ENZO Simulations at PetaScale

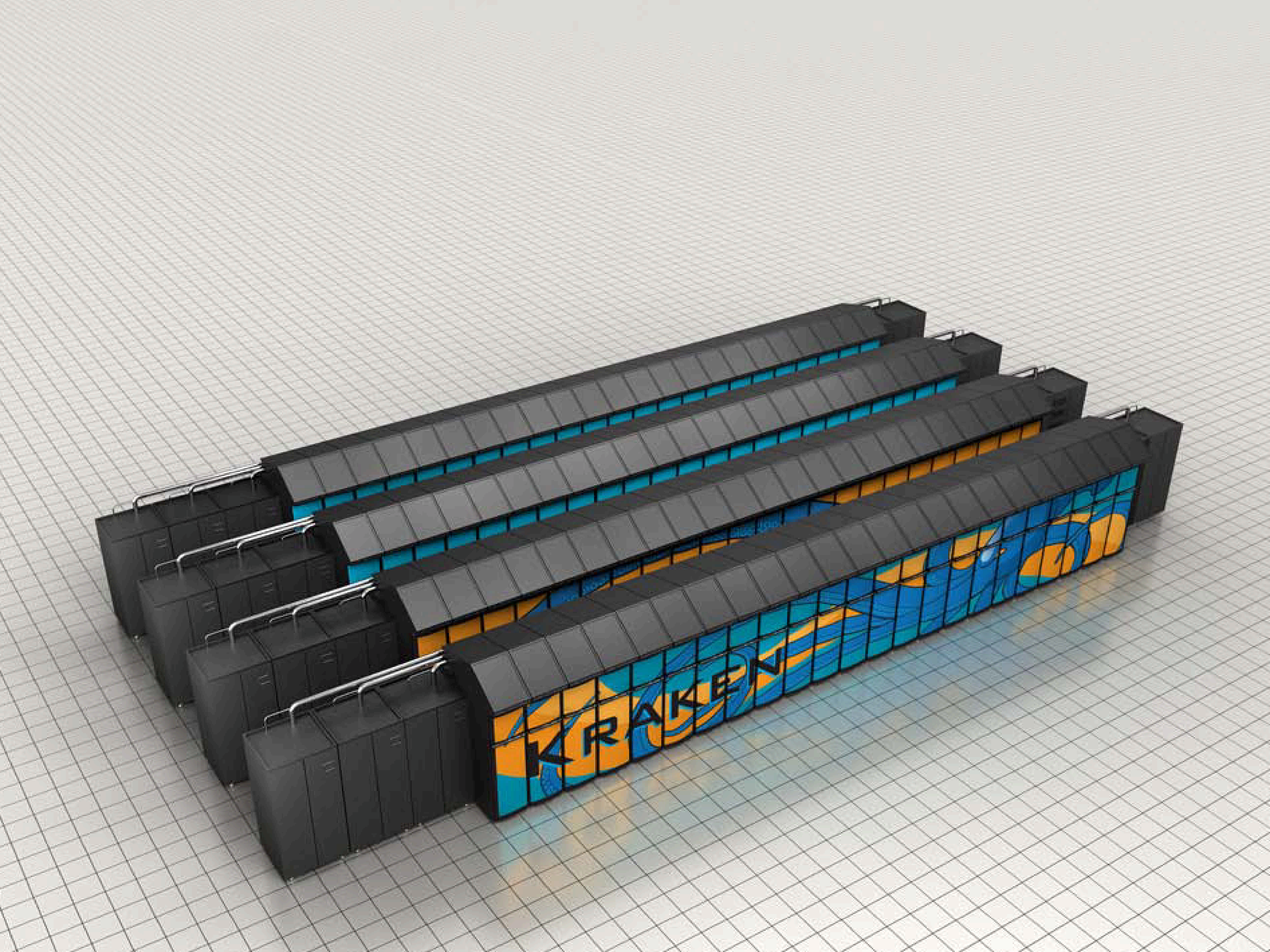
Robert Harkness

UCSD/SDSC

December 17th, 2010

Acknowledgements

- LCA team members past and present
- Phil Andrews and all the staff at NICS
  - Especially Glenn Brook, Mark Fahey
  - Outstanding support by all concerned
- The HDF5 Group
  - Thanks for those in-core drivers!


KRAKEN

The ENZO Code(s)

- General-purpose Adaptive Mesh Refinement (AMR) code
  - Hybrid physics capability for cosmology
    - PPM Eulerian hydro and collisionless dark matter (particles)
    - Grey radiation diffusion, coupled chemistry and RHD
  - Extreme AMR to > 35 levels deep
    - > 500,000 subgrids
    - AMR load-balancing and MPI task-to-processor mapping
  - Ultra large-scale non-AMR applications at full scale on NICS XT5
  - High performance I/O using HDF5
  - C, C++ and Fortran90, >> 185,000 LOC

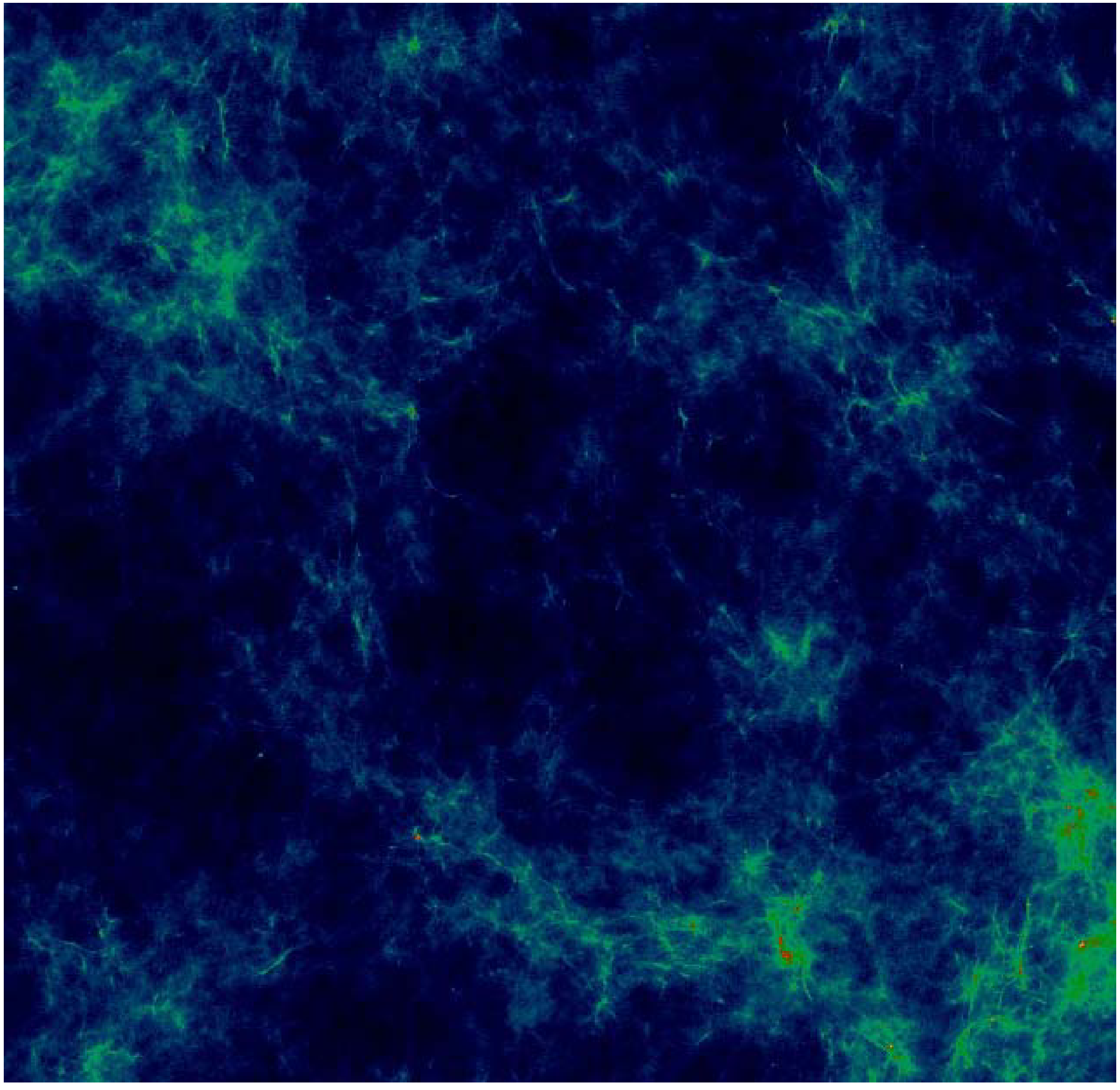
ENZO - One code, different modes

- ENZO-C
  - Conventional ENZO cosmology code
  - MPI and OpenMP hybrid, AMR and non-AMR
- ENZO-R
  - ENZO + Grey flux-limited radiation diffusion
    - Coupled chemistry and radiation hydrodynamics
  - MPI and OpenMP hybrid (in ENZO and HYPRE)
- Two simultaneous levels of OpenMP threading
  - Root grid decomposition (static work distribution)
  - Loop over AMR subgrids on each level (dynamic)
  - Allows memory footprint to grow at fixed MPI task count
    - E.g. 1 to 12 OpenMP threads per task, 10x memory range

Hybrid ENZO on the Cray XT5

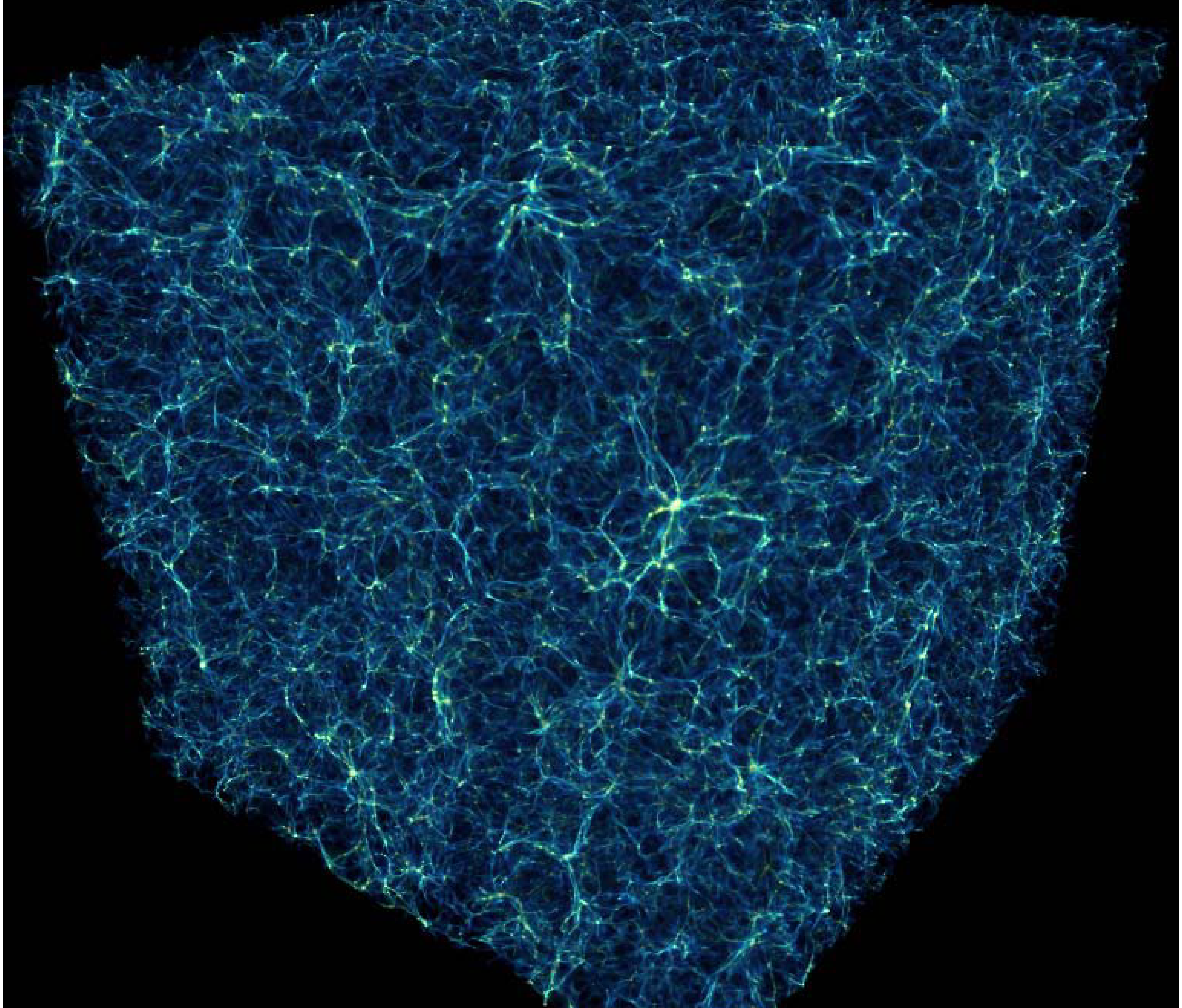
- ULTRA : non-AMR 6400^3 80 Mpc box
  - Designed to “fit” on the upgraded NICS XT5 Kraken
  - **268 billion** zones, **268 billion** dark matter particles
  - 15,625 (25^3) MPI tasks, 256^3 root grid tiles
  - 6 OpenMP threads per task, 1 MPI task per socket
  - 93,750 cores, 125 TB memory
  - 30 TB per checkpoint/re-start/data dump
  - >15 GB/sec read, >7 GB/sec write, non-dedicated
  - 1500 TB of output
  - Cooperation with NICS staff essential for success

1% of the 6400^3 simulation



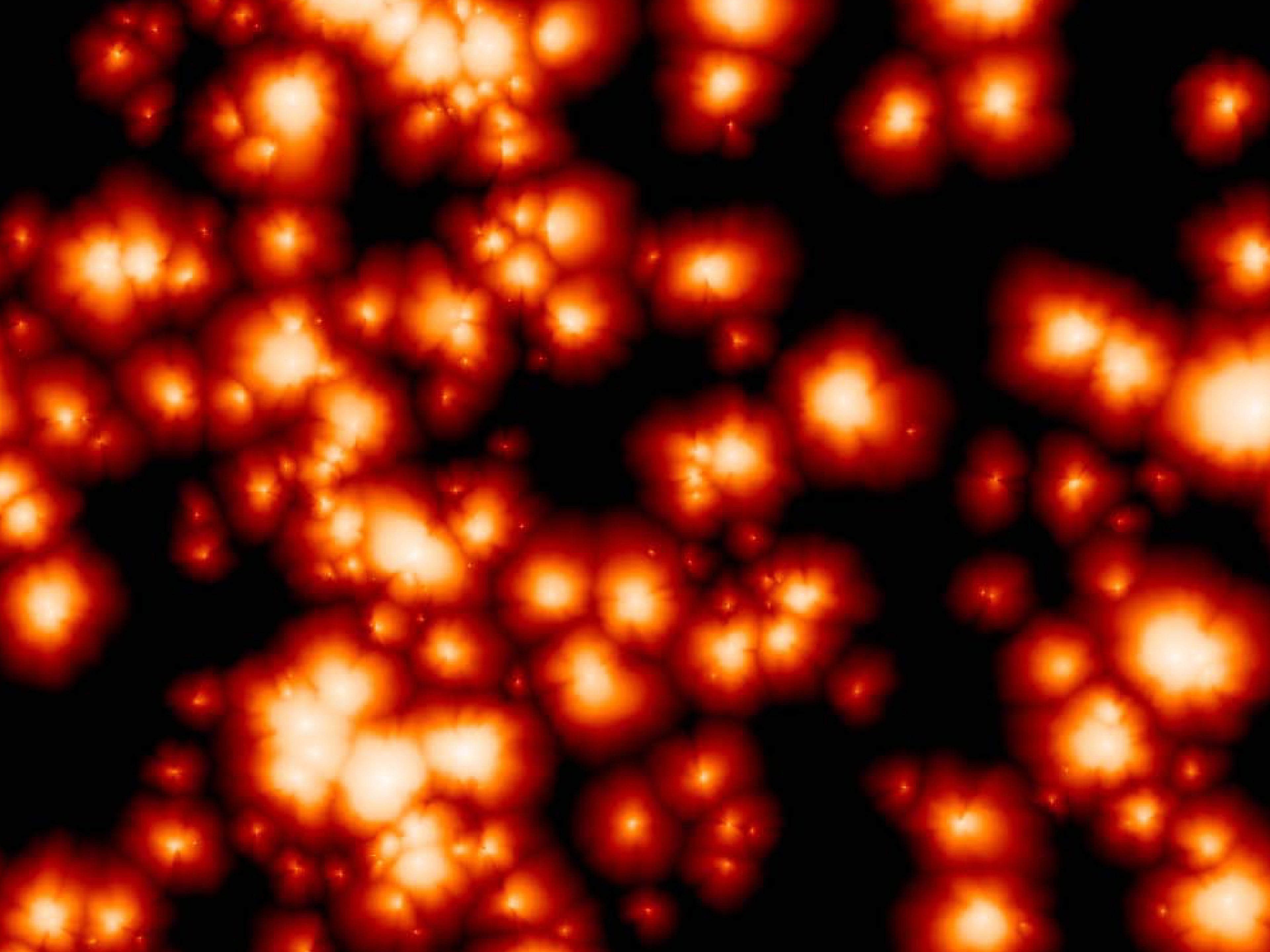
Hybrid ENZO-C on the Cray XT5

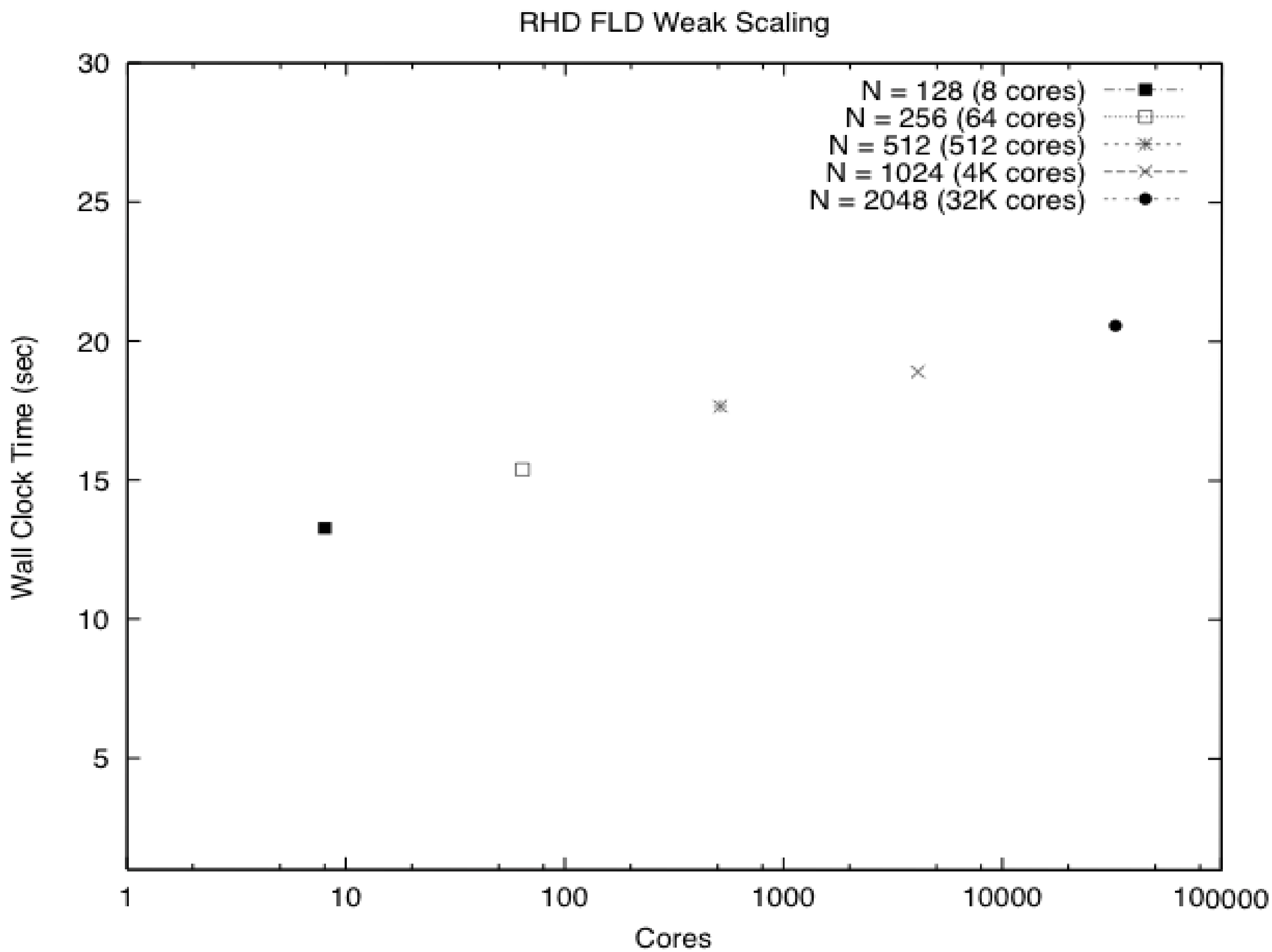
- AMR 1024^3 50 Mpc box, 7 levels of refinement
  - 4096 (16^3) MPI tasks, 64^3 root grid tiles
  - Refine “everywhere”
  - 1 to 6 OpenMP threads per task - 4096 to 24576 cores
- Increase thread count with AMR memory growth
  - Fixed number of MPI tasks
  - Initially 12 MPI tasks per node, 1.3 GB/task
  - As AMR develops
    - Increase node count => larger memory per task
    - Increase threads per MPI task => keep all cores busy
    - On XT5 this can allow for up to 12x growth in memory
    - Load balance can be poor when Ngrid << Nthread



ENZO-R on the Cray XT5

- Non-AMR 1024^3 8 and 16 Mpc to Z=4
  - 4096 (16^3) MPI tasks, 64^3 root grid tiles
  - LLNL Hypre precondioner & solver for radiation
    - near ideal scaling to at least 32K MPI tasks
  - Hypre is threaded with OpenMP
    - LLNL working on improvements
    - Hybrid Hypre built on multiple platforms
  - Power7 testing in progress for Blue Waters
    - performance ~2x AMD Istanbul
    - Very little gain from Power7 VSX (so far)





RHD FLD Weak Scaling
Wall Clock Time (sec)
Cores
N = 128 (8 cores)
N = 256 (64 cores)
N = 512 (512 cores)
N = 1024 (4K cores)
N = 2048 (32K cores)
30
25
20
15
10
5
1
10
100
1000
10000
100000

2011 INCITE : Re-Ionizing the Universe

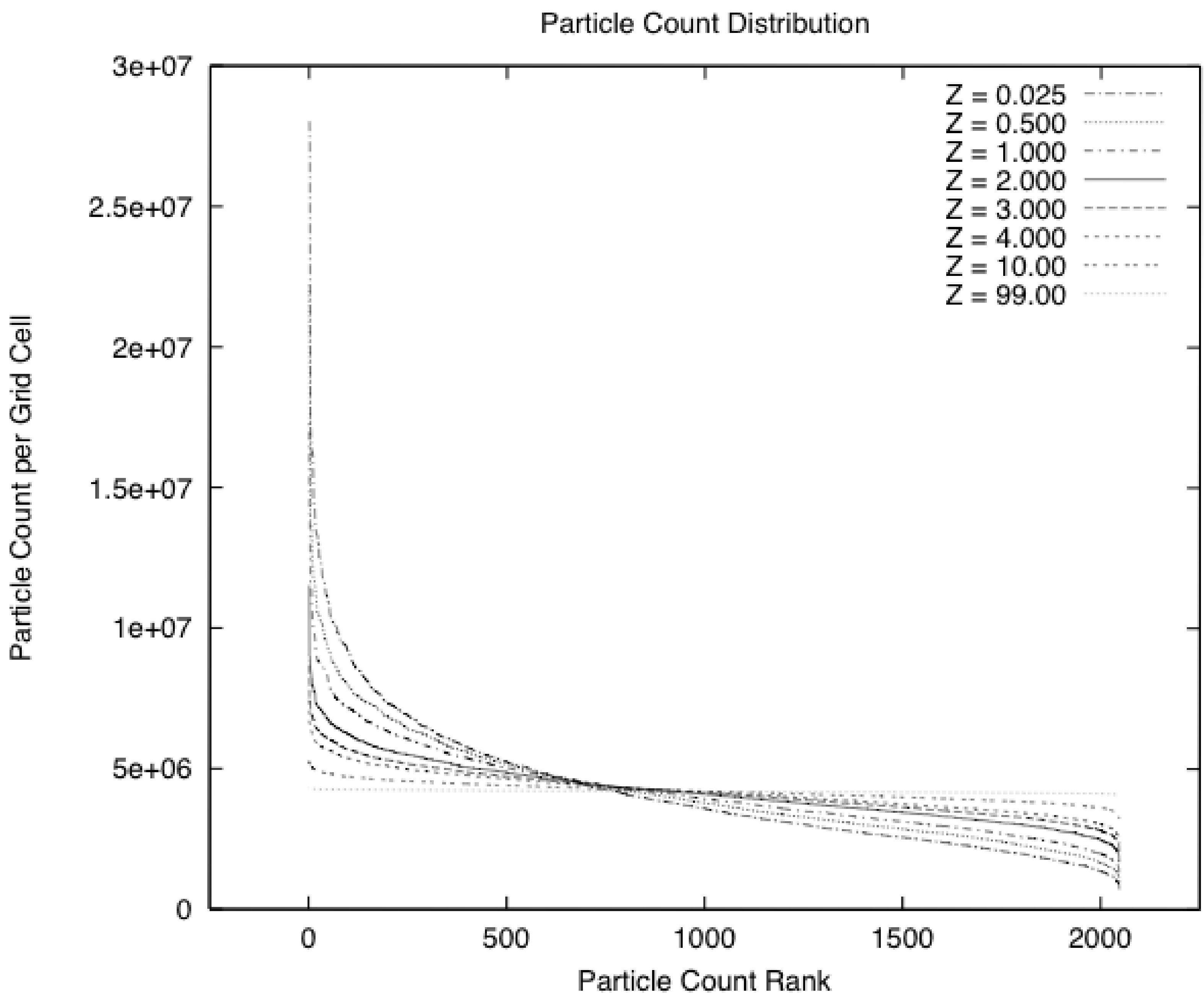
- Non-AMR 3200^3 to 4096^3 RHD with ENZO-R
  - Hybrid MPI and OpenMP on NCCS Jaguar XT5
  - SMT and SIMD tuning
  - 80^3 to 200^3 root grid tiles
  - 1-6 OpenMP threads per task
  - > 64 - 128K cores total
  - > 8 TBytes per checkpoint/re-start/data dump (HDF5)
  - Asynchronous I/O and/or inline analysis
  - In-core intermediate checkpoints
  - 64-bit arithmetic, 64-bit integers and pointers
  - 35 M hours

Near-term Future Developments

- Enhancements to OpenMP threading
  - Prepare for at least 8 threads per task
- Prototype RHD Hybrid ENZO + Hypre
  - Running on NCSA Blue Drop
  - Performance is ~2x Cray XT5, per core
  - SIMD tuning for Power7 VSX
- PGAS with UPC
  - 4 UPC development paths
  - Function and Scalability
- 8192^3 HD, 4096^3 RHD and 2048^3 L7 AMR
  - All within the range of NCSA/IBM Blue Waters

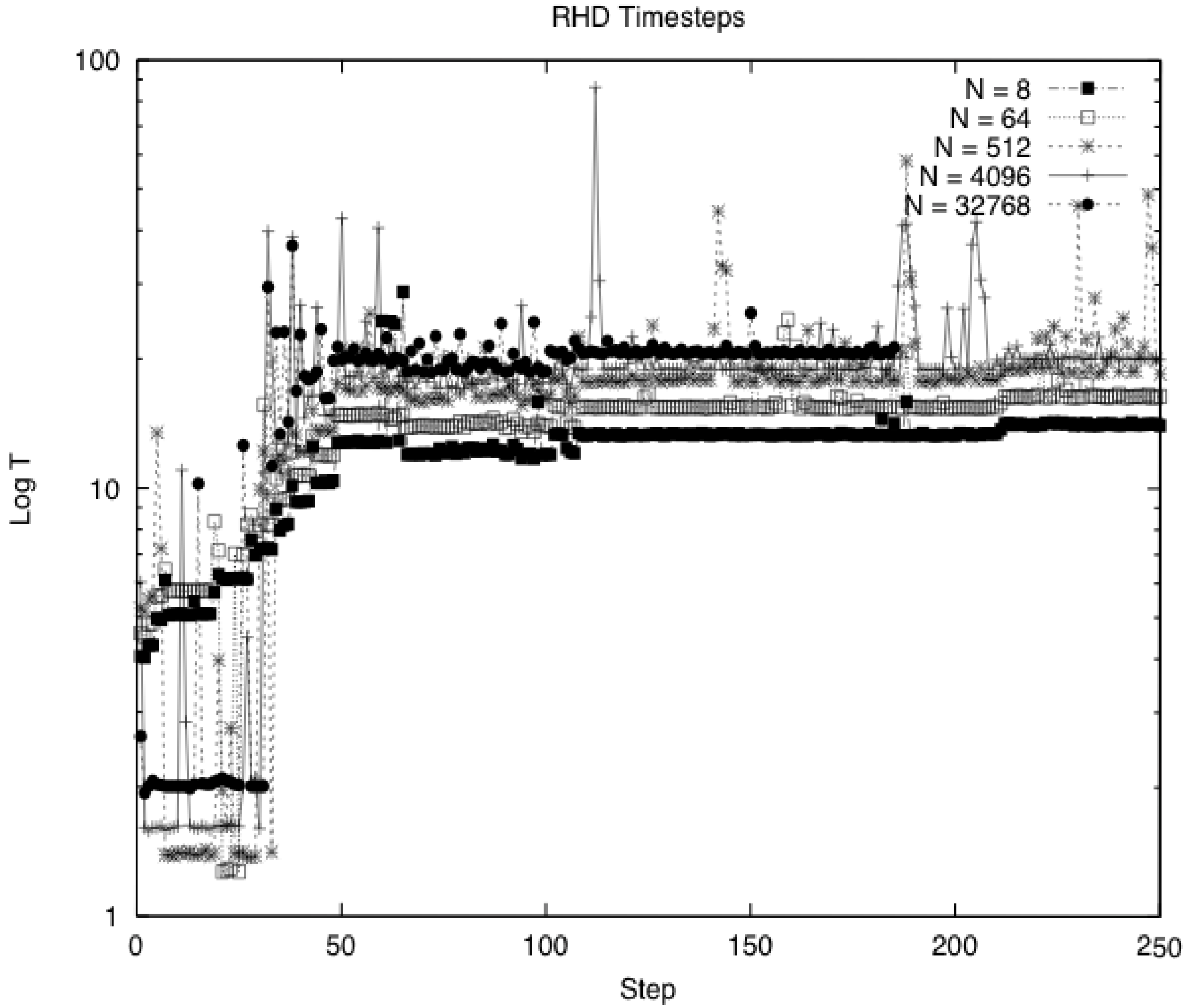
PGAS in ENZO

- Dark Matter Particles
  - Use UPC to distribute particles evenly
  - Eliminates potential node memory exhaustion
- AMR Hierarchy
  - UPC to eliminate replication
  - Working with DK Panda (Ohio)
- Replace 2-sided MPI
  - Gradually replace standard MPI
  - Replace blocking collectives
- Replace OpenMP *within* a node


Particle Count Distribution
Z = 0.025
Z = 0.500
Z = 1.000
Z = 2.000
Z = 3.000
Z = 4.000
Z = 10.00
Z = 99.00
Particle Count per Grid Cell
3e+07
2.5e+07
2e+07
1.5e+07
1e+07
5e+06
0
0
500
1000
1500
2000
Particle Count Rank

Dirty Laundry List

- Full-scale runs are severely exposed to
  - Hardware MTBF on 100K cores
  - Any I/O errors
  - Any interconnect link errors, MPI tuning
  - Scheduling and sharing (dedicated is best)
  - OS jitter
  - **SILENT** data corruption!
- Large codes are more exposed to:
  - Compiler bugs and instability (especially OpenMP)
  - Library software revisions (incompatibility)
    - NICS & NCCS do a great job of controlling this
  - Heap fragmentation (especially AMR)



RHD Timesteps
100
10
1
Log T
0
50
100
150
200
250
Step
N = 8
N = 64
N = 512
N = 4096
N = 32768

More Dirty Laundry

- HW MTBF => checkpointing @ 6hrs
  - With failures ~50% overhead in cost
- I/O is relatively weak on Kraken
  - Phased I/O to spare other users
  - Reduced I/O performance by 30-40%
  - Re-start ~12 GB/sec (45 min)
  - Checkpoint write ~7 GB/sec (75 min)
- Remote file xfer ~ 500 MB/sec
  - But no other sites can manage 30 TB!
- Archive file xfer ~300 MB/sec
  - Only ORNL/NICS HPSS can manage ~1 PB

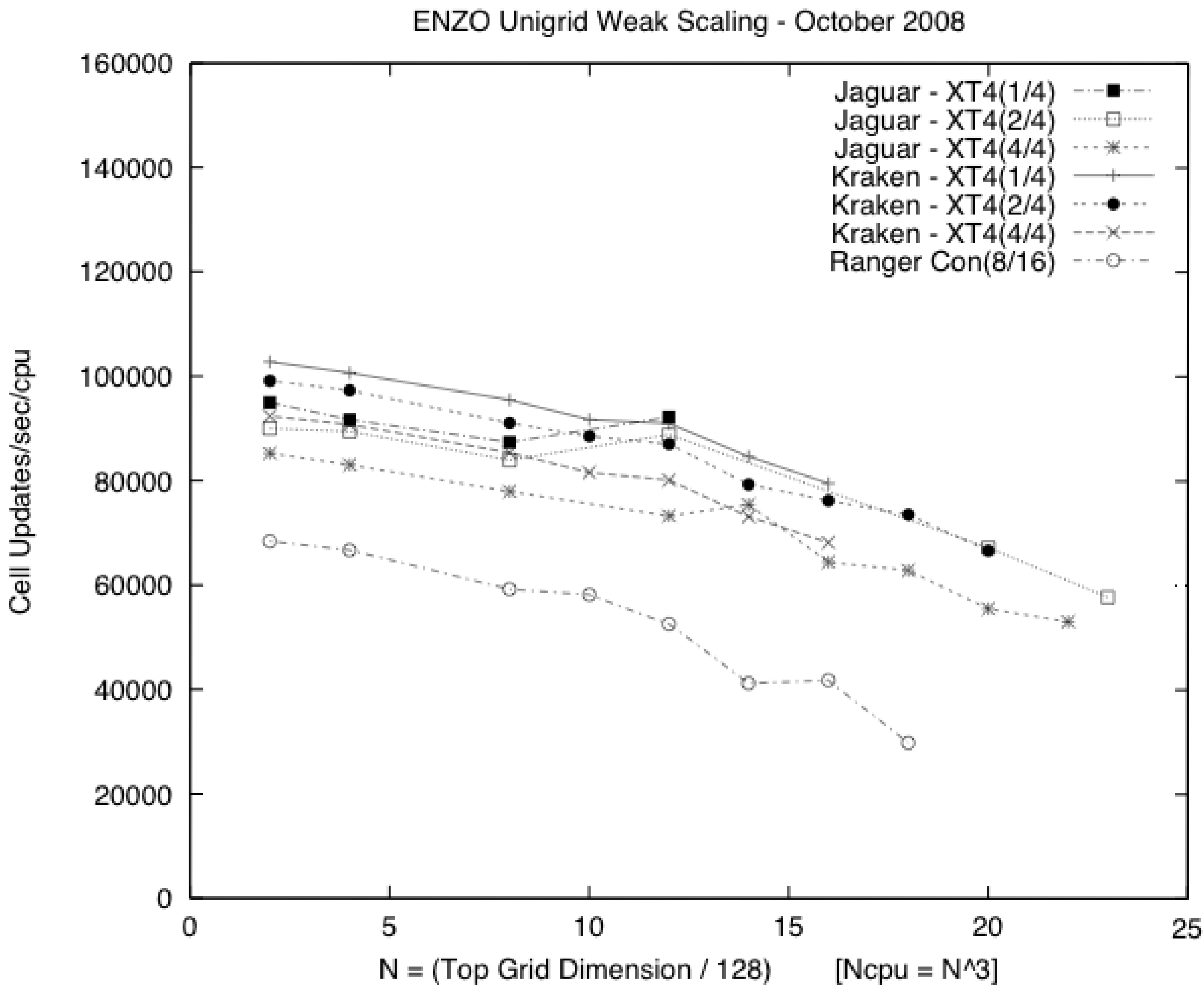
Choose a machine, choose your future

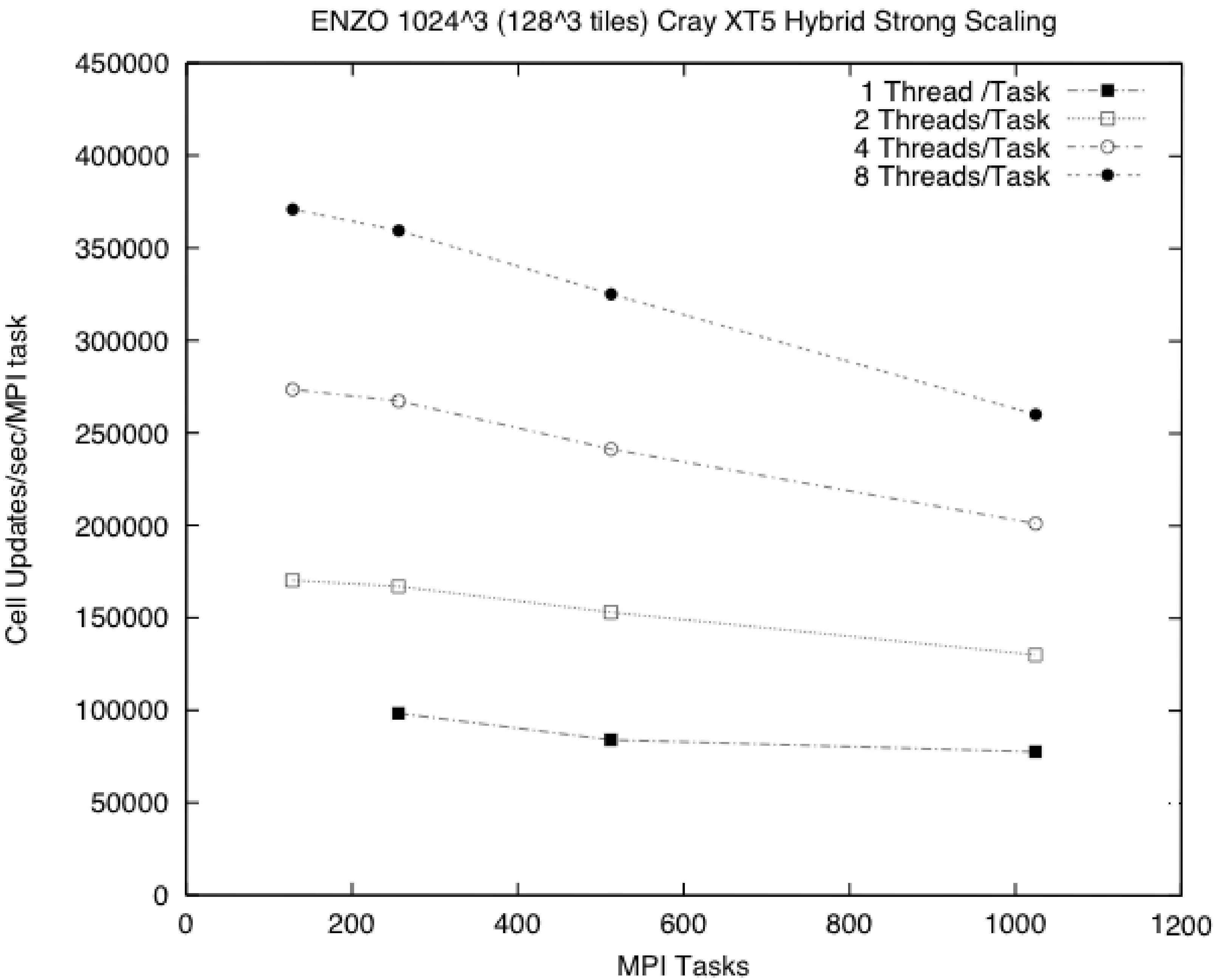
- Aggregate memory limits what you could do
- Cost decides what you **can** do ~100M hrs/sim?
- End of the weak scaling era with Blue Waters?
- I/O for data and benchmarking is now critical
  - Traditional checkpointing is *impossible* at exascale
- Current GPUs require contiguous, aligned access
  - Re-structuring for this can require new algorithms
    - E.g. consider directionally-split strides 1, N, N^2
- GPU data must reside permanently in GPU memory
  - External functions as “decelerators” (LANL Cell)
  - GPU memory is smaller - what can fit given the flops?
- Memory bandwidth often determines the bottom line

Future without GPGPUs?

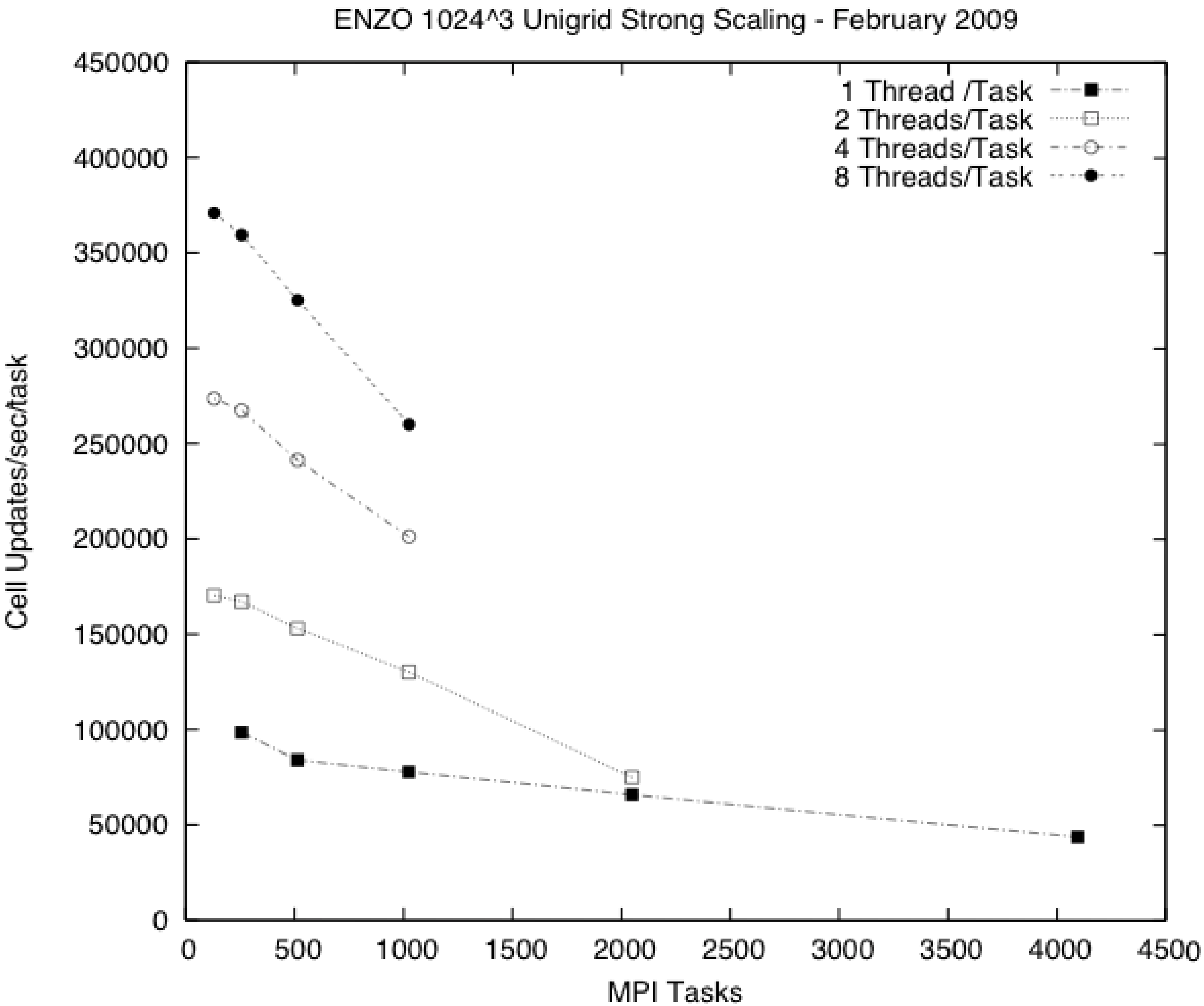
- Larrabee-like instruction set (LRBni)
  - Vector registers, masks, gather-scatter
  - Traditional vectorization / compilers
  - No restrictions on stride or alignment
  - X86 code
  - Can run the O/S!
  - Intel Knight's Ferry/Knight's Corner
- Custom accelerators, FPGAs, PIM?
- PGAS at multiple levels
  - UPC is the leading choice, lowest risk
- At Exascale, HW MTBF is probably a killer

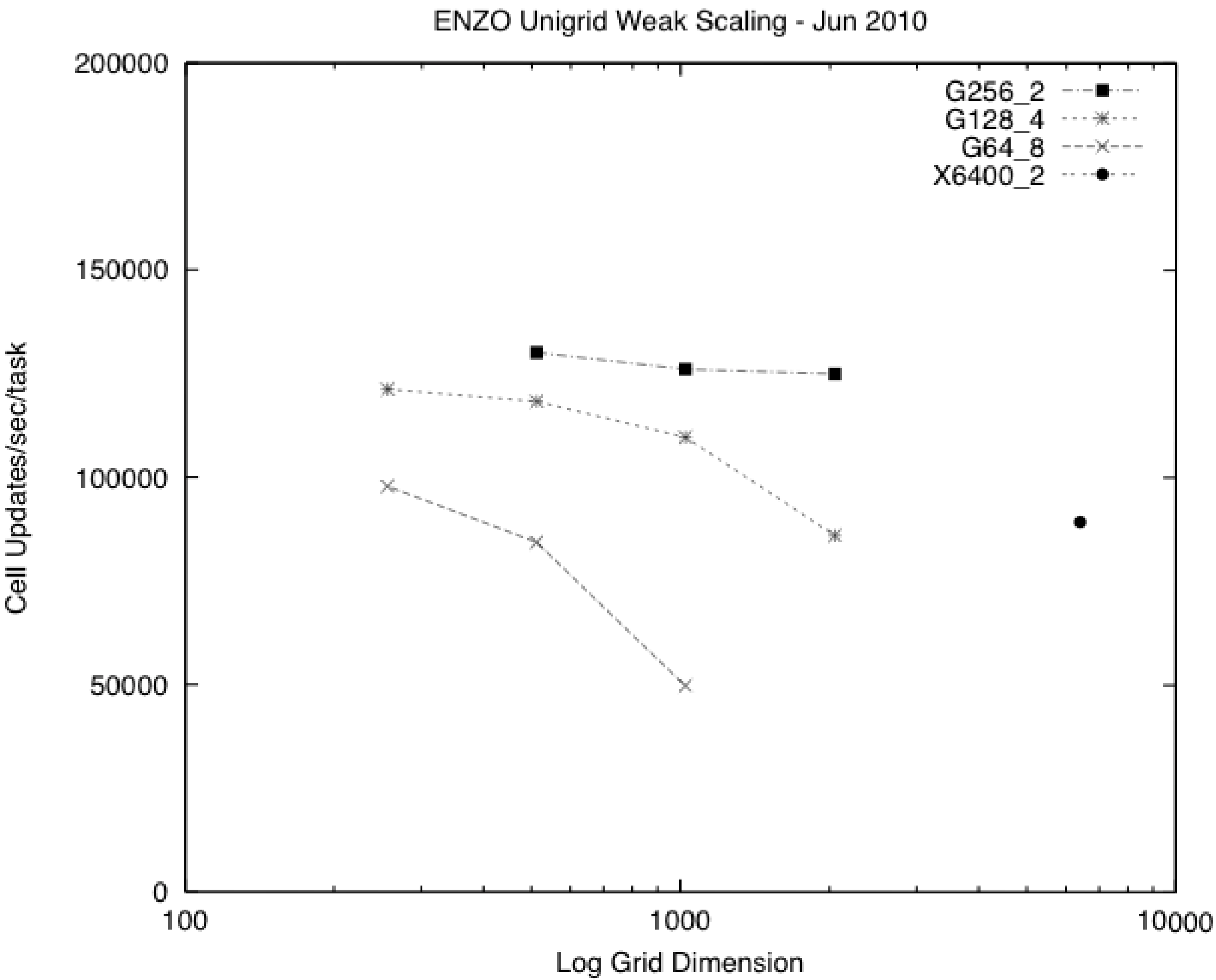

HIGH EFFICIENCY
HIGH EFFICIENCY
KRAKEN
35


ENZO Unigrid Weak Scaling - October 2008
Cell Updates/sec/cpu
N = (Top Grid Dimension / 128)
[Ncpu = N^3]
Jaguar - XT4(1/4)
Jaguar - XT4(2/4)
Jaguar - XT4(4/4)
Kraken - XT4(1/4)
Kraken - XT4(2/4)
Kraken - XT4(4/4)
Ranger Con(8/16)
0
20000
40000
60000
80000
100000
120000
140000
160000
0
5
10
15
20
25



ENZO 1024^3 (128^3 tiles) Cray XT5 Hybrid Strong Scaling
1 Thread /Task
2 Threads/Task
4 Threads/Task
8 Threads/Task
Cell Updates/sec/MPI task
MPI Tasks
0
50000
100000
150000
200000
250000
300000
350000
400000
450000
200
400
600
800
1000
1200


ENZO 1024^3 Unigrid Strong Scaling - February 2009
1 Thread /Task
2 Threads/Task
4 Threads/Task
8 Threads/Task
Cell Updates/sec/task
MPI Tasks
0
50000
100000
150000
200000
250000
300000
350000
400000
450000
500
1000
1500
2000
2500
3000
3500
4000
4500


ENZO Unigrid Weak Scaling - Jun 2010
G256_2
G128_4
G64_8
X6400_2
Cell Updates/sec/task
0
50000
100000
150000
200000
100
1000
10000
Log Grid Dimension