Software performance issues in computational astrophysics

Olivier Iffrig
with Patrick Hennebelle

Astrophysics division, CEA Saclay

HEP Software Foundation Workshop
May 4, 2016

Introduction

A quick overview of computational astrophysics

Two main types of software

- Simulation
  - Instrument simulation

  $\rightarrow$ Physics codes
- Data analysis
  - Real-time instrument control
  - Experiment-specific pipelines

  $\rightarrow$ Simulation results
  - Generic data-crunching software

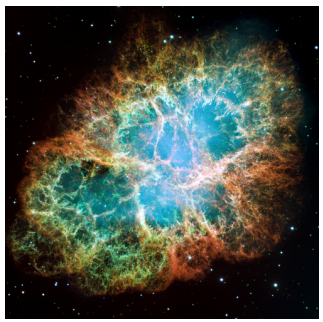
Physical simulation codes

Fundamental ingredients

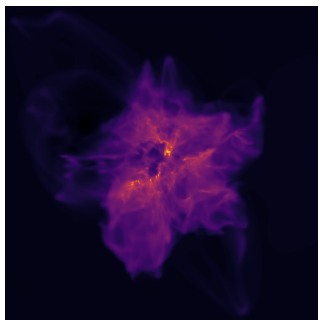
- A physical model
- An algorithm to solve the problem

General evolution problem

$$\frac{\mathrm{d}\mathbb{X}}{\mathrm{d}t}(t) = f(\mathbb{X}(t), t)$$



$$\mathbb{X}_{n+1} = F(\mathbb{X}_n)$$



$\rightarrow$

26.91523170
25.32744789
37.50995255
23.80325317
91.78591156
27.07069778
29.40552711
42.50288010
$\vdots$

The Challenge

The Challenge: example of a fluid dynamics code

Memory

- Often 1000s of points in each direction, 3 dimensions
- About 10 double precision variables

$\Rightarrow$ $\sim$ 80 GB of physical data *per time step*

- Often at least 2 time steps in memory (current and next)
- Other metadata (mesh properties, ...)

Time

- Single-thread: around 10^4 updates per second
- Often 10^4 time steps needed

$\Rightarrow$ 10^9 seconds (a bit less than 32 years) CPU time

Simulations

Massively parallel computing

A typical parallel run

- Several thousands cores
- A few GB per core
- Several 24-hour jobs

$\Rightarrow$ A few TB of memory

$\Rightarrow$ Around 10 days of wall time

© IDRIS

Splitting the data

The MPI point of view

- The same code runs on several processes
- Each process has its own data
- The processes exchange the data at their borders

Bottlenecks

- Communications take time
- Load balancing can be really hard
- Border data is duplicated

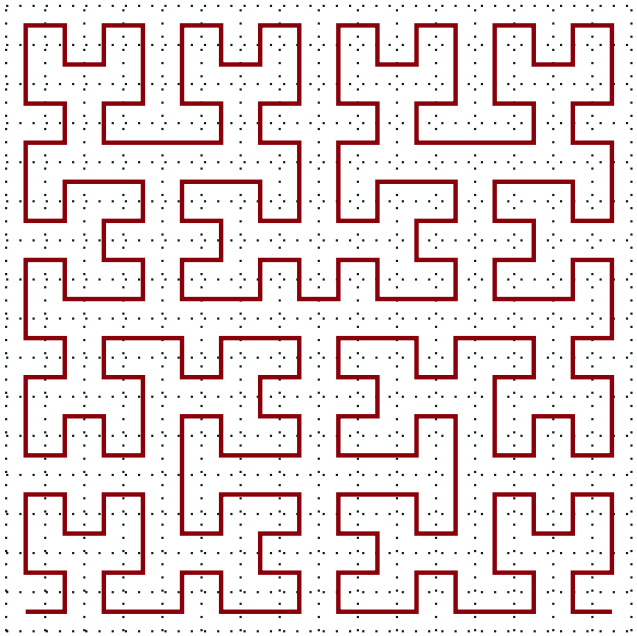
$\Rightarrow$ There is a maximum efficiency (Amdahl's law)

The RAMSES code

`https://bitbucket.org/rteyssie/ramses`

Physics

- (Magneto-) Hydrodynamics
- Self-gravity
- Dark matter, expansion
- Radiative transfer



Numerical

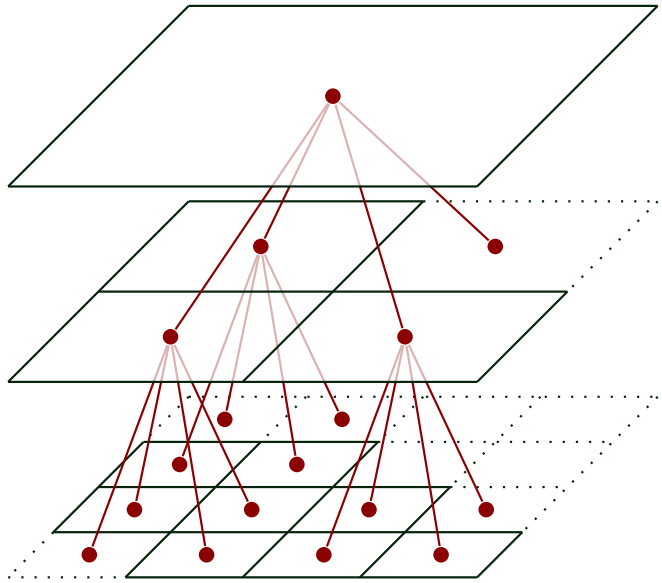
- Domain decomposition
- Adaptive mesh refinement
- 2^{nd}-order solver
- Works on 1000's of cores

The RAMSES code

`https://bitbucket.org/rteyssie/ramses`

Physics

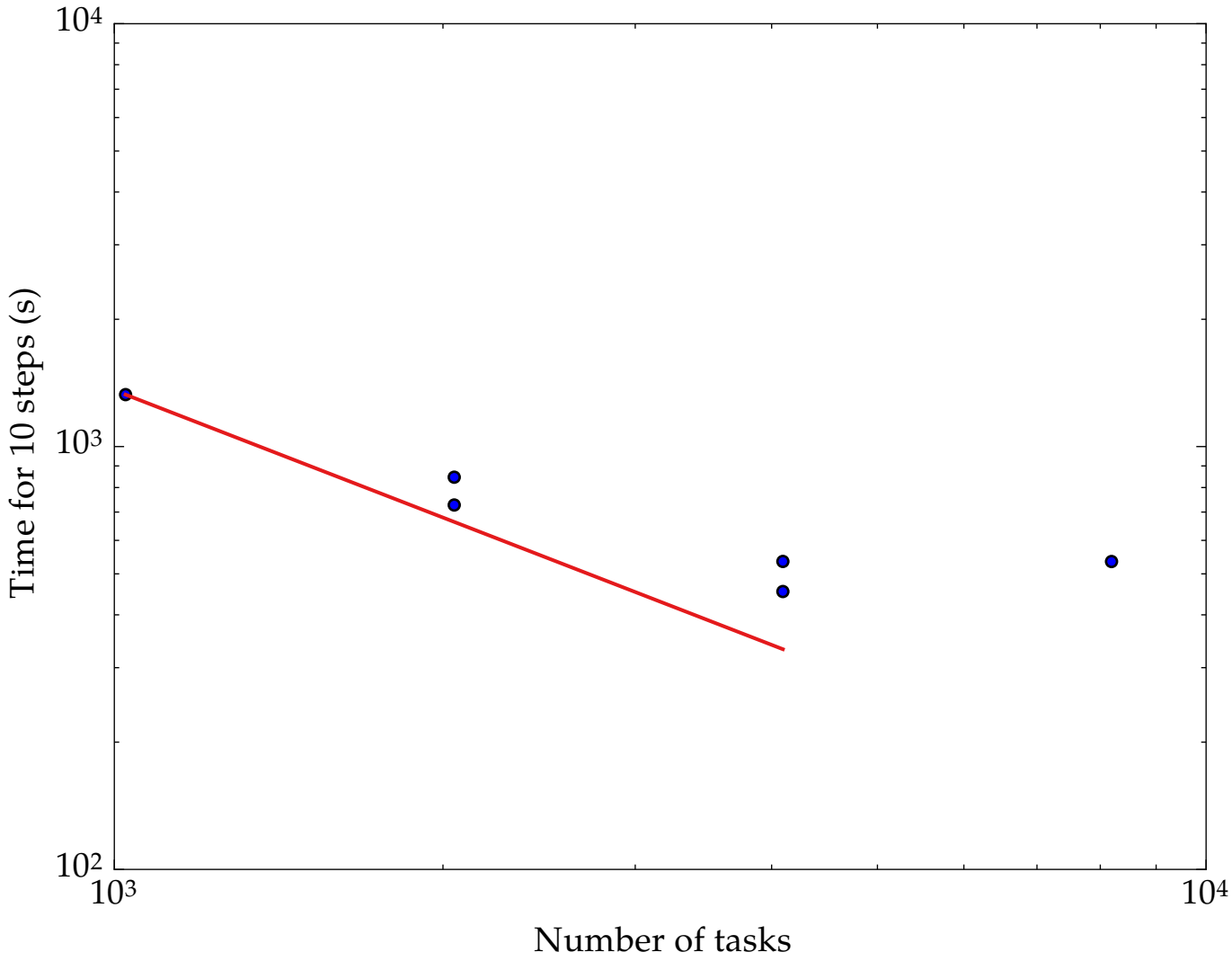
- (Magneto-) Hydrodynamics
- Self-gravity
- Dark matter, expansion
- Radiative transfer



Numerical

- Domain decomposition
- Adaptive mesh refinement
- 2^{nd}-order solver
- Works on 1000's of cores

Scaling



To do better

CPU / Many-core

- Use several threads per MPI task (OpenMP, pthreads, ...)
- Benefit from shared memory

GPU

- Vectorize as much as possible
- Handle data exchange efficiently

Bottlenecks

- Handling non-uniform grids is tricky
- Big production codes are hard to adapt

Post-processing

What to do with those data?

Main issues

- Data size: several 10s of GB per output
- Many algorithms are I/O-bound

"Solutions"

- Read data once and work on it afterwards
- Lots of RAM
- Algorithms taking advantage of the domain decomposition
- Parallel algorithms
- Be patient: batch processing

Conclusion

Conclusion

What we have

- Efficient MPI codes
- Some attempts to use GPU / OpenMP

What's next

- New codes prepared for hybrid architectures (CPU + accelerators)
- Other techniques: task-based parallelism, skeletons, ...

Thanks for your attention!