Mechanic:

numerical framework for cluster simulations

M. Słonina, K. Goździewski, C. Migaszewski

A node

Consider the following

- Large number of initial conditions to test (like an initial orbit)
- Large data processing (from a satellite, a telescope etc.)
- A repeatable task to do

It's easy if computations are really fast, but what if not?

The classical approach

- Create a software for specific case
- Split (by hand) computation to smaller parts
- Handle (by hand) the computations
- Combine (by hand) the results

Assume we have

- 10k simulations, 1 min each
- On 1 2GHz CPU it will take over a week ($\sim$ 170h)
- On 1 2GHz 4-Core CPU it will take 2 days
- We have to combine by hand the results
- We have to take care of the setup and checkpoints

It's full of mistakes, and time consuming

Now, assume each simulation takes an hour…

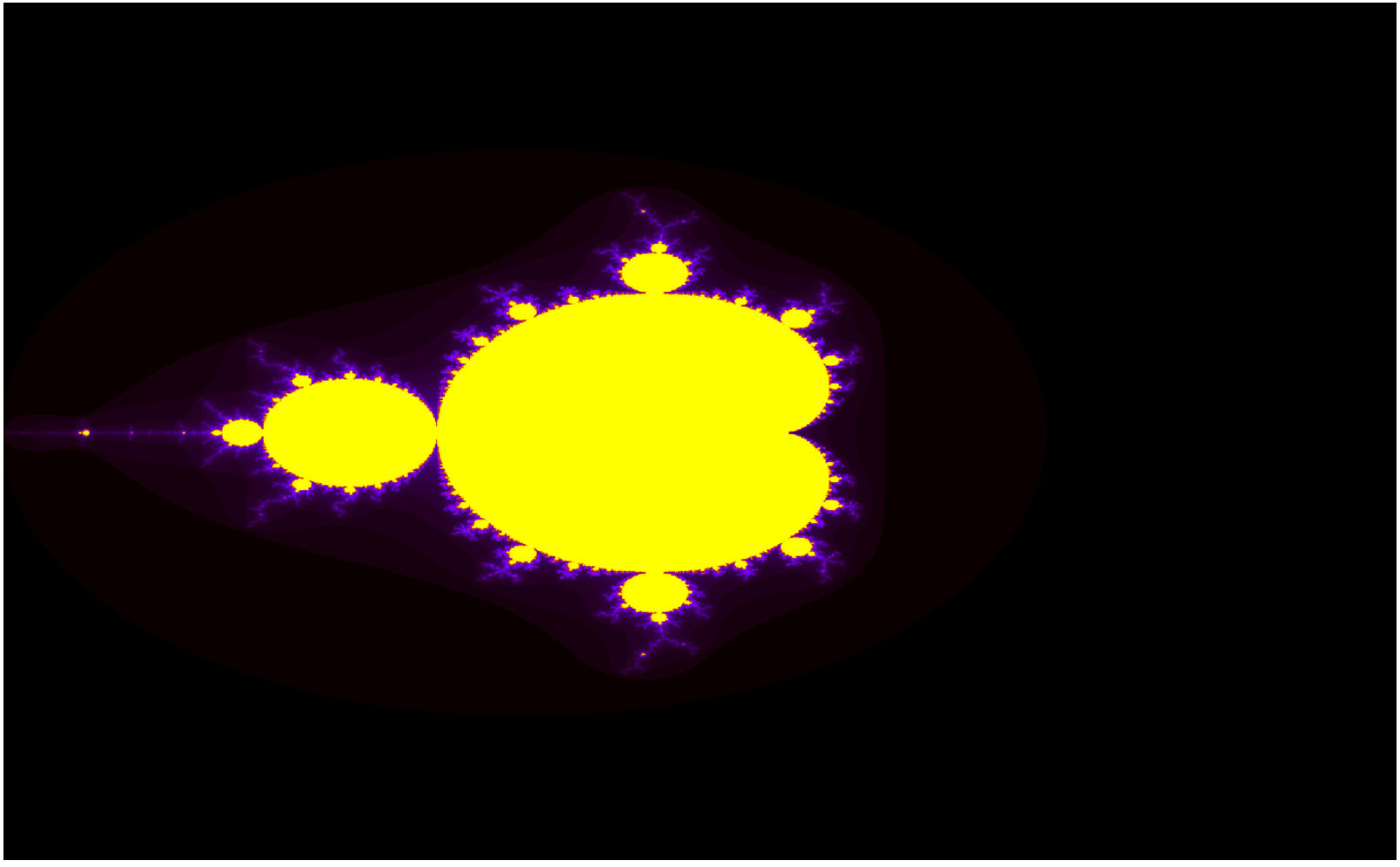
The idea behind

A software that

- Will split our task to smaller parts – **parallelize tasks that could not be parallelized in the classical meaning**
- Will handle computations
- Will combine the output
- Will take care of the setup, storage and checkpoints
- Will be independent on the problem
- Will be as flexible as possible

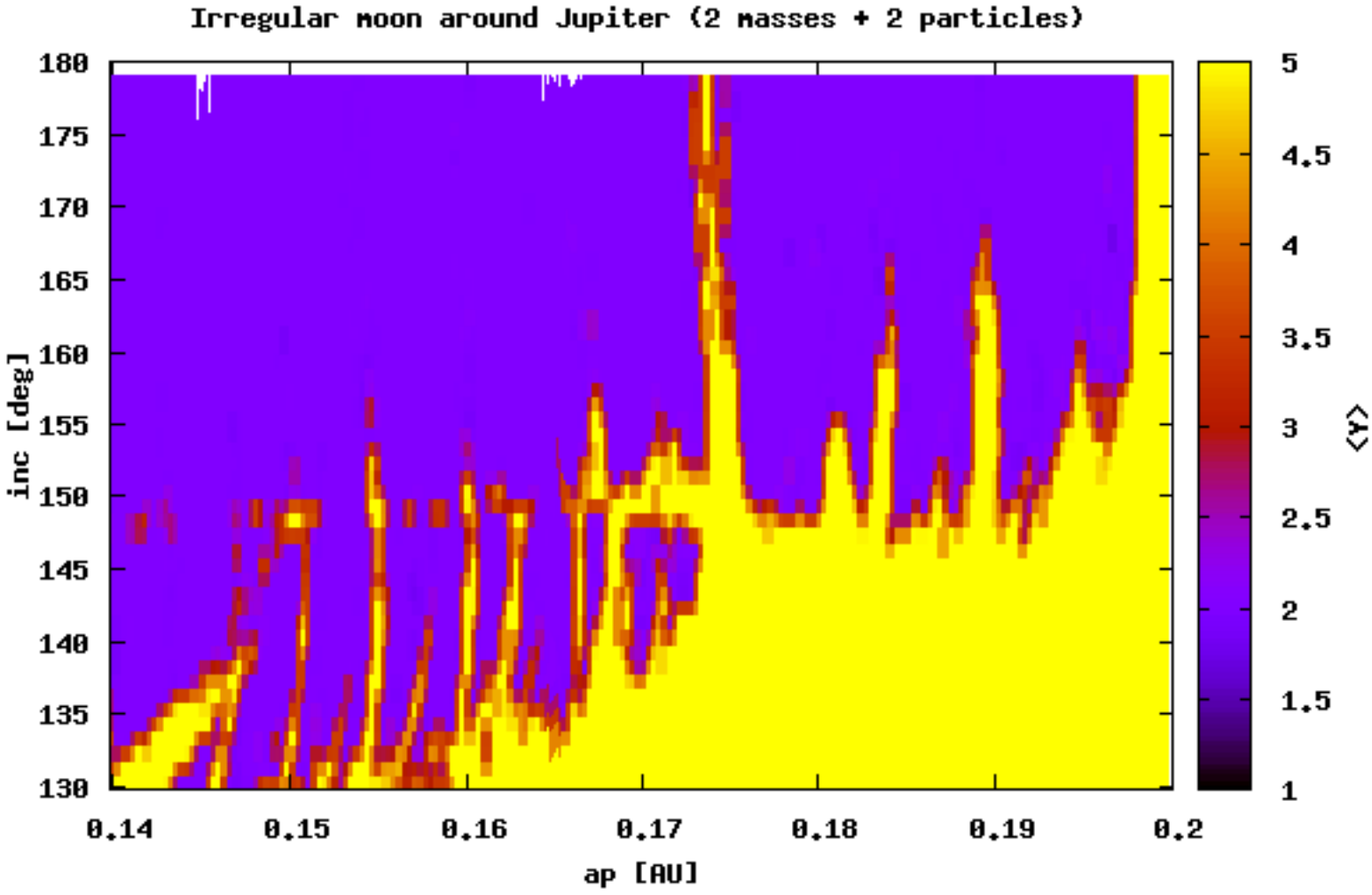
DRY (Don't Repeat Yourself)

The results I



(1+7) Intel Xeon 2.5GHz, 1920x1200px, 2.3 mln simulations, ≈ 20s

The results II



(1+19) AMD Opteron 2GHz, 200x50px, 10k simulations, ≈ 16h

The Mechanic I

- POSIX-compliant numerical framework / interface
- MPI2/HDF5/C Core
- Setup and checkpoint system
- Modes: MPI Task Farm (Spool), Masteralone
- 3-clause BSD license

- The idea is not new, but:
  Is there any other Open Source MPI Farm code?

The Mechanic II

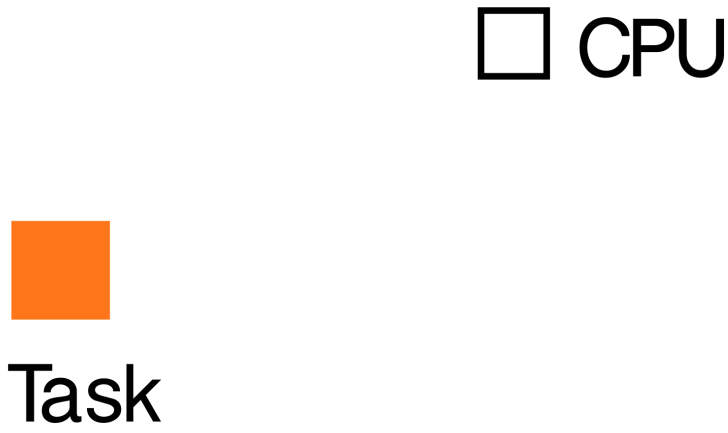
- Loadable module support (user-provided)
- Function template system
- Fortran 2003 bindings
- Tested platforms:
  OpenMPI, MPICH, FakeMPI Environments (1 CPU)

The Mechanic III

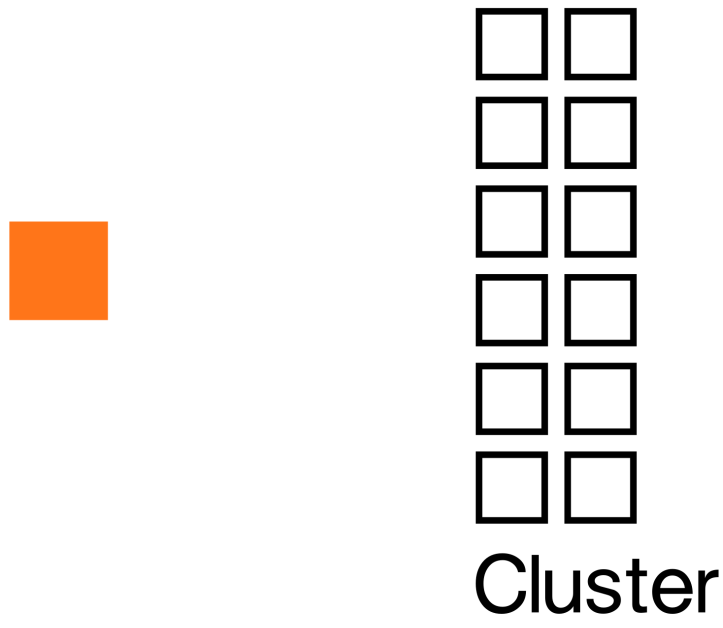
Core:
provide tools for handling computations, independent on the problem

Module:
does the computations, a place where you put your code

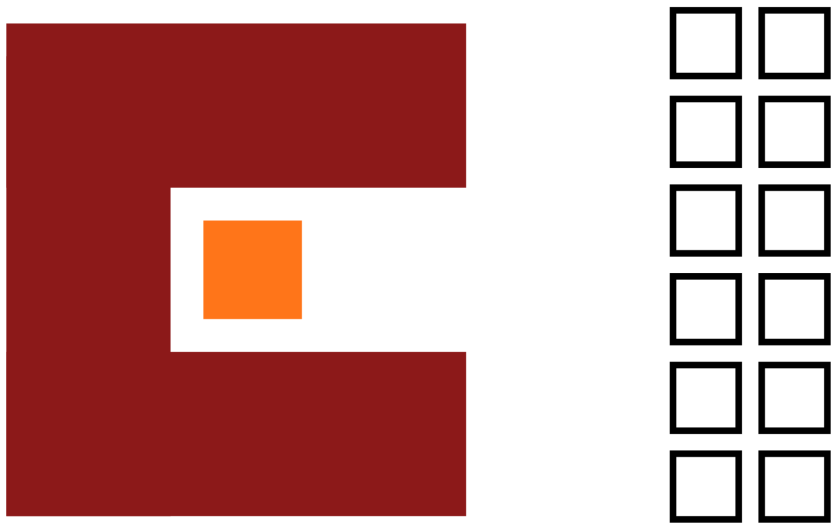
How it works? I



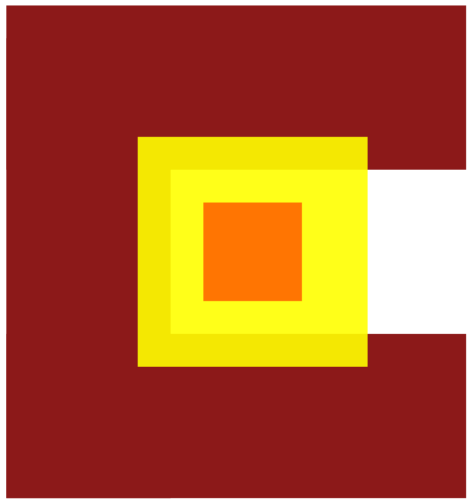
How it works? II



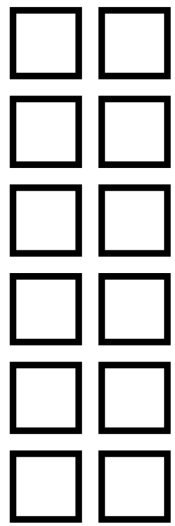
How it works? III



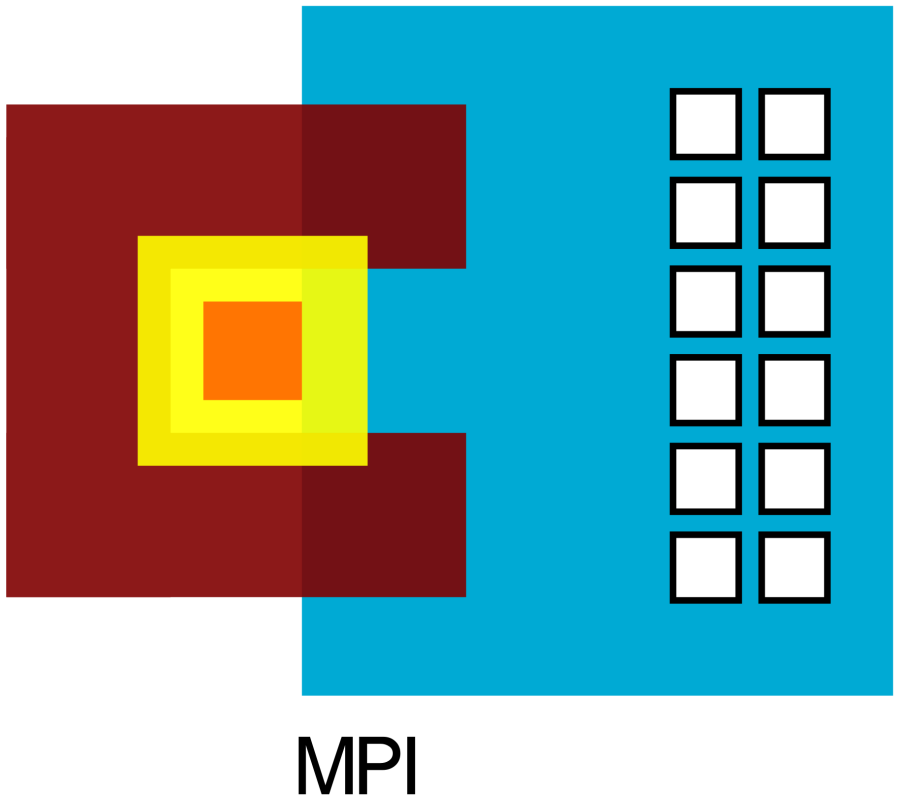
How it works? IV



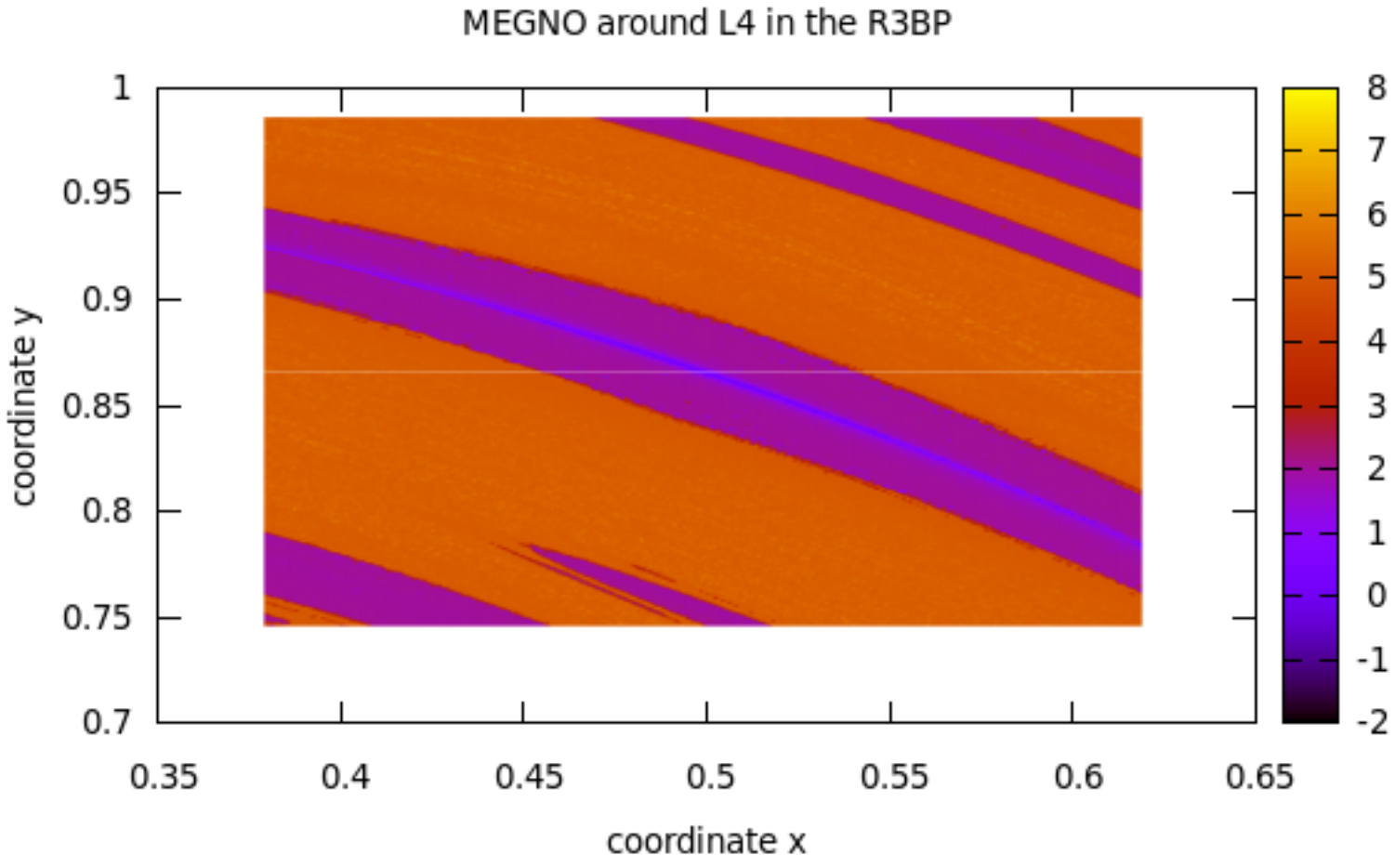
Module



How it works? V

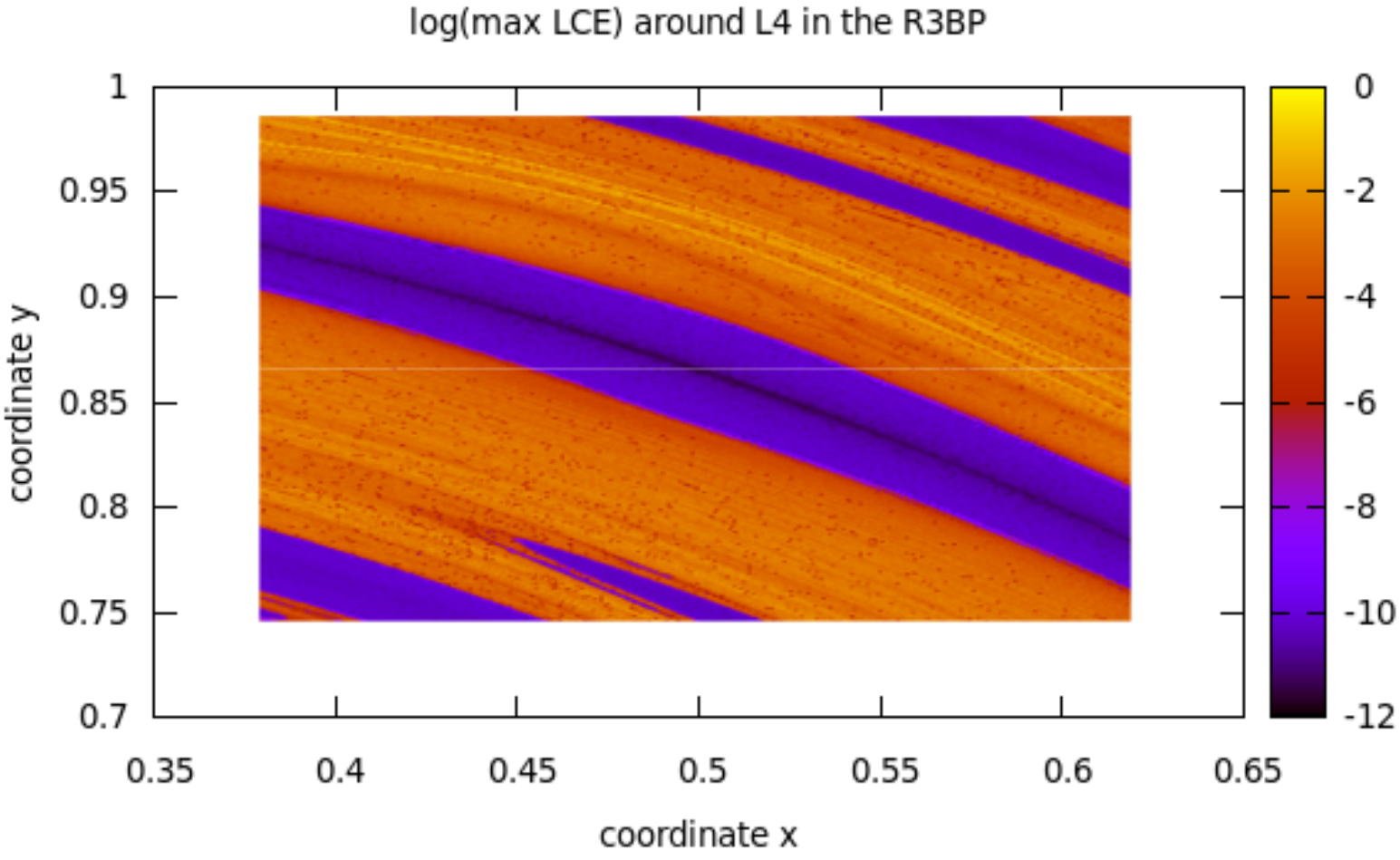


How it works? VI



(1+4) AMD Opteron 2.5GHz CPU 400x400px, $\sim$ 14h

How it works? VII



How it works? VIII

— Example module, **hello**:
`hello_init()`, `hello_cleanup()` and `hello_processPixel()`

— Compilation → shared library `libmechanic_hello.so`

— Run: `mpirun -np 3 mechanic -p hello`

— Modular interface allows to include almost any type of numerical problem (C/F)

— In Spool mode: The master node sends and recieves data

The core – under the hood

```
Setup subsystem (popt, lrc)
Storage subsystem (HDF5)
Checkpoint subsystem
Message / Error subsystem
Simulation board subsystem
Operating mode subsystem
```

The module – under the hood I

`module_init()` (required)

`module_query()`

`module_cleanup()` (required)

`module_farmResolution()`

`module_pixelCoordsMap()`

`module_pixelCoords()`

The module – under the hood II

```
module_node_init()
module_node_in()
module_node_preparePixel()
module_node_processPixel() (required)
module_node_out()
```

The module – under the hood III

```
module_node_beforeProcessPixel()
module_node_afterProcessPixel()
module_node_beforeSend()
module_node_afterSend()
module_node_beforeReceive()
module_node_afterReceive()
```

Function template system

Any `_node_` function can be overriden, i.e.:

`module_node_init()` → `module_master_init()`

`module_node_init()` → `module_slave_init()`

Development

- CUDA-based task farm
- Liborbit (part of MECHANIC core)

- Glusterfs 3.0.4 bug (submitted to mainstream)

Possible usage

— Testing dynamical models (R3BP, ER3BP, NB, …)
— Different initial conditions
— Different dynamical indicators

— Can be combined with genetic algorithms
— Reuse of old code

Put your science into HPC

Where to get it?

http://git.astri.umk.pl/projects/mechanic

git://git.astri.umk.pl/Mechanic

current release: 0.12-unstable-2-6,
which is quite stable and feature complete

Detailed documentation comes with the software

Patches and suggestions are welcome

Acknowledgments

Work supported by

AstroNet – Marie Curie Research Training Network
European Community's 6^{th} Framework Programme

Thank You