Charm++ Applications

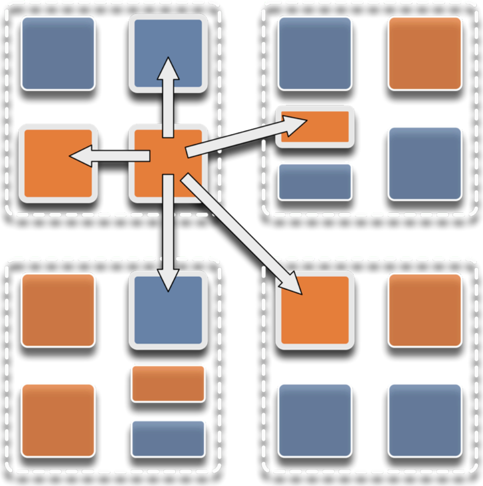
Laxmikant (Sanjay) Kale
http://charm.cs.illinois.edu
Parallel Programming Laboratory
Department of Computer Science
University of Illinois at Urbana Champaign

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

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

Overdecomposition

- Decompose the work units & data units into many more pieces than execution units
  - Cores/Nodes/..
- Not so hard: we do decomposition anyway



Migratability

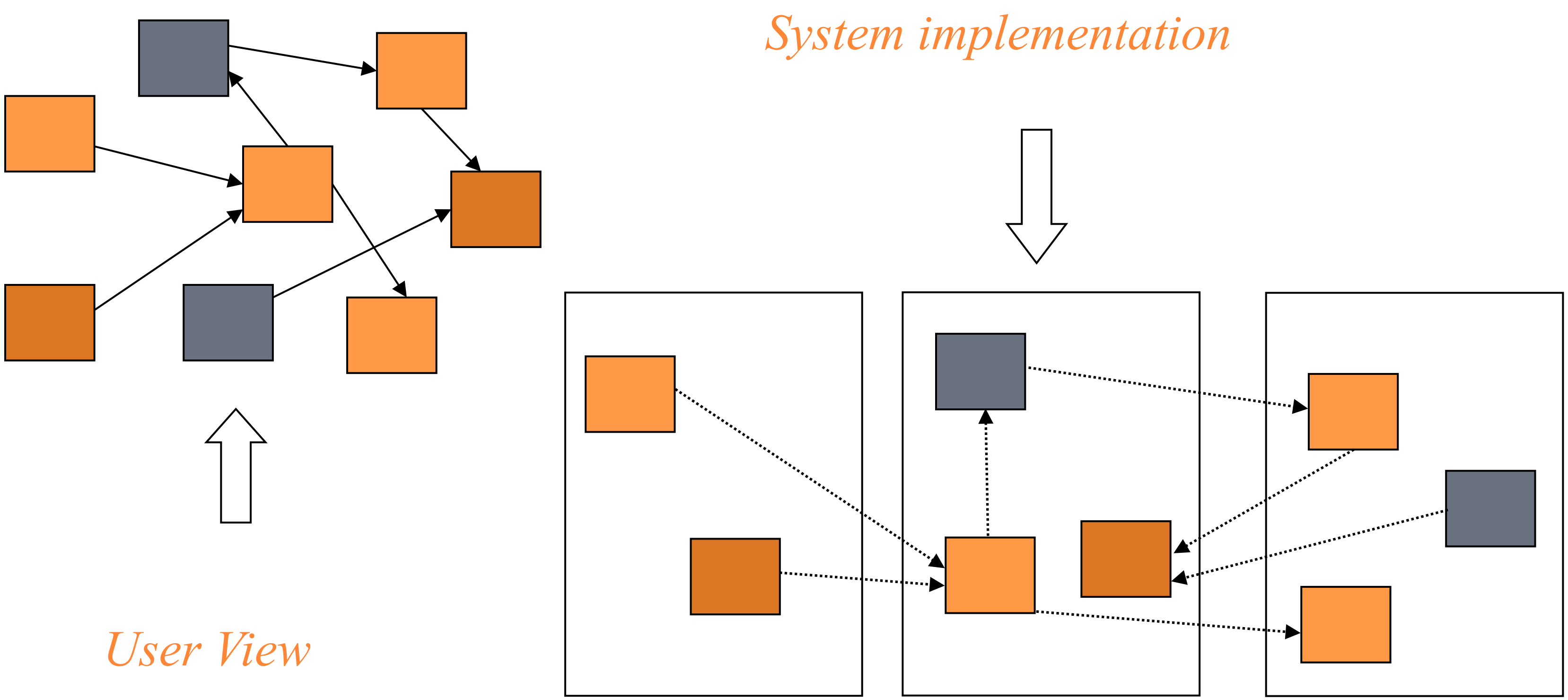
- Allow these work and data units to be migratable at runtime
  - i.e. the programmer or runtime, can move them
- Consequences for the app-developer
  - Communication must now be addressed to logical units with global names, not to physical processors
  - But this is a good thing
- Consequences for RTS
  - Must keep track of where each unit is
  - Naming and location management

Asynchrony: Message-Driven Execution

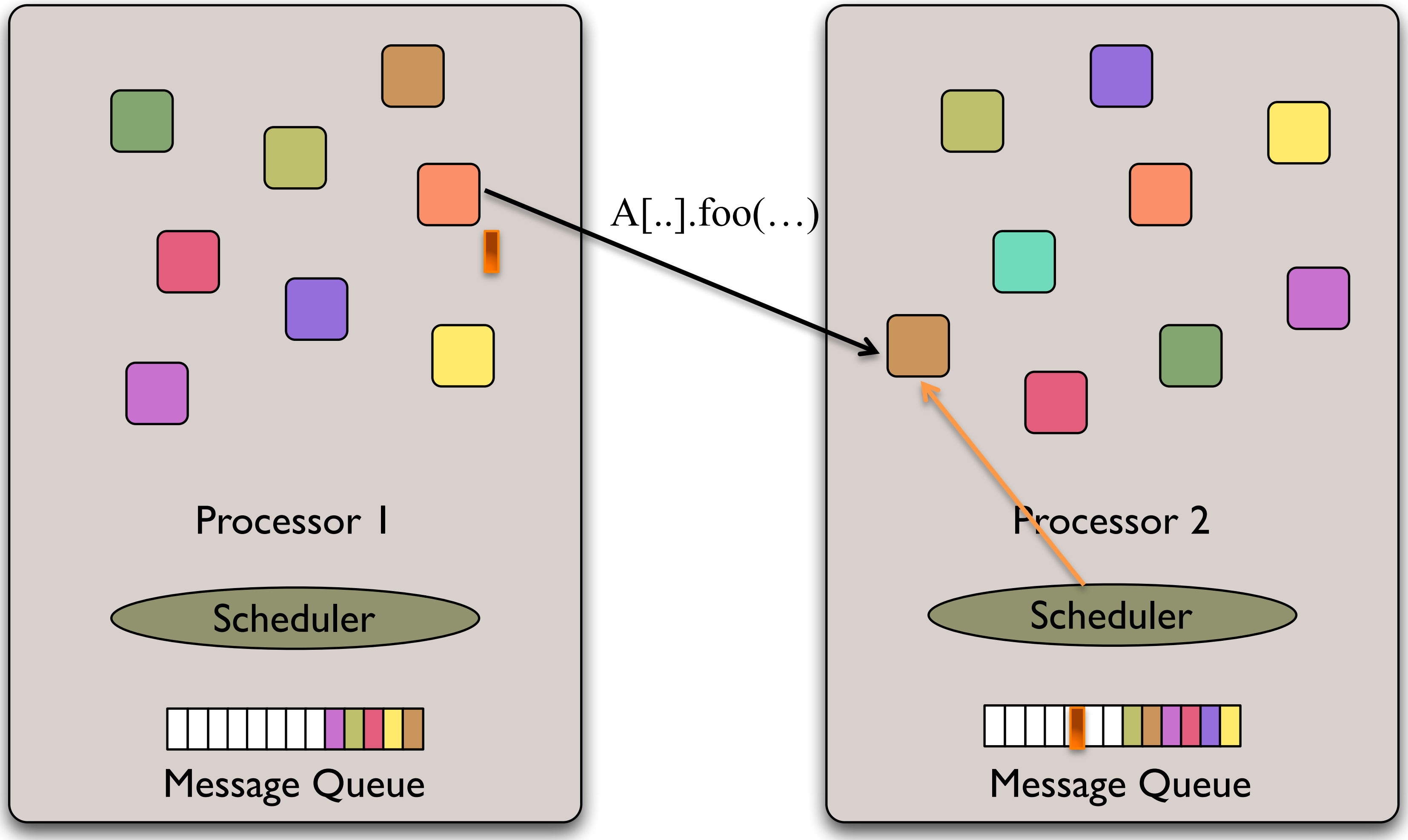
- You have multiple units on each processor
- They address each other via logical names
- Message-driven execution:
  - Let the work-unit that happens to have data ("message") available for it execute next
  - Let the RTS select among ready work units
  - Programmer should not specify what executes next, but can influence it via priorities

Charm++: Object-based overdecomposition

- Multiple "indexed collections" of C++ objects
- Indices can be multi-dimensional and/or sparse
- Programmer expresses communication between objects
  - with no reference to processors : A[i].foo(…)



Message-driven Execution



Empowering the RTS

Adaptive
Runtime System

Introspection

Adaptivity

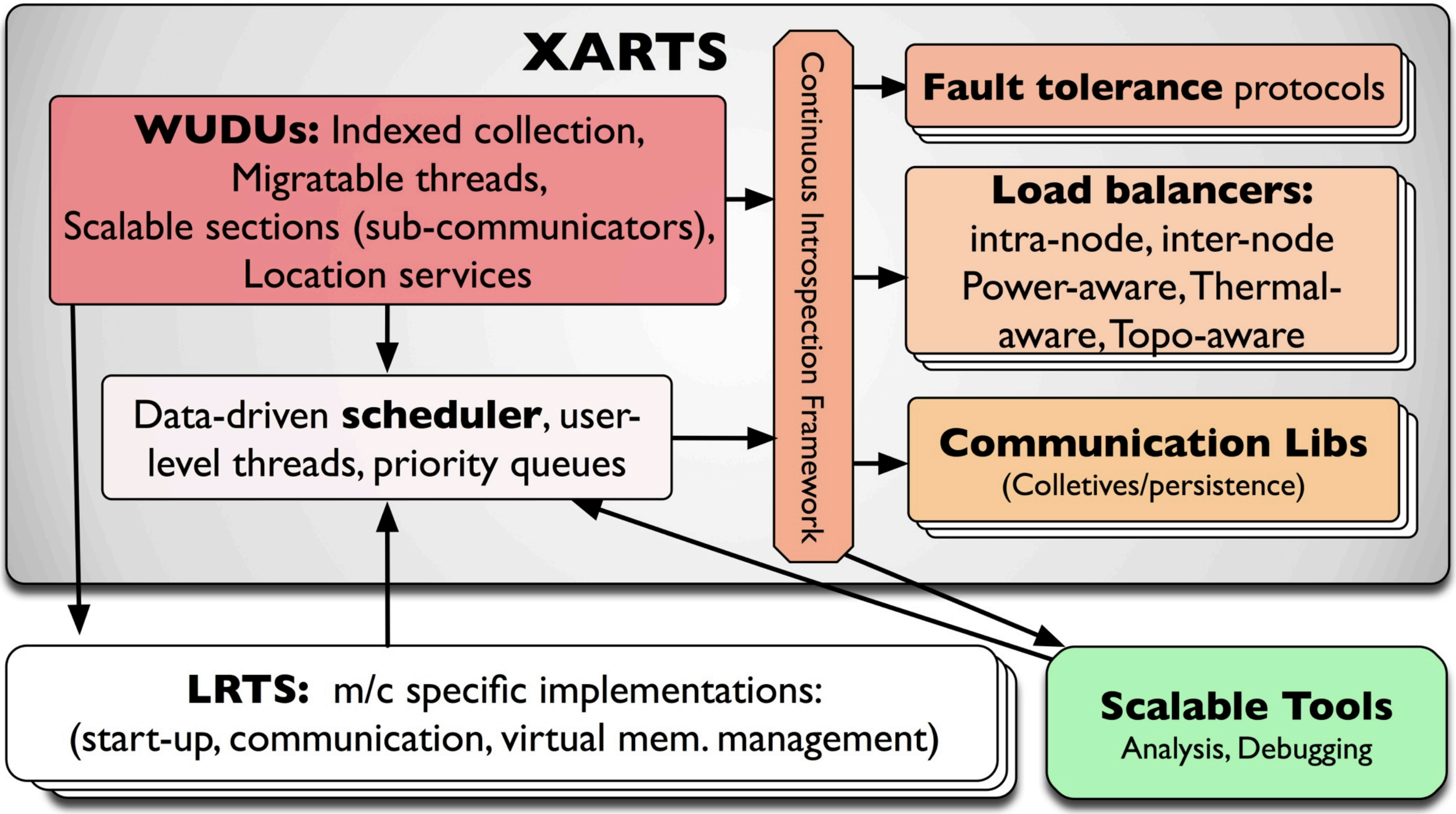
Asynchrony

Overdecomposition

Migratability

- The Adaptive RTS can:
  - Dynamically balance loads
  - Optimize communication:
    - Spread over time, async collectives
  - Automatic latency tolerance
  - Prefetch data with almost perfect predictability

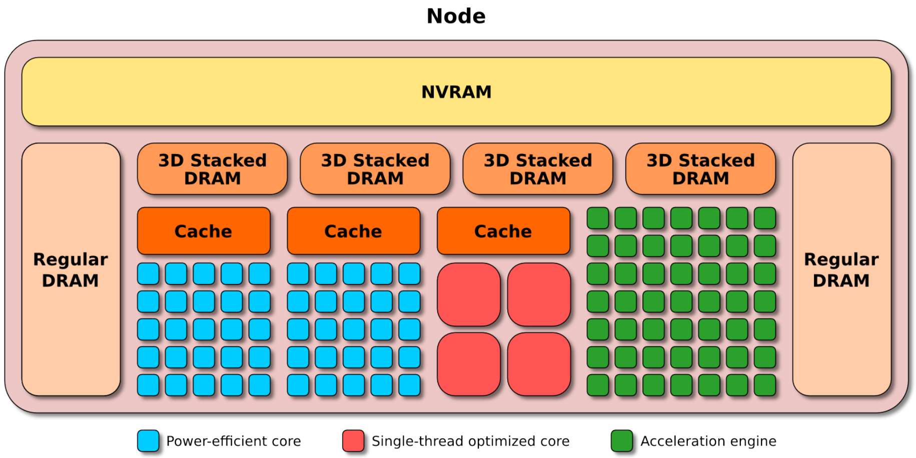
Charm++ RTS



XARTS
WUDUs: Indexed collection, Migratable threads, Scalable sections (sub-communicators), Location services
Data-driven **scheduler**, user-level threads, priority queues
Continuous Introspection Framework
Fault tolerance protocols
Load balancers: intra-node, inter-node Power-aware, Thermal-aware, Topo-aware
Communication Libs (Colletives/persistence)
LRTS: m/c specific implementations: (start-up, communication, virtual mem. management)
Scalable Tools Analysis, Debugging

ARGO

An Exascale Operating System and Runtime



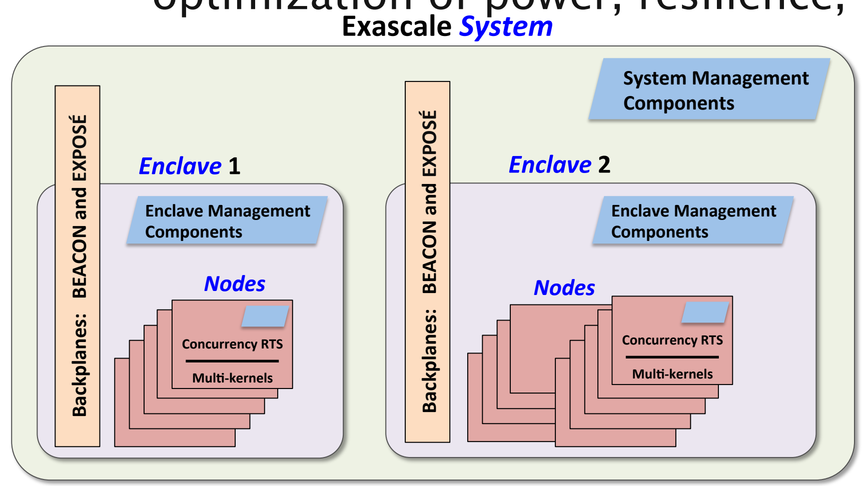
$9.7M ASCR DOE
3 year project, launched Aug 2013

THE CREW OF THE ARGO:

Argonne National Laboratory;
Principle Investigator and Chief Architect: Pete Beckman
Chief Scientist: Marc Snir
P. Balaji, R. Gupta, K. Iskra, R. Thakur, K. Yoshii, F. Cappello
Boston University: J. Appavoo, O. Krieger
Lawrence Livermore National Laboratory:
M. Gokhale, E. Leon, B. Rountree, M. Schulz, B. Van Essen
Pacific Northwest National Laboratory: S. Krishnamoorthy, R. Gioiosa
University of Chicago: H. Hoffmann
University of Illinois at UC: L. Kale, E. Bohm, R. Venkataraman
University of Oregon: A. Malony, S. Shende, K. Huck
University of Tennesee Knoxville: J. Dongarra, G. Bosilca

Key Areas of Innovation:

- NodeOS/R
  - Core-specialization permits multiple, concurrent kernels
- Lightweight Concurrency
  - Embed fine-grained tasks and lightweight threads into OS for massive parallelism
- Backplane
  - Event, Control, and Performance backplanes to support global optimizations
- Global View
  - "Enclave" abstraction to allow global optimization of power, resilience, perf.



PPL UIUC

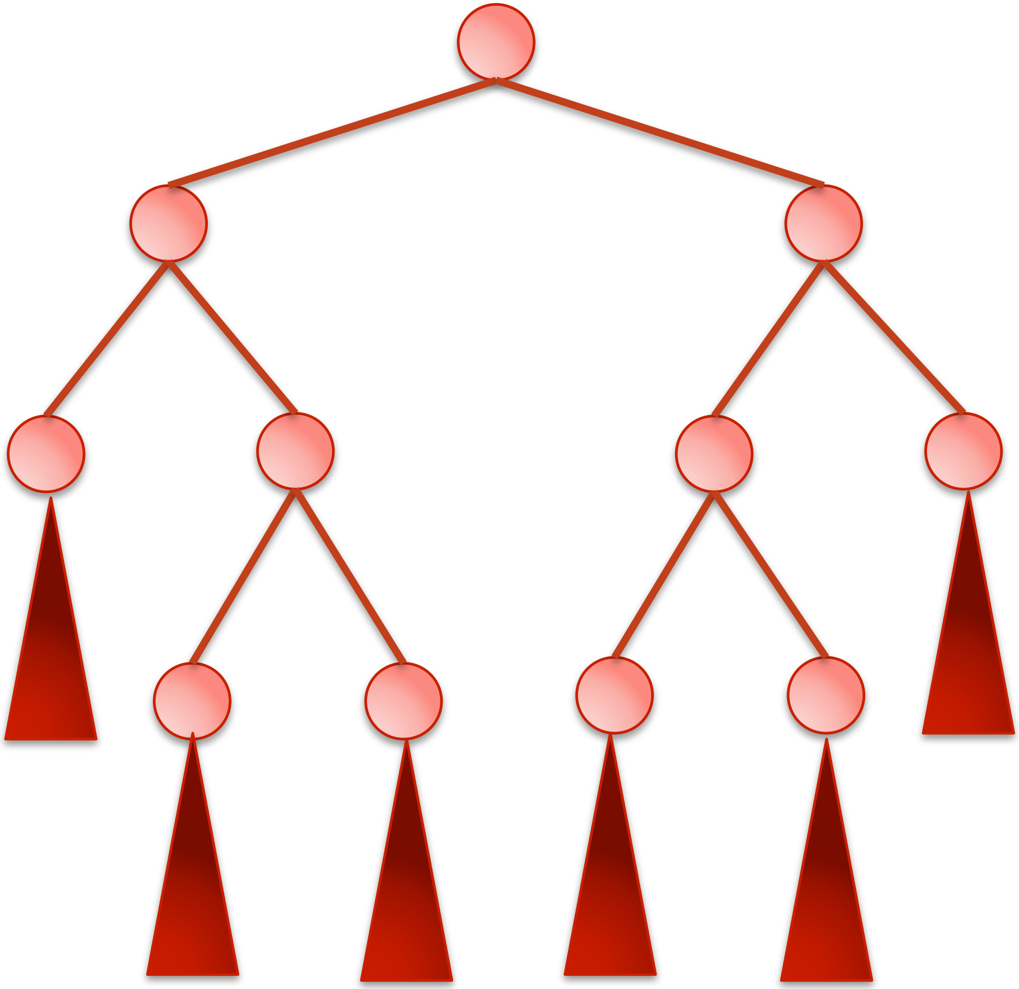
ChaNGa: Parallel Gravity

- Collaborative project (NSF)
  - with Tom Quinn, Univ. of Washington
- Gravity, gas dynamics
- Barnes-Hut tree codes
  - Oct tree is natural decomp
  - Geometry has better aspect ratios, so you "open" up fewer nodes
  - But is not used because it leads to bad load balance
  - Assumption: one-to-one map between sub-trees and PEs
  - Binary trees are considered better load balanced

Evolution of Universe and Galaxy Formation

With Charm++: Use Oct-Tree, and let Charm++ map subtrees to processors

ChaNGa: Cosmology Simulation



Collaboration with
Tom Quinn UW

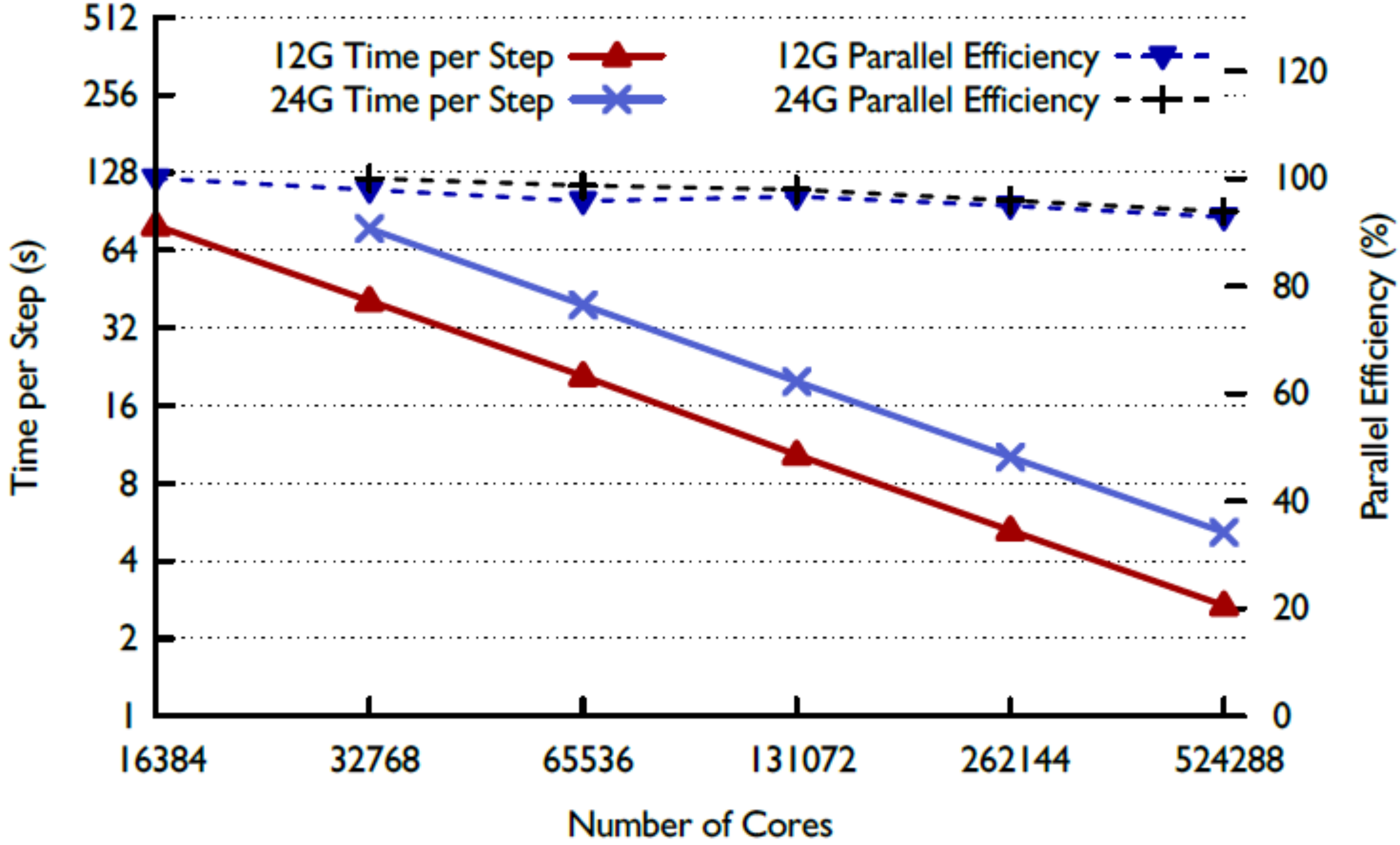
- Tree: Represents particle distribution
- TreePiece: object/ chares containing particles

PPL
UIUC

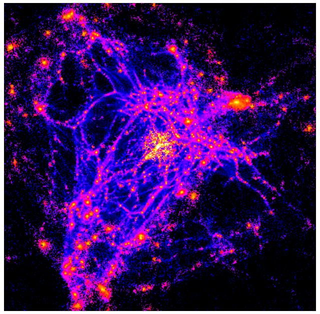
ChaNGa: Optimized Performance

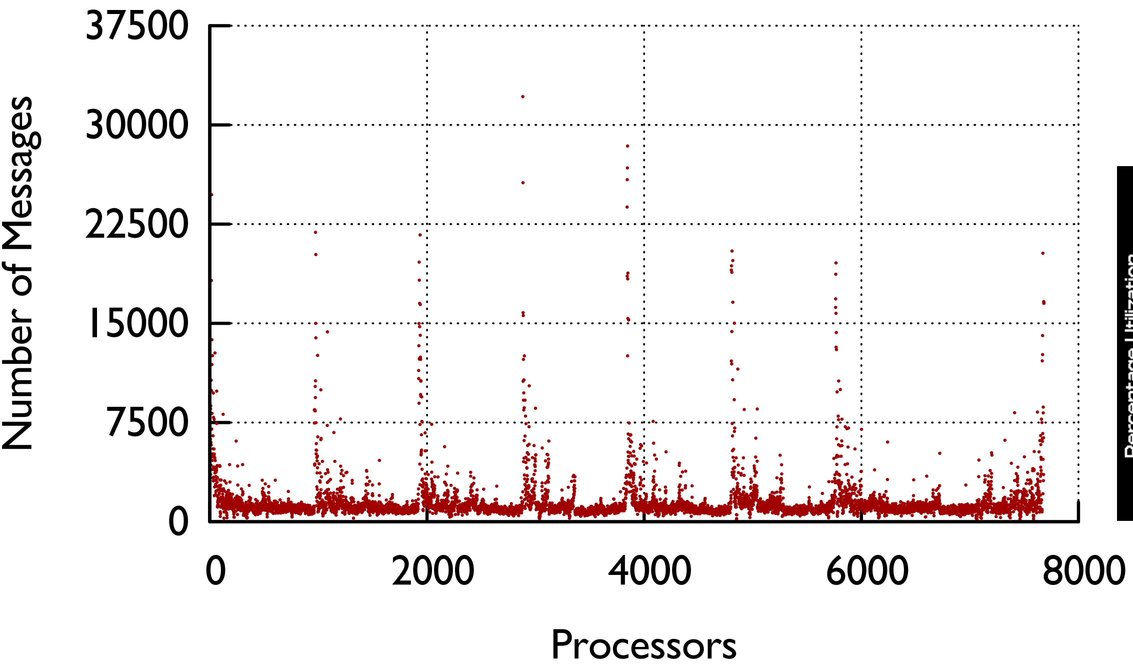
- Asynchronous, highly overlapped, phases
- Requests for remote data overlapped with local computations

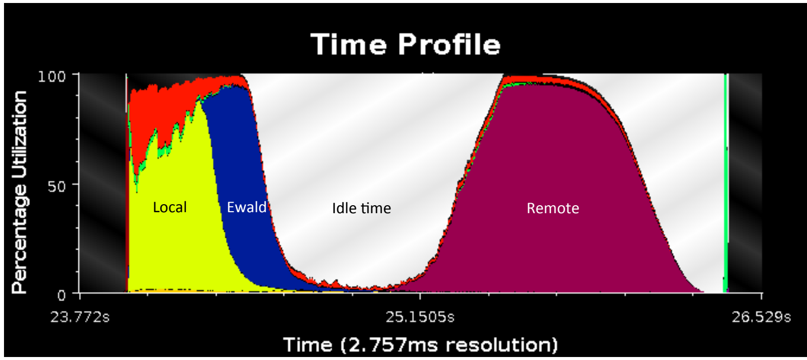
ChaNGa : a recent result



Clustered Dataset – Dwarf





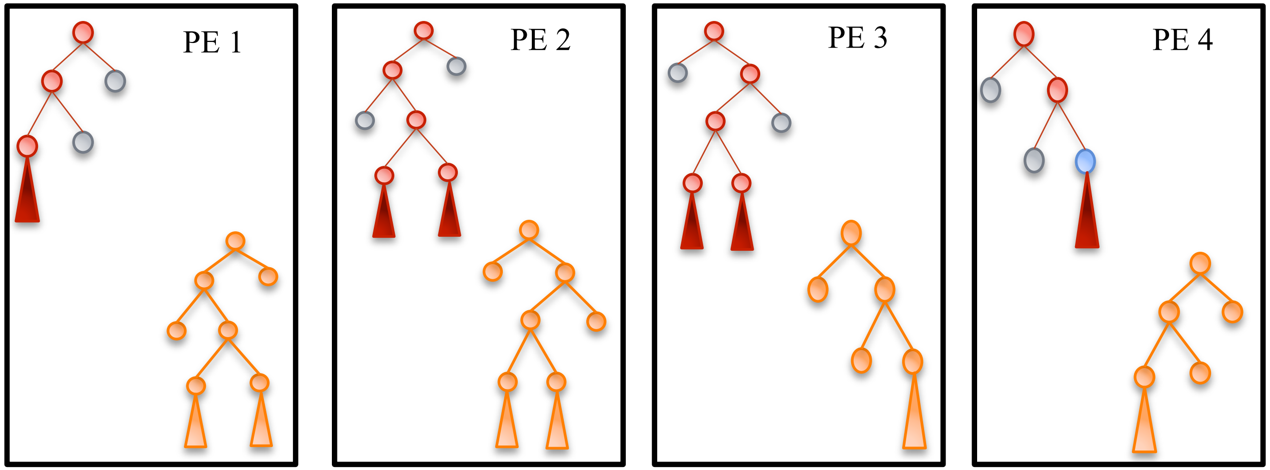


- Highly clustered
- Maximum request per processor: > 30K

- Idle time due to message delays
- Also, load imbalances: solved by Hierarchical balancers

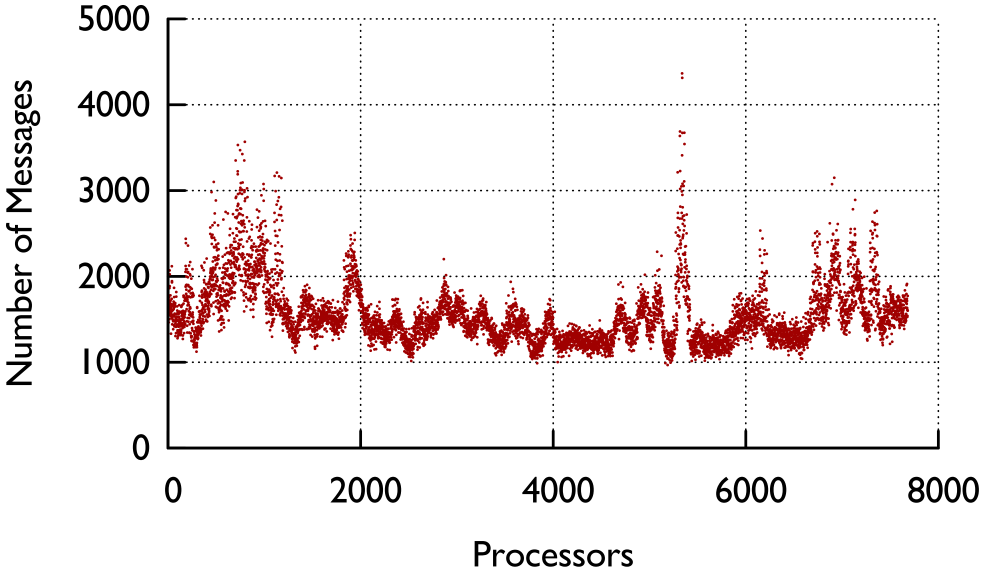
PPL UIUC

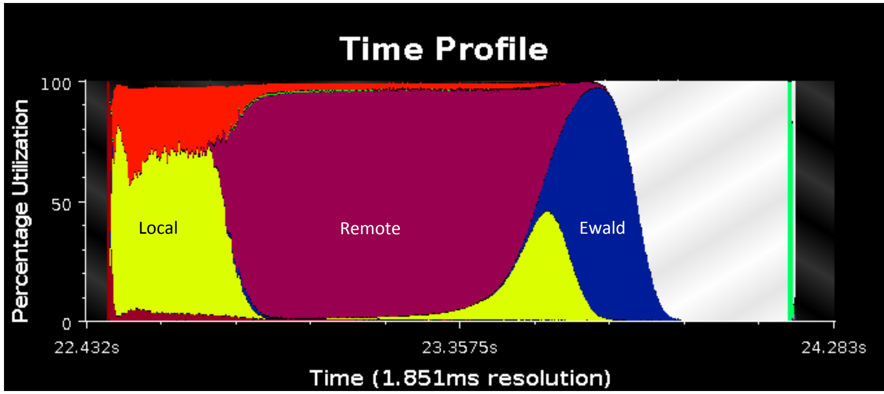
Solution: Replication

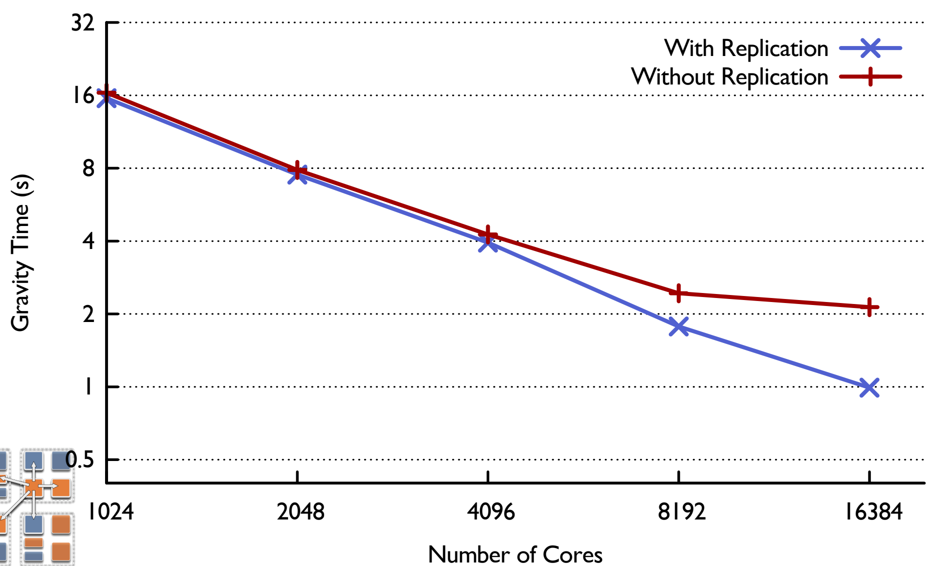


- Replicate tree nodes to distribute requests
- Requester randomly selects a replica

Replication Impact







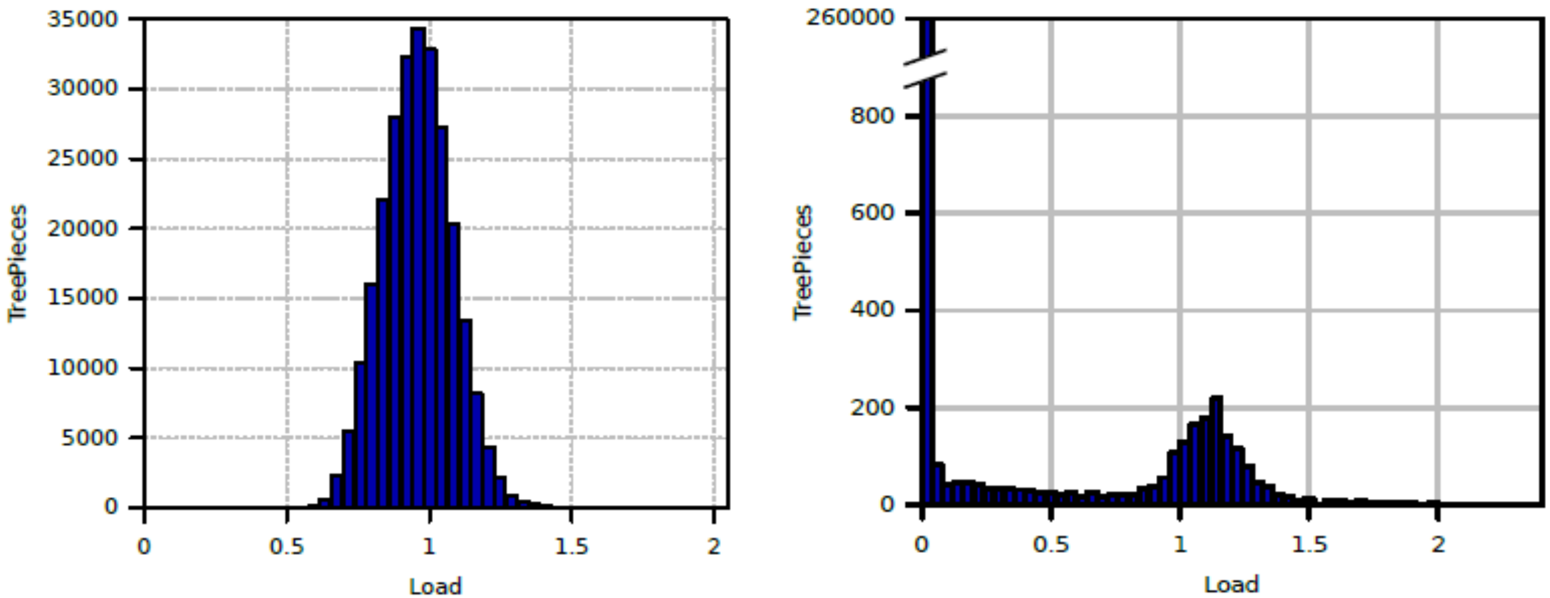
- Replication distributes requests
- Maximum request reduced from 30K to 4.5K
- Gravity time reduced from 2.4 s to 1.7 s, on 8k

Multiple time-stepping!

- Our scientist collaborators suggest an algorithmic optimization:
  - Don't move slow-moving particles every step
    - i.e. don't calculate forces on them either
  - In fact, make many (say 5) categories (rungs) of particles based on their velocities
  - Rung sequence (with 5 rungs)
    - 4 3 4 2 4 3 4 1 4 3 4 2 4 3 4 0
    - Rung 0: all particles, Rung 4: fastest-moving particles
  - Each tree-piece object now presents a different load when different "rungs" are being calculated

Multiple time-stepping!

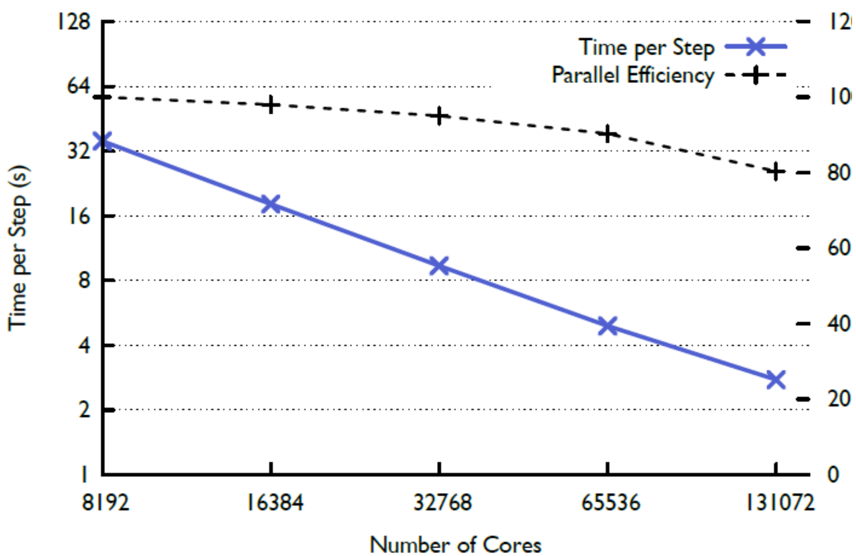
- Load (for the same object) changes across rungs
  - Yet, there is persistence within the same rung!
  - So, specialized phase-aware balancers were developed



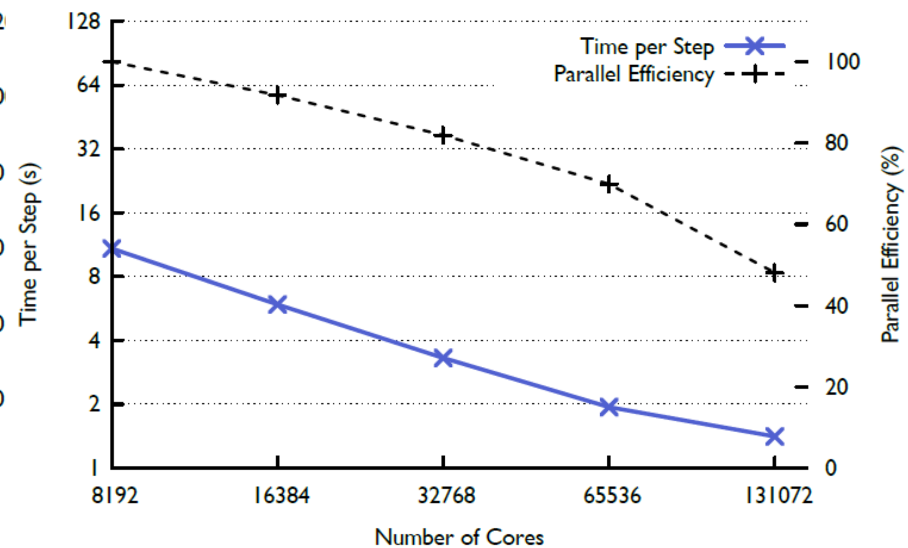
(a) Rung 0 (b) Rung 4

Multi-stepping tradeoff

- Parallel efficiency is lower, but performance is improved significantly



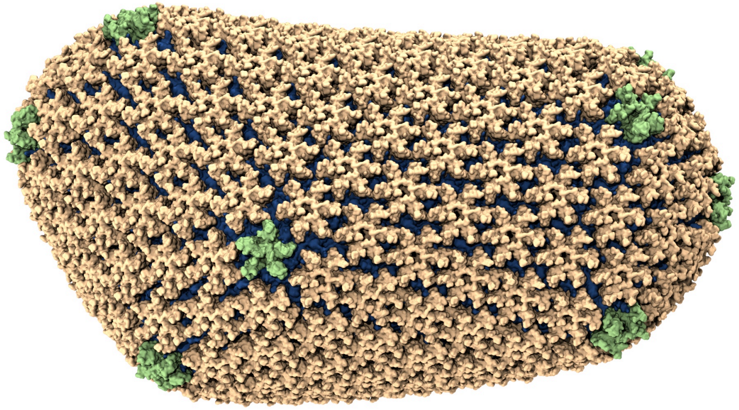
Single Stepping



Multi Stepping

NAMD: Biomolecular Simulations

- Collaboration with K. Schulten
- With over 50,000 registered users
- Scaled to most top US supercomputers
- In production use on supercomputers and clusters and desktops
- Gordon Bell award in 2002

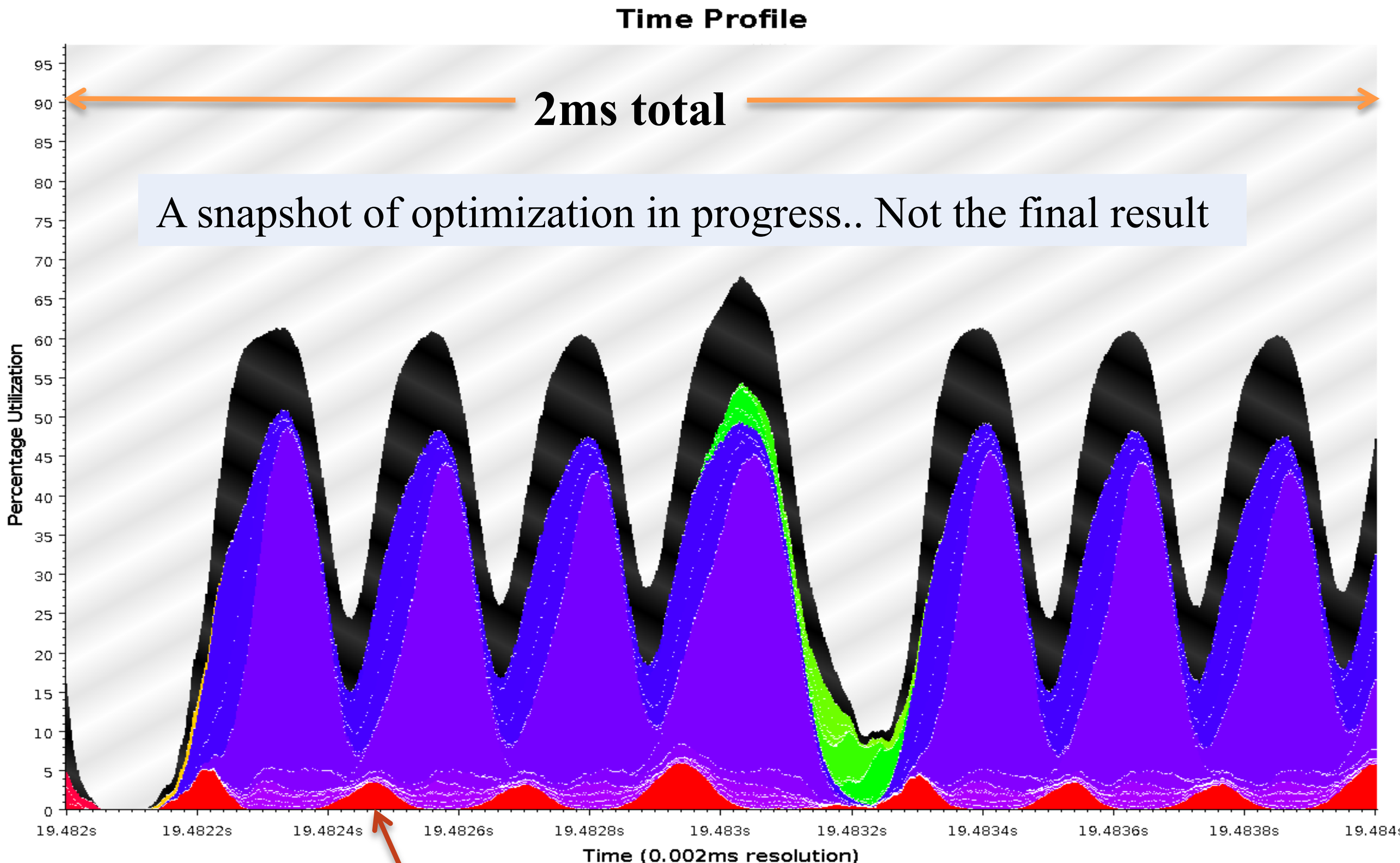


Recent success: Determination of the structure of HIV capsid by researchers including Prof Schulten

PPL UIUC

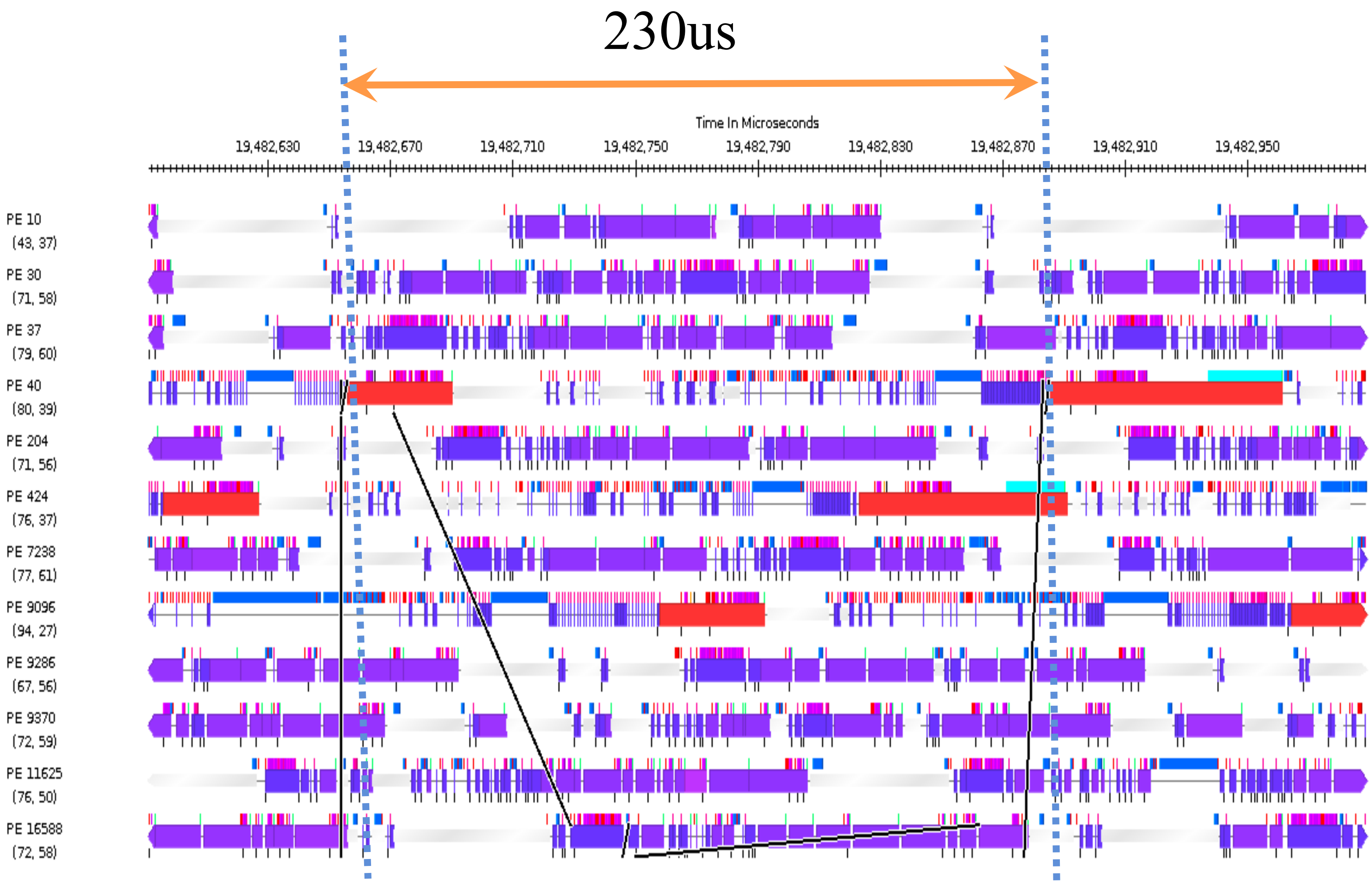
Time Profile of ApoA1 on Power7 PERCS

92,000 atom system, on 500+ nodes (16k cores)



Overlapped steps, as a result of asynchrony

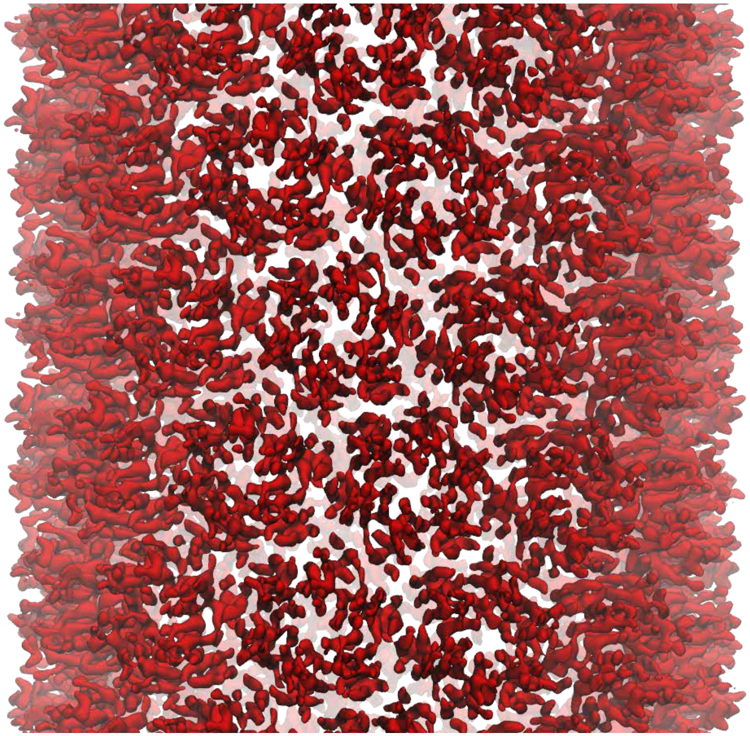
Timeline of ApoA1 on Power7 PERCS



PPL
UIUC

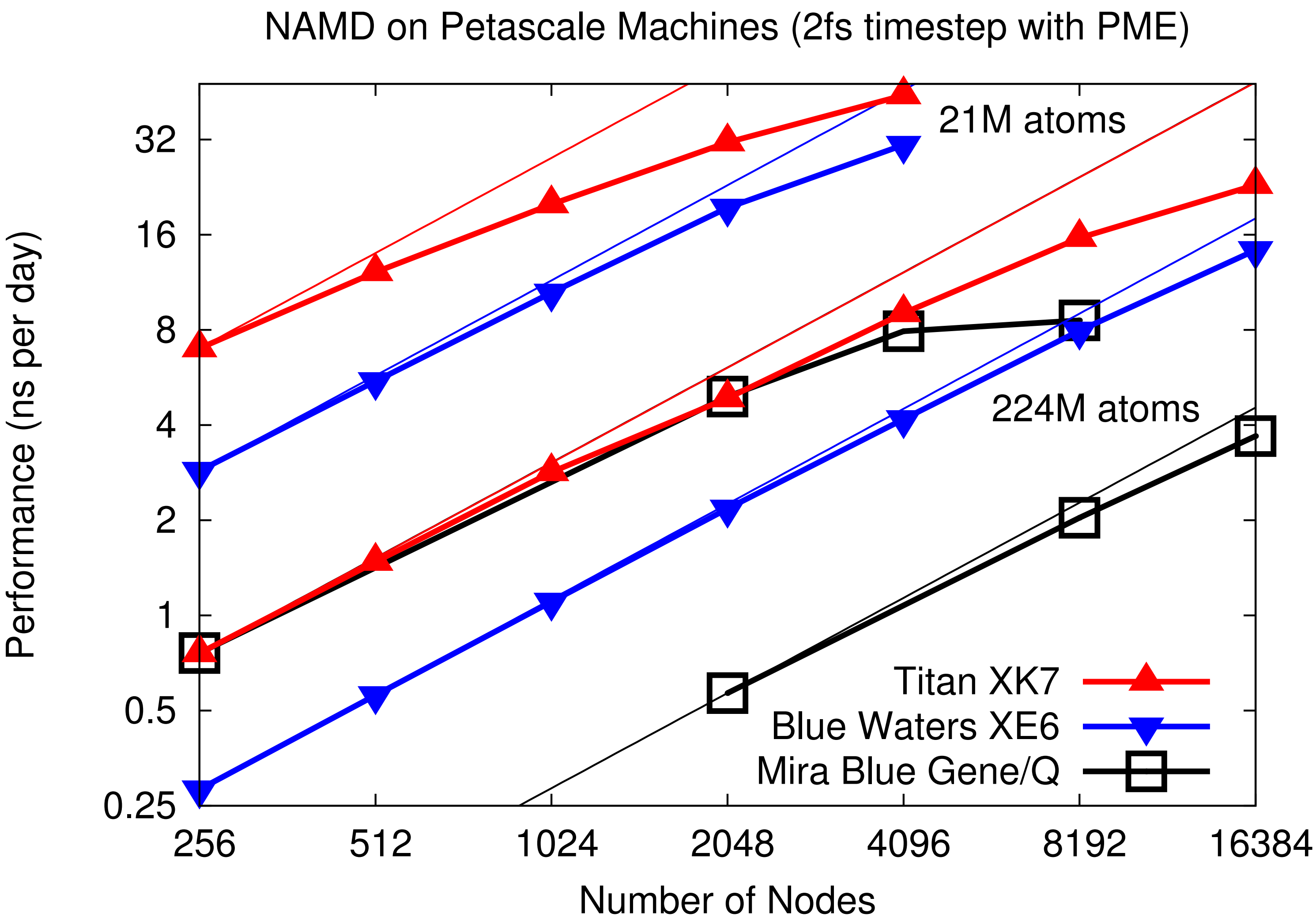
NAMD: Strong Scaling

- HIV Capsid was a 64 million atom simulation, including explicit water atoms
- Most biophysics systems of interests are 10M atoms or less… maybe 100M
- Strong scaling desired to billions of steps



Enhancing Asynchrony in NAMD

- Charm++ reductions are non-blocking
  - So, you *can* do other work while reduction is progressing through the system
- Synchronization:
  - NAMD, when used with a barostat (NPT ensemble), needs pressure from the current step to rescale volume
  - So, no other work was performed during reduction
- Enhancing asynchrony:
  - For strong scaling, the algebra was reworked to use the results of the reduction one step later
  - Overlapped reduction with an entire force computation step
  - 10% performance improvement on 16k <u>nodes</u> on Titan

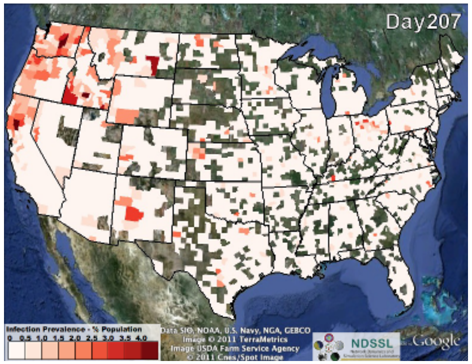


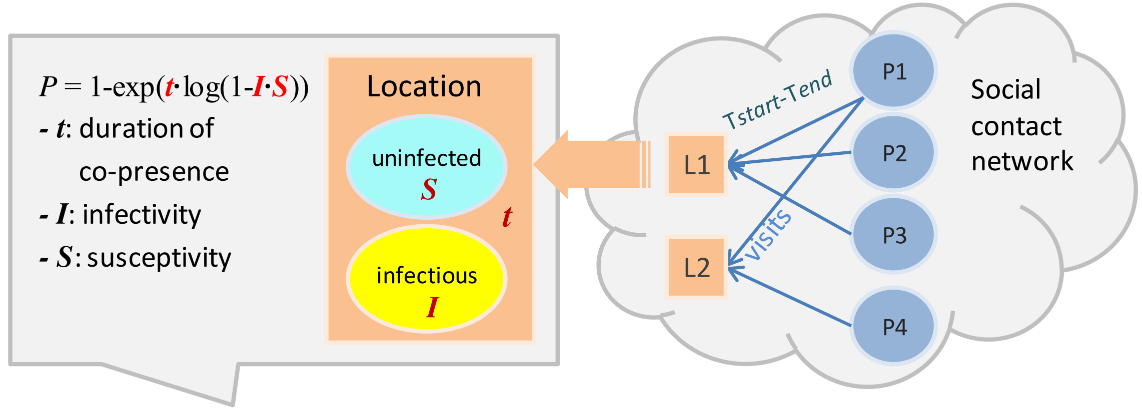
NAMD strong scaling on Titan Cray XK7, Blue Waters Cray XE6, and Mira IBM Blue Gene/Q for 21M and 224M atom benchmarks

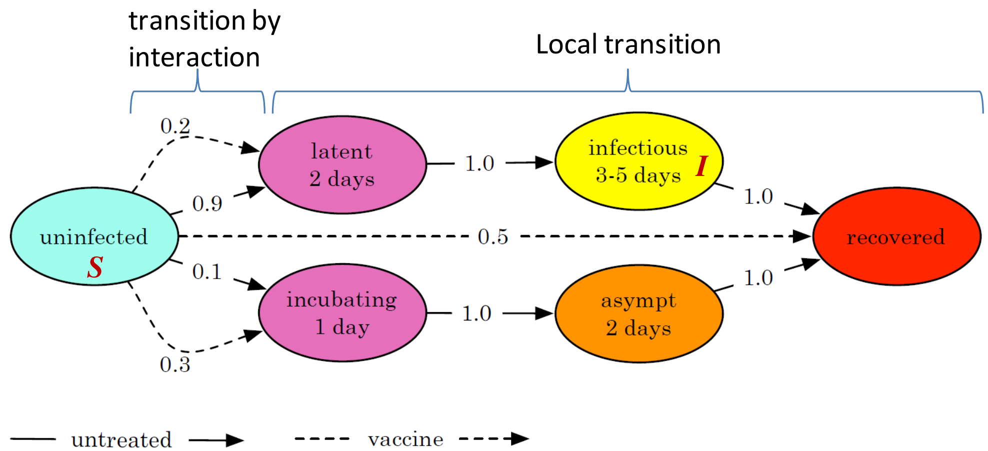
Episimdemics

- Simulation of spread of contagion
  - Code by Madhav Marathe, Keith Bisset, .. Vtech
  - Original was in MPI
- Converted to Charm++
  - Benefits: asynchronous reductions improved performance considerably

Simulating contagion over dynamic networks







EpiSimdemics[1]

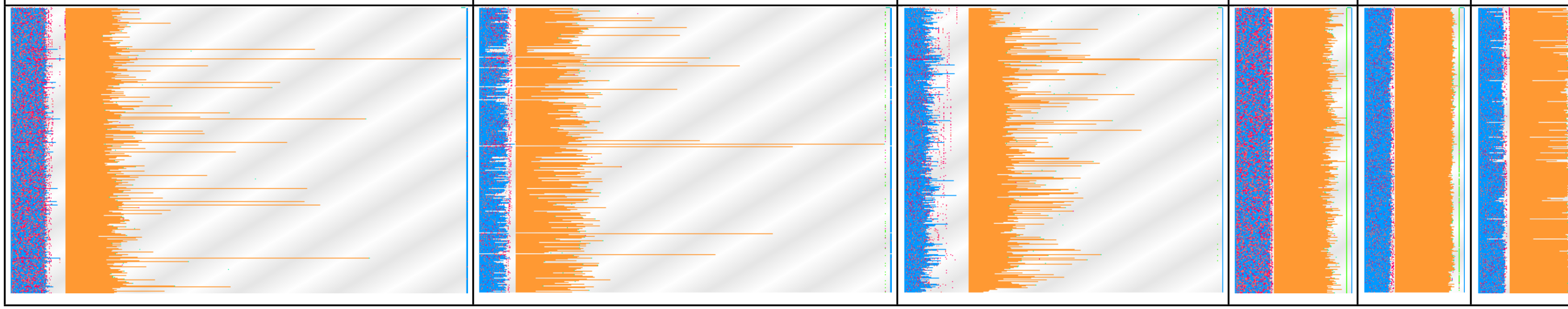
- Agent-based
- Realistic population data
- Intervention[2]
- Co-evolving network, behavior and policy[2]

[1] C. Barrett et al.,"EpiSimdemics: An Efficient Algorithm for Simulating the Spread of Infectious Disease over Large Realistic Social Networks," SC08

[2] K. Bisset et al., "Modeling Interaction Between Individuals, Social Networks and Public Policy to Support Public Health Epidemiology," WSC09.

Load distribution (Vulcan)

RR (1.755 s)	GP (1.583 s)	Z splitLoc (1.222 s)	RR splitLoc (0.438 s)	ZC splitLoc (0.369 s)	GP splitLoc (0.368 s)



splitLoc: **no peak in location computation**
Z-splitLoc: **no load balance**

GP: **shorter person phase**
ZC-splitLoc: similar perf. w/ GP-splitLoc

- Blue: person computation
- Red: receiver's msg handling
- Orange: location computation

X-axis: **Time** Y-axis: **Processor**

Timeline of an iteration from sampled subset of 332 cores of total 4K using Michigan data on Vulcan

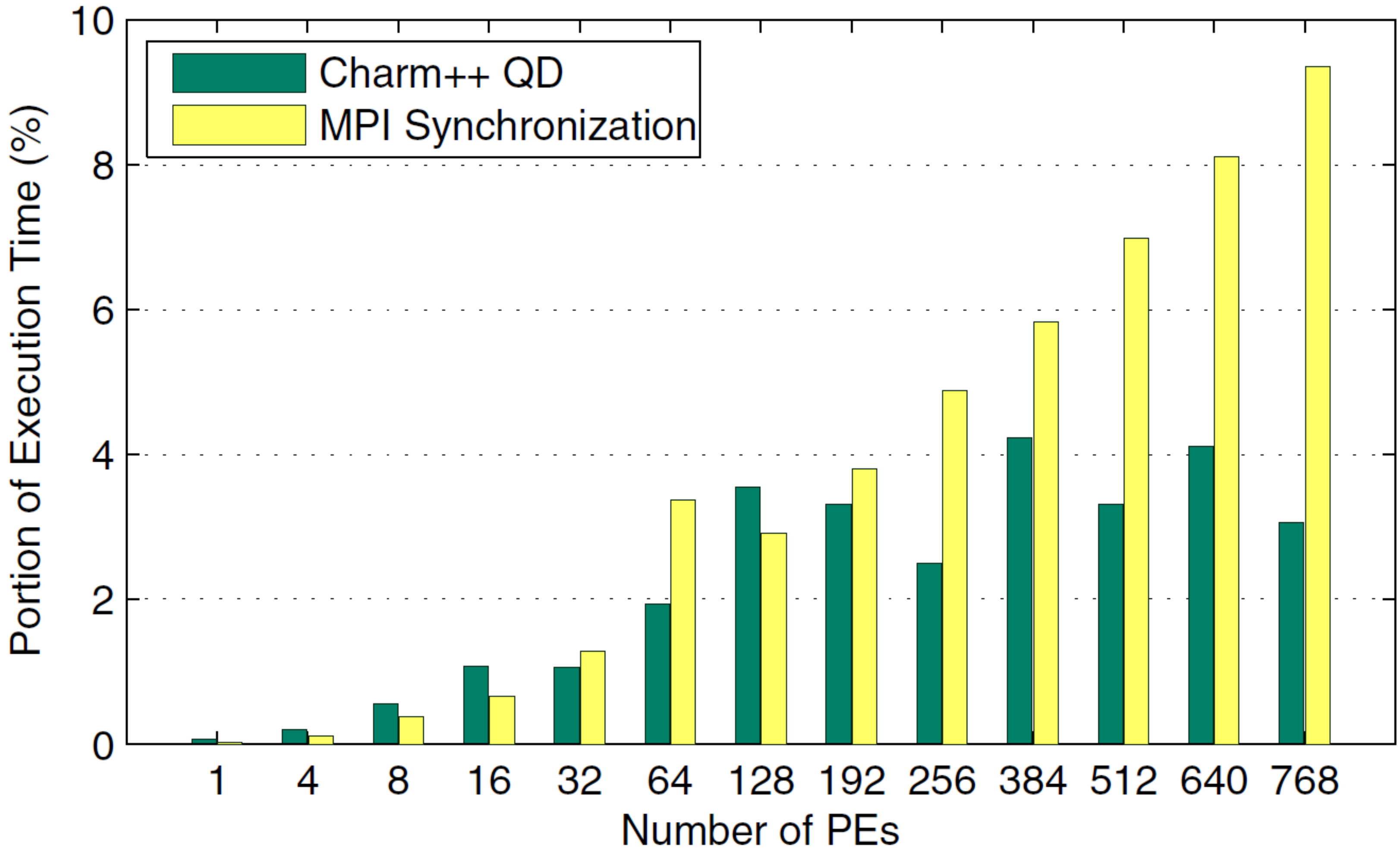
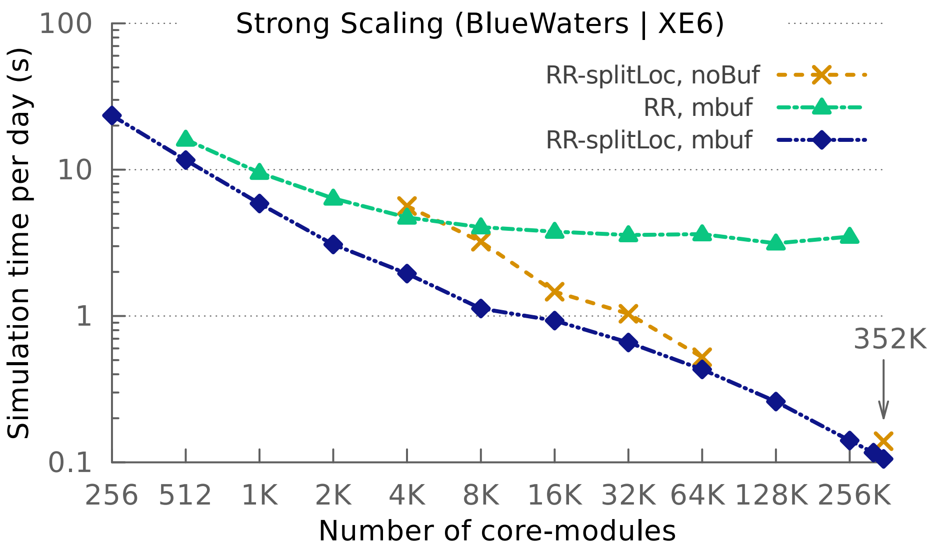
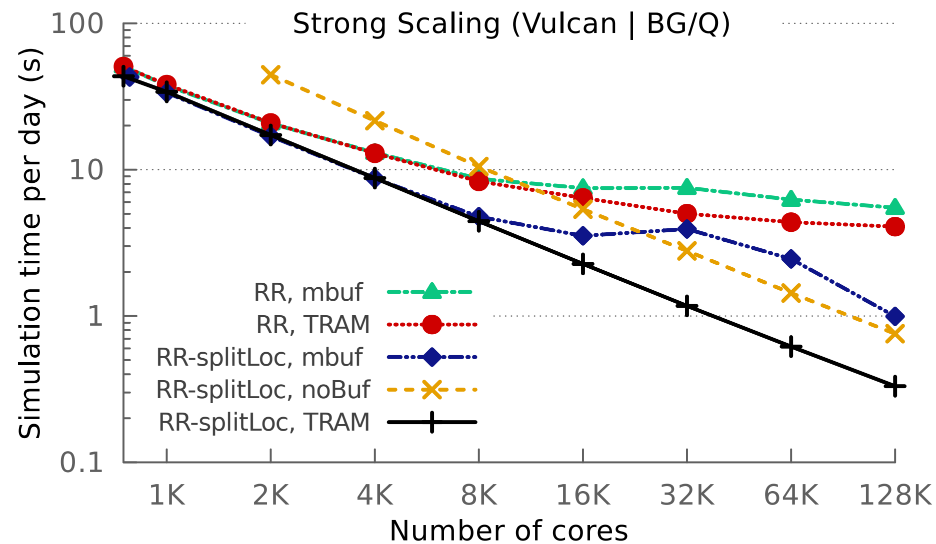
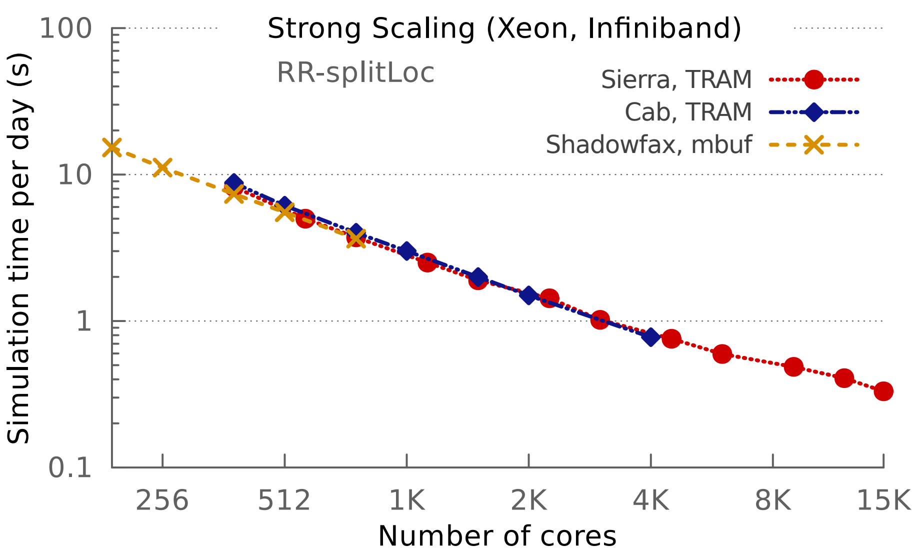


Figure 10. The synchronization cost using contribute() and QD method takes at most 4.23% of the total execution time while the MPI synchronization cost linearly increases up to 14.5% as the number of PEs used increases for simulating Arkansas population.

Strong scaling performance with the largest data set







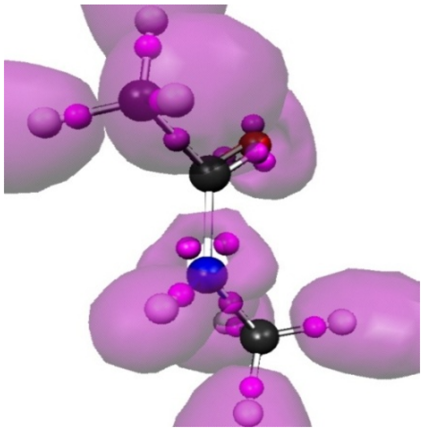
- Contiguous US population data
- **XE6**: **the largest scale** (**352K cores**)
- **BG/Q**: good scaling up to 128K cores
- Strong scaling helps timely reaction to pandemic

OpenAtom

Car-Parinello Molecular Dynamics

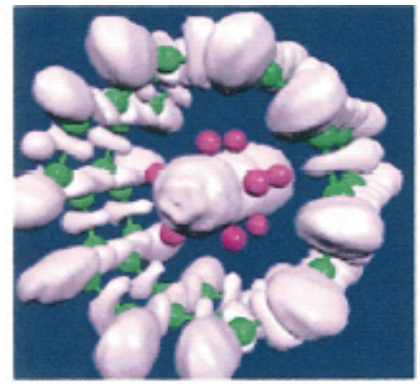
NSF ITR 2001-2007, IBM, DOE,NSF

Molecular Clusters :

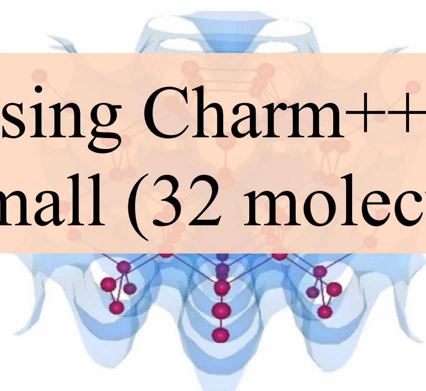


Recent NSF SSI-SI2 grant
With
G. Martyna (IBM)
Sohrab Ismail-Beigi

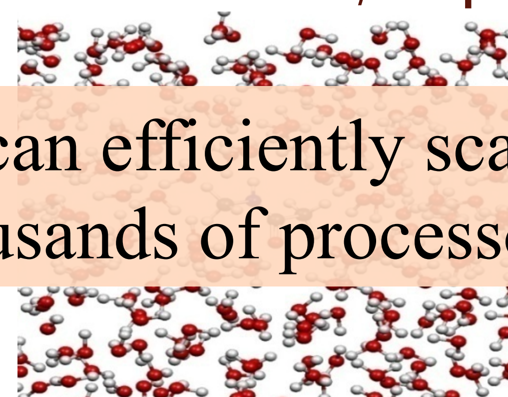
Nanowires:



Semiconductor Surfaces:

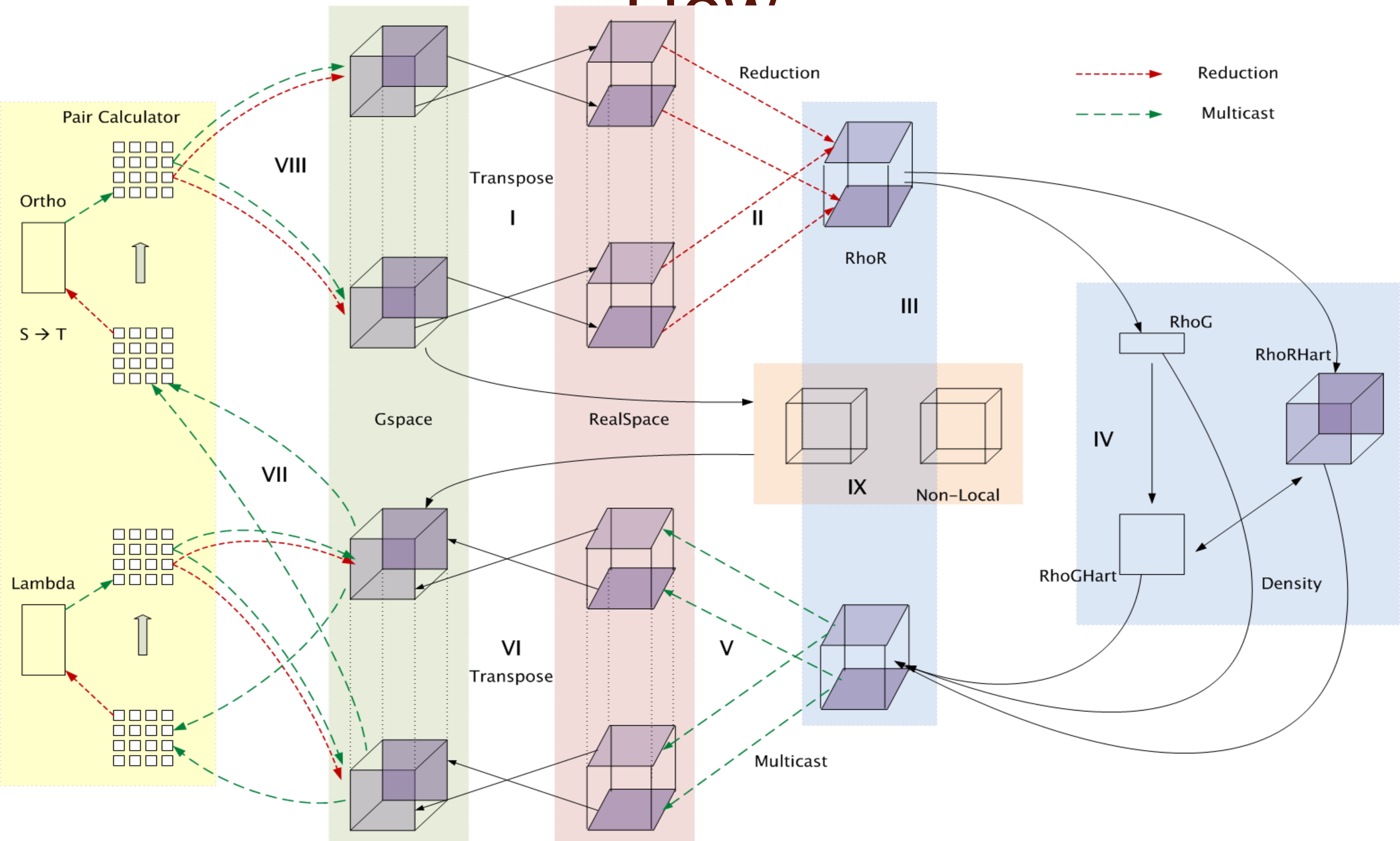


3D-Solids/Liquids:

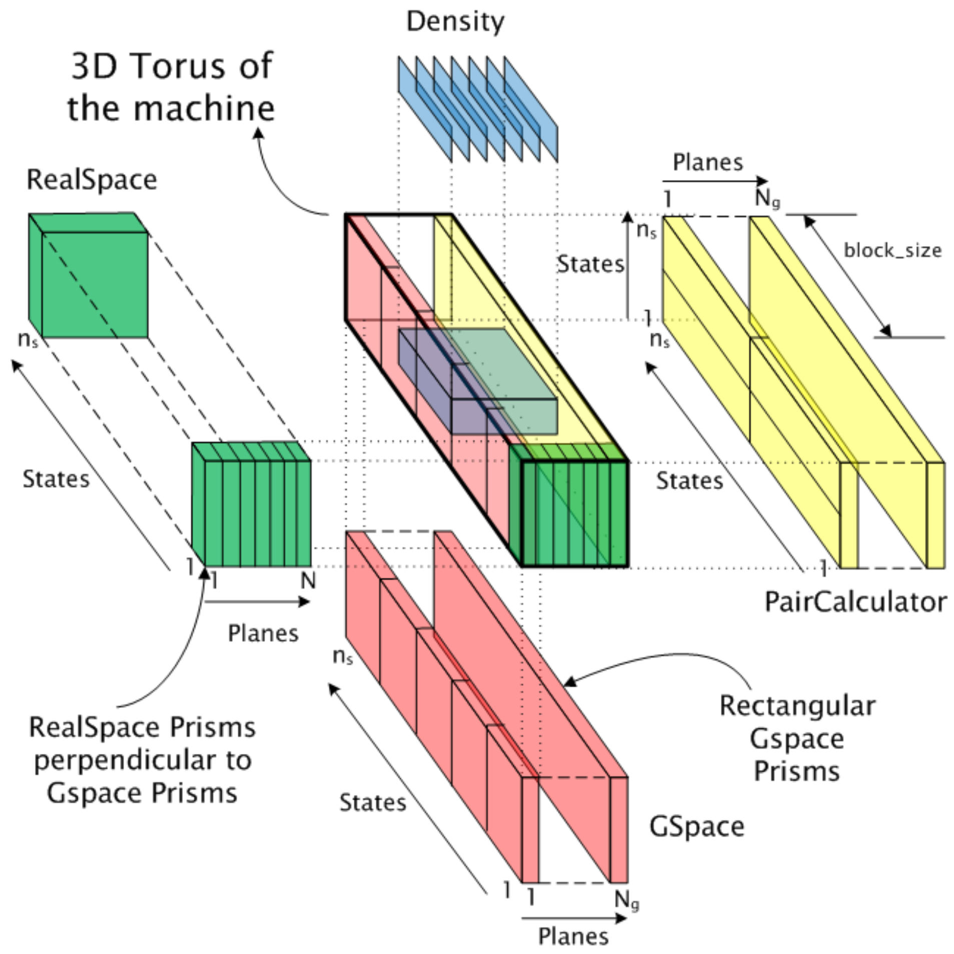


Using Charm++ virtualization, we can efficiently scale small (32 molecule) systems to thousands of processors

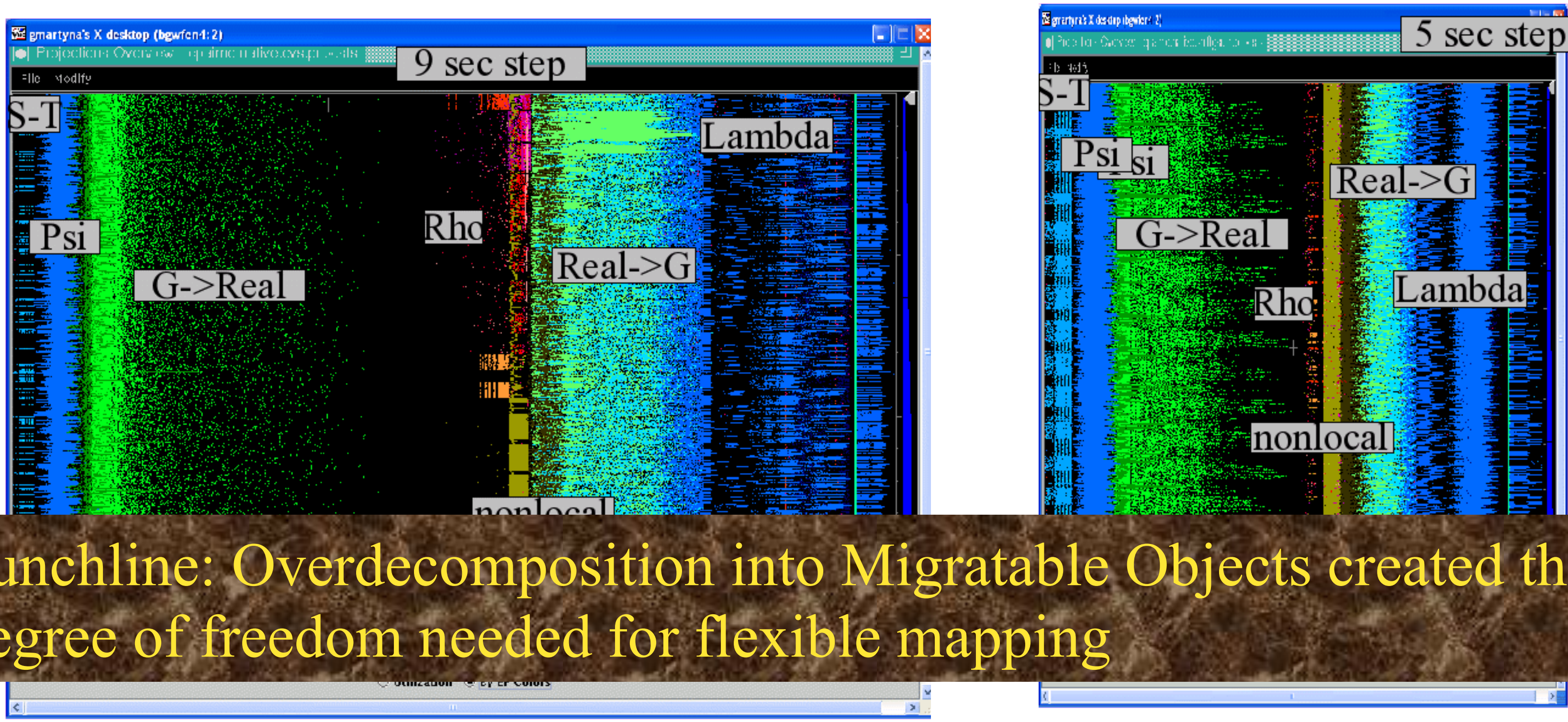
Decomposition and Computation Flow



Topology Aware Mapping of Objects



Improvements by topological aware mapping of computation to processors

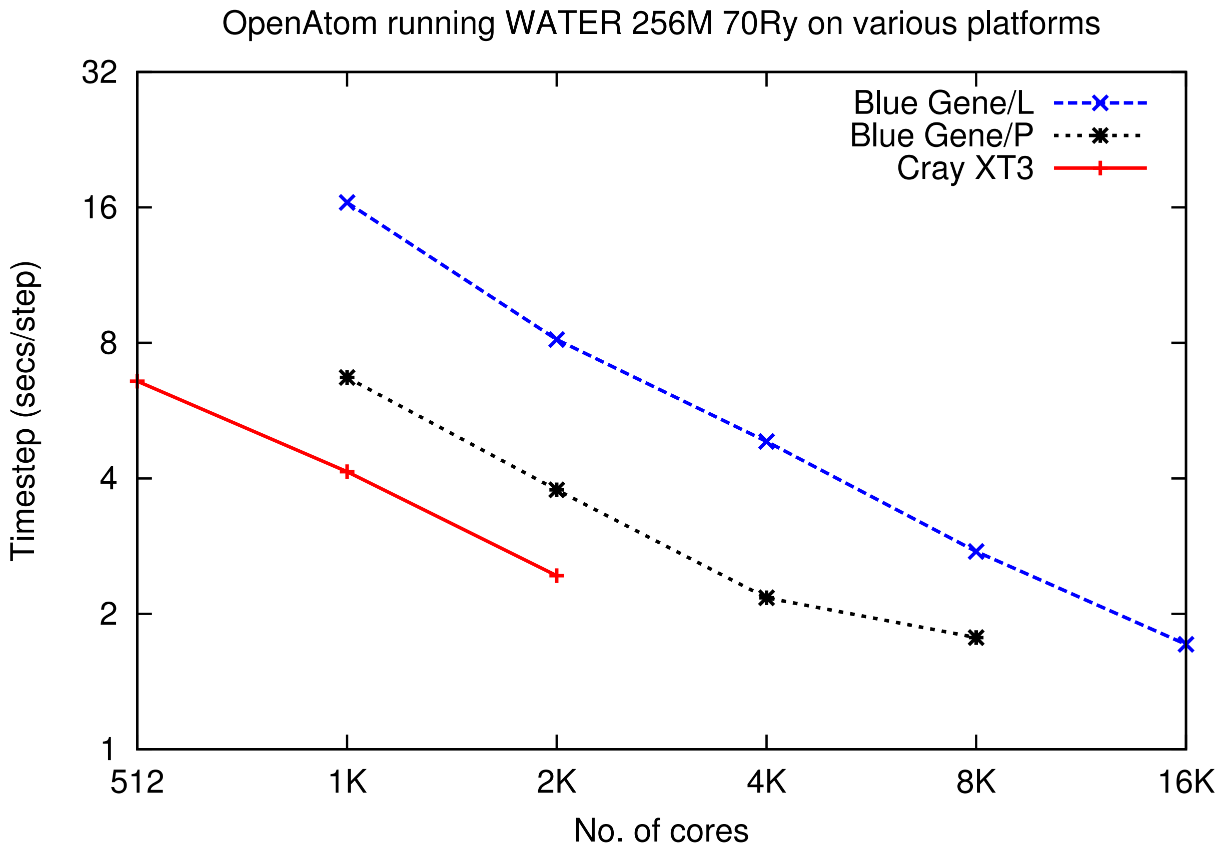


Punchline: Overdecomposition into Migratable Objects created the degree of freedom needed for flexible mapping

The simulation of the left panel, maps computational work to processors taking the network connectivity into account while the right panel simulation does not. The "black" or idle time processors spent waiting for computational work to arrive on processors is significantly reduced at left. (256waters, 70R, on BG/L 4096 cores)

OpenAtom Performance Sampler

Ongoing work on:
K-points



MiniApps

Mini-App	Features	Machine	Max cores
AMR	Overdecomposition, Custom array index, Message priorities, Load Balancing, Checkpoint restart	BG/Q	131,072
LeanMD	Overdecomposition, Load Balancing, Checkpoint restart, Power awareness	BG/P BG/Q	131,072 32,768
Barnes-Hut (n-body)	Overdecomposition, Message priorities, Load Balancing	Blue Waters	16,384
LULESH 2.02	AMPI, Over-decomposition, Load Balancing	Hopper	8,000
PDES	Overdecomposition, Message priorities, TRAM	Stampede	4,096

More MiniApps

Mini-App	Features	Machine	Max cores
1D FFT	Interoperable with MPI	BG/P BG/Q	65,536 16,384
Random Access	TRAM	BG/P BG/Q	131,072 16,384
Dense LU	SDAG	XT5	8,192
Sparse Triangular Solver	SDAG	BG/P	512
GTC	SDAG	BG/Q	1,024
SPH		Blue Waters	-

PPL UIUC

Where are Exascale Issues?

- I didn't bring up exascale at all so far..
  - Overdecomposition, migratability, asynchrony were needed on yesterday's machines too
  - And the app community has been using them
  - But:
    - On *some* of the applications, and maybe without a common general-purpose RTS
- The same concepts help at exascale
  - Not just help, they are necessary, and adequate
  - As long as the RTS capabilities are improved
- We have to apply overdecomposition to all (most) apps

PPL UIUC

A message of this talk

Intelligent, introspective, Adaptive Runtime Systems, developed for handling application's dynamic variability, already have features that can deal with challenges posed by exascale hardware

Fault Tolerance in Charm++/AMPI

- Four approaches available:
  - Disk-based checkpoint/restart
  - In-memory double checkpoint w auto. restart
  - Proactive object migration
  - Message-logging: scalable fault tolerance
- Common Features:
  - Easy checkpoint: migrate-to-disk
  - Based on dynamic runtime capabilities
  - Use of object-migration
  - Can be used in concert with load-balancing schemes

Demo at Tech Marketplace

Extensions to fault recovery

- Based on the same over-decomposition ideas
  - Use NVRAM instead of DRAM for checkpoints
    - Non-blocking variants
    - [Cluster 2012] Xiang Ni et al.
  - Replica-based soft-and-hard-error handling
    - As a “gold-standard” to optimize against
    - [SC 13] Xiang Ni, E. Meneses, N. Jain, et al.

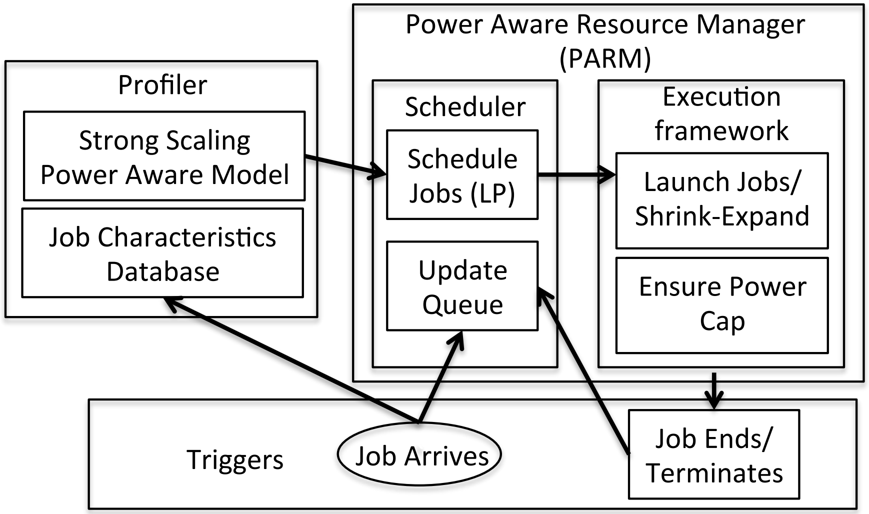
Saving Cooling Energy

Demo at Tech Marketplace

- Easy: increase A/C setting
  - But: some cores may get too hot
- So, reduce frequency if temperature is high (DVFS)
  - Independently for each chip
- *But,* this creates a load imbalance!
- No problem, we can handle that:
  - Migrate objects away from the slowed-down processors
  - Balance load using an existing strategy
  - Strategies take speed of processors into account
- Implemented in experimental version
  - SC 2011 paper, IEEE TC paper
- Several new power/energy-related strategies
  - PASA '12: Exploiting differential sensitivities of code segments to frequency change

PARM:Power Aware Resource Manager

- Charm++ RTS facilitates malleable jobs
- PARM can improve throughput under a fixed power budget using:
  - overprovisioning (adding more nodes than conventional data center)
  - RAPL (capping power consumption of nodes)
  - Job malleability and moldability



PPL UIUC

Summary

- Charm++ embodies an adaptive, introspective runtime system
- Many applications have been developed using it
  - NAMD, ChaNGa, Episimdemics, OpenAtom, …
  - Many miniApps, and third-party apps
- Adaptivity developed for apps is useful for addressing exascale challenges
  - Resilience, power/temperature optimizations, ..

More info on Charm++:
http://charm.cs.illinois.edu
Including the miniApps

Overdecomposition *Asynchrony* *Migratability*

A recently published book surveys seven major applications developed using Charm++

More info on Charm++:
http://charm.cs.illinois.edu
Including the miniApps