Platform for Advanced Scientific Computing

Accelerated Radiative Transfer Simulations for the Square Kilometre Array Observatory

PASC Program 2025–2028

ETH zürich
Michele Bianco
Tommaso Boschi
Alexandre Refregier

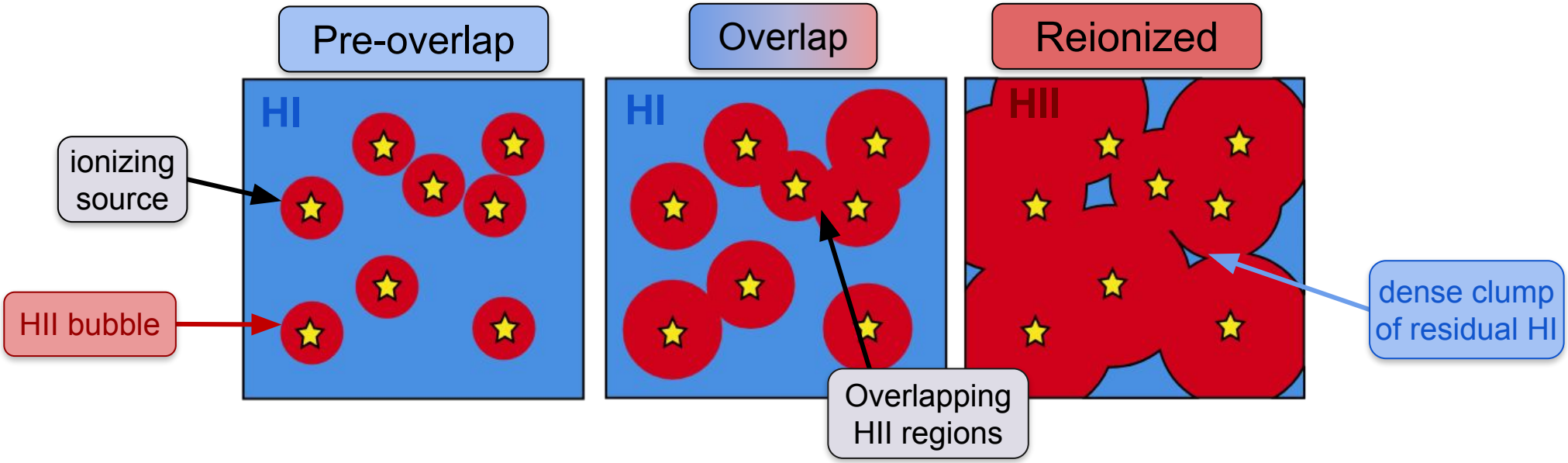
Universität Basel
Florina Ciorba
Osman Seckin Simsek

zhaw
Elena Gavagnin
Philipp Denzel

CSCS
Christopher Bignamini
Sebastian Keller

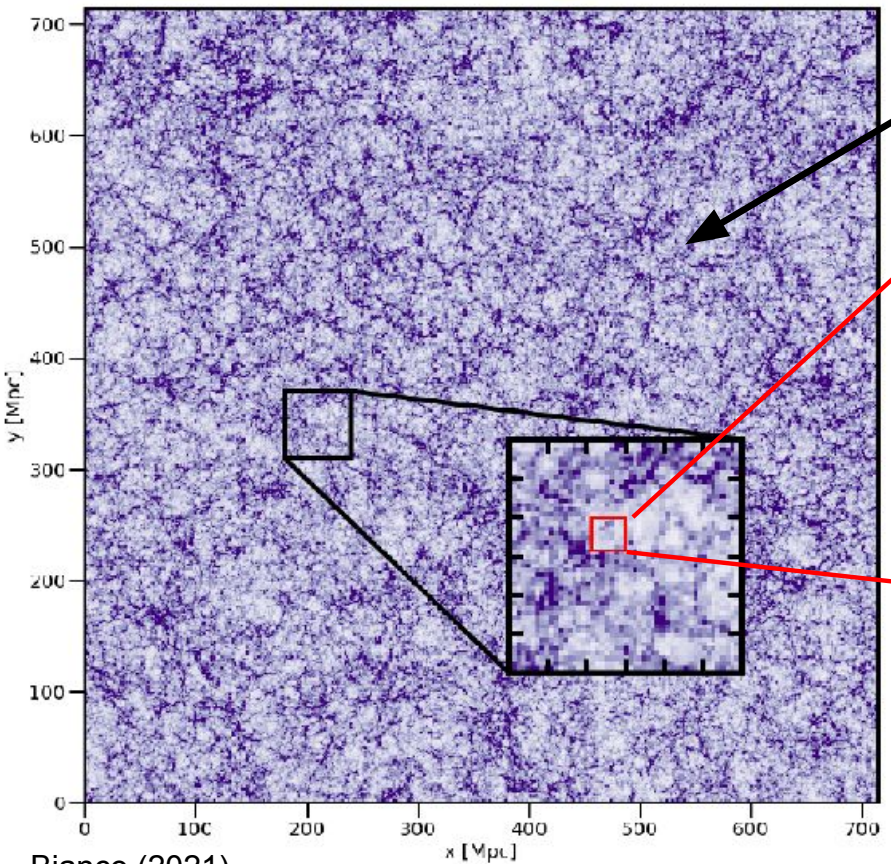
EPFL
Yves Revaz

The Epoch of Reionization (EoR)



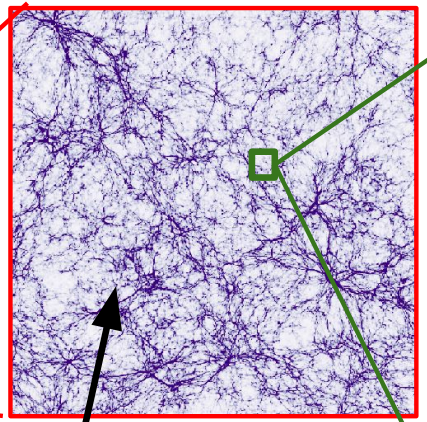
- Appearance of the **first luminous sources** (UV and X-Ray radiation).
- Sources start to **ionize and heat** the Intergalactic Medium (IGM):
  - Formation of ionized regions (HII bubble) around collapse structures.
- Universe transition from a **neutral/cold** state to a **hot/ionized**:
  - Pre-overlap (neutral)→Overlap (partially ionized)→Reionized

An inherently multi-scale problem...



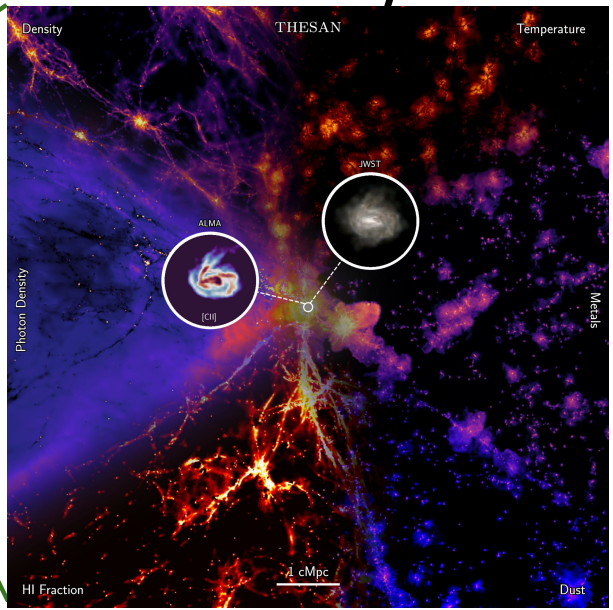
Bianco (2021)

Large-scale simulation
~700 Mpc



Intermediate scale
~ 100 Mpc

Galactic scale
< 1 Mpc

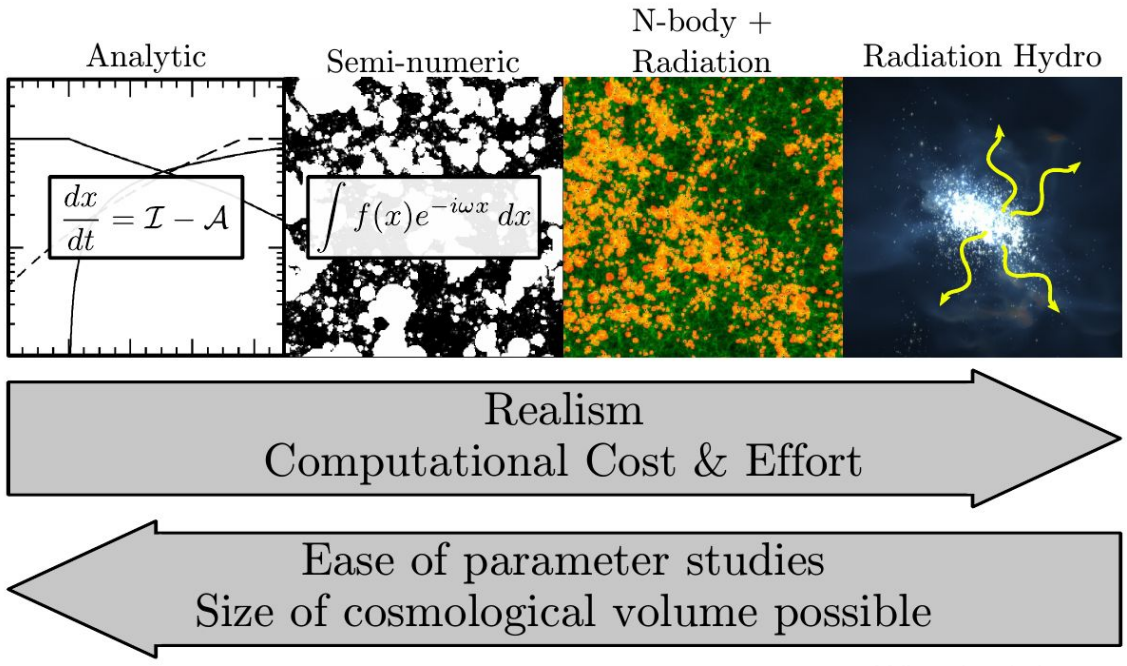


Thesan Collaboration

Cosmological EoR simulations

General **Radiative Transfer** algorithm:

1. Photons emitted from ionizing sources are **propagated in IGM**
2. **Ionization & heating rates** are calculated for HI, HeI and HeII
3. Multi-frequency **chemistry ODE** is solved to determine ionized matter density
4. Simulate a wide range of the cosmic time, but **radiative & hydrodynamic feedback** act on small time step for **convergence**



Wise (2019)

Typical EoR simulation takes up to **millions of core–hours** on several thousand nodes, run in parallel on high performance computers.

Simulating reionization with Radiative Transfer

Solve the differential equation of the **evolution of neutral hydrogen** (HI) during EoR

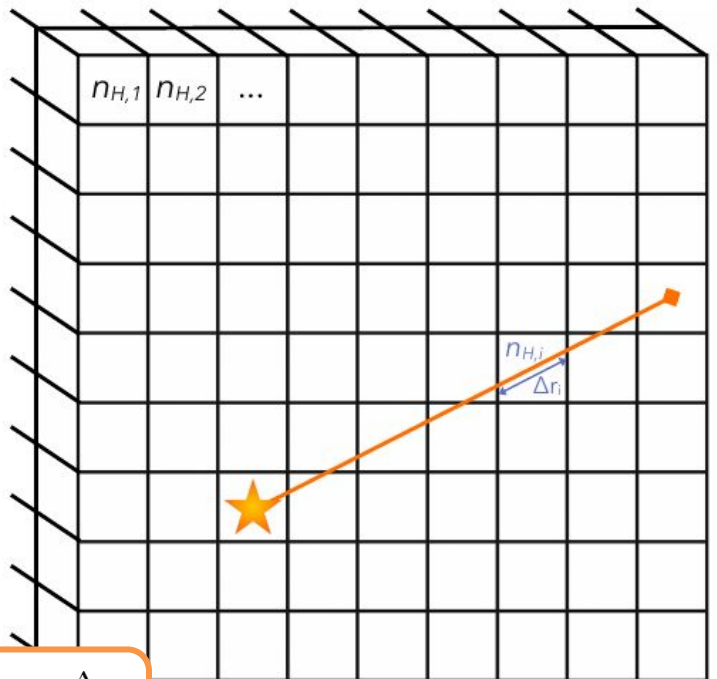
$$\frac{dx_{\mathrm{HI}}}{dt} = \underbrace{-x_{\mathrm{HI}}\Gamma_{\mathrm{ion}}}_{\text{Ionization Rate}} + \underbrace{(1-x_{\mathrm{HI}})n_{\mathrm{e}}\alpha_{\mathrm{H}}}_{\text{Recombination Rate}}$$

Short-characteristic ray-tracing algorithm: propagate photons and determine **column density** along the radiation line of sight.

$$\Gamma_{\mathrm{ion}}(r) = \frac{1}{4\pi r^2}\int_{\nu_{\mathrm{th}}}^{\infty}\frac{L_\nu}{h\nu}\sigma(\nu)e^{-\tau_\nu(r)}d\nu$$

$$\tau_\nu = \sigma_\nu N_{HI}$$

$$N_{HI} = \sum_{i\,\in\,\mathrm{L.o.S}} n_{\mathrm{HI},i}\Delta r_i$$

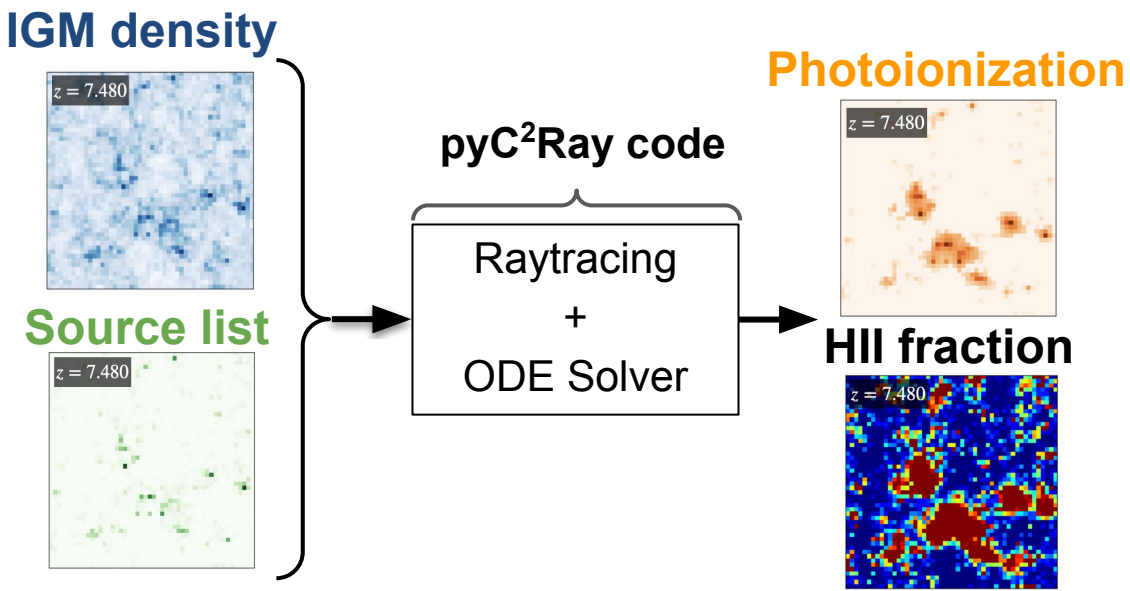


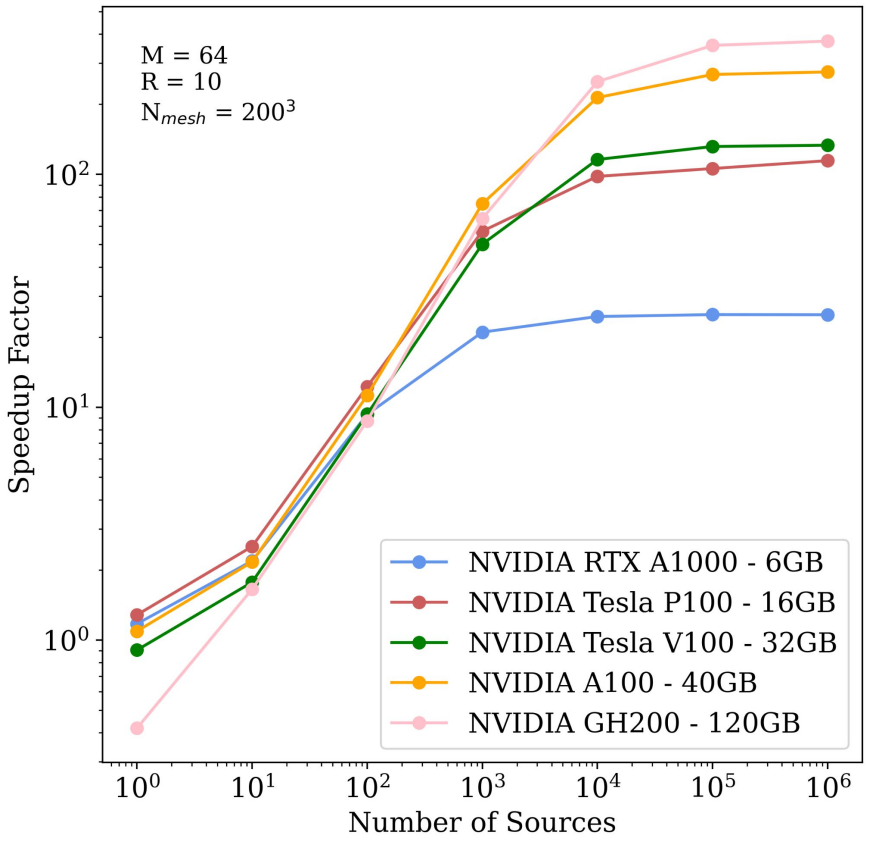
Similar equations can be defined for helium **HeI** and **HeII**

EoR simulations with pyC²Ray

pyC²Ray [2311.01492] is an update of the fully-numerical, Fortran90 *C²-Ray* code [Mellema+06]

- ➔ User-friendly Python interface
- ➔ **ASORA**: new ray-tracing algorithm in CUDA
- ➔ **MPI** parallelisation enabled
- ➔ ~400 speed-up factor on Alps (GH200) vs CPU





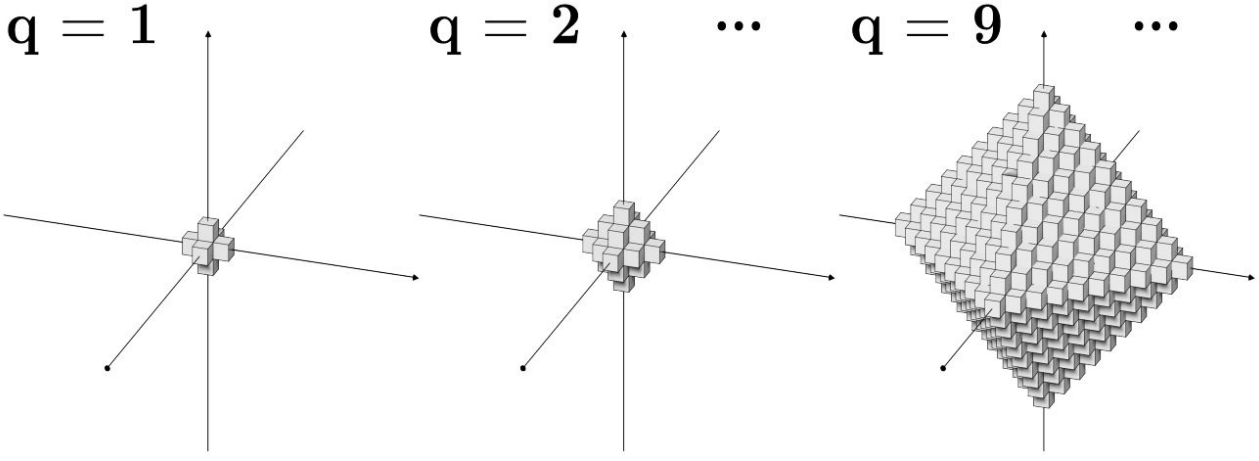
ASORA parallelization details

Accelerated Short-Characteristics Octahedral RAy-tracing

Levels of parallelization across a GPU cluster (e.g. **Daint** on Alps):

1. **Thread-block:** photons propagating from 1 ionizing source, expand in 6-cell octahedral neighborhood, **pixels at each q-shell are independent**

   a. (q, s) → (i, j, k) map
      s ≡ thread index, **q-shell** depends only on pixels from **previous** shell
   b. High occupancy of threads when q ≳ 6

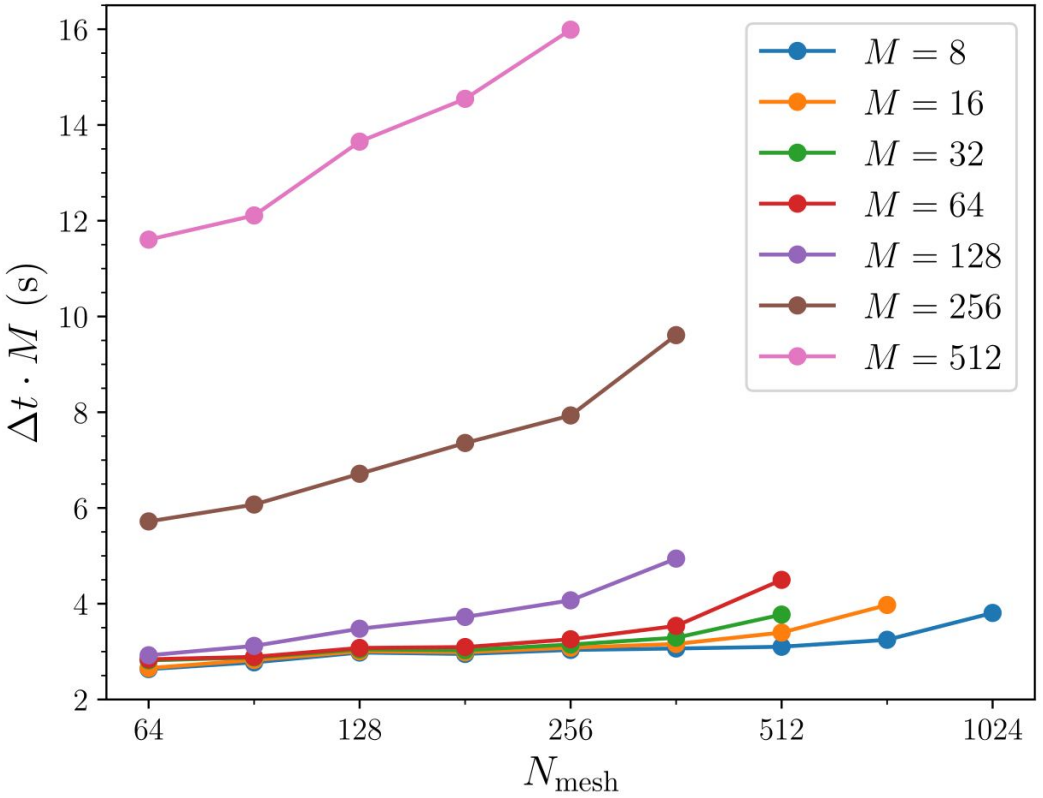


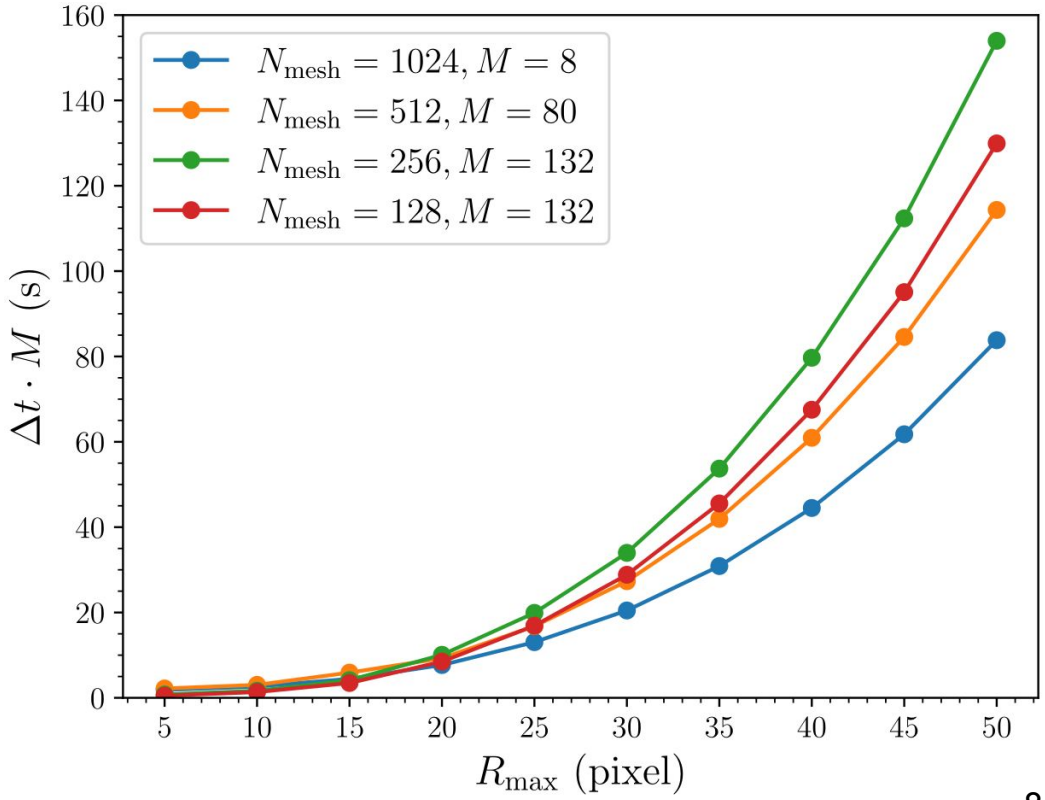
2. **GPU unit:** multiple sources in **batch**, as many as can fit in **global memory**
3. **HPC nodes:** batches are distributed on the cluster using **MPI**

ASORA profiling on Alps with GH200

Performance depends

- directly on: R_{max} maximum propagation distance for γs, K thread-block size
- indirectly on: N_{mesh} grid size of the Universe, M batch size, N_s number of sources





ASORA improvements – memory footprint

Before: column density used 3D mesh grid for storage and calculations

$$\text{Pixels} = N_{\text{mesh}}^3$$

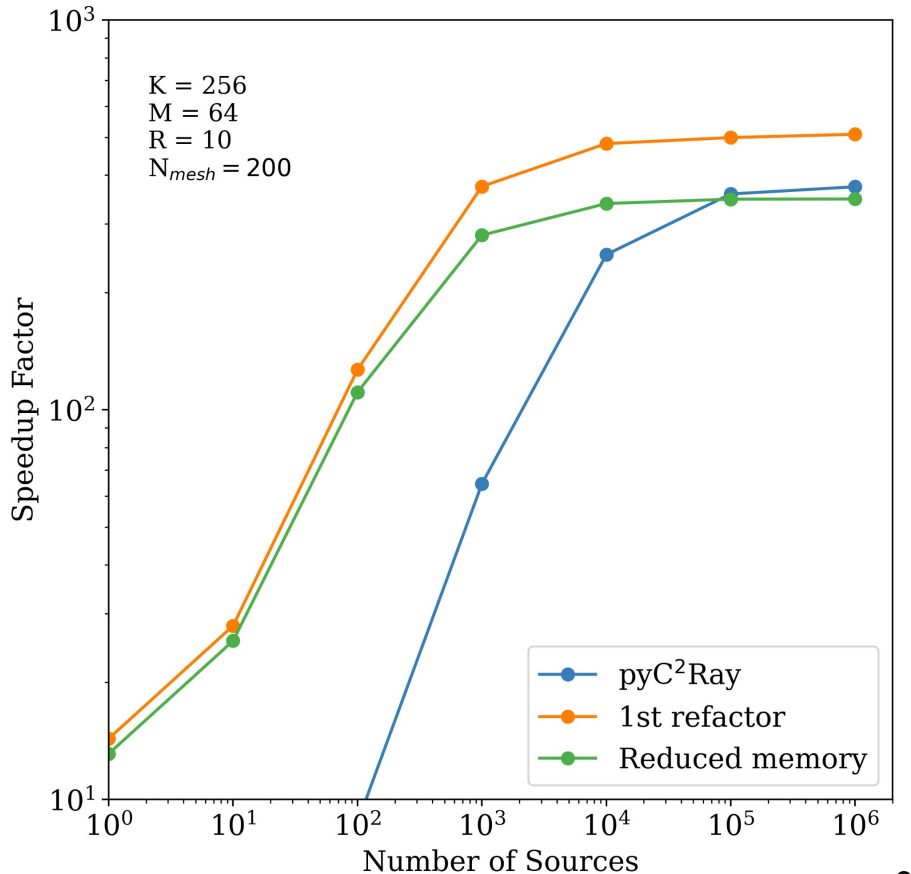
After: memory array that contains exactly the octahedron at a R_{max} distance

$$q_{\text{max}} = \lceil \sqrt{3} R_{\text{max}} \rceil$$

$$\text{Pixels} = 1 + \sum_{1}^{q_{\text{max}}} (4q^2 + 2)$$

Example: with N_{mesh} = 200, R_{max} = 10 column density memory storage reduced to 0.1%

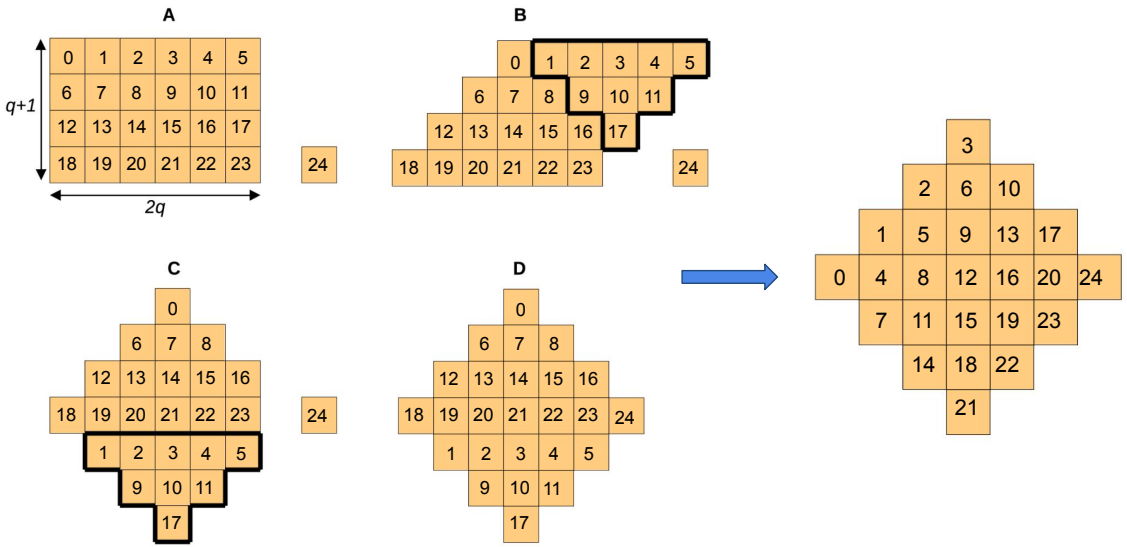
This requires **inverse mapping** (q, s) ↔ (i, j, k) to calculate short-characteristic weights.

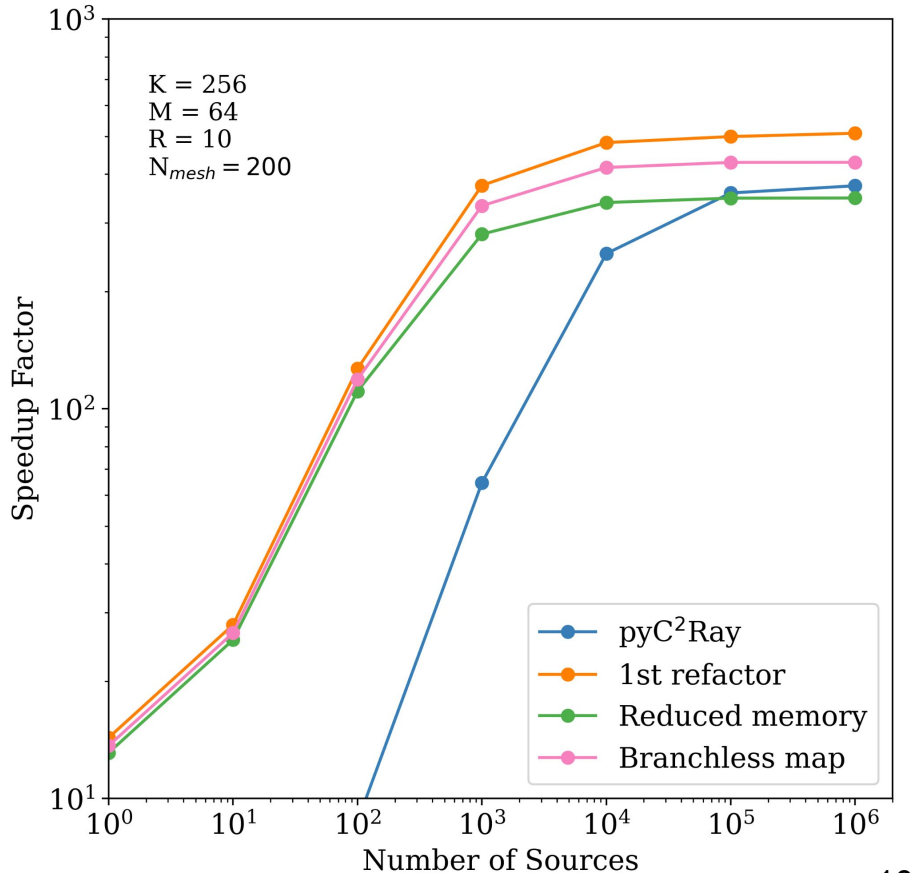


ASORA improvements – branchless mapping

Before: forward and backward mappings have at least 2 branches and 1 integer division

After: branchless forward mapping with 3 integer divisions and linear backward mapping



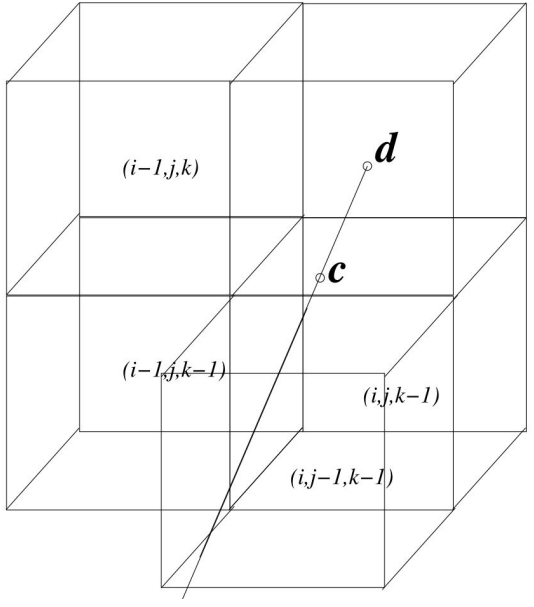


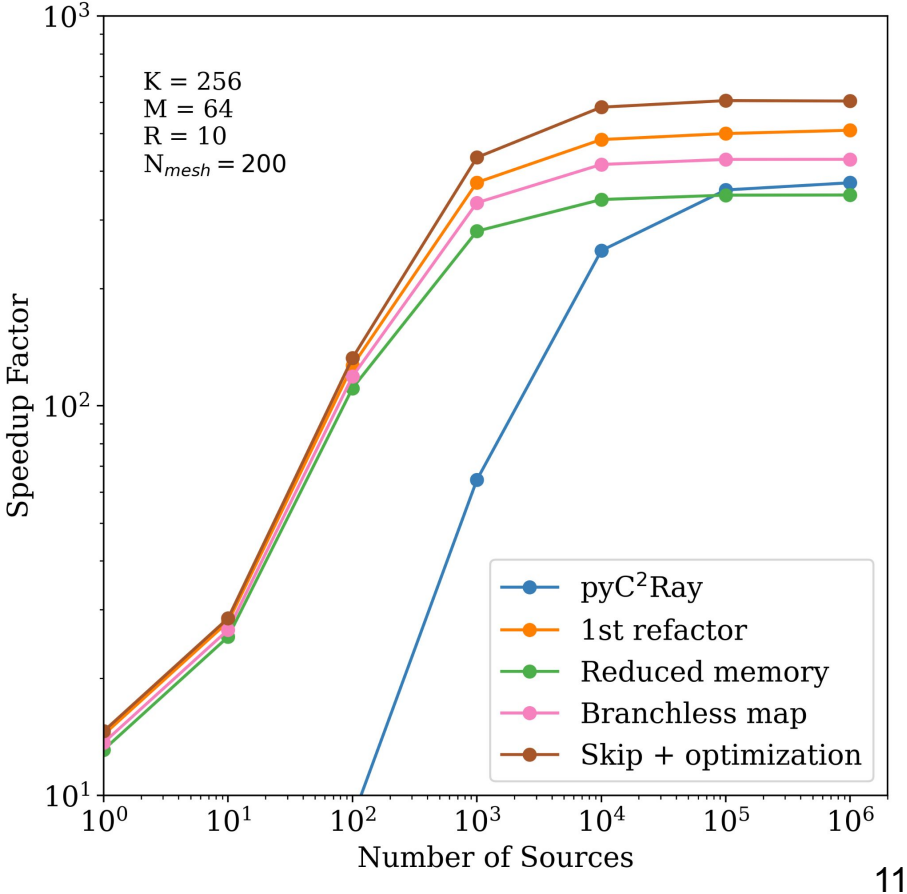
ASORA improvements – skip null pixels

With short-characteristic, a pixel gets **contributions from neighboring 4 pixels** in the direction of the source, for example:
(i, j, k-1), (i, j-1, k-1), (i-1, j, k-1), (i-1, j-1,k-1)

Some pixels have null contribution → **skip expensive memory reads**

These changes led to more optimization possibilities...





pyC^{2}Ray: today and tomorrow

- Hydrogen implementation of ASORA achieves a **~600 speed up factor** (+50% than first iteration) and internal **memory usage down to 0.1%**
- Important code refactors and optimizations for future extensions (see below)

1. **Multi-frequency extension for helium**
   a. Helium can be ionized twice: HeII and HeIII species
   b. Recombination is temperature dependent → compute photo-heating rate
   c. Absorption cross-section is frequency dependent
2. **Particle to mesh framework**
   a. Smoothing kernel to integrate particle-based hydrodynamics simulation
   b. Algorithm to convert to homogeneous or inhomogeneous/adaptive grids
3. **ODE porting to CUDA**
   a. Pixels are treated independently → high throughput application
   b. Reduce memory transfers from/to host when used with ASORA

Platform for Advanced Scientific Computing

Thank you for your attention