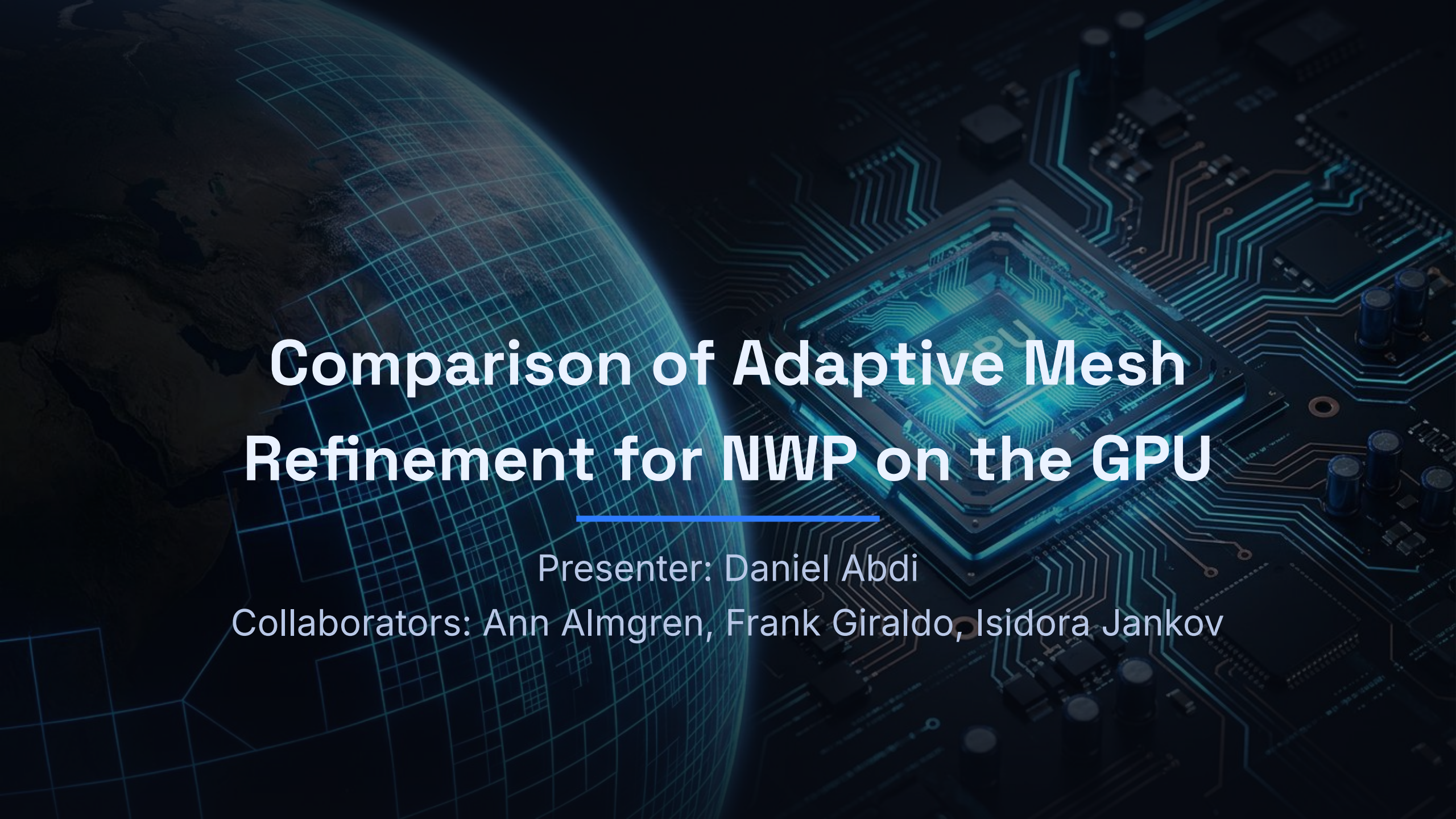


# Comparison of Adaptive Mesh Refinement for NWP on the GPU

Presenter: Daniel Abdi

Collaborators: Ann Almgren, Frank Giraldo, Isidora Jankov

# Why AMR + GPUs (and why now)

- **Resolution Crisis:** Uniform global grids at 1km are too expensive.
- **Localized Features:** Fronts, TCs, and convection only occupy ~1% of domain.
- **GPU Opportunity:** Massive compute, but codes must be regular/parallel.
- **The Question:** Can AMR map effectively to many-core architectures?

Exponential Compute Demand

GPU-Driven Throughput

# AMR in One Slide: Two Approaches

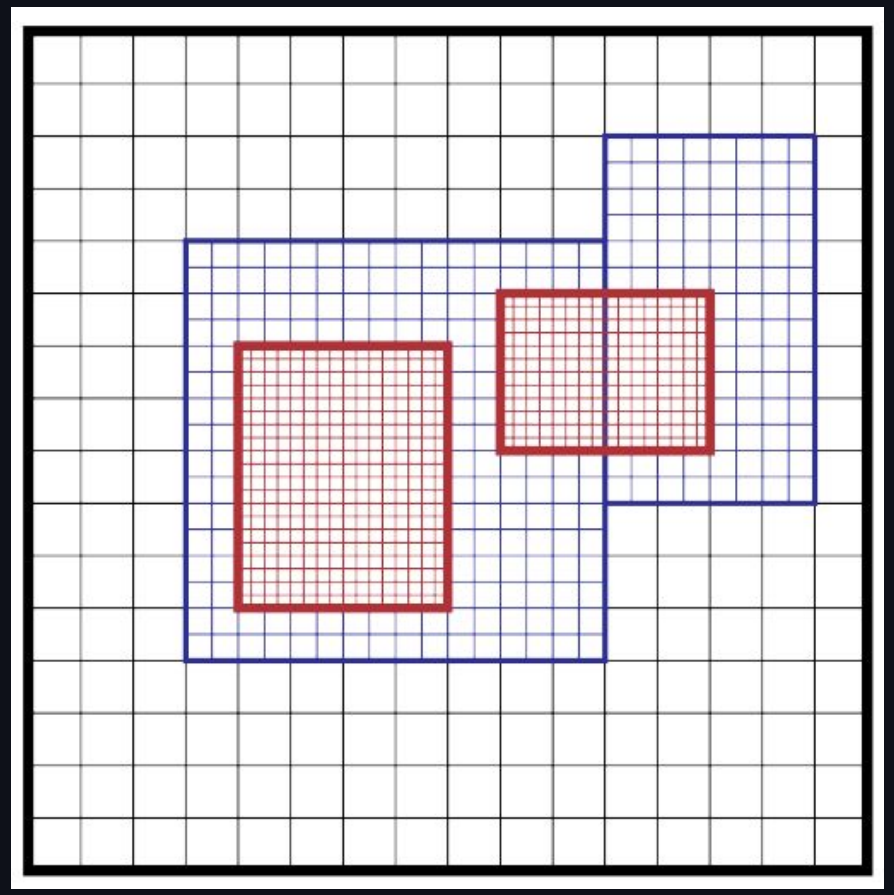
## Level-based AMR

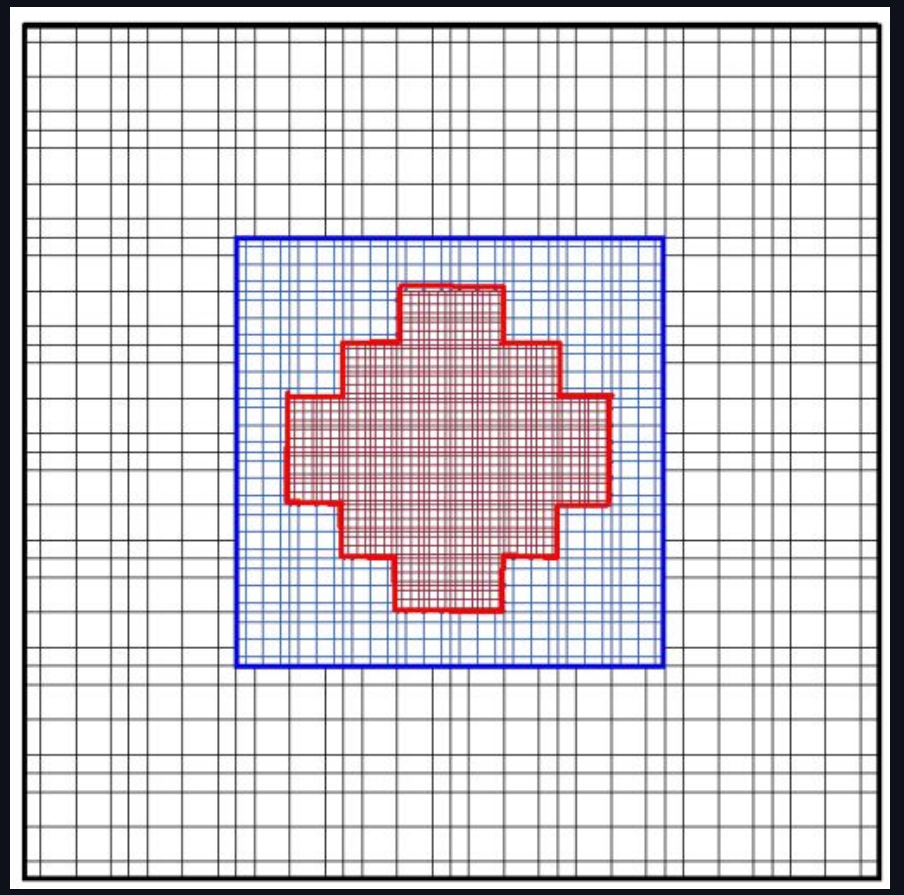
- Hierarchy of rectangular patches
- Maps well to existing NWP nesting
- Easier to block-structure for GPUs
- Standard: AMReX framework

## Tree-based AMR

- Element-wise refinement (Quad/Oct)
- Flexible for complex grids
- Recursive management complexity
- Standard: p4est

# AMR in One Slide: Two Approaches





# Implementations and Benchmarks

### ERF (Level-based)

Built on AMReX. Finite-difference on Arakawa C-grid. Standard for patch-based high-res regional modeling.

### NebulaSEM (Tree-based)

Uses dGSEM discretization. Spherical quad refinement. Flexible for global wave and transport problems.

**Benchmarks:** Rising Thermal Bubble (RRTB) • Transport on Sphere • Acoustic Waves

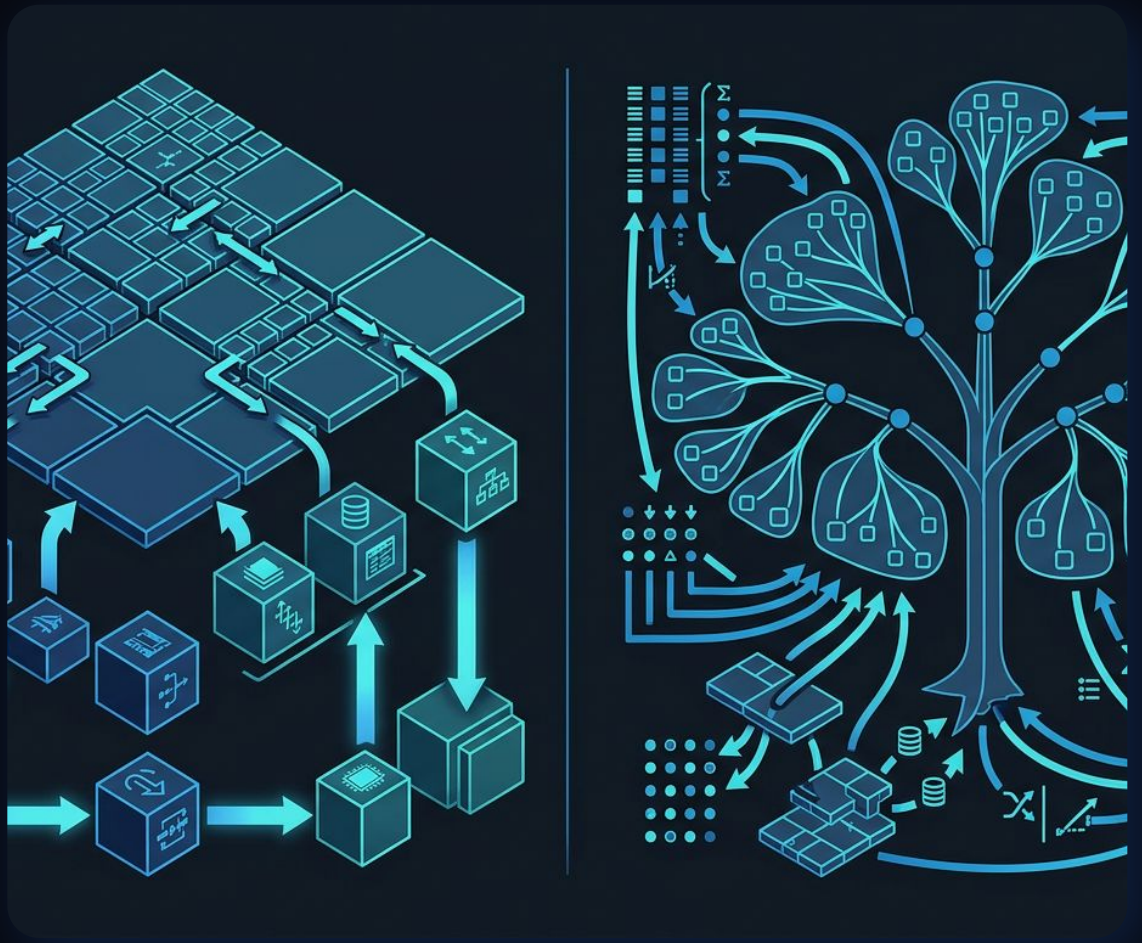
# Software Architecture: GPU Memory Handling

## ERF + AMReX

- Contiguous block-structured arrays (MultiFabs).
- Predictable memory layout for GPU coalescing.
- Explicit device field residency.

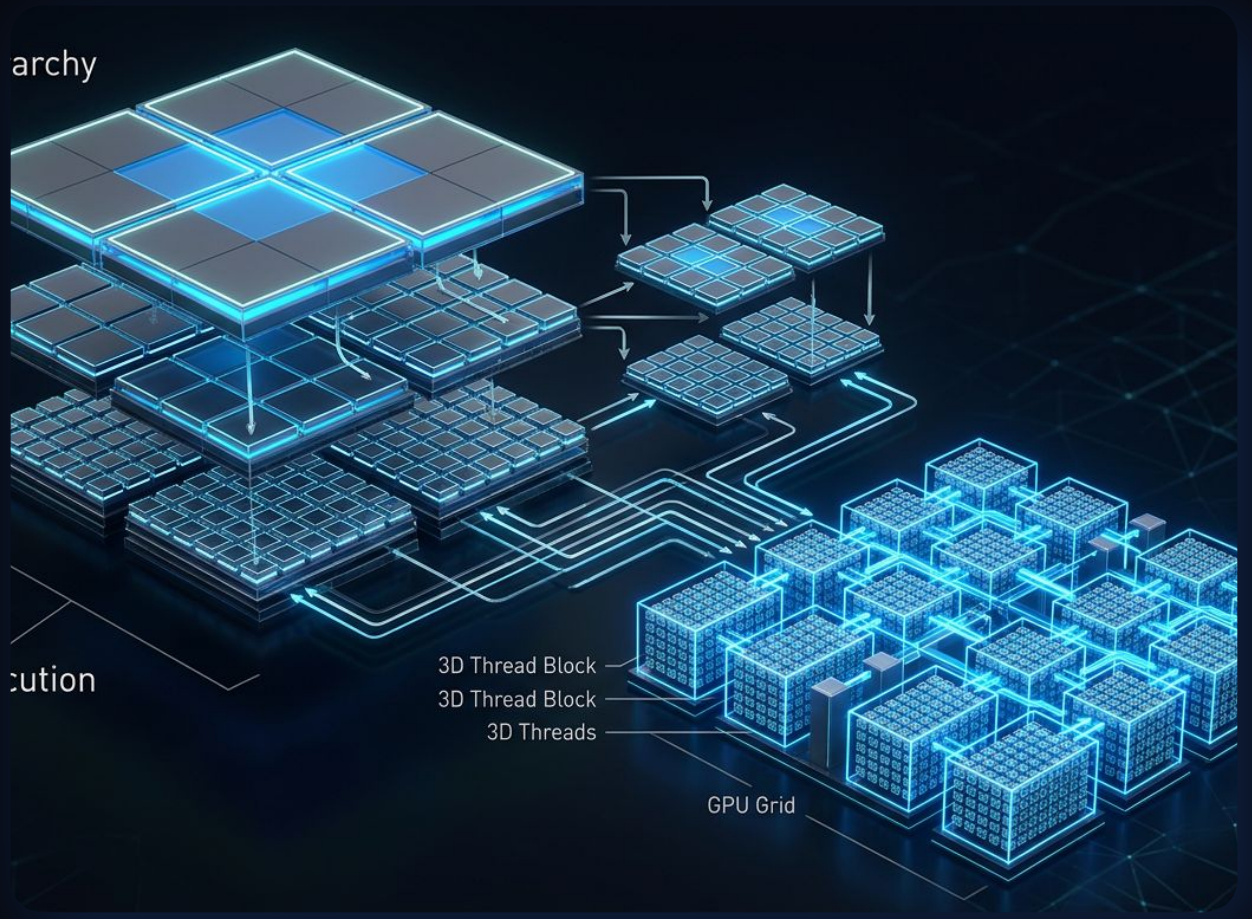
## NebulaSEM (Tree-based)

- Element-tree indirection for refinement tracking.
- Irregular memory access via gathers/scatters.
- Higher complexity for GPU memory management.



# GPU Porting Pain Points for AMR

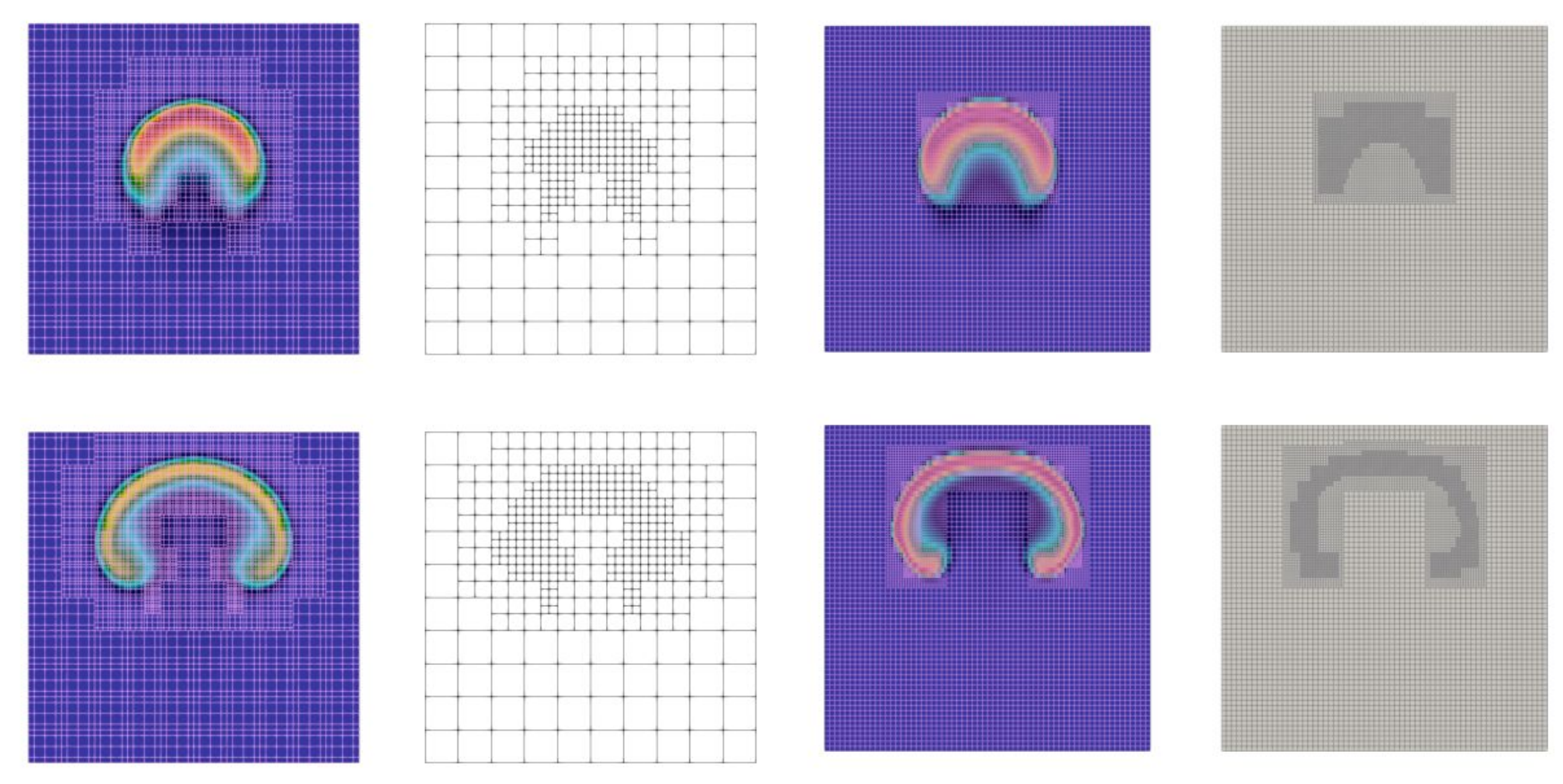
- **Irregularity:** Multi-level patches break GPU's preferred data regularity.
- **Bookkeeping:** Patch intersection and neighbor finding on CPU creates latency.
- **Communication:** Refluxing and coarse-fine transfers add sync overhead.
- **Bottom line:** Mesh management cost can dwarf compute kernels.



# Sustainable GPU Portability (AMReX)

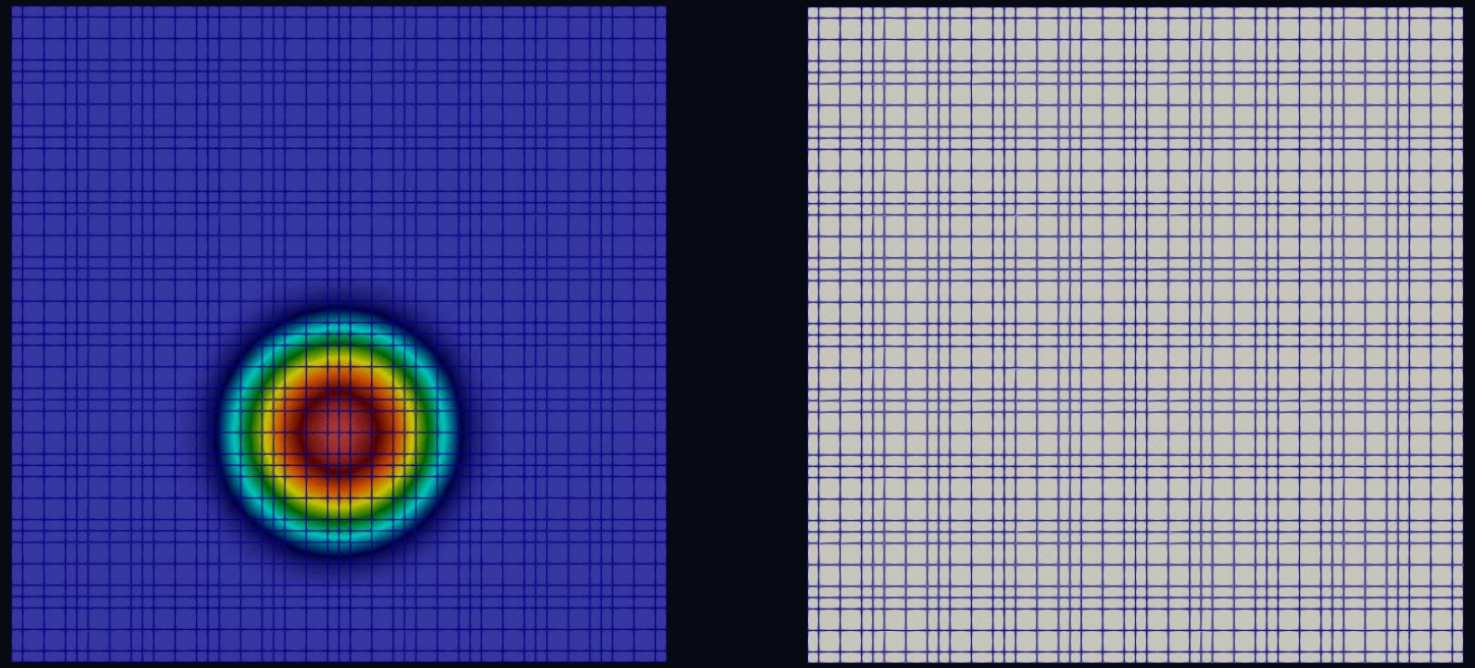
- **Hierarchical Parallelism:** MPI + (CUDA / HIP / SYCL / OpenMP).
- **Abstracted Backends:** Portable dynamics kernels across devices.
- **Independent Load Balance:** SFC/Knapsack at each level.
- **Design Goal:** Hide implementation details; focus on physics.

# Validation: Tracking Dynamic Features



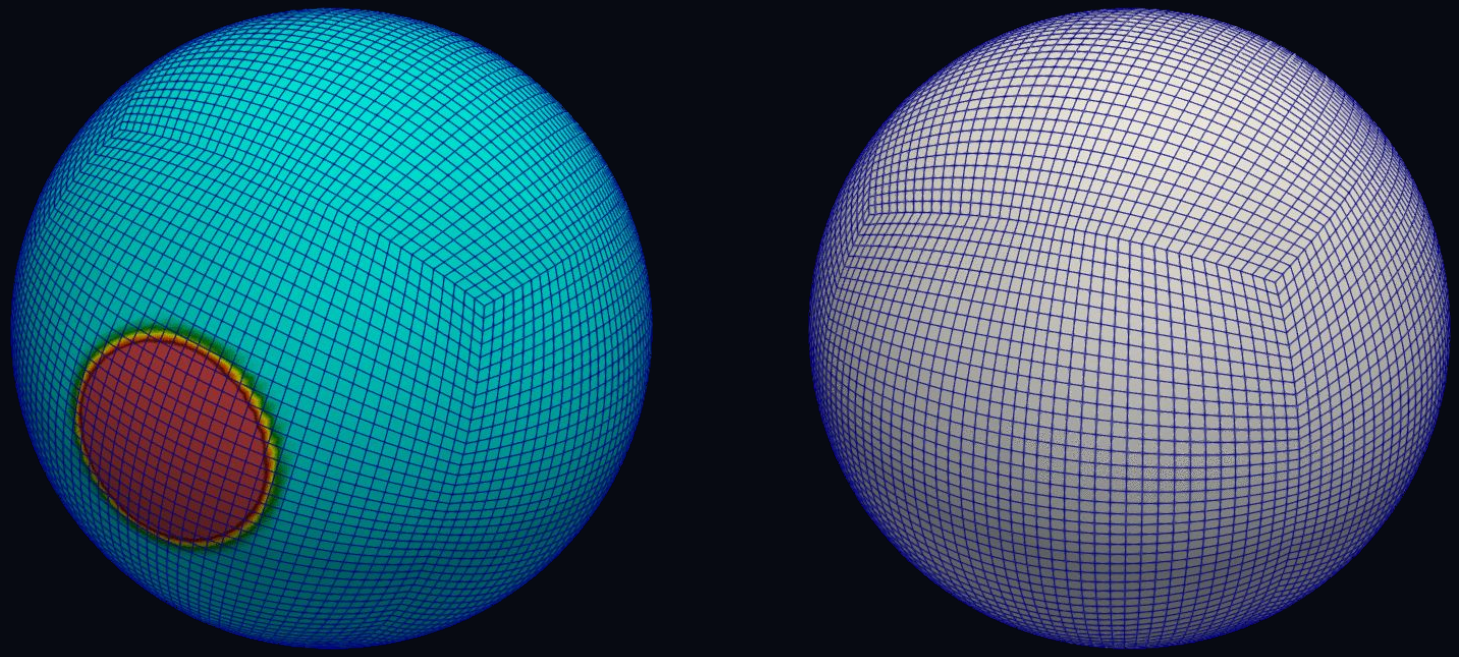
**Rising Thermal Bubble (RRTB):** Comparison of potential temperature and mesh levels for tree-based (left) and level-based (right) AMR tracking the buoyant structure.

# Validation: Tracking Dynamic Features



**Rising Thermal Bubble (RRTB):** Comparison of potential temperature and mesh levels for tree-based (left) and level-based (right) AMR tracking the buoyant structure.

# Validation: Tracking Dynamic Features



**Rising Thermal Bubble (RRTB):** Comparison of potential temperature and mesh levels for tree-based (left) and level-based (right) AMR tracking the buoyant structure.

# The AMR Payoff: Time-to-Solution


**Speedup vs Uniform Grid**

2D RRTB (Tree-based): **32 min (AMR)** vs **182 min (Fine Grid)**

# GPU Weak Scaling (ERF on Hera)

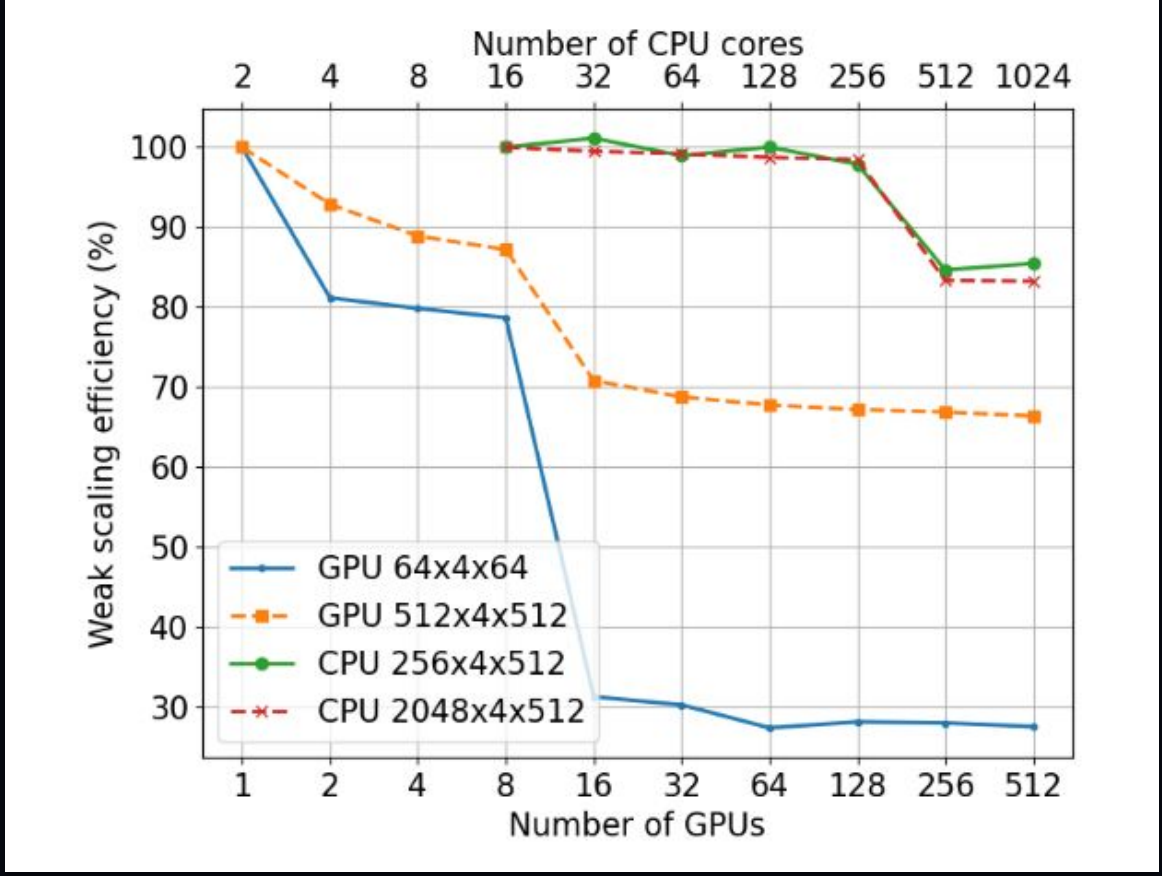
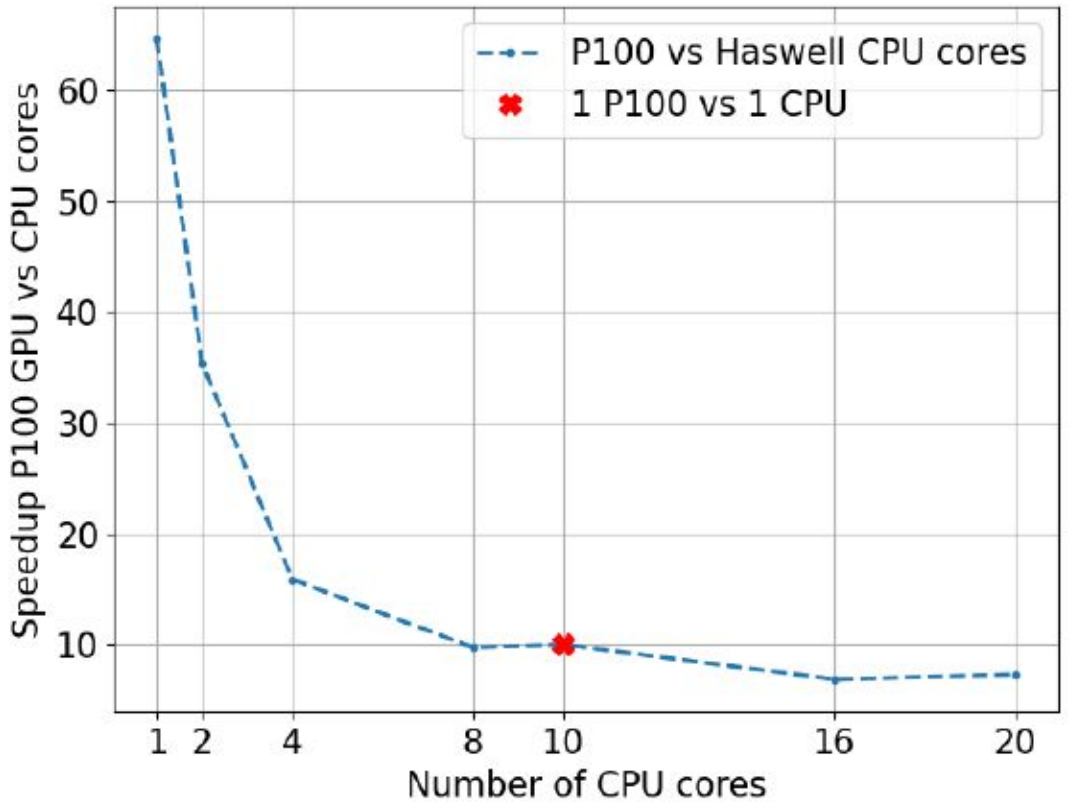



*Fig: Weak scaling efficiency for varying grid sizes and configurations.*

- **Scalability:** Excellent beyond 2 nodes (16 GPUs).
- **Penalties:** Inter-node and intra-node latencies at small scales.
- **Efficiency:** Up to 93% utilization with larger workloads.
- **Tested:** Up to 64 nodes (512 P100 GPUs).

# GPU vs CPU: Throughput Comparison




- **Massive Gain:** 1 P100 GPU ≈ 10× faster than Haswell CPU.
- **Scaling:** Speedup remains compelling up to full node saturation.
- **Economics:** Justifies the engineering overhead of AMR GPU porting.
- **Legacy:** P100 is older-gen; A100/H100 gains expected to be higher.

# Practical GPU Porting Lessons

### 1. Framework First

Use AMReX or similar. Don't rebuild mesh management from scratch.

### 2. Expose Overhead

Keep mesh/comm costs visible in profiling to avoid sync bottlenecks.

### 3. Workload Density

Ensure sufficient work per GPU. Avoid under-filling via patch sizing.

### 4. GPU-First Transfer

Treat coarse-fine transfers as dynamics kernels, not CPU glue code.

# Conclusions: The Future is Refined

- **Practicality:** AMReX enables GPU-portable NWP with AMR.
- **Performance:** ~10× GPU/CPU speedup and strong weak scaling.
- **Utility:** Up to 6× time-savings vs uniform global grids.

## Questions?