
N-body Simulations
On
GPU Clusters

PPL PARALLEL PROGRAMMING LAB
DEPT OF COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS

Thomas Quinn
Graeme Lufkin
Joachim Stadel
James Wadsley
Fabio Governato

Laxmikant Kale
Filippo Gioachin
Pritish Jetley
Celso Mendes
Amit Sharma
Lukasz Wesolowski
Edgar Solomonik
Orion Lawlor

Outline

- Types of N-Body simulations
  - Small N (SS, GC)
  - Large N
- Need for Exascale
- GPU details
- Single node performance
- Scaling
- Multistepping issues

Cosmology at 130,000 years

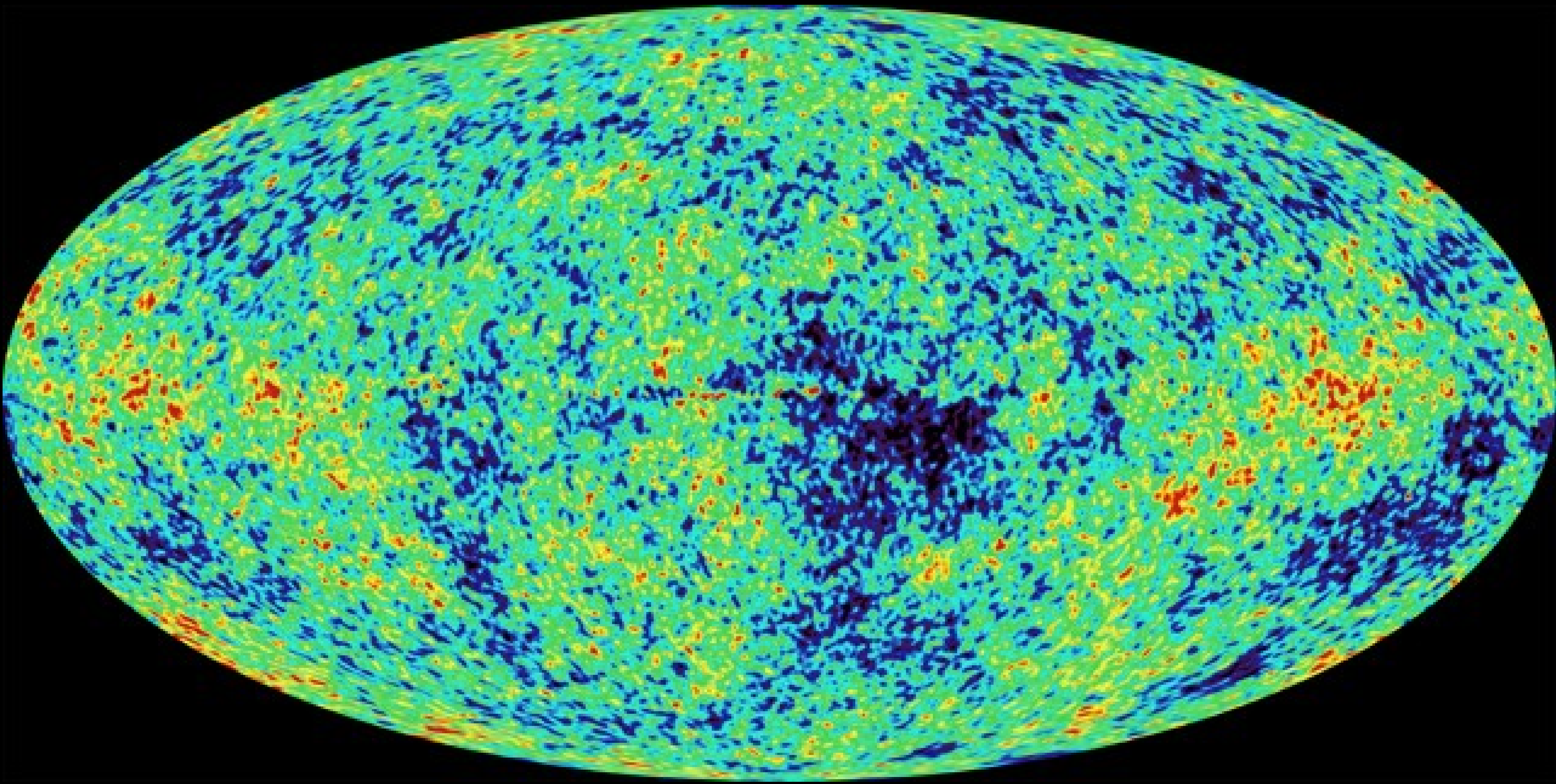
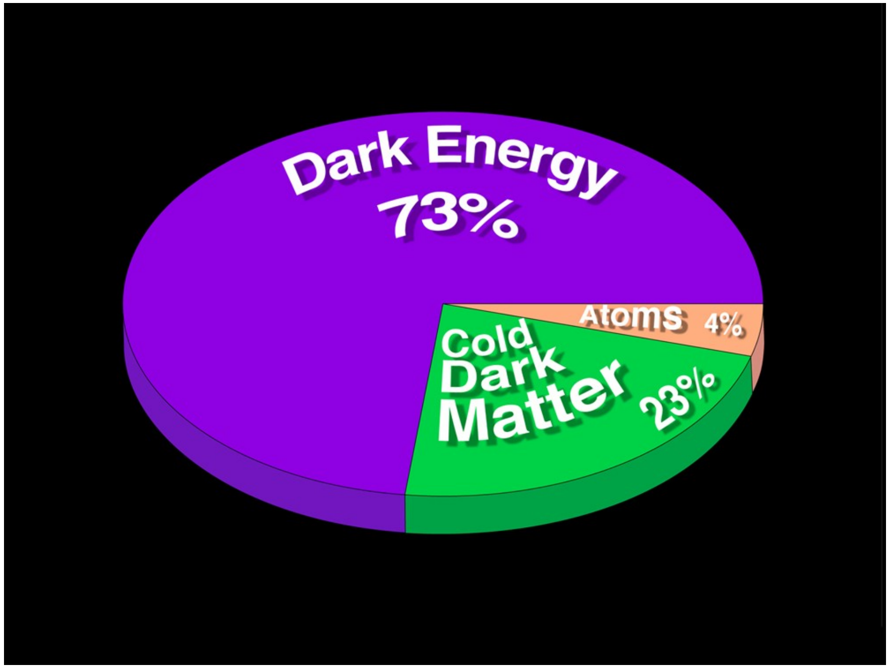


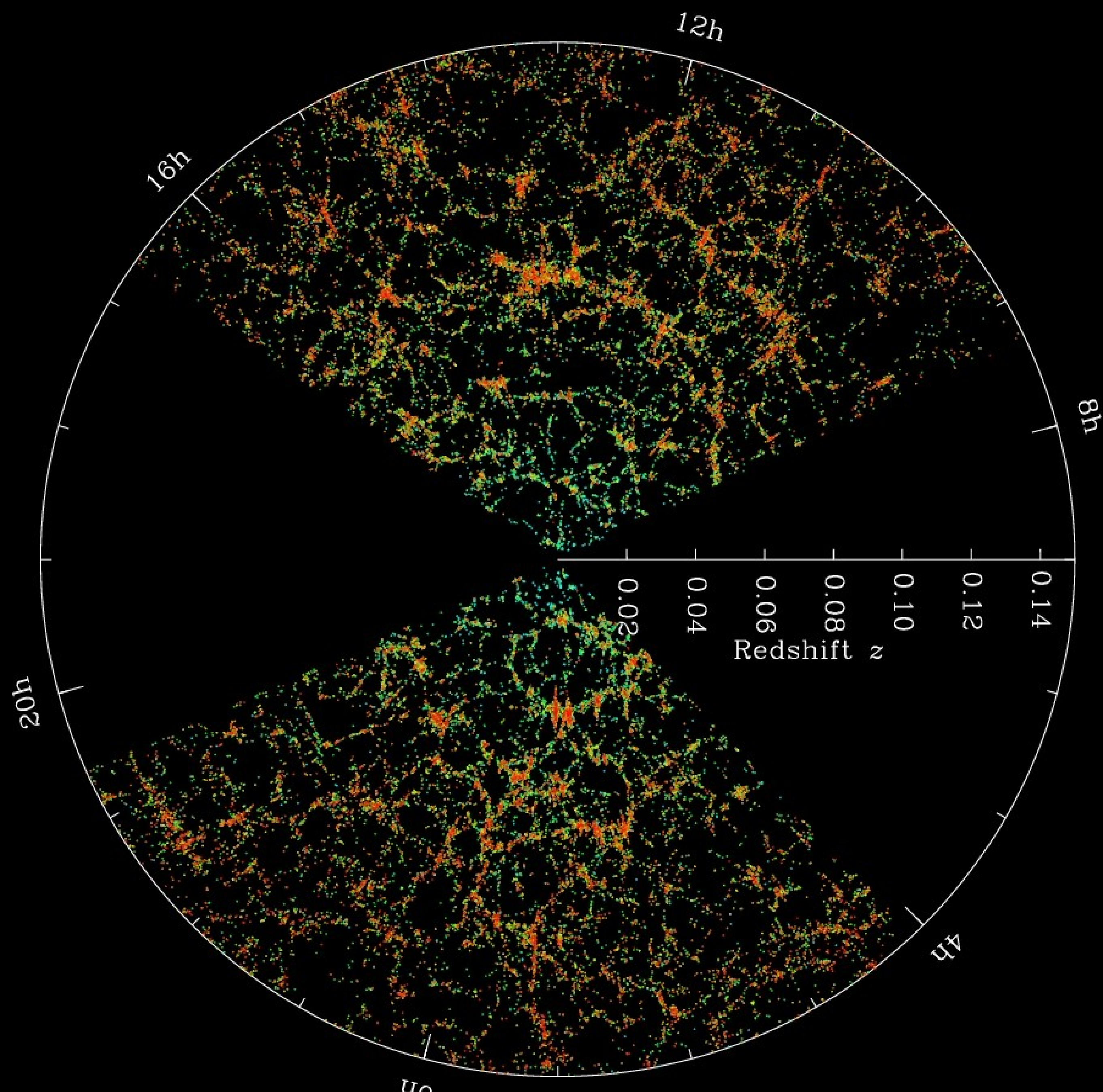
Image courtesy NASA/WMAP

Fundamental Problem: Dark Matter and Energy: What is it?

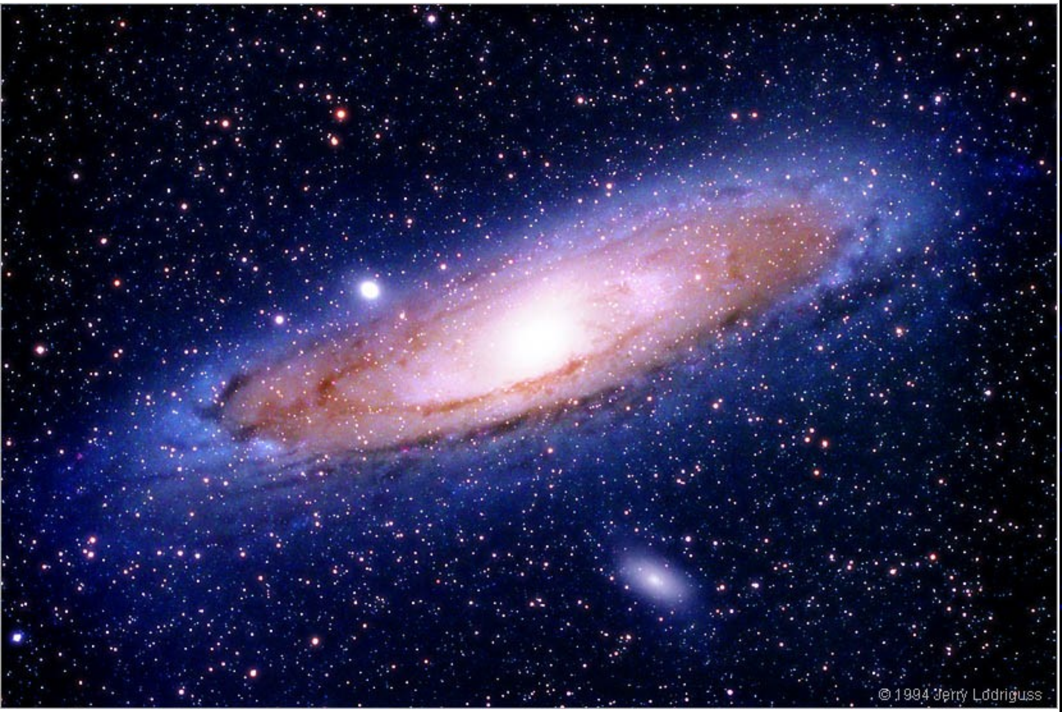
- Not baryons
- Simulations show: not known neutrinos
- Candidates:
  - Sterile Neutrinos
  - Axions
  - Lightest SUSY Particle (LSP)

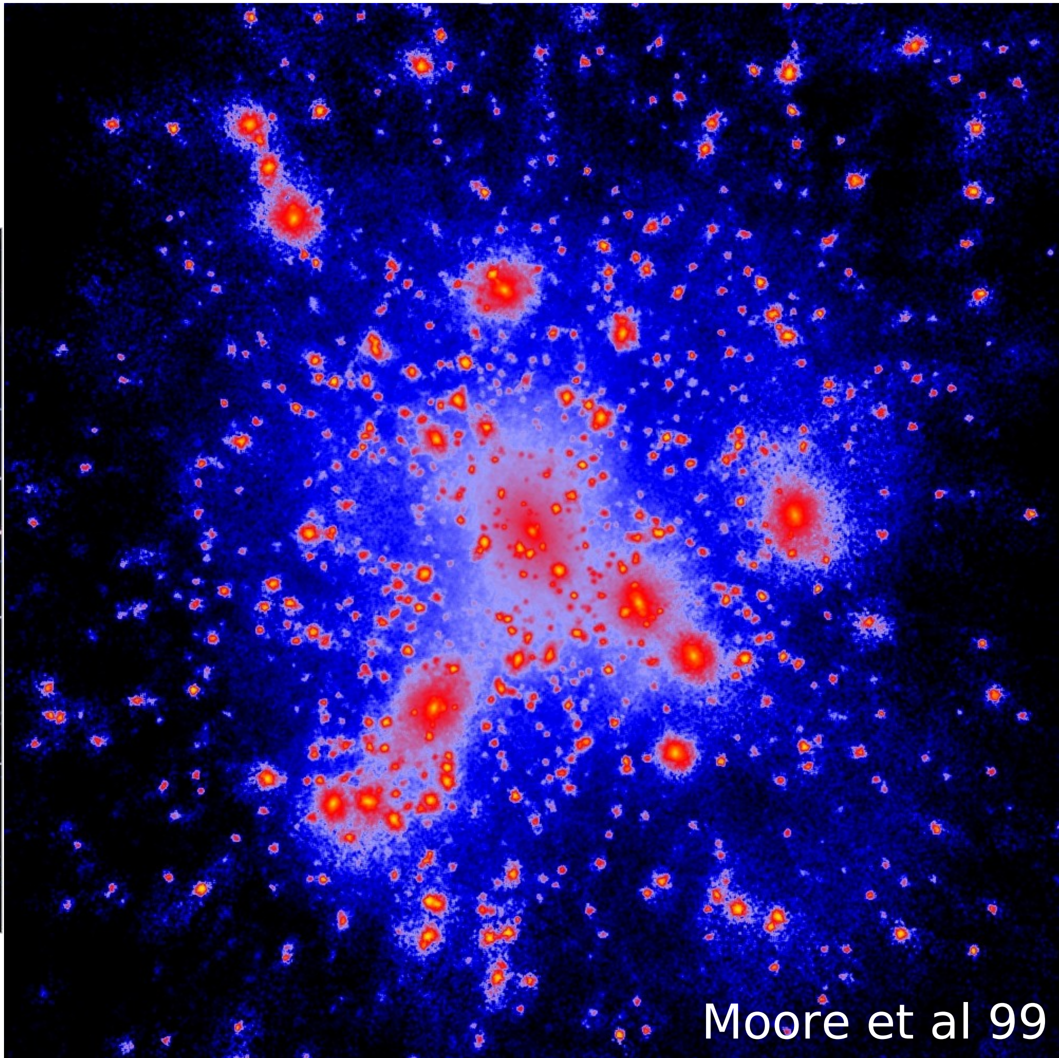


Cosmology at 13.6 Gigayears



Light vs. Matter





Computational Cosmology

- CMB has fluctuations of 1e-5
- Galaxies are overdense by 1e7
- It happens (mostly) through **Gravitational Collapse**
- Making testable predictions from a cosmological hypothesis requires
  - Non-linear, dynamic calculation
  - e.g. **Computer simulation**

What is N?

Are we solving for orbits of particles:

$$\ddot{\mathbf{x}}_i = -\sum_{j \neq i}^{N} \frac{G m_j \mathbf{r}_{ij}}{|r_{ij}|^3}?$$

We should be solving the Collisionless Boltzmann equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \boldsymbol{\nabla} f - \boldsymbol{\nabla} \Phi \cdot \frac{\partial f}{\partial \mathbf{v}} = 0.$$

On the surface this is difficult, but we can use the method of characteristics where we follow the motion of packets of f:

$$\delta f(\mathbf{x}(t), \mathbf{v}(t)).$$

Now the equations of motion for these packets are:

$$\dot{\mathbf{x}} = \mathbf{v},$$

$$\dot{\mathbf{v}} = -\boldsymbol{\nabla} \Phi.$$

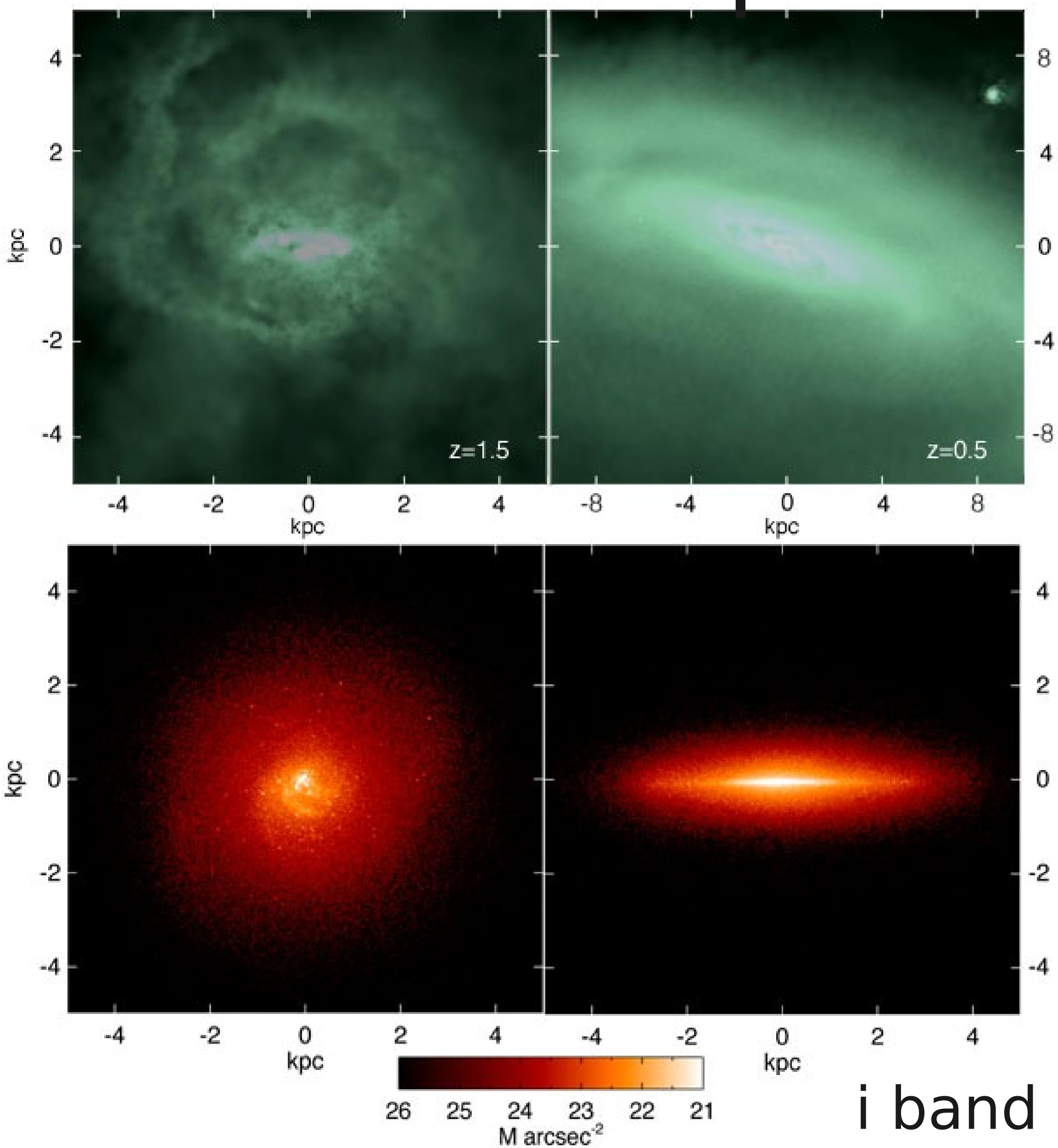
Smooth Particle Hydrodynamics

- Making testable predictions needs Gastrophysics
  - High Mach number
  - Large density contrasts
- Gridless, Lagrangian method
- Galilean invariant
- Monte-Carlo Method for solving Navier-Stokes equation.
- Natural extension of particle method for gravity.

Simulating Galaxy Formation: Current Methodology

- Full cosmological context with high resolution
  - Dynamic range of 1e5 in time and space
  - Treecode/SPH or similar adaptive method is required.
- Physically motivated subgrid effects of star formation and feedback
- Complete simulations to present epoch.
- Analyze with multiple simulated observations

Dwarf galaxy simulated to the present



Reproduces:
* Light profile
* Mass profile
* Star formation
* Angular momentum

i band image

Galactic structure in the local Universe: What's needed

- 1 Million particles/galaxy for proper morphology/heavy element production
- 25 Mpc volume
- 800 M core-hours
- Necessary for:
  - Comparing with Hubble Space Telescope surveys of the local Universe
  - Interpreting HST images of high redshift galaxies

Large Scale Structure: What's needed

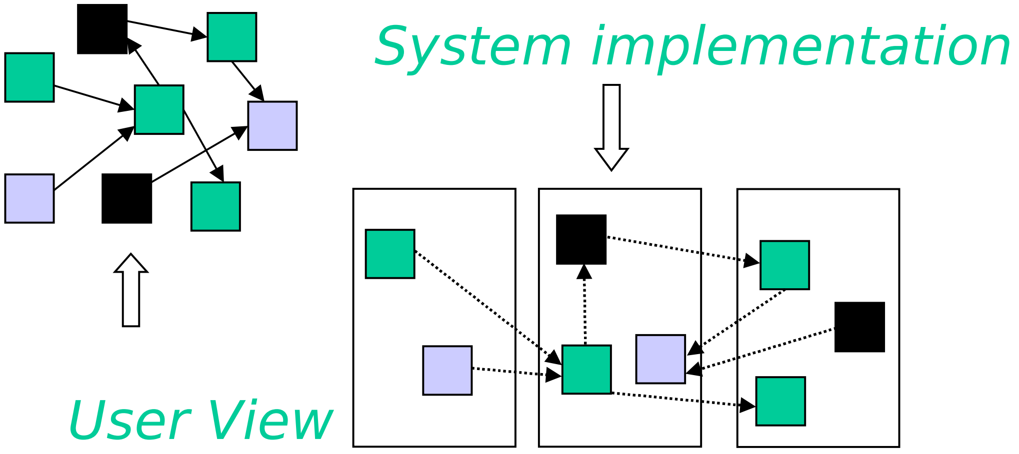
- 700 Megaparsec volume for "fair sample" of the Universe
- 18 trillion core-hours (~ exaflop year)
- Necessary for:
  - Interpreting future surveys (LSST)
  - Relating Cosmic Microwave Background to galaxy surveys

Charm++: Migratable Objects

Programmer: **[Over] decomposition into virtual processors**

Runtime: **Assigns VPs to processors**

Enables *adaptive runtime strategies*



Benefits

- Software engineering
  - Number of virtual processors can be independently controlled
  - Separate VPs for different modules
- Message driven execution
  - Adaptive overlap of communication
- Dynamic mapping
  - Heterogeneous clusters
    - Vacate, adjust to speed, share
  - Automatic checkpointing
  - Change set of processors used
  - Automatic dynamic load balancing
  - Communication optimization

Charm++ at scale

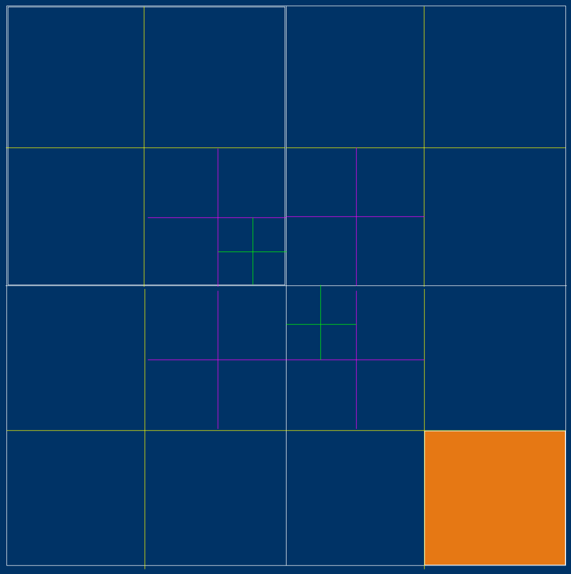
- Composability, object oriented
- Load balancing framework
  - Topology aware
- Available development tools:
  - Profiling at scale
  - Debugging at scale
  - Visualization at scale (http://hpcc.astro.washington.edu/tools/salsa)
  - Machine simulation

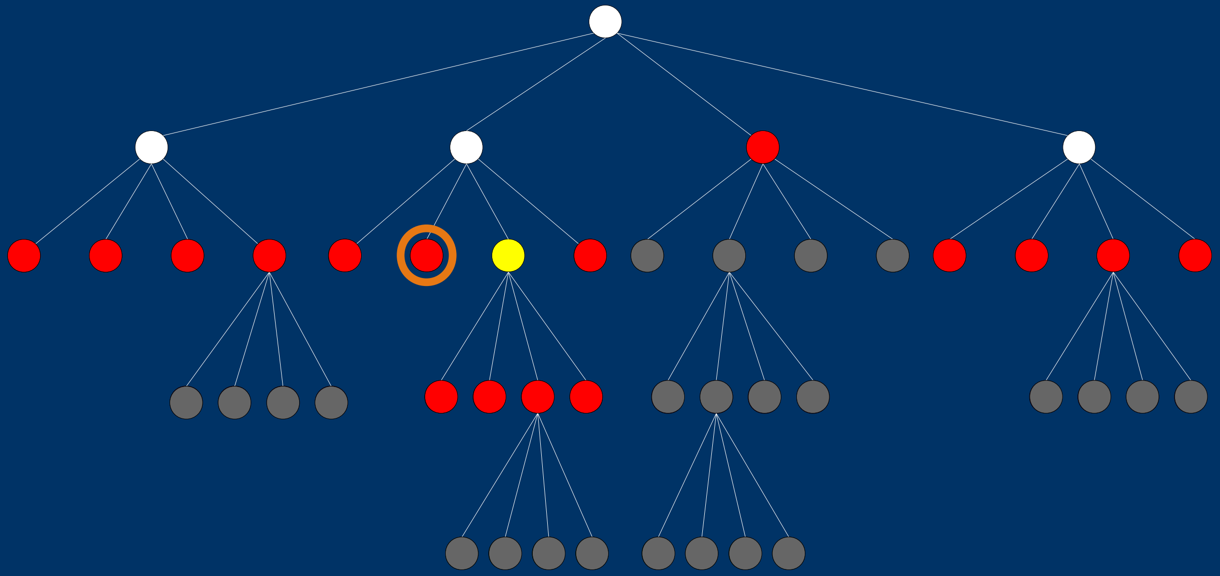
ChaNGa (CHArm N-body GrAvity) Features

- Tree-based gravity solver
- High order multipole expansion
- Periodic boundaries (if needed)
- Individual multiple timesteps
- Dynamic load balancing with choice of strategies
- Checkpointing (via migration to disk)
- Visualization

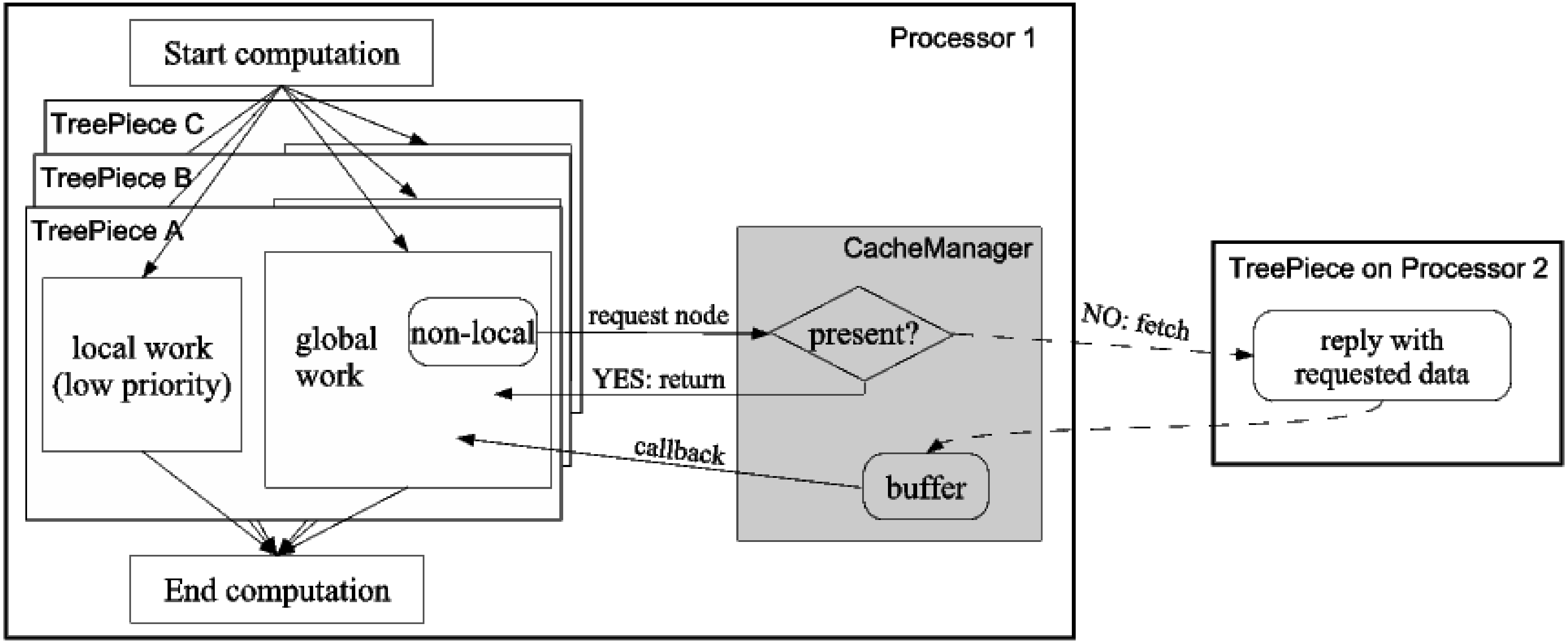
Basic Gravity algorithm ...

- Newtonian gravity interaction
  - Each particle is influenced by all others: O(n^2) algorithm
- Barnes-Hut approximation: O($n\log n$)
  - Influence from distant particles combined into center of mass

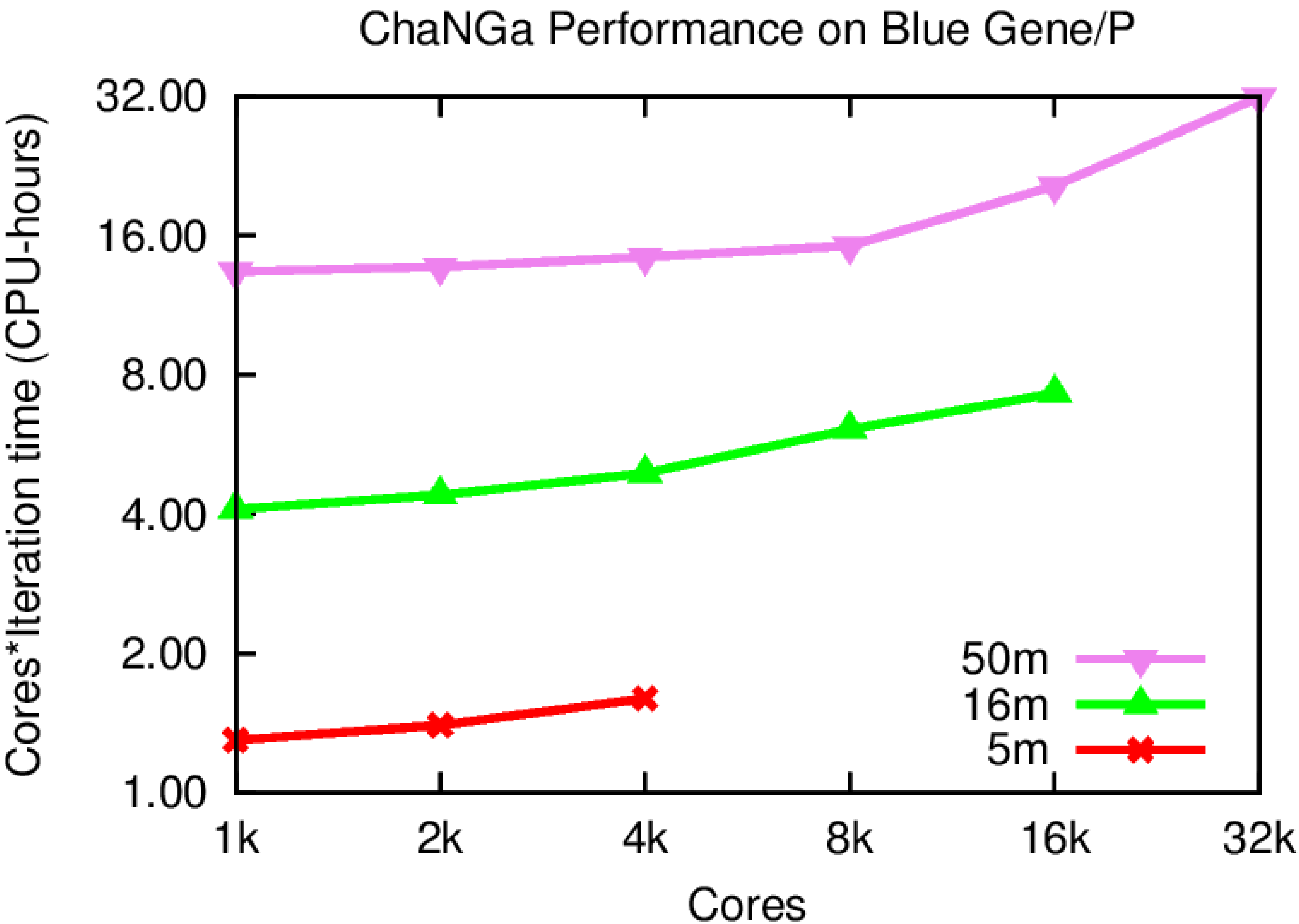




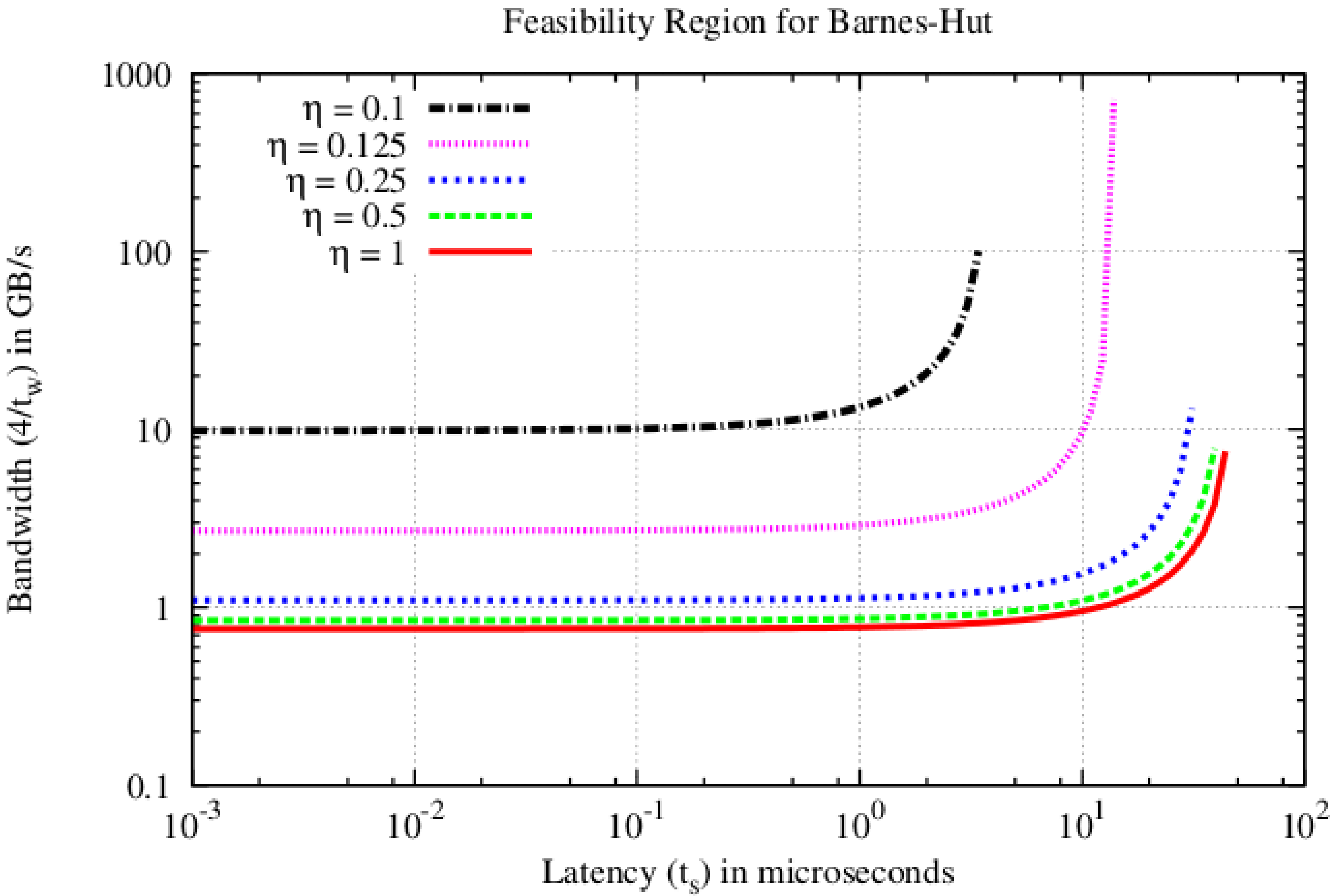
Overall Algorithm



Strong Scaling on BG/P



6.8e12 particles @ 1 Exaflop



200Mpc^3 volume at 1e4 Msun

Cosmology at Exascale

- The Universe is big
  - Build a computer and a cosmologist will fill it.
  - With compelling problems to solve
- Scaling to Exaflops is conceivable
  - Despite use of irregular algorithms/data structures
  - But with significant investment in newer languages/libraries

General Purpose GPUs

- Graphics chips adapted for general purpose programming
- Impressive floating point performance
  - 4.6 Tflops single precision (AMD Radeon HD 5970)
  - Cmp. 100 Gflop for 3 GHz quad-core quad-issue CPU
- Good for large scale data parallelism
- Consumer driven technology

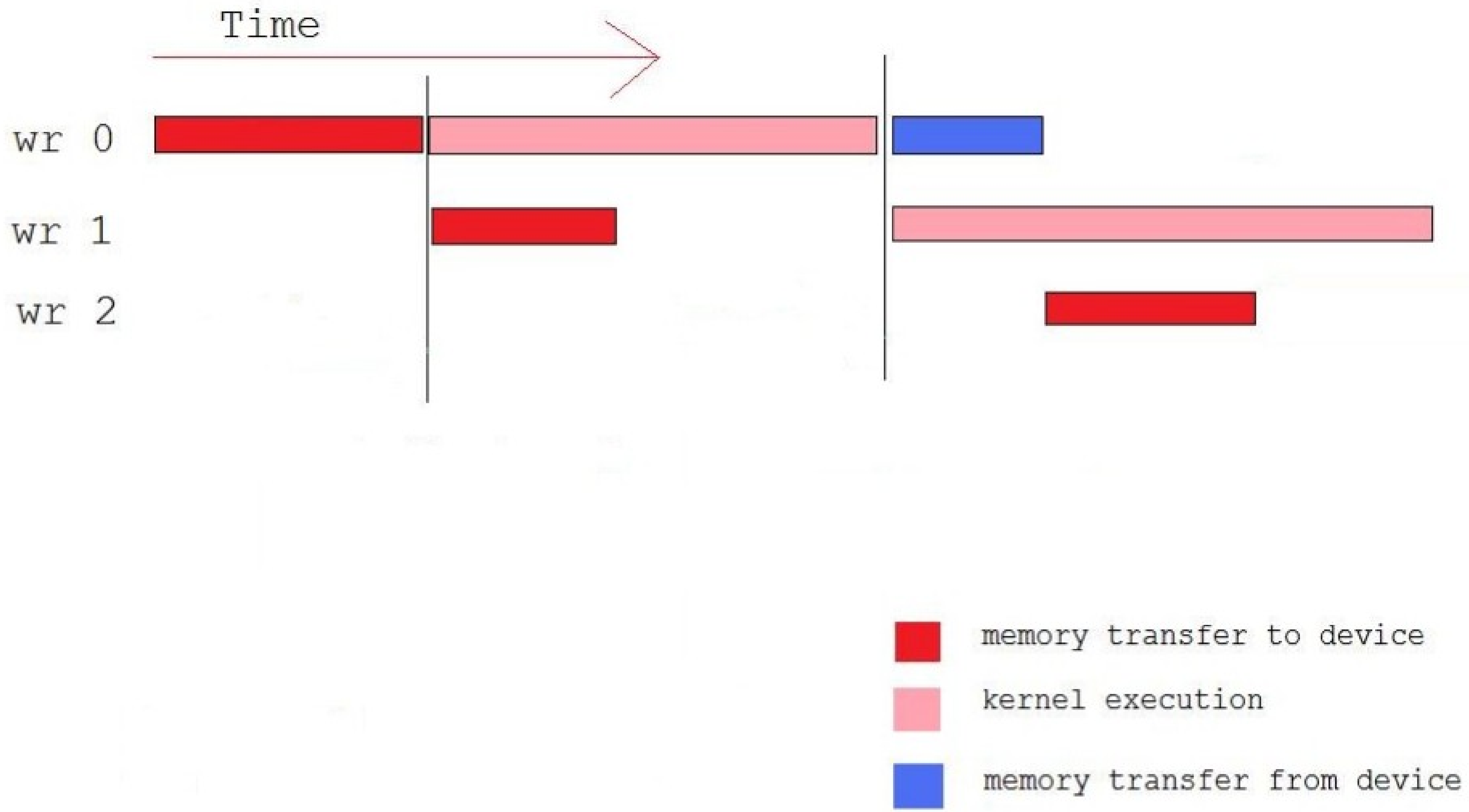
GPU Stream Management

- Common stream usage
  - CPU -> GPU data transfer
  - kernel_call
  - GPU -> CPU data transfer
  - Poll for completion
- Third operation blocks DMA engine until kernel is finished
- Avoid by delaying GPU -> CPU transfer until kernel is finished
  - Requires additional polling call

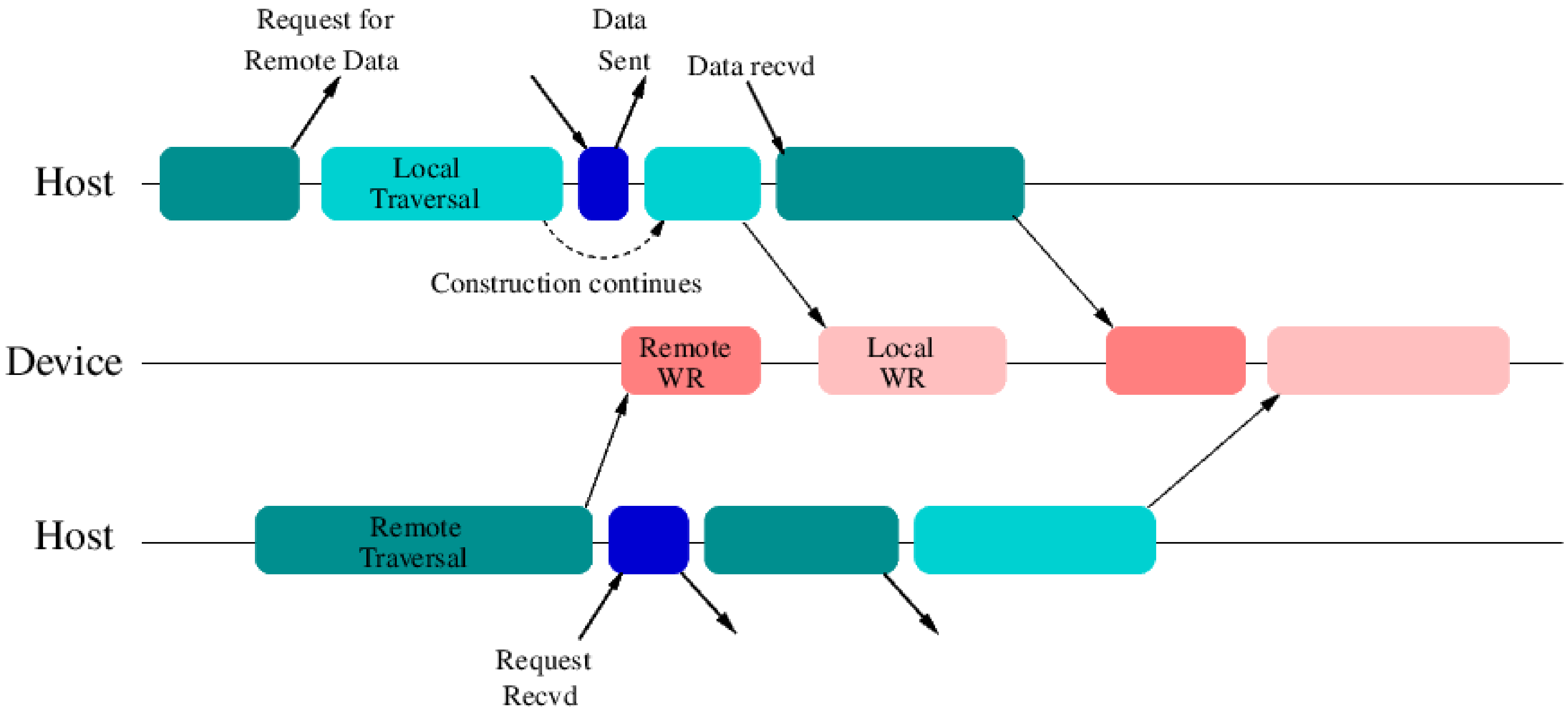
GPU Manager

- User submits “work requests” with GPU kernel, associated buffers and callback
- System transfers memory between CPU and GPU, executes kernel, and returns via a callback
- GPU operations performed asynchronously
- Pipelined execution
- Consistent with Charm++ model
- Charm++ tools (profiler) available

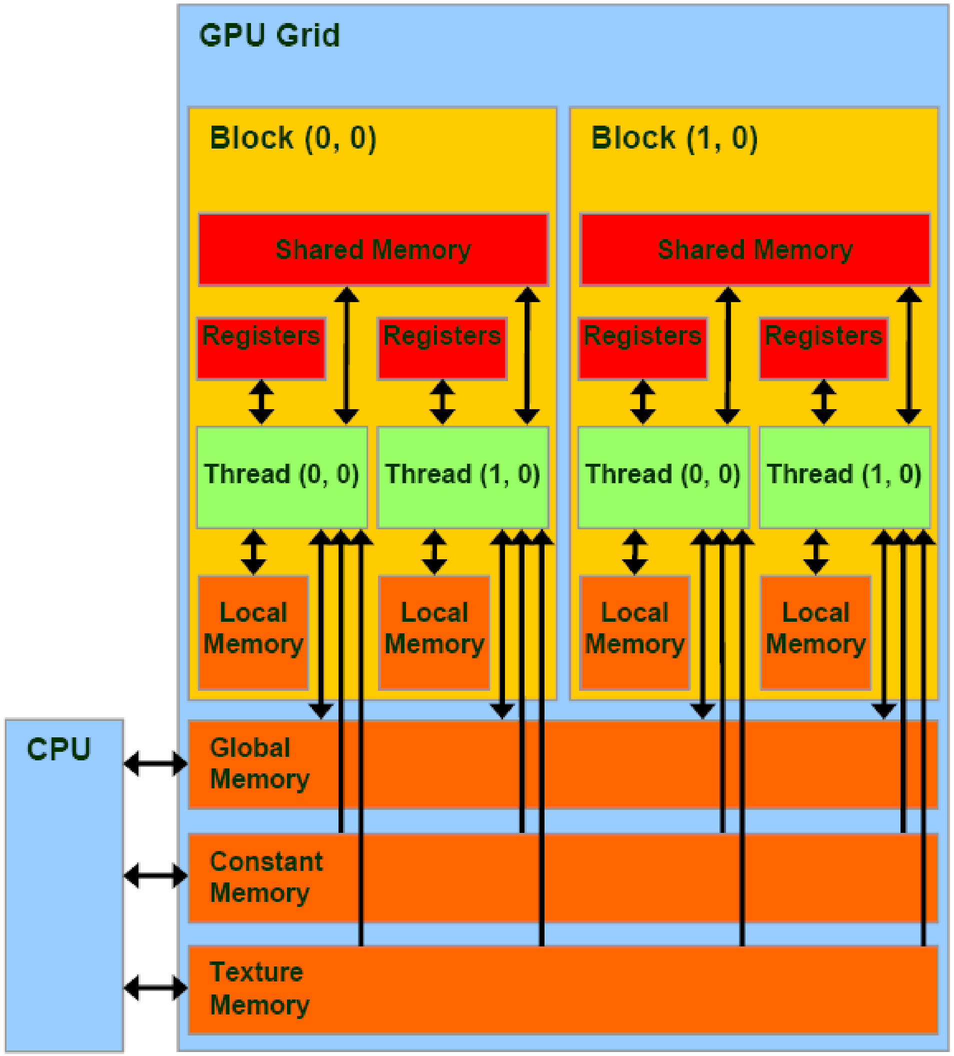
Execution of Work Requests



Overlapping CPU and GPU Work

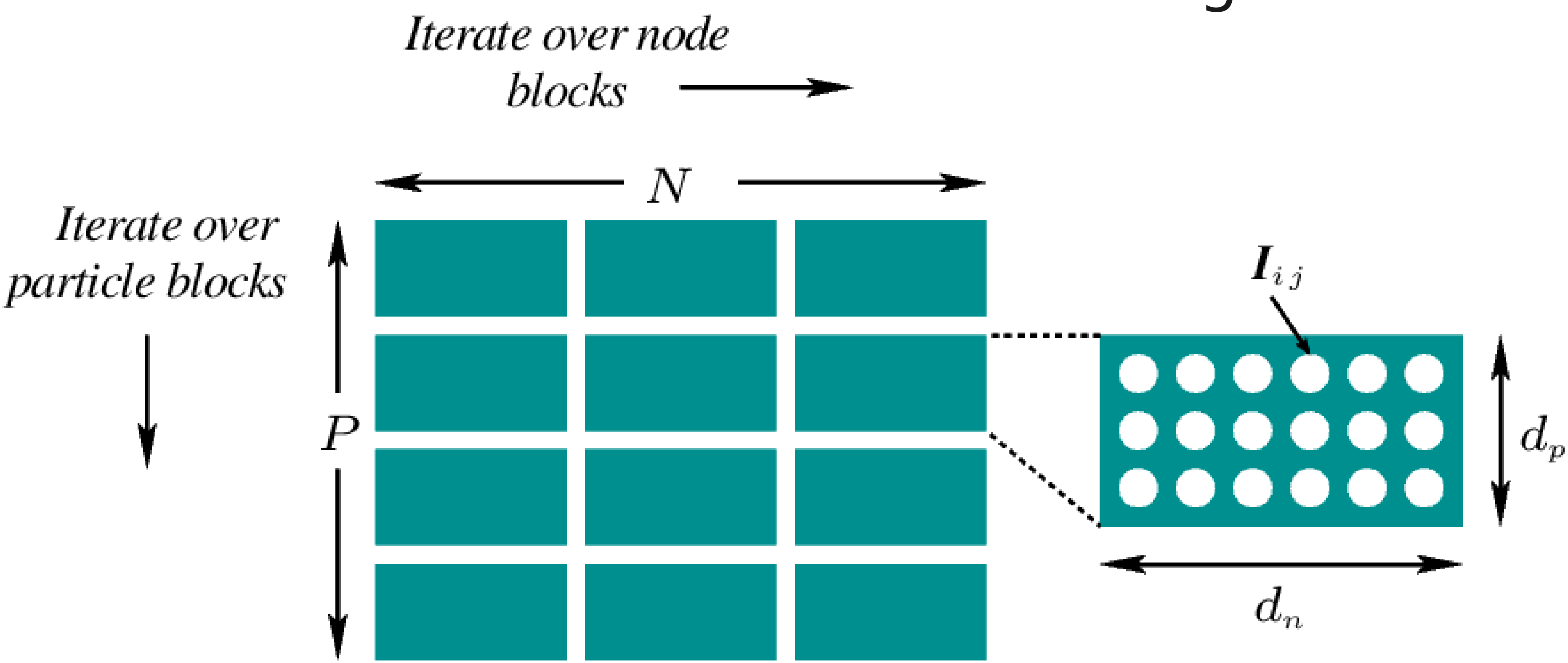


CUDA memory model

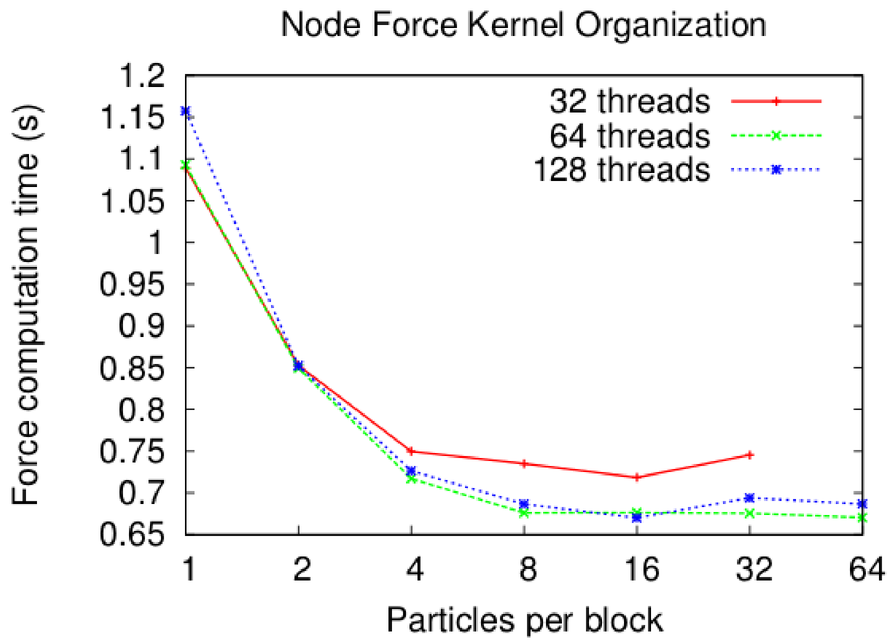


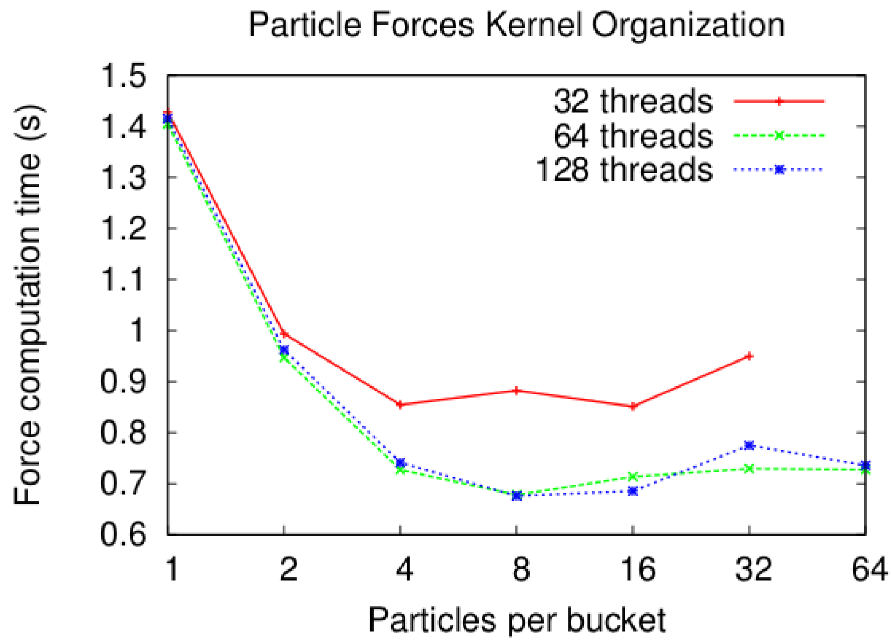
Force Kernel Optimization

More particles->fewer loads
More particles->larger shared memory use
Fewer executing blocks



Kernel Optimization Results





Optimum at 128 threads, 16 particles, 8 nodes/block

Ewald on the GPU

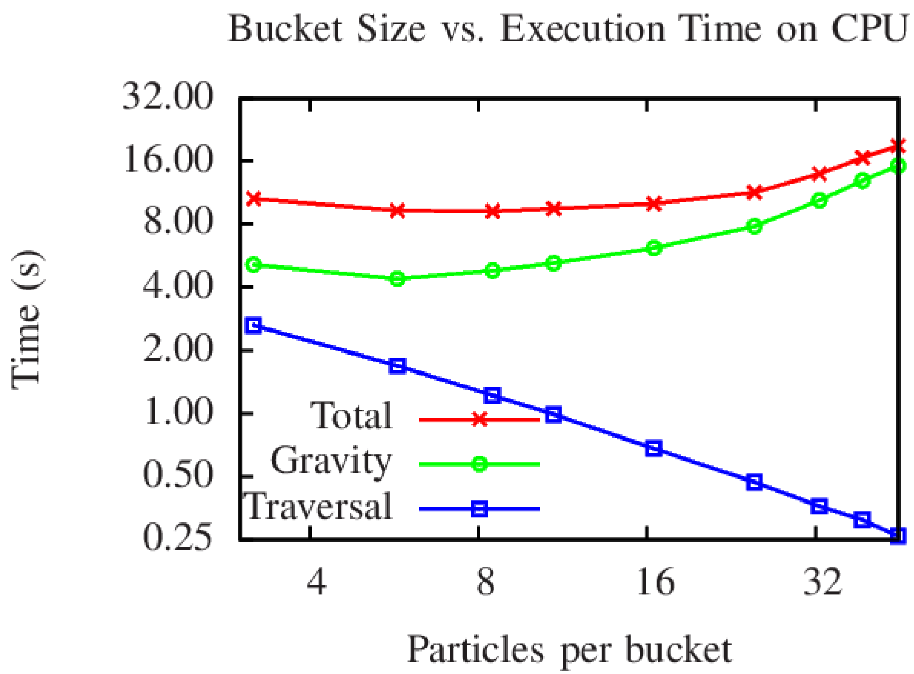
- Real space loop and Fourier space loop
- Separate kernels for each loop

  More concurrent blocks/SM
- Constant memory for cos/sin tables
- Factor of 20 speedup over CPU

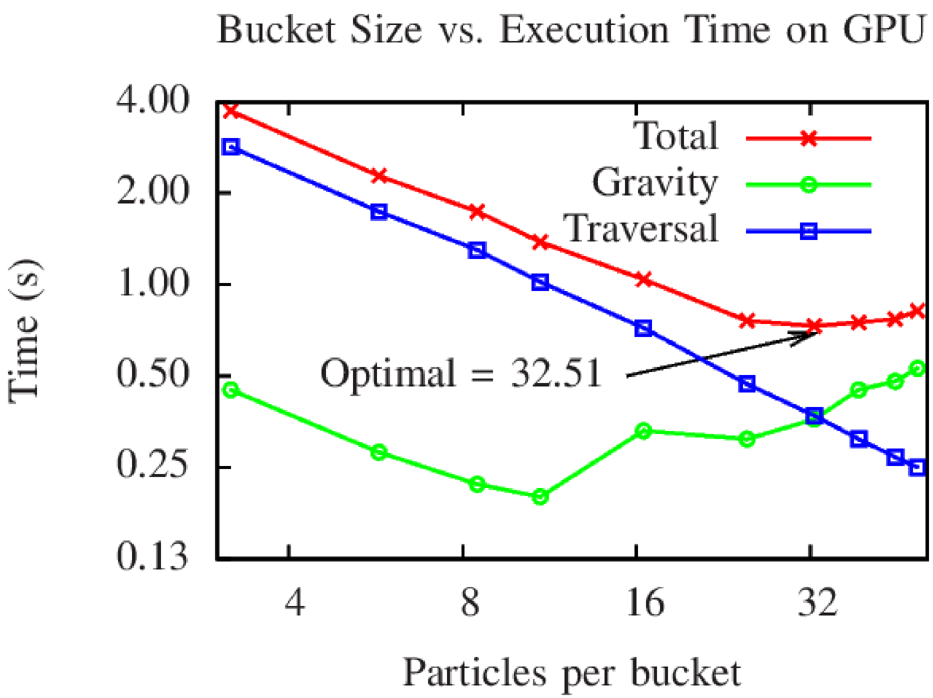
Tree Traversal and Computation

- GPU is hungry for work
  - CPU should hold back GPU
  - Decrease tree walk time to generate more computing
- Increase average bucket size
  - Tree is shallower: CPU less busy
  - More computation: GPU more busy
  - Balance for optimum

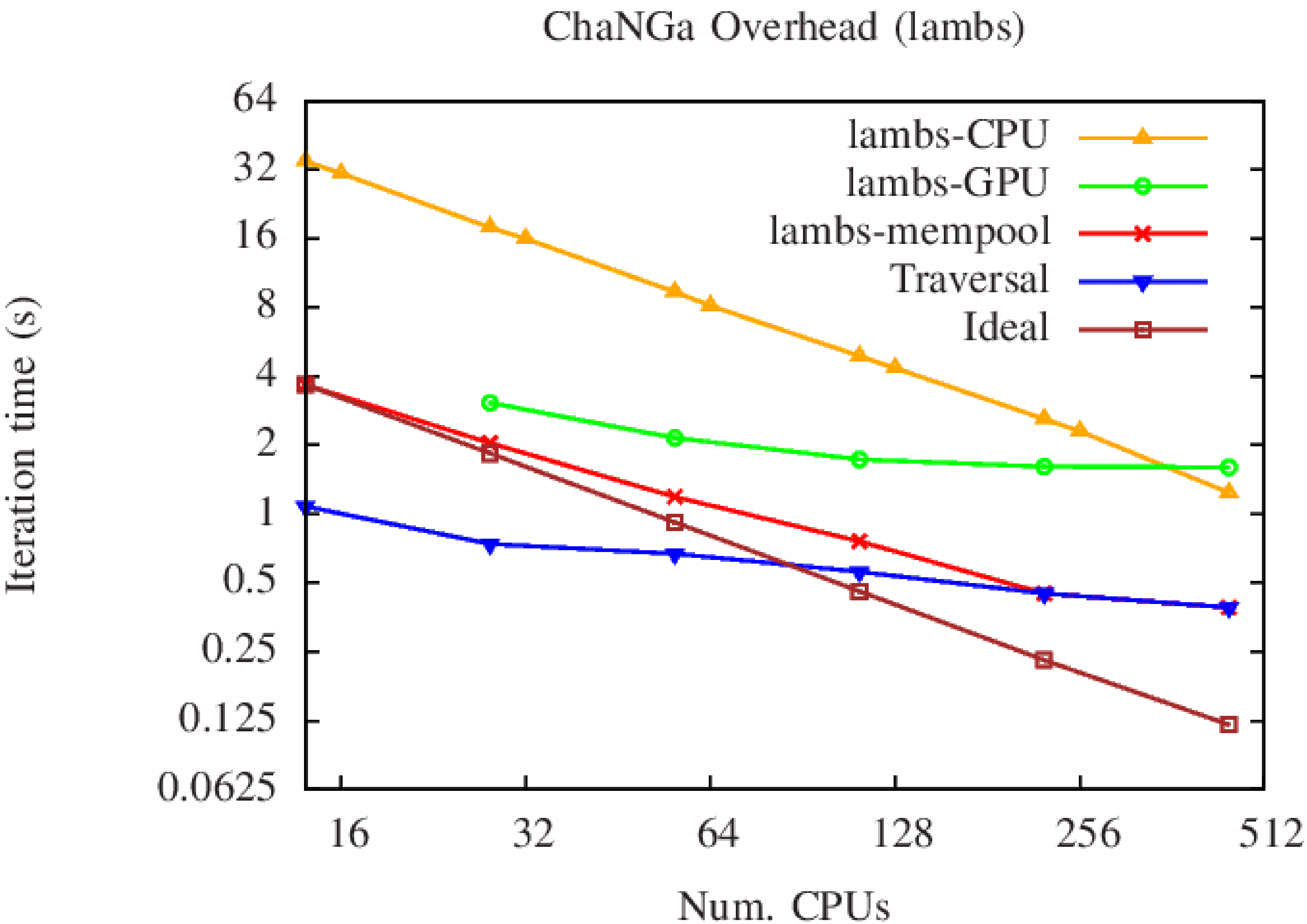
Work-throughput tradeoff

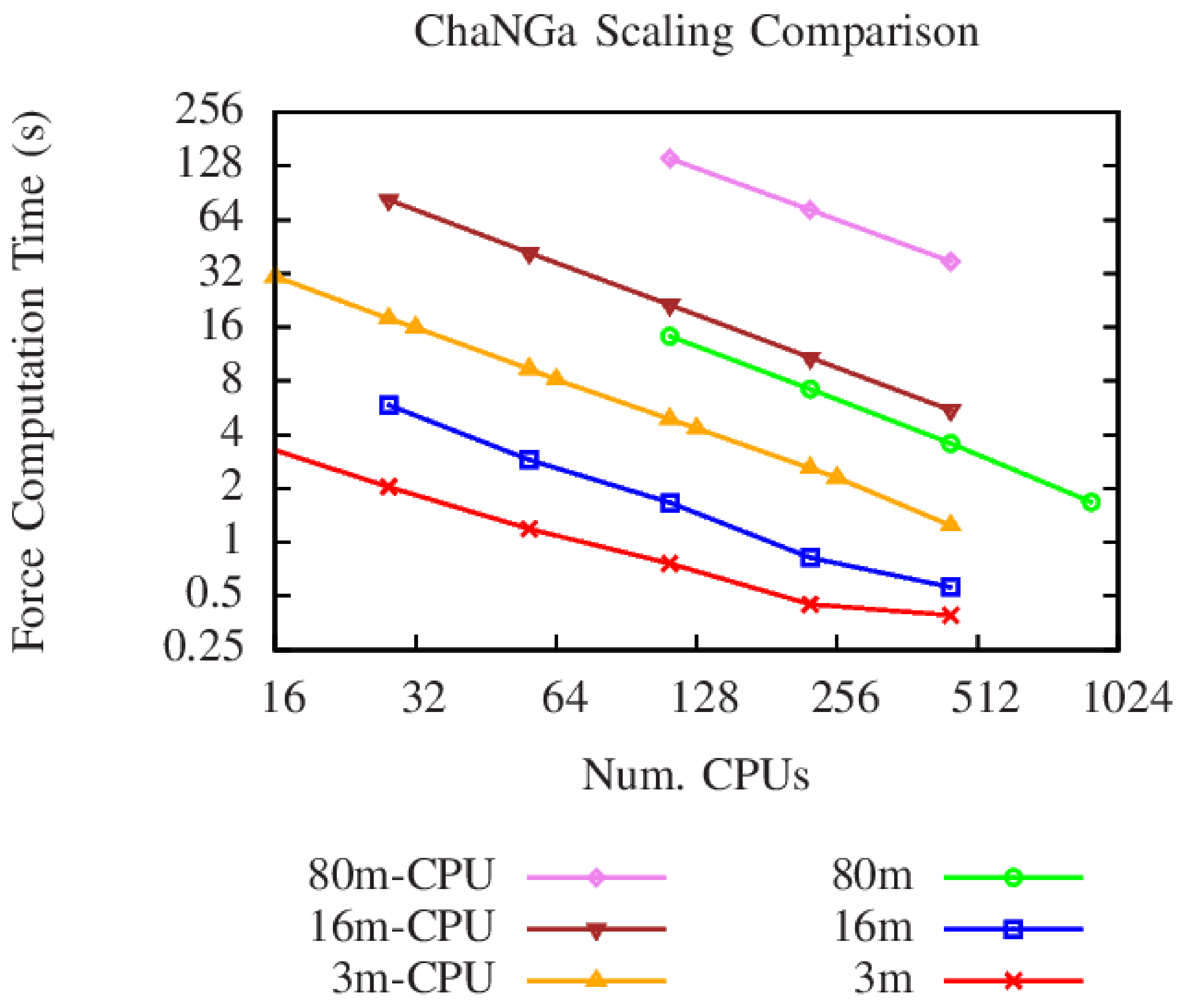


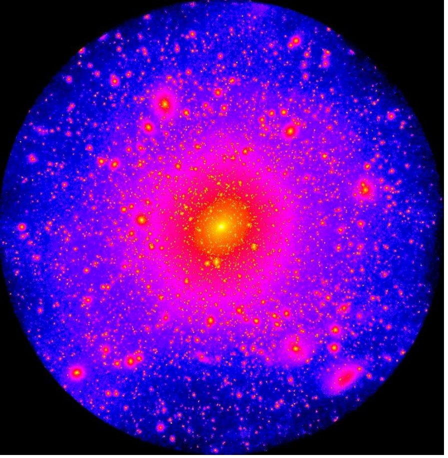
CPU



GPU

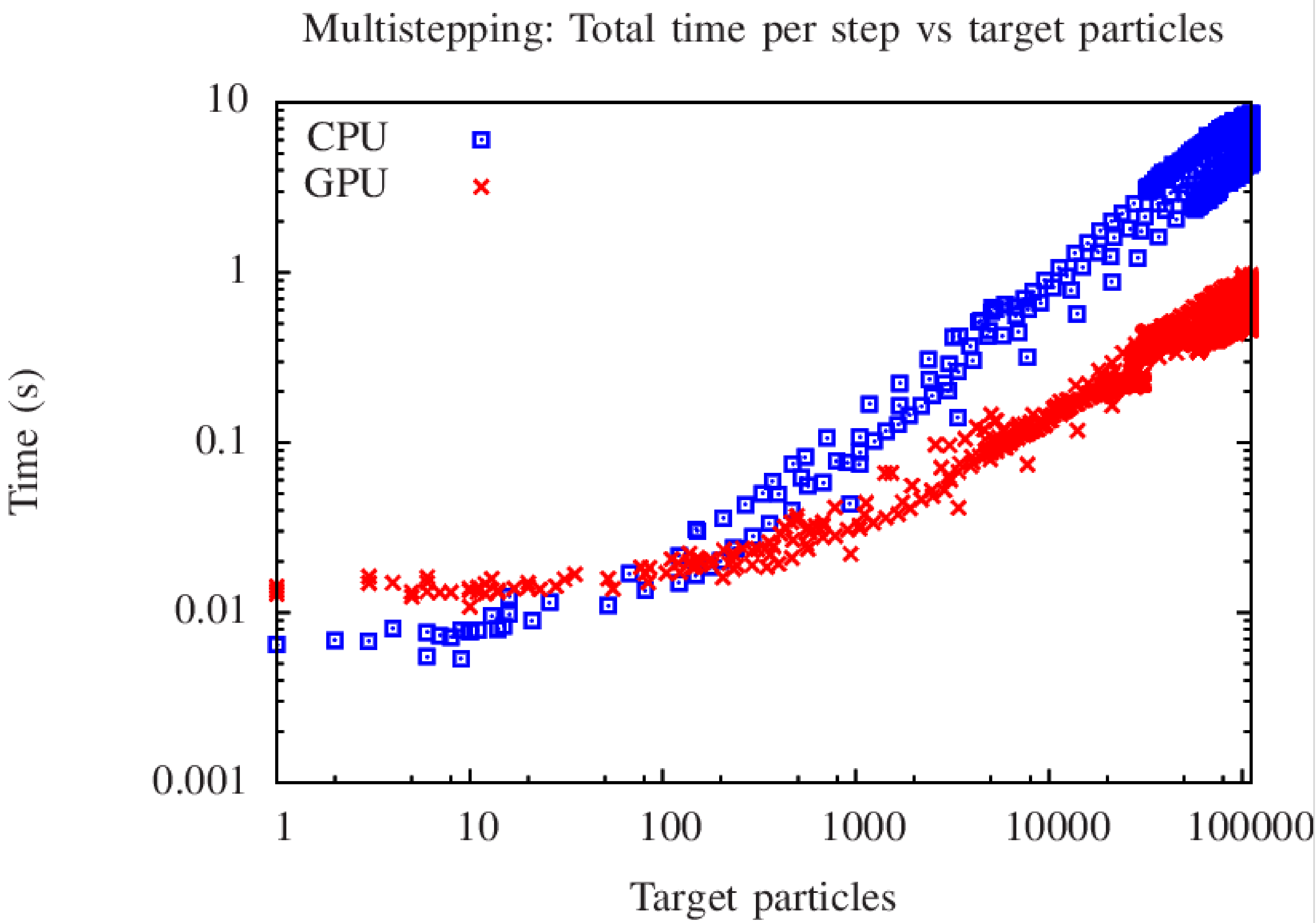

ChaNGa Overhead (lambs)
lambs-CPU
lambs-GPU
lambs-mempool
Traversal
Ideal
Iteration time (s)
64
32
16
8
4
2
1
0.5
0.25
0.125
0.0625
16
32
64
128
256
512
Num. CPUs


ChaNGa Scaling Comparison
Force Computation Time (s)
256
128
64
32
16
8
4
2
1
0.5
0.25
16
32
64
128
256
512
1024
Num. CPUs
80m-CPU
16m-CPU
3m-CPU
80m
16m
3m



Timestepping Challenges

- 1/*m* particles need *m* times more force evaluations
- Naively, simulation cost scales as N^(4/3)ln(N)
  - This is a problem when N ~ 1e9 or greater
- If each particle an individual timestep scaling reduces to N (ln(N))^2
- A difficult dynamic load balancing problem



Multistepping: Total time per step vs target particles
10
1
0.1
0.01
0.001
Time (s)
CPU
GPU
1
10
100
1000
10000
100000
Target particles

GPU Summary/prognosis

- Successfully kept the monster fed
- More floats yet better throughput
- More work to do:
  - Load balancing needs more sophistication
  - Higher order multipoles/single precision
  - Multistepping optimization
  - Tree traversal on the GPU?
  - Ease of Programming

hpcc.astro.washington.edu/tools/changa.html