5th Annual Workshop on Charm++ and Applications

Welcome and Introduction
“State of Charm++”

Laxmikant Kale

http://charm.cs.uiuc.edu

Parallel Programming Laboratory

Department of Computer Science

University of Illinois at Urbana Champaign

A Glance at History

- 1987: Chare Kernel arose from parallel Prolog work
  - Dynamic load balancing for state-space search, Prolog, ..
- 1992: Charm++
- 1994: Position Paper:
  - Application Oriented yet CS Centered Research
  - NAMD : 1994, 1996
- Charm++ in almost current form: 1996-1998
  - Chare arrays,
  - Measurement Based Dynamic Load balancing
- 1997 : Rocket Center: a trigger for AMPI
- 2001: Era of ITRs:
  - Quantum Chemistry collaboration
  - Computational Astronomy collaboration: ChaNGa

Outline

- What is Charm++
  – and why is it good
- Overview of recent results
  – Language work: raising the level of abstraction
  – Domain Specific Frameworks: ParFUM
    - Guebelle: crack propoagation
    - Haber: spae-time meshing
  – Applications
    - NAMD (picked by NSF, new scaling results to 32k procs.)
    - ChaNGa: released, gravity performance
    - LeanCP:
  – Use at National centers
  – BigSim
  – Scalable Performance tools
  – Scalable Load Balancers
  – Fault tolerance
  – Cell, GPGPUs, ..
  – Upcoming Challenges and opportunities:
    - Multicore
    - Funding

PPL Mission and Approach

- To enhance *Performance and Productivity* in programming complex parallel applications
  - Performance: scalable to thousands of processors
  - Productivity: of human programmers
  - Complex: irregular structure, dynamic variations
- Approach: *Application Oriented yet CS centered research*
  - Develop enabling technology, for a wide collection of apps.
  - Develop, use and test it in the context of real applications
- How?
  - Develop novel Parallel programming techniques
  - Embody them into easy to use abstractions
  - So, application scientist can use advanced techniques with ease
  - Enabling technology: reused across many apps

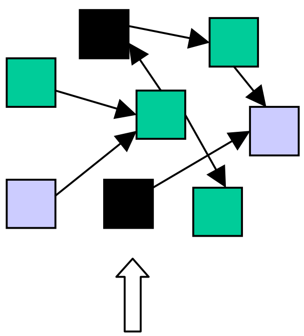
Migratable Objects (aka Processor Virtualization)

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

Runtime: **Assigns VPs to processors**

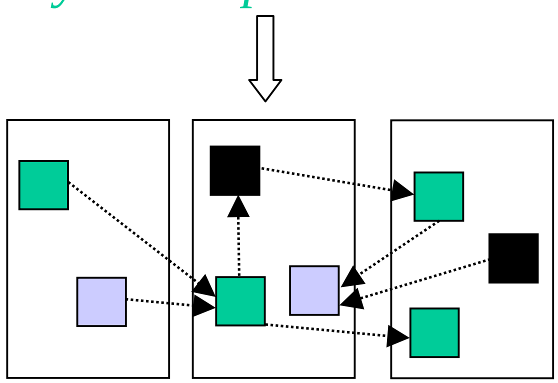
Enables *adaptive runtime strategies*

Implementations: Charm++, AMPI



User View

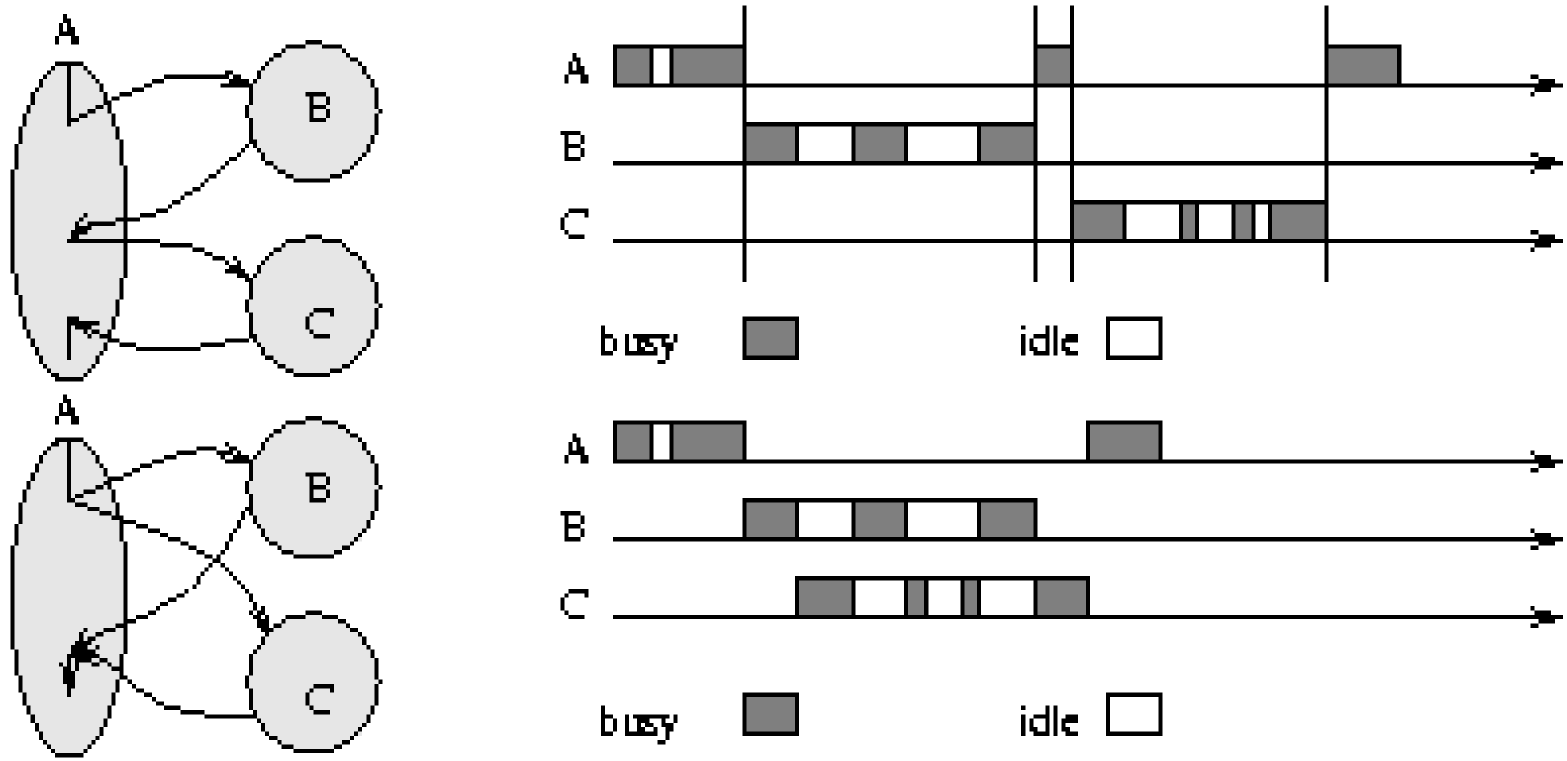
System implementation



Benefits

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

Adaptive overlap and modules



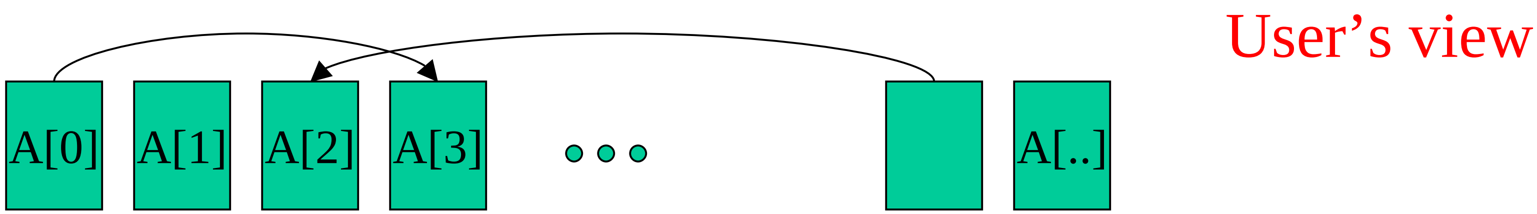
SPMD and Message-Driven Modules

(From A. Gursoy, *Simplified expression of message-driven programs and quantification of their impact on performance*, Ph.D Thesis, Apr 1994.)

Modularity, Reuse, and Efficiency with Message-Driven Libraries: Proc. of the Seventh SIAM Conference on Parallel Processing for Scientific Computing, San Fransisco, 1995

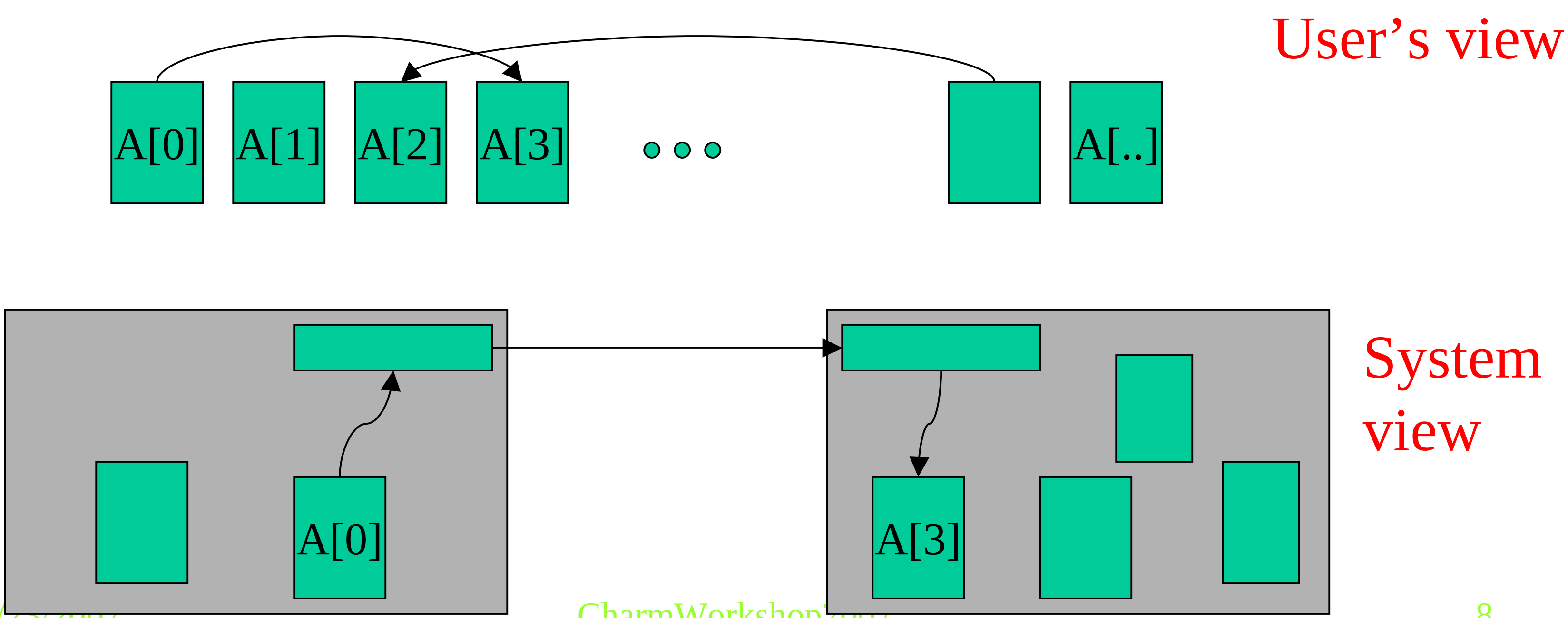
Realization: Charm++'s Object Arrays

- A collection of data-driven objects
  - With a single global name for the collection
  - Each member addressed by an index
    - [sparse] 1D, 2D, 3D, tree, string, ...
  - Mapping of element objects to procS handled by the system



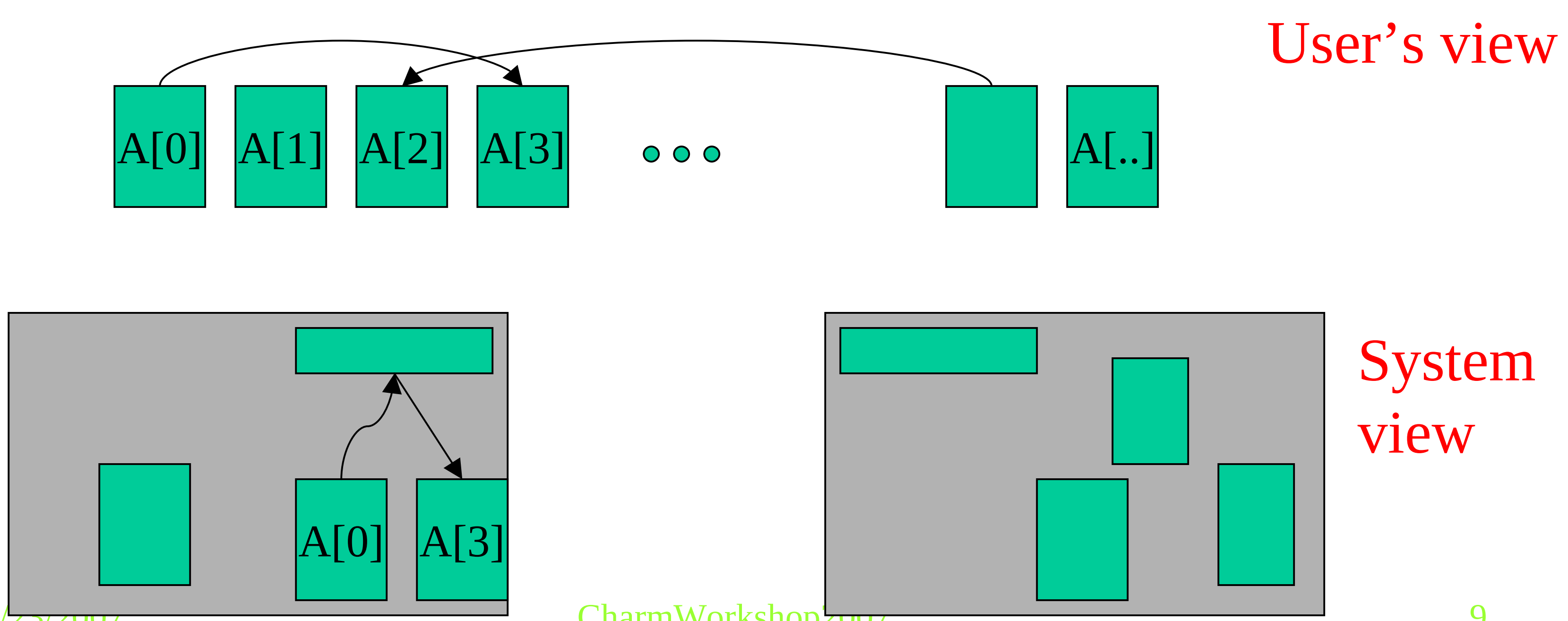
Realization: Charm++'s Object Arrays

- A collection of data-driven objects
  - With a single global name for the collection
  - Each member addressed by an index
    - [sparse] 1D, 2D, 3D, tree, string, ...
  - Mapping of element objects to procS handled by the system

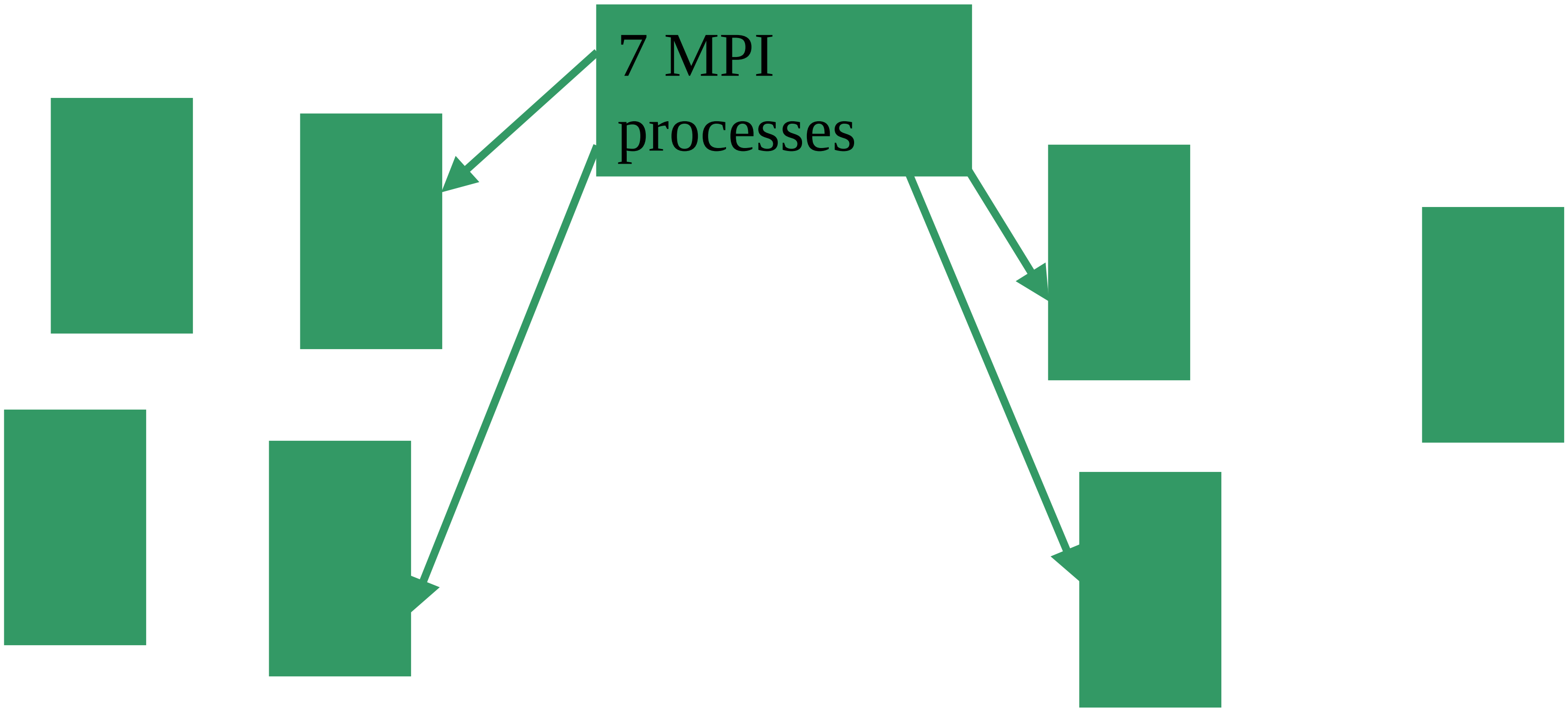


Charm++: Object Arrays

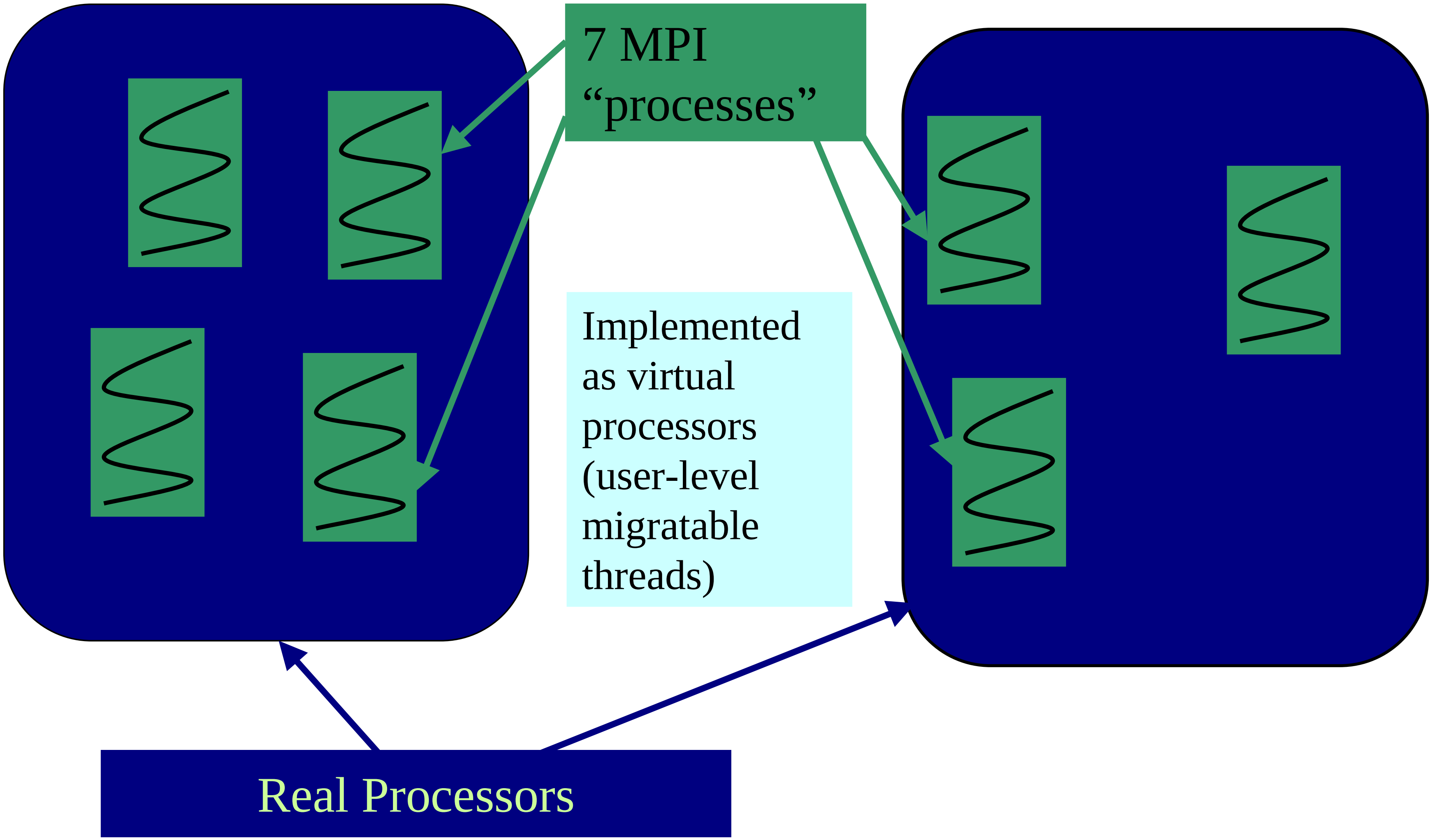
- A collection of data-driven objects
  - With a single global name for the collection
  - Each member addressed by an index
    - [sparse] 1D, 2D, 3D, tree, string, ...
  - Mapping of element objects to procS handled by the system

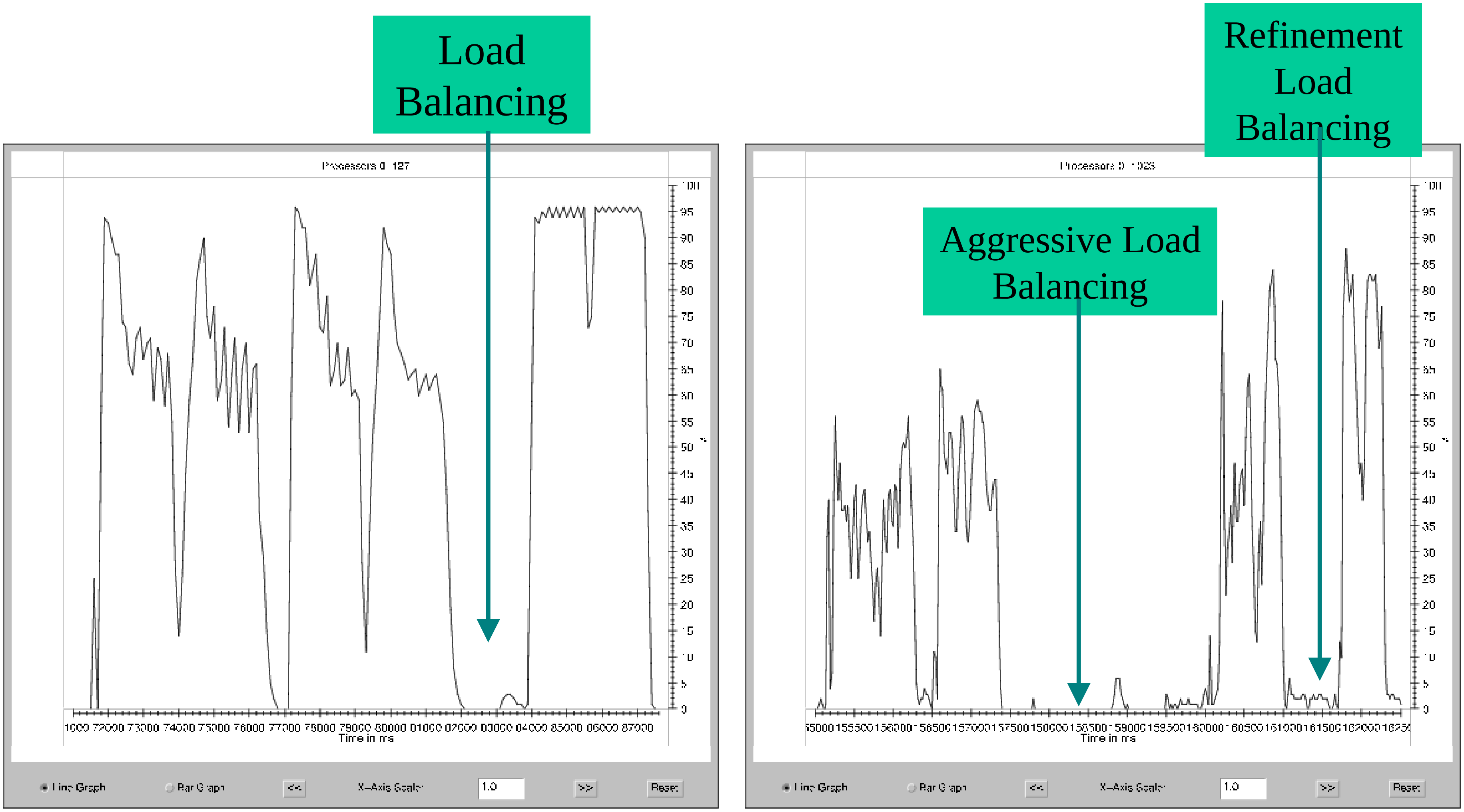


AMPI: Adaptive MPI



AMPI: Adaptive MPI





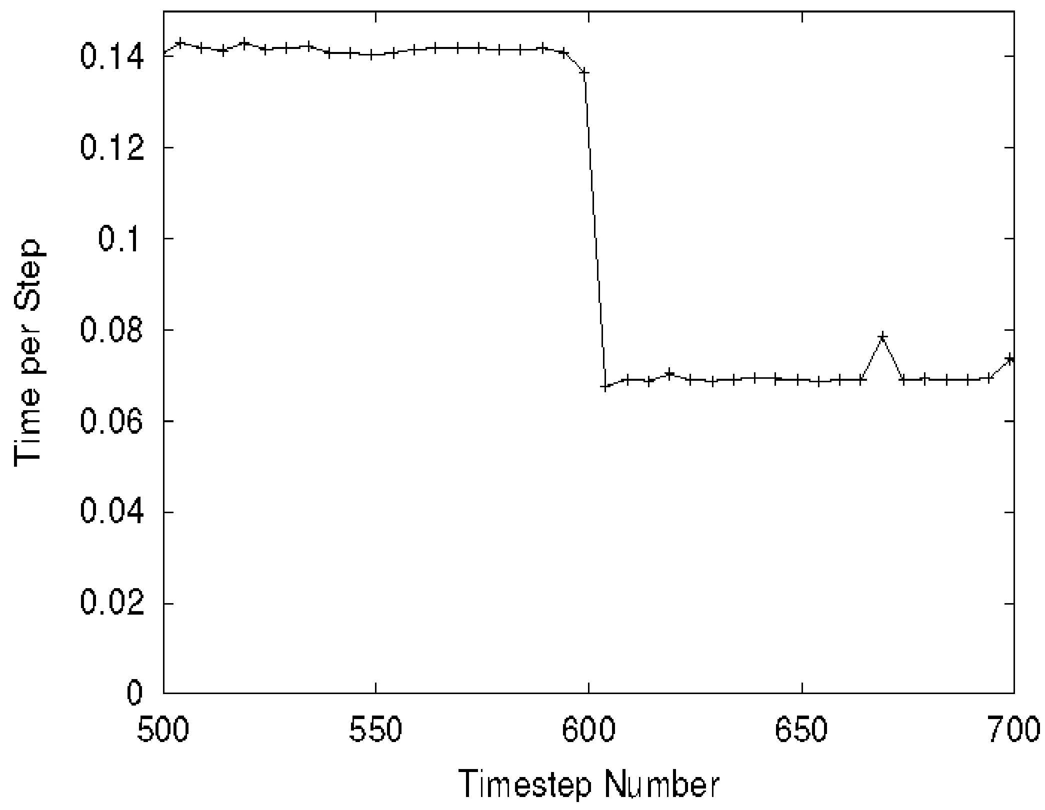
Processor Utilization against Time on 128 and 1024 processors

On 128 processor, a single load balancing step suffices, but

On 1024 processors, we need a “refinement” step.

Shrink/Expand

- Problem: Availability of computing platform may change
- Fitting applications on the platform by object migration



Time per step for the million-row CG solver on a 16-node cluster
Additional 16 nodes available at step 600

So, Whats new?

New Higher Level Abstractions

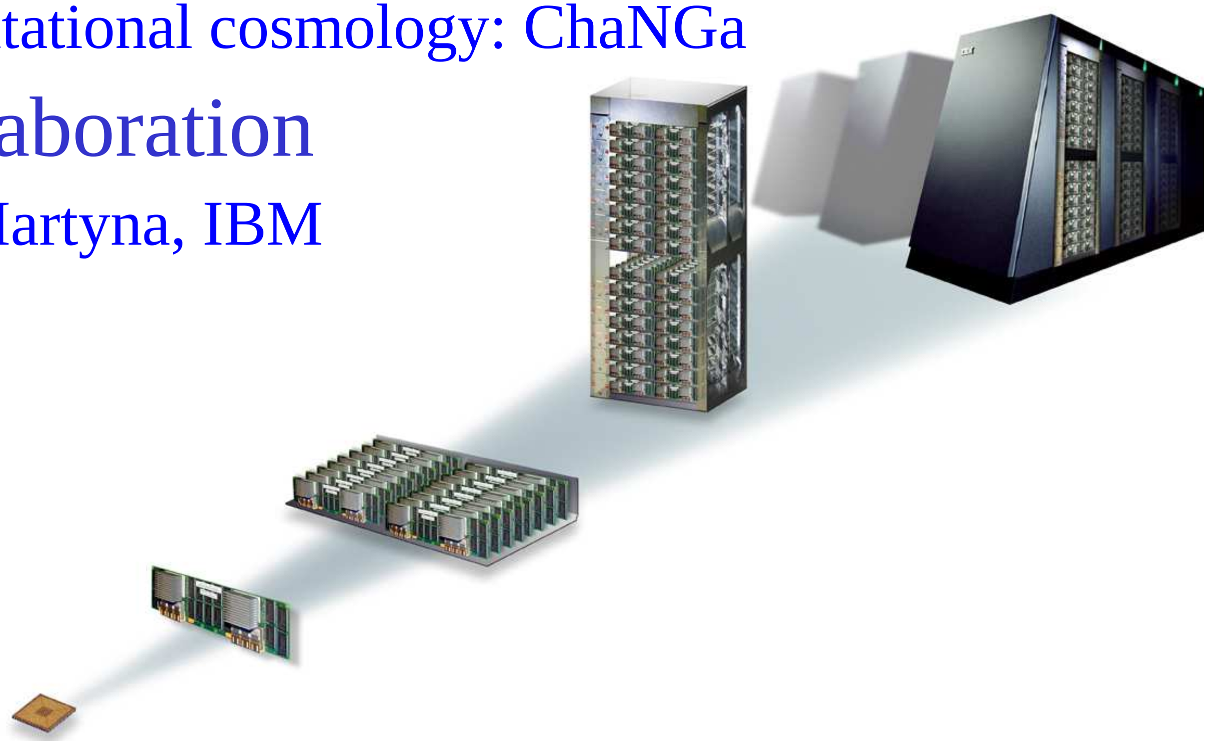
- Previously: Multiphase Shared Arrays
  - Provides a disciplined use of global address space
  - Each array can be accessed only in one of the following modes:
    - ReadOnly, Write-by-One-Thread, Accumulate-only
  - Access mode can change from phase to phase
  - Phases delineated by per-array “sync”
- Charisma++: Global view of control
  - Allows expressing global control flow in a charm program
  - Separate expression of parallel and sequential
  - Functional Implementation (Chao Huang PhD thesis)
  - LCR’04, HPDC’07

Multiparadigm Interoperability

- Charm++ supports concurrent composition
- Allows multiple module written in multiple paradigms to cooperate in a single application
- Some recent paradigms implemented:
  - ARMCI (for Global Arrays)
- Use of Multiparadigm programming
  - You heard yesterday how ParFUM made use of multiple paradigms effetively

Blue Gene Provided a Showcase.

- Co-operation with Blue Gene team
  - Sameer Kumar joins BlueGene team
- BGW days competetion
  - 2006: Computer Science day
  - 2007: Computational cosmology: ChaNGa
- LeanCP collaboration
  - with Glenn Martyna, IBM



Cray and PSC Warms up

- 4000 fast processors at PSC
- 12,500 processors at ORNL
- Cray support via a gift grant

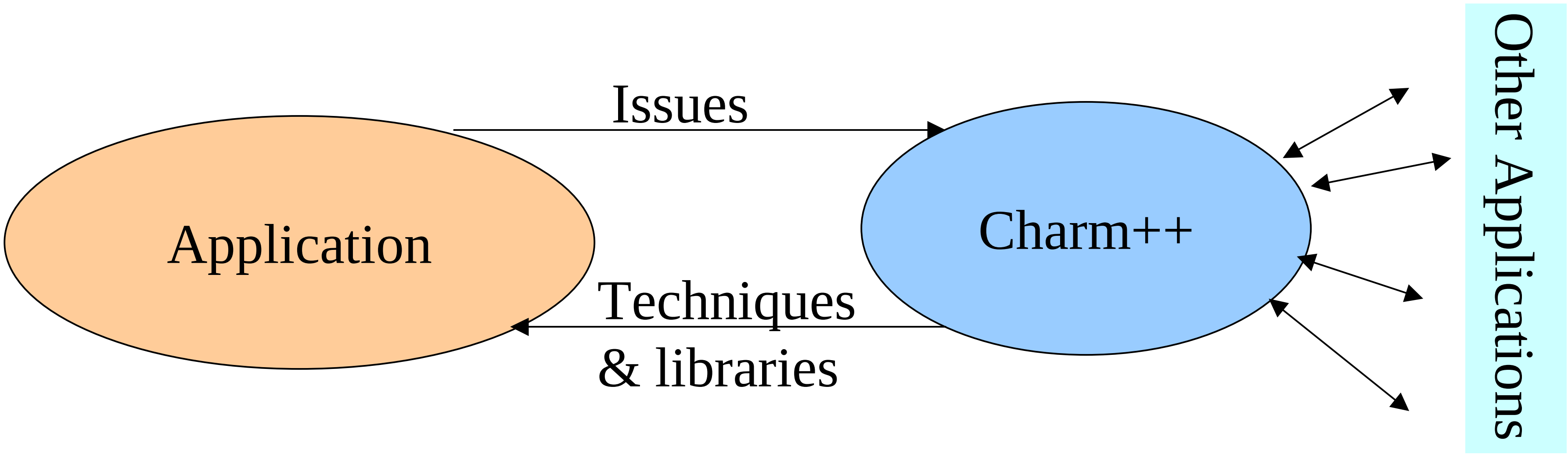
IBM Power7 Team

- Collaborations begun with NSF Track 1 proposal

Censored.
NDA

Our Applications Achieved Unprecedented Speedups

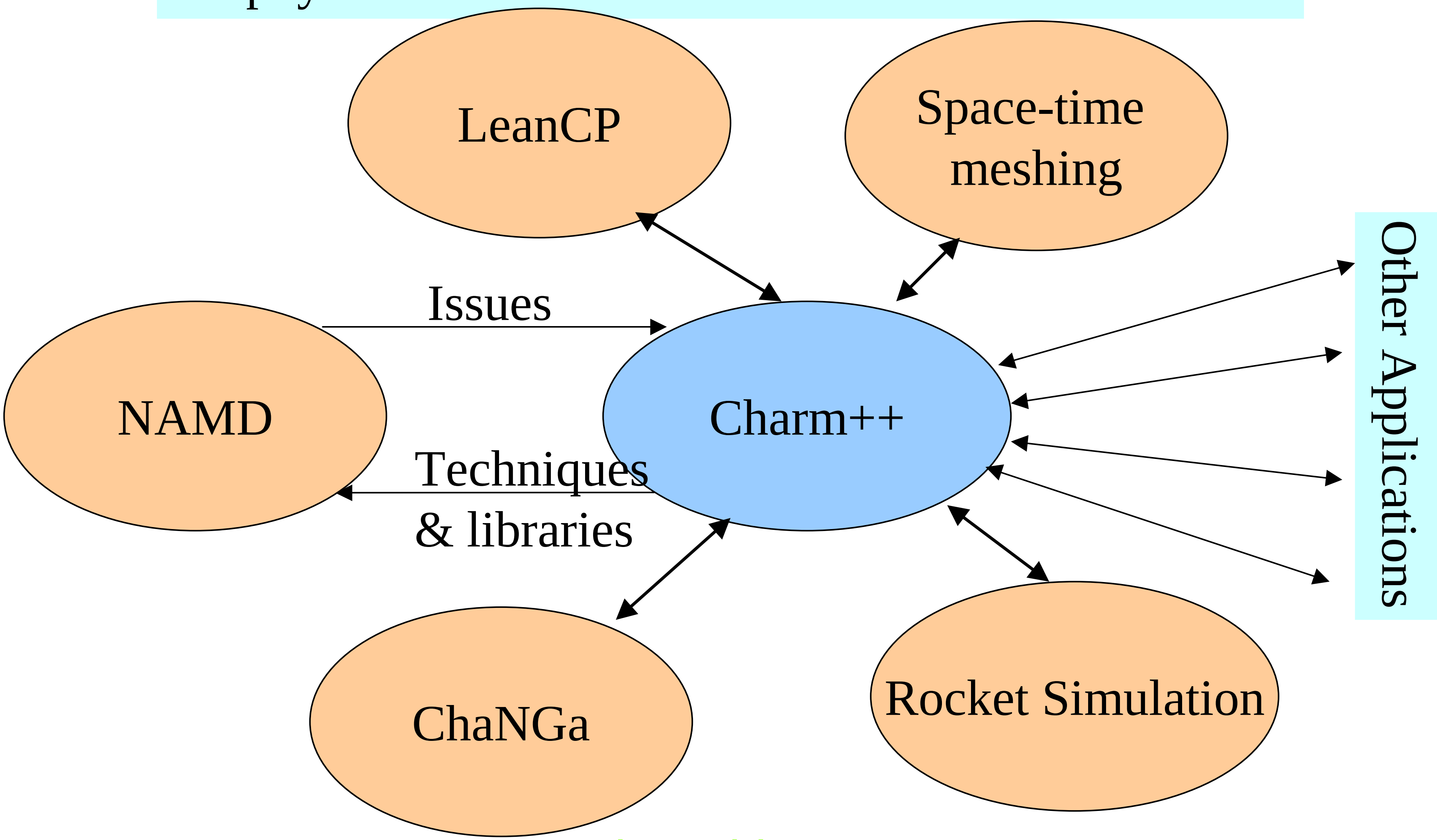
Applications and Charm++



Synergy between Computer Science Research and Biophysics has been beneficial to both

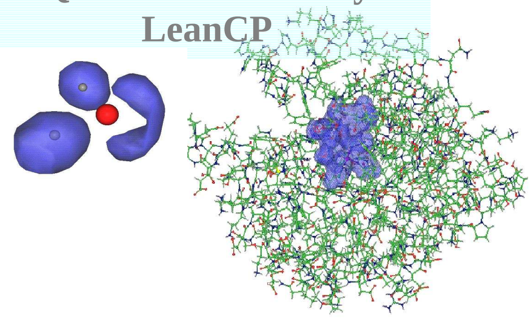
Charm++ and Applications

Synergy between Computer Science Research and Biophysics has been beneficial to both

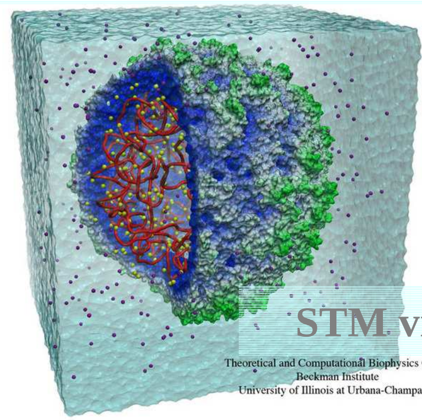


Develop abstractions in context of full-scale applications

Quantum Chemistry
LeanCP

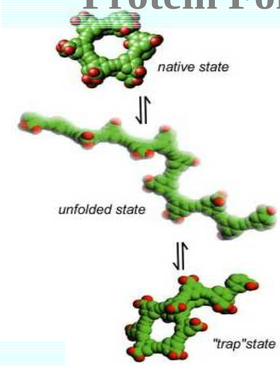


NAMD: Molecular Dynamics

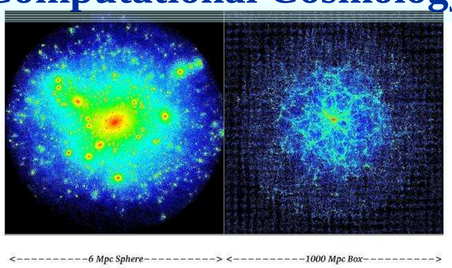


STM virus simulation

Protein Folding



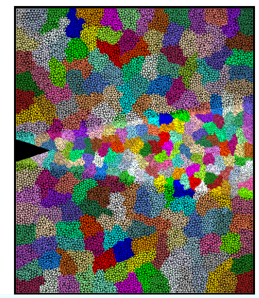
Computational Cosmology



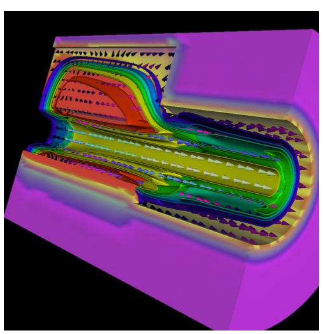
Parallel Objects,

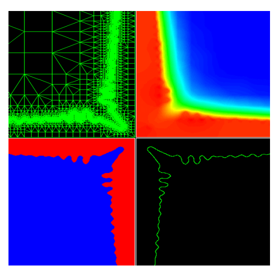
Adaptive Runtime System

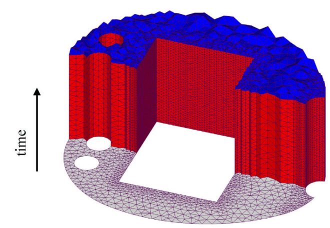
Libraries and Tools



Crack Propagation







Rocket Simulation

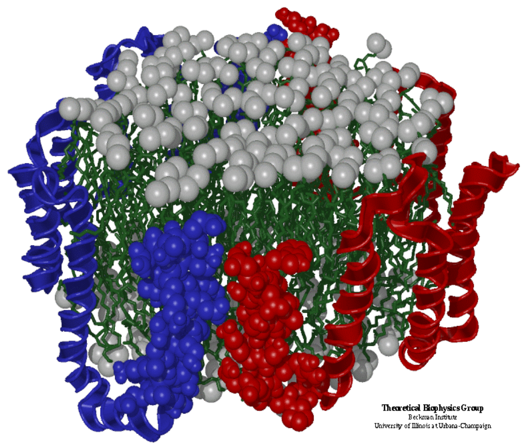
Dendritic Growth

Space-time meshes

The enabling CS technology of parallel objects and intelligent Runtime systems has led to several collaborative applications in CSE

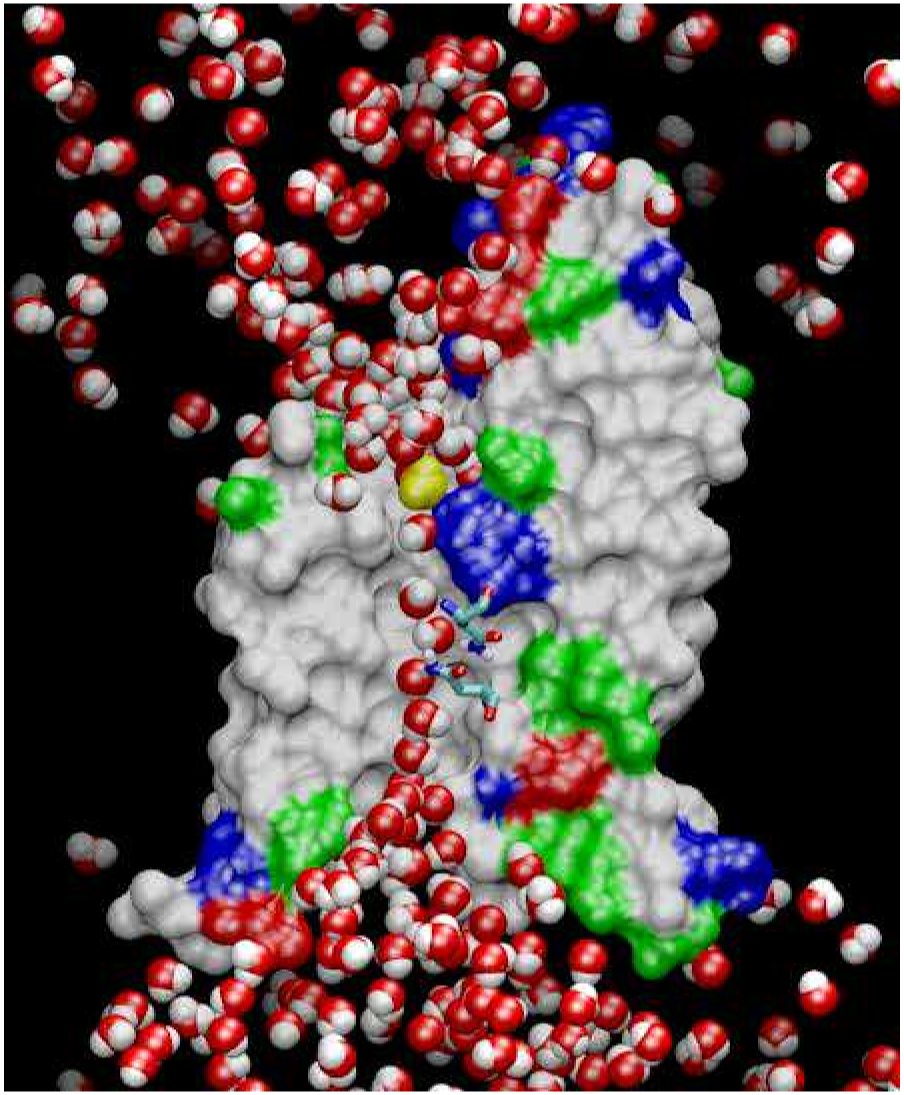
Molecular Dynamics in NAMD

- Collection of [charged] atoms, with bonds
  - Newtonian mechanics
  - Thousands of atoms (10,000 - 5000,000)
  - 1 femtosecond time-step, millions needed!
- At each time-step
  - Calculate forces on each atom
    - Bonds:
    - Non-bonded: electrostatic and van der Waal's
      - Short-distance: every timestep
      - Long-distance: every 4 timesteps using PME (3D FFT)
      - Multiple Time Stepping
  - Calculate velocities and advance positions



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

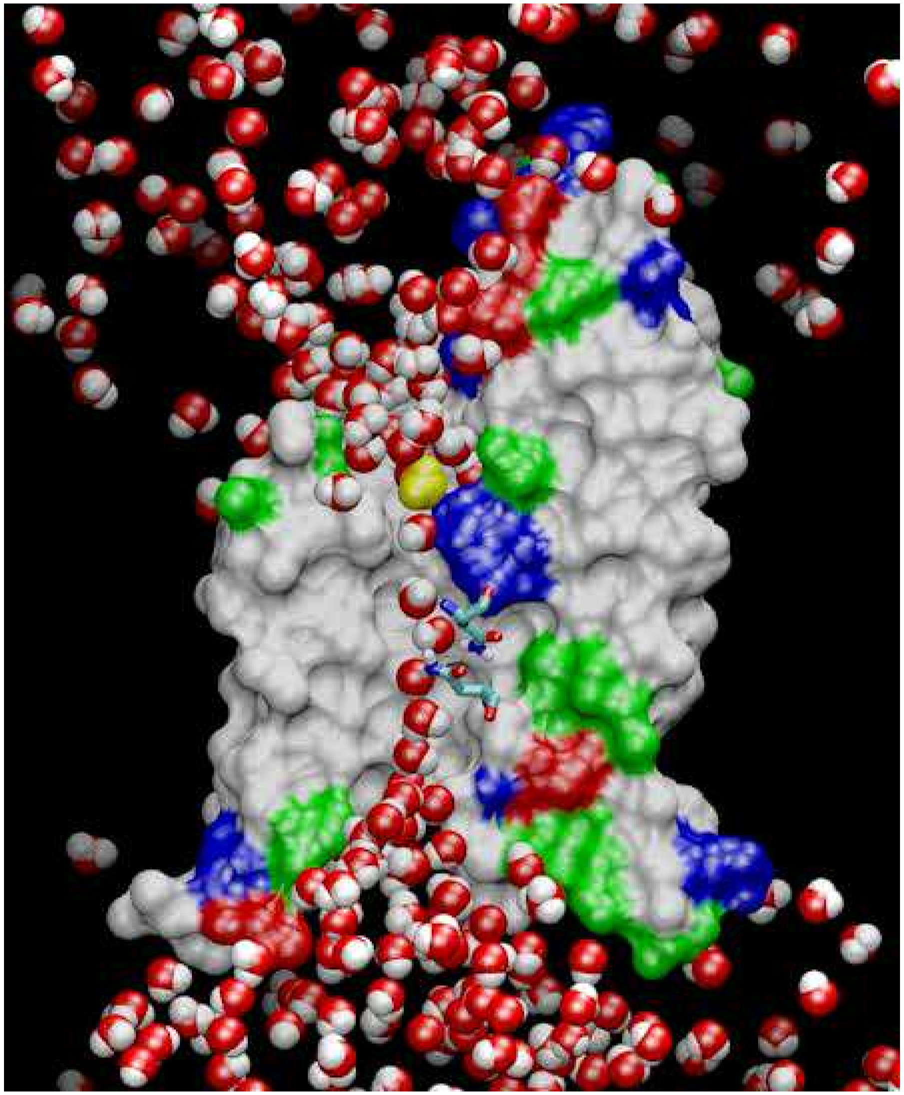
NAMD: A Production MD program



NAMD

- Fully featured program
- NIH-funded development
- Distributed free of charge (~20,000 registered users)
- Binaries and source code
- Installed at NSF centers
- User training and support
- Large published simulations

NAMD: A Production MD program



NAMD

- Fully featured program
- NIH-funded development
- Distributed free of charge (~20,000 registered users)
- Binaries and source code
- Installed at NSF centers
- User training and support
- Large published simulations

NAMD Design

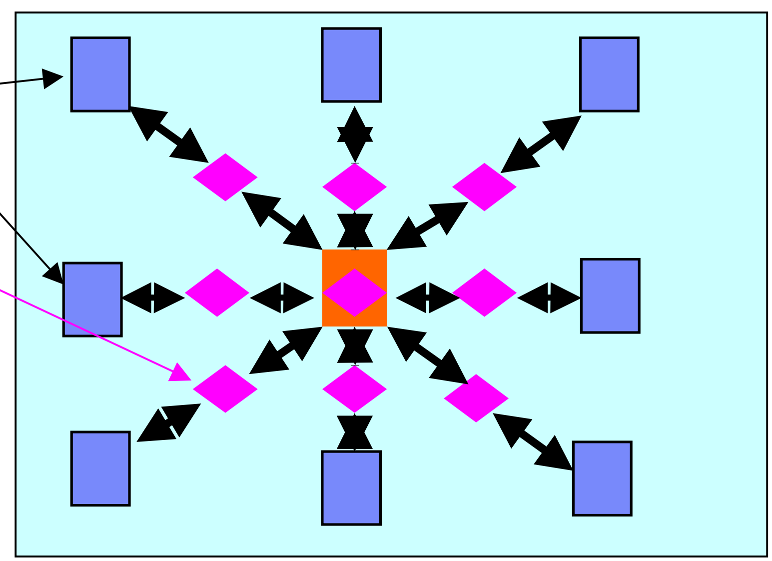
- Designed from the beginning as a parallel program
- Uses the Charm++ idea:
  - Decompose the computation into a large number of objects
  - Have an Intelligent Run-time system (of Charm++) assign objects to processors for dynamic load balancing

Hybrid of spatial and force decomposition:

•Spatial decomposition of atoms into cubes (called patches)

•For every pair of interacting patches, create one object for calculating electrostatic interactions

•Recent: Blue Matter, Desmond, etc. use this idea in some form



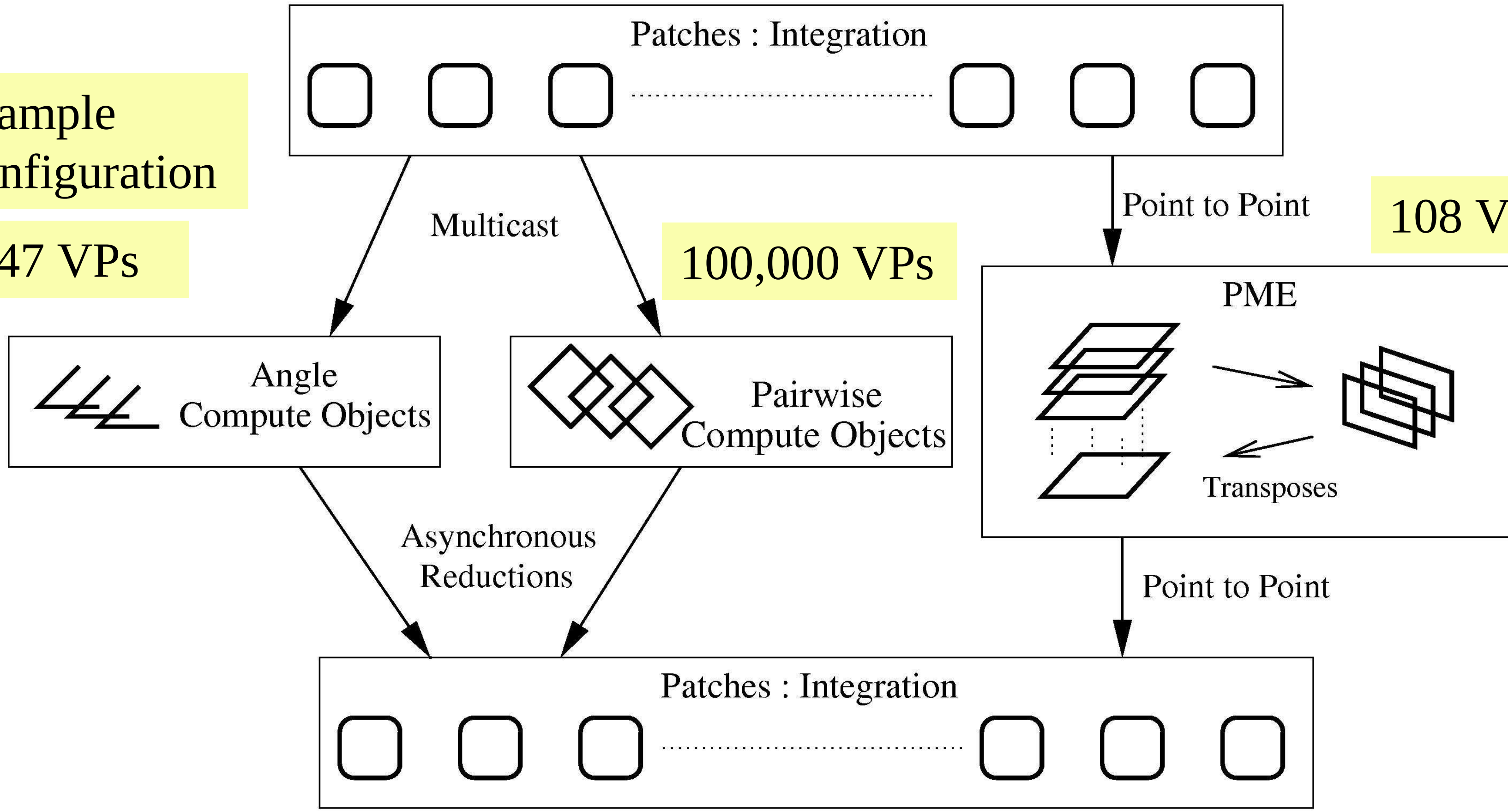
NAMD Parallelization using Charm++

Example Configuration

847 VPs

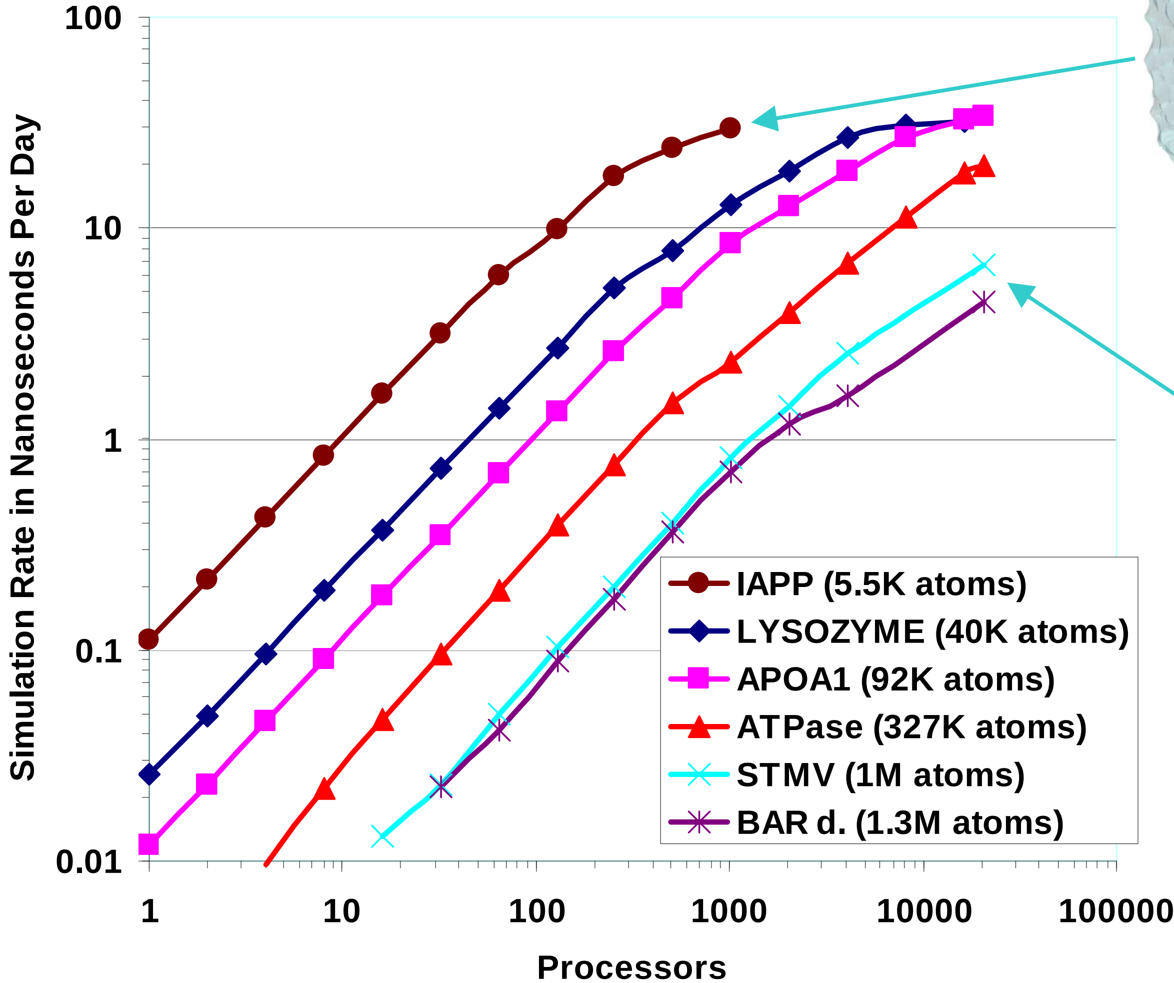
100,000 VPs

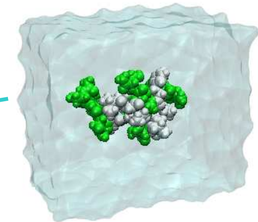
108 VPs



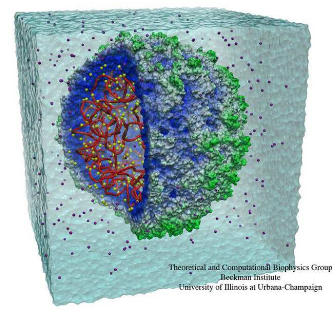
These 100,000 Objects (virtual processors, or VPs) are assigned to real processors by the Charm++ runtime system

Performance on BlueGene/L





IAPP simulation
(Rivera, Straub, BU)
at 20 ns per day
on 256 processors
1 us in 50 days



STMV simulation
at 6.65 ns per day
on 20,000 processors

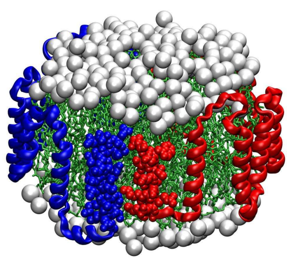
Comparison with Blue Matter

ApoLipoprotein-A1 (92K atoms)

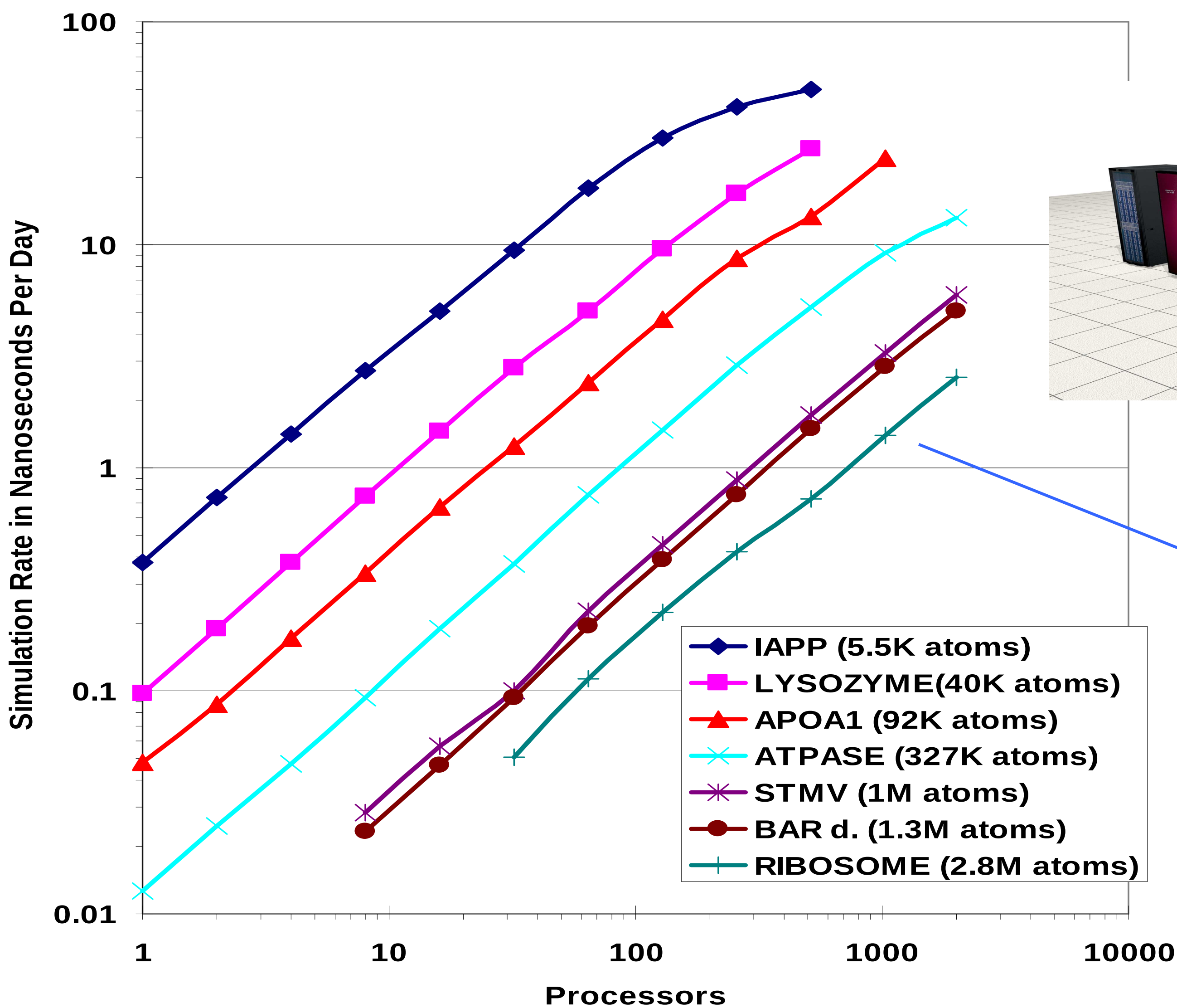
Nodes	512	1024	2048	4096	8192	16384	
Blue Matter (SC'06)	38.42	18.95	9.97	5.39	3.14	2.09	ms/step
NAMD	18.6	10.5	6.85	4.67	3.2	2.33	ms/step
NAMD (Virtual Node)	11.3	7.6	5.1	3.7	3.0		ms/step

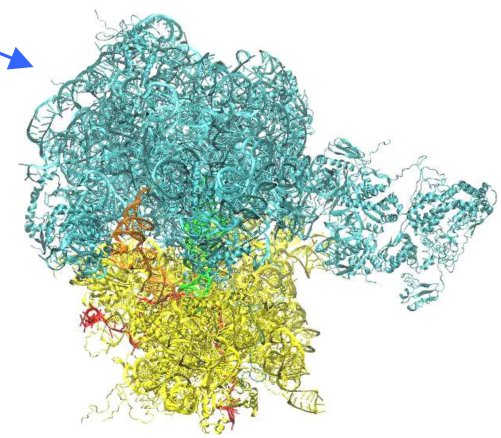
NAMD is about 1.8 times faster than Blue Matter on 1024 nodes (and 2.4 times faster with VN mode, where NAMD can use both processors on a node effectively).

However: Note that NAMD does PME every 4 steps.



Performance on Cray XT3





Computational Cosmology

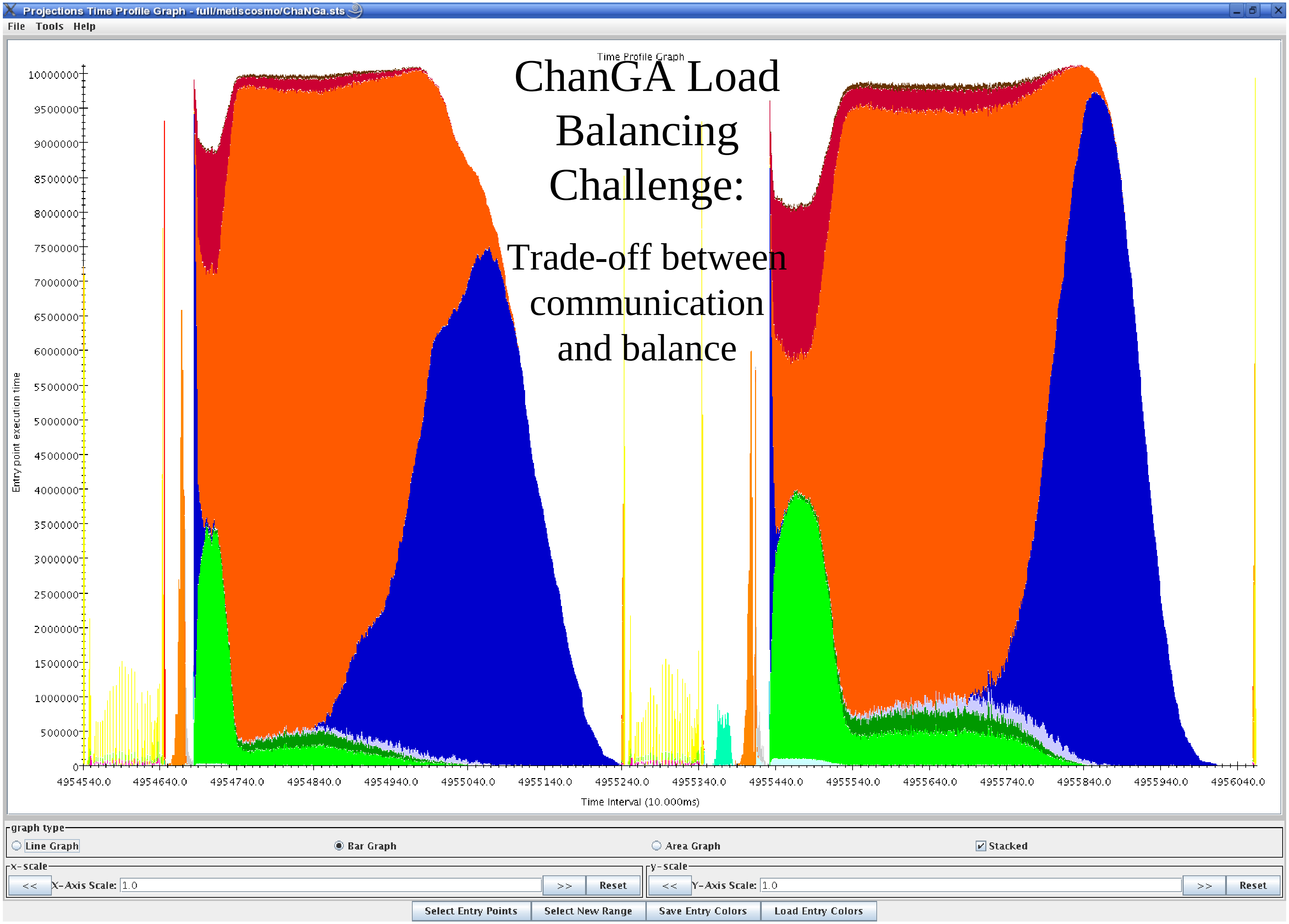
- N body Simulation (NSF)
  - N particles (1 million to 1 billion), in a periodic box
  - Move under gravitation
  - Organized in a tree (oct, binary (k-d), ..)
- Output data Analysis: in parallel (NASA)
  - Particles are read in parallel
  - Interactive Analysis
- Issues:
  - Load balancing, fine-grained communication, tolerating communication latencies.
  - Multiple-time stepping
- New Code Released: ChaNGa

Collaboration with T. Quinn (Univ. of Washington)

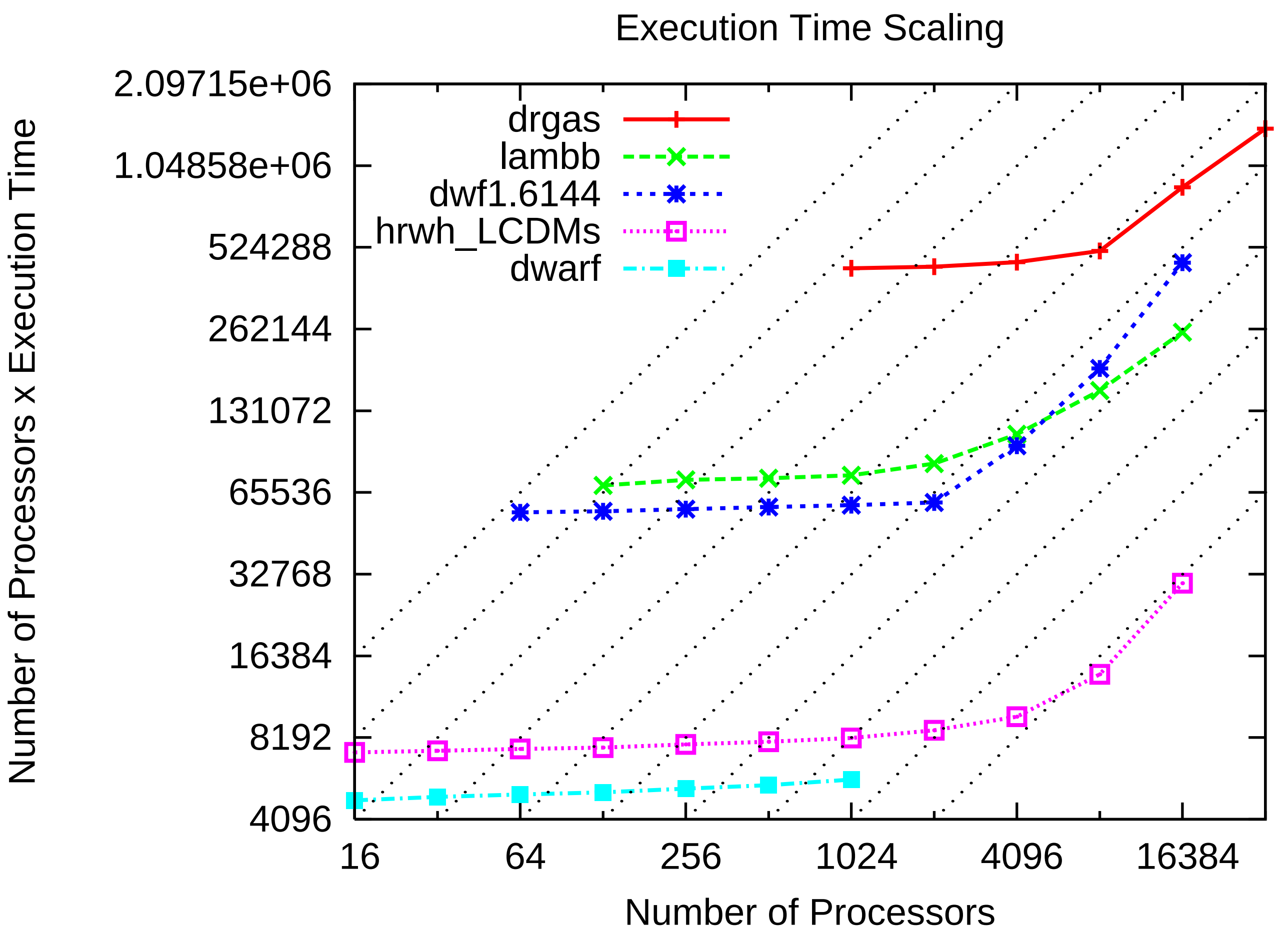
UofI Team: Filippo Giaochin, Pritish Jetley, Celso Mendes

ChanGA Load Balancing Challenge:

Trade-off between communication and balance

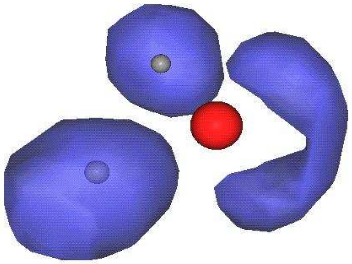


Recent Sucesses in Scaling ChaNGa



Quantum Chemistry: LeanCP

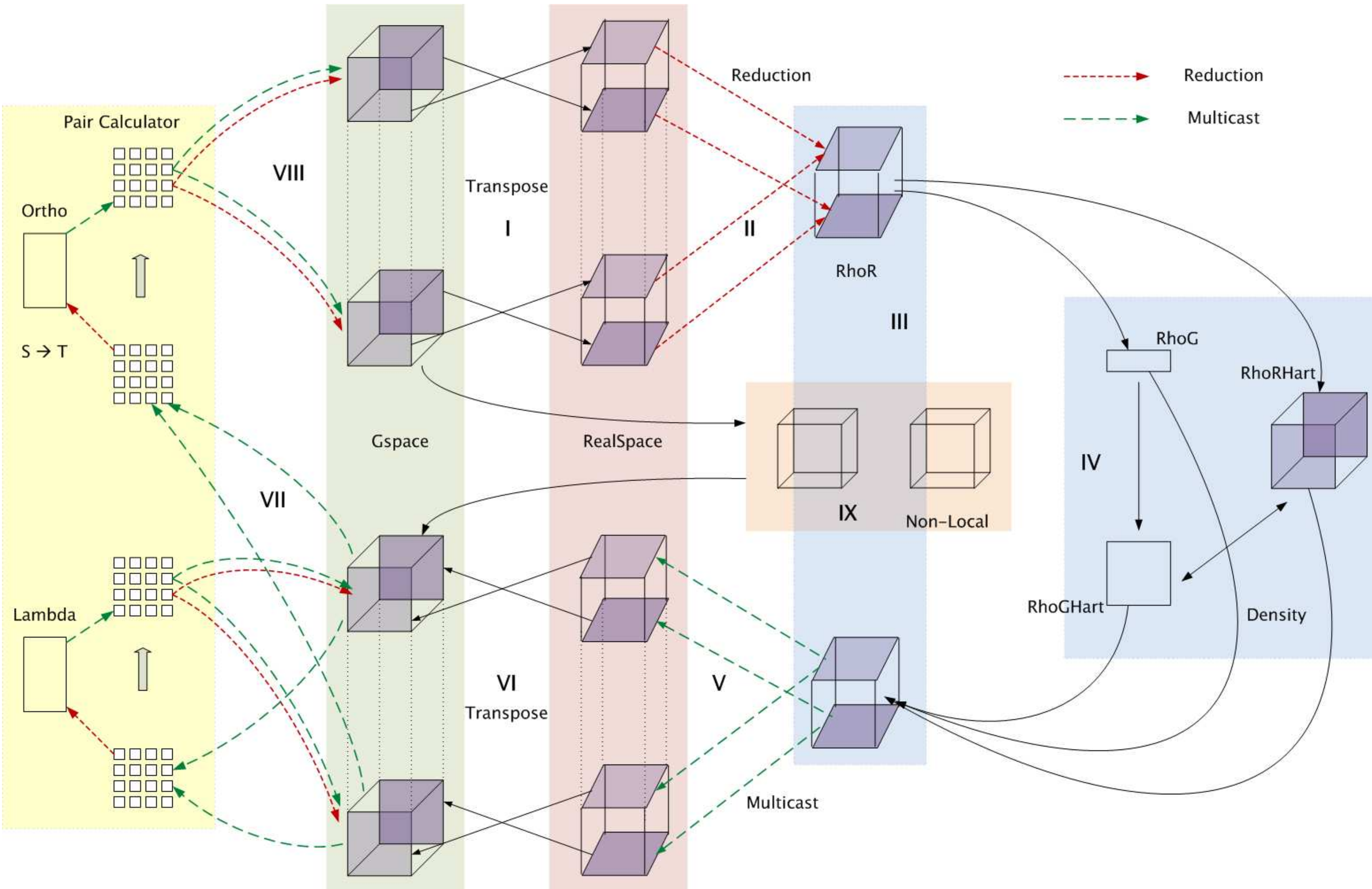
- Car-Parinello MD
- Illustrates utility of separating decomposition and mapping
- Very complex set of objects and interactions
- Excellent scaling achieved



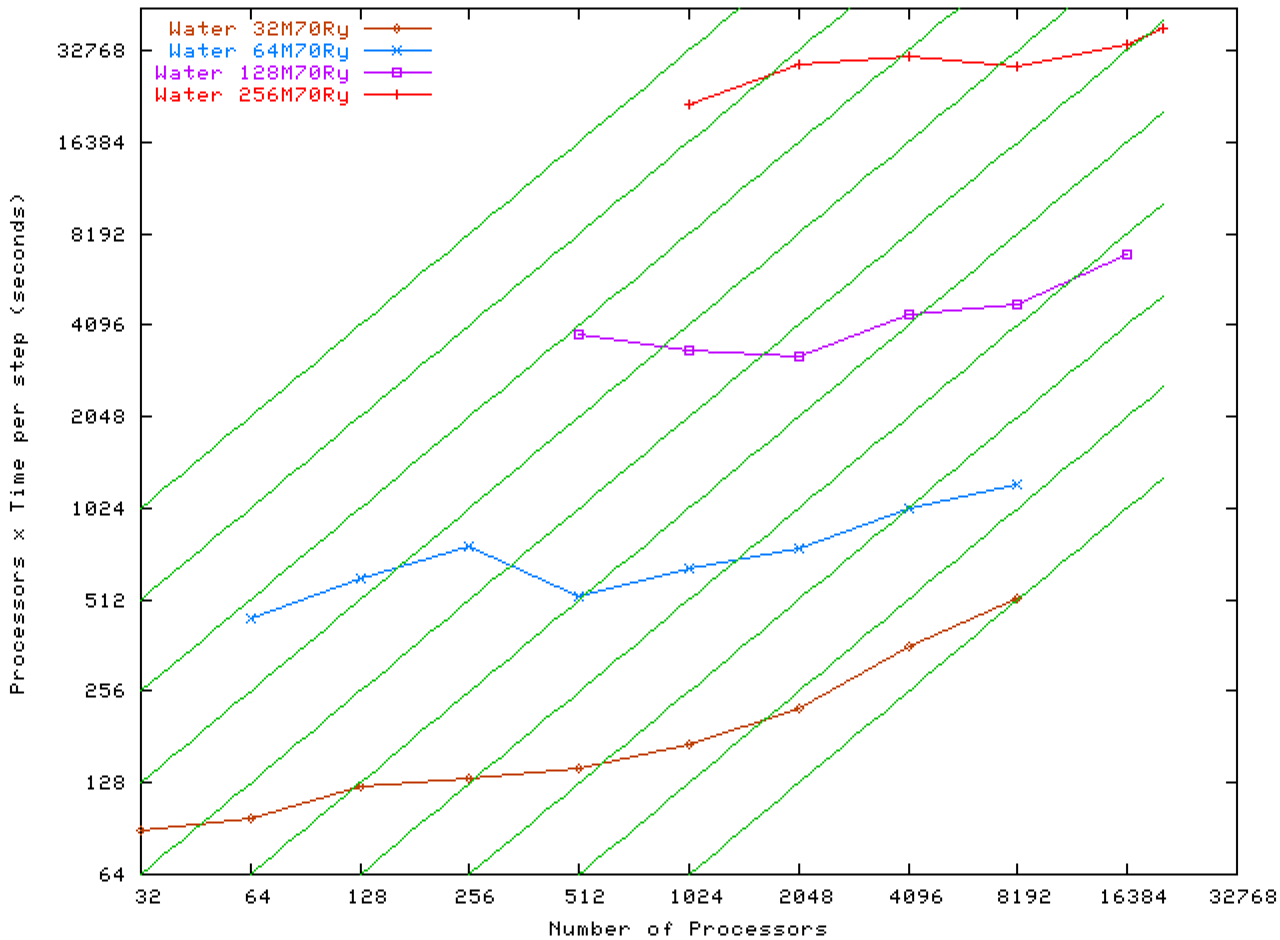
Collaboration with Glenn Martyna (IBM), Mark Tuckerman (NYU)

UofI team: Eric Bohm, Abhinav Bhatele

LeanCP Decomposition

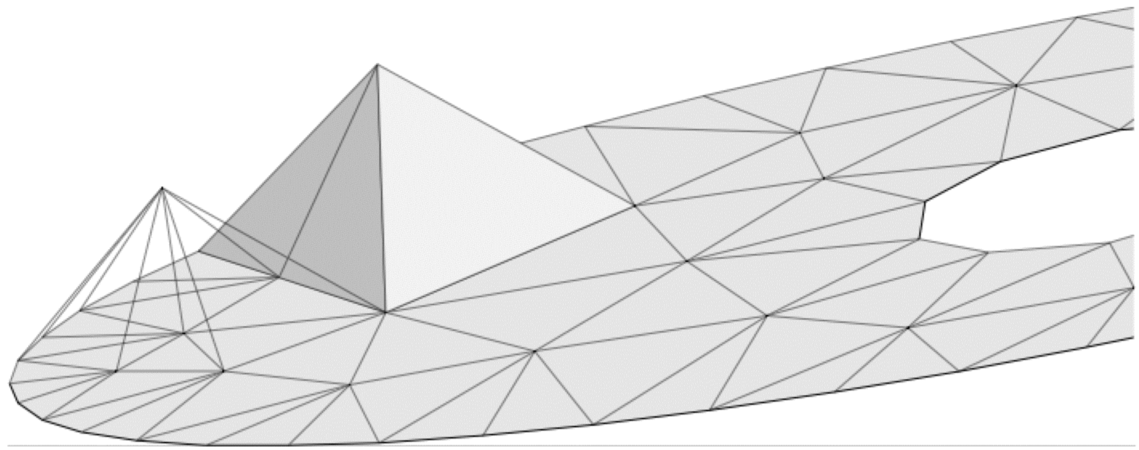


LeanCP Scaling



Space-time meshing

- Discontinuous Galerkin method
- Tent-pitcher algorithm

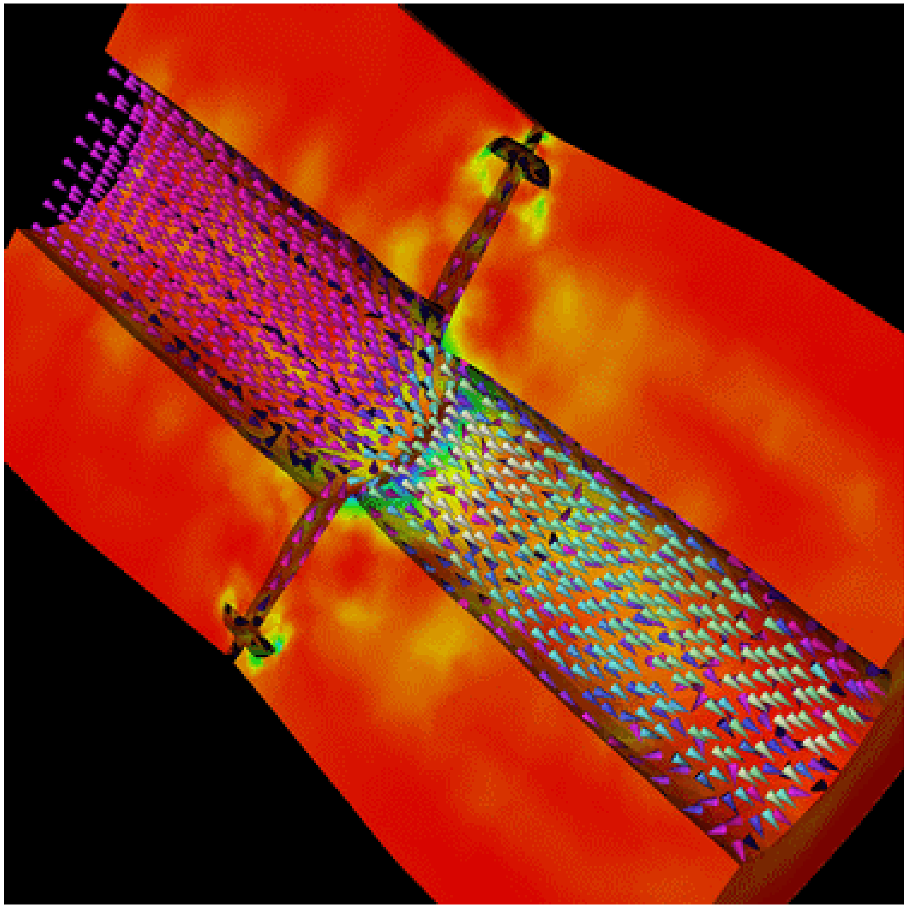


Collaboration with Bob Haber, Jeff Ericsson, Michael Garland

PPL team: Aaron Baker, Sayantan Chakravorty, Terry Wilmarth

Rocket Simulation

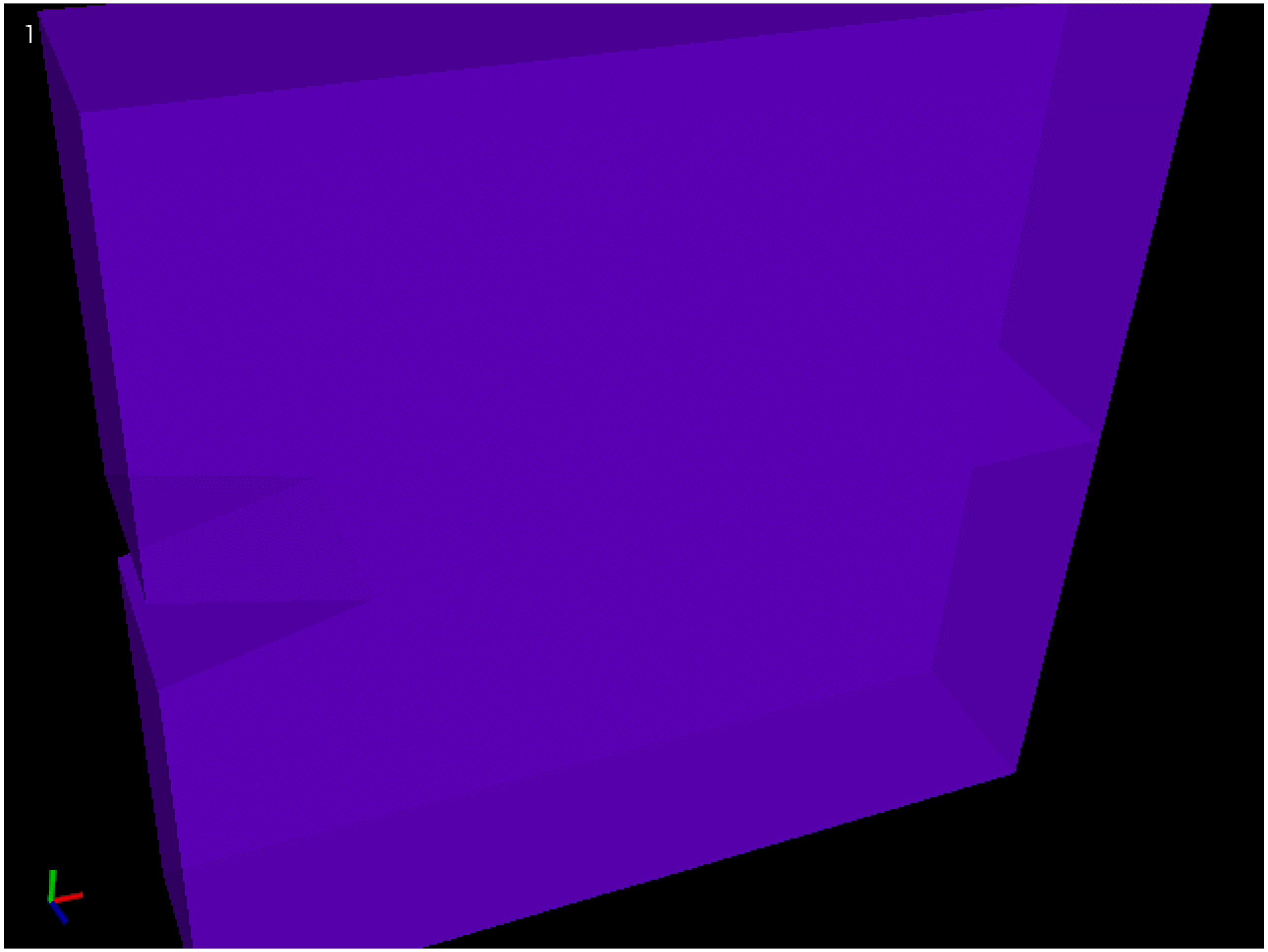
- Dynamic, coupled physics simulation in 3D
- Finite-element solids on unstructured tet mesh
- Finite-volume fluids on structured hex mesh
- Coupling every timestep via a least-squares data transfer
- Challenges:
  - Multiple modules
  - Dynamic behavior: burning surface, mesh adaptation



Robert Fielder, Center for Simulation of Advanced Rockets

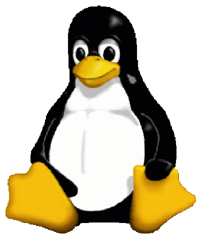
Collaboration with M. Heath, P. Geubelle, others

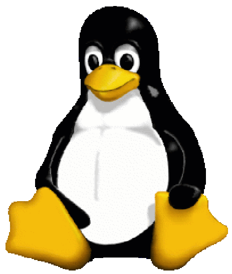
Dynamic load balancing in Crack Propagation

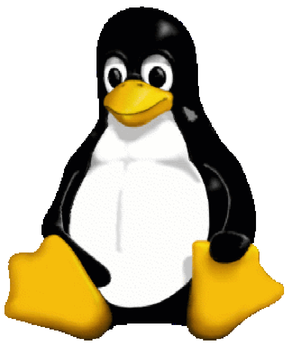


Colony: FAST-OS Project

- DOE funded collaboration
- Terry Jones: LLNL
- Jose Moreira, et al IBM
- At Illinois: supports
  - Scalable Dynamic Load Balancing
  - Fault tolerance







Colony Project Overview

Collaborators

Lawrence Livermore National Laboratory
Terry Jones

University of Illinois at Urbana-Champaign
Laxmikant Kale
Celso Mendes
Sayantan Chakravorty

International Business Machines
Jose Moreira
Andrew Tauferner
Todd Inglett

Title

Services and Interfaces to Support Systems with Very Large Numbers of Processors

Topics

- Parallel Resource Instrumentation Framework
- Scalable Load Balancing
- OS mechanisms for Migration
- Processor Virtualization for Fault Tolerance
- Single system management space
- Parallel Awareness and Coordinated Scheduling of Services
- Linux OS for cellular architecture

Load Balancing on Very Large Machines

- Existing load balancing strategies don't scale on extremely large machines
  - Consider an application with 1M objects on 64K processors

- Centralized
  - Object load data are sent to processor 0
  - Integrate to a complete object graph
  - Migration decision is broadcast from processor 0
  - Global barrier

- Distributed
  - Load balancing among neighboring processors
  - Build partial object graph
  - Migration decision is sent to its neighbors
  - No global barrier

A Hybrid Load Balancing Strategy

- Dividing processors into independent sets of groups, and groups are organized in hierarchies (decentralized)
- Each group has a leader (the central node) which performs centralized load balancing
- A particular hybrid strategy that works well

Gengbin Zheng, PhD Thesis, 2005

Fault Tolerance

- Automatic Checkpointing
  - Migrate objects to disk
  - In-memory checkpointing as an option
  - Automatic fault detection and restart
- Proactive Fault Tolerance
  - "Impending Fault" Response
  - Migrate objects to other processors
  - Adjust processor-level parallel data structures
- Scalable fault tolerance
  - When a processor out of 100,000 fails, all 99,999 shouldn't have to run back to their checkpoints!
  - Sender-side message logging
  - Latency tolerance helps mitigate costs
  - Restart can be speeded up by spreading out objects from failed processor

BigSim

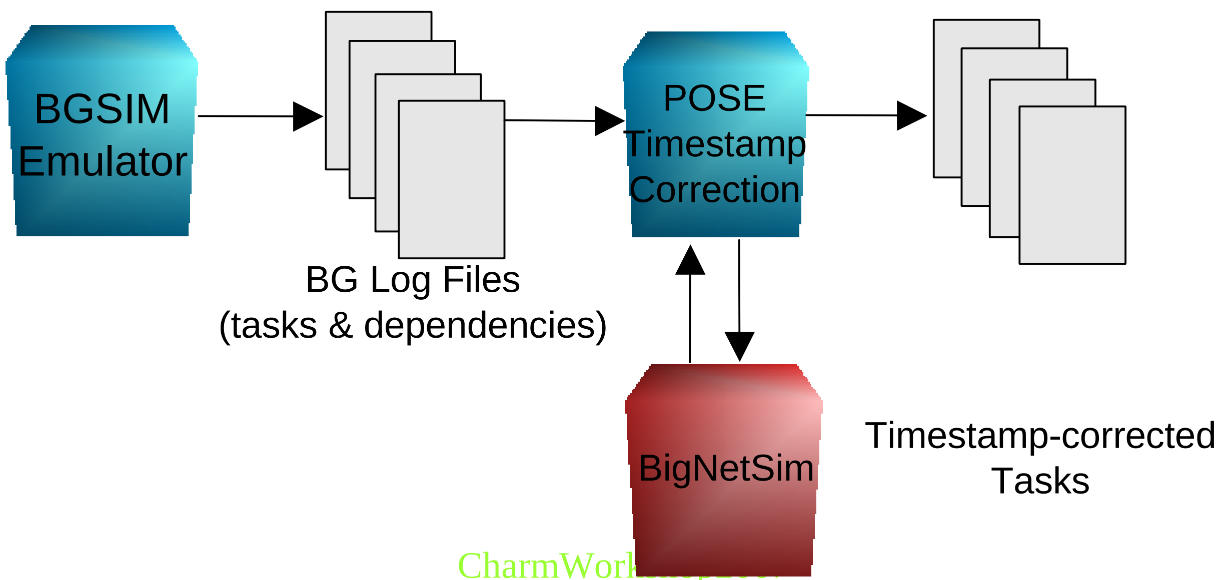
- Simulating very large parallel machines
  - Using smaller parallel machines
- Reasons
  - Predict performance on future machines
  - Predict performance obstacles for future machines
  - Do performance tuning on existing machines that are difficult to get allocations on
- Idea:
  - Emulation run using virtual processor processors (AMPI)
    - Get traces
  - Detailed machine simulation using traces

Objectives and Simualtion Model

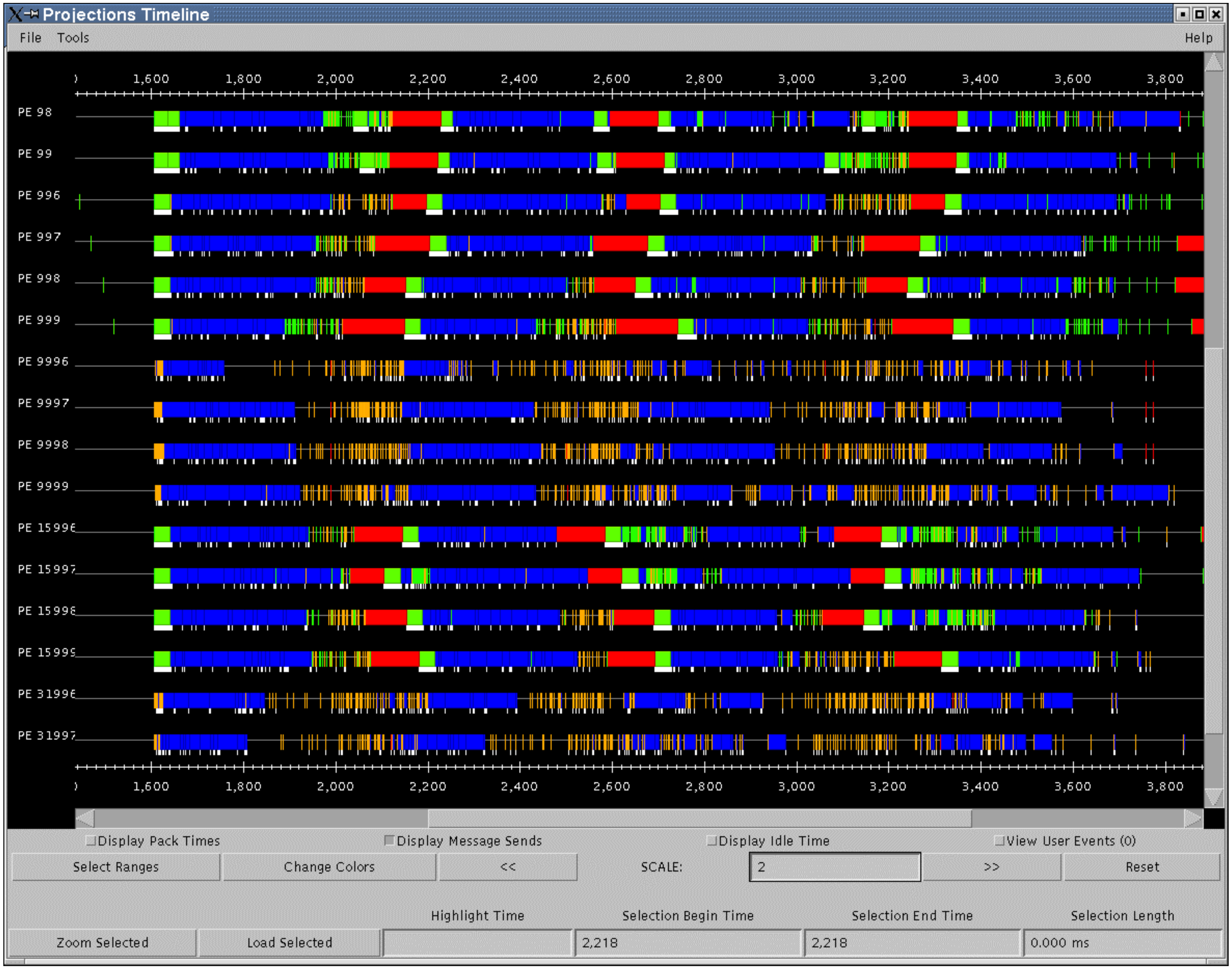
- Objectives:
  - Develop techniques to facilitate the development of efficient peta-scale applications
  - Based on performance prediction of applications on large simulated parallel machines
- Simulation-based Performance Prediction:
  - Focus on Charm++ and AMPI programming models Performance prediction based on PDES
  - Supports varying levels of fidelity
    - processor prediction, network prediction.
  - Modes of execution :
    - online and post-mortem mode

Big Network Simulation

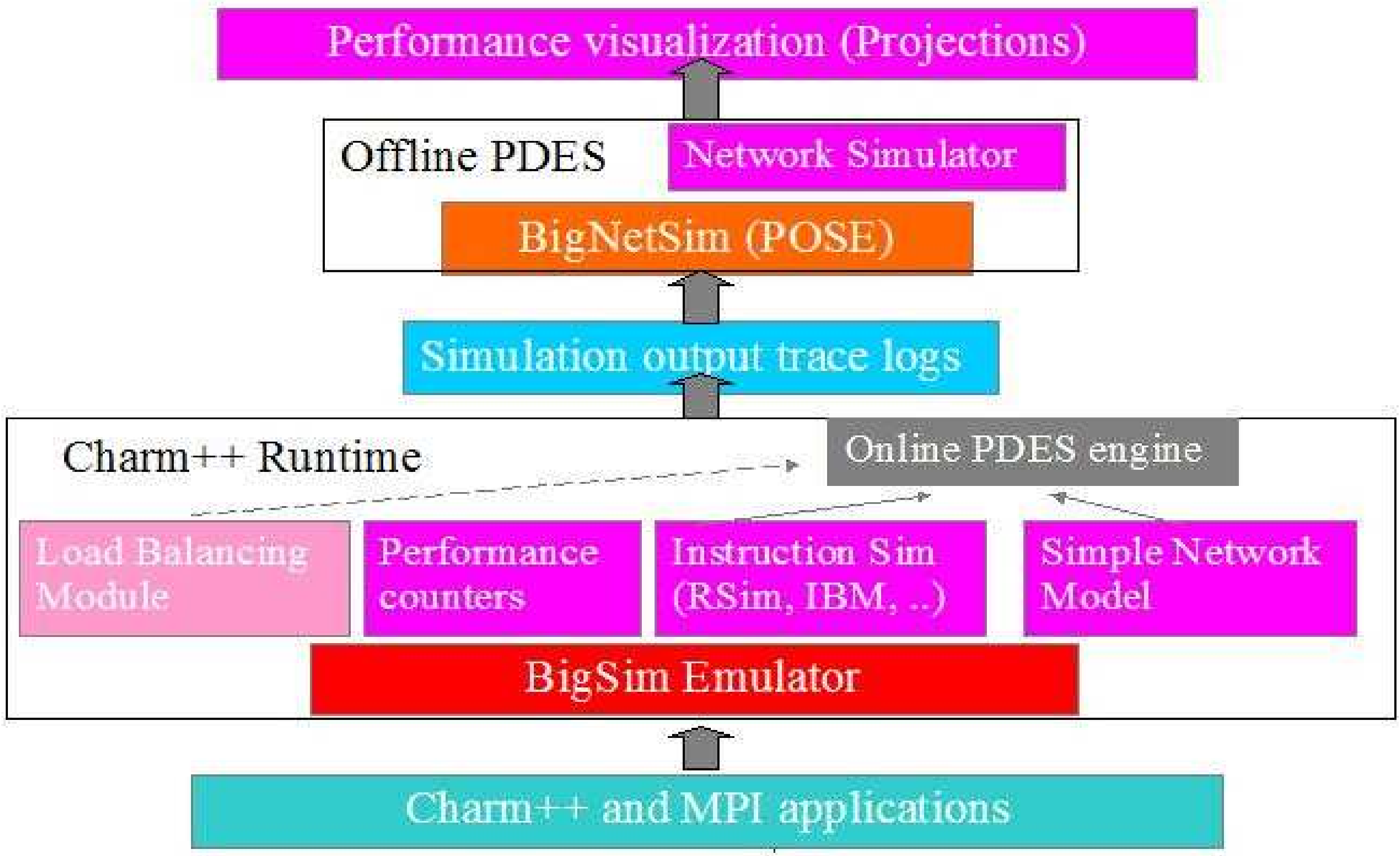
- Simulate network behavior: packetization, routing, contention, etc.
- Incorporate with post-mortem simulation
- Switches are connected in torus network



Projections: Performance visualization



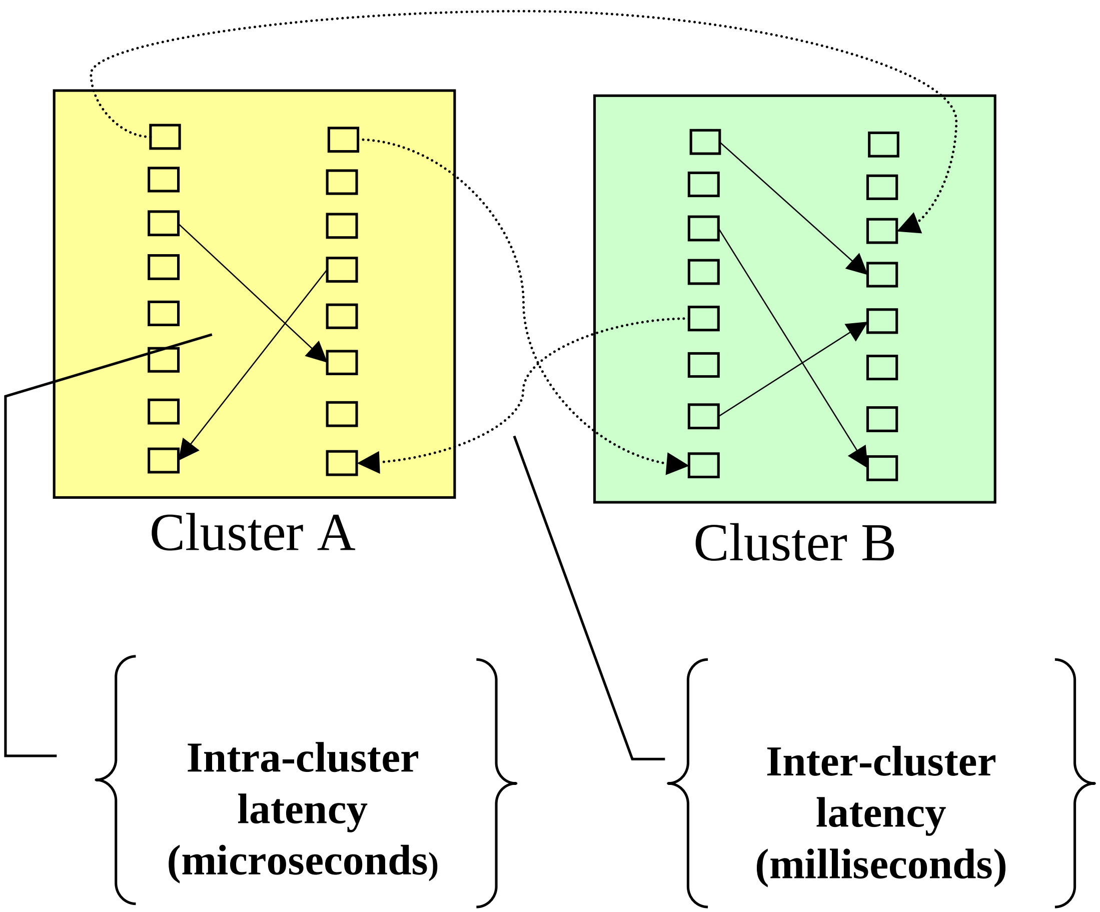
Architecture of BigNetSim



Performance Prediction (contd.)

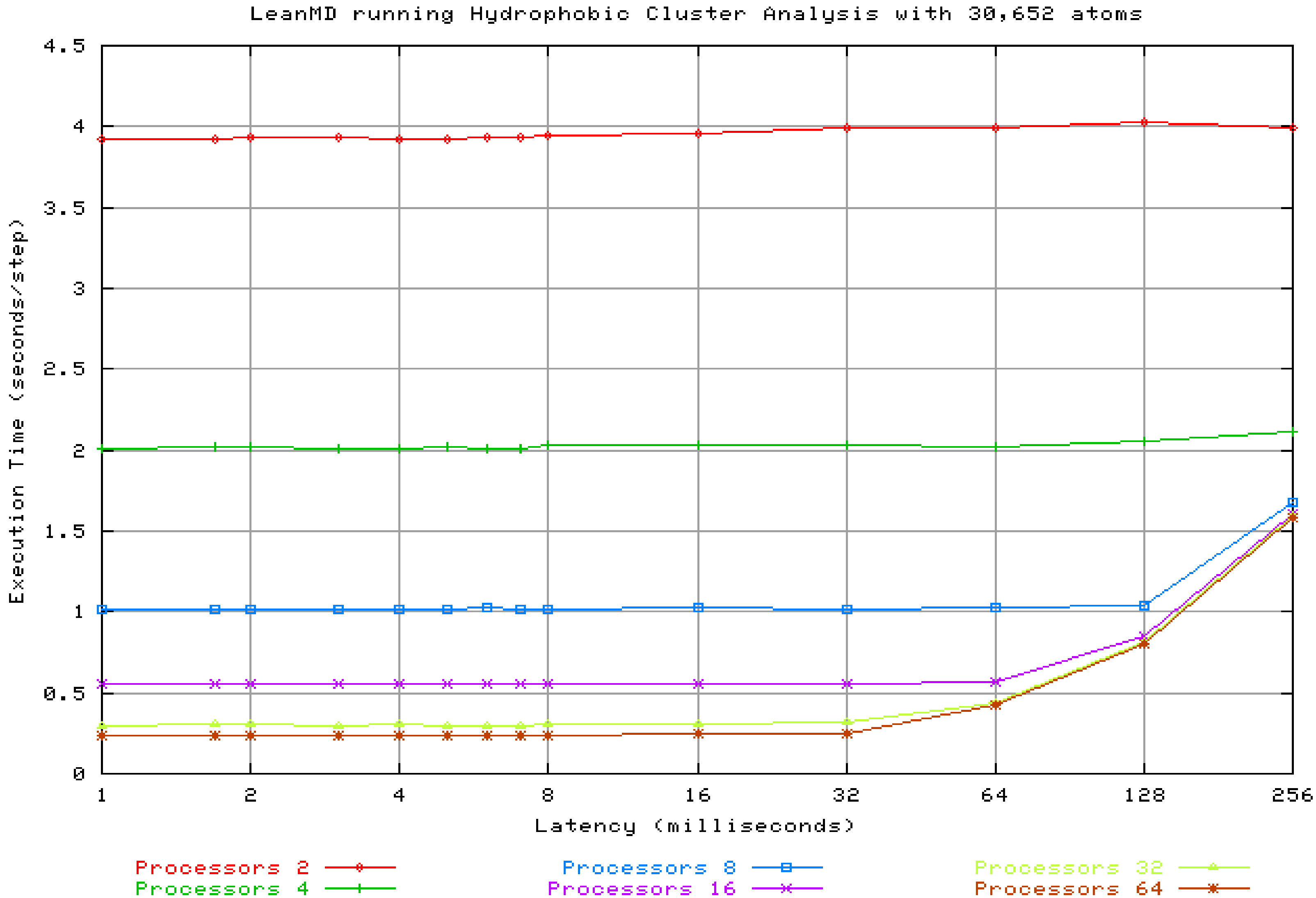
- Predicting time of sequential code:
  - User supplied time for every code block
  - Wall-clock measurements on simulating machine can be used via a suitable multiplier
  - Hardware performance counters to count floating point, integer, branch instructions, etc
    - Cache performance and memory footprint are approximated by percentage of memory accesses and cache hit/miss ratio
  - Instruction level simulation (not implemented)
- Predicting Network performance:
  - No contention, time based on topology & other network parameters
  - Back-patching, modifies comm time using amount of comm activity
  - Network-simulation, modelling the netowrk entirely

Multi-Cluster Co-Scheduling

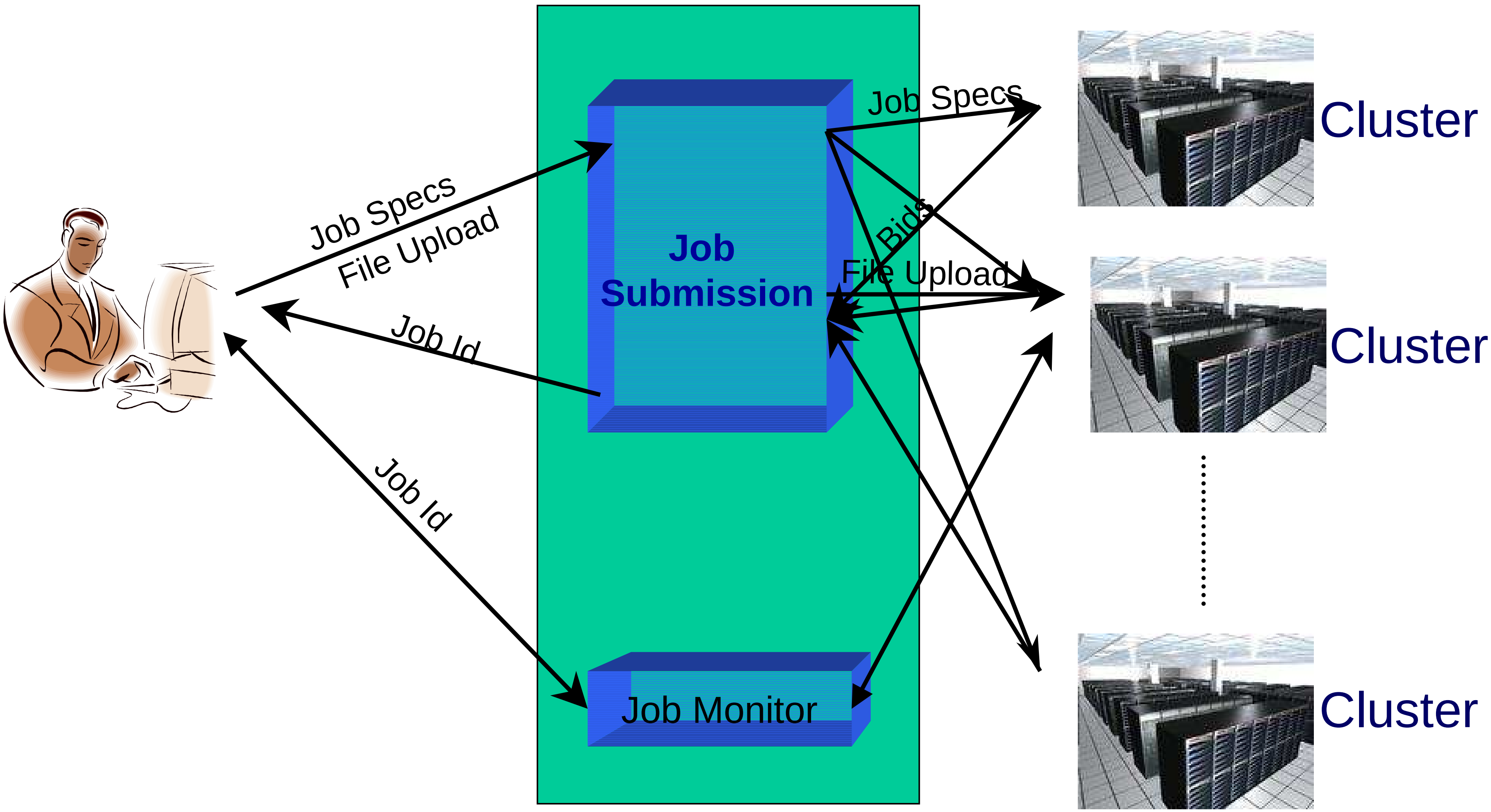


- Job co-scheduled to run across two clusters to provide access to large numbers of processors
- But cross cluster latencies are large!
- Virtualization within Charm++ masks high inter-cluster latency by allowing overlap of communication with computation

Multi-Cluster Co-Scheduling



Faucets: Optimizing Utilization Within/across Clusters



http://charm.cs.uiuc.edu/research/faucets

Other Ongoing Projects

- Parallel Debugger
- Automatic out-of-core execution
- Parallel algorithms
  - Current: Prim's spanning tree algorithm, sorting, ..
- New collaborations being explored
  - Prof. Paulino, Prof. Pantano, ..
-

Domain Specific Frameworks

Motivation

- Reduce tedium of parallel programming for commonly used paradigms and parallel data structures
- Encapsulate parallel data structures and algorithms
- Provide easy to use interface
- Used to build concurrenltly composible parallel modules

Frameworks

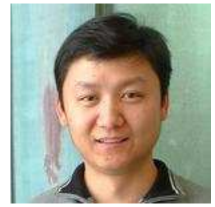
- Unstructured Meshes:ParFUM
  - Generalized ghost regions
  - Used in *Rocfrac, Rocflu at rocket center, and* Outside CSAR
  - Fast collision detection
- Multiblock framework
  - Structured Meshes
  - Automates communication
- AMR
  - Common for both above
- Particles
  - Multiphase flows
  - MD, tree codes

Summary and Messages

- We at PPL have advanced migratable objects technology
  - We are committed to supporting applications
  - We grow our base of reusable techniques via such collaborations
- Try using our technology:
  - AMPI, Charm++, Faucets, ParFUM, ..
  - Available via the web http:// charm.cs.uiuc.edu

Parallel Programming Laboratory

Sr.STAFF



IBM PERCS High Productivity

GRANTS

- NSF: ITR Chemistry Car-Parinello MD, QM/MM
- NCSA Faculty Fellows Program
- NSF: ITR , NASA Computational Cosmology and Visualization
- DOE HPC-Colony Services and Interfaces for Large Computers
- NIH Biophysics NAMD
- DOE CSAR Rocket Simulation
- NSF: ITR CPSD Space / Time Meshing
- NSF Next Generation Software BlueGene

ENABLING PROJECTS

- **Faucets**: Dynamic Resource Management for Grids
- **Load-Balance**: Centralized, Distributed, Hybrid
- **Fault-Tolerance**: Checkpointing, Fault-Recovery, Proc.Evacuation
- **ParFUM**: Supporting Unstructured Meshes (Comp.Geometry)
- **BigSim**: Simulating Big Machines and Networks

- Charm++ and Converse
- AMPI Adaptive MPI
- Projections: Performance Analysis
- Orchestration and Parallel Languages

Over the next two days

Keynote: Kathy Yelick

PGAS Languages and Beyond

System progress talks

- Adaptive MPI
- BigSim: Performance prediction
- Scalable Performance Analysis
- Fault Tolerance
- Cell Processor
- Grid Multi-cluster applications

Applications

- Molecular Dynamics
- Quantum Chemistry (LeanCP)
- Computational Cosmology
- Rocket Simulation

Tutorials

- Charm++
- Projections
- AMPI
- BigSim