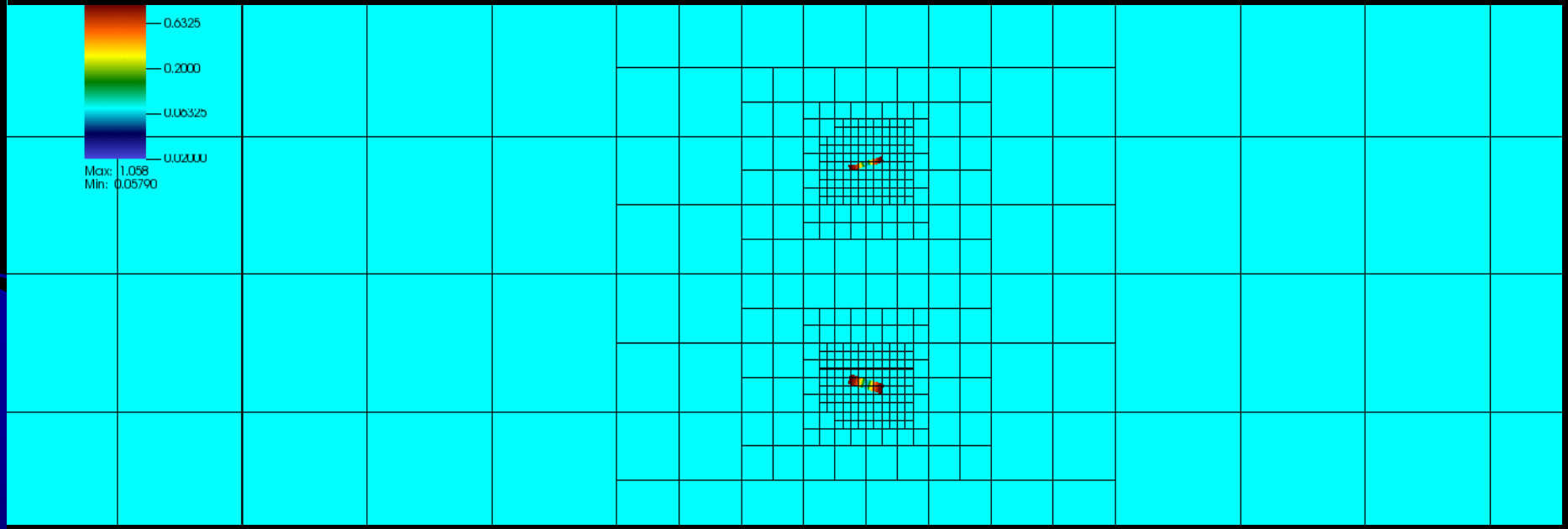
Computational Astrophysics & GPU

at Valentine's day ...

Hsi-Yu Schive (薛熙于)

Tzihong Chiueh (闕志鴻)

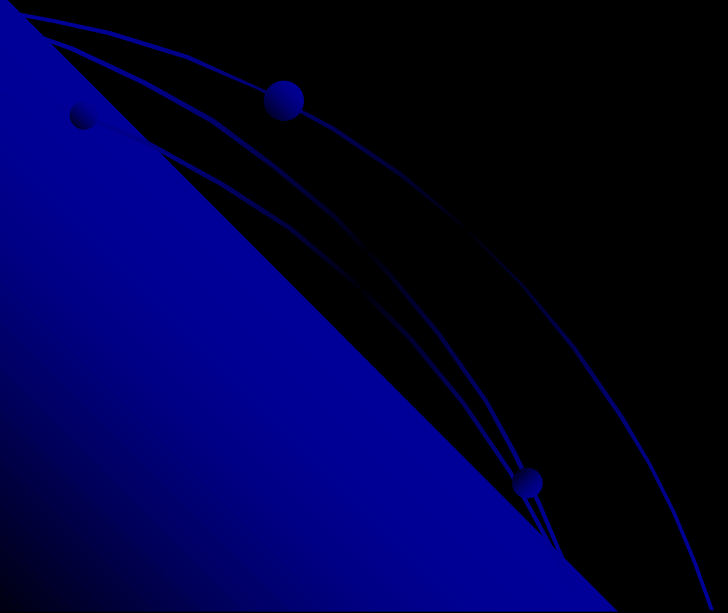


ASIAA Winter School (Feb. 14, 2014)

Outline

- **Particle Simulations**
  - **Gravitational force calculation**
- **Numerical Hydrodynamics**
  - **Finite-volume shock-capturing schemes**
  - *Adaptive Mesh Refinement (AMR)*
  - *Graphic Processing Unit (GPU) acceleration*
  - **GAMER : AMR + GPU framework**
    - ***GPU-accelerated Adaptive-MEsh-Refinement Code for Astrophysics***
- **Wave Dark Matter (ψDM)**

Particle Simulations



Basic Concepts

- ***'Particles'*** **can represent different objects for simulations at different scales. E.g.,**
  - **Satellites orbiting the Earth**
  - **Planets orbiting the Sun**
  - **Stars in a galaxy**
  - **Galaxies in a galaxy cluster**
  - **Cold dark matter**
- **Governing eqs. for the ith particle:**
  - $\dot{\boldsymbol{r}}_i = \boldsymbol{v}_i \rightarrow \boldsymbol{r}_i(\boldsymbol{t} + \Delta\boldsymbol{t}) \approx \boldsymbol{r}_i(\boldsymbol{t}) + \boldsymbol{v}_i\Delta\boldsymbol{t}$
    $\dot{\boldsymbol{v}}_i = \boldsymbol{a}_i \rightarrow \boldsymbol{v}_i(\boldsymbol{t} + \Delta\boldsymbol{t}) \approx \boldsymbol{v}_i(\boldsymbol{t}) + \boldsymbol{a}_i\Delta\boldsymbol{t}$
  - **Higher-order time integration can be adopted**
- **Key ingredient: gravitational acceleration estimation (a_i)**

Gravitational Acceleration Estimation

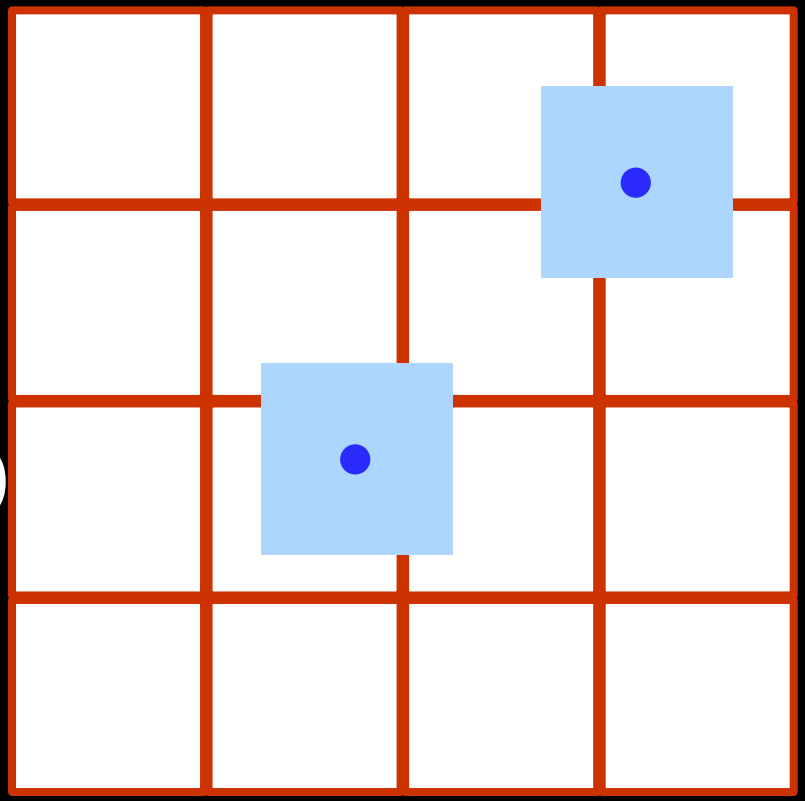
- Most straightforward method: direct N-body
  - Calculate all pairwise interactions one-by-one

$$a_i = \sum_{j=1, j\neq i}^{N} \frac{m_j(r_j - r_i)}{|r_j - r_i|^3}$$

  - Useful when high accuracy and close encounter are required (e.g., globular clusters where N ~ 10^6)
  - Extremely time-consuming and become impractical for large N due to its N^2 scaling
    - E.g., N ~ 10^{11} in our Milky Way → ~10^{22} pairs → takes ~10^4 years using a single CPU (or a few days using the top-1 supercomputer Tianhe-2)
- Approximate methods are required
  - E.g., particle-mesh, tree, fast multipole, hybrid schemes

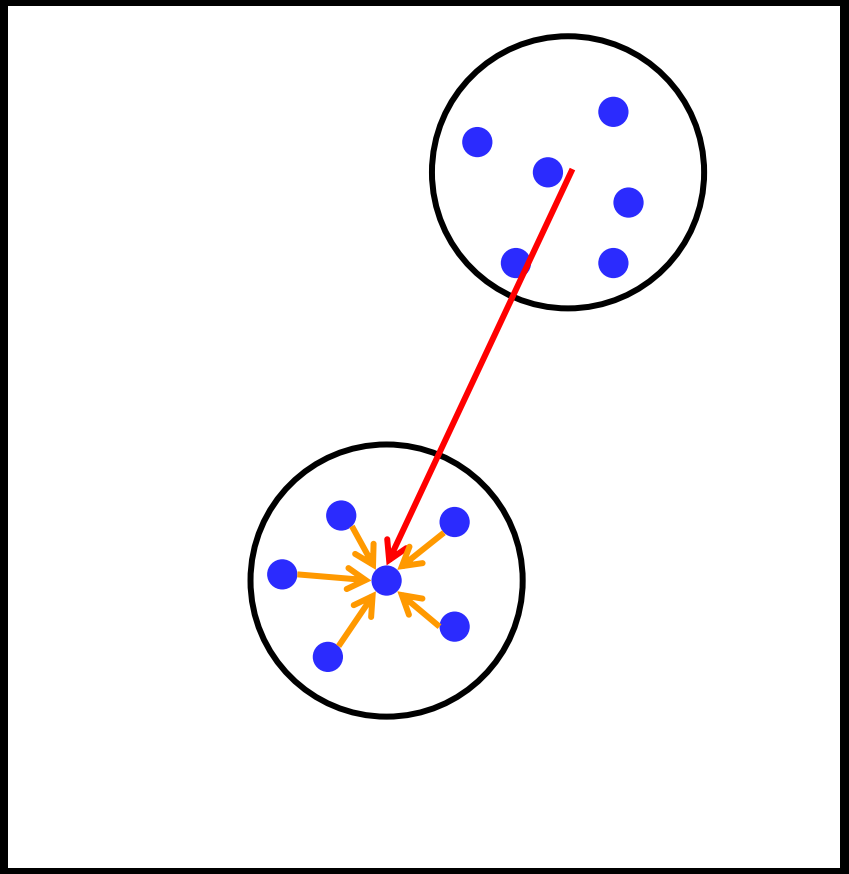
Example I : Particle-Mesh

- Assume a density distribution for each particle
- Assign particle mass to the underlying grids
- Use a grid-based gravity solver (e.g., Fast Fourier Transform → N log(N) scaling)
- Interpolate force back to the position of each particle
- WARNING: resolution is limited by the grid size
- Extension → P^3M method



Example II: Tree Method

- Nearby particles can be grouped into clusters
- For distant particles, only **cluster-cluster interaction** (center-of-mass plus higher moments if necessary) is computed
- Particles in the same cluster can be furthered grouped into smaller and smaller subclusters → form a **tree structure**
- **N log(N) scaling**
- **Accuracy can be controlled by the grouping criteria**
- Extension → **Tree-PM**



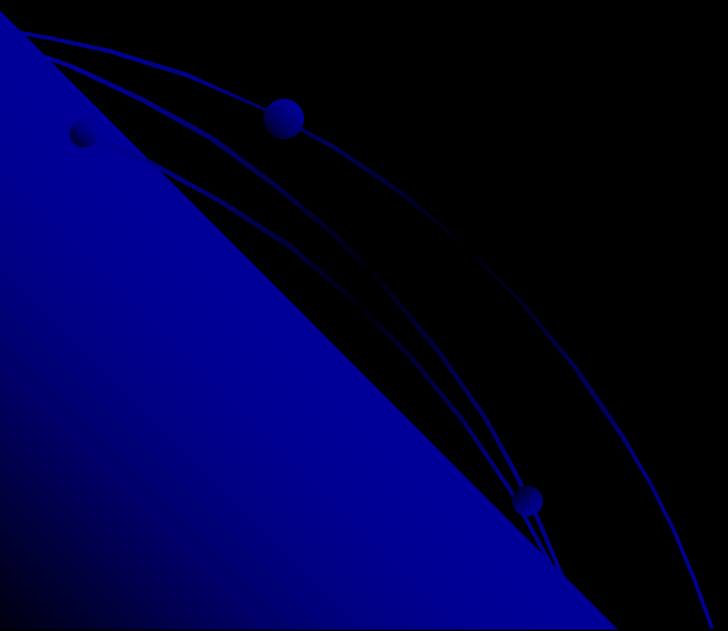
Cosmological Particle Simulations

- **Particles** are not really 'particles' in the CDM large-scale structure simulations
  - E.g., 100 Mpc/h box with 1024^3 particles → particle mass ~ 10^8 $M_\odot$ → **Each particle actually represent a massive dark matter 'cloud'**
- Soften length ε:

$$a_i = \sum_{j=1, j\neq i}^{N} \frac{m_j(r_j - r_i)}{[|r_j - r_i|^2 + \varepsilon^2]^{3/2}} \quad (a_{ij} \to 0 \; as \; r_i \sim r_j)$$

  - Avoid large-angle scattering
- Comoving coords. → expand together with the Universe
- Periodic boundary condition

Numerical Hydrodynamics



Solve PDEs Numerically

- **Example: advection equation**

$$\frac{\partial \rho(x,t)}{\partial t} + v(x)\frac{\partial \rho(x,t)}{\partial x} = 0$$

ρ : density
v : velocity

finite difference (forward time centered space)

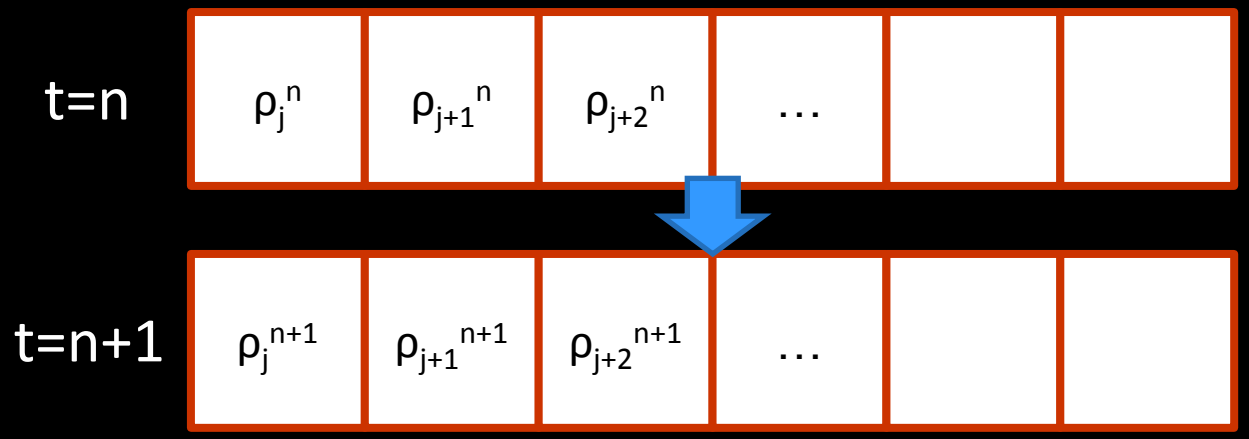
$$\frac{\rho(x,t+\Delta t)-\rho(x,t)}{\Delta t} + v(x)\frac{\rho(x+\Delta x,t)-\rho(x-\Delta x,t)}{2\Delta x} = 0$$

ρ(x,t) → ρ(x,t+Δt)

$$\rho(x,t+\Delta t) = \rho(x,t) - \Delta t v(x)\frac{\rho(x+\Delta x,t)-\rho(x-\Delta x,t)}{2\Delta x}$$

Solve PDEs Numerically

- Compact notation: x→j, t→n



- $\partial_t \rho + v\partial_x \rho = 0 \rightarrow \rho_j^{n+1} = \rho_j^n - \frac{v\Delta t}{2\Delta x}(\rho_{j+1}^n - \rho_{j-1}^n)$
- **Initial and boundary conditions** are required
  - E.g., cosmological simulations
    - Initial condition: from CMB power spectrum
    - Boundary condition: periodic

von Neumann Stability Analysis

- Numerical solutions can be unstable if the finite difference scheme and/or the evolution time (Δt) are not designed carefully
  - Small errors can grow exponentially and screw up your simulations !!
- von Neumann stability analysis
  - Linearize the (discretized) PDEs
  - Insert the plane-wave solution: $\rho_j^n = \rho_k e^{i(kj\Delta x - wn\Delta t)} = \rho_k \xi^n e^{i(kj\Delta x)}$, where w $= w_R + iw_I$ is in general complex and $\xi \equiv e^{-iw\Delta t}$
  - Solve the stability condition: $w_I < 0 \rightarrow |\xi| < 1$
    - Similar to the Jeans instability analysis

von Neumann Stability Analysis

- Example: advection eq. again with the FTCS scheme

$$\rho_j^{n+1} = \rho_j^n - \frac{v\Delta t}{2\Delta x}\left(\rho_{j+1}^n - \rho_{j-1}^n\right)$$

$$\xi^{n+1} e^{ik\Delta x j}$$

$$= \xi^n e^{ik\Delta x j} - \frac{v\Delta t}{2\Delta x}\left[\xi^n e^{ik\Delta x(j+1)} - \xi^n e^{ik\Delta x(j-1)}\right]$$

$$\xi = 1 - i\frac{v\Delta t}{\Delta x}\sin(k\Delta x)$$

$$|\xi|^2 = 1 + \frac{v^2\Delta t^2}{\Delta x^2}\sin^2(k\Delta x) \geq 1$$

- FTCS scheme is unconditionally unstable !!

von Neumann Stability Analysis

- **Lax method:** $\rho_j^n \rightarrow \frac{1}{2}\left(\rho_{j+1}^n + \rho_{j-1}^n\right)$

  $\xi = \cos(k\Delta x) - i\frac{v\Delta t}{\Delta x}\sin(k\Delta x)$

  $|\xi|^2 = 1 + \left[\frac{v^2\Delta t^2}{\Delta x^2} - 1\right]\sin^2(k\Delta x) \leq 1$

  assuming $\Delta t \leq \frac{\Delta x}{|v|}$ **→ Courant condition**

- **Designing a finite difference scheme is non-trivial !!**

- **In general:** $\Delta t \leq \frac{\Delta x}{\left|v_{info}^{max}\right|}$, **where** v_{info}^{max} **is the maximum information propagation speed**

Hydrodynamics

- **Euler + Poisson equations:**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0$$

$$\frac{\partial (\rho \vec{v})}{\partial t} + \nabla \cdot (\rho \vec{v}\vec{v} + P) = -\rho \nabla \varphi \; (+\text{sources}\ldots)$$

$$\frac{\partial e}{\partial t} + \nabla \cdot (\vec{v}(e + P)) = -\rho \vec{v} \cdot \nabla \varphi \; (+\text{sources}\ldots)$$

$$\nabla^2 \varphi = 4\pi G \rho$$

ρ: density
v : velocity
P : pressure
φ: potential
e : energy density

- **Cosmology → universe is expanding → better to work in the comoving coords.**
  - **By choosing proper comoving variables (e.g., $\tilde{\rho} \equiv a^3 \rho$), the governing eqs. can remain almost untouched except $\partial_t \rightarrow a^2 \partial_t$ and $G \rightarrow aG$**

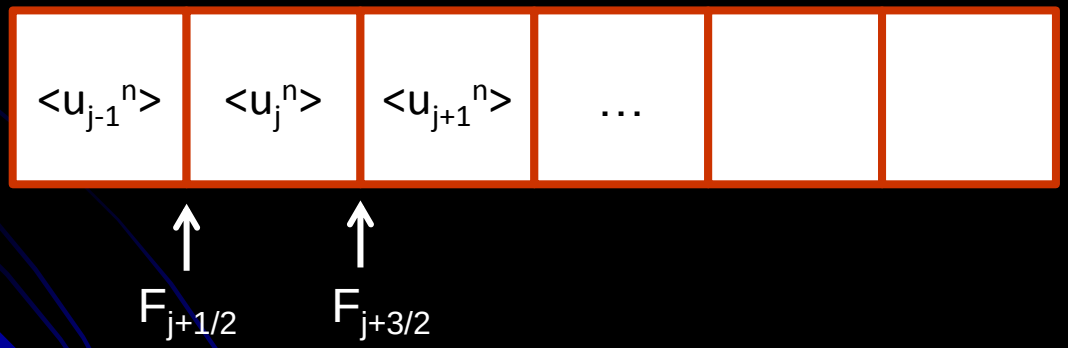
Finite Volume Methods

- **Rewrite the Euler eqs. into the integral forms:**

$$\partial_t u + \nabla \cdot \vec{F}(u) = 0$$

$$\rightarrow \partial_t \langle u \rangle + \frac{1}{V} \oint \vec{F} \cdot d\vec{s}, \text{ where } \langle u \rangle \equiv \frac{1}{V} \int u dV$$

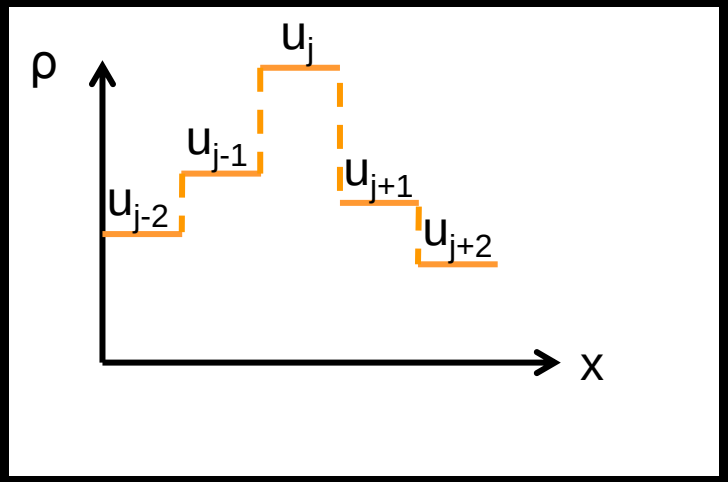
  - **Primary variables become the volume-averaged quantities at the jth cell → $\langle u \rangle_j$**
  - **Main problem is to estimate the fluxes across the jth cell interfaces → $F_{j-1/2}$ & $F_{j+1/2}$**



  - **Advantage: guarantee the conservation laws (even with self-gravity)**

Shock-Capturing Flux Estimation

- Godunov method:
  - Assuming $\langle u \rangle_j$ is piecewise constant within each cell
    - Higher-order data reconstruction can be adopted (piecewise linear, piecewise parabolic …)

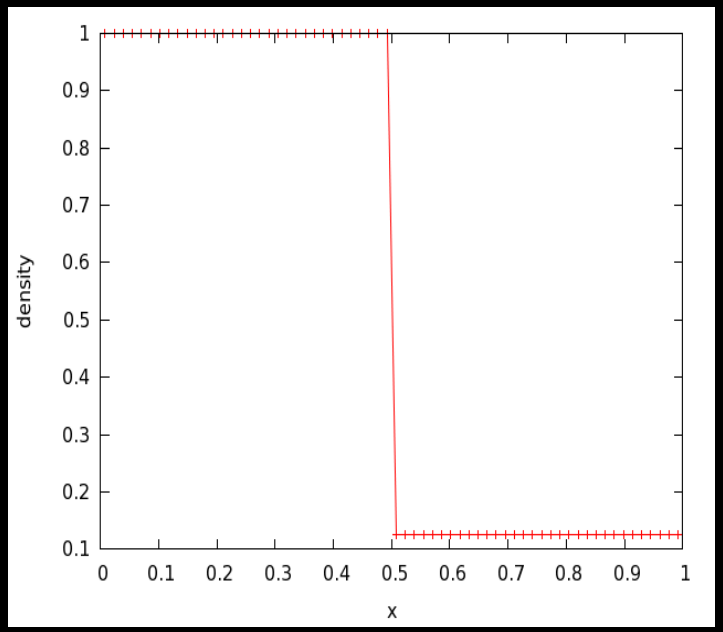


  - Solve the Riemann problem

$$u(x, t = 0) = \begin{cases} u_L, x \leq 0 \\ u_R, x > 0 \end{cases}$$

→ $u(x, t > 0)$ can be solved analytically

- But is computationally expensive
- Approximate Riemann solver (e.g., Roe's method, HLLE, HLLE, …)

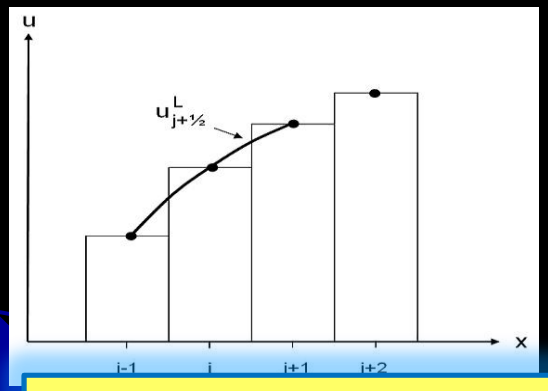


Time Integration

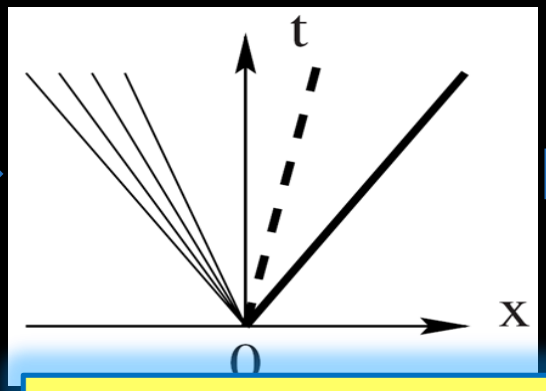
- $\partial_t u(x,t) \rightarrow \frac{1}{\Delta t}[u(x,t+\Delta t) - u(x,t)]$ is only first-order accurate in time
- 2nd-order Runge-Kutta (mid-point) method
  - (1) u(x,t) $\xrightarrow{\text{Riemman solver}}$ F(x,t)
  - (2) u(x,t) $\xrightarrow{\nabla \cdot F(x,t)}$ u(x,t+Δt/2)
  - (3) u(x,t+Δt/2) $\xrightarrow{\text{Riemman solver}}$ F(x,t+Δt/2)
  - (4) u(x,t) $\xrightarrow{\nabla \cdot F(x,t+\Delta t/2)}$ u(x,t+Δt)
- Courant condition: : $\Delta t \leq \alpha_{CFL} \frac{\Delta x}{|v_{max}|+c_s}$
  - α_{CFL} is a constant depending on the numerical scheme, and c_s is the sound speed

Corner Transport Upwind Scheme

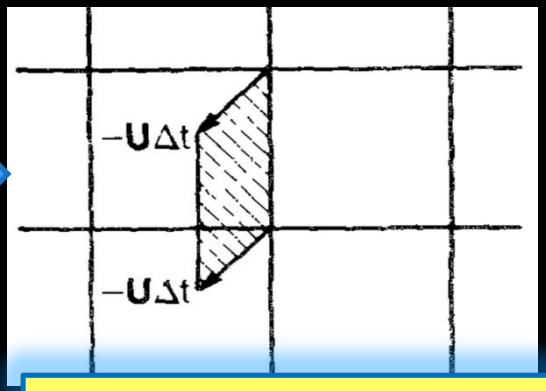
- **Colella, P., 1990. J. Comput. Phys. 87, 171.**
- **Extended to MHD and well tested in the Athena code**
  - Stone, J.M., Gardiner, T.A., Teuben, P., Hawley, J.F., Simon, J.B., 2008. ApJS 178, 137.



[1] Data reconstruction
- Piecewise linear
- Piecewise parabolic
- Different slope limiters



[2] Riemann solver
- Exact solver
- Roe's solver
- HLLE/HLLC



[3] Transverse flux gradients

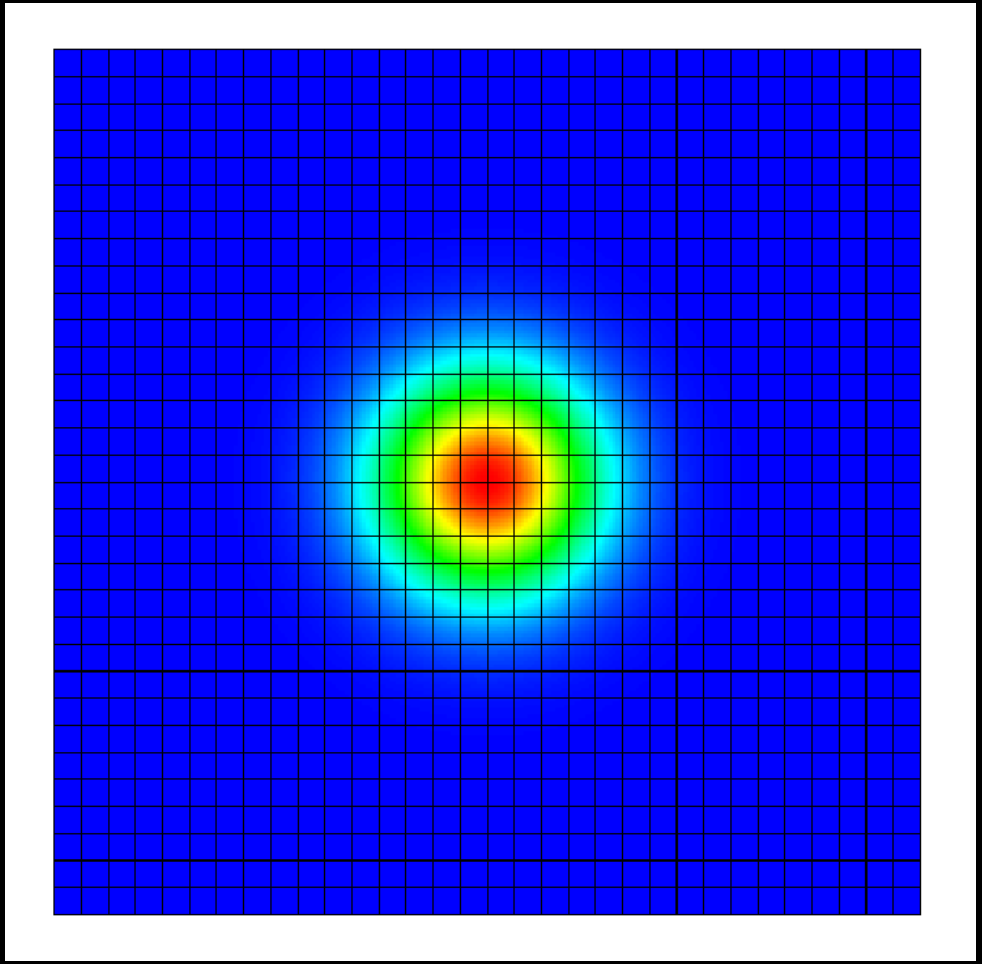
[4] Riemann solver

[5] Update solution

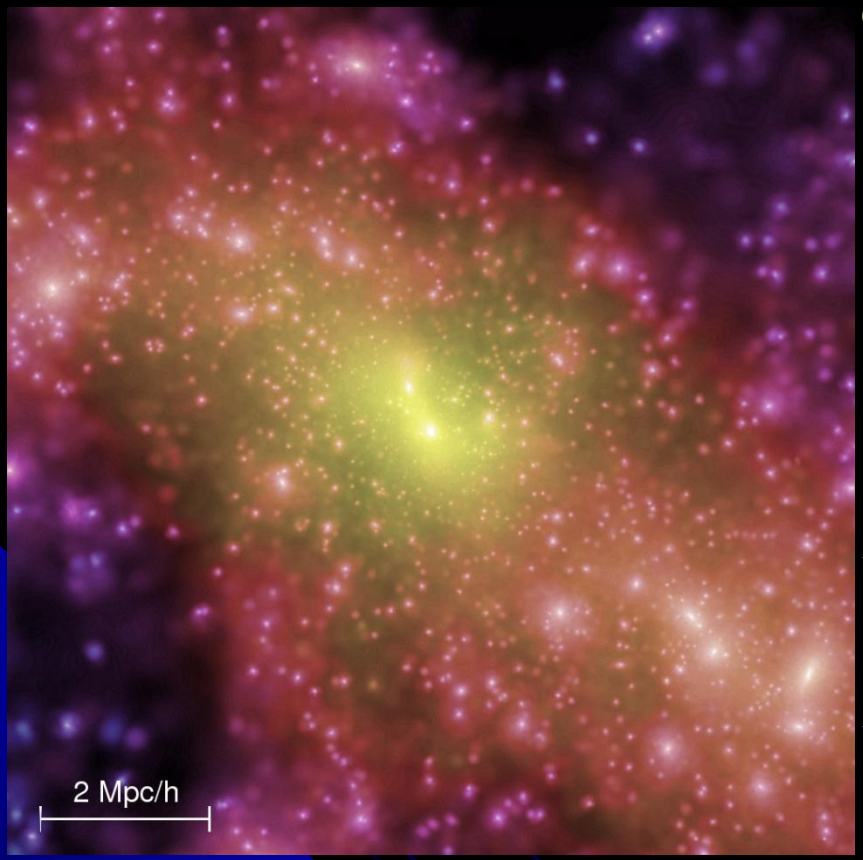
Adaptive-Mesh-Refinement

Uniform Mesh

- Pros
  - Relatively easy to program
  - Relatively easy to parallelize
- Cons
  - Waste computational time
  - Waste memory
  - Lower resolution



Why AMR ?

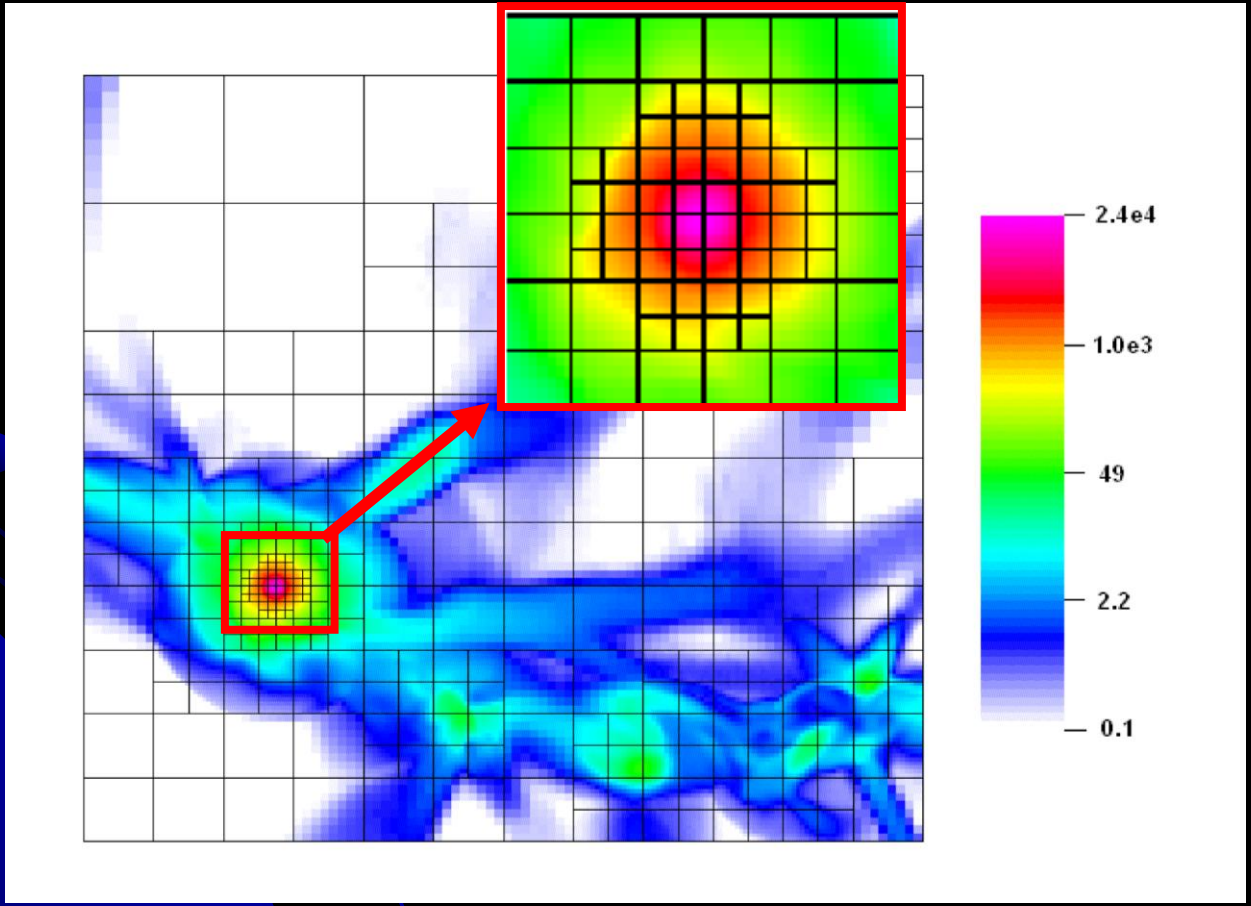


Millennium simulation by Springel et al. 2005

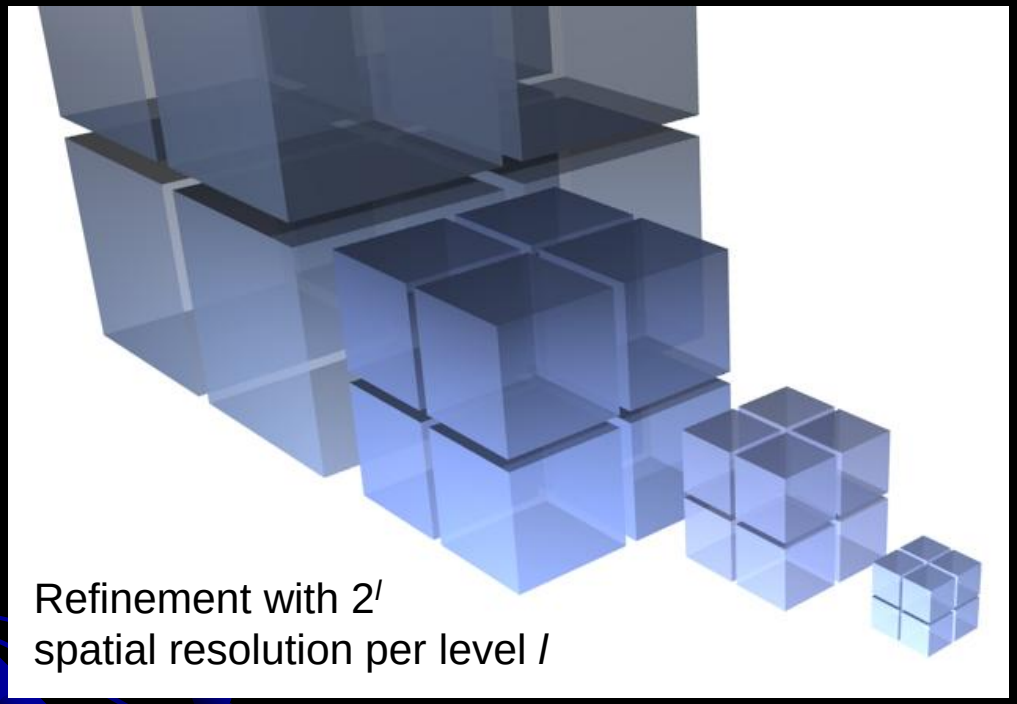
- **Cosmological simulations**
  - **Typical scale ~ 100 Mpc/h**
  - **Minimum resolution for resolving galaxies ~ 1 kpc/h**
  - **~$(10^5)^3 = 10^{15}$ cells required**
  - **memory requirement ~ 10^8 GB**
  - **→ infeasible in most supercomputers**
- **Hint: lots of void regions in the universe**
- **Solution: use computational resource more efficiently by focusing only on the high-overdensity regions**

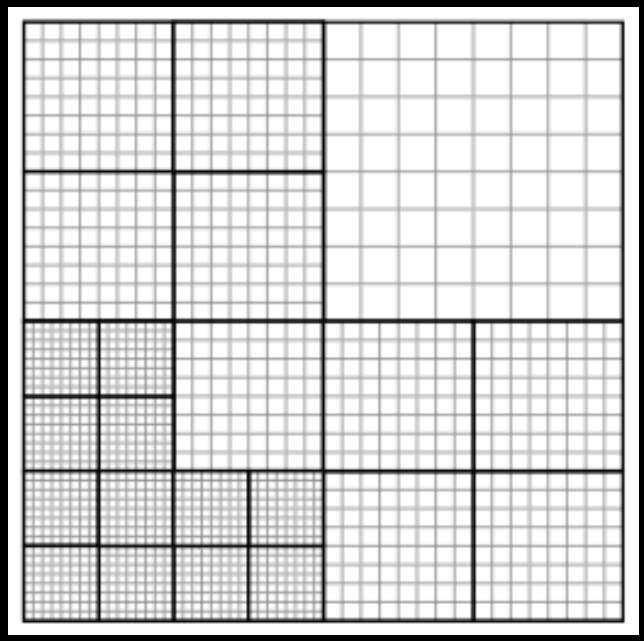
Adaptive Mesh Refinement (AMR)

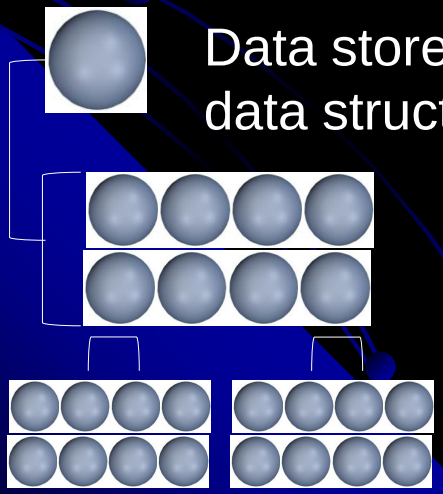
- Make resolution adaptively change with space and time
- Flexible refinement criteria (density, velocity, …)



Octree Data Structure







Data stored in octree data structure

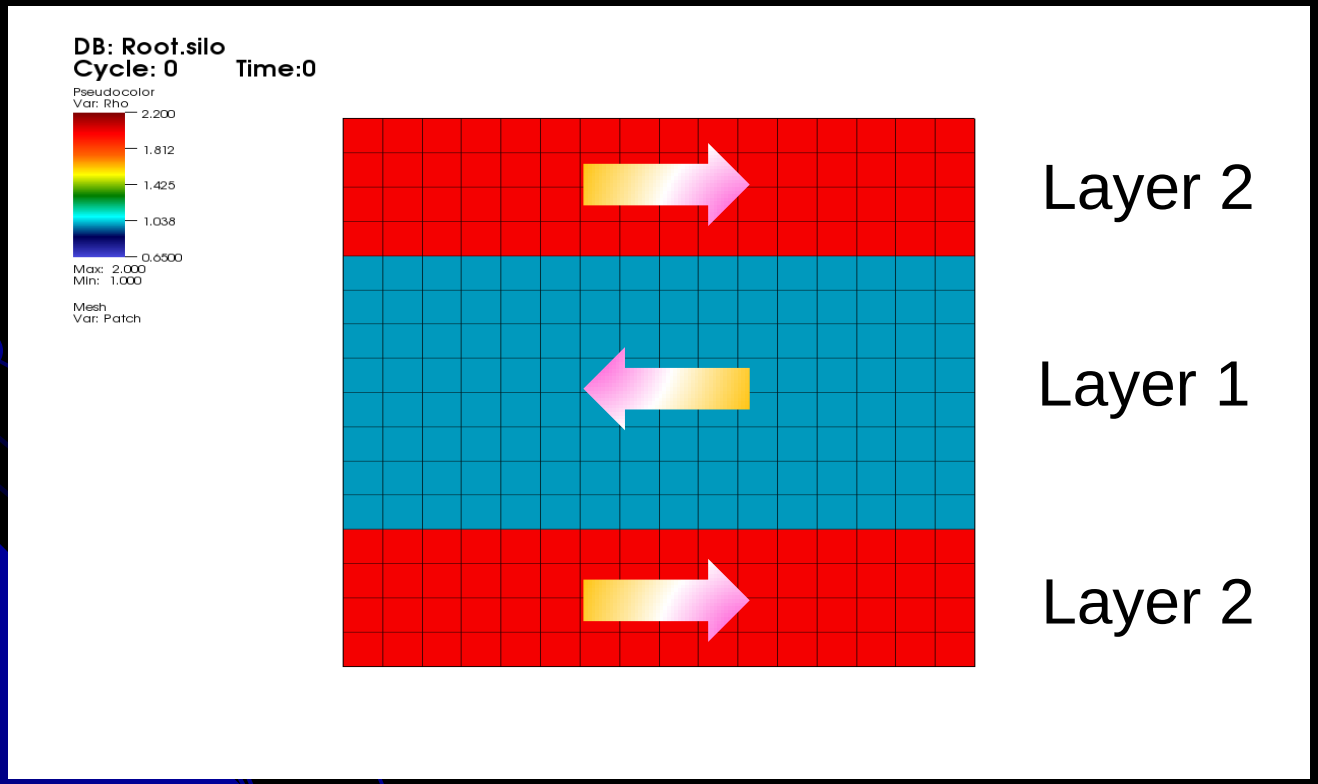
8^3 cells per patch

Identical spatial geometry

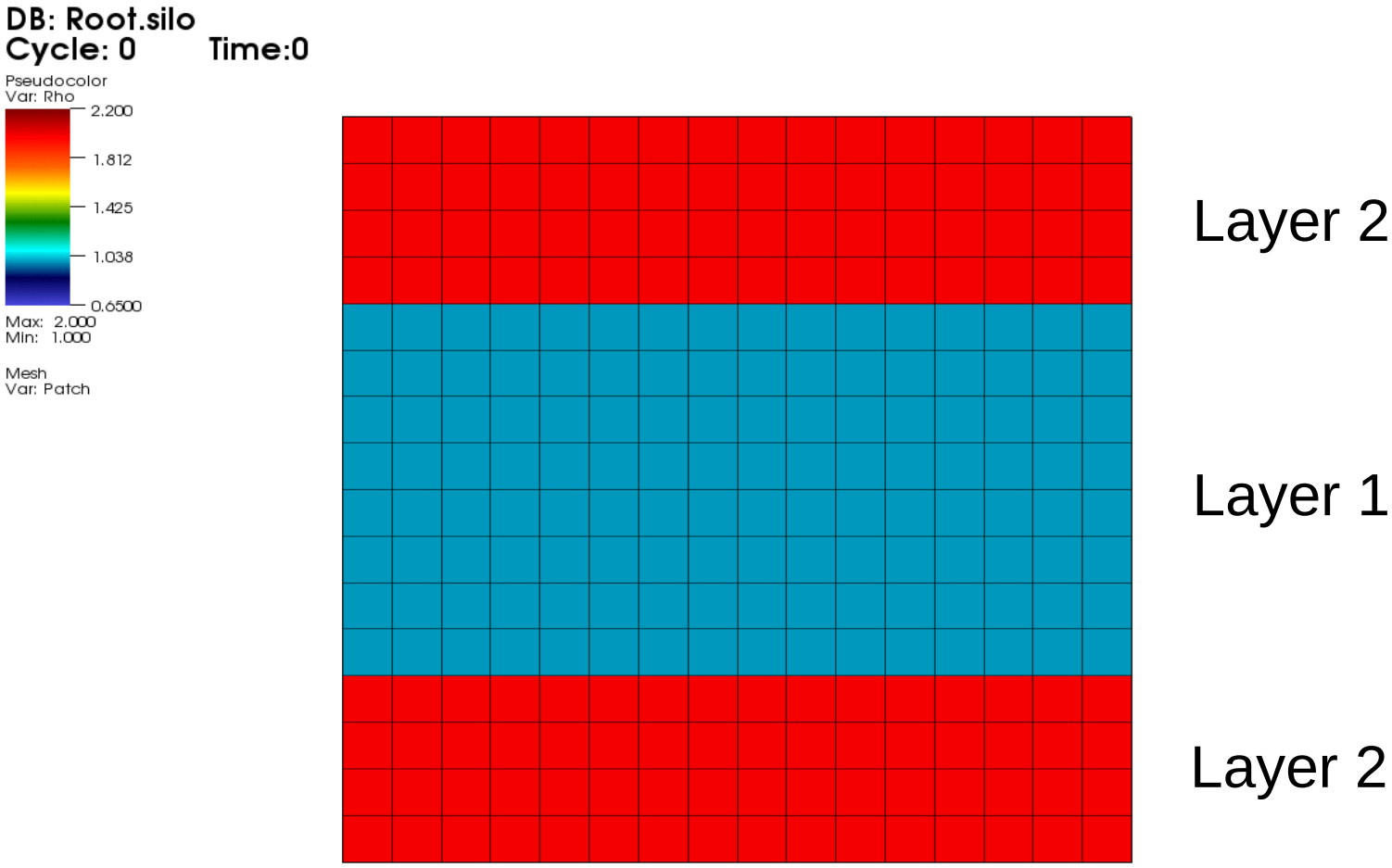
Slide created by Hemant Shukla at LBNL

AMR Example I

- Kelvin-Helmholtz instability
- Refinement criterion: vorticity magnitude $|\nabla \times \vec{V}|$
- Base level 128^2, refined level 4 → $2{,}048^2$ effectively

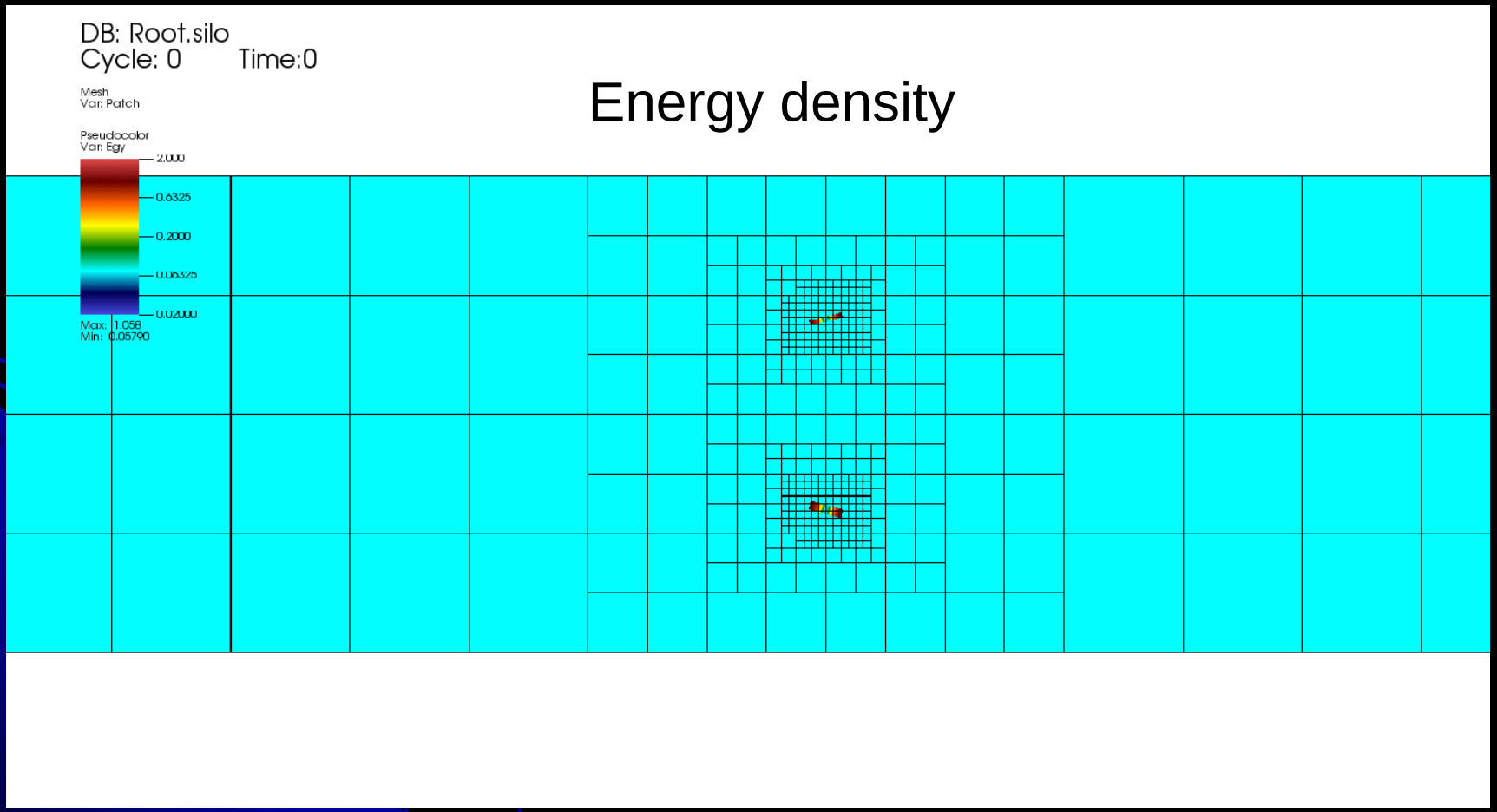


AMR Example I



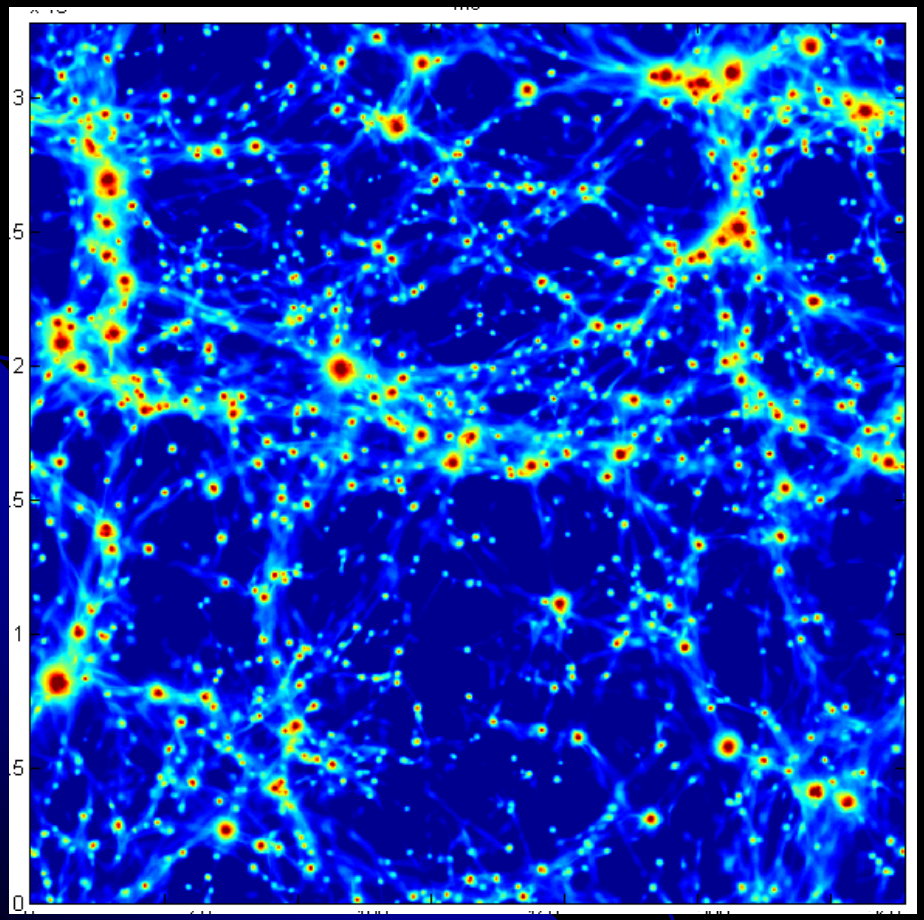
AMR Example II: AGN Jet

- Bidirectional outflow is continuously injected (by Sandor Molnar)



AMR Example III: LSS

- Large-scale structure simulations (purely baryonic) with different refinement levels
- Refinement criterion: density

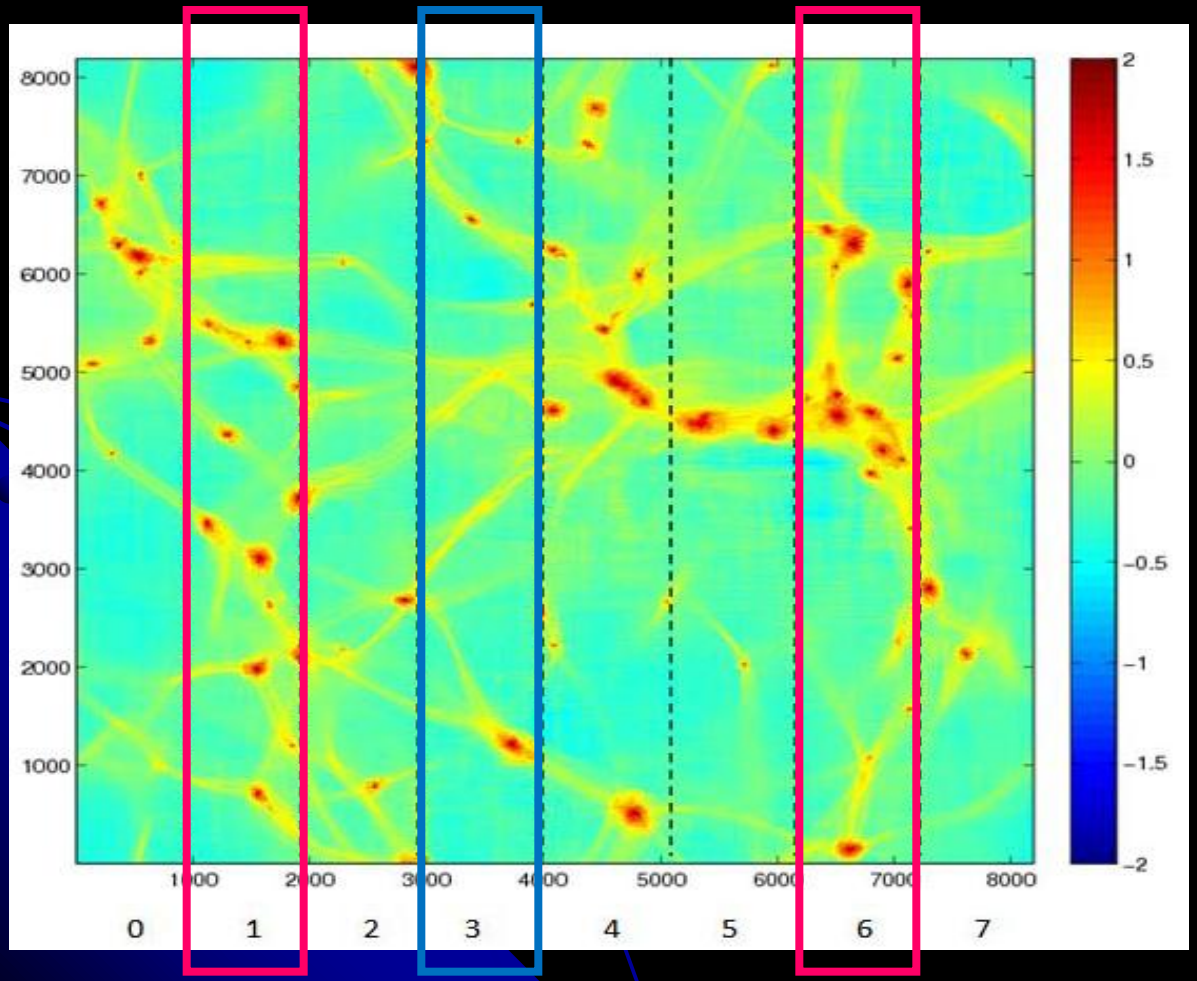


Base level 256^3, Refined level 0

Base level 256^3, Refined level 5
(effective resolution = $8{,}192^3$)

Issue of Load Imbalance

- The rectangular domain decomposition can lead to an issue of load imbalance.



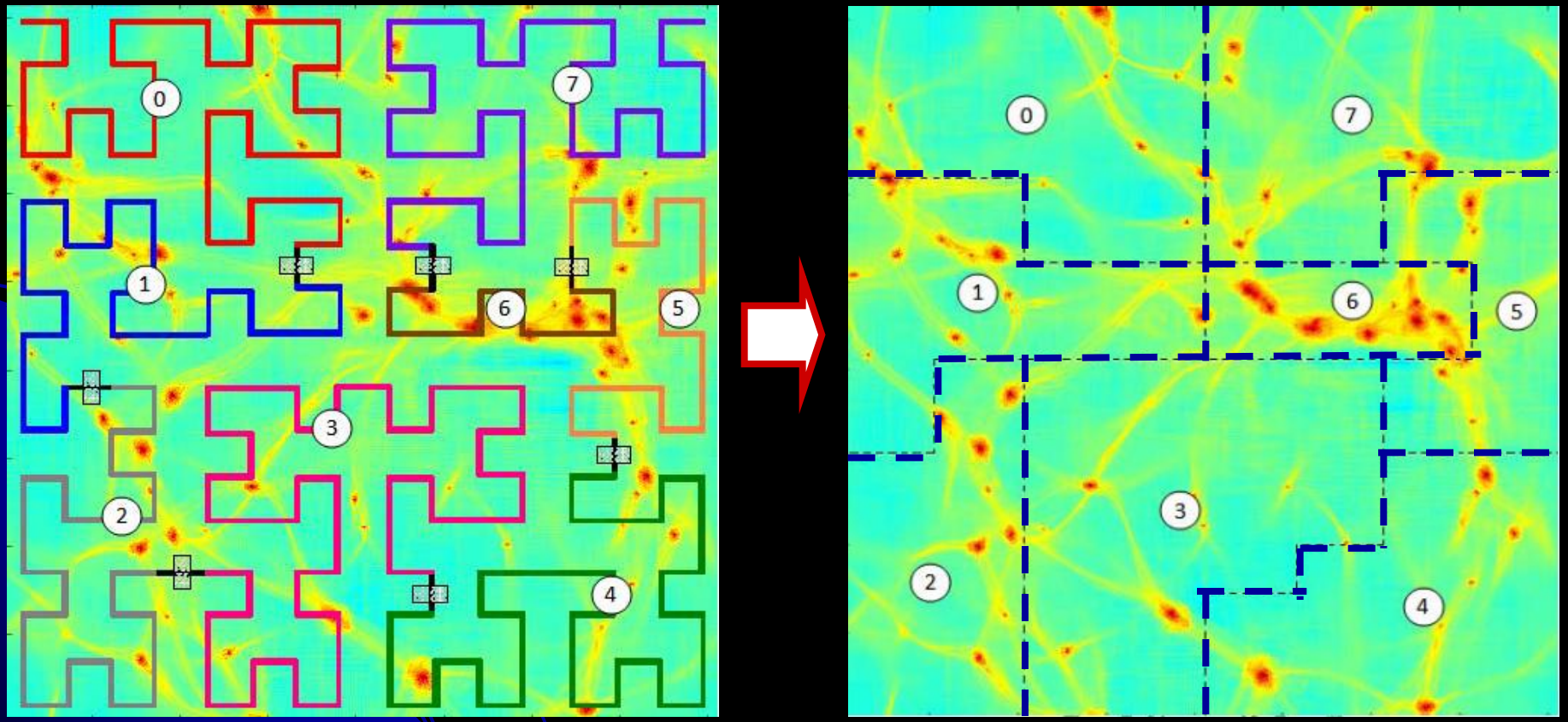
More load

Less load

Space-filling Curve for Domain Decomposition

- **Hilbert space-filling curve** domain decomposition

  → load balance, data locality



GPU

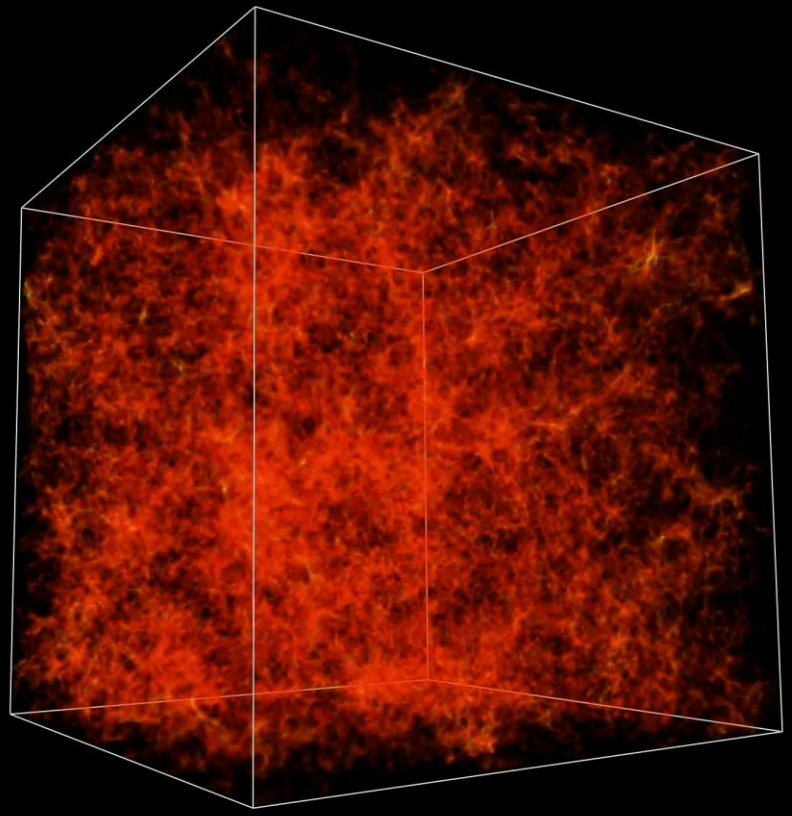
Graphic-Processing-Unit

Graphic-Processing-Unit (GPU)

GeForce GTX 680

Animations, video games, data visualization …

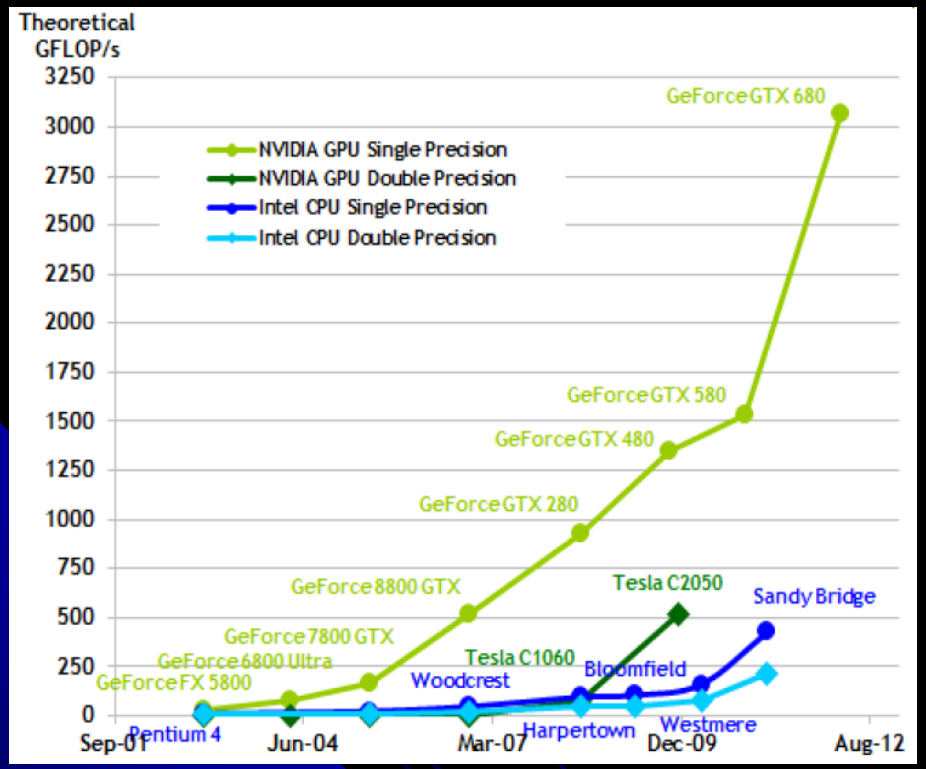
Graphic-Processing-Unit (GPU)

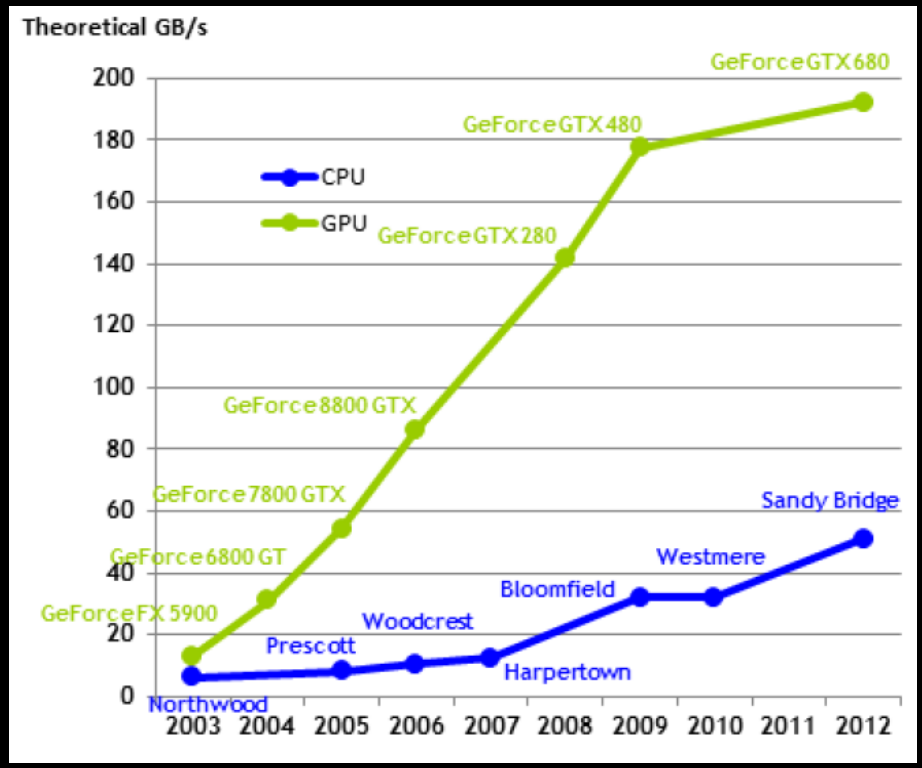


GeForce GTX 680

Astrophysics !?

GPU: High Performance & High Bandwidth





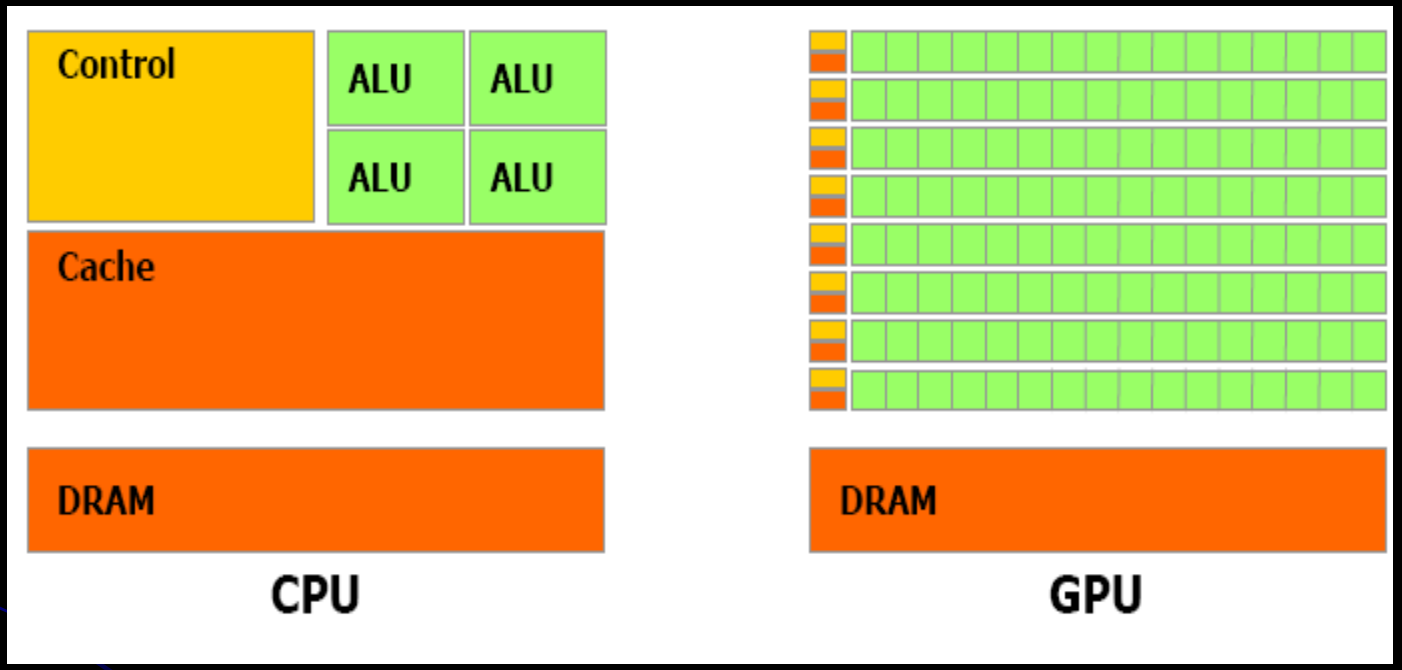
Performance: GPU vs. CPU

~ 7x

Bandwidth: GPU vs. CPU

~ 4x

Why is GPU faster ?

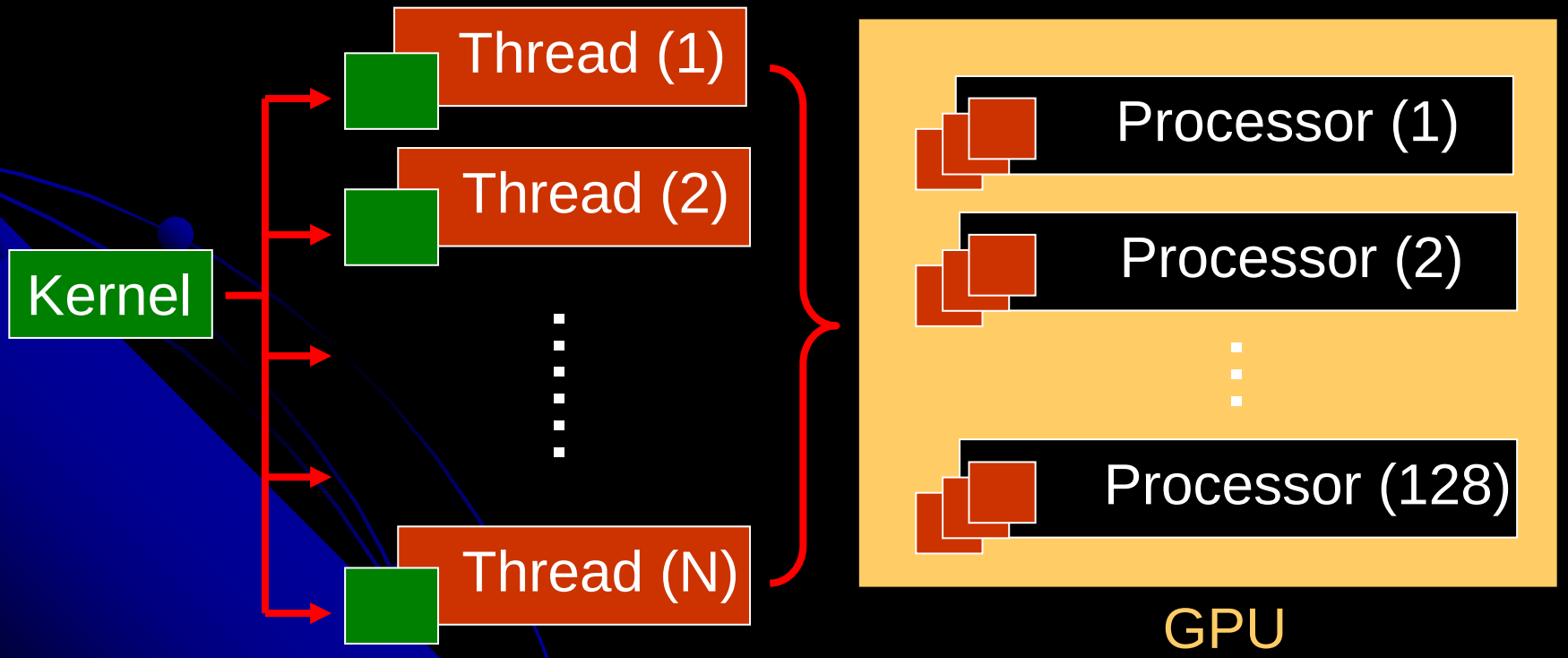


- In GPU, more transistors are devoted to data processing
- GPU is suitable for data-parallel computations with high arithmetic intensity

Programming interface : CUDA

(Compute Unified Device Architecture)

- GPU → multithreaded coprocessor to CPU
  - Execute thousands of threads in parallel
  - All threads execute the same kernel



C vs. CUDA

- **C-like** programming language
  - Example: vector manipulation for Array[N]

```
void Kernel( float Array[] )
{
    for (int i=0; i<N; i++)
    Array[i] += 1.0;
}
```

C

```
// declare N threads in advance
__global__
void Kernel( float Array[] )
{
    int i = threadIdx.x;
    Array[i] += 1.0;
}
```

CUDA

Our Home-made GPU Cluster

34 nodes, 2 GPUs per node
→ 68 GPUs in total

CPU : Intel i7-3930K (six-core)
GPU : NVIDIA GeForce GTX 560 Ti

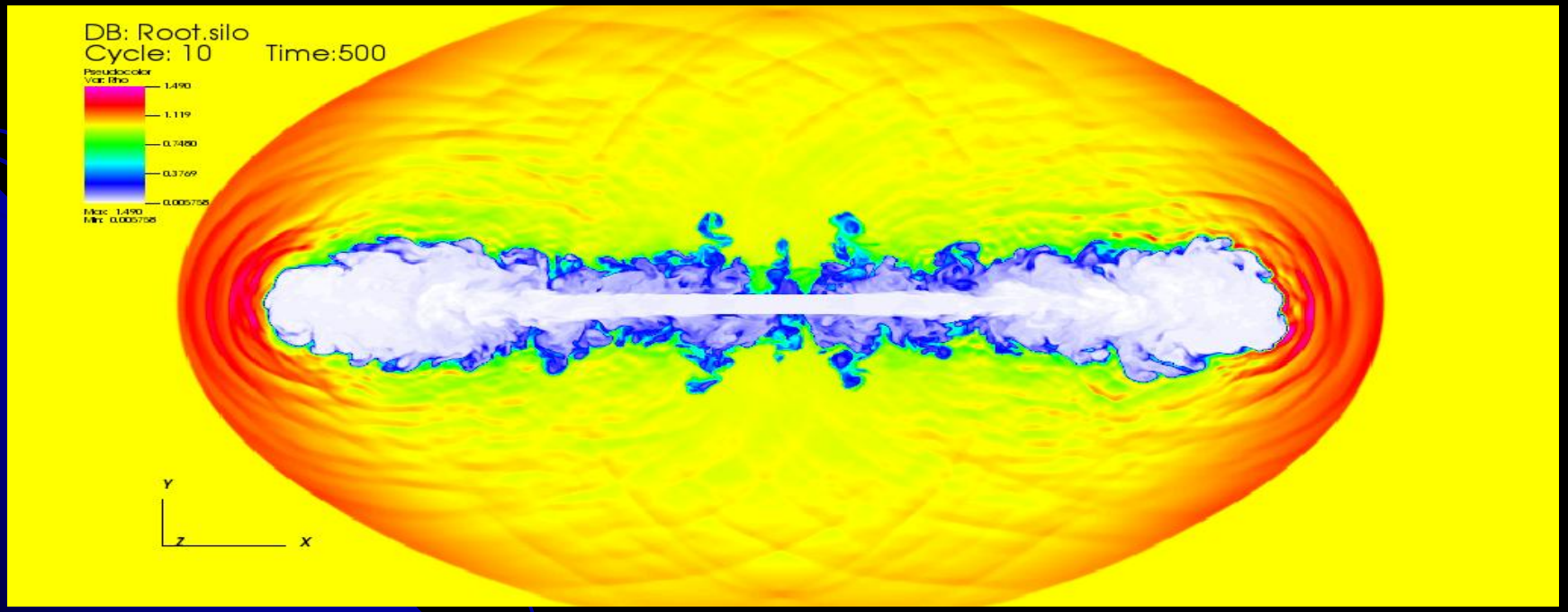
Network : Gigabit Ethernet

Theoretical performance :
49 TFLOPS

GAMER

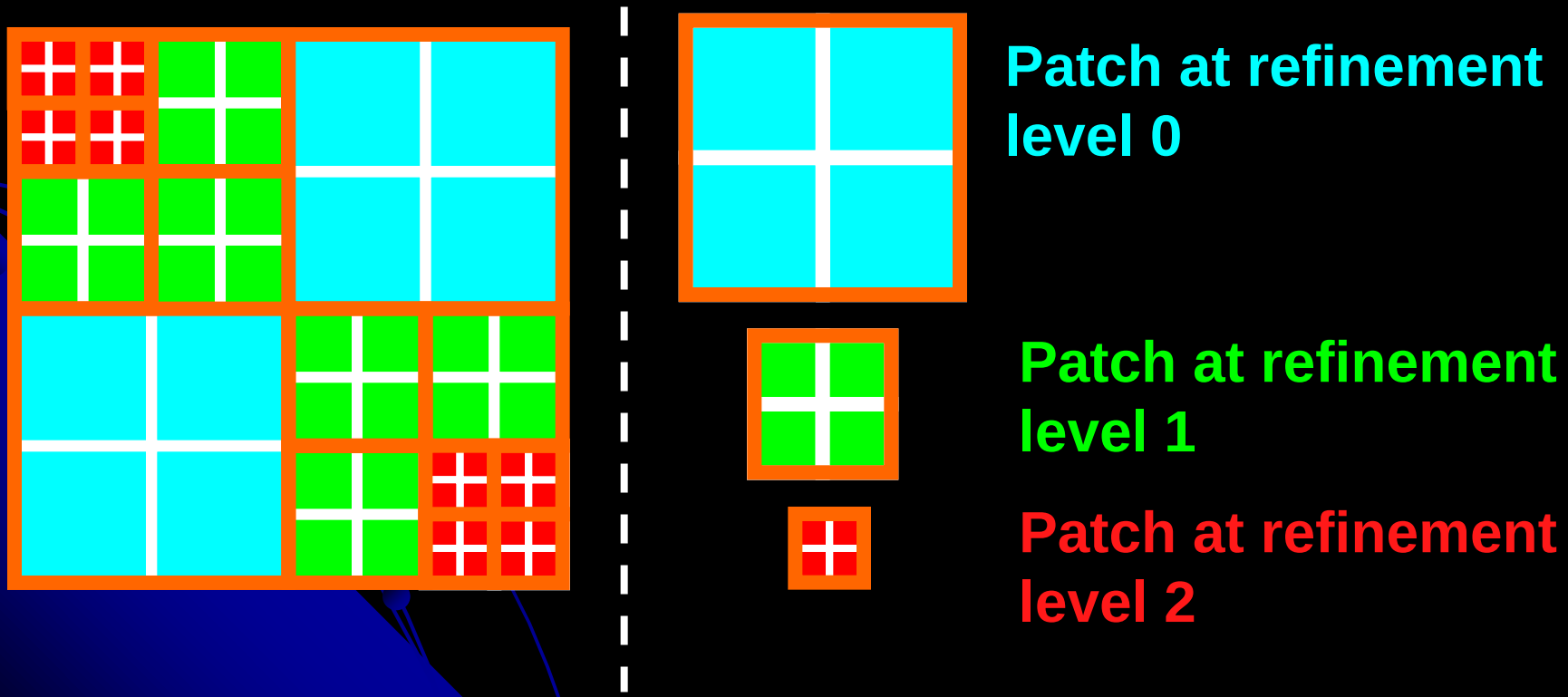
GPU-accelerated Adaptive MEsh Refinement

Code for Astrophysics

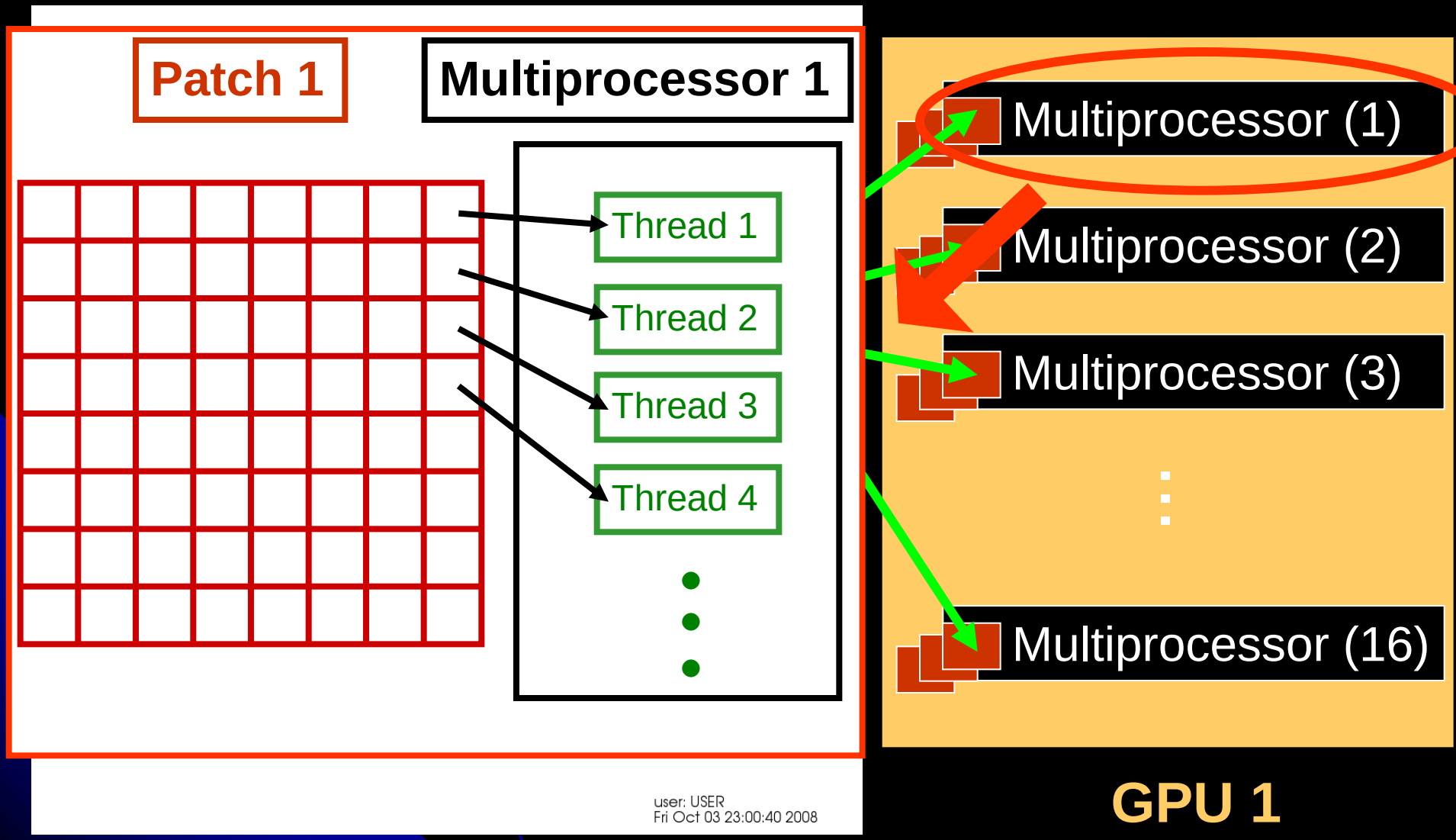


AMR in GAMER

- Refinement unit : patch (containing a fixed number of cells, e.g., 8^3)
- Hierarchical octree data structure
- Ref: Schive, H-Y., et al. 2010, ApJS, 186 457



Example : Blast Wave Test



CPU-GPU Collaboration

- **Two main tasks in AMR:**

1. **Patch construction : decision making, interpolation, complex data-structure, data assignment …**

   ~ complicated, but consume less time

   ➡ CPUs

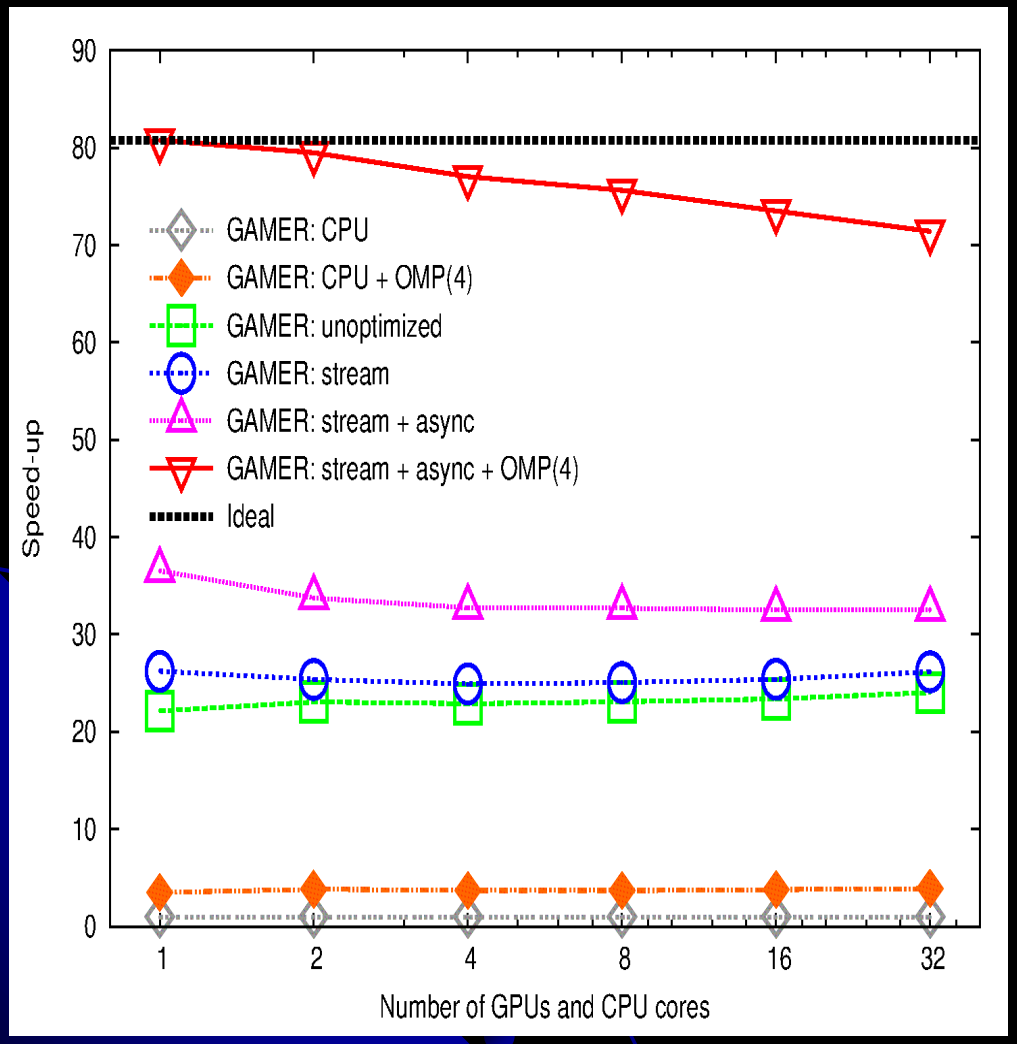
2. **3-D PDE solvers (e.g., hydrodynamics, Poisson) :**

   ~ straightforward, but time-consuming

   ➡ GPUs

   ➡ feed with hundreds of patches simultaneously

Multi-GPU Performance



NERSC Dirac GPU Cluster

GPU: 1-32 NVIDIA Tesla C2050

CPU: 1-32 Intel Xeon E5530

With self-gravity (80x speed-up in GPU) and individual time-step

Stream : PCI-E/GPU overlap

Async : CPU/GPU overlap

OMP(4) : 4 OpenMP threads

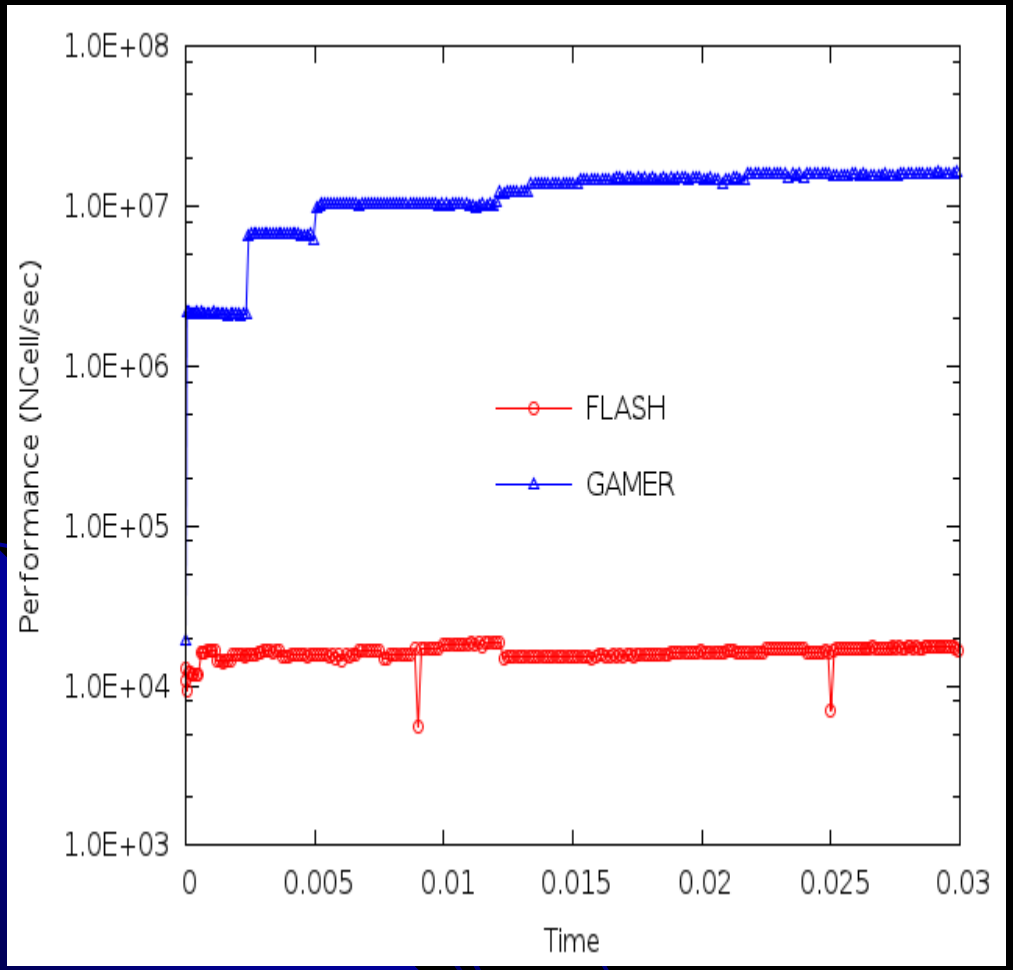
32 GPU vs. 32 CPU cores: 71x

32 GPU vs. 128 CPU cores: 18x

→ Equivalent to 2,304 CPU cores

MPI ~ 11% of T_{total}

GAMER vs. FLASH



NASA Pleiades GPU Cluster

GPU: 64 NVIDIA Tesla M2090

CPU: 768 Intel Xeon X5670

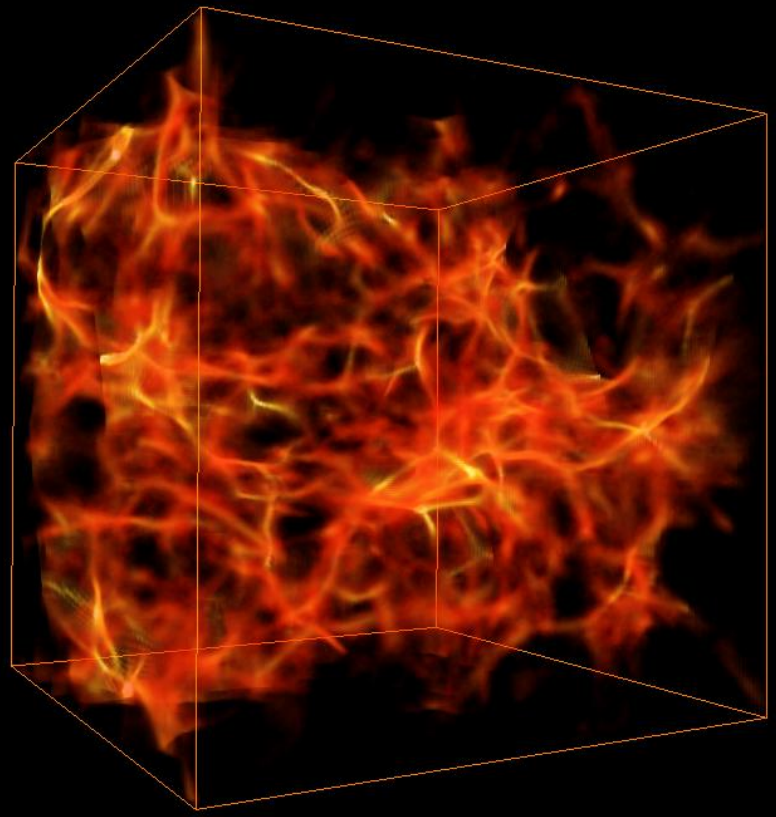
Blast-wave test (pure hydro)

8 GPUs vs. 360 CPU cores : 2x

1 GPU vs. 1 CPU core : 1000x

ψDM

Wave Dark Matter



Cold Dark Matter

- **CDM (Cold Dark Matter):**
  - **Collisionless particles with self-gravity**
  - **Work very well on large scales**
  - **Controversial on small scales (dwarf galaxies)**

- **Main issues on small scales:**
  - **Missing satellites problem**
    - **→ Over abundance of dwarf galaxies ?**
  - **Cusp-core problem**
    - **→ Mass is too concentrated at the center ?**

Wave Dark Matter (ψDM)

- Governing eq.: **Schrödinger-Poisson** eq. in the comoving frame

$$i\frac{\partial\psi(x)}{\partial t} = -\frac{1}{2\eta}\nabla^2\psi(x) + \eta\varphi(x)\psi(x),$$

$$\nabla^2\varphi(x) = 4\pi G a(t)(|\psi(x)|^2 - 1)$$

η ≡ m/ħ: particle mass, **ψ**: wave function
φ: gravitational potential, **a**: scale factor

- Background density has been normalized to unity

Quantum Fluid

- **Schrödinger eq. can be rewritten into conservation laws**

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0,$$

$$\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} = \nabla\left(\frac{1}{2\eta^2}\frac{\nabla^2 f}{f}\right) - \nabla\varphi$$

$$\psi = f e^{iS/\hbar},$$

$$\rho = m f^2,$$

$$v = \eta^{-1}\nabla S$$

$$\text{Hydro}: \frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \nabla \vec{v} = -\frac{1}{\rho}\nabla P - \nabla\varphi$$

$$\tilde{P}_{ij} = \frac{\hbar^2}{m}\left(\partial_i f \partial_j f - \frac{1}{4}\delta_{ij}\nabla^2 f^2\right)$$

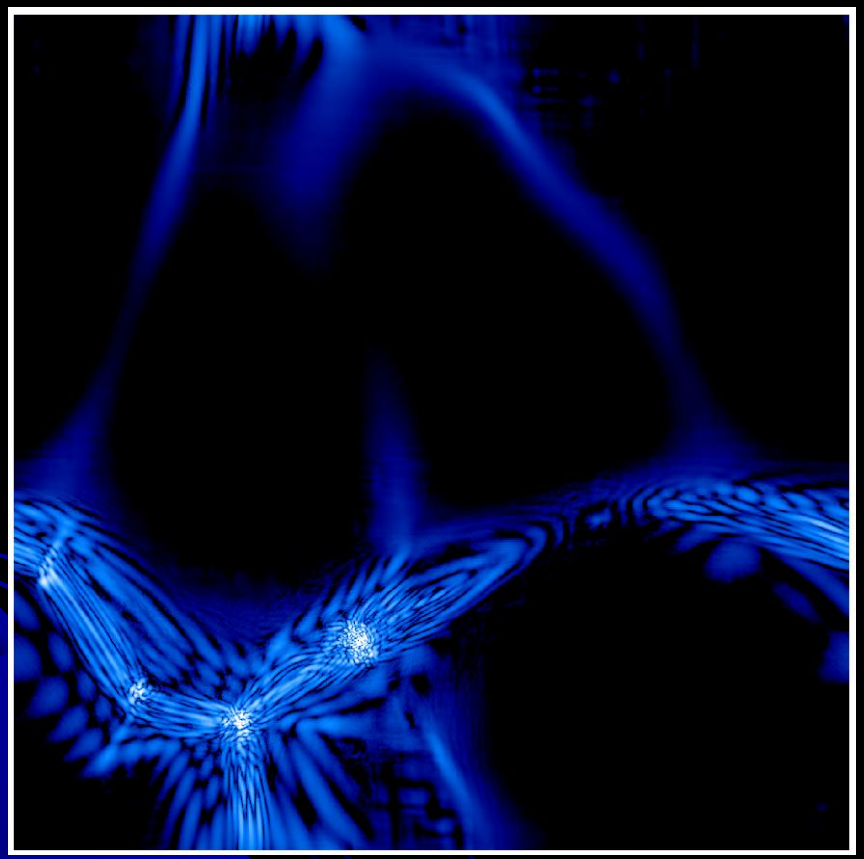
quantum stress

$$k_J = (6a)^{1/4}(H_0\eta)^{1/2}$$

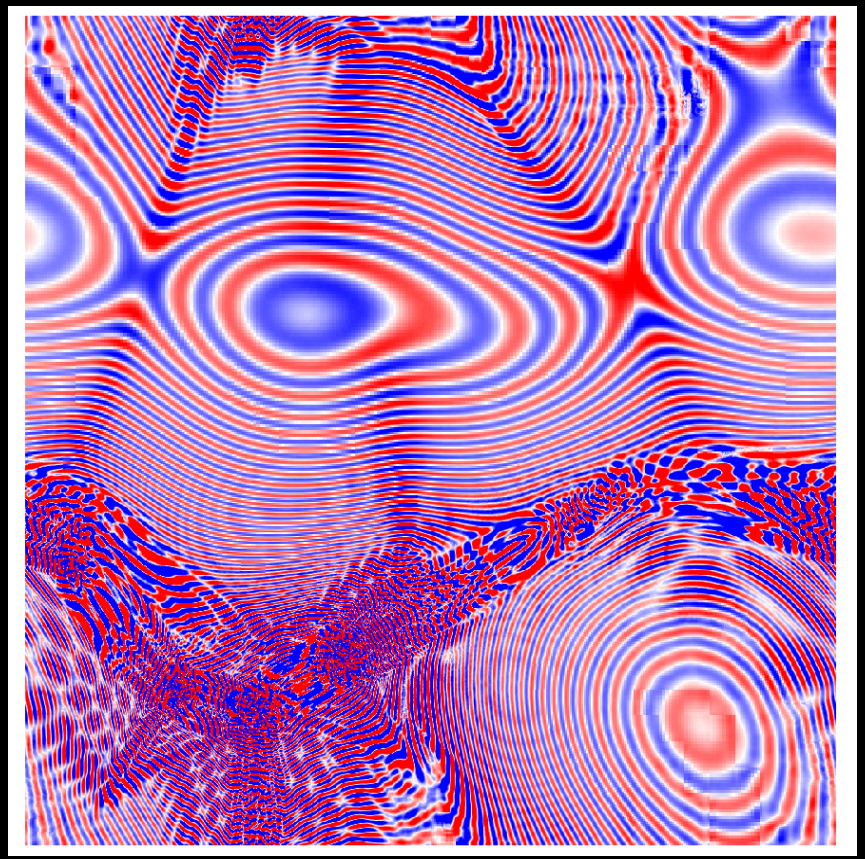
Jeans wavenumber in ψDM

Numerical Challenge

Density

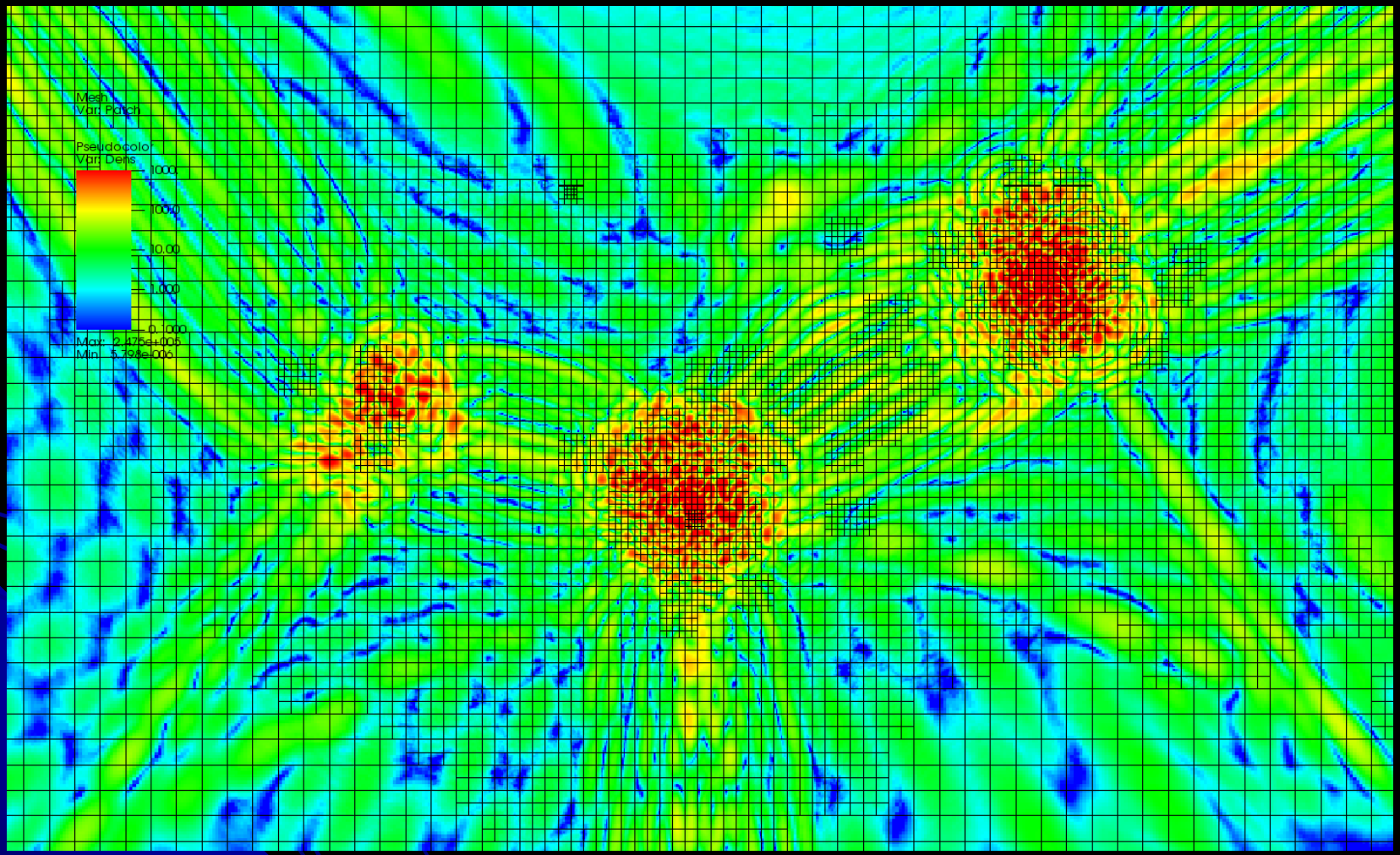


Wave function

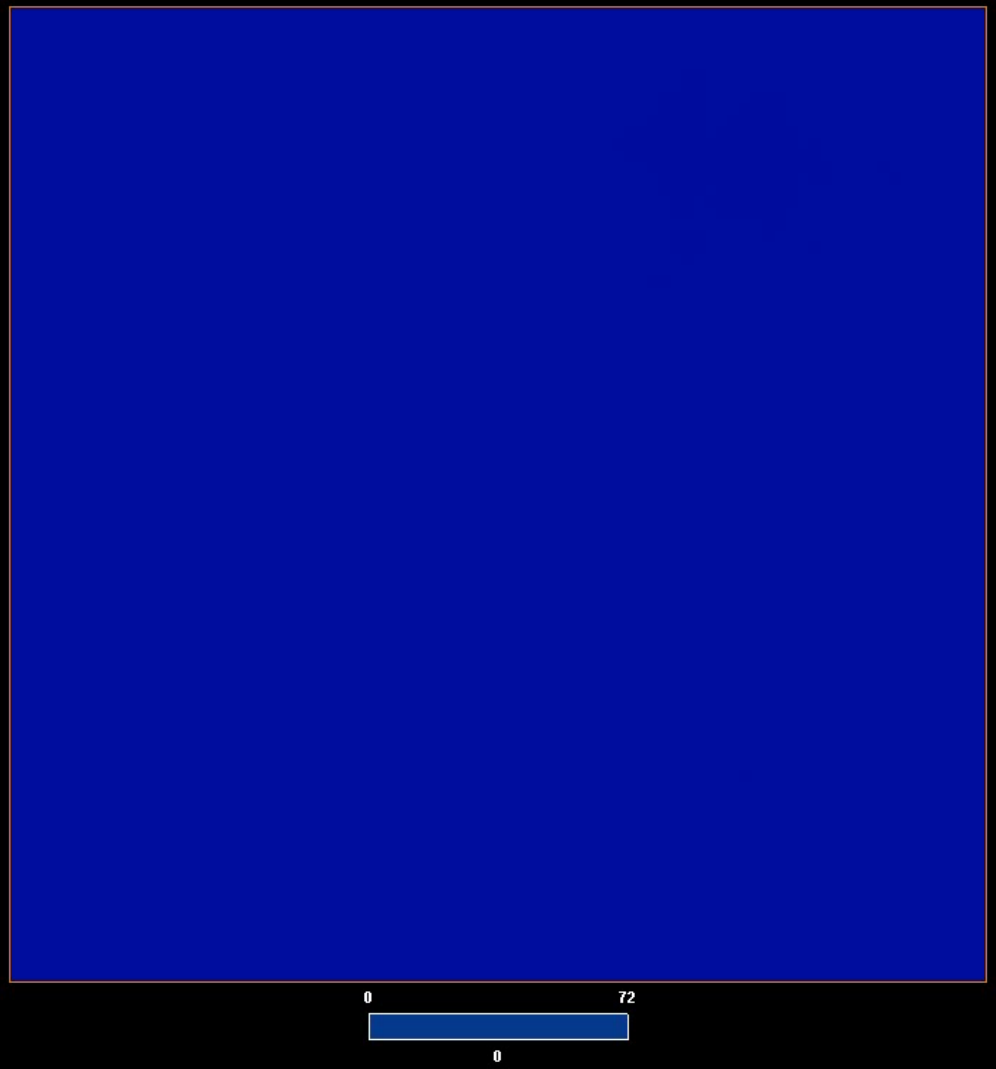


→ **Ultra-high resolution ($v \propto \lambda^{-1}$) and extremely small time-step ($\Delta t \propto \Delta x^{-2}$) are required !!**

AMR Hierarchy

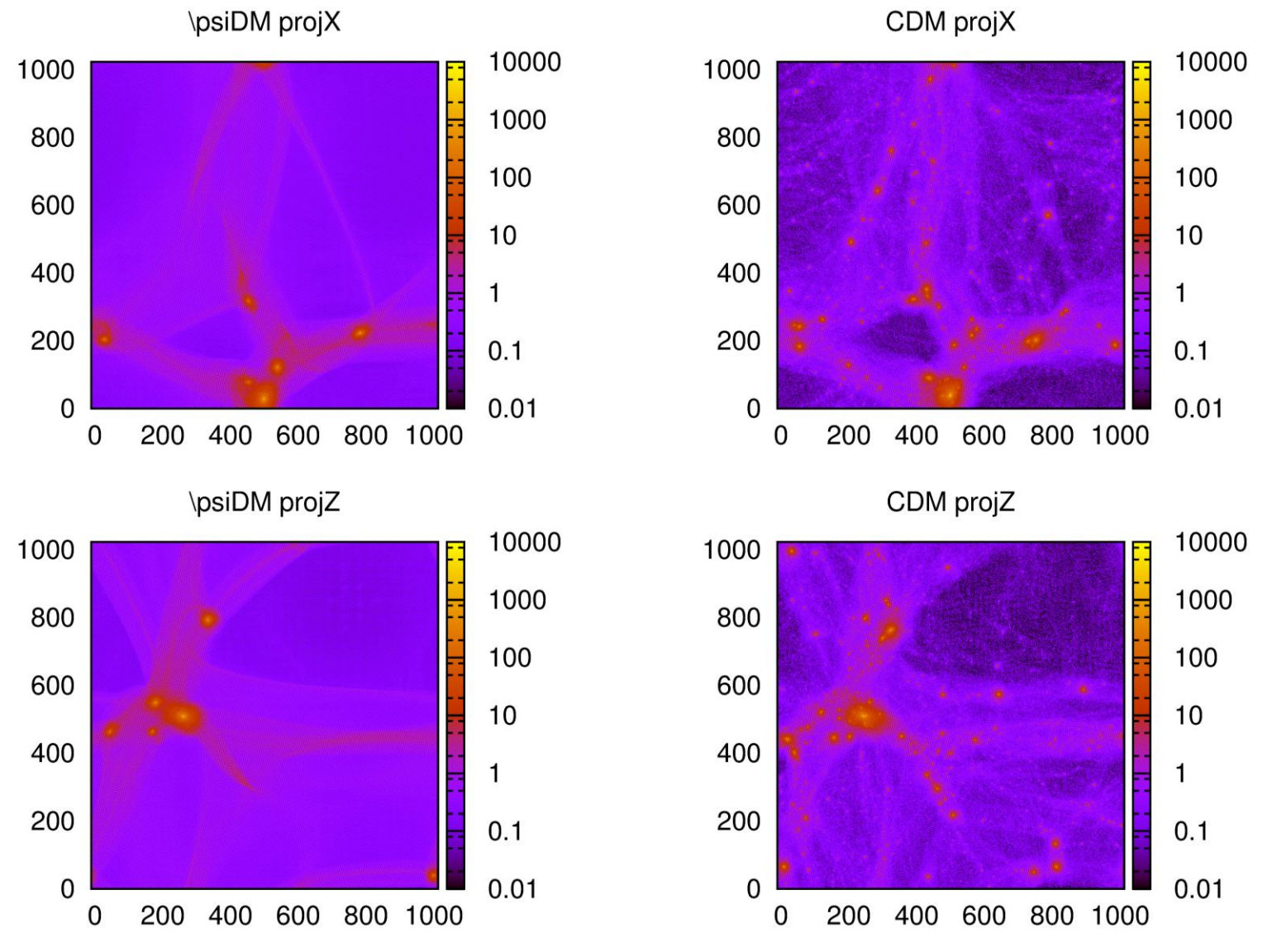


LSS Simulation



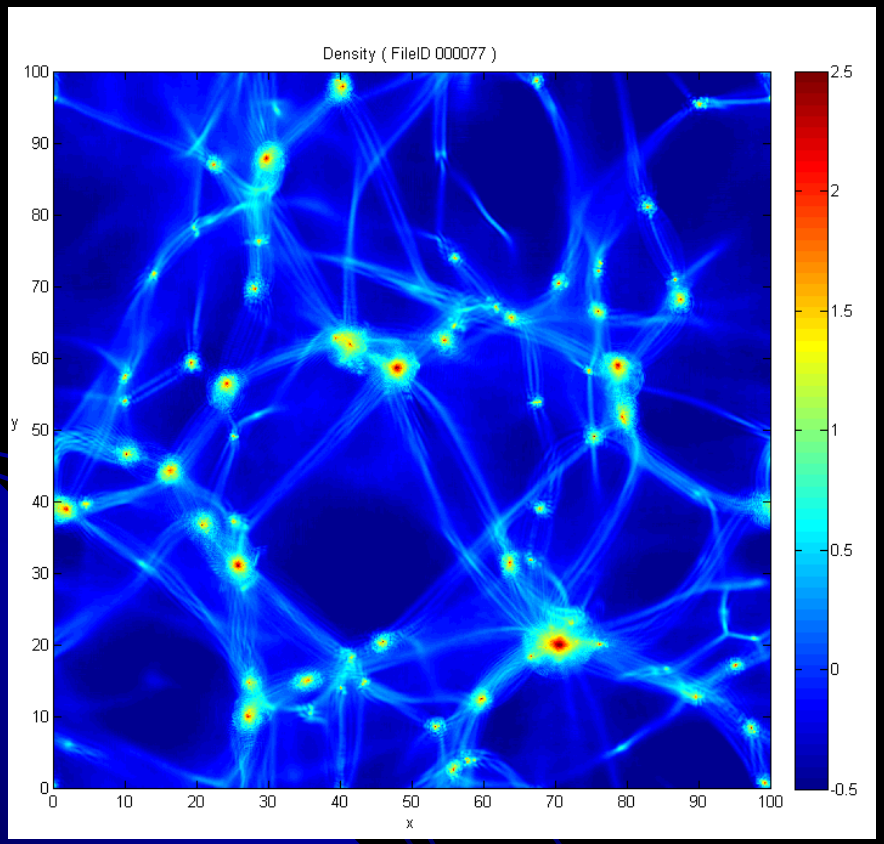
- 2 Mpc box, with 60 pc resolution
- From z=3,200 to z=0
- Substructures are suppressed due to the quantum pressure
- Large-scale structure is similar to CDM

ψDM vs. CDM

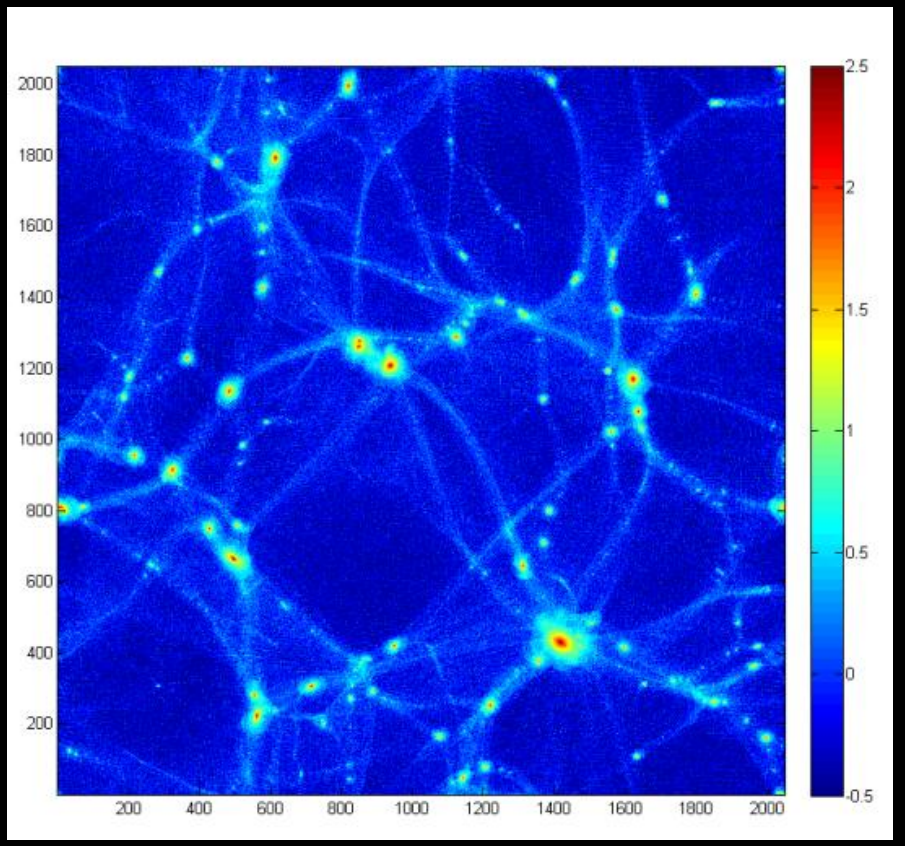

\psiDM projX
CDM projX
\psiDM projZ
CDM projZ
0
200
400
600
800
1000
0.01
0.1
1
10
100
1000
10000

ψDM on Large Scale

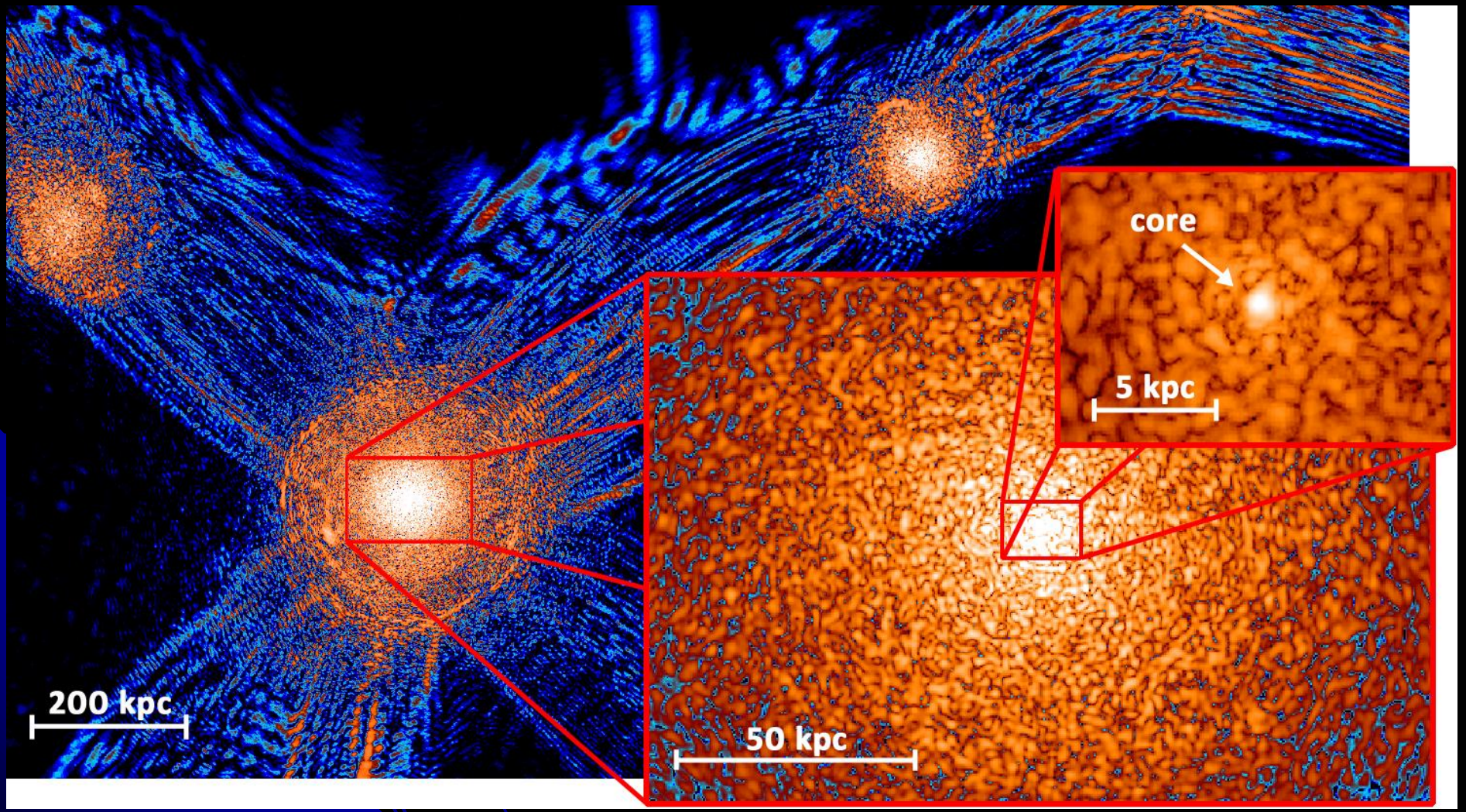
- ψDM (GAMER)



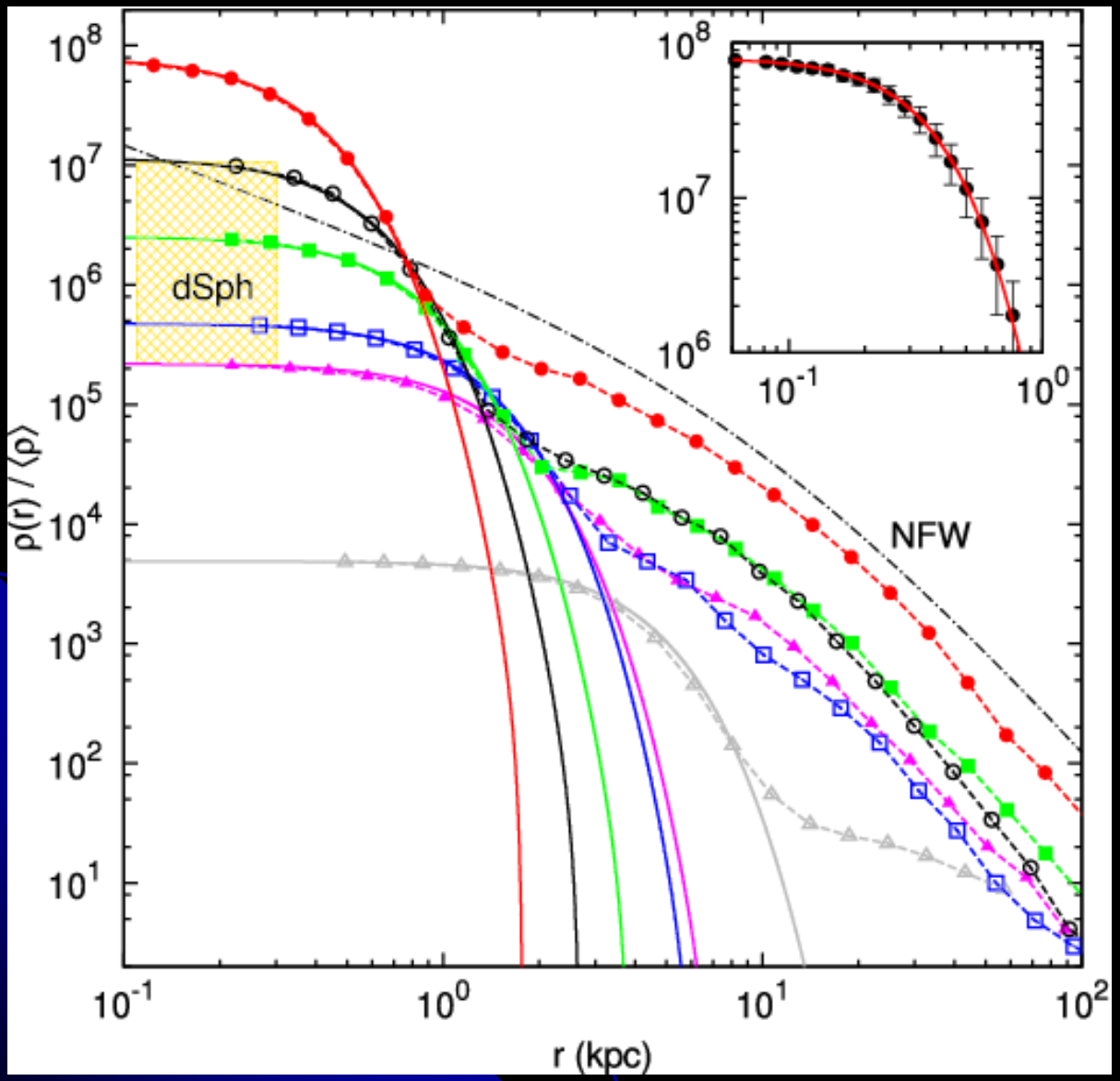
- CDM (GADGET)



ψDM on Small Scale



Halo Density Profile



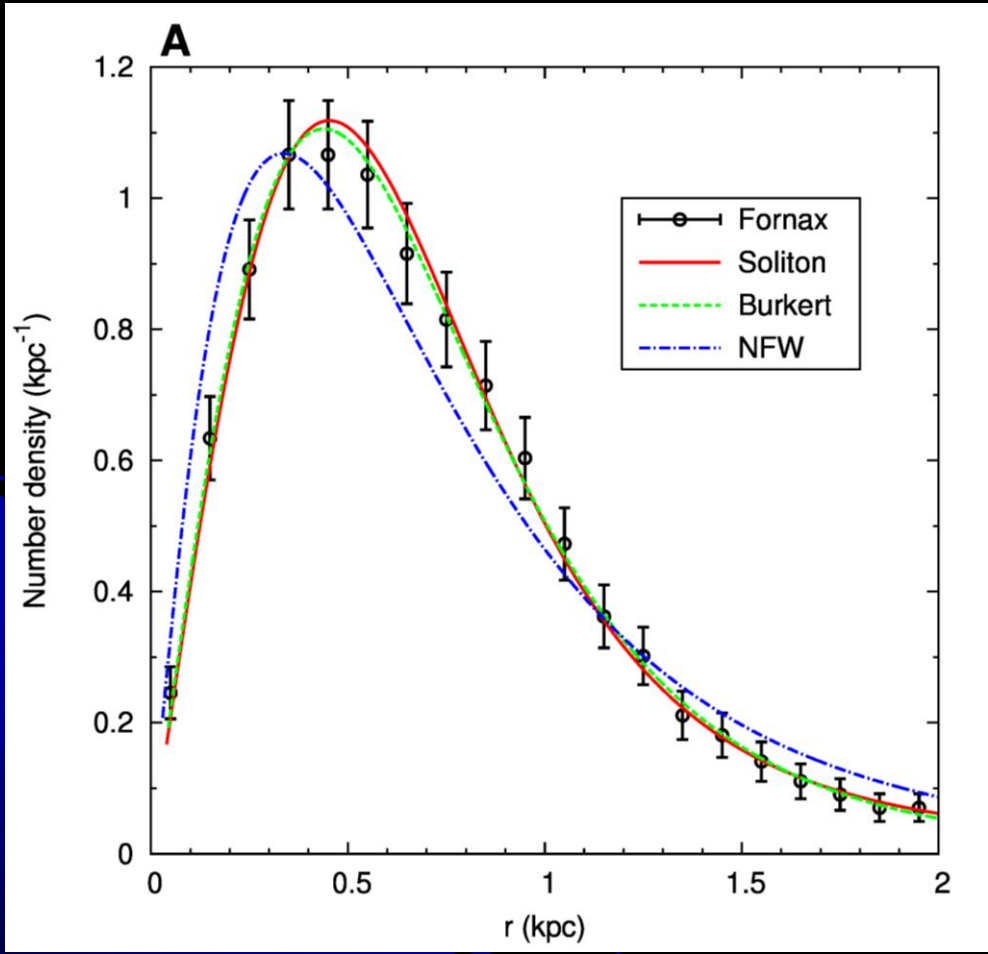
Cored instead of cuspy profiles

Lower limit in M_{300}
→ consistent with Milky Way dwarf spheroidal galaxies (dSph)

Core profiles satisfy the **soliton** solution

Particle Mass Determination:

Stellar Phase-space Distribution



Jeans Eq.:

$$\frac{d(\rho_\star \sigma_r^2)}{dr} = -\rho_\star \frac{d\Phi}{dr} - \frac{2\beta\rho_\star \sigma_r^2}{r}$$

$\rho_\star$: *star number density*

$\sigma_r^\star$: *radial velocity dispersion*

Φ : *gravitational potential*

β : *velocity anisotropy*

Assuming constant and isotropic velocity dispersion

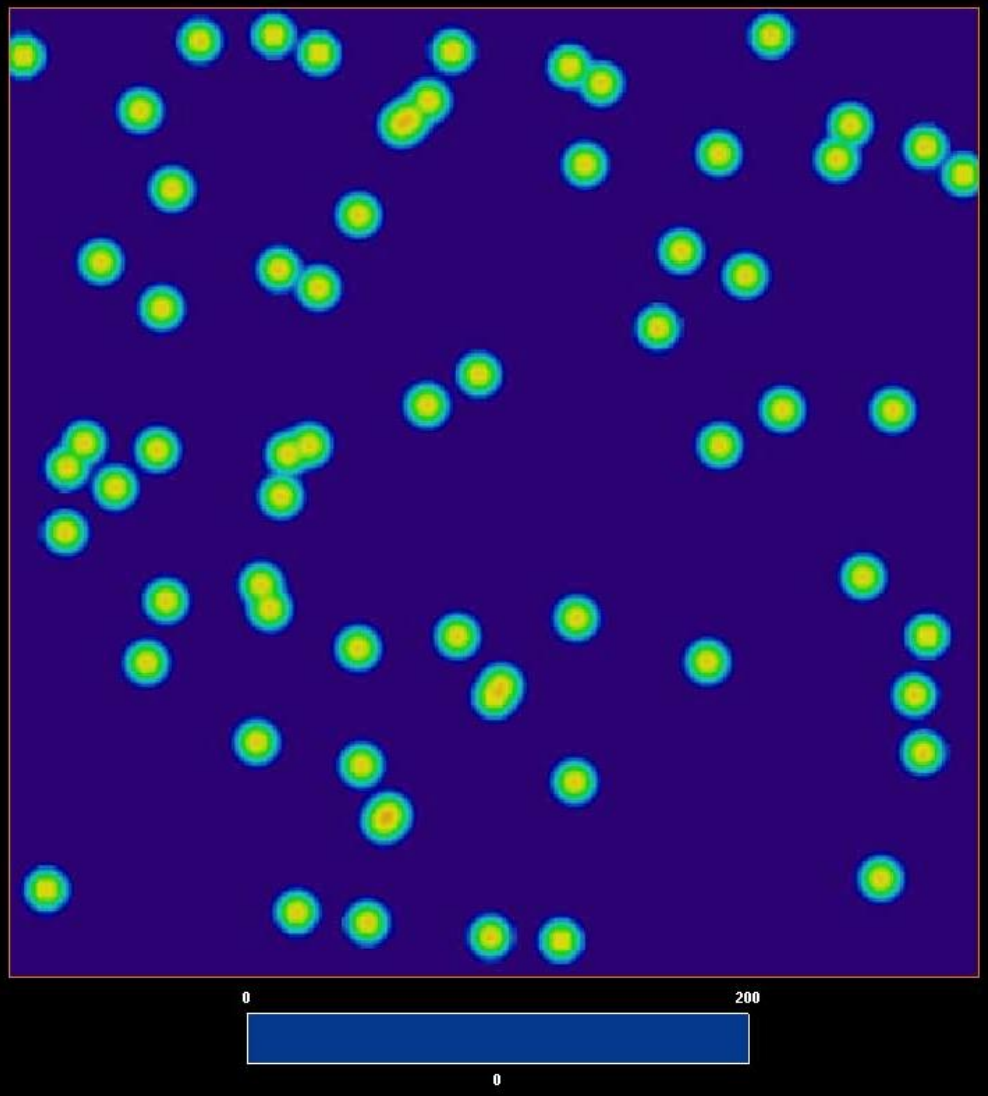
$$\rho_\star = \rho_0 exp[-\Phi(r)/\sigma_r^2]$$

Find the best-fit m_B & r_c

→ $m_B \sim 8.1 \times 10^{-23}$ eV

$r_c \sim 0.92$ kpc

Soliton Merger Experiments



- Multiple solitons are initially put randomly with zero velocity
- **Core-halo duet** is verified to be a generic configuration after relaxation
- Core profile always satisfies the **soliton solution**

Summary

- Gravitational force can be calculated efficiently with different approximate gravity solver
  - PM, Tree, P^3M, Tree-PM, …
- Finite difference schemes need to be designed carefully
  - Courant condition
  - Shock capturing (Riemann solver)
- AMR: increase resolution only around the interesting regions
- GPU: boost performance by ~ 10 – 100
- ψDM: promising dark matter candidate
  - Solve CDM problems at the small scales