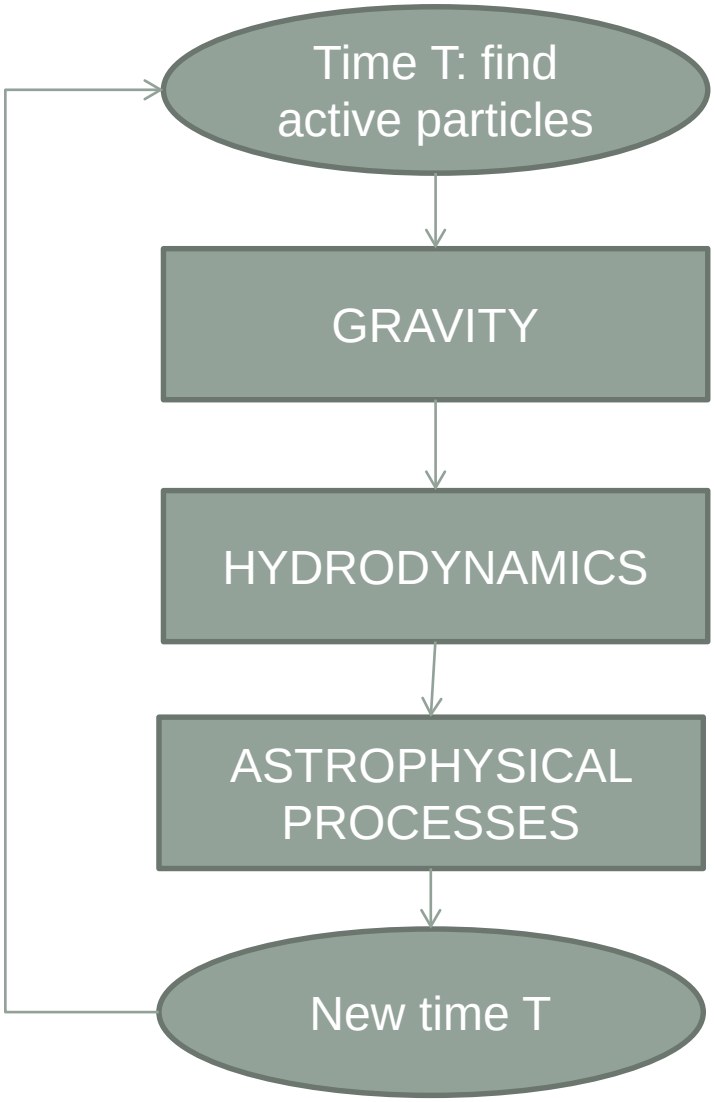
- Our group's simulations run on several machines:
  - Linux clusters (from Beowulf with a 10Mb network to bgp, raijin..)
  - Intel SP3-7
  - Server many-cores shared memory
  - SuperMUC, MareNostrum, Raijin, USC…
  - Plx, Eurora (but: no GPU)
  - …we got troubles with Fermi



On massively parallel architectures we need extreme work-loar balance! Our kind of problem not Very well suited.

(not only us:
Eris run on 512 SP6 cores for 9 months)

Code structure



LOCAL to single task

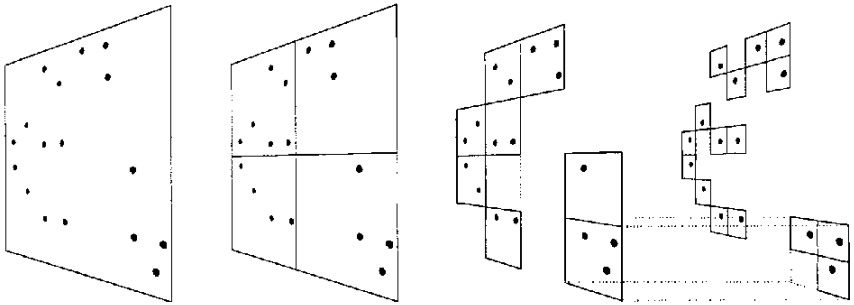
GLOBAL- needs communications

PHYSICALLY local – but needs communications

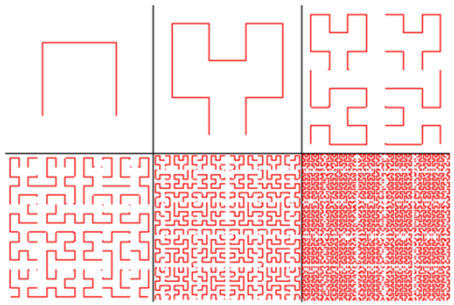
LOCAL (not energy exchanges)

Number of active particles **crucial**!

Code parallelization



A tree is used for gravity computation (approximate, but less communications)



DOMAIN DECOMPOSITION using a Peano space-filling curve: work-load balance at the cost of memory unbalance

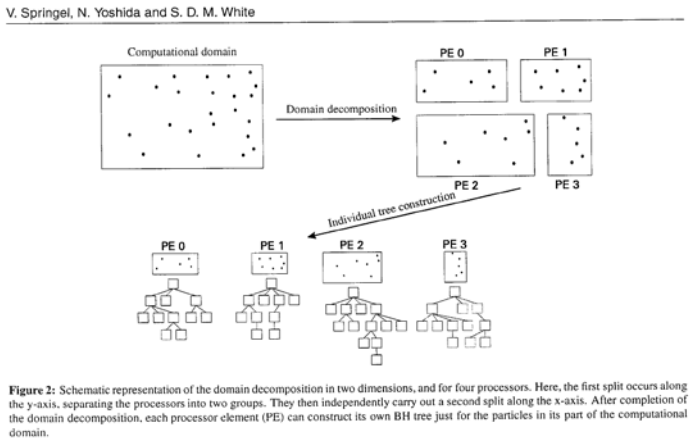


Figure 2: Schematic representation of the domain decomposition in two dimensions, and for four processors. Here, the first split occurs along the y-axis, separating the processors into two groups. They then independently carry out a second split along the x-axis. After completion of the domain decomposition, each processor element (PE) can construct its own BH tree just for the particles in its part of the computational domain.

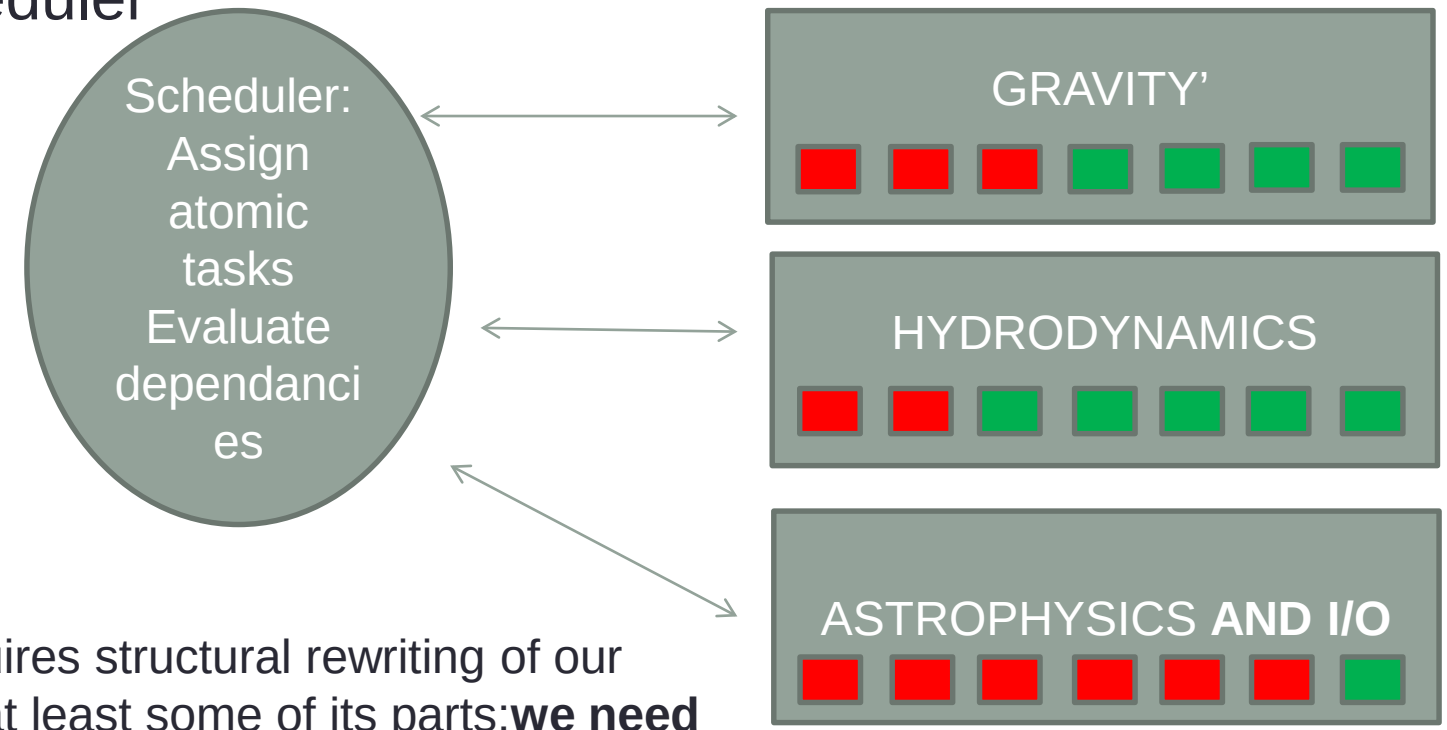
Computation assigned at single MPI tasks. Inside them, OpenMP for shared memory parallelization

Problems with the current HPC computers generation

- **Work-load** balance scheme **costly** in terms of **memory**: a **FEW** MPI tasks allowed for each computing node.
- Inside node, OpenMP parallelization not so efficientNel nodo la parallelizzazione e' fatta con OPENMP: poc
- **I/O** can be **extremely costly** on BlueGene type computers
- In single object/high resolution calculations, **our problem is intrinsecally unbalanced**: a few particles always active (maybe less particles than cores!)

Possible optimizations

- **De-syncronization** of all possible calculations, via algorithm analysis, atomic task and dependance identification, and the use of a client-server kind of scheduler



This requires structural rewriting of our code or at least some of its parts:**we need a software expert**…

Accelerators

- Historical problem with accelerators: they are effective when **flop/byte** is **high**
- ...in our case **flop/byte** is embarassingly **low**: in increasing order, gravity, hydrodynamics, astrophysics
- *Simpler solution: bring astrophysics (and/or hydro?) on accelerator and de-syncronize it*
- Problem: very good syncronization needed between accelerator and CPU calculations
- However, at least partially, a scheme as that described above has to be implemented

Hardware solutions

- In the past: **GRAPE**. Board designed to calculate gravitational interactions. Not extremely successfull.
- **Accelerators**: only solution (?), **increase bandwidth** between CPU and accelerators (or between accelerators).
- The ideal supercomputers for our kind of calculation remains ortogonal to the currend direction of HPC developement: **few CPUs, with a lot of RAM, very powerful**
- En passant, *other scientific communities have similar needs* (climatology, turbulence…)

Conclusions: possible collaborations

- Our group would benefit from a high-level training programme in which one person could deal with code optimization on specific architectures
- Our experience as hardware and software tester can be exploited
- Scientific visualization.