Status Update

CAST group retreat, 01.10.2019, Prien/Chiemsee

Existing Packages:

— Two Language problem: Python + C++
— Slow: e.g. Python slow in reading RAMSES data
— Fast, command line, difficult to extend
— use interpolated full grid —> huge memory consumption

- Effort to program personal extensions
- Not shared with community

Solution:

— one language
— high level
— easy to use, extend and share with community
— interactive
— fast
— memory leightweight (AMR)
— easy to install

- Workflow enables to concentrate more on science
- MERA can be used on with other grid-based or N-body codes

Status Update

MERA

Based on a database

Multiple dispatch

Predefined quantities

Macros

Fast projections

Compressed MERA files

Detailed documentation

Platform to easily share your notebooks

Little paper

```
[15]: select(gas.data, (:level,:cx, :cy, :cz, :rho) )

[15]: Table with 50553133 rows, 5 columns:
      level  cx     cy    cz    rho
      ──────────────────────────────────
      6      1      1     1     1.47746e-8
      6      1      1     2     1.47746e-8
      6      1      1     3     1.47746e-8
      6      1      1     4     1.47746e-8
      6      1      1     5     1.47746e-8
      6      1      1     6     1.47746e-8
      6      1      1     7     1.47746e-8
      6      1      1     8     1.47746e-8
      6      1      1     9     1.47746e-8
      6      1      1     10    1.47746e-8
      6      1      1     11    1.47746e-8
      6      1      1     12    1.47746e-8
      ⋮
      14     13038  8855  8122  1.41561
      14     13038  8855  8123  1.29724
      14     13038  8855  8124  1.27011
      14     13038  8856  8117  1.89495
      14     13038  8856  8118  1.83967
      14     13038  8856  8119  1.67429
      14     13038  8856  8120  1.59913
      14     13038  8856  8121  1.45658
      14     13038  8856  8122  1.41468
      14     13038  8856  8123  1.3037
      14     13038  8856  8124  1.29413
```

Advantages e.g.:

- easy to filter data
- easy to parallize

Status Update

MERA

Based on a database

Multiple dispatch

Predefined quantities

Macros

Fast projections

Compressed MERA files

Detailed documentation

Platform to easily share your

Little paper

One single function name
For different data-types

gas

```
# exported
mutable struct HydroDataType <: DataSetType
    data::JuliaDB.AbstractIndexedTable #todo: check
    info::InfoType
    lmin::Int
    lmax::Int
    boxlen::Float64
    ranges::Array{Float64,1}
    selected_hydrovars::Array{Int,1}
    used_descriptors::Dict{Any,Any}
    smallr::Float64
    smallc::Float64
    scale::ScalesType
    HydroDataType() = new()
end
```

```
function projection(   dataobject::HydroDataType, vars::Array{Symbol,1};
                        units::Array{Symbol,1}=[:standard],
                        #coordinates::Symbol=:cartesian,
                        lmax::Number=dataobject.lmax,
                        mask=[false],
                        direction::Symbol=:z,
                        plane_orientation::Symbol=:perpendicular,
                        weighting::Bool=true,
                        mode::Symbol=:weighting,
                        gamm::Number=1.4,
                        xrange::Array{<:Any,1}=dataobject.ranges[1:2],
                        yrange::Array{<:Any,1}=dataobject.ranges[3:4],
                        zrange::Array{<:Any,1}=dataobject.ranges[5:6],
                        center::Array{<:Number,1}=[0., 0., 0.],
                        range_units::Symbol=:standard,
                        data_center::Array{<:Number,1}=[0.5, 0.5, 0.5],
                        data_center_units::Symbol=:standard,
                        verbose::Bool=verbose_mode)
```

Particles

```
# exported
mutable struct PartDataType <: DataSetType
    data::JuliaDB.AbstractIndexedTable #todo:check
    info::InfoType
    lmin::Int
    lmax::Int
    boxlen::Float64
    ranges::Array{Float64,1}
    selected_partvars::Array{Symbol,1}
    scale::ScalesType
    PartDataType() = new()
end
```

```
function projection(   dataobject::PartDataType, vars::Array{Symbol,1},
                            units::Array{Symbol,1};
                            parttypes::Array{Symbol,1}=[:stars],
                            coordinates::Symbol=:cartesian,
                            lmax::Int=9,
                            mask=[false],
                            direction::Symbol=:z,
                            plane_orientation::Symbol=:perpendicular,
                            mode::Symbol=:mass,
                            xrange::Array{<:Any,1}=dataobject.ranges[1:2],
                            yrange::Array{<:Any,1}=dataobject.ranges[3:4],
                            zrange::Array{<:Any,1}=dataobject.ranges[5:6],
                            center::Array{<:Number,1}=[0., 0., 0.],
                            range_units::Symbol=:standard,
                            data_center::Array{<:Number,1}=[0.5, 0.5, 0.5],
                            data_center_units::Symbol=:standard,
                            verbose::Bool=verbose_mode)
```

Status Update

MERA

Based on a database

Multiple dispatch

Predefined quantities

Macros

Fast projections

Compressed MERA files

Detailed documentation

Platform to easily share your notebooks

Little paper

```
projection()

Predefined Quantities for Projections:
-----------------------------------------------
=====================[gas]:=====================
       -all the non derived hydro vars-
:cpu, :level, :rho, :cx, :cy, :cz, :vx, :vy, :vz, :p, var6,...

              -derived hydro vars-
:x, :y, :z
:sd or :Σ or :surfacedensity
:mass, :cellsize, :freefall_time
gamm needed => :cs, :mach, :jeanslength, :jeansnumber

==================[particles]:==================
        all the non derived  vars:
:cpu, :level, :id, :x, :y, :z, :vx, :vy, :vz, :mass, :age,....

==============[gas or particles]:===============
:v, :ekin
squared => :vx2, :vy2, :vz2
velocity dispersion => σx, σy, σz, σ

related to a given center:
---------------------------
:vr_cylinder, vr_sphere
:vϕ, :vθ
squared => :vr_cylinder2, :vϕ2
velocity dispersion => σr_cylinder, σϕ
:l, :lx, :ly, :lz :lr, :lϕ, :lθ

2d maps:
:r_cylinder, :r_sphere
:ϕ, :θ
-----------------------------------------------
```

Quantities internally defined,
And can be used by many functions
In MERA or extended by the user.

Status Update

MERA

Based on a database

Multiple dispatch

Predefined quantities

Macros

Fast projec

Compresse

Detailed d

Platform to

Little pape

Use Pipeline Macros

```
boxlen = info.boxlen
cv = boxlen/2.
density = 3. /gas.scale.Msol_pc3
radius  = 3. /gas.scale.kpc
height  = 2. /gas.scale.kpc

filtered_db = @apply gas.data begin
     @where :rho >= density
     @where sqrt( (:cx * boxlen/2^:level - cv)^2 + (:cy * boxlen/2^:level - cv)^2 ) <= radius
     @where abs(:cz * boxlen/2^:level -cv) <= height
end

var_filtered = getvar(gas, :mass, filtered_db=filtered_db, unit=:Msol)
sum(var_filtered) # [Msol]
```

Status Update

MERA

Based on a database

Multiple dispatch

Predefined quantities

Macros

Fast projections

Compressed MERA files

Detailed documentation

Platform to easily share your noteb

Little paper

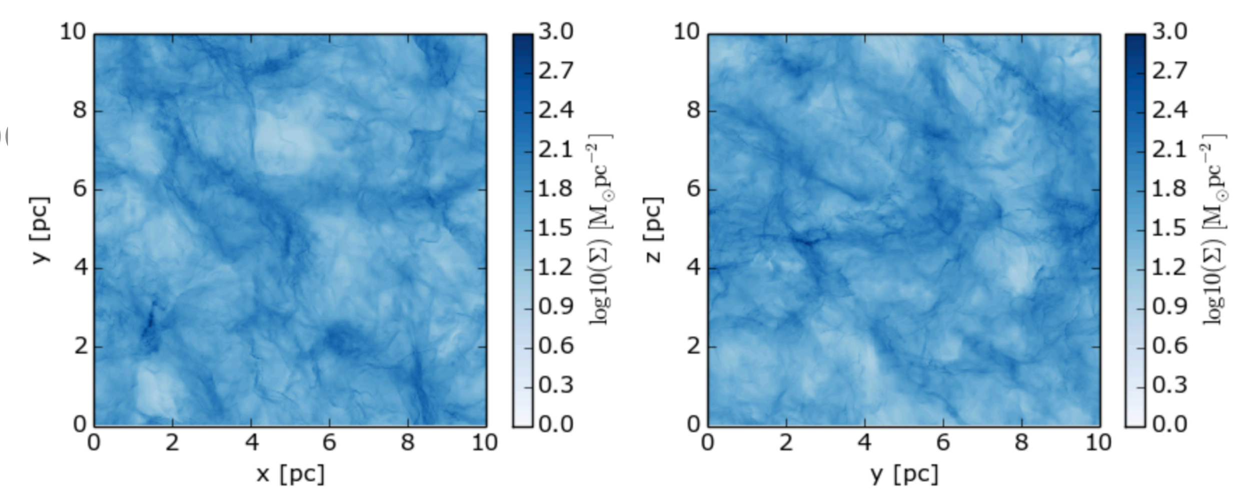
Is fast due to :
- AMR data only
- Julia
- and a trick

Fast high resolution projections

onto a (2^14)^2 grid !!!

```
proj_z = projection(gas, vars=[:sd], direction=:z);
[Mera]: 2018-12-29T16:54:18.666

domain
xmin::xmax: 0.0 :: 1.0          ==> 0.0 [pc] :: 10.0 [pc]
ymin::ymax: 0.0 :: 1.0          ==> 0.0 [pc] :: 10.0 [pc]
zmin::zmax: 0.0 :: 1.0          ==> 0.0 [pc] :: 10.0 [pc]

Selected vars: Symbol[:sd]
100%|██████████████████████████████████████████| Time: 0:02:08
proj_x = projection(gas, vars=[:sd], direction=:x);
[Mera]: 2018-12-29T16:56:29.763

domain
xmin::xmax: 0.0 :: 1.0          ==> 0.0 [pc] :: 10.0 [pc]
ymin::ymax: 0.0 :: 1.0          ==> 0.0 [pc] :: 10.0 [pc]
zmin::zmax: 0.0 :: 1.0          ==> 0.0 [pc] :: 10.0 [pc]

Selected vars: Symbol[:sd]
100%|██████████████████████████████████████████| Time: 0:02:10
```

Moeckels et al. - turbulent box



Status Update

MERA

Based on a database

Multiple dispatch

Predefined quantities

Macros

Fast projections

Compressed MERA files

Detailed documentation

Platform to easily share your notebooks

Little paper

Reduces significantly
- disc space: factor 5 (my simulations)
- data loading: factor 17 (2048 cores sim)

-> good for time-sequence Post-processing

Status Update

Based on a database

Multiple dispatch

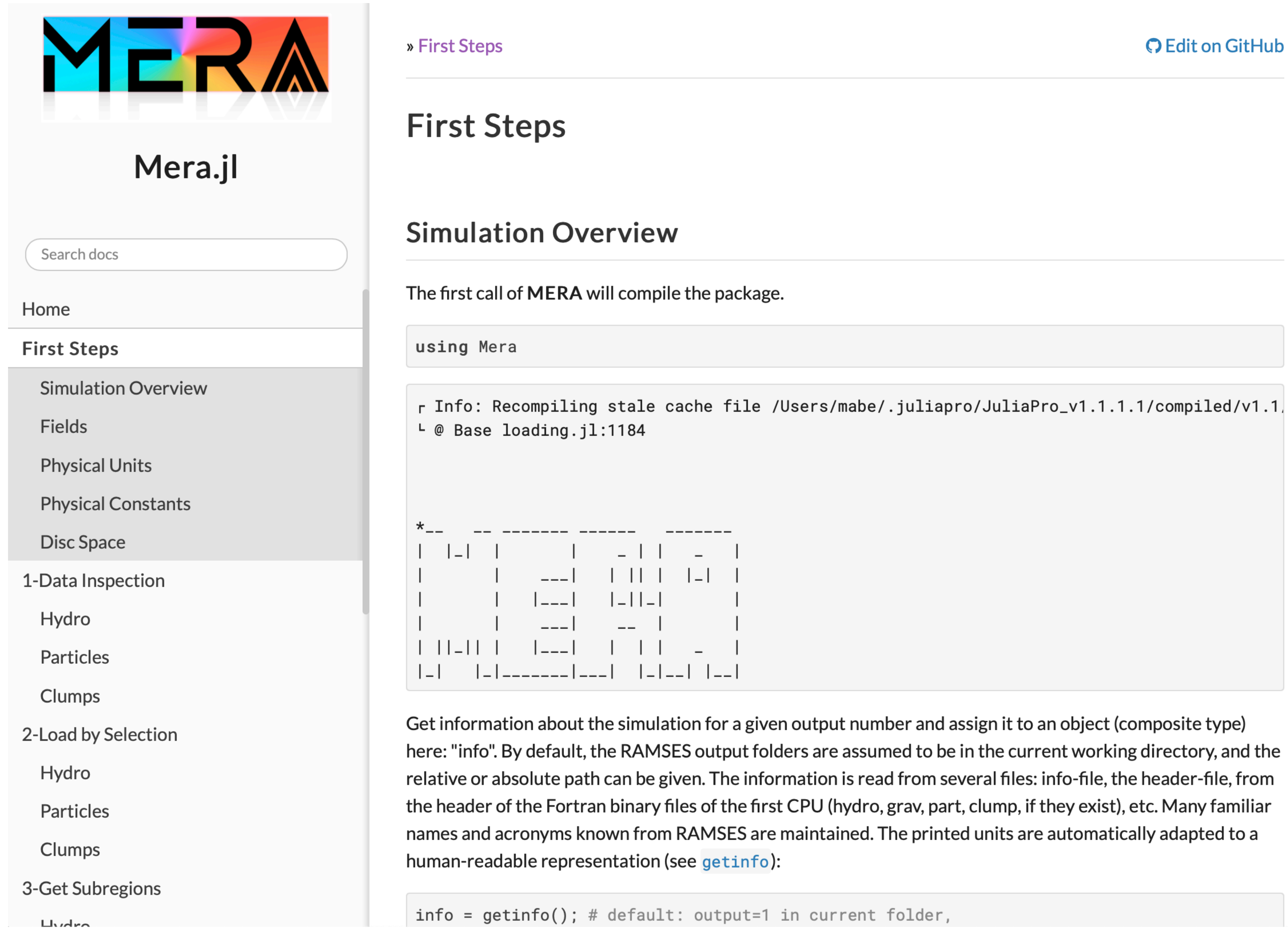
Predefined quantities

Macros

Fast projections

Compressed MERA files

Detailed documentation



Mera.jl

Search docs

- Home
- First Steps
  - Simulation Overview
  - Fields
  - Physical Units
  - Physical Constants
  - Disc Space
- 1-Data Inspection
  - Hydro
  - Particles
  - Clumps
- 2-Load by Selection
  - Hydro
  - Particles
  - Clumps
- 3-Get Subregions
  - Hydro

» First Steps

Edit on GitHub

First Steps

Simulation Overview

The first call of **MERA** will compile the package.

```
using Mera
```

```
┌ Info: Recompiling stale cache file /Users/mabe/.juliapro/JuliaPro_v1.1.1.1/compiled/v1.1
└ @ Base loading.jl:1184
```

Get information about the simulation for a given output number and assign it to an object (composite type) here: "info". By default, the RAMSES output folders are assumed to be in the current working directory, and the relative or absolute path can be given. The information is read from several files: info-file, the header-file, from the header of the Fortran binary files of the first CPU (hydro, grav, part, clump, if they exist), etc. Many familiar names and acronyms known from RAMSES are maintained. The printed units are automatically adapted to a human-readable representation (see `getinfo`):

```
info = getinfo(); # default: output=1 in current folder,
```

Platform to easily share your notebooks

Little paper

Documentation contains detailed explanations, many examples and Jupyter notebooks to try it on your simulation data.

Status Update

Based on a database

Multiple dispatch

Predefined quantities

Macros

Fast projections

Compressed MERA files

Detailed documentation

Platform to easily share your notebooks

Little paper