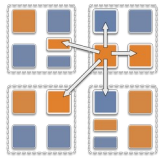
Charm++ Applications as case studies

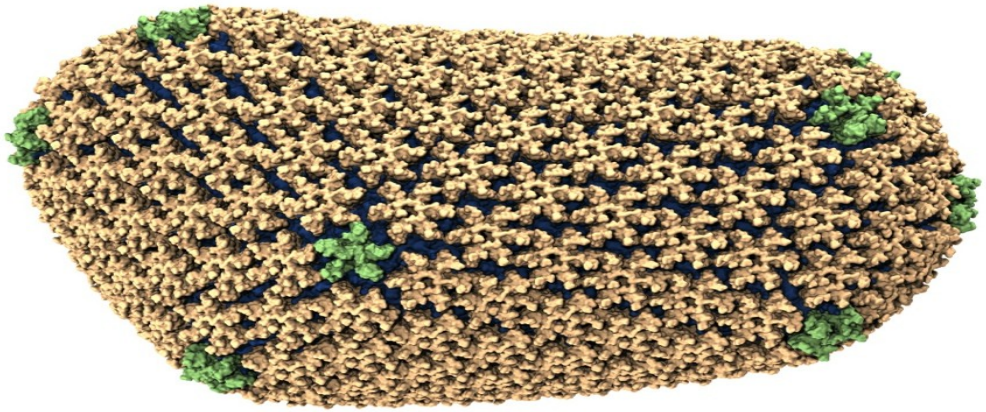
Only brief overview today



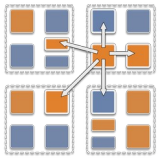
PPL UIUC

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

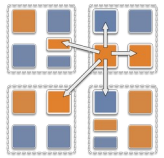


PPL UIUC

NAMD: Molecular Dynamics

- Collection of [charged] atoms, with bonds
- Newtonian mechanics
- At each time-step
  - Calculate forces on each atom
    - Bonds:
    - Non-bonded: electrostatic and van der Waal's
  - Calculate velocities and advance positions
- 1 femtosecond time-step, millions needed!
- Thousands of atoms (1,000 - 100,000)

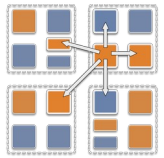
Collaboration with K. Schulten, R. Skeel, and coworkers



PPL UIUC

Further MD

- Use of cut-off radius to reduce work
  - 8 - 14 Å
  - Faraway charges ignored!
- 80-95 % work is non-bonded force computations
- Some simulations need faraway contributions



PPL UIUC

Traditional Approaches: non isoefficient

In 1996-2002

- Replicated Data:
  - All atom coordinates stored on each processor
    - Communication/Computation ratio: P log P

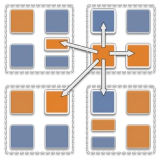
Not Scalable

- Partition the Atoms array across processors
  - Nearby atoms may not be on the same processor
  - C/C ratio: O(P)

Not Scalable

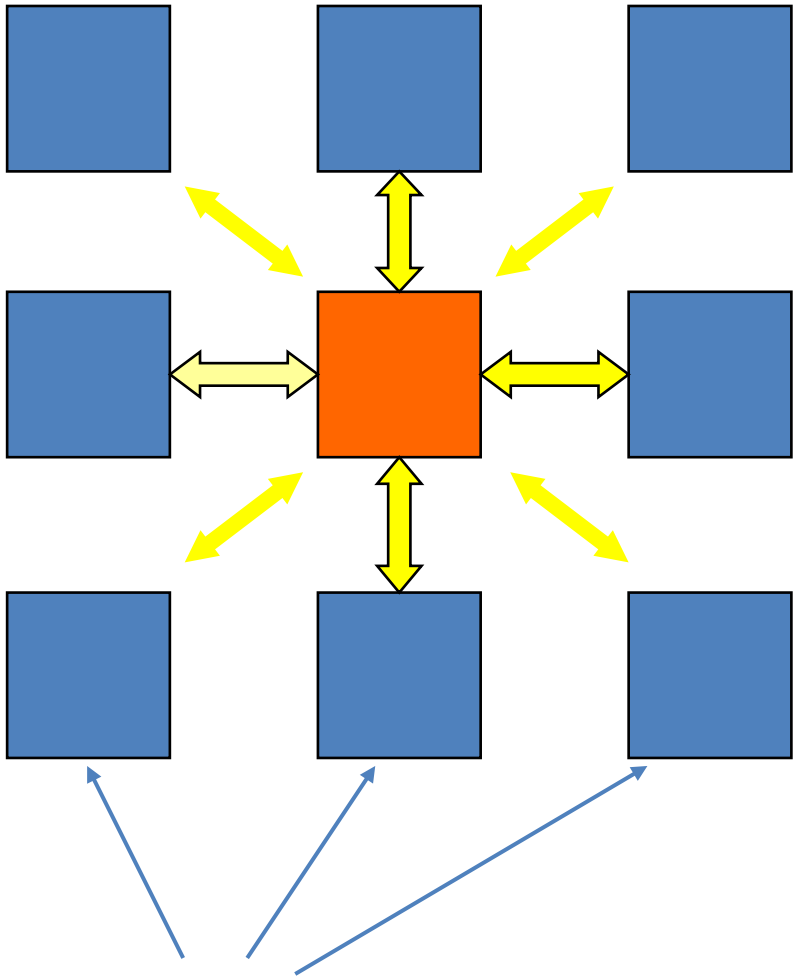
- Distribute force matrix to processors
  - Matrix is sparse, non uniform,
  - C/C Ratio: sqrt(P)

Not Scalable



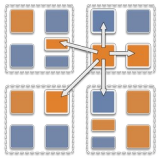
PPL UIUC

Spatial Decomposition Via Charm



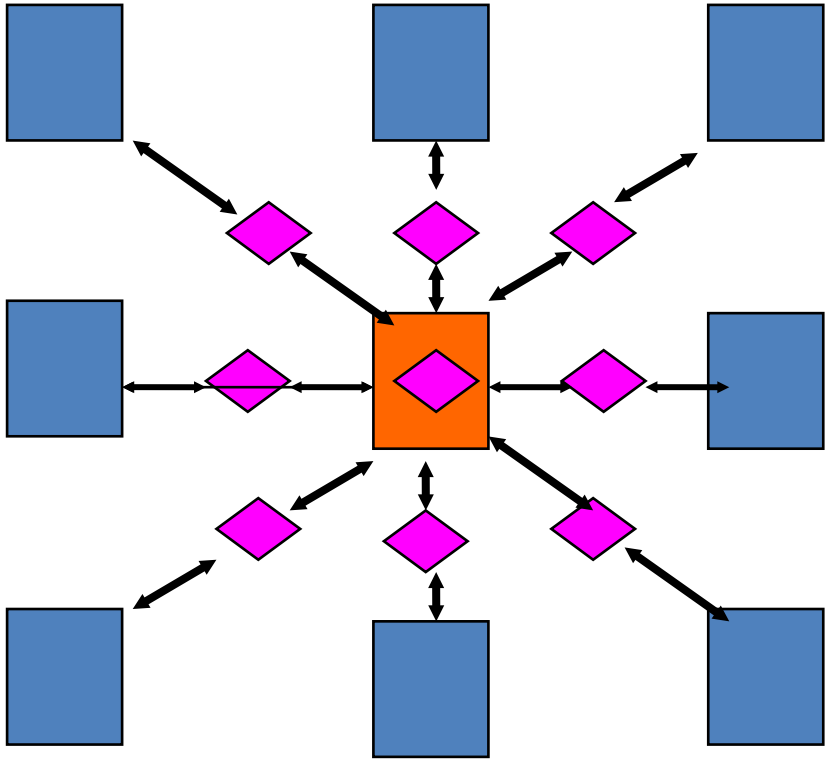
- Atoms distributed to cubes based on their location
- Size of each cube :
  - Just a bit larger than cut-off radius
  - Communicate only with neighbors
  - Work: for each pair of nbr objects
- C/C ratio: O(1)
- ***However:***
  - ***Load Imbalance***
  - ***Limited Parallelism***

Charm++ is useful to handle this

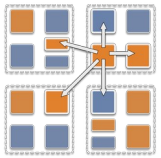


Object Based Parallelization for MD:

Force Decomposition + Spatial Decomposition

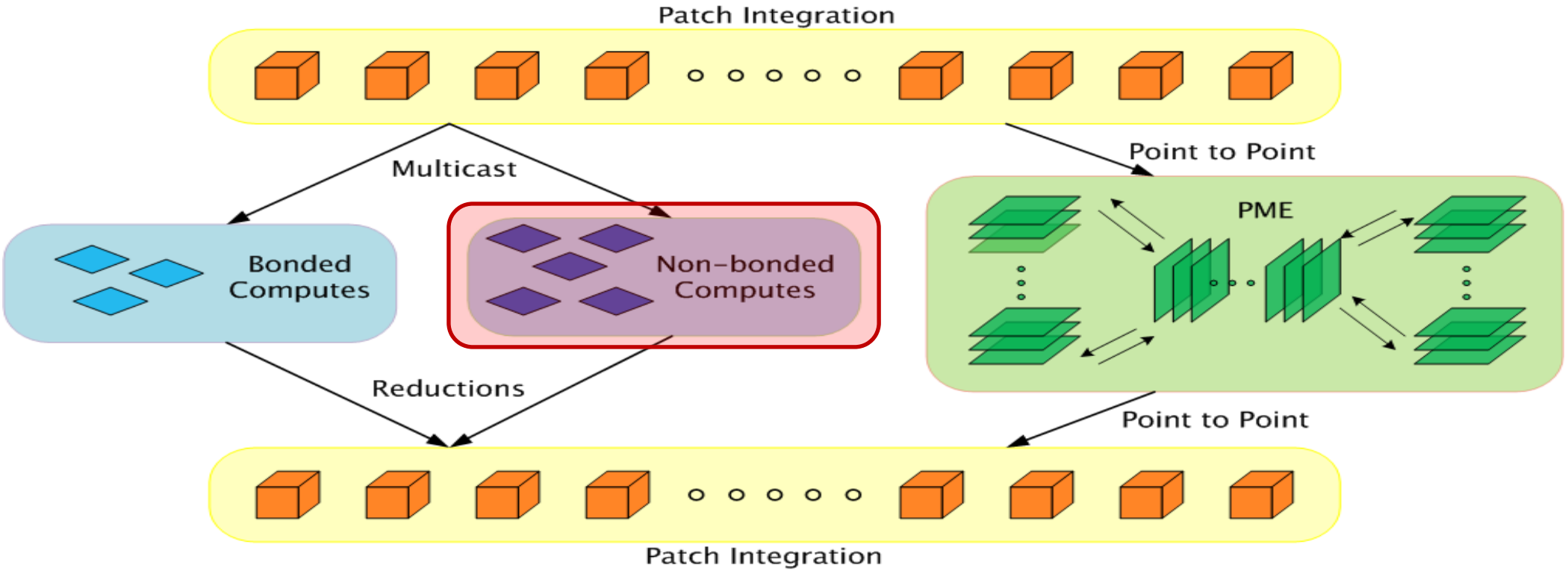


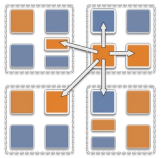
- Now, we have many objects to load balance:
  - Each diamond can be assigned to any proc.
  - Number of diamonds (3D):
    - 14·Number of Patches
- 2-away variation:
  - Half-size cubes
  - 5x5x5 interactions
- 3-away interactions: 7x7x7



PPL UIUC

Parallelization using Charm++

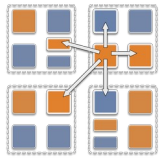




PPL UIUC

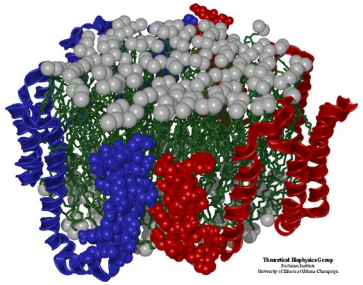
Amdahl and variants

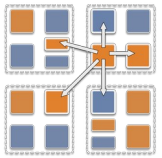
- The original Amdahl's law, interpreted as:
  - If there is a x% sequential component, speedup can't be more than 100/x.
- Variations:
  - If you decompose a problem into many parts, then the parallel time cannot be less than the largest of the parts
  - If the critical path through a computation is T1, you cannot complete in less time than T, no matter how many processors you use
  - …



Grainsize and Amdahls's law

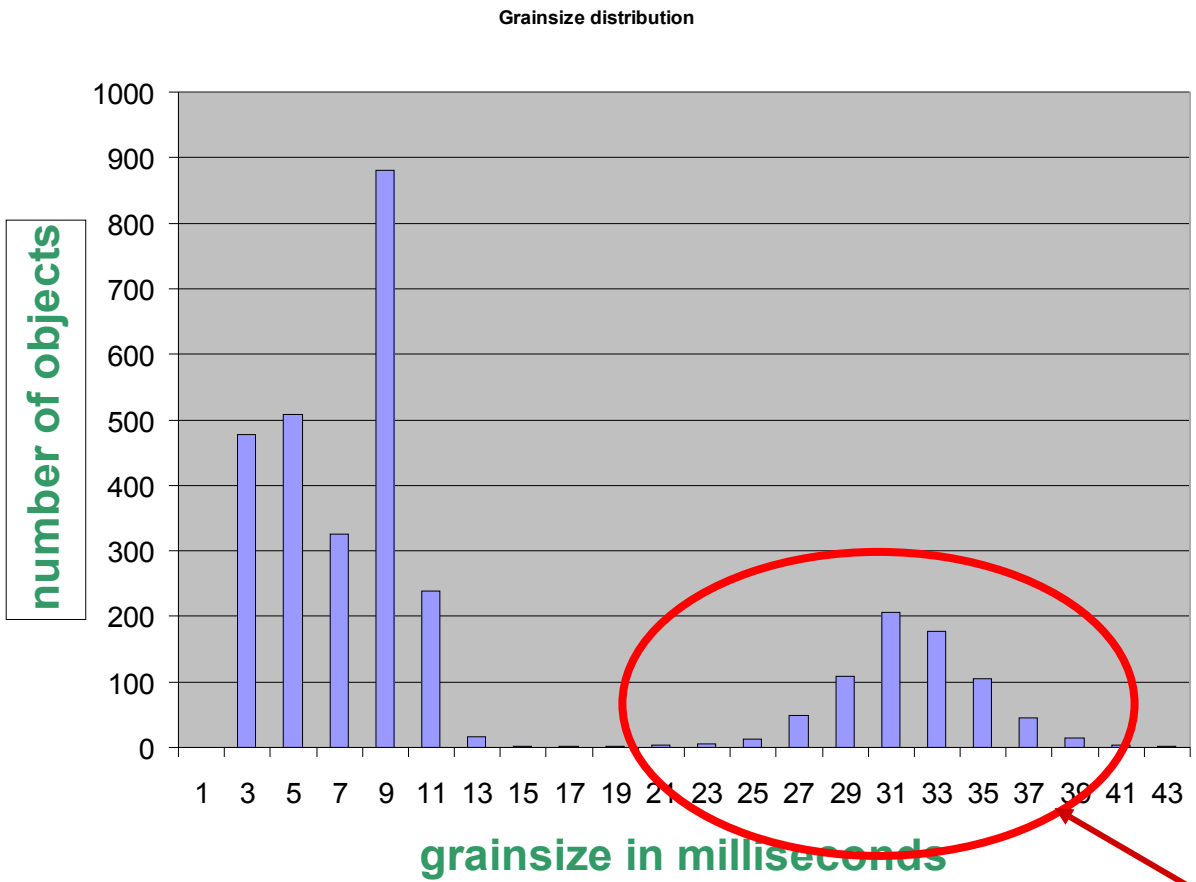
- A variant of Amdahl's law, for objects:
  - The fastest time can be no shorter than the time for the biggest single object!
- How did it apply to us?
  - Sequential step time was 57 seconds
  - To run on 2k processors, no object should be more than 28 msecs.
  - Analysis using our tools showed:





PPL UIUC

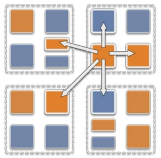
Grainsize analysis



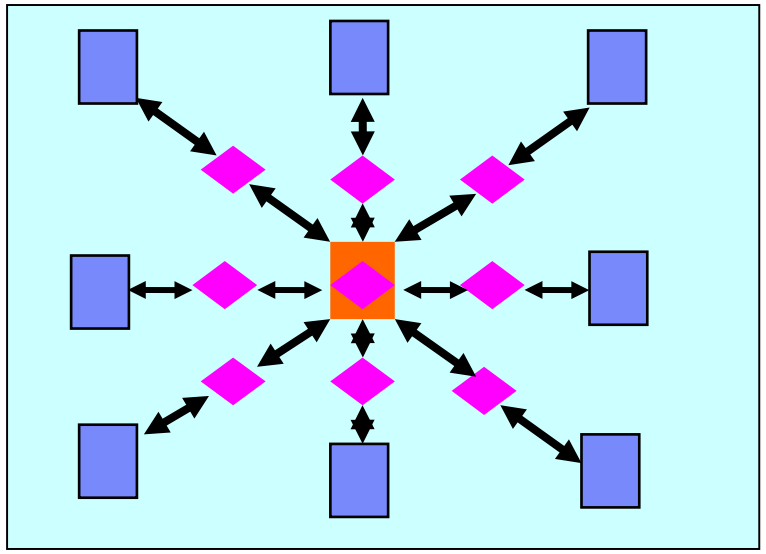
Solution:

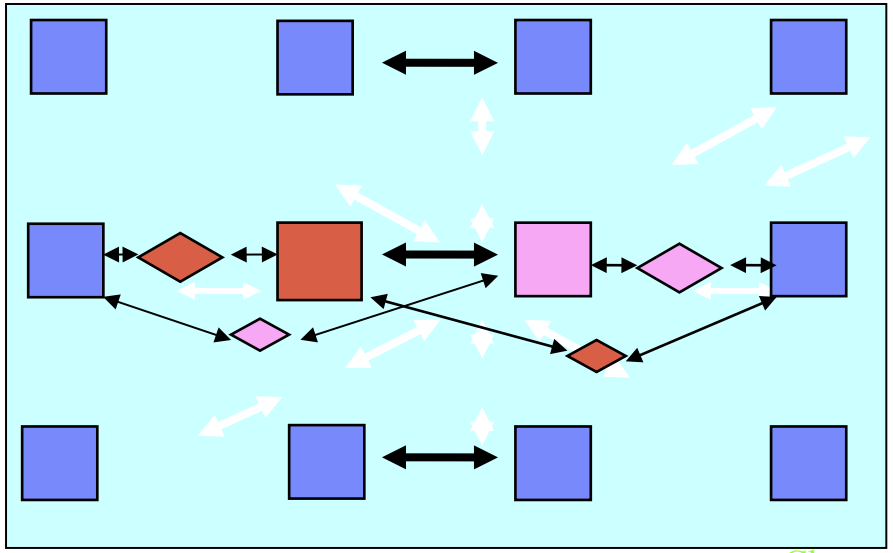
Split compute objects that may have too much work:

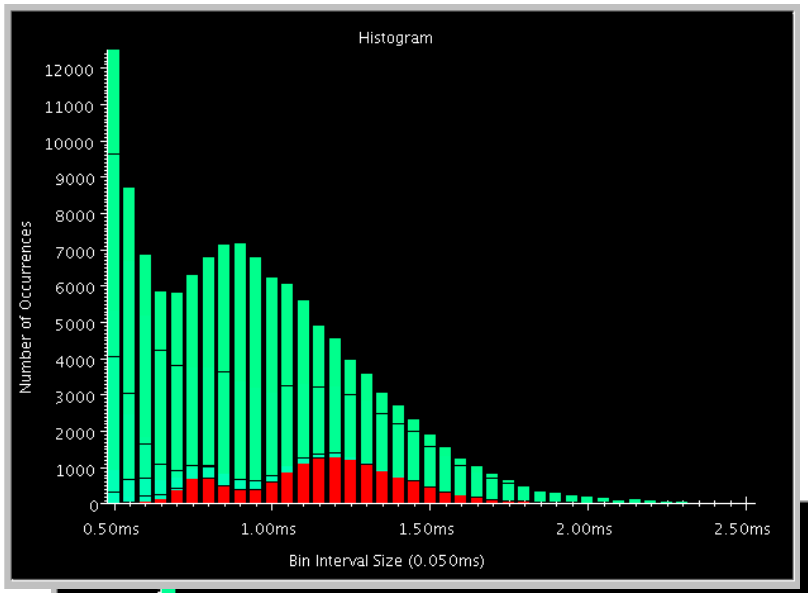
using a heuristics based on number of interacting atoms

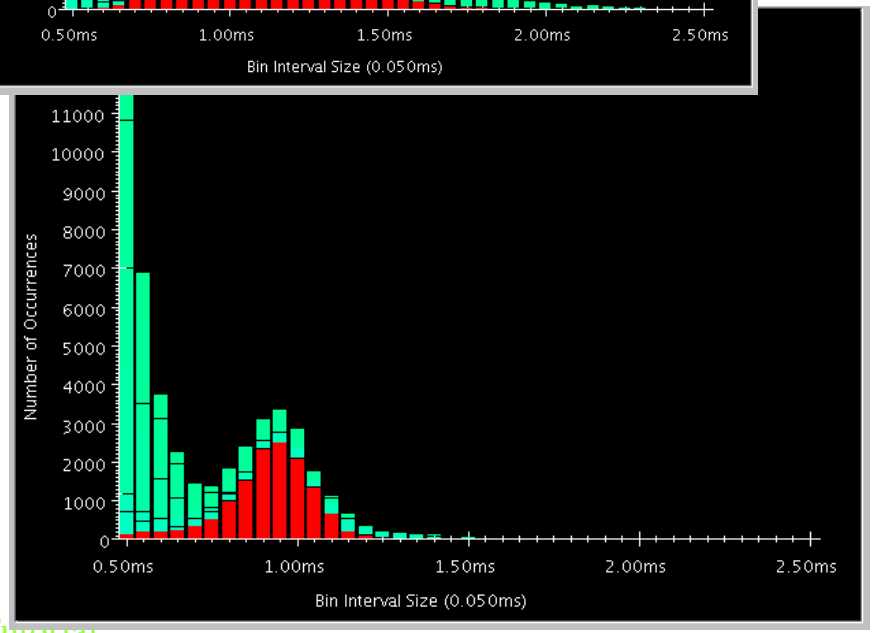


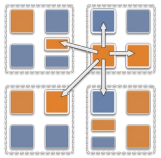
Fine Grained Decomposition on BlueGene





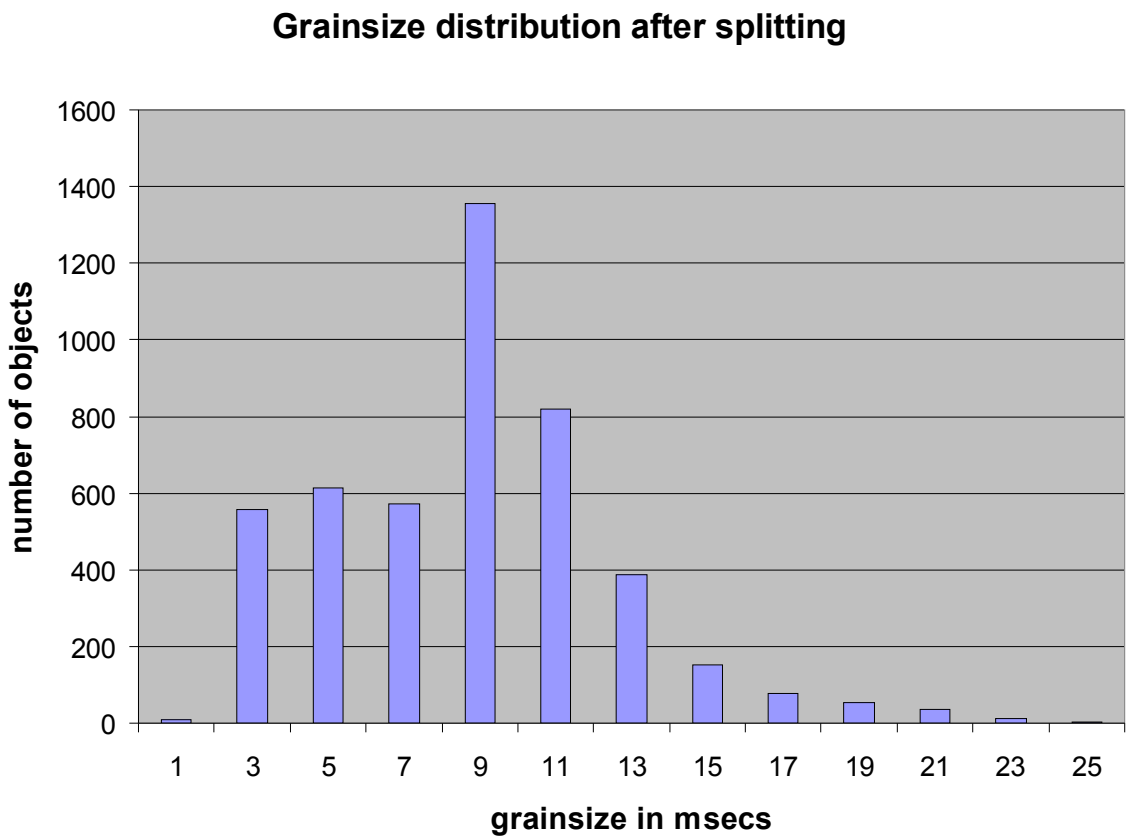


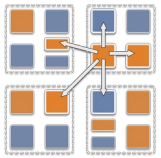




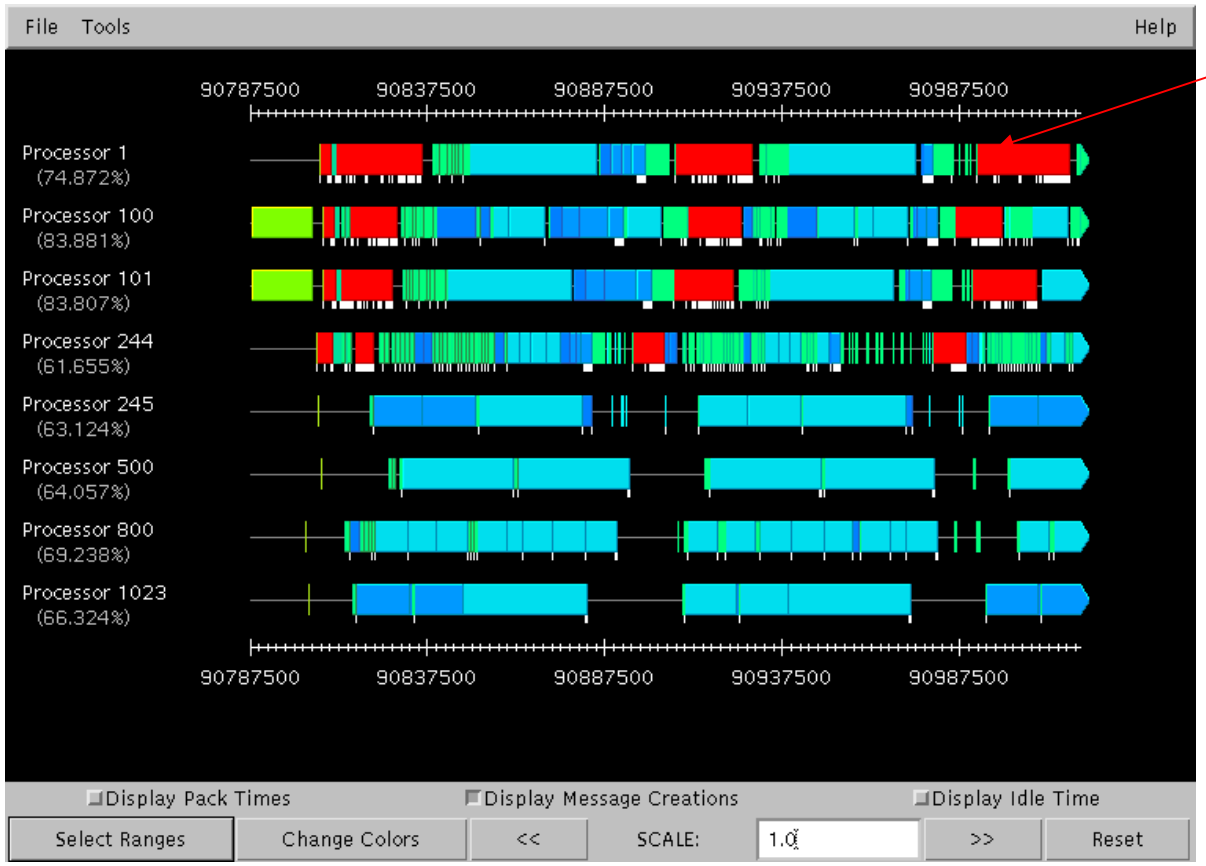
PPL UIUC

Grainsize reduced



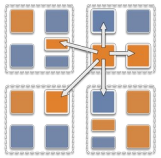


Integration overhead analysis



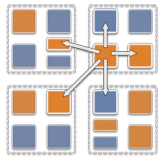
integration

Problem: integration time had doubled from sequential run



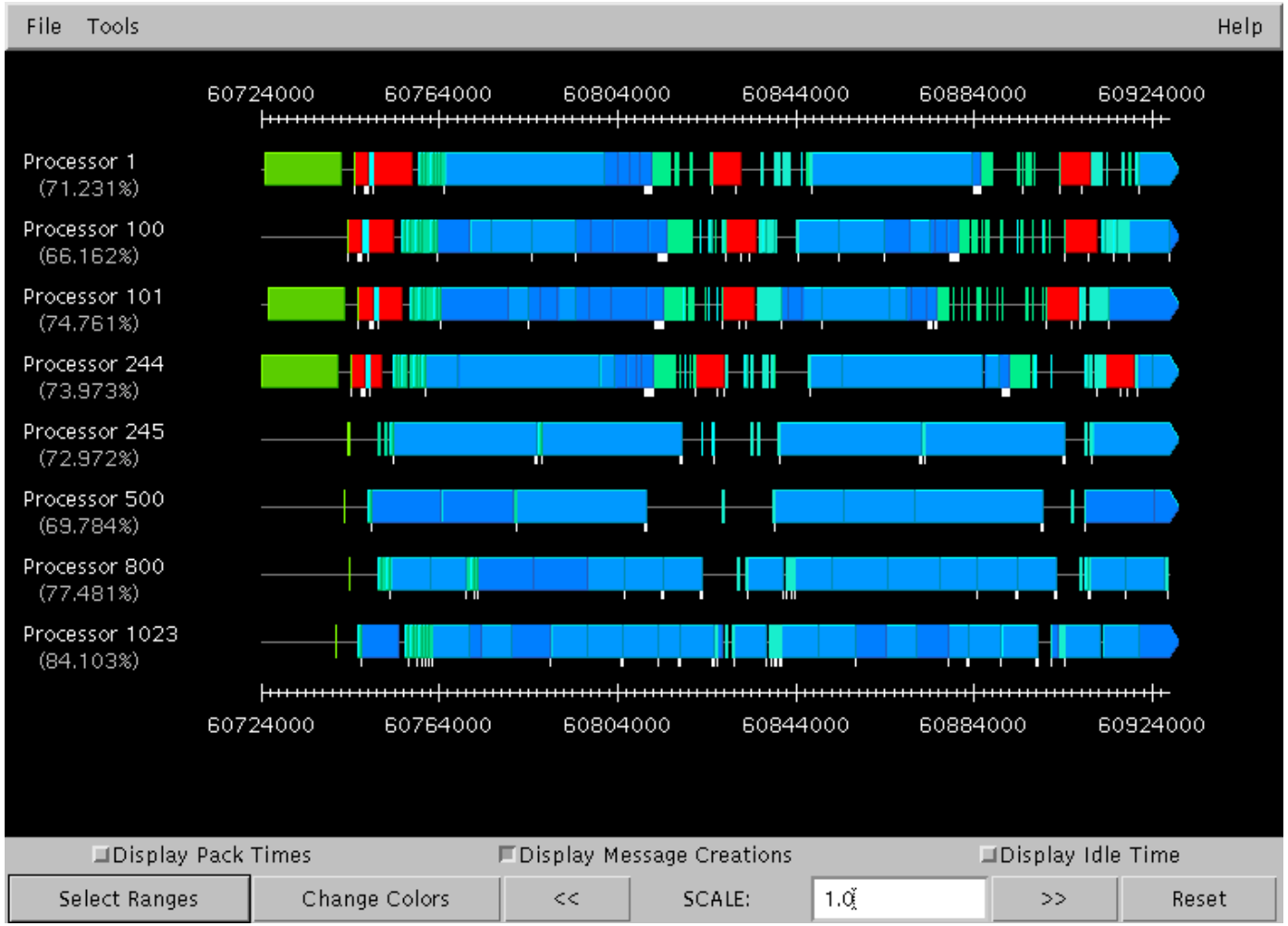
Integration overhead example:

- The visualization showed: the overhead was associated with sending messages.
- Many cells were sending 30-40 messages.
  - The overhead per message was too high
  - Code analysis: memory allocations!
  - Identical message being sent to 30+ processors.
- Multicast support was added to Charm++
  - Mainly eliminates memory allocations

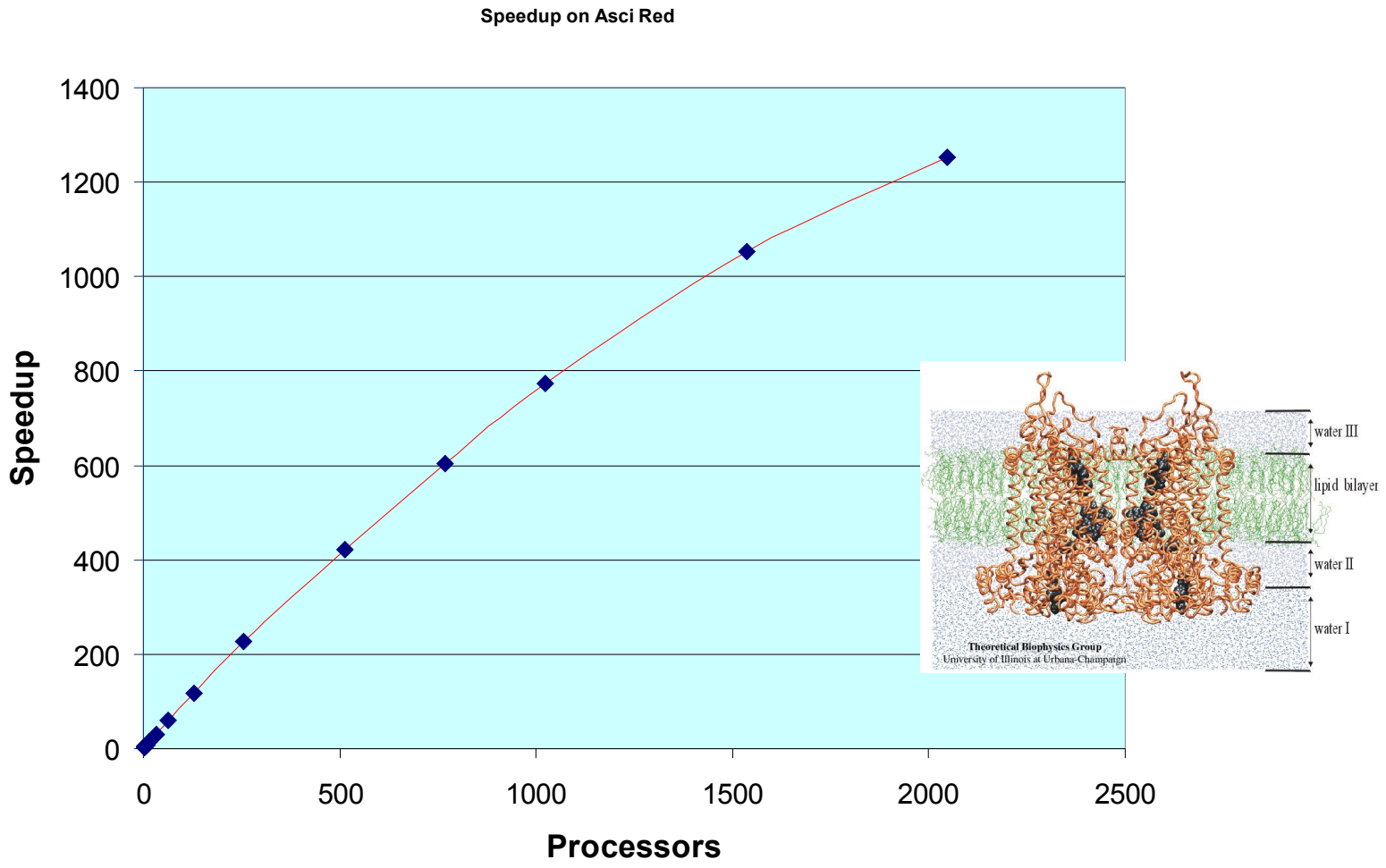


PPL UIUC

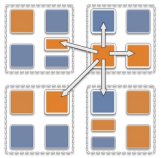
Integration overhead: After



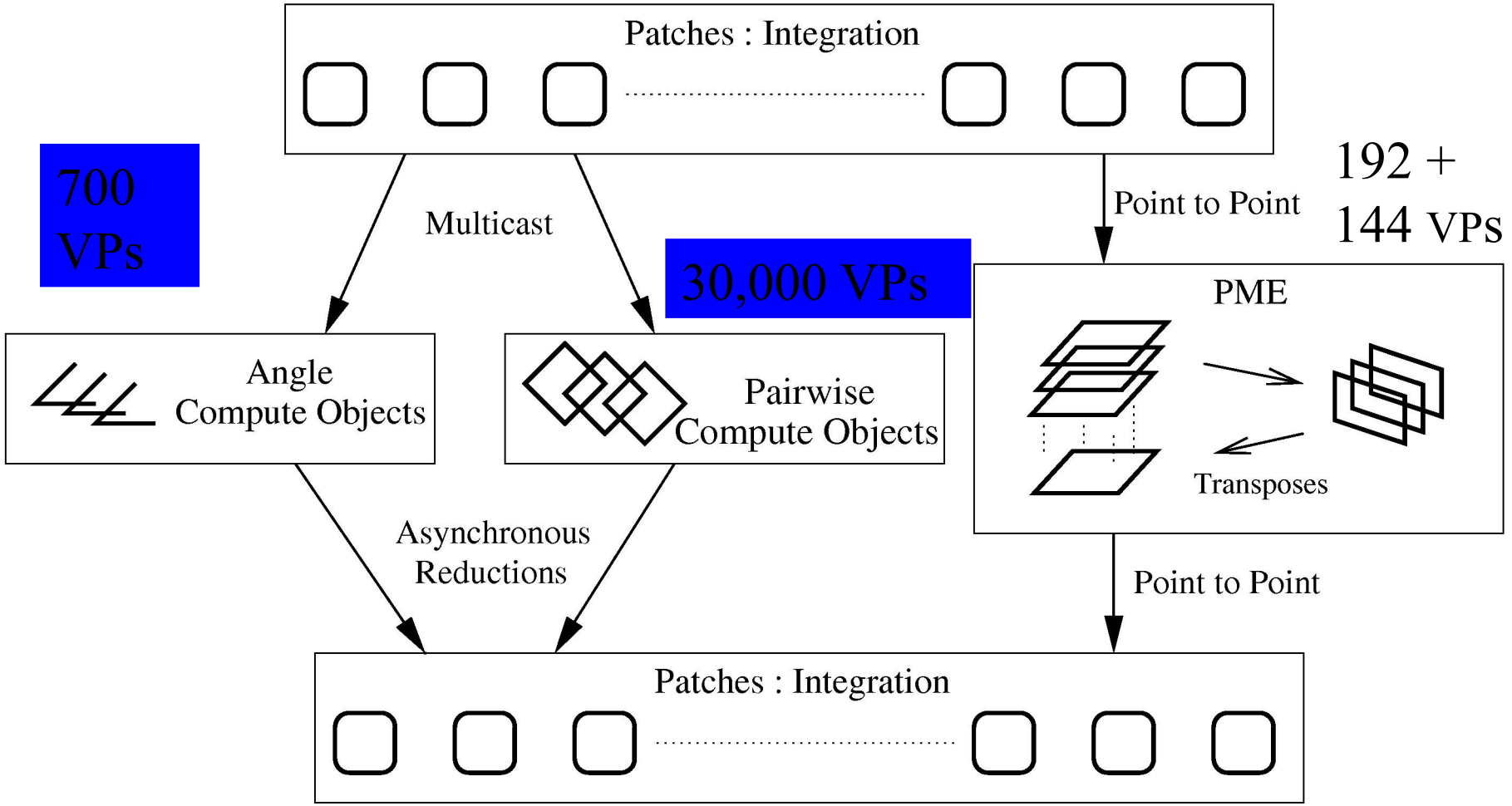
Improved Performance Data



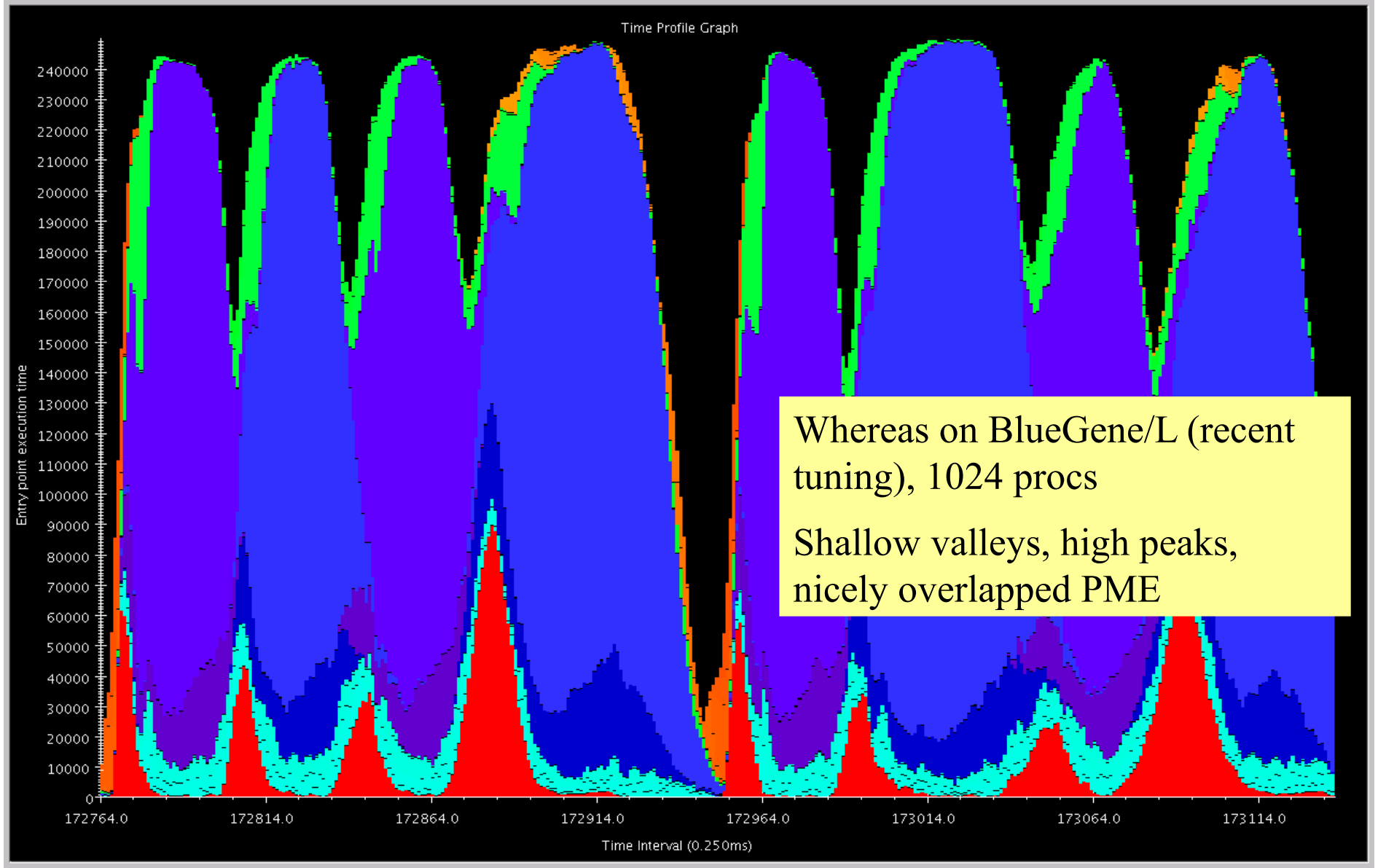
Published in SC2000: Gordon Bell Award Finalist



NAMD Parallelization using Charm++ : PME

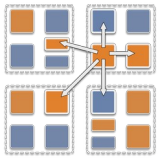


These 30,000+ Virtual Processors (VPs) are mapped to real processors by charm runtime system

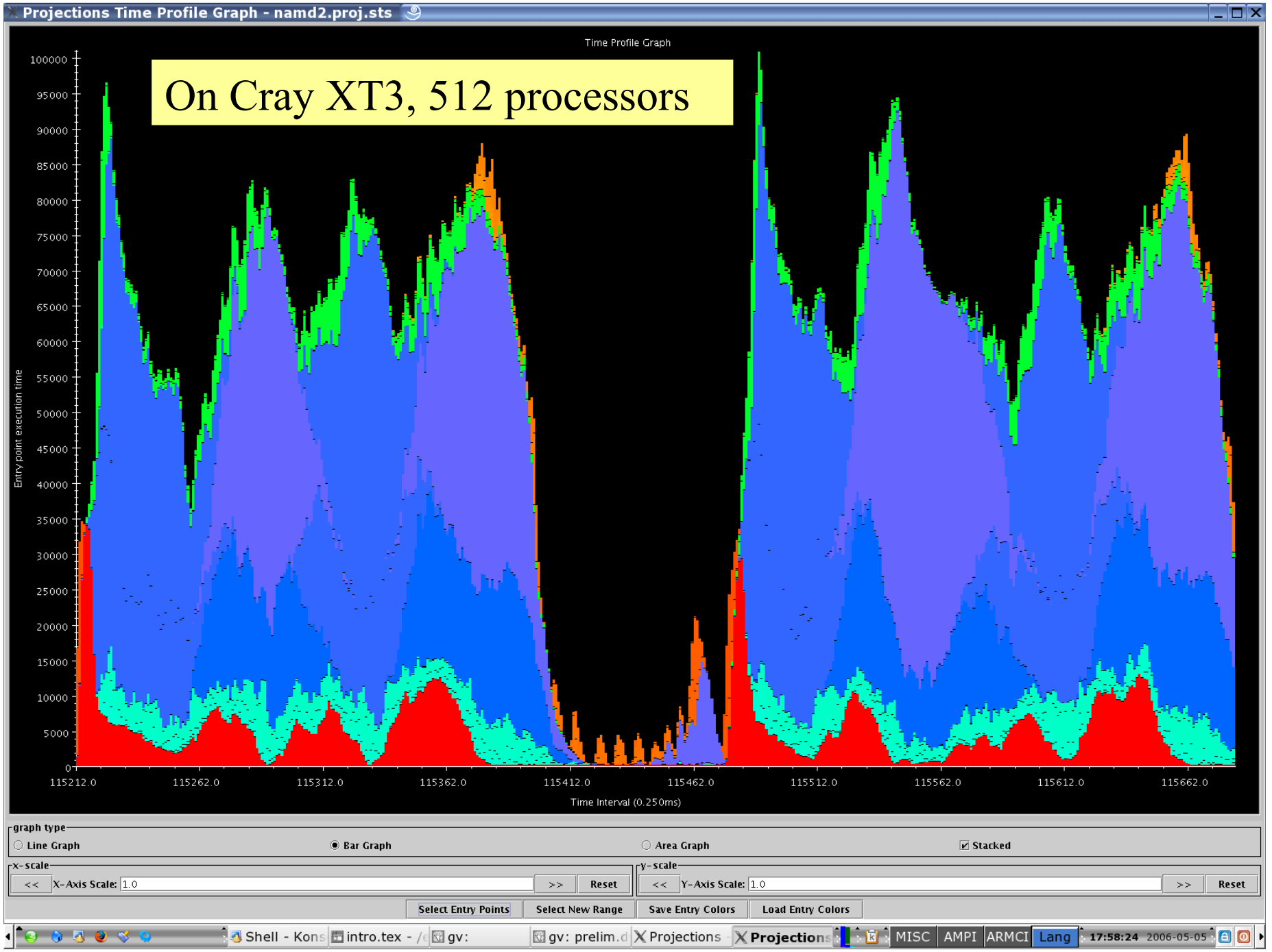


Whereas on BlueGene/L (recent tuning), 1024 procs

Shallow valleys, high peaks, nicely overlapped PME

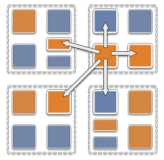


PPL UIUC

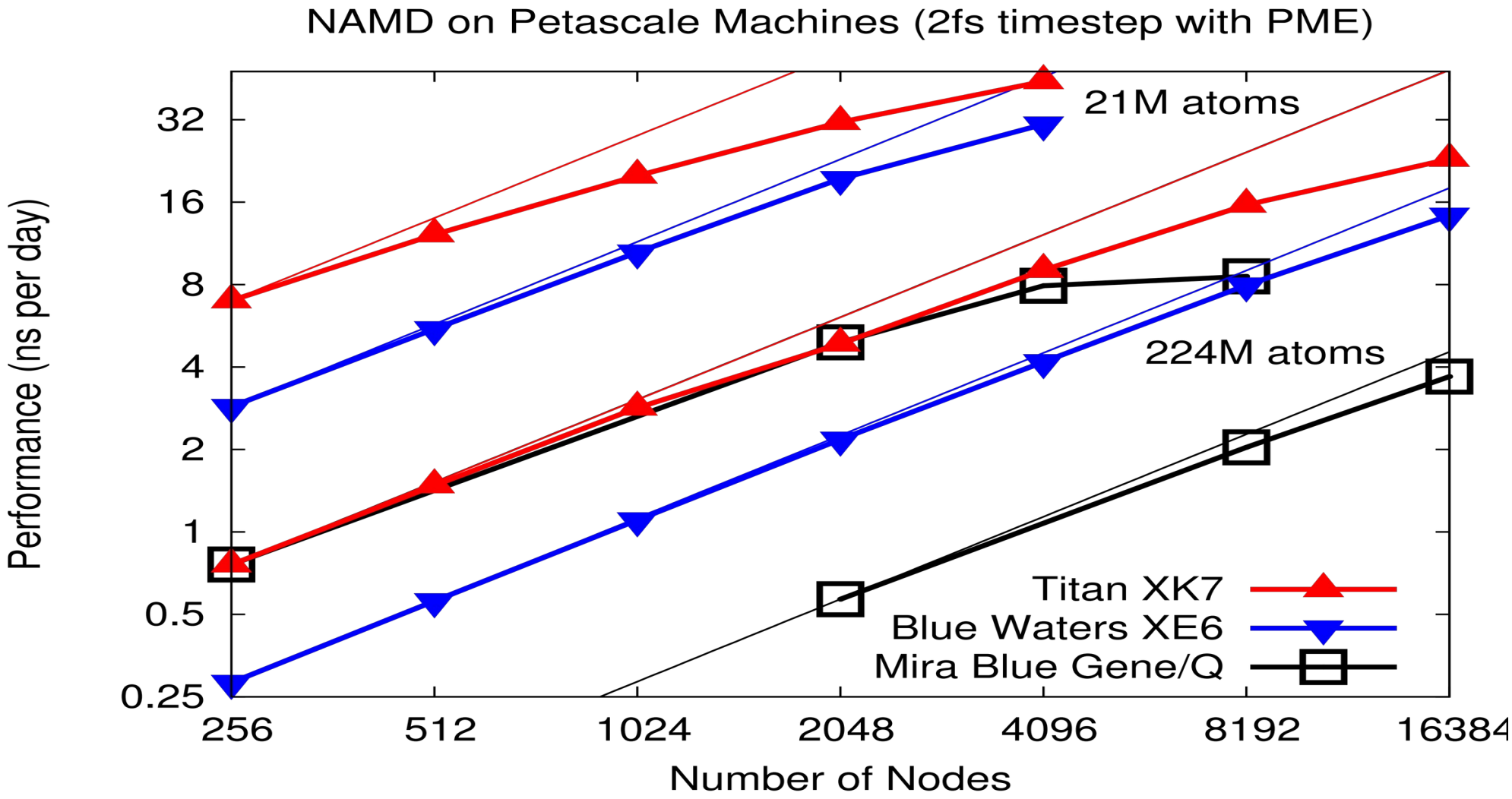

Projections Time Profile Graph - namd2.proj.sts
Time Profile Graph
On Cray XT3, 512 processors
Entry point execution time
Time Interval (0.250ms)
graph type
Line Graph
Bar Graph
Area Graph
Stacked
x-scale
X-Axis Scale: 1.0
y-scale
Y-Axis Scale: 1.0
Reset
Select Entry Points
Select New Range
Save Entry Colors
Load Entry Colors
MISC
AMPI
ARMCI
Lang
17:58:24
2006-05-05

Grainsize example: NAMD

- High performing examples (objects are the work-data units in Charm++):
- On Blue Waters, 100M atom simulation
  - 128K cores (4K nodes): 5,510,202 objects
- Edison, Apoa1 (92K atoms)
  - 4K cores: 33,124 objects
- Hopper, STMV (1M atoms)
  - 15,360 cores: 430,612 objects



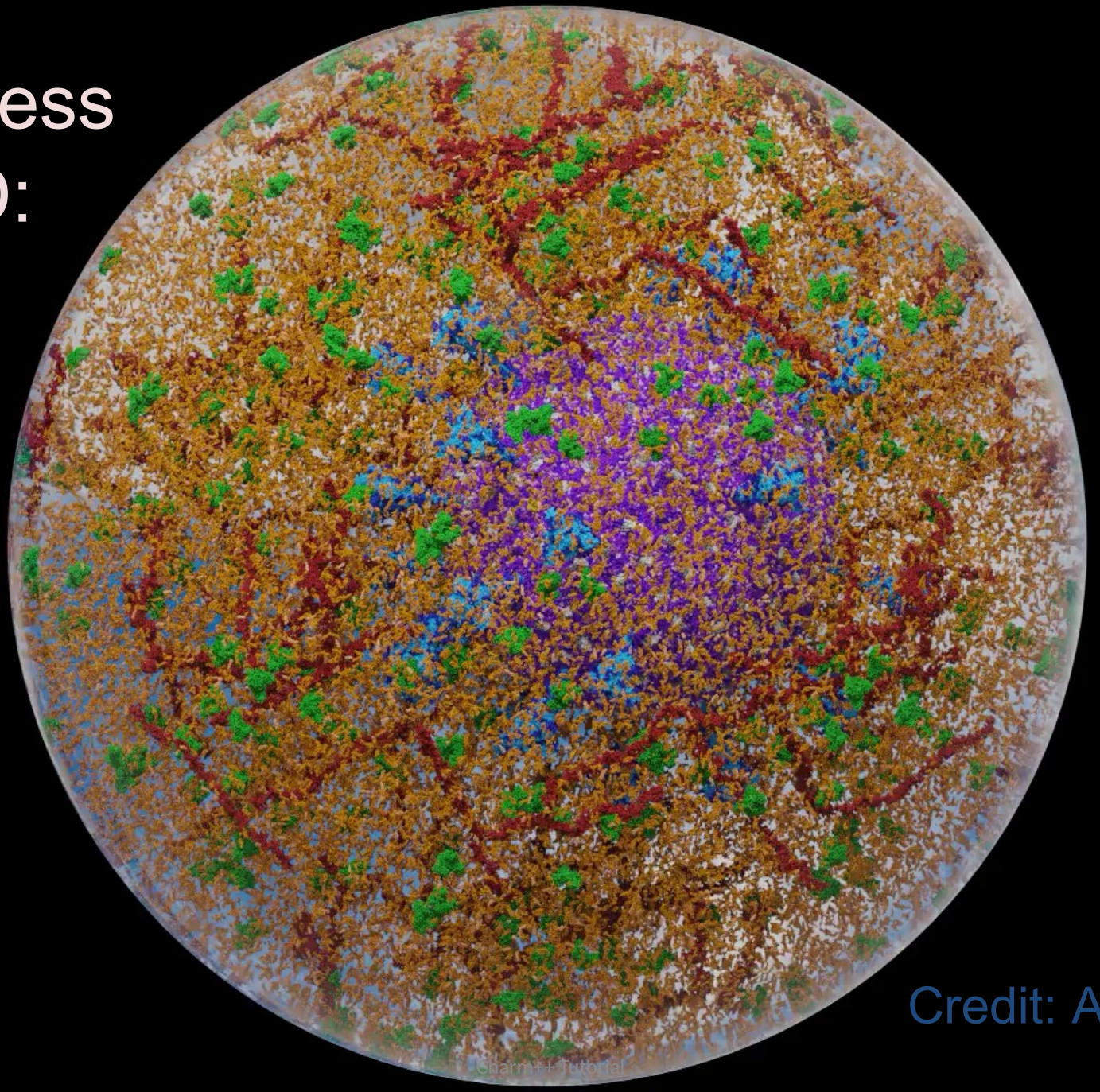
PPL UIUC



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

Recent success with NAMD:

Coronavirus simulations



Credit: Amaro Lab UCSD

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

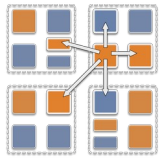
With Charm++: Use Oct-Tree, and let Charm++ map subtrees to processors

ChaNGa: Parallel Gravity

Evolution of Universe and Galaxy Formation

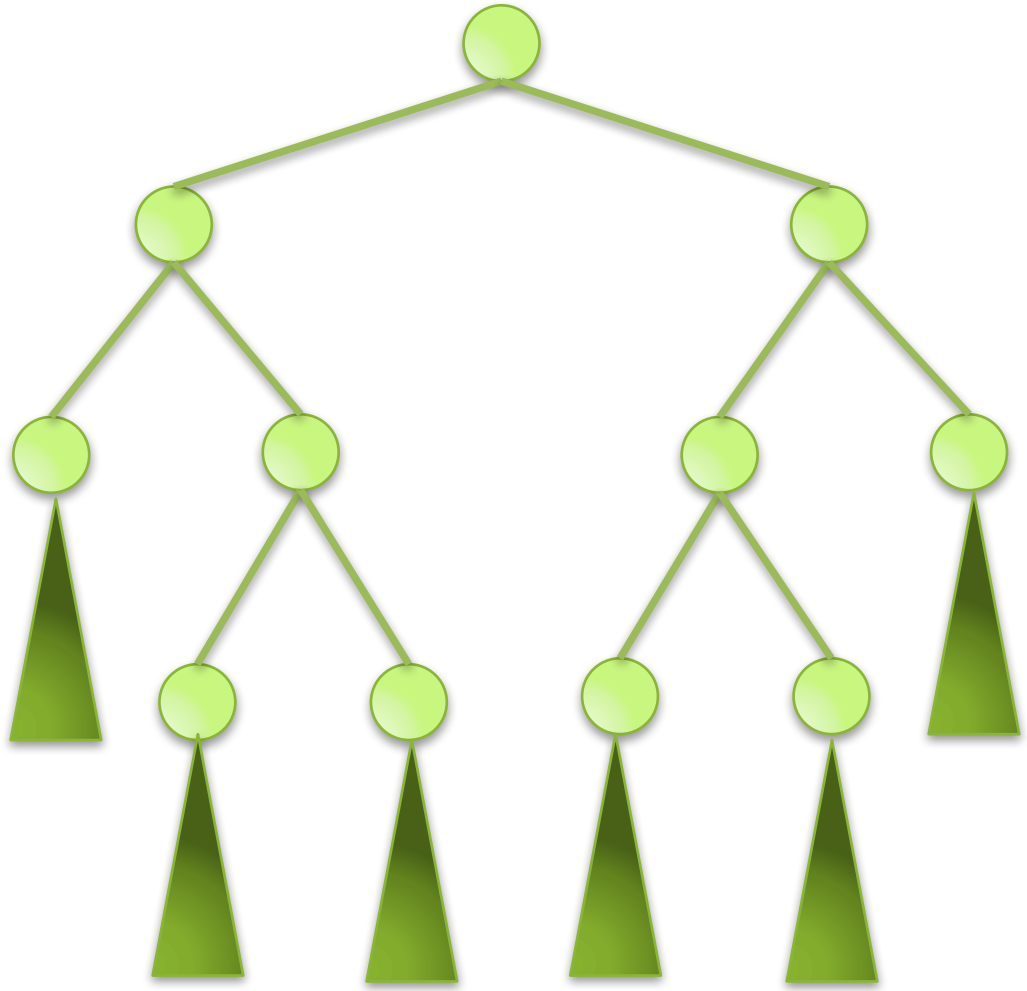
- Collaborative project (NSF)
  - with Tom Quinn, Univ. of Washington
- Gravity, gas dynamics
- Barnes-Hut tree codes
  - Oct tree is natural decomp
  - Geometry has better aspect ratios, so you "open" up fewer nodes
  - But is not used because it leads to bad load balance
  - Assumption: one-to-one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

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



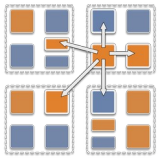
PPL UIUC

ChaNGa: Cosmology Simulation



Collaboration with Tom Quinn UW

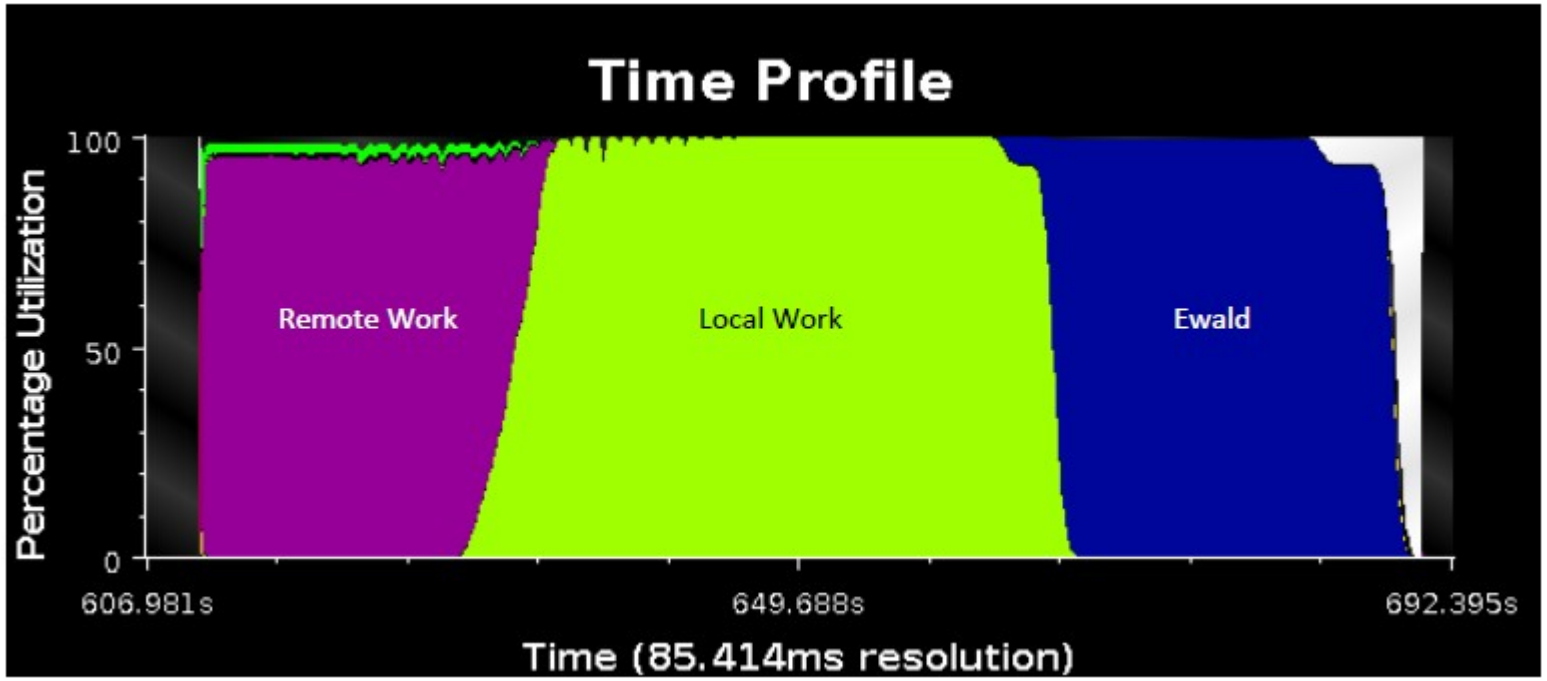
- Tree: Represents particle distribution
- TreePiece: object/chares containing particles

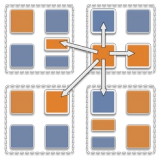


PPL UIUC

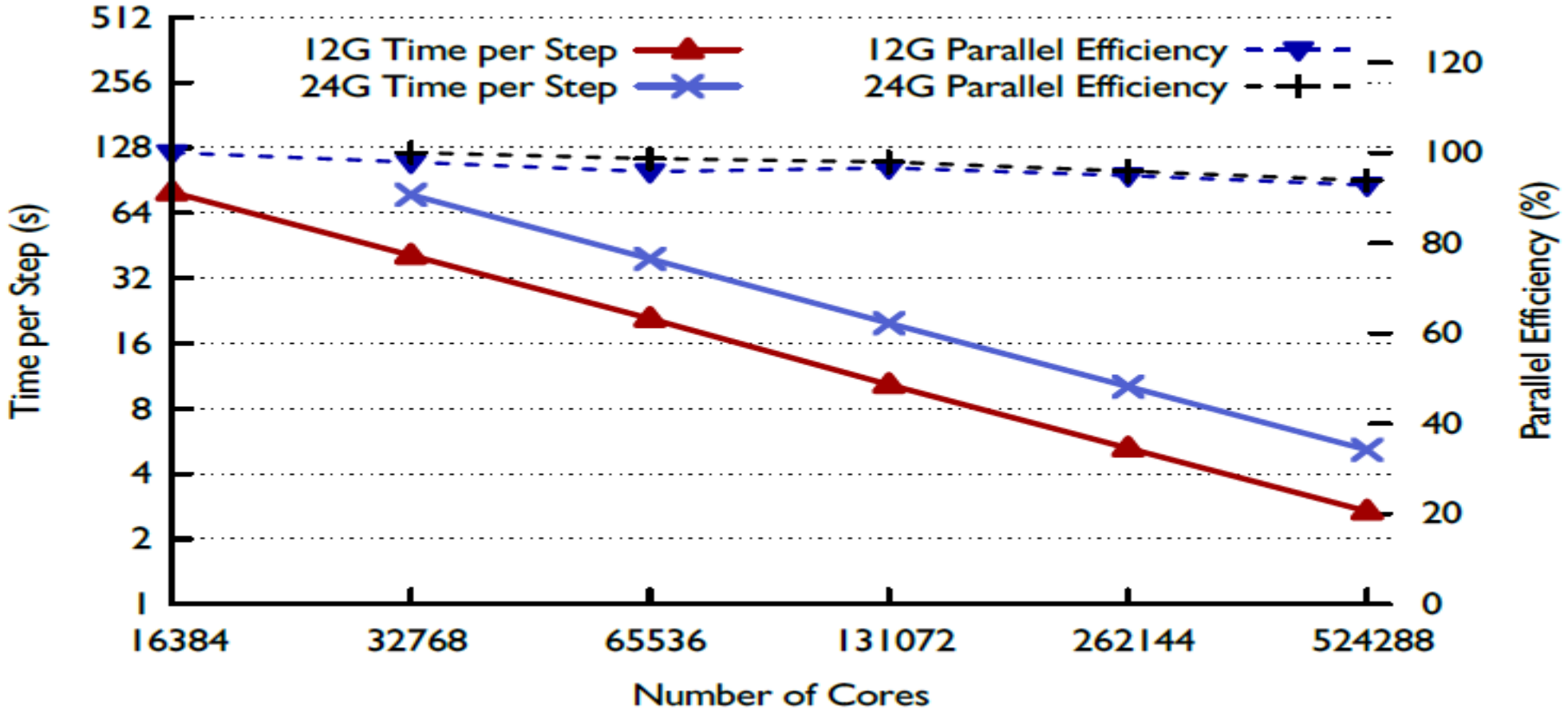
ChaNGa: Optimized Performance

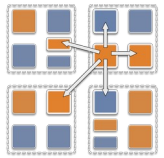
- Asynchronous, highly overlapped, phases
- Requests for remote data overlapped with local computations



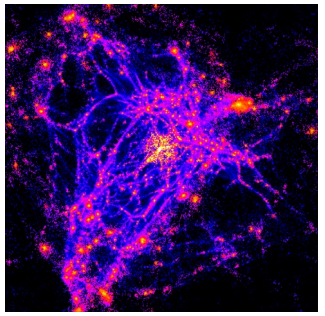


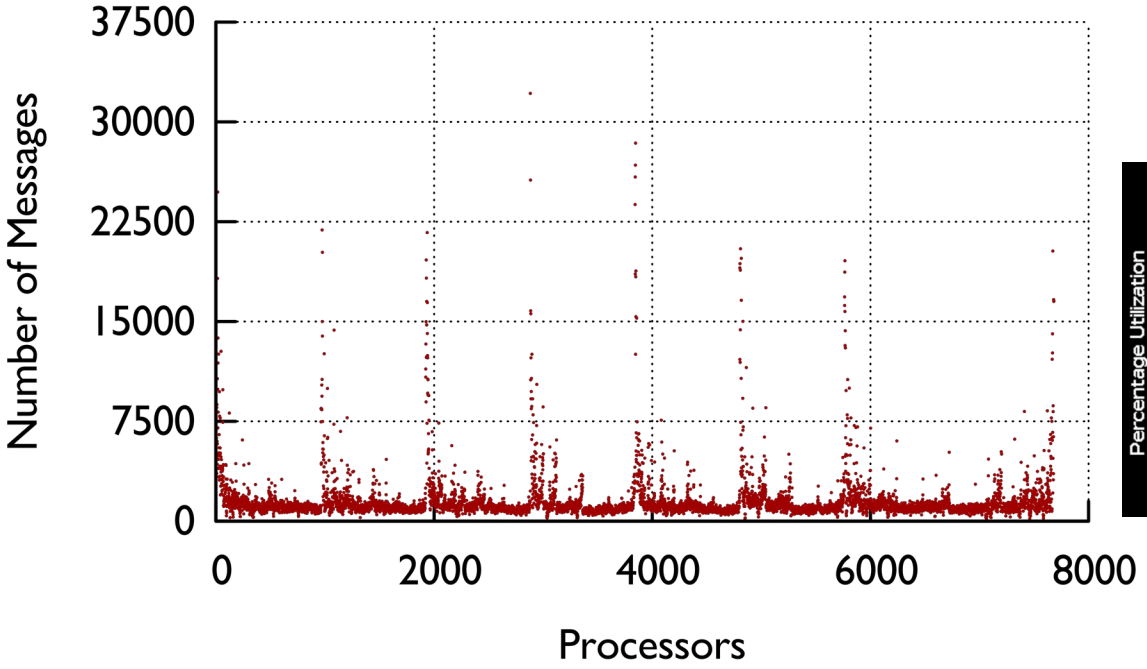
ChaNGa : Resultant Performance on Blue Waters

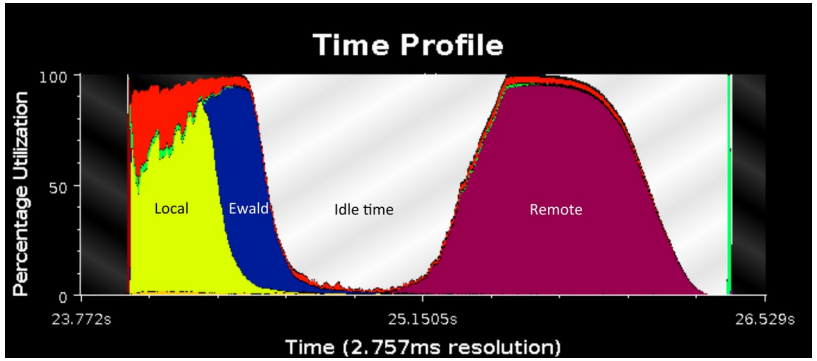




Clustered Dataset - Dwarf

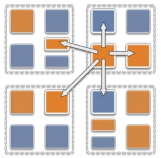




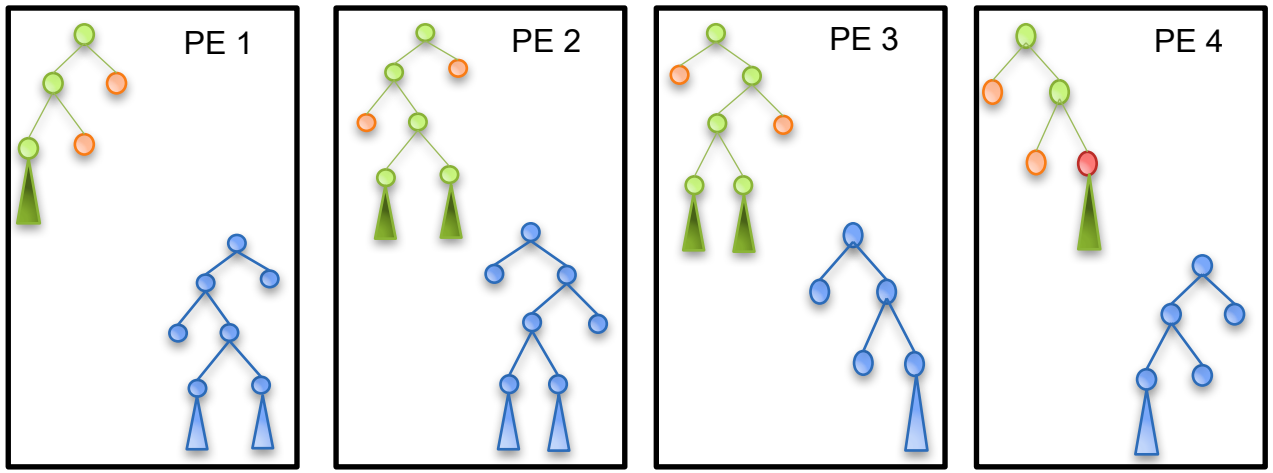


- Highly clustered
- Maximum request per processor: > 30K

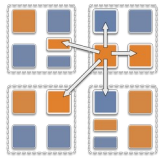
- Idle time due to message delays
- Also, load imbalances: solved by Hierarchical balancers



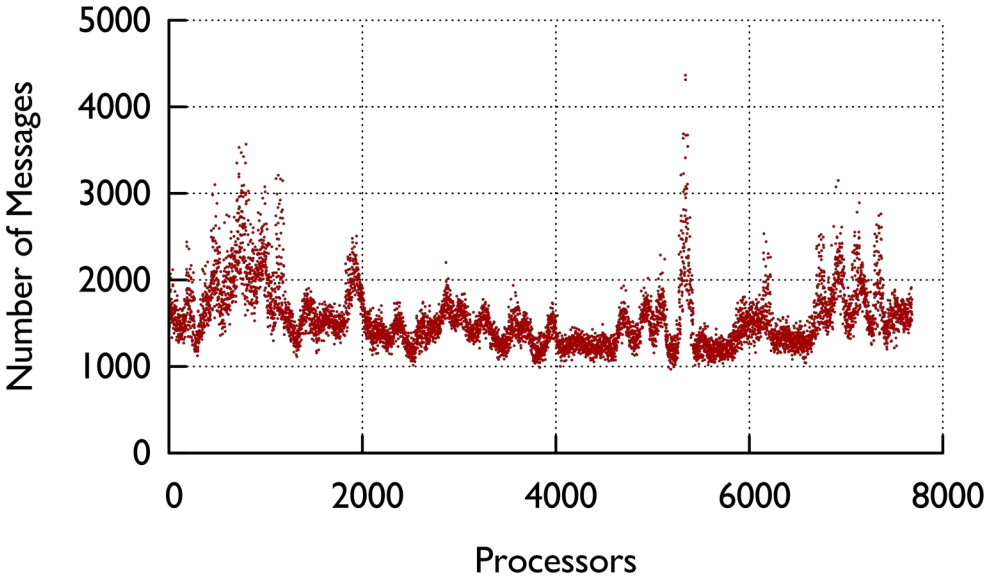
Solution: Replication

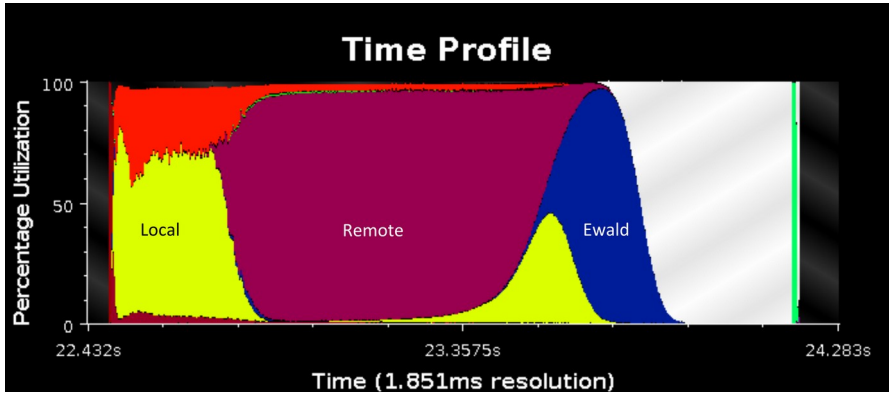


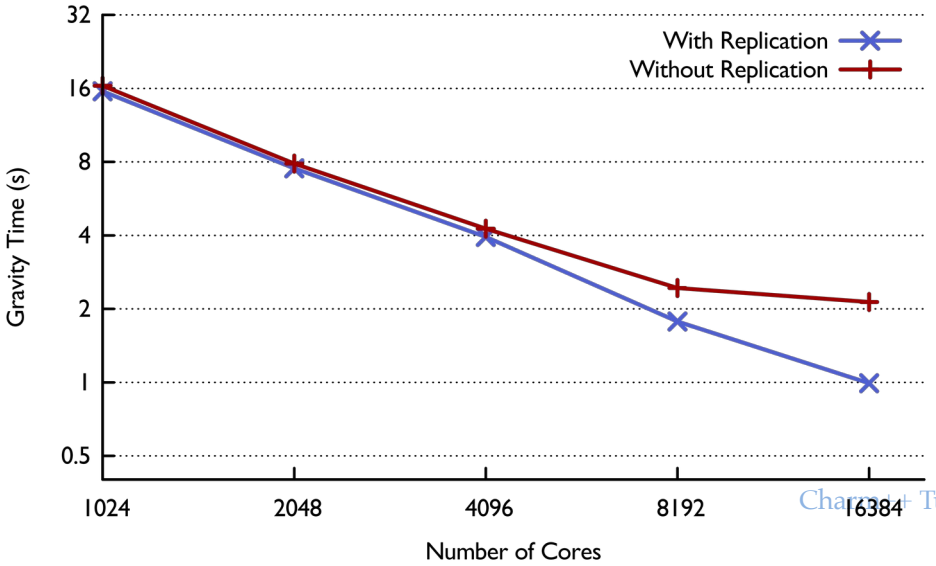
- Replicate tree nodes to distribute requests
- Requester randomly selects a replica



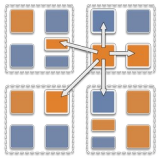
Replication Impact





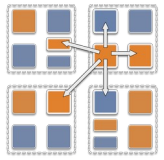


- Replication distributes requests
- Maximum request reduced from 30K to 4.5K
- Gravity time reduced from 2.4 s to 1.7 s, on 8k



Multiple time-stepping!

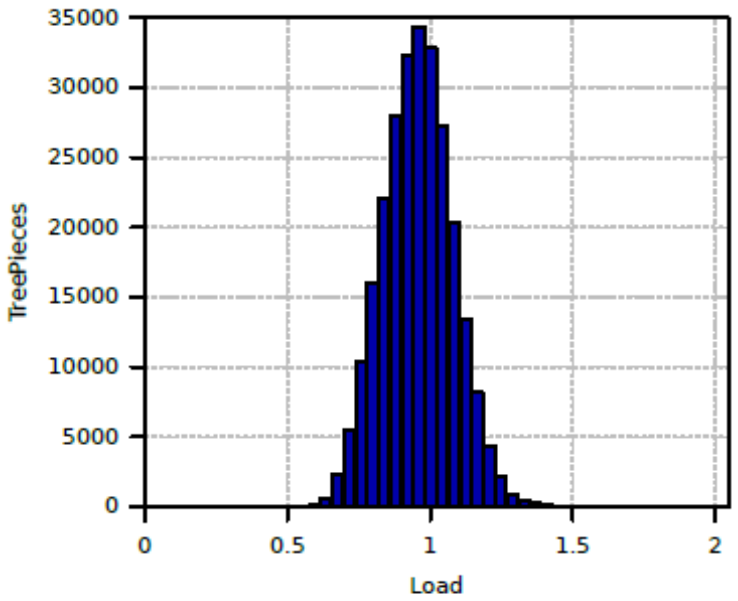
- Our scientist collaborators suggest an algorithmic optimization:
  - Don't move slow-moving particles every step
    - i.e. don't calculate forces on them either
  - In fact, make many (say 5) categories (rungs) of particles based on their velocities
  - Rung sequence (with 5 rungs)
    - 4 3 4 2 4 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



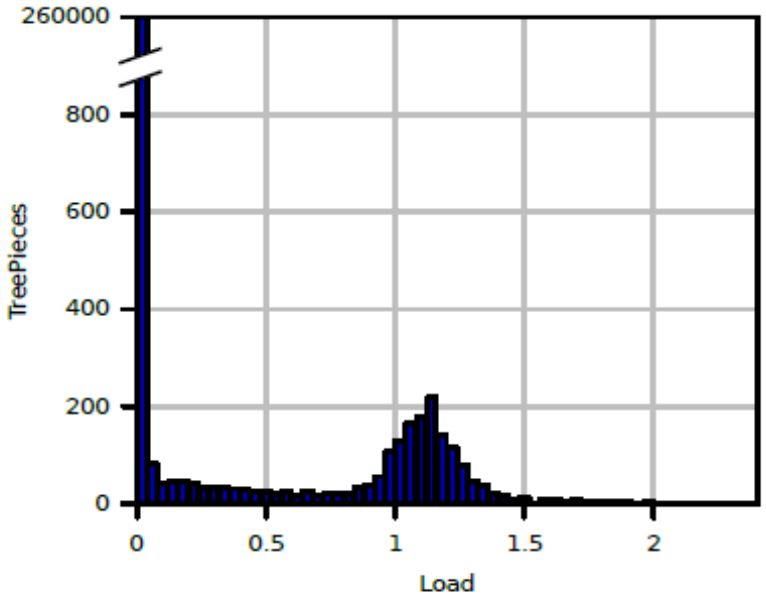
PPL UIUC

Multiple time-stepping!

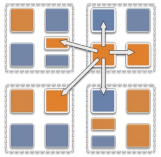
- Load (for the same object) changes across rungs
  - Yet, there is persistence within the same rung!
  - So, specialized phase-aware balancers were developed



(a) Rung 0



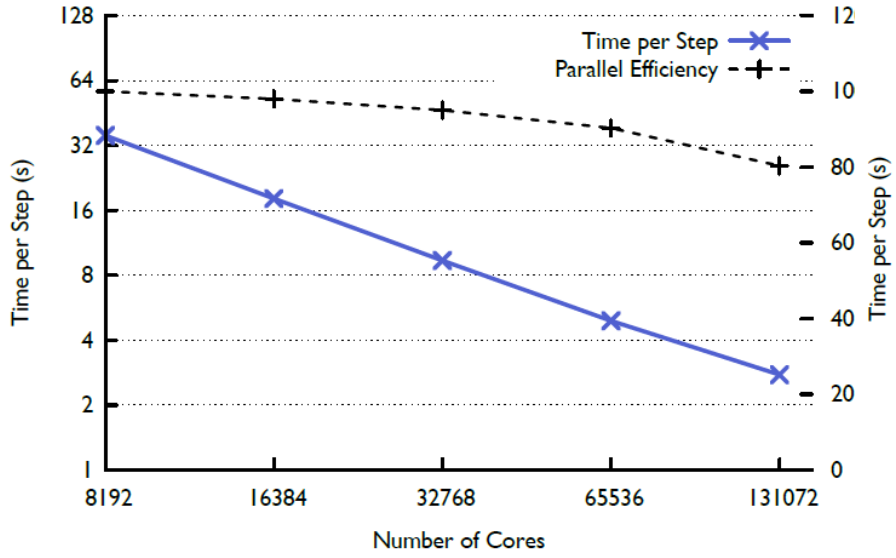
(b) Rung 4



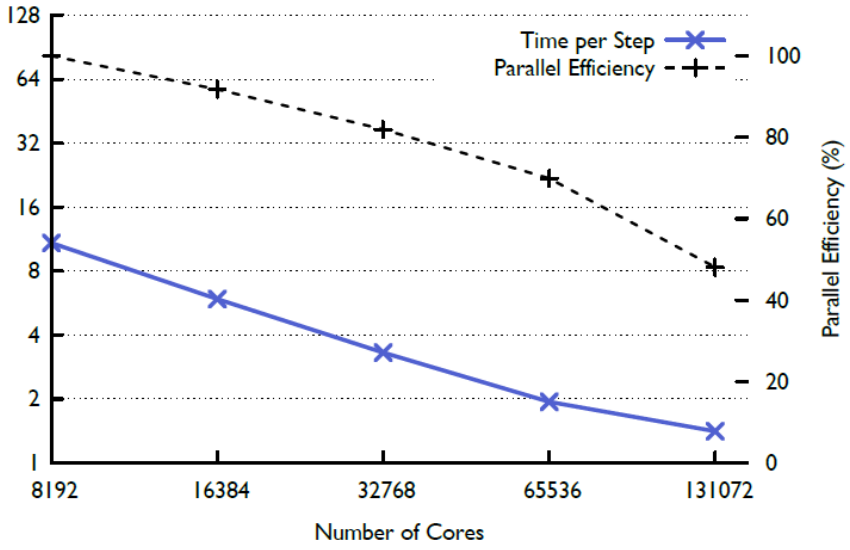
PPL UIUC

Multi-stepping tradeoff

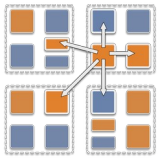
- Parallel efficiency is lower, but performance is improved significantly



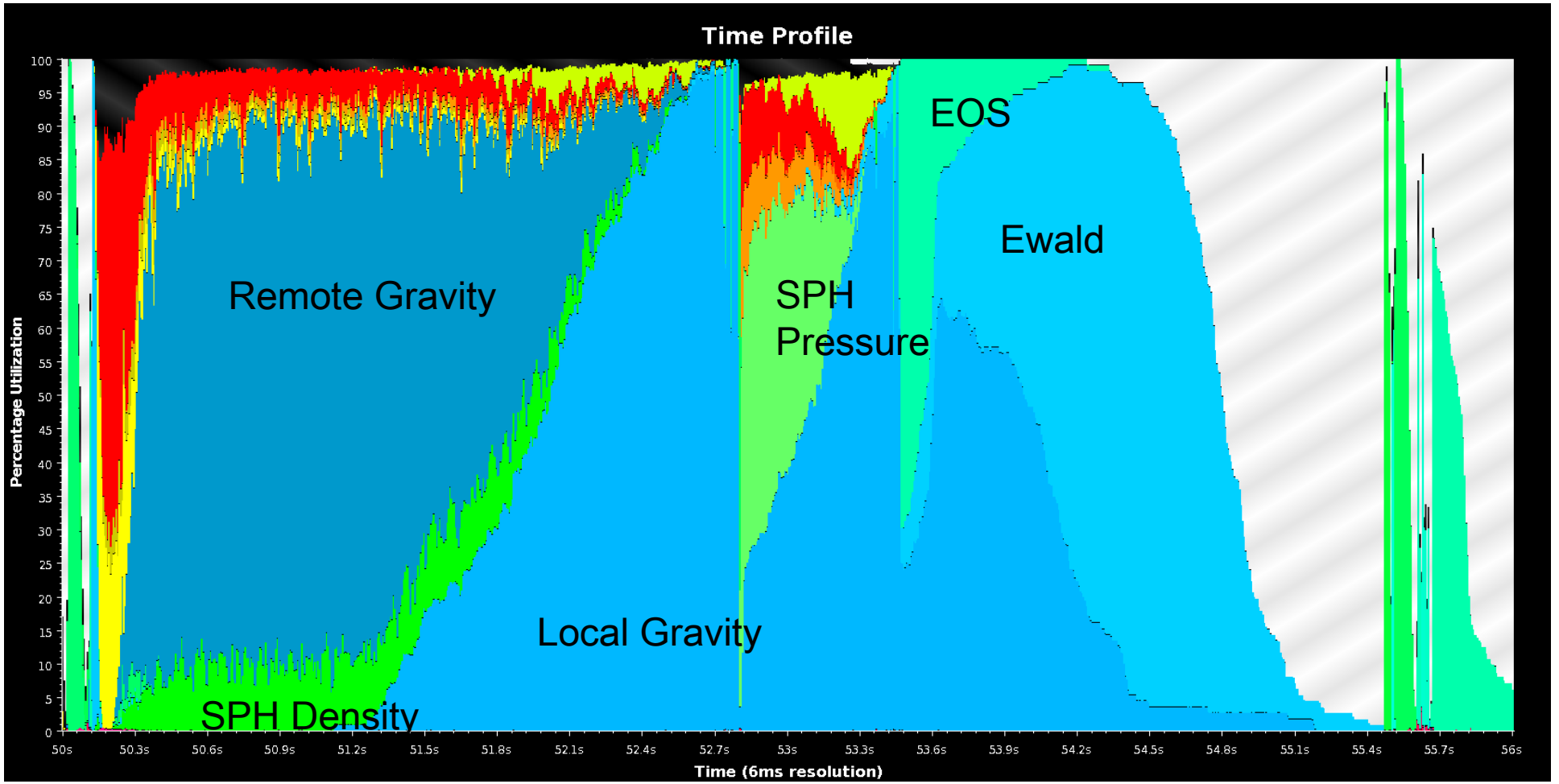
Single Stepping



Multi Stepping

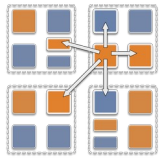


Overlapping of Phases



ChaNGA Design and Optimization: Lessons

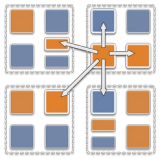
- Many details in: https://charm.cs.illinois.edu/papers/14-30
- Rethink relationship to processors
  - Oct-trees, overdecomposition
- Don't take performance and scaling losses for granted
  - Rage against them!
  - Detailed analysis , in part with projections, helps
  - Request-clustering was unexpected problem, needed a clever solution
- Other optimizations not discussed here:
  - Task-based within node balancing
  - SMP cache (more in ParaTreeT)



PPL UIUC

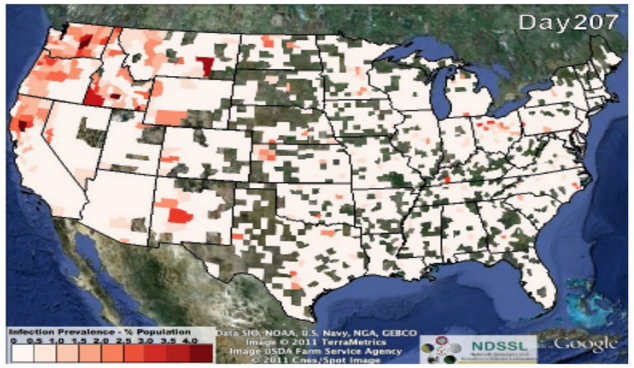
Episimdemics

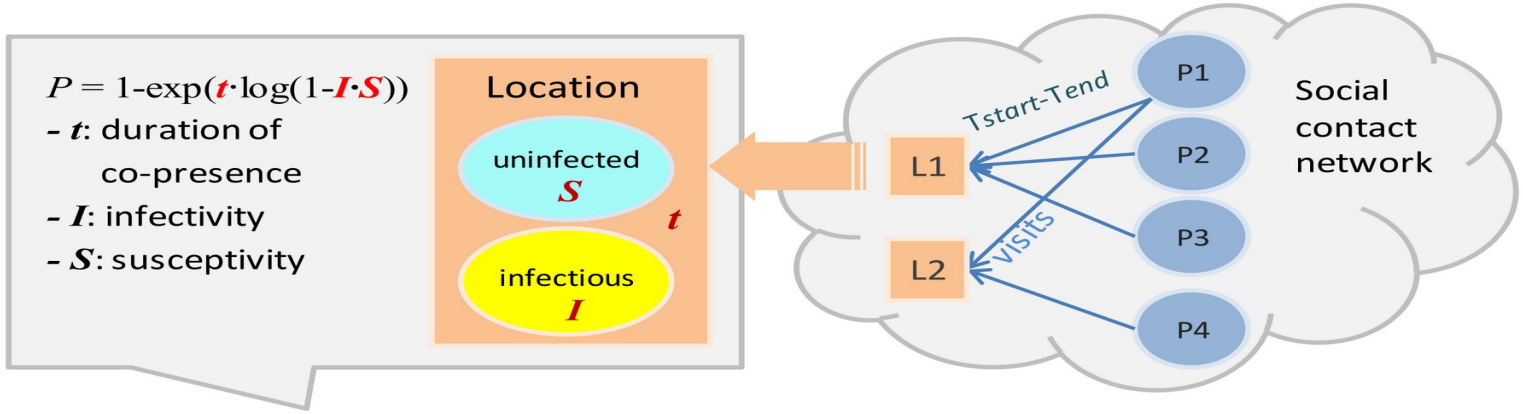
- Simulation of spread of contagion
  - Code by Madhav Marathe, Keith Bisset, .. Vtech
  - Original was in MPI
- Converted to Charm++
  - Benefits: asynchronous reductions improved performance considerably



PPL UIUC

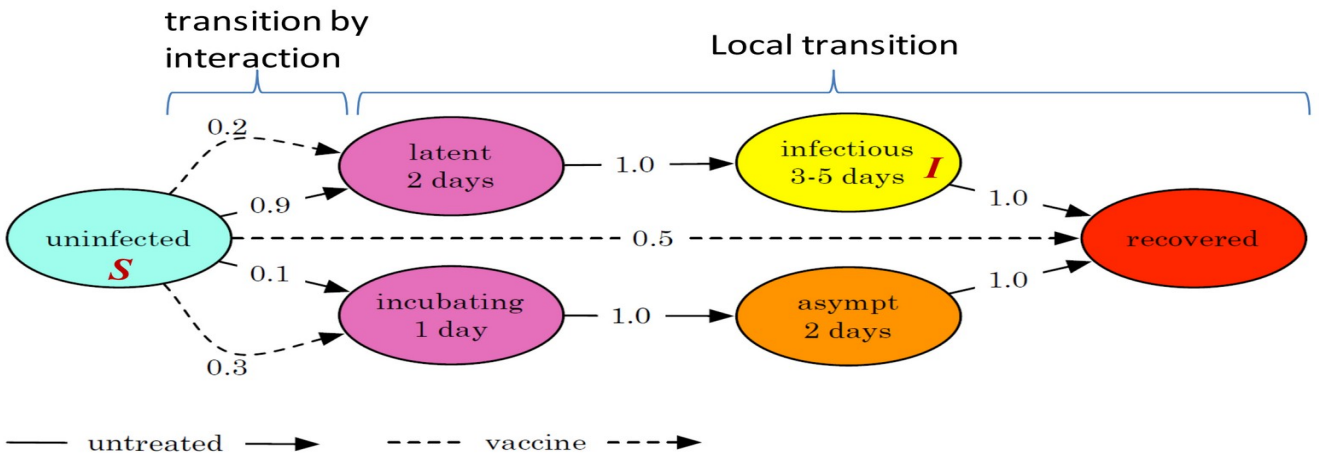
Simulating contagion over dynamic networks





EpiSimdemics[1]

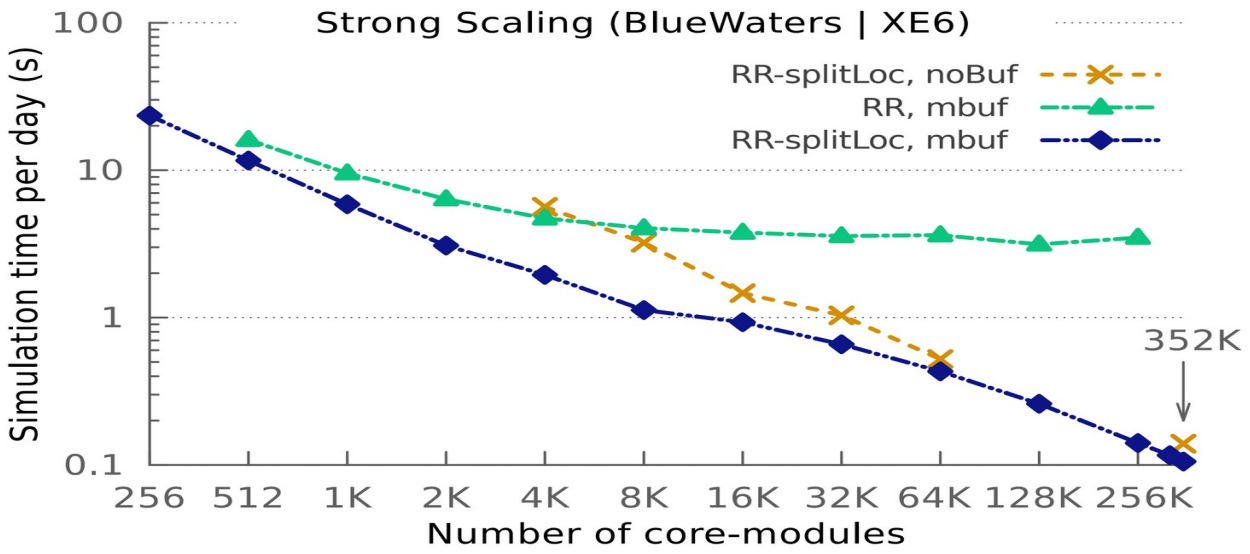
- Agent-based
- Realistic population data
- Intervention[2]
- Co-evolving network, behavior and policy[2]

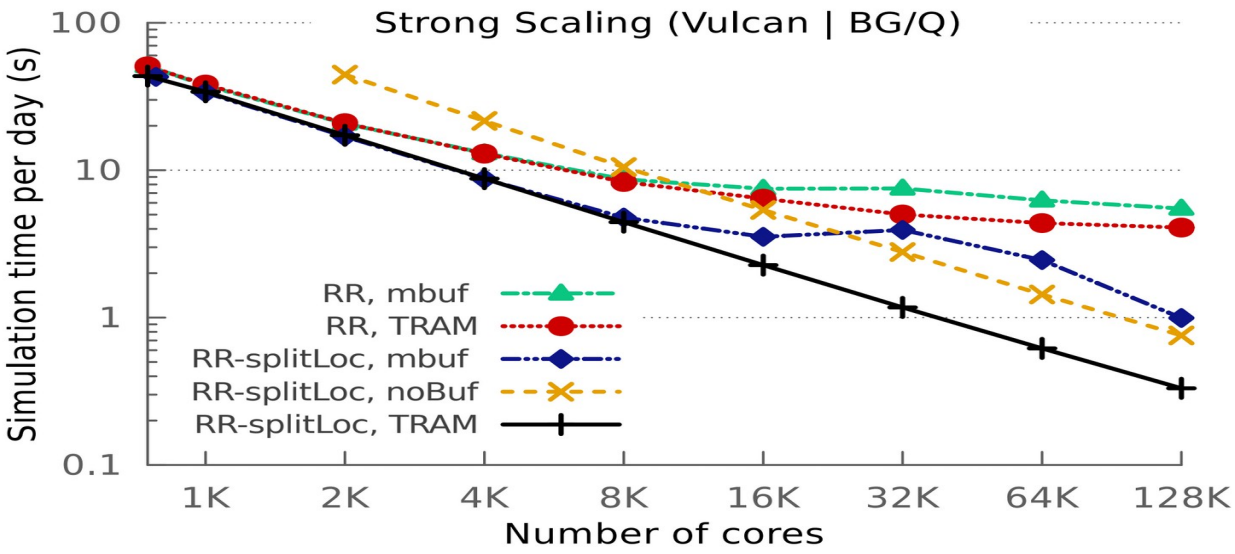


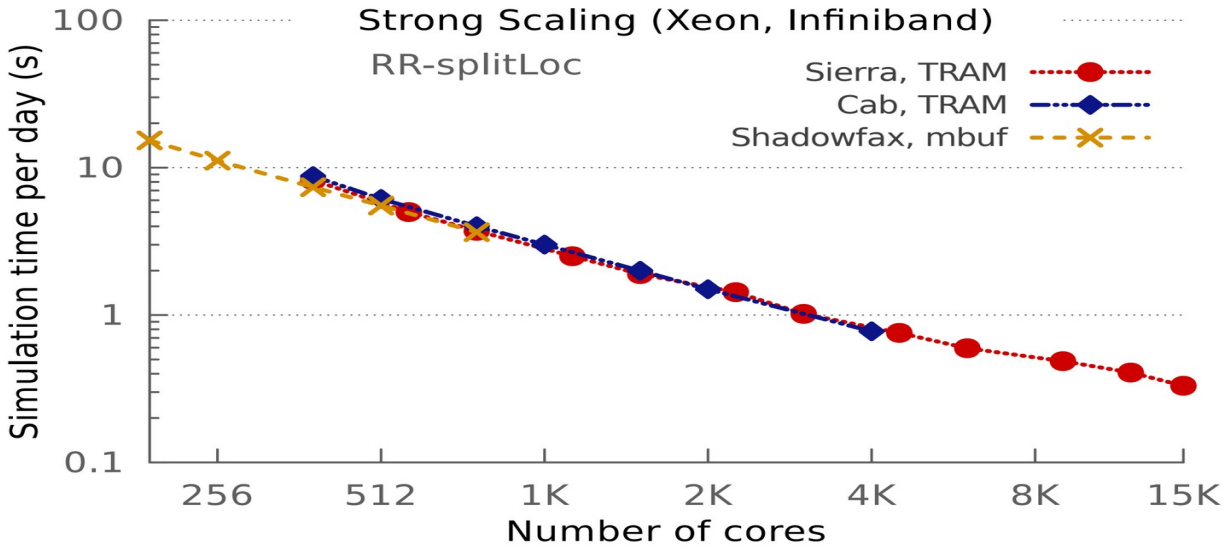
[1] C. Barrett et al.,"EpiSimdemics: An Efficient Algorithm for Simulating the Spread of Infectious Disease over Large Realistic Social Networks," SC08

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

Strong scaling performance with the largest data set







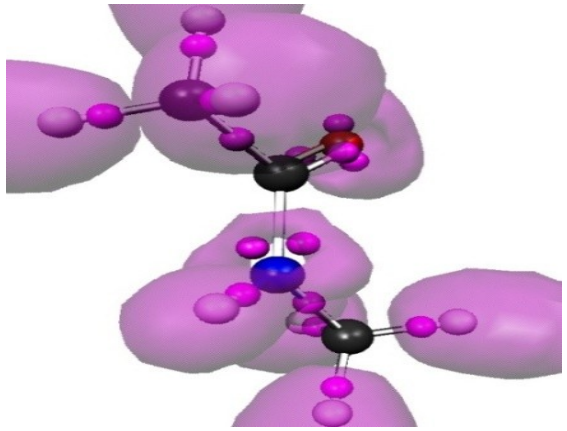
- Contiguous US population data
- **XE6**: **the largest scale** (**352K cores**)
- **BG/Q**: good scaling up to 128K cores
- Strong scaling helps timely reaction to pandemic

OpenAtom

Car-Parinello Molecular Dynamics

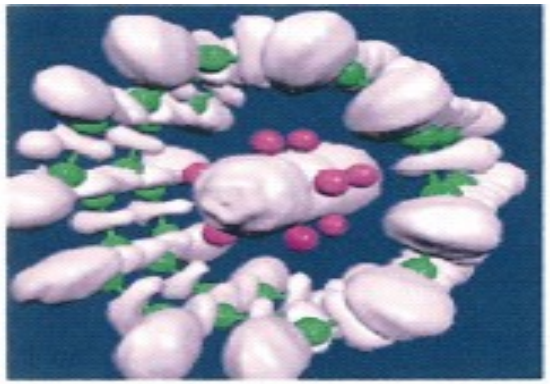
NSF ITR 2001-2007, IBM, DOE,NSF

Molecular Clusters :

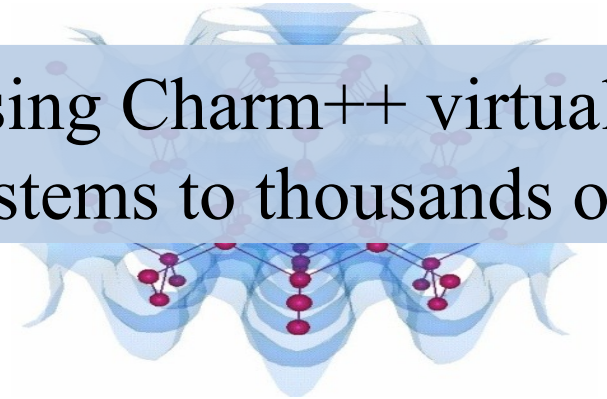


Recent NSF SSI-SI2 grant
With
G. Martyna (IBM)
Sohrab Ismail-Beigi

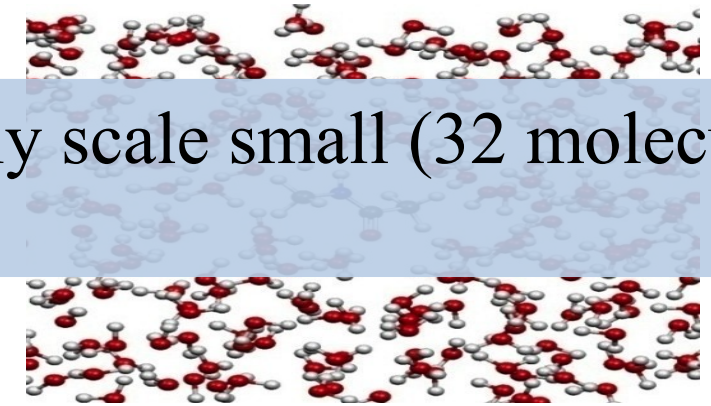
Nanowires:



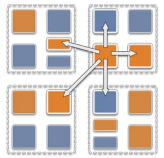
Semiconductor Surfaces:



3D-Solids/Liquids:

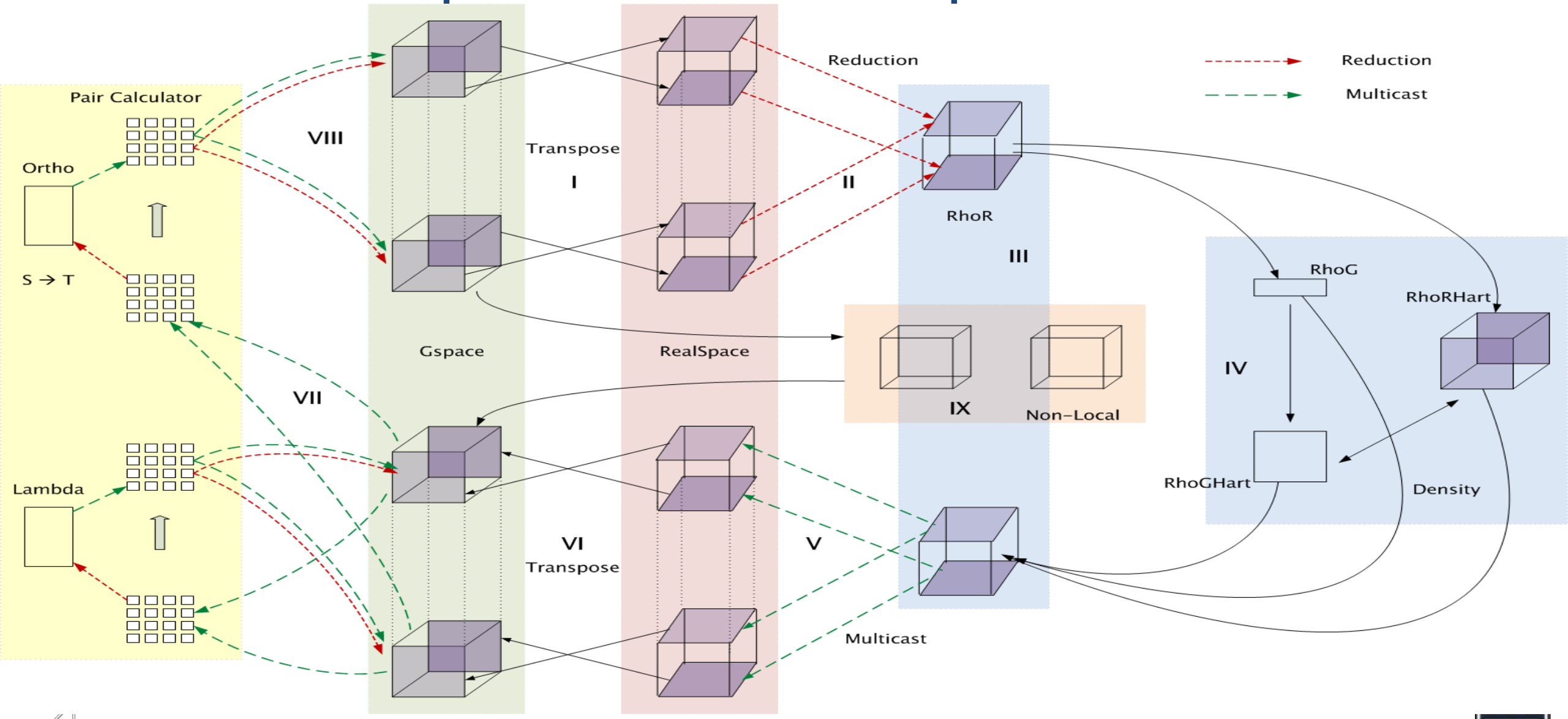


Using Charm++ virtualization, we can efficiently scale small (32 molecule) systems to thousands of processors

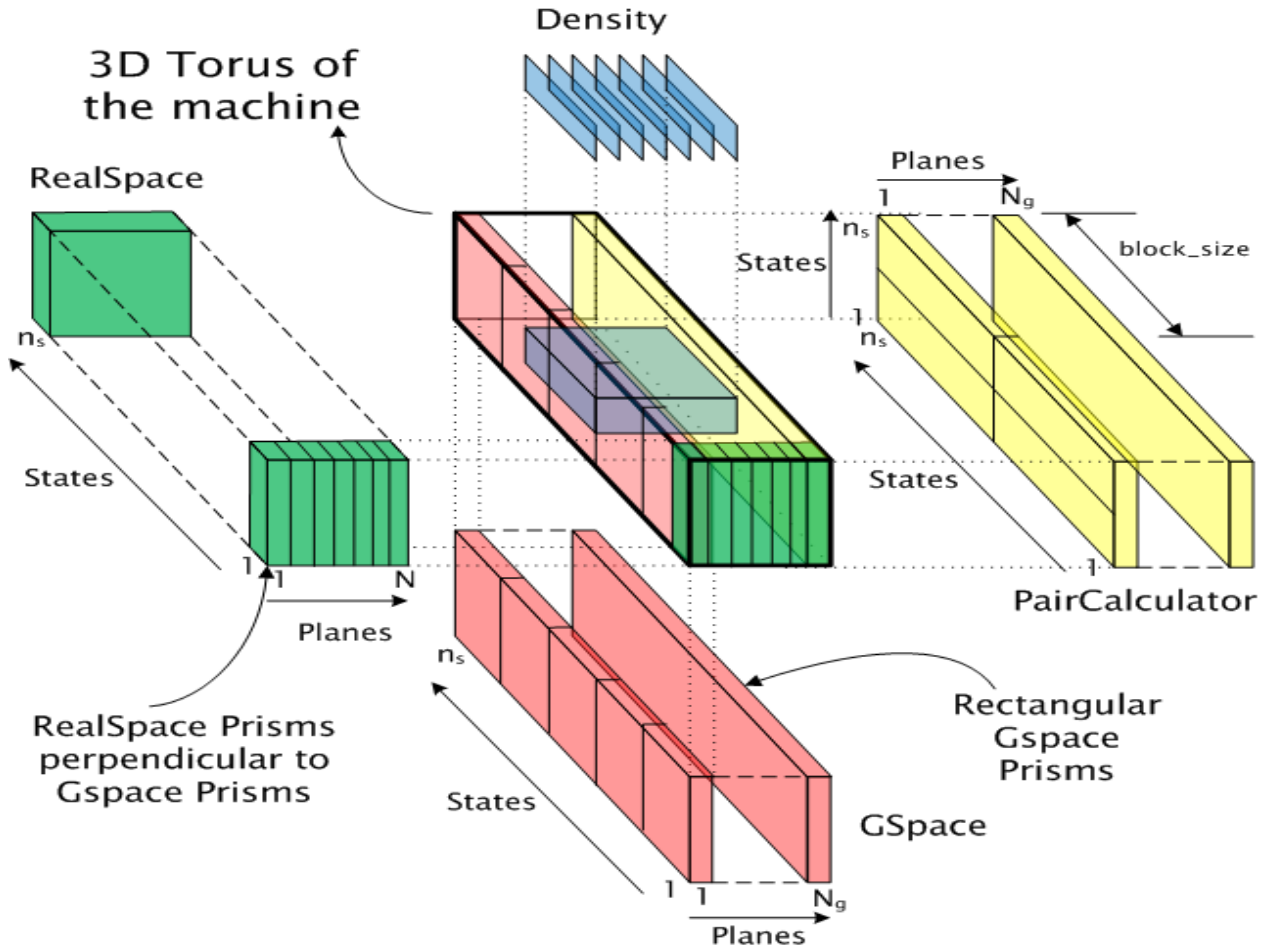


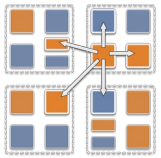
PPL UIUC

Decomposition and Computation Flow



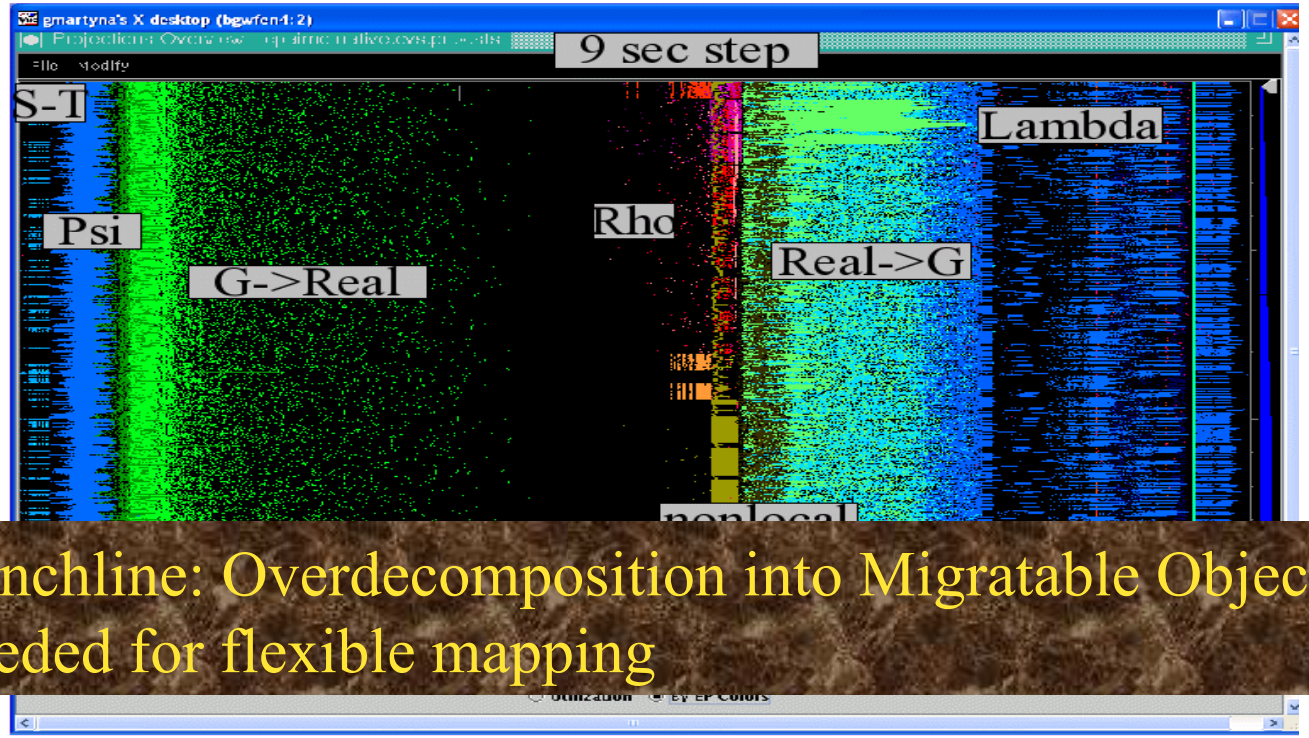
Topology Aware Mapping of Objects

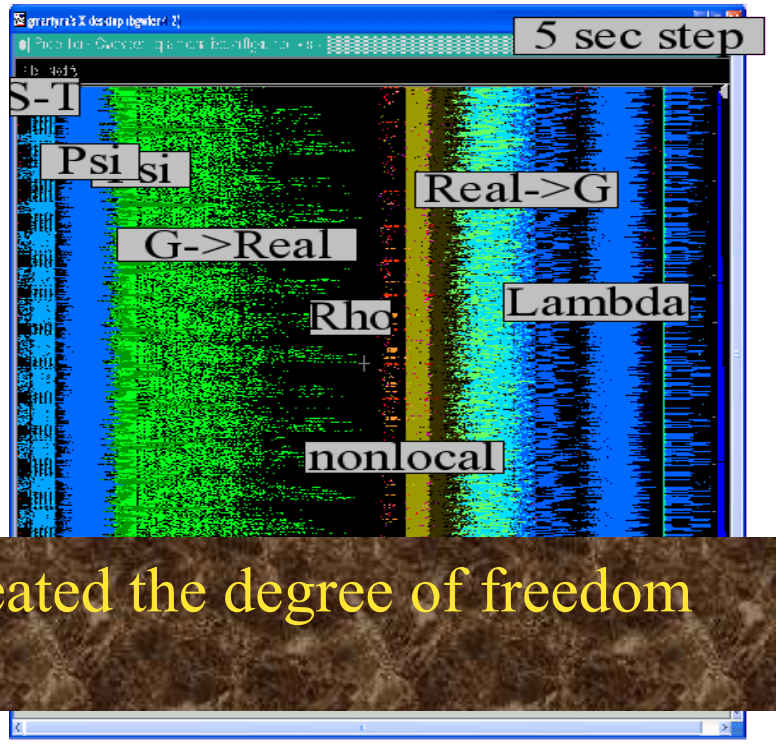




PPL UIUC

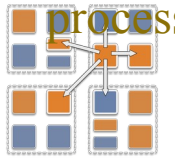
Improvements by topological aware mapping of computation to processors





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)

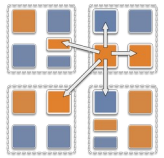


PPL UIUC

MiniApps

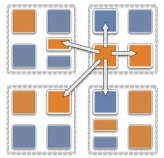
Available at: http://charmplusplus.org/miniApps/

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

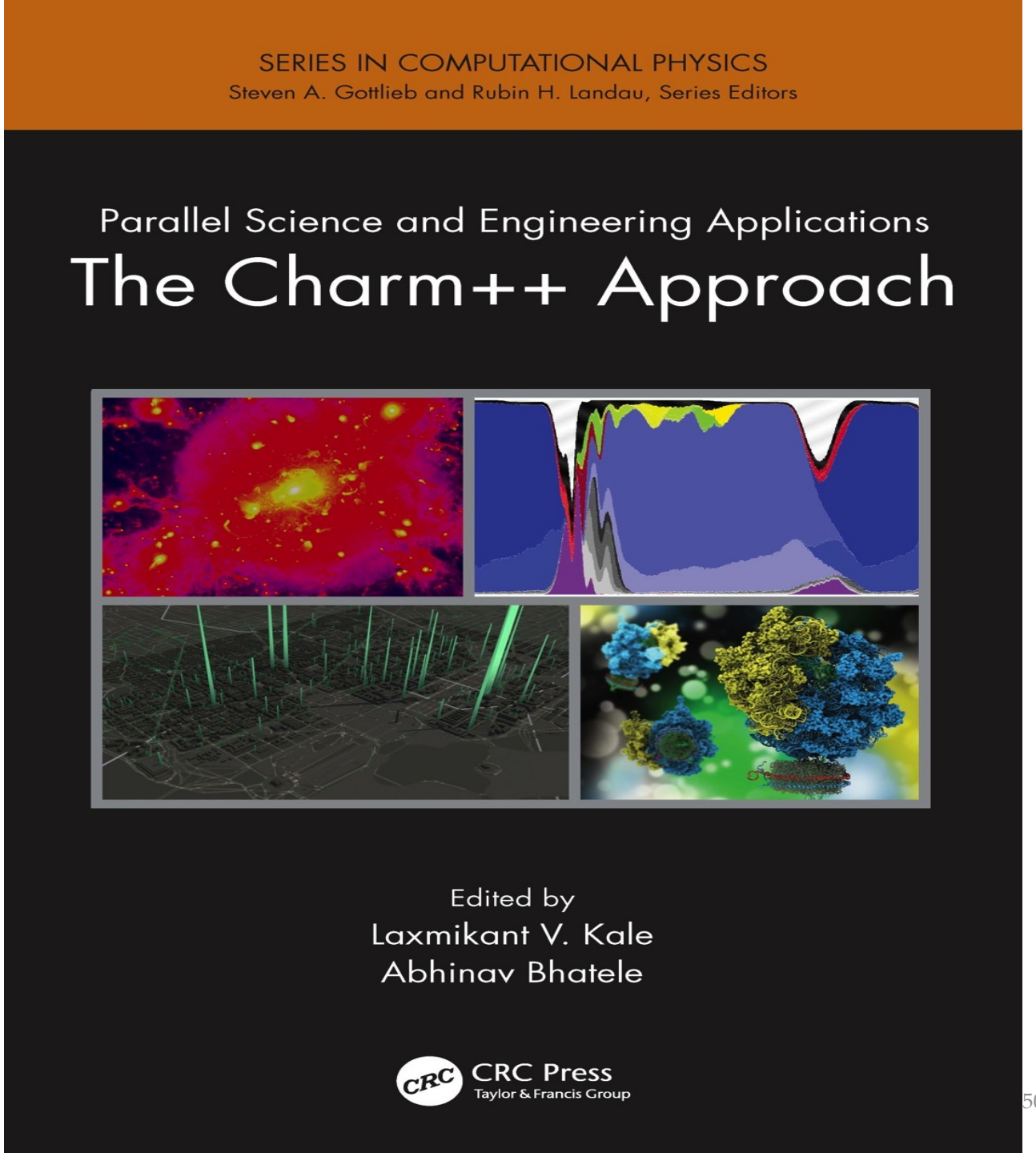


More MiniApps

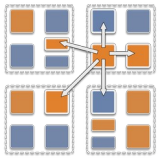
Mini-App	Features	Machine	Max cores
1D FFT	Interoperable with MPI	BG/P BG/Q	65,536 16,384
Random Access	TRAM	BG/P BG/Q	131,072 16,384
Dense LU	SDAG	XT5	8,192
Sparse Triangular Solver	SDAG	BG/P	512
GTC	SDAG	BG/Q	1,024
SPH		Blue Waters	-



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



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



PPL UIUC