

Team Ramses: Day 2



OPEN
HACKATHONS



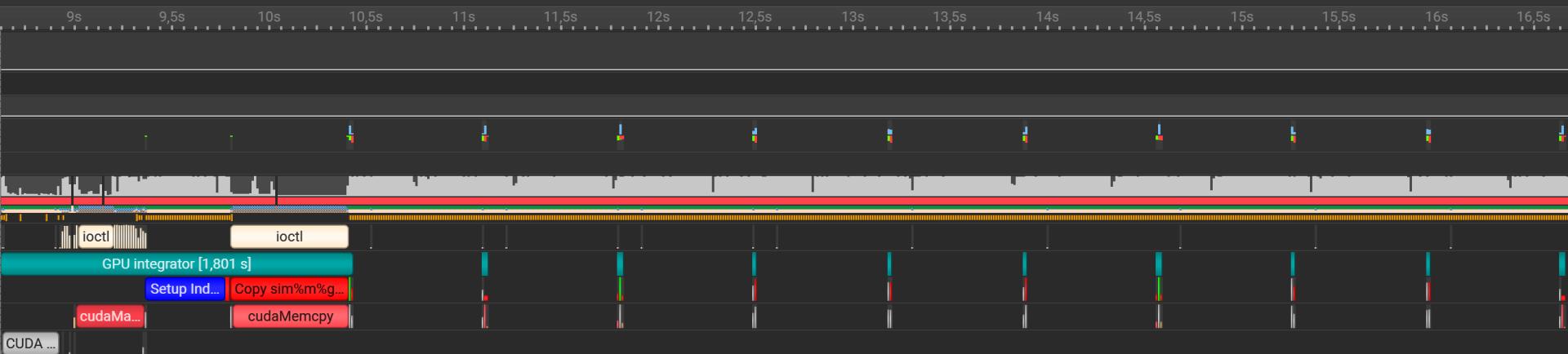
PRINCETON
UNIVERSITY

GPU Hydro Kernel: Initial State

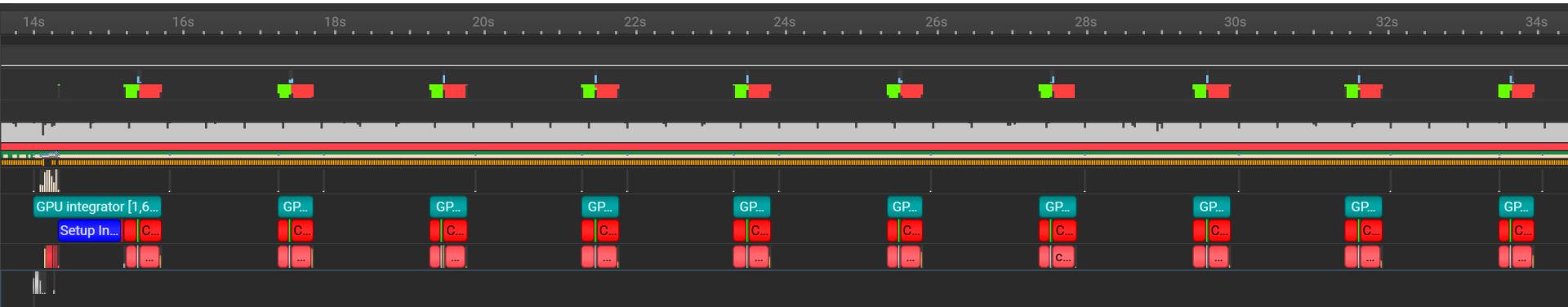
- Operation sequence:
 - Compute time step (`cmp_dt`)
 - Set new state vector to old one (`set_unew`)
 - Update `unew` from `uold` (`hydro_integrator`)
 - Set old state vector to new one (`set_uold`)
- Initial state of the project
 - `hydro_integrator` on the GPU (written by Bob Caddy)
 - Host: `cmp_dt`, `set_unew`, `memcpy(Host2Device)`
 - Device: `hydro_integrator`
 - Host: `memcpy(Device2Host)`, `set_uold`

Night Systems Profile: Initial State

Della Grace Hopper

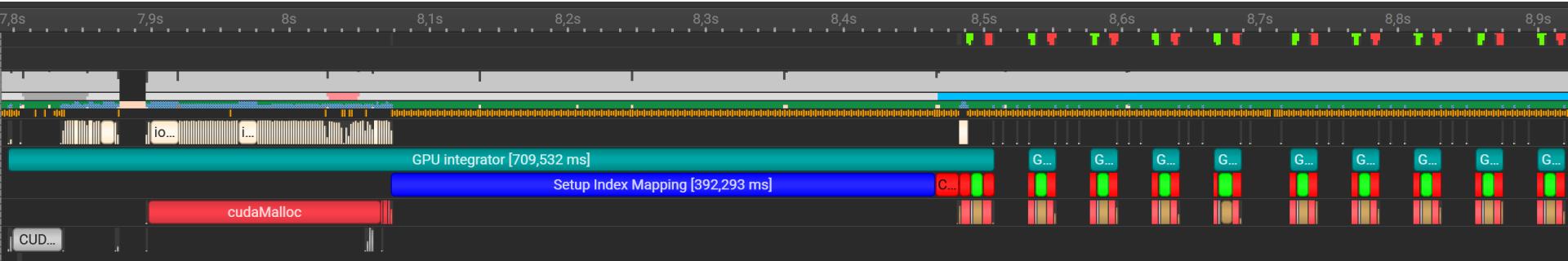


Della A100

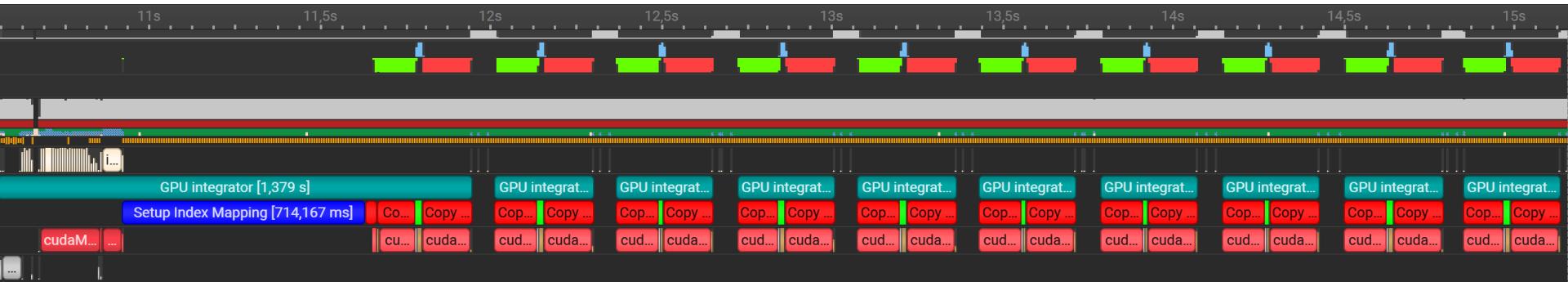


OpenMP parallelization of the CPU part

Della Grace Hopper



Della A100



Porting the entire code to GPU

- Current state of the project
 - `hydro_integrator` on the GPU (written by Bob Caddy)
 - `cmp_dt`, `set_unew`, `set_uold` on the GPU (this work)
 - One `memcpy`(Host2Device) during the first time step
- Simple kernel with one oct per device block (2x2x2 grid)
 - Time step requires a reduction
- We used single precision arithmetics and double the speed.

One time step with 1 CPU and gfortran on della-gpu: 7s

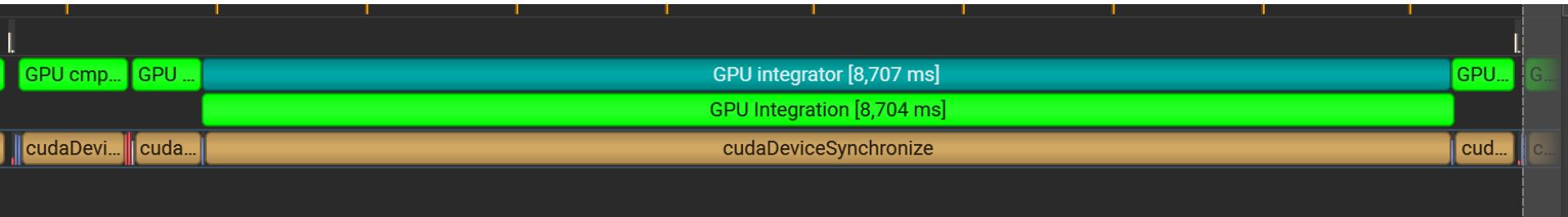
One time step with 1 GPU only hydro kernel: 2sec

One time step with 1 GPU all kernels: 40ms (200x speedup)

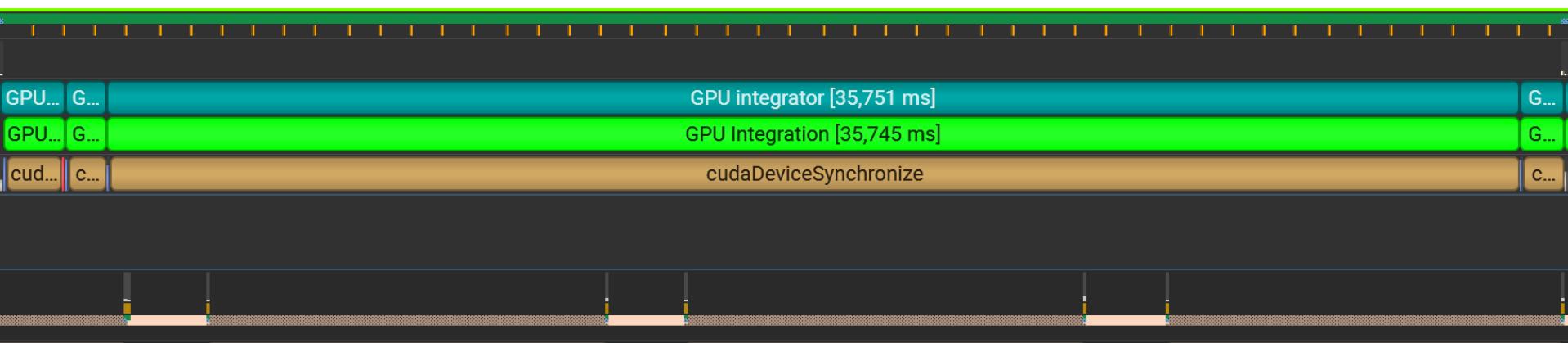
Current performance with single precision: 1.5 billion cell updates/GPU/s!

Porting the entire code to GPU

Della Grace Hopper



Della A100



Next Steps: Multiple GPUs

- Modify the original hydro kernel from 4x4x4 blocks to 3x3x3 blocks
 - First version working with `ncentral=1` and `ncentral=2` ✓
 - Need to merge with main branch ✗
- Use the original hydro kernel with multiple GPUs
 - All other routines on the CPU ✓
 - 2 memCopy for the full grid per time step ✓
 - MPI on the CPU ✓
 - Need to merge with main branch ✗
- Use the new kernel with multiple GPUs
 - All routines on the GPU ✓
 - 2 memCopy only for halo cells ✗
 - MPI on the CPU ✗

mini-ramses: data structure

- Optimized execution using **clean octs**, **dirty octs** and **ghost octs**
- Clean octs have **all neighbors** in local memory
- Dirty octs have **at least one neighbor** in:
 - Physical boundaries
 - Other MPI domain
 - Coarser refinement level
- Ghost octs are created temporarily in the cache memory from:
 - Physical BC rules (reflexive, zero gradient, imposed...)
 - MPI communications
 - Coarse-to-fine interpolation
- Use GPU or OpenMP **asynchronicity** to:
 - Build the cache while updating the clean octs
 - Then update the dirty octs

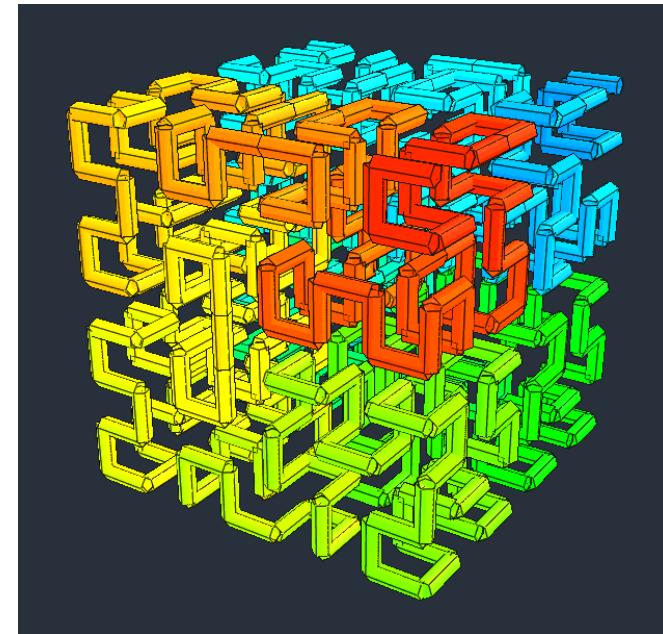
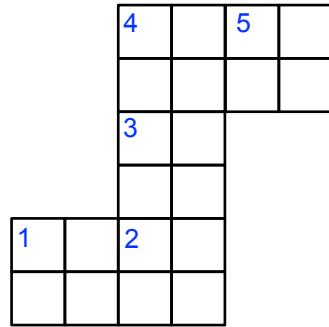
mini-ramses: hydro kernel

Memory space

1		2		3		4		5	

Octs are sorted in memory using the Hilbert space filling curve: [index = hilbert\(l,i,j,k\)](#)

Physical space



mini-ramses: hydro kernel

Memory space

1		2		3		4		5		

6		7		8		9		10		11

We create 6 new ghost octs in cache memory (here in green) and copy the central oct in the hydro kernel at position 1.

Physical space

			4			5				
7			6			3				
8			1			2				
9			10			11				

Kernel workspace and connectivity map

7			6			3				
8			1			2				
9			10			11				



mini-ramses: hydro kernel

Memory space

1	2	3	4	5		

6	7	8	9	10	11	12	13	14

We create 3 new ghost octs in cache memory (here in green) and copy the central oct in the hydro kernel at position 2.

Physical space

			4		5		
7		6	3		14		
8		1	2		13		

7		6	3		14		
8		1	2		13		
9		10	11		12		

Kernel workspace and connectivity map

7		6	3	
8		1	2	
9		10	11	

6		3	14	
1		2	13	
10		11	12	



mini-ramses: hydro kernel

Memory space

1	2	3	4	5	

6	7	8	9	10	11	12	13	14	15	

We create 1 new ghost oct in cache memory (here in green) and copy the central oct in the hydro kernel at position 3.

Physical space

	15	4	5	
7	6	3	14	
8	1	2	13	
9	10	11	12	

Kernel workspace and connectivity map

7	6	3	
8	1	2	
9	10	11	

6	3	14	
1	2	13	
10	11	12	

15	4	5	
6	3	14	
1	2	13	



mini-ramses: hydro kernel

Memory space

1	2	3	4	5	

6	7	8	9	10	11	12	13	14	15

We copy the main memory (black octs) and the cache memory (red octs) to the GPU memory, together with the connectivity map to update the 5 local octs hydro variables.

Physical space

	15		4		5	
7		6	3		14	
8		1	2		13	
9		10	11		12	

Kernel workspace and connectivity map

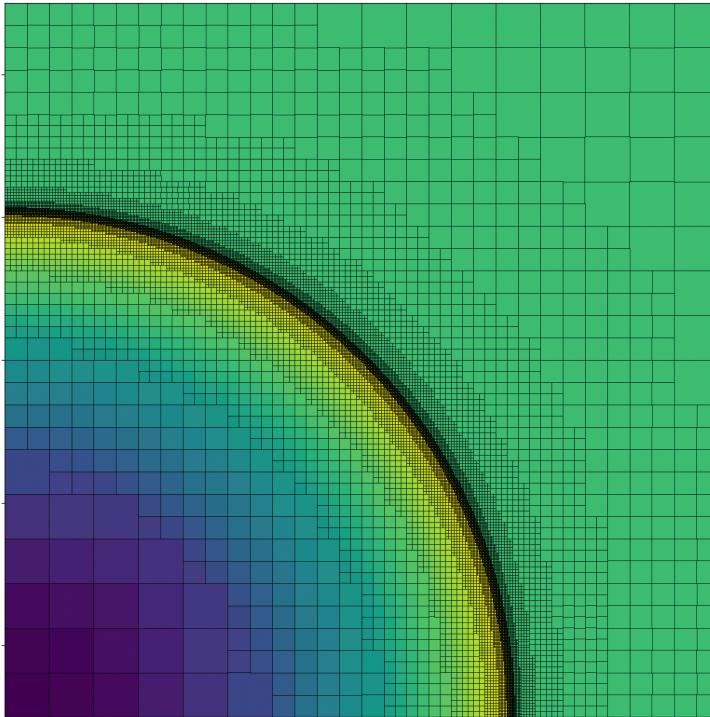
7		6		3	
8		1		2	
9		10		11	

6		3		14	
1		2		13	
10		11		12	

15		4		5	
6		3		14	
1		2		13	



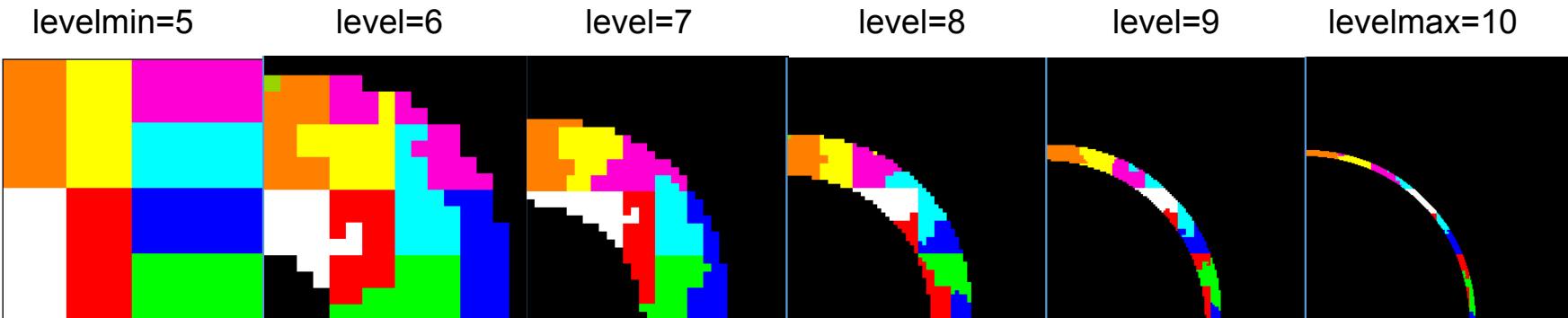
mini-ramses: level-by-level load balancing



Sedov blast wave test in 2D
with 6 levels of refinement
distributed over 8 processors.

Dynamical load balancing
every time step.

Each level is domain
decomposed independently.



mini-ramses: initial conditions

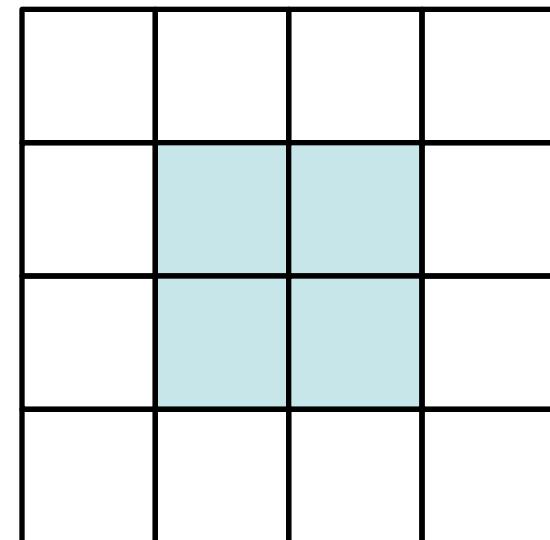
- Various predefined initial conditions
 - Use `condinit.f90` initial condition setup file.
 - Default using regions and namelist block `INIT_PARAMS`
 - Predefined ICs: `INIT=COEUR, INSTA, DOUBLEMACH, OT, PONO, ABC, LOOP...`
 - `PATCH=../patch/init/halo` and `PATCH=../patch/init/merger`
- MUSIC initial conditions using `filetype='grafic'` or `filetype='grafic_zoom'`
- GADGET initial conditions using `filetype='gadget'`
 - Generated using e.g. the DICE code
- RAMSES initial conditions using `filetype='ramses'`
 - Reads RAMSES output files (not restart files)
 - Still need to specify the initial magnetic field separately

mini-ramses: new features

- Various boundary conditions (default is periodic)
 - Arbitrary box geometry using oct Cartesian indexing at `levelmin`:
 - Reflexive, zero gradient, imposed...

Example from `sedov2d.nml`

```
levelmin=3          periodic=.false.,.false.  
box_size=1.0        nbound=4  
box_xmin=1          bound_dir=1,1,2,2  
box_ymin=1          bound_type=1,1,1,1  
box_xmax=-1         bound_shift=1,-1,1,-1  
box_ymax=-1         bound_xmin= 0,-1, 0, 0  
                      bound_xmax= 1, 0, 0, 0  
                      bound_ymin= 1, 1, 0,-1  
                      bound_ymax=-1,-1, 1, 0
```

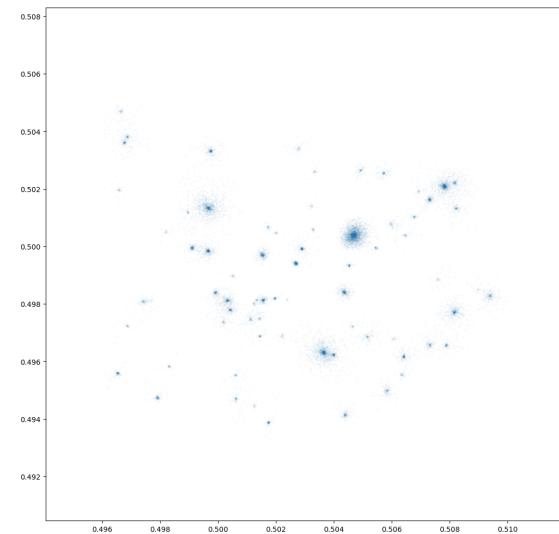
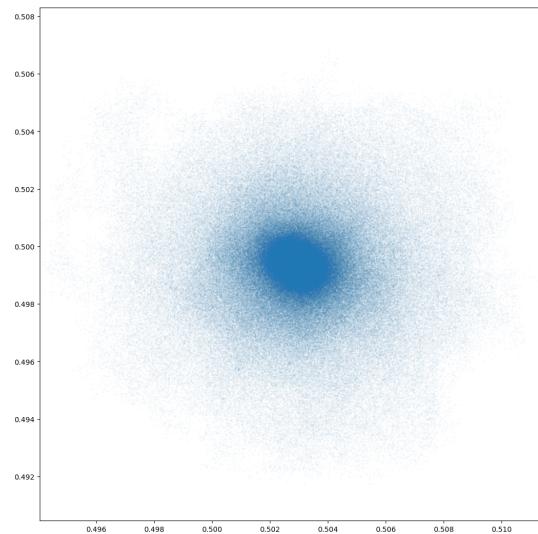
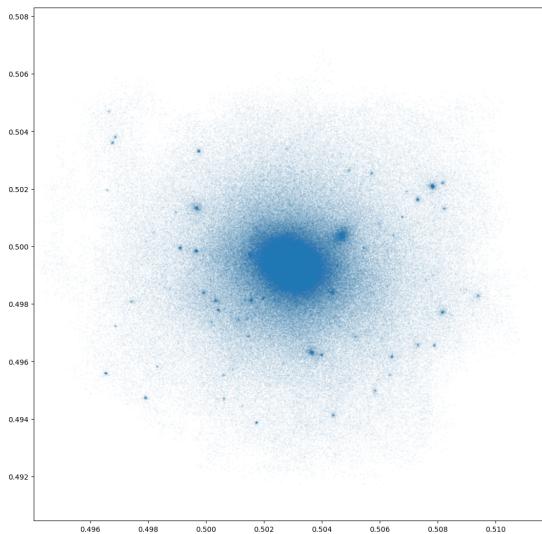


mini-ramses: new features

- Physics routines
 - Star formation recipe with 3 different models
 - Dual energy update using entropy
 - Thermal supernova feedback
 - Mechanical supernova feedback
 - Subgrid turbulence LES model
 - Equilibrium cooling, constant UV background with self-shielding
 - Non-equilibrium cooling with H and He chemistry
 - MHD solver now merged with hydro solver
 - `switch_llf_dmin=-1` and `switch_llf_pmin=-1`

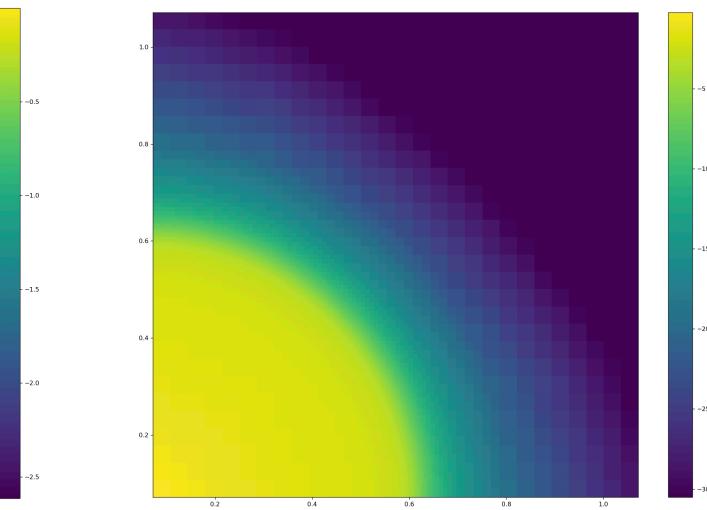
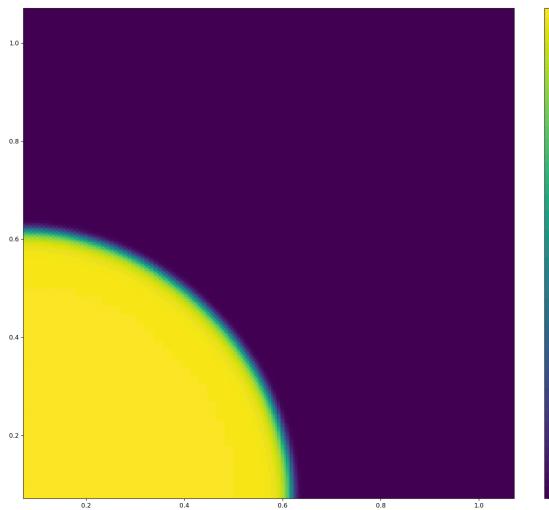
mini-ramses: new features

- Clump finder
 - PHEW implementation with software cache
 - Introducing central halos and satellite halos
 - Particle unbinding
 - Merger tree particles



mini-ramses: new features

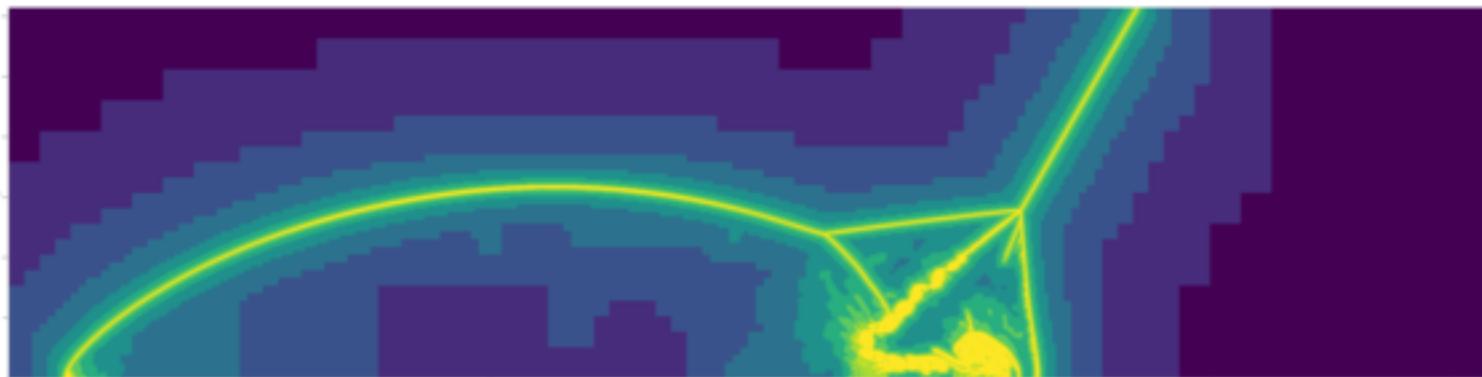
- Sink formation and evolution
 - Seeding using clump finder
 - Reset position to clump center and/or artificial friction
 - Accretion and feedback (see Nick's talk)
- Radiative transfer with M1 solver (see Joki's talk)



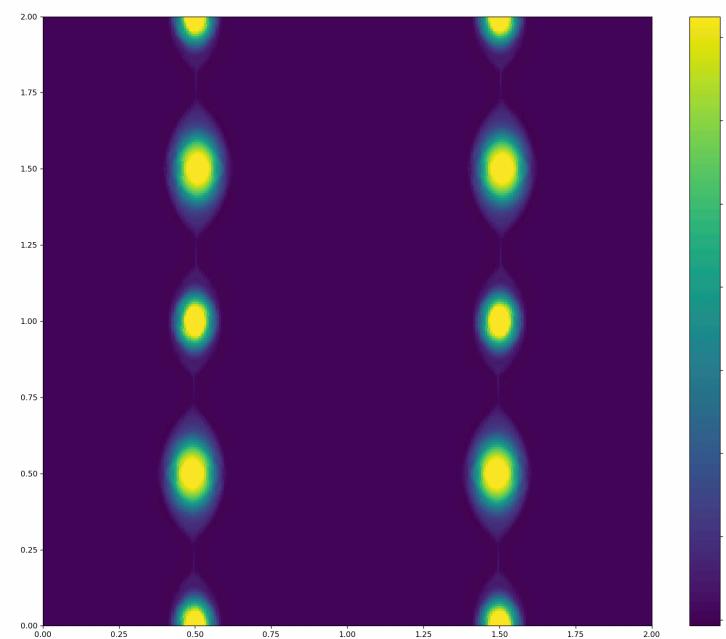
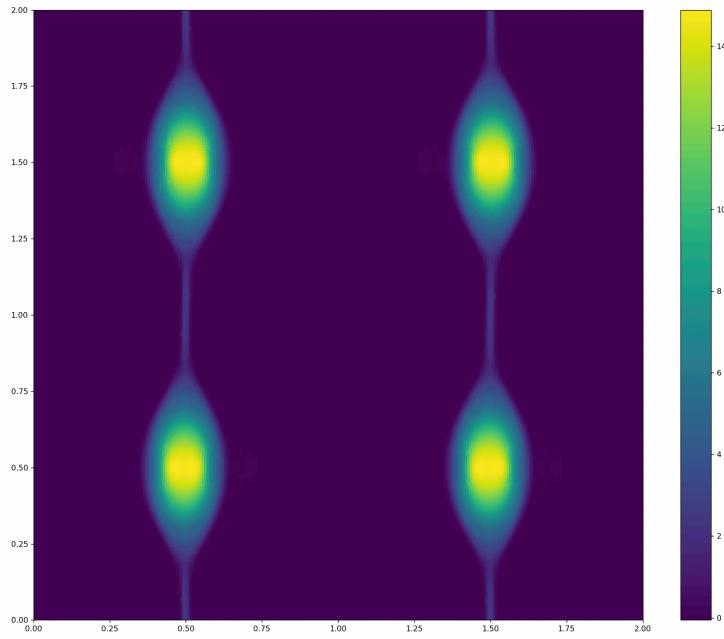
mini-ramses: initial conditions

- 1D tests:
 - HYDRO: advect1d.nml, blast1d.nml, sedov1d.nml, tube1d.nml
 - MHD: brio_wu.nml, dai_woodward.nml
- 2D tests:
 - HYDRO: sedov2d.nml, [double_mach.nml](#), rt.nml
 - MHD: loop.nml, ot.nml, [current_sheet.nml](#)
 - RT: [stromgren2d.nml](#), beam2d.nml, shadow2d.nml
- 3D tests:
 - N-BODY: dmo.nml, dmo_zoom.nml
 - HYDRO: sedov3d.nml, coeur.nml, cosmo.nml, [cosmo_zoom.nml](#), halo.nml, merger.nml, dice.nml
 - MHD: [pono.nml](#), [abc.nml](#), [halo_mhd.nml](#)
 - SINK: [bondi.nml](#), [agn_jet.nml](#)

Double Mach reflection



Current sheet



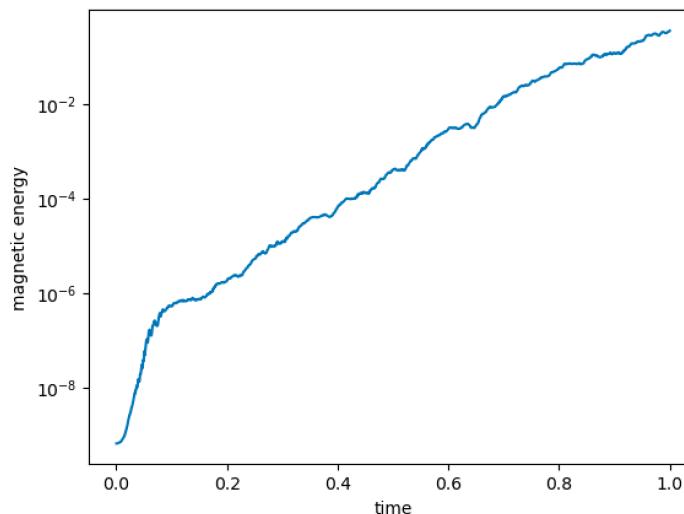
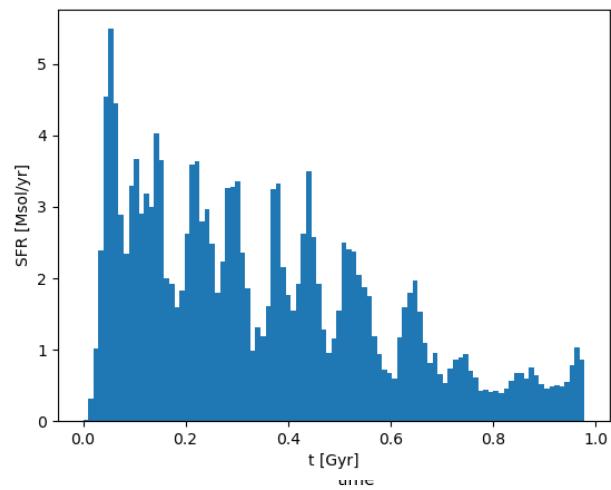
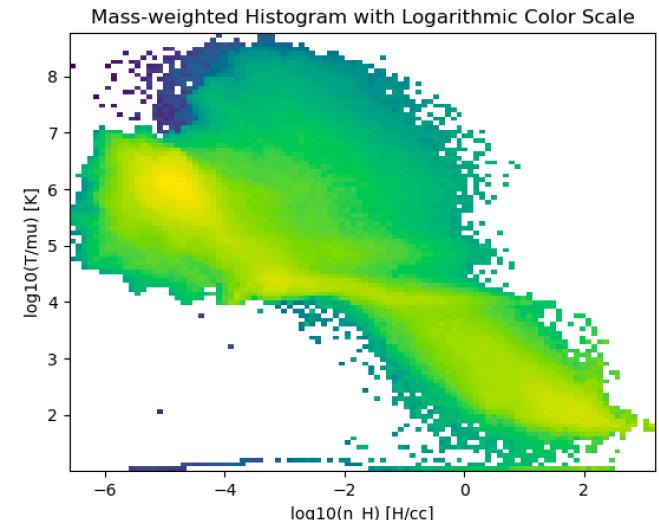
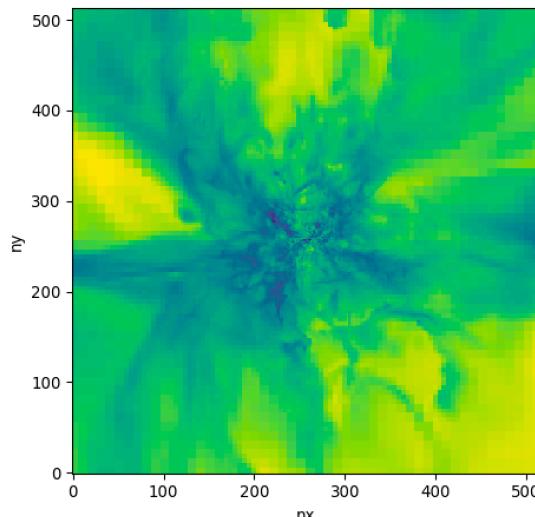
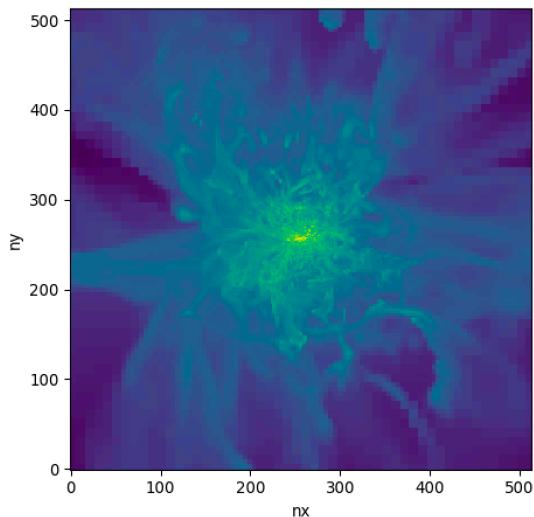
mini-ramses: new features

- New output strategy
 - `access='stream'` python read using `numpy.fromfile`
 - Backup frequency `bkp_timrs_hrs=2` and `bkp_modulo=3`
 - Restart folder: double precision, face-centered magnetic field (6 vars)
 - You can change the number of cpu at restart
 - Output folder: single precision, cell-centered magnetic field (3 vars)
 - You can specify the number of files per output folder
 - Default `nfile=1` with $1 \leq nfile \leq ncpu$
- Maximum level 64 using `HILBERT=3` (no time penalty).

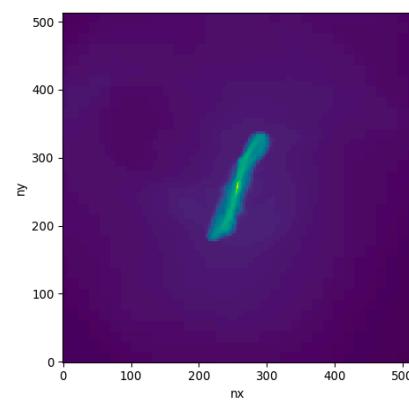
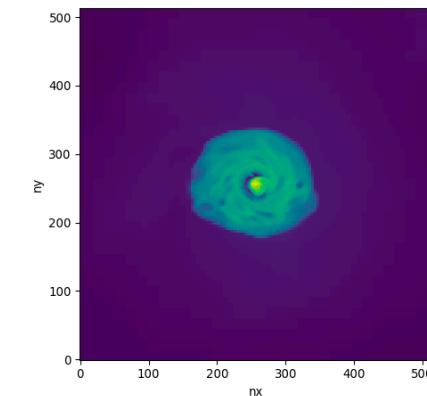
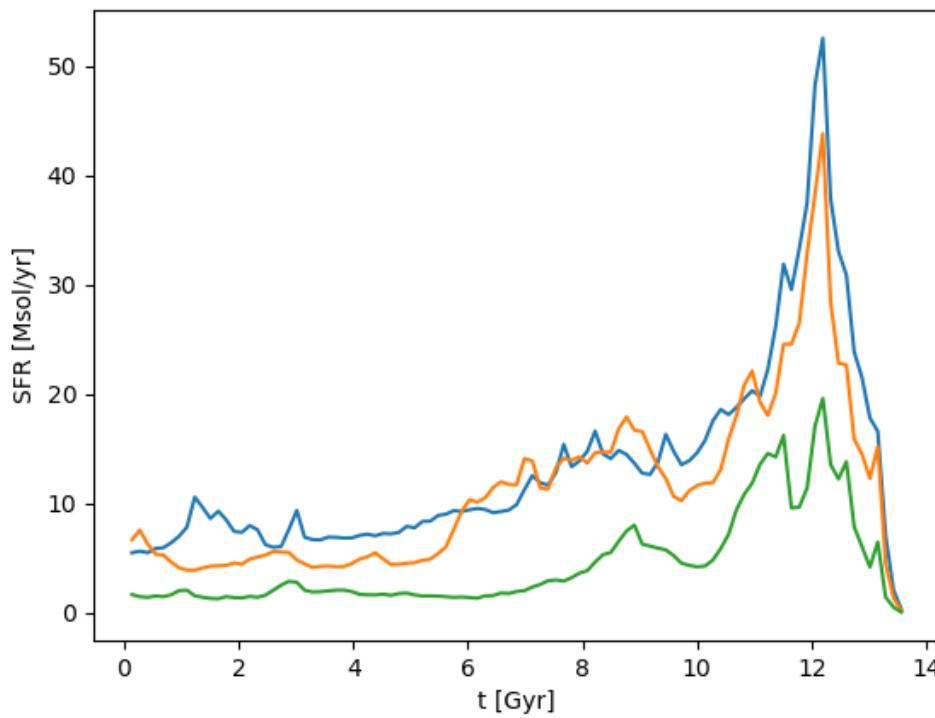
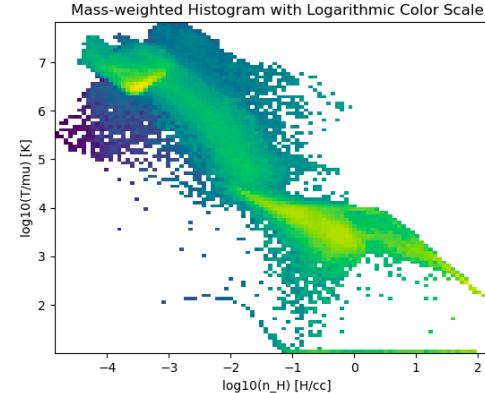
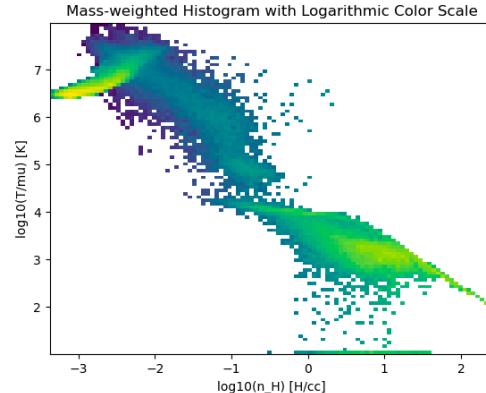
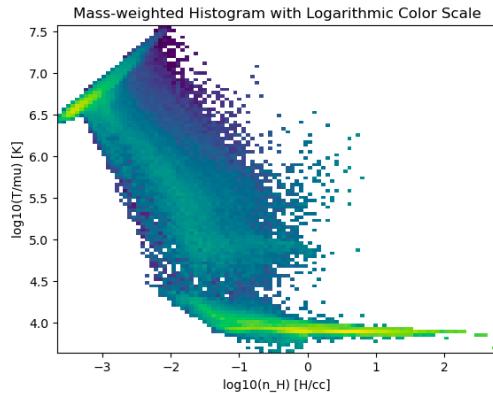
mini-ramses: new features

- Makefile:
 - `make NDIM=3 MPI=1 DEBUG=1 UNITS=MERGER GRAV=1 PATCH=../patch/init/merger NPSCAL=2 MHD=1`
- cmake also available:
 - `cmake .. -DNDIM=3 -DINIT=PONO -DMHD=1`
- `NENER=0` implemented but not tested
- 5 hydro variables for `NDIM=1, 2 and 3` and also for MHD
- `NPSCAL=0` for additional passive scalars
- Python package `miniramses.py` (reading and plotting)
- Python utils `amr2img.py`, `map2img.py`, `run2img.py`, `debug.py`, `history.py`, `log.py`, `sfr.py`...
- F90 utils `amr2map.f90` and `part2map.f90`

mini-ramses: galactic dynamo in a cooling halo



mini-ramses: cosmological galaxy formation



mini-ramses: software engineering

- <https://bitbucket.org/rteyssie/mini-ramses>
- GPU hydro kernel prototype developed in Nvidia Fortran by Bob Caddy (Princeton Research Software Engineer) in folder [mini-ramses/gpu](#)
- <https://robertcaddy.com/posts/GPU-Mini-Ramses-Port/>
- Princeton Open Hackathon with Nvidia in June 2025 to complete the hydro kernel (MPI and AMR) and explore options for the Poisson kernel.
- GPU kernel for RT should follow quickly (non-equilibrium chemistry)
- OpenMP/MPI implementation in progress
- Folder [mini-ramses/doc](#)