Scaling molecular dynamics to 25,000 GPU's on Sierra and Summit

Mar. 25, 2017
LLNL
7000 East Avenue
Livermore, CA 94550

Presenters: S. Sundram, T. Oppelstrup
Collaboration: F. H. Streitz, F. Lightstone, J. Glosli, L. Stanton, M. Surh, T. Carpenter, H. Ingolfsson, Y. Yang, X. Zhang, S. Kokkila-Schumacher, A. Voter, …

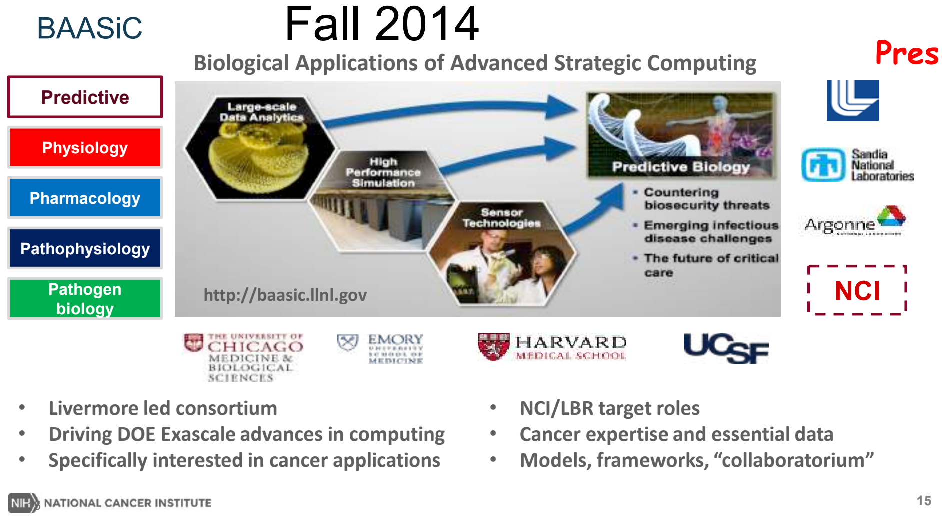
LLNL-PRES-747066-DRAFT
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Outline

- Background for new molecular dynamics development
- JDACS4C NCI/DOE collaboration
  - Advanced simulation techniques and application of HPC to cancer research
  - Three pilot programs to investigate potential impact on cancer resarch
- Pilot 2: Simulation of RAS protein on cell membranes
  - Multi-scale simulation effort
  - Ecosystem of connected applications
- Layout on heterogeneous hardware
- Key component: fast scalable molecular dynamics
  - Short range forces and MARTINI force field
  - Long range electrostatic forces and CHARMM

Environment Leading to the DOE-NCI Collaboration on Cancer Research


Large-scale Data Analytics
High Performance Simulation
Sensor Technologies
Predictive Biology
UNIVERSITY OF CHICAGO MEDICINE & BIOLOGICAL SCIENCES
EMORY UNIVERSITY SCHOOL OF MEDICINE
HARVARD MEDICAL SCHOOL
UCSF
Sandia National Laboratories
Argonne

BAASiC

Fall 2014

Biological Applications of Advanced Strategic Computing

- Predictive
- Physiology
- Pharmacology
- Pathophysiology
- Pathogen biology

- Countering biosecurity threats
- Emerging infectious disease challenges
- The future of critical care

http://baasic.llnl.gov

NCI

- Livermore led consortium
- Driving DOE Exascale advances in computing
- Specifically interested in cancer applications

- NCI/LBR target roles
- Cancer expertise and essential data
- Models, frameworks, "collaboratorium"

NIH NATIONAL CANCER INSTITUTE

15

President Obama Announces the Precision Medicine Initiative

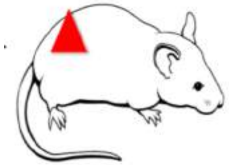
Photo by F. Collins

The East Room, January 30, 2015

Integrated Precision Oncology

Crosscut: Integrated Precision and Predictive Oncology

Pilot 1
Pre-clinical Model Development

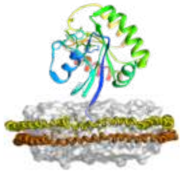


Aim 1: Predictive Models of Drug Response (signatures)

Aim 2: Uncertainty Quantification and Improved Experimental Design

Aim 3: Develop Hybrid Predictive Models

Pilot 2
RAS Therapeutic Targets



Aim 1: Adaptive time and length scaling in dynamic multi-scale simulations

Aim 2: Validated model for Extended RAS/RAS-complex interactions

Aim 3: Development of machine learning for dynamic model validation

Pilot 3
Precision Oncology Surveillance

Aim 1: Information Capture Using NLP and Deep Learning Algorithms

Aim 2: Information Integration and Analysis for extreme scale heterogeneous data

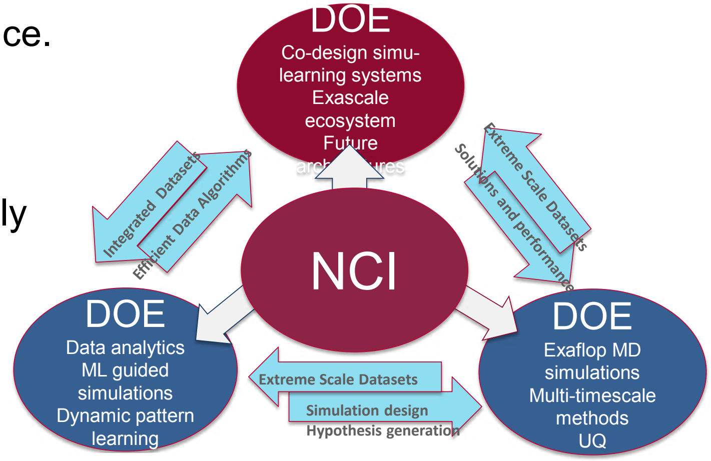
Aim 3: Modeling for patient health trajectories

Crosscut: Uncertainty Quantification (UQ) and CANDLE exascale technologies

Extending the Frontiers for DOE and NCI

DOE Exascale Computing – Extending the Frontiers

- Broaden CORAL functionality through co-design of highly scalable machine learning tools able to exploit node coherence.
- Explore how deep learning can define dynamic multi-scale validation, uncertainty quantification and optimally guide experiments and accelerate time-to solution.
- Shape the design of architectures for exascale simultaneously optimized for big data, machine learning and large-scale simulation.

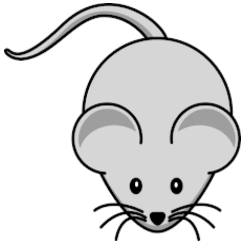


NCI Precision Oncology – Extending the Frontiers

- Identify promising new treatment options through the use of advanced computation to rapidly develop, test and validate predictive pre-clinical models for precision oncology.
- Deepen understanding of cancer biology and identify new drugs through the integrated development and use of new simulations, predictive models and next-generation experimental data.
- Transform cancer care by applying advanced computational capabilities to population-based cancer data to understand the impact of new diagnostics, treatments and patient factors in real world patients.

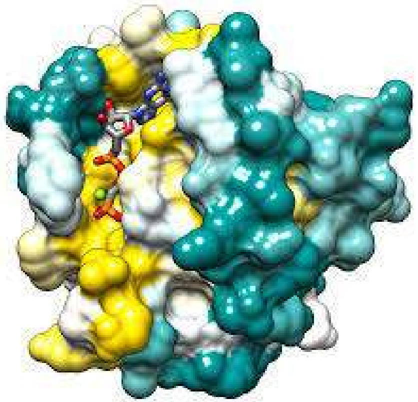
Pilot 1 : Pre-clinical Models
DOE: Machine Learning

- Pre-clinical Model Development and Therapeutic Evaluation
- Scientific lead: Dr. James Doroshow
- Key points:
  - Rapid evaluation of large arrays of small compounds for impact on cancer
  - Deep understanding of cancer biology
  - Development of *in silico* models of biology and predictive models capable of evaluating therapeutic potential of billions of compounds



NIH NATIONAL CANCER INSTITUTE

Pilot 2: RAS Related Cancers
DOE: Multiscale Simulations

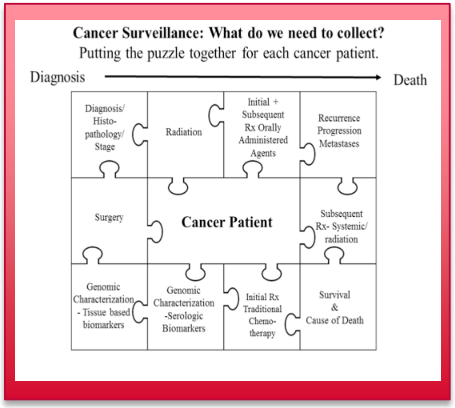


- Improving Outcomes for RAS Related Cancers
- Scientific lead: Dr. Frank McCormick
- Key points:
  - Mutated RAS is found in nearly one-third of cancers, yet remains untargeted with known drugs
  - Advanced multi-modality data integration is required for model development
  - Simulation and predictive models for RAS related molecular species and key interactions
  - Provide insight into potential drugs and assays

NIH NATIONAL CANCER INSTITUTE

Pilot 3: Evidence-base Precision Medicine
DOE: Machine Learning

Pilot Project 3: Evidence-based Precision Medicine



Cancer Surveillance: What do we need to collect?

Putting the puzzle together for each cancer patient.

Diagnosis → Death

Diagnosis/ Histo-pathology/ Stage

Radiation

Initial + Subsequent Rx Orally Administered Agents

Recurrence Progression Metastases

Surgery

Cancer Patient

Subsequent Rx- Systemic/ radiation

Genomic Characterization - Tissue based biomarkers

Genomic Characterization -Serologic Biomarkers

Initial Rx Traditional Chemo-therapy

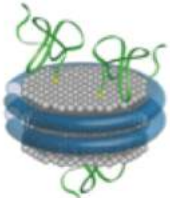
Survival & Cause of Death

- Information Integration for Evidence-based Cancer Precision Medicine
- Scientific lead: Dr. Lynne Penberthy
- Key points:
  - Integrates population and citizen science into improving understanding of cancer and patient response
  - Gather key population-wide data on treatment, response and outcomes
  - Leverages existing SEER and tumor registry resources
  - Novel avenues for patient consent, data sharing and participation

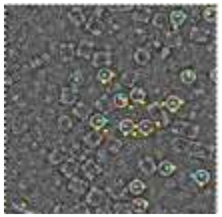
NIH NATIONAL CANCER INSTITUTE

Pilot 2: Overview

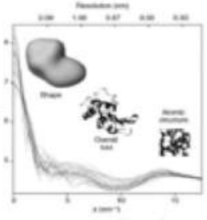
RAS activation experiments (FNLCR)



Experiments on nanodisc



CryoEM imaging



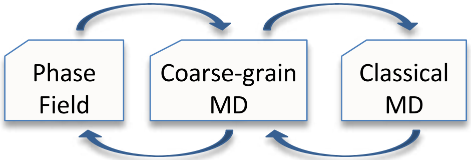
X-ray/neutron scattering

Multi-modal experimental data, image reconstruction, analytics

Protein structure databases

New adaptive-resolution multi-scale modeling capability

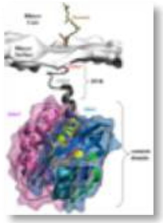
Adaptive time stepping



Adaptive spatial resolution

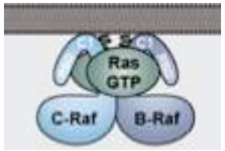
High-fidelity subgrid modeling

Predictive simulation and analysis of RAS-complex activation



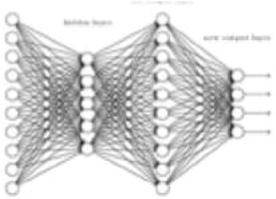
Granular RAS membrane interaction simulations

High res simulation of RAS-RAF interaction



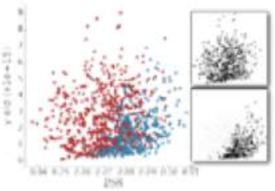
Inhibitor target discovery

Machine learning guided dynamic validation



Unsupervised deep feature learning

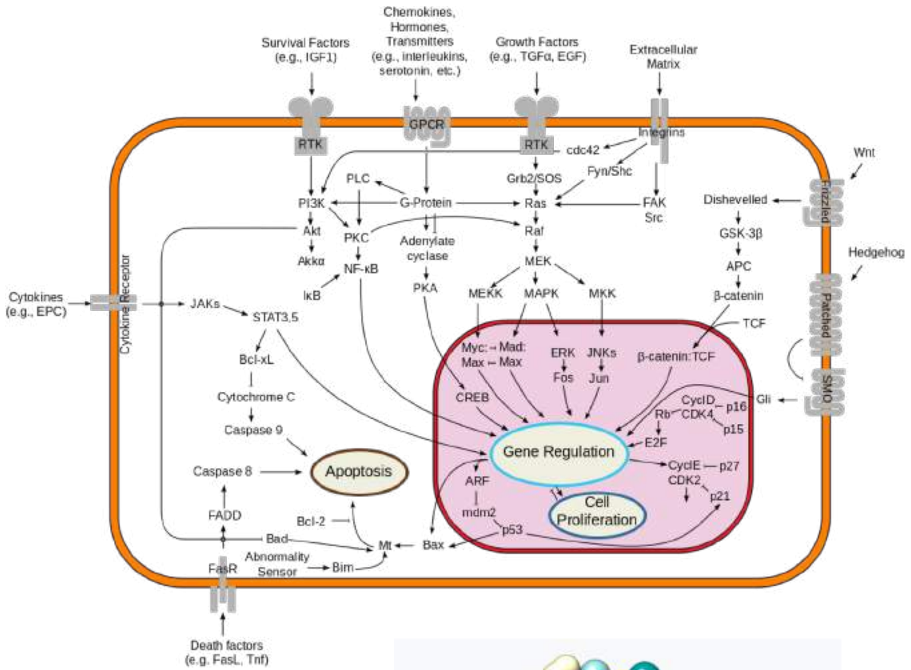
Mechanistic network models



Uncertainty quantification

RAS proteins and their relevance to cancer

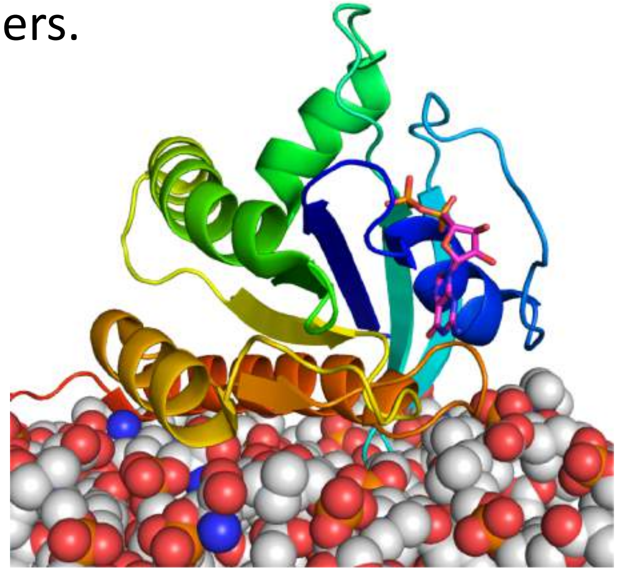
- Found in human cancers in the ‘80s
- Involved in cell signaling pathways for cell growth and division
- Mutation in RAS can leave it constantly activated, instead of temporarily



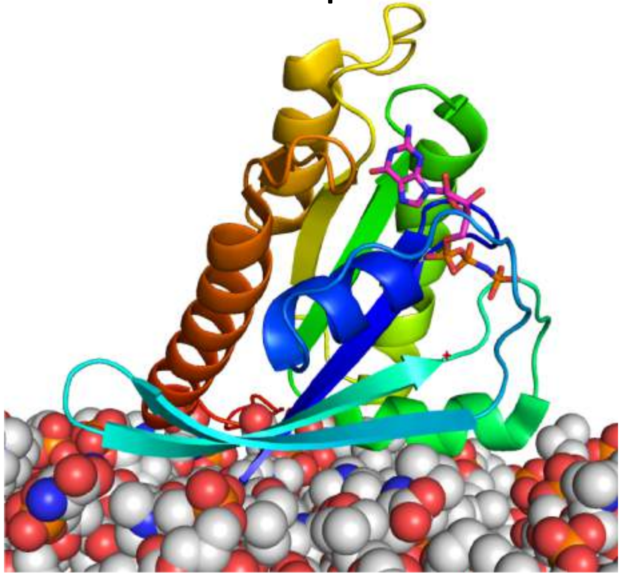
Rendering of RAS protein bound with GDPase molecule

RAS-Lipid Bilayer Simulations

- Most MD studies of RAS have been in solution with no membrane.
- RAS *only* has biological activity when embedded in a membrane.
- NMR experiments have shown that RAS dynamics in membranes are complicated and are affected by the membrane composition and binding partners.



Inactive K-Ras binding GDP

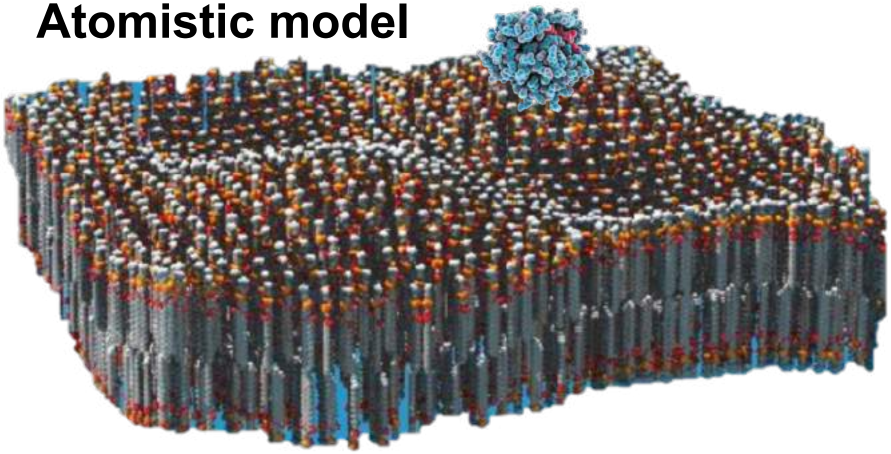


Active K-Ras binding GNP

Multiscale simulation of RAS on bilayer

Atomistic model

RAS diffusion ~1 nm^2/µs

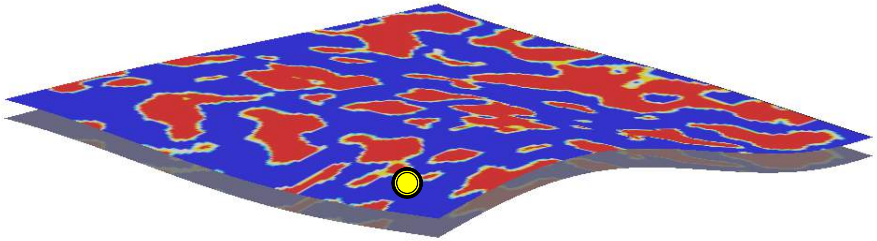


Lipid diffusion < 40 nm^2/µs

Timestep for atomistic simulation

- All-atom (CHARMM) ~2 fs
- Coarse-grained (MARTINI) ~30 fs

Continuum + particles model



Timescale to resolve different processes

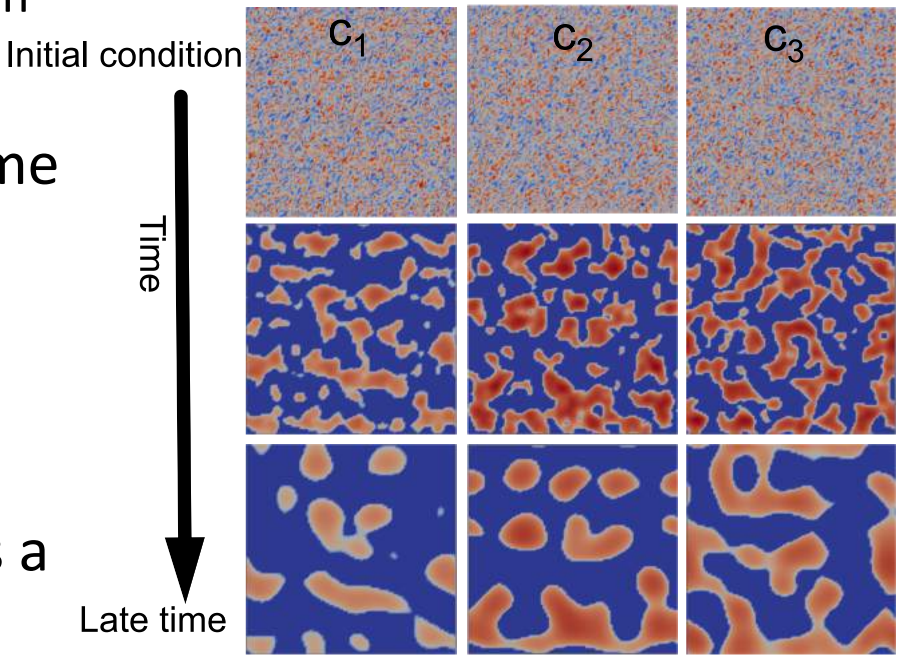
- RAS diffusion ~1 µs
- Fastest lipid diffusion ~25 ns
- Lipid flips ~1 µs
- Close RAS-RAS interaction ~40 ps

Adaptive cost and level of detail of continuum model

- Adapt resolution to match spatial length scales of interest
- Implicit integration: time-step matches timescale of studied feature, not fastest timescale in system
- 1.5-60 ms/day for 1µm x 1µm bilayer patch
- Relaxation / finding steady state through direct solve

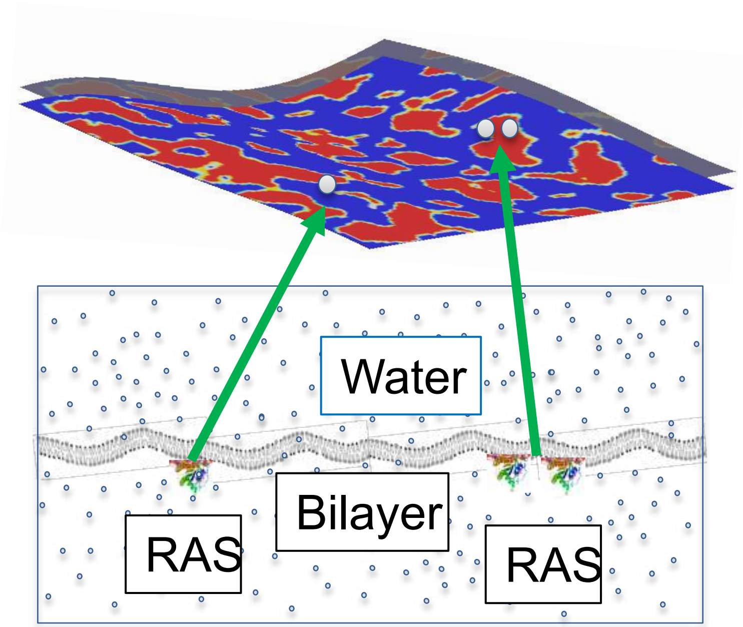
Phase field (continuum) simulation of lipid layer

- Continuum simulation is cheap!
  - Tunable resolution
  - 1 000 – 10 000 times faster than atomistic simulation
- Implicit solvers allow long time step, and quick approach to steady state
  - Can never be reached with molecular dynamics
- Simulation on the right takes a minute on a workstattion
  - Atomistic simulation Would take days on a cluster



Phase field simulation of lipid aggregation

Coupled Phase-Field + Hyper-Coarsened Protein (HyCoP)



Free Energy

$$\mathcal{F}(\mathbf{c}, h) = \int_\Omega d^3\mathbf{r}\,(f_b + f_i + f_c + f_{mp})$$

Bulk

$$f_b = \phi(\mathbf{c}_1, \mathbf{c}_2, h)$$

Interfacial

$$f_i = \frac{1}{2}\sum_{i=1}^{2}\sum_{j=1}^{n} \nabla_\mathbf{r} c_{ij} \cdot (\hat{\mathbf{I}}_{ij} \nabla_\mathbf{r} \mathbf{c}_{ij})$$

Curvature

$$f_c = \frac{1}{2}\kappa(\mathbf{c}_1, \mathbf{c}_2)\left(\nabla^2 h - \mathcal{C}(\mathbf{c}_1, \mathbf{c}_2)\right)^2$$

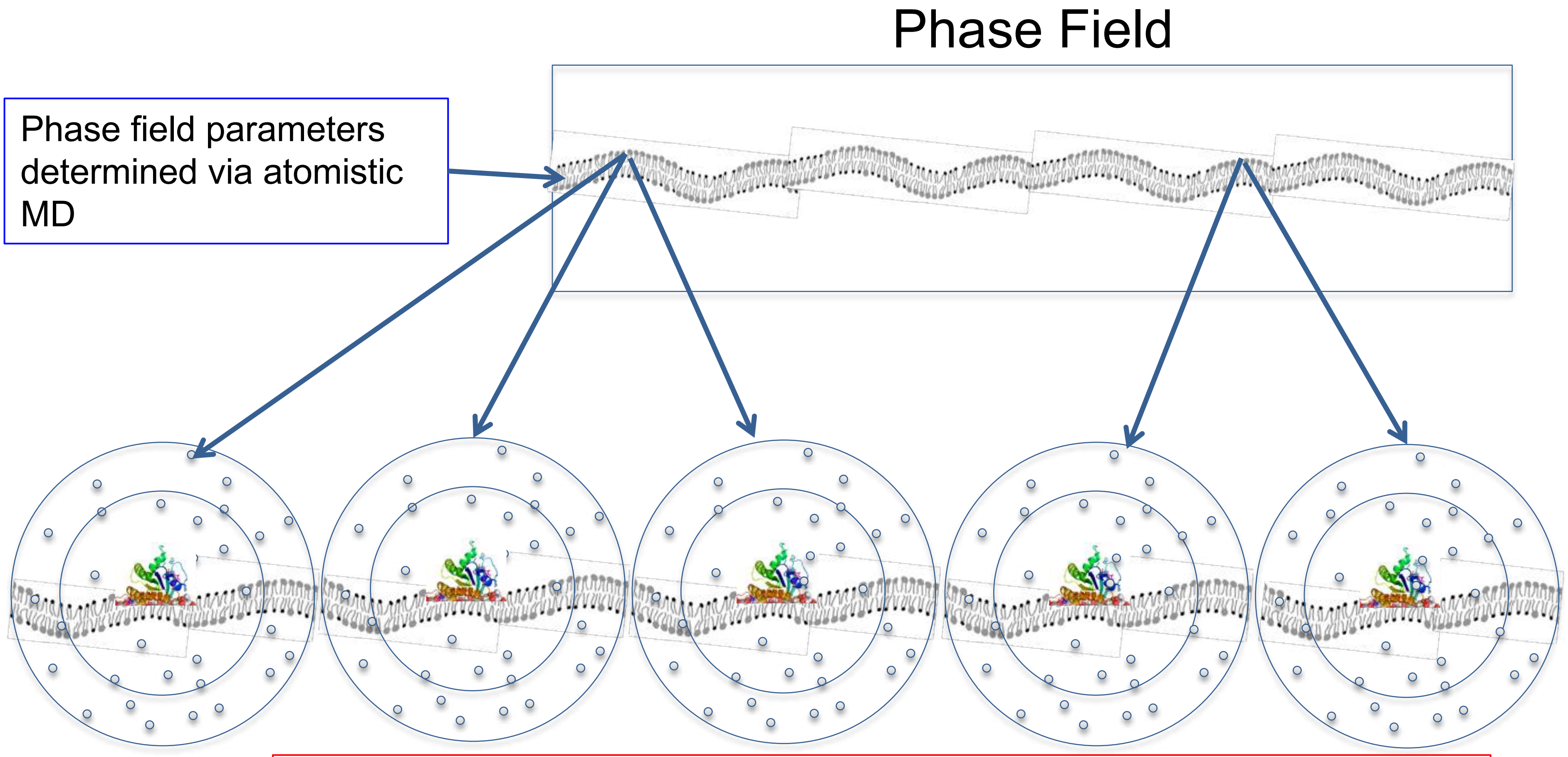
Membrane-Protein

$$f_{mp} = \sum_{j,n} c_{1j}(\mathbf{r}) V_j^{s_n}(\mathbf{r} - \mathbf{R}_n)$$

Evolution

$$\frac{\partial c_{ij}}{\partial t} = \sum_{k=1}^{N} \nabla \cdot \beta D_{jk}^{(i)} c_{ik} \nabla_\mathbf{r} \left(\frac{\delta \mathcal{F}}{\delta c_{ik}}\right)$$

Ensemble Multi-scale – select interesting domain for atomistic zoom-in


Phase Field

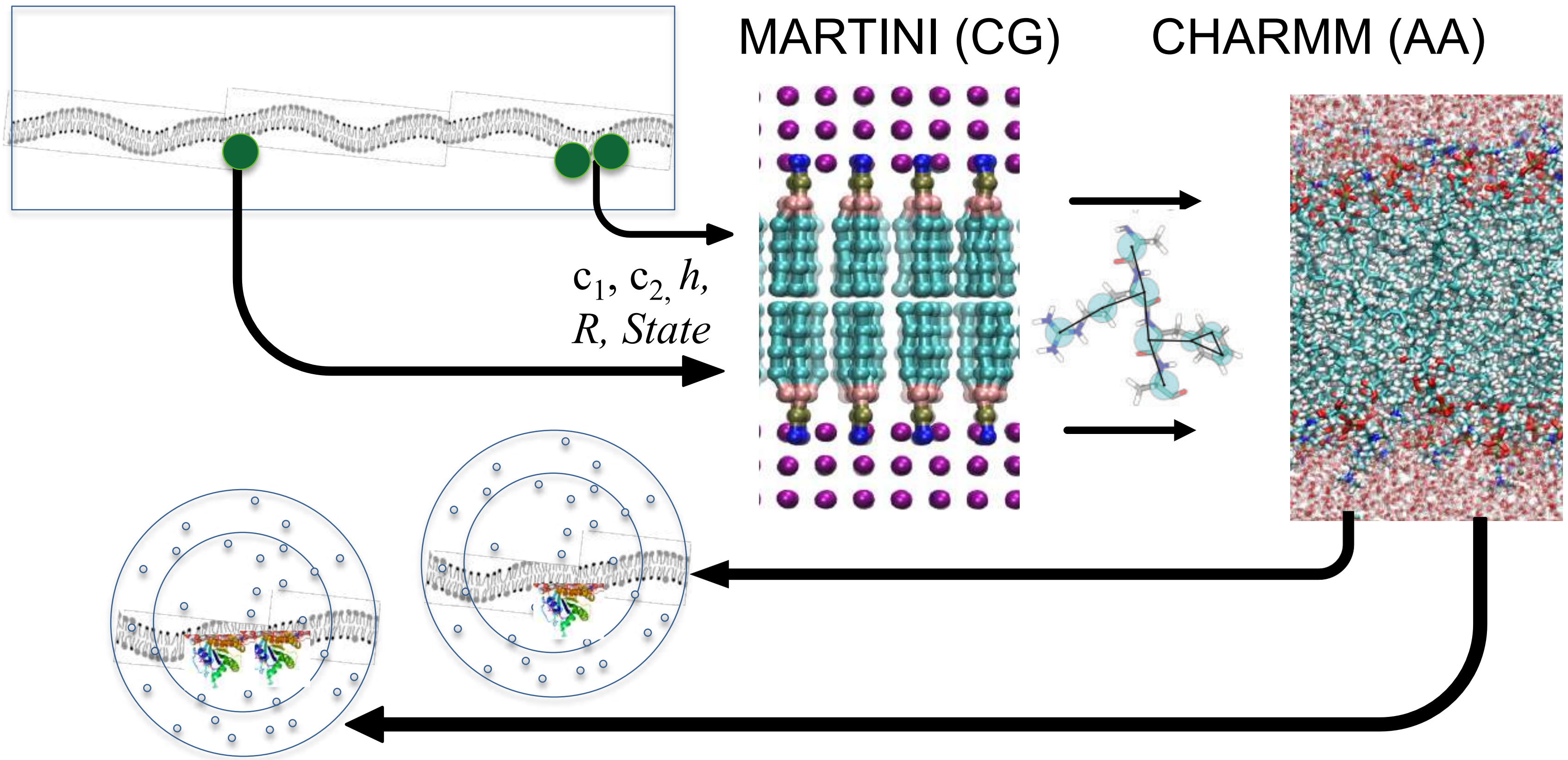
Phase field parameters determined via atomistic MD

Many 1 million atom MD simulations

Back Mapping Phase Field to MD

Phase field + HyCoP simulations

Back-mapping PF to MD



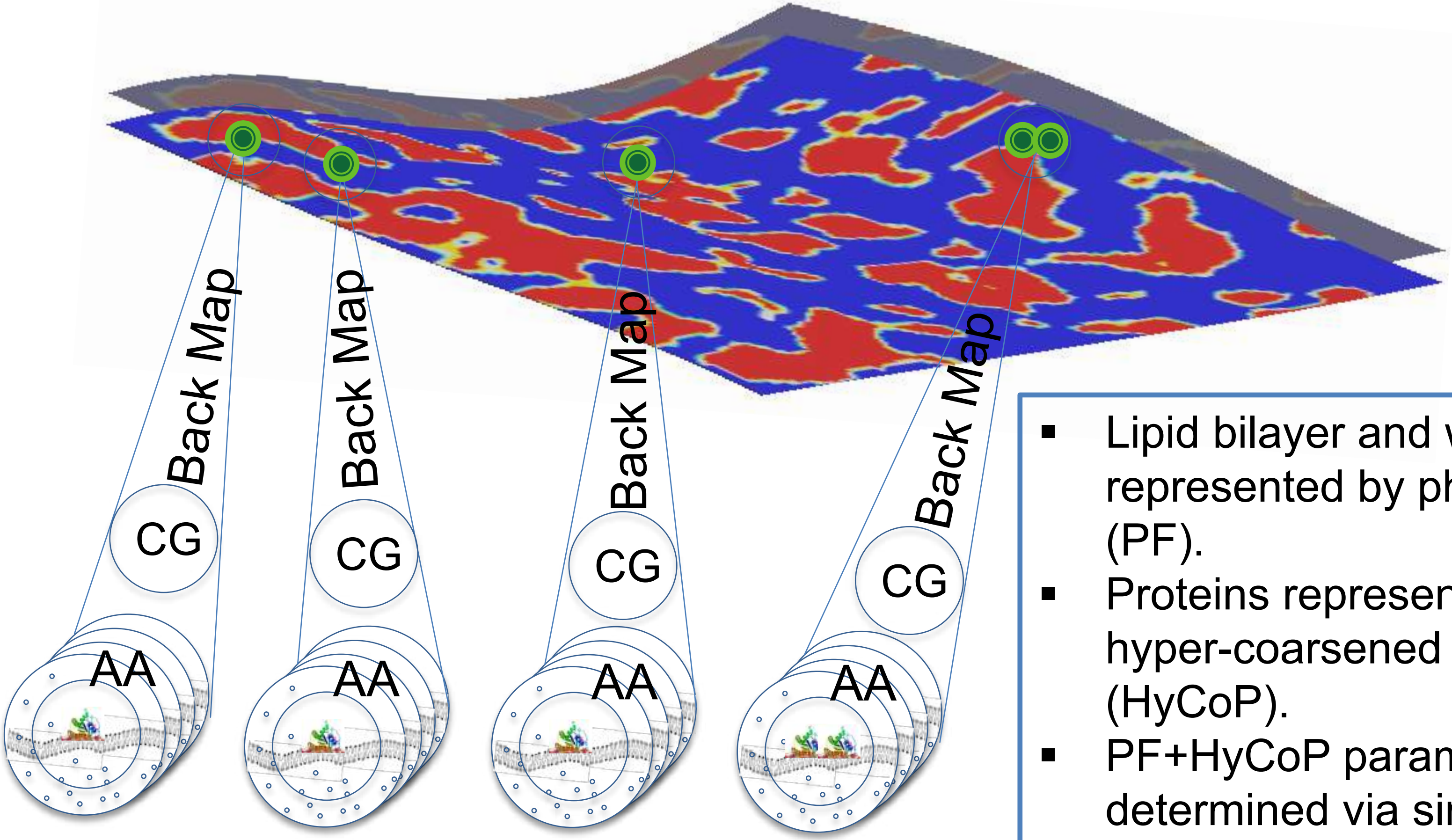
MARTINI (CG)

CHARMM (AA)

c_1, c_2, *h*, *R*, *State*

Back-mapped atomistic regions with RAS proteins

Phase Field + HyCoP informed ensembles of all atom MD simulations



~ 1000,000-atom simulations

- Lipid bilayer and water represented by phase field (PF).
- Proteins represented by hyper-coarsened particles (HyCoP).
- PF+HyCoP parameters determined via simulations and experiment

Moose FEM Phase Field Code from INL

MOOSE is an open source finite element code

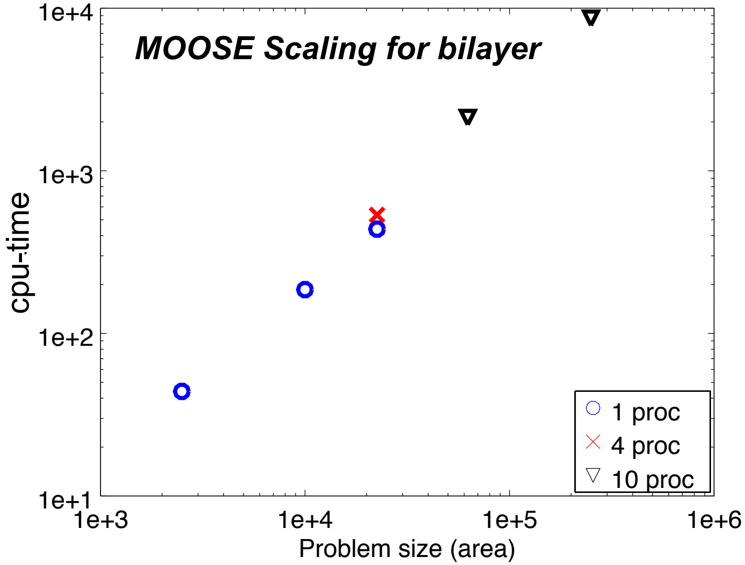
Has support for high order elements (needed for Cahn-Hilliard equation)

Uses PETSc for efficient algebra and solvers

Uses HYPRE multigrid preconditioners

User extendable to arbitrary weak forms

Can run on >10,000 cores



Implicit integrators:

Tune timestep to timescale studied, up to tens of microseconds

Very efficient at driving toward steady state

Cost

1um x 1um patch, at 1nm resolution (~20M degrees of freedom)

40 cores/node, 10 nodes

4000 timesteps per day, or 1-60 milliseconds per day or **0.25-15 us/timestep**

1000-10000 times faster than molecular dynamics, and 100 times bigger area

Proportional to simulated area, and code scales linearly with number of processors.

Simulation challenges

- Time and length scales
- Mapping from low continuum to atomistic simulation
- Extracting biologically relevant and targetable behavior

- Need eco-system of simulator and support software:
  - Continuum simulator
  - Atomistic (molecular dynamics) simulator
  - Mapping software
  - On-the-fly analysis to determine what to zoom in on, and what to report back to user
  - Machine learning libraries
  - Python scripts

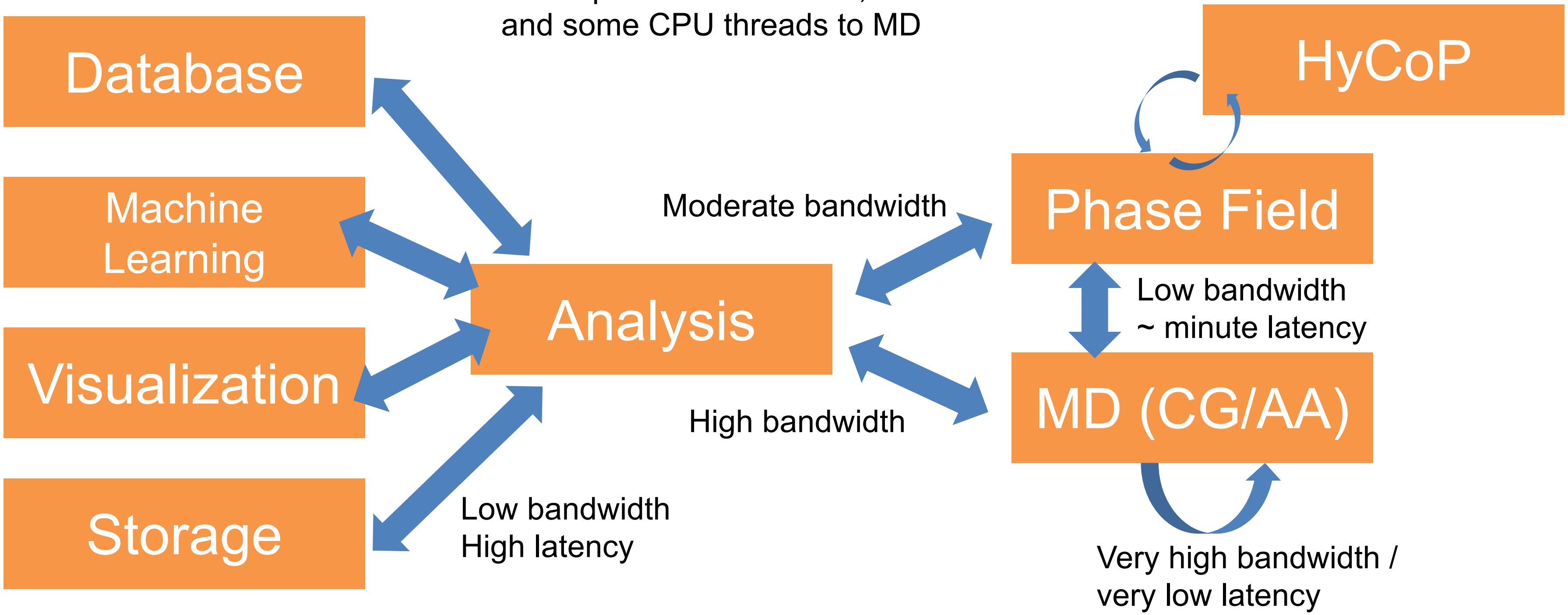
Coupling strategy

- Need to couple multiple loosely connected applications:
  - — Phase field
  - — Molecular dynamics
  - — Continuum-to-atomistic mapping software
  - — Trajectory/configuration analysis
  - — Machine learning
  - — Decision making on creation of new simulations

- New approach: Central data broker
  - — Serves as communication between applications with high latency tolerance (e.g. ms and s, as opposed to µs)
  - — Parallel distributed database
  - — Living largely cached in DRAM
  - — Rapid access to objects through HPC network (e.g. Infiniband)
  - — Simple API for tuple (key/value pair) retrieval
  - — Persistent storage of selected objects – flush to disk

Data flow in multiscale application

Run on each node:
Some analysis threads,
Some phase field threads, allocate all GPUs
and some CPU threads to MD



Database

Machine Learning

Visualization

Storage

Analysis

HyCoP

Phase Field

MD (CG/AA)

Moderate bandwidth

High bandwidth

Low bandwidth
High latency

Low bandwidth
~ minute latency

Very high bandwidth /
very low latency

Note:
MD runs will take (at least) hours (CG) to days (AA),
So MD’s is the only fast data generator, and all other
Bandwidths are low and latencies can be very long
(minutes!).

Challenge: Heterogeneous hardware

Previous experiences

- Running long time and large scale MD on ~100k nodes of BG/L
  - Two Gordon-Bell prizes
    - Kelvin-Helmholtz instability,
    - Solidificaiton with quantum derived potential
- Shock simulation using 6 million MPI tasks
  - Efficient despite very inhomogeneous density distribution

Coming machine

- Coming machine
  - 5 000 nodes
  - 200 000 cores
  - 800 000 threads
  - 600 000 000 GPU threads
- Almost all the flops are on the GPU's
  - Dedicate GPU's to most heavy task
  - Rest of applications can be run as is on conventional CPU's using threads or MPI

Focus on Molecular dynamics

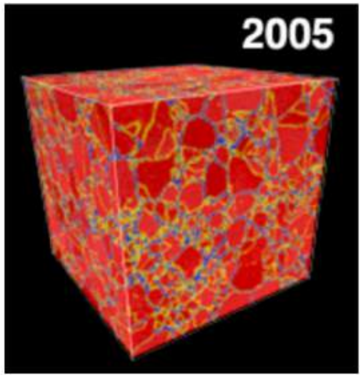
Molecular dynamics component I: Short range potential and MARTINI

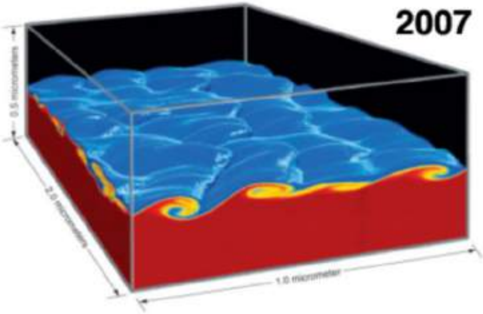
Overview GPU molecular dynamics

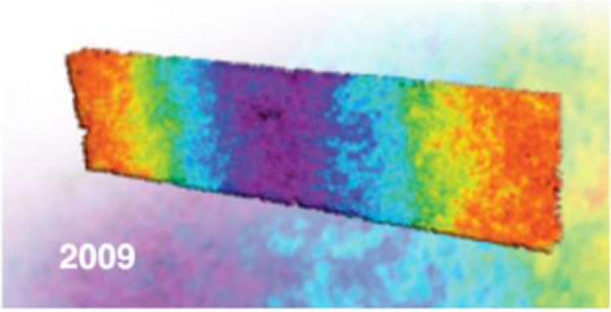
1. Objective: move a massive CPU MD code to GPU, get optimal performance, maintain portability
    1. Choice of language
    2. Code design
    3. Parallelization/offload strategy
2. Portability strategies
3. Speedup results
4. Optimizations
5. Scalability and the Fast Multipole Method

ddcMD intro

- Code is fast (2 Gordon Bell Prizes + 1 Finalist)
  - — Achieves strong + weak scaling
  - — Mostly used for plasma physics, materials science, now biology
- Originally No GPU support, MPI+C
- Data structures not amenable to GPUs
  - — (array of structs, linked lists traversals, function pointers in deep loops)







Molecular Dynamics (MD)
Heavily simplified Neighbor list algorithm

1. Sort particles spatially into 3D bins

2. For each particle, use bins to quickly find all neighbors of said particle within a certain “cutoff” radius. Store results in “neighbor list”

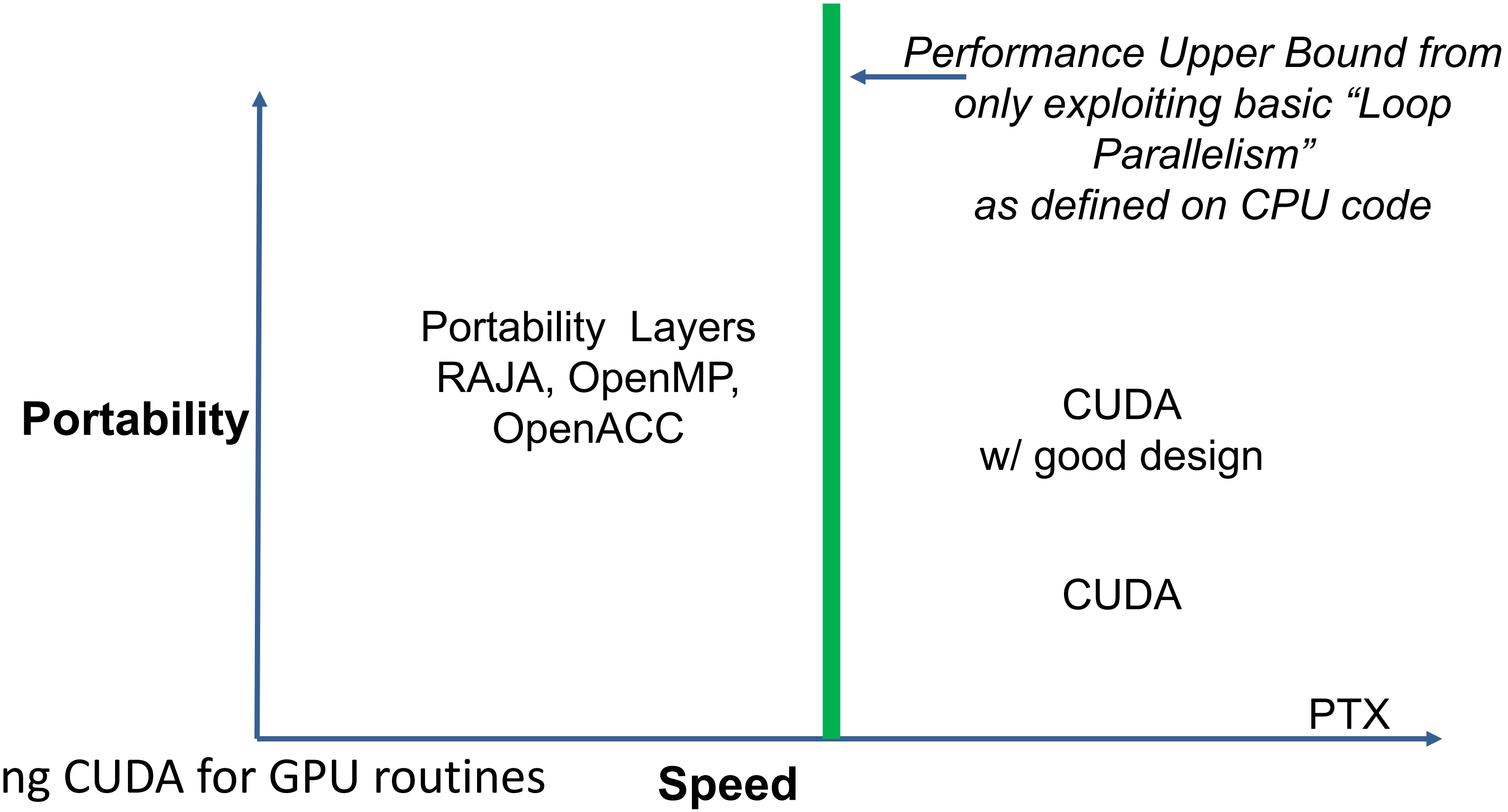
3. Use neighbor list to calculate all forces acting on each particle (aka Force evaluation)

4. Force evaluation for bond forces (bonds are predefined, so this is cheap)

5. Sum forces acting on each particle, use to update velocity and position

6. Repeat 4-5. Only need to reconstruct neighbor list when it becomes “stale”

Choosing Porting Strategy/Language



Performance Upper Bound from only exploiting basic "Loop Parallelism" as defined on CPU code

Portability

Portability Layers
RAJA, OpenMP,
OpenACC

CUDA
w/ good design

CUDA

PTX

Speed

- Using CUDA for GPU routines
  — Need fastest MD possible, need to hand-optimize kernels & mem access patterns

Portability & Performance via Templating & Kernel Inlining

```
double
processPairsCPU<class K>(K Kernel)
{
    for (i,j) in pairs
    {
        r = distance(i,j);
        e[i]+=Kernel.kernel(r, i, j)
    }
}
```

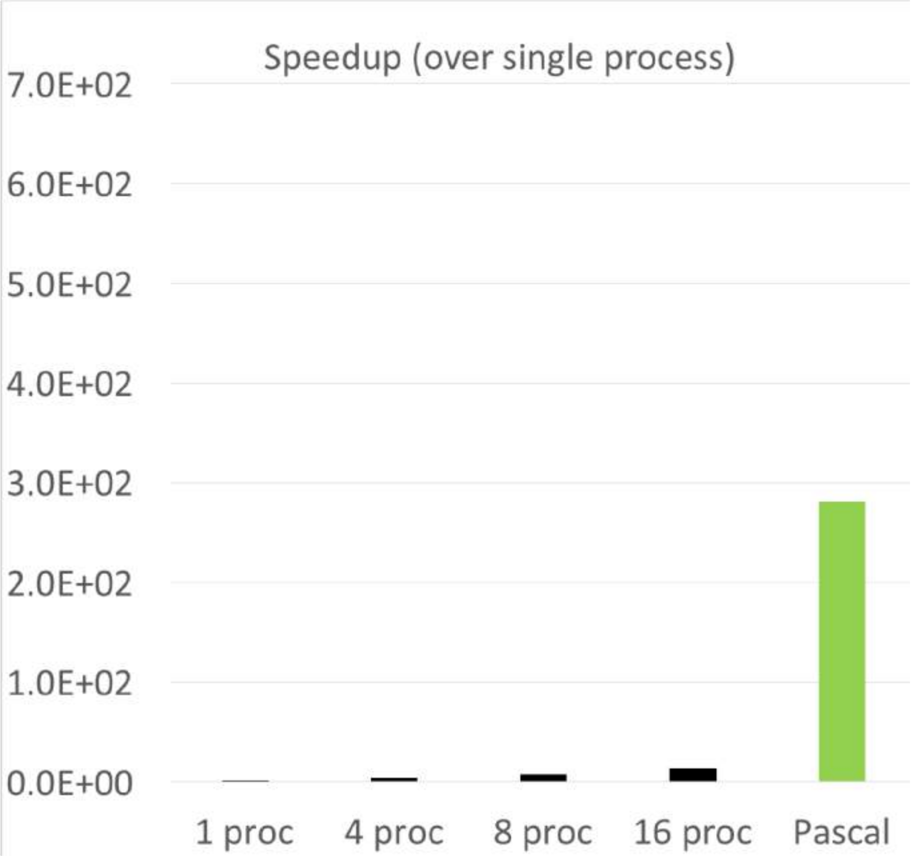
```
__global__ void
processPairsGPU<class K> (K kernel)
{
    int pid = blockIdx.x*blockDim.x+threadIdx
    Particle i,j = pairs[pid];
    r = distance(i,j);
    e[pid]+=Kernel.kernel(r, i, j);
}
```

```
//RUNS ON CPU AND GPU
inline double
 LennardJonesKernel::kernel(double r, int i, int
j)
{
    return 4*epsilon*( (sigma/r)^12 – (sigma/r)^6
);
}
```

Because of templates, users can write custom CPU pair kernels/routines, and run on GPU w/o writing any CUDA

GPU Speedup of Primary Kernel (over single process)

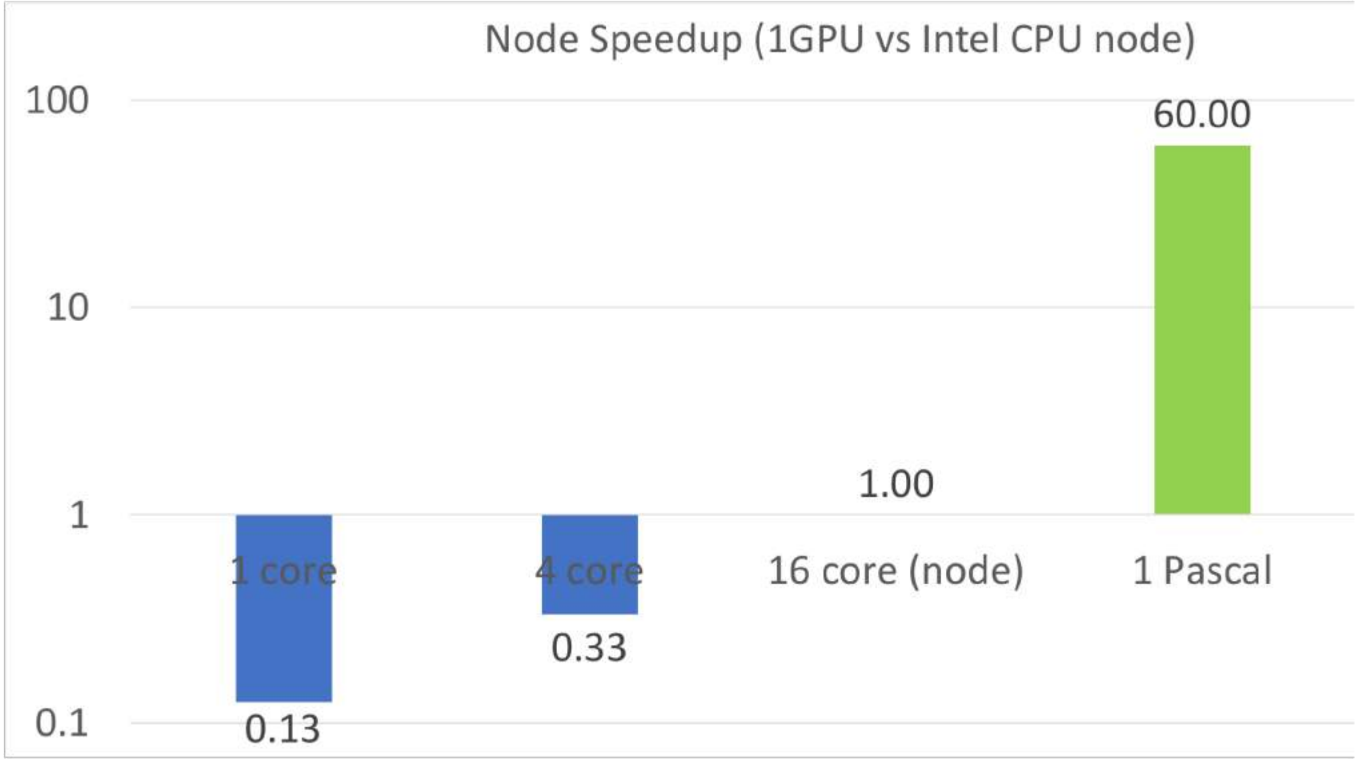
120,000 particle Lennard Jones Ag simulation



1Pascal GPU beats 1 Power8 process by ~280x

Total GPU Speedup (over single node)

120,000 particle Lennard Jones Ag simulation



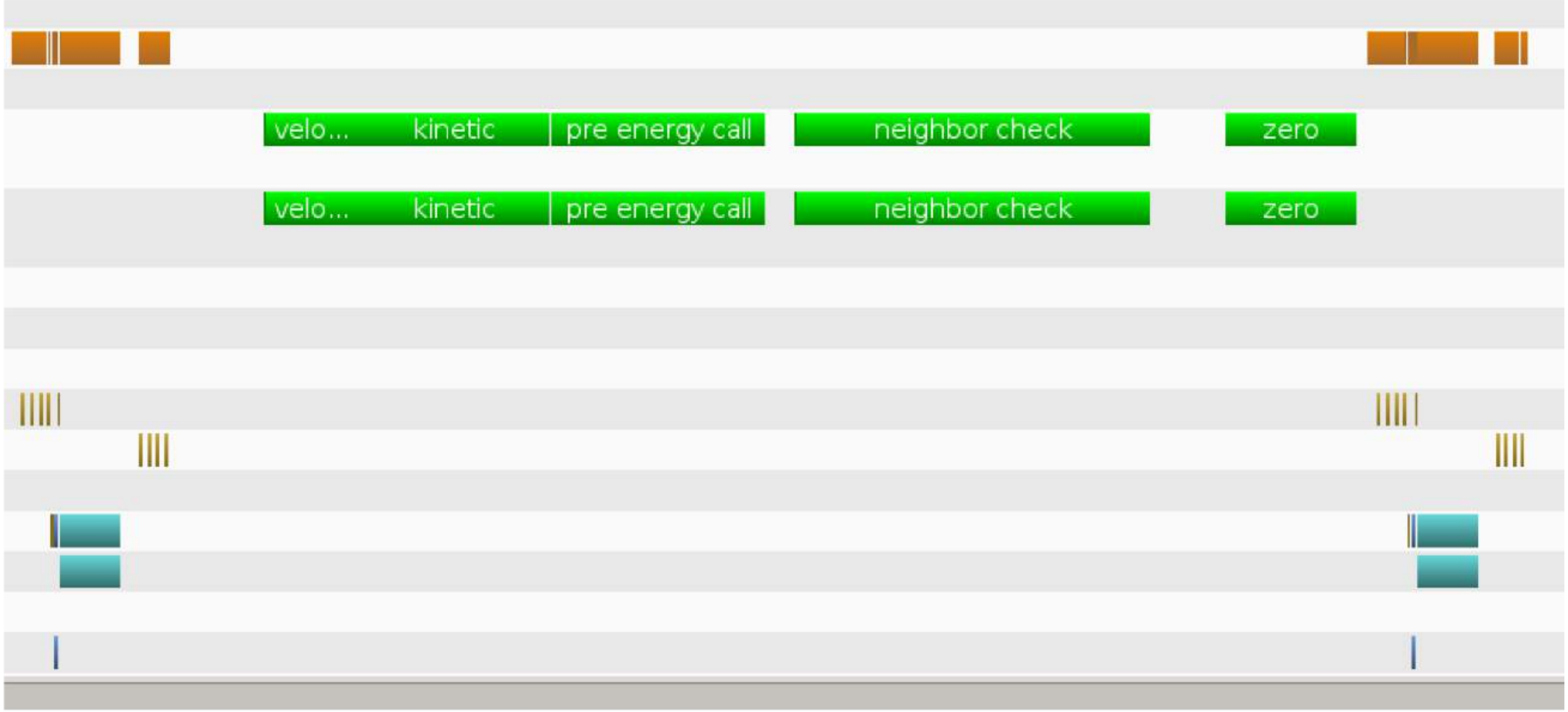
1 Pascal GPU beats full 8-core Intel node by 60x
20 - 30% of peak GPU double precision performance

Porting to GPUs: Avoid CPU Bottlenecks

- Put all non-constant time operations on GPU
  - Otherwise the remaining CPU code will bottleneck you

- Example:
  - **80%** of molecular dynamics runtime/flops = 1 Kernel
    - Force Evaluation
  - **20%** = various other kernels
    - Integration, bonded interactions, binning, etc,
  - Some codes actually leave this 20% on CPU
    - Bad idea

Porting to GPUs
With only the "80% kernel" on GPU



Turquoise boxes = GPU routine   Green boxes = CPU routines

- Green boxes/CPU represent only 20% of flops, but take > 90% of runtime because still on CPU

Porting MD to GPUs (recap)

- Put EVERYTHING on GPU
  - **80%** of molecular dynamics runtime = 1 Kernel
    - Force Evaluation
  - **20%** = various other kernels
    - Integration, bonded interactions, binning, etc,
  - **>12X speedup from putting 20% on GPU**

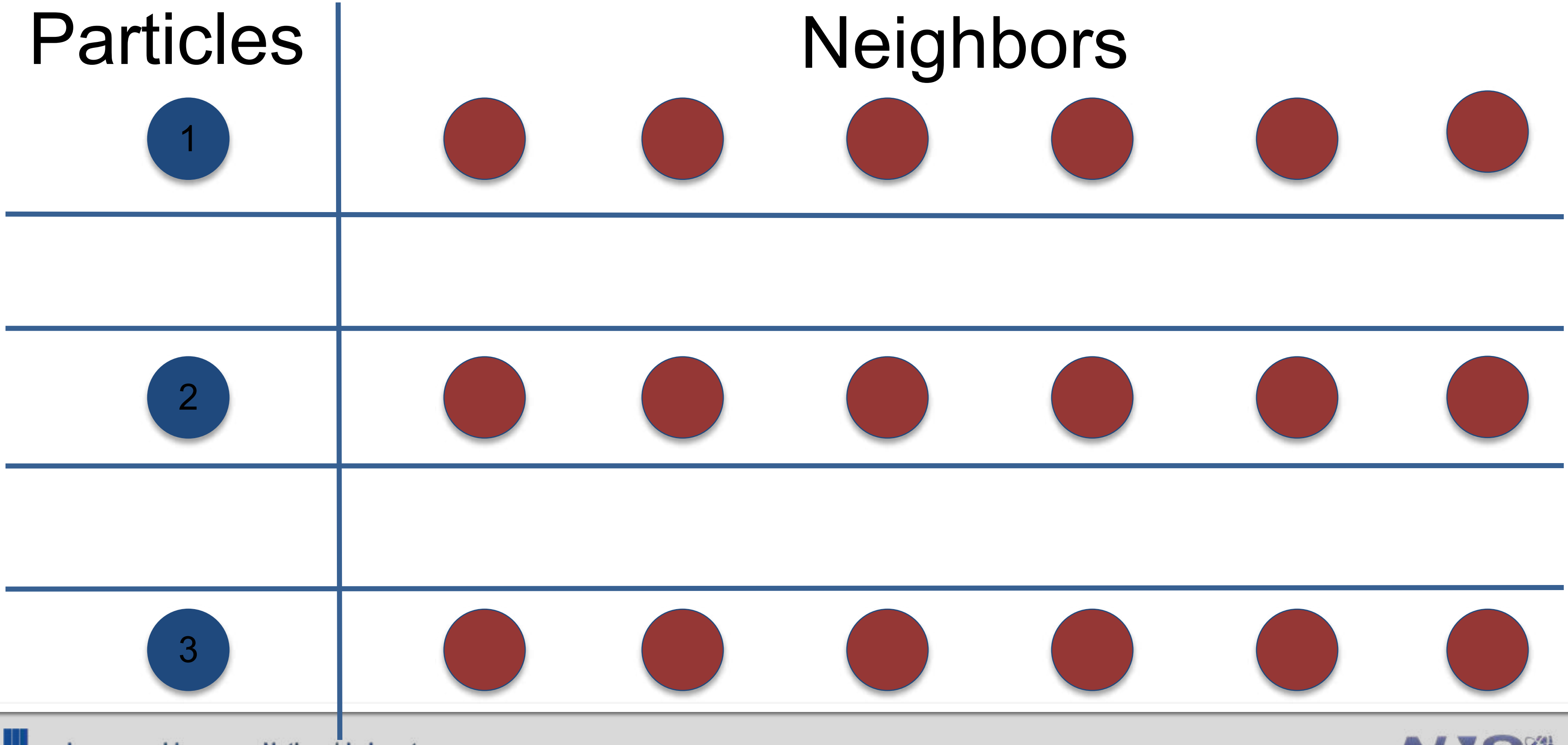
- CUDA
  - Portability layers do not allow for finer optimization control/flexibility

- C++ templating
  - Use templated CUDA kernels to define how to iterate over data
  - Use inlined template functions to define how to process data

Kernel optimizations
Defining proper thread->data iteration strategy

Consider a system with

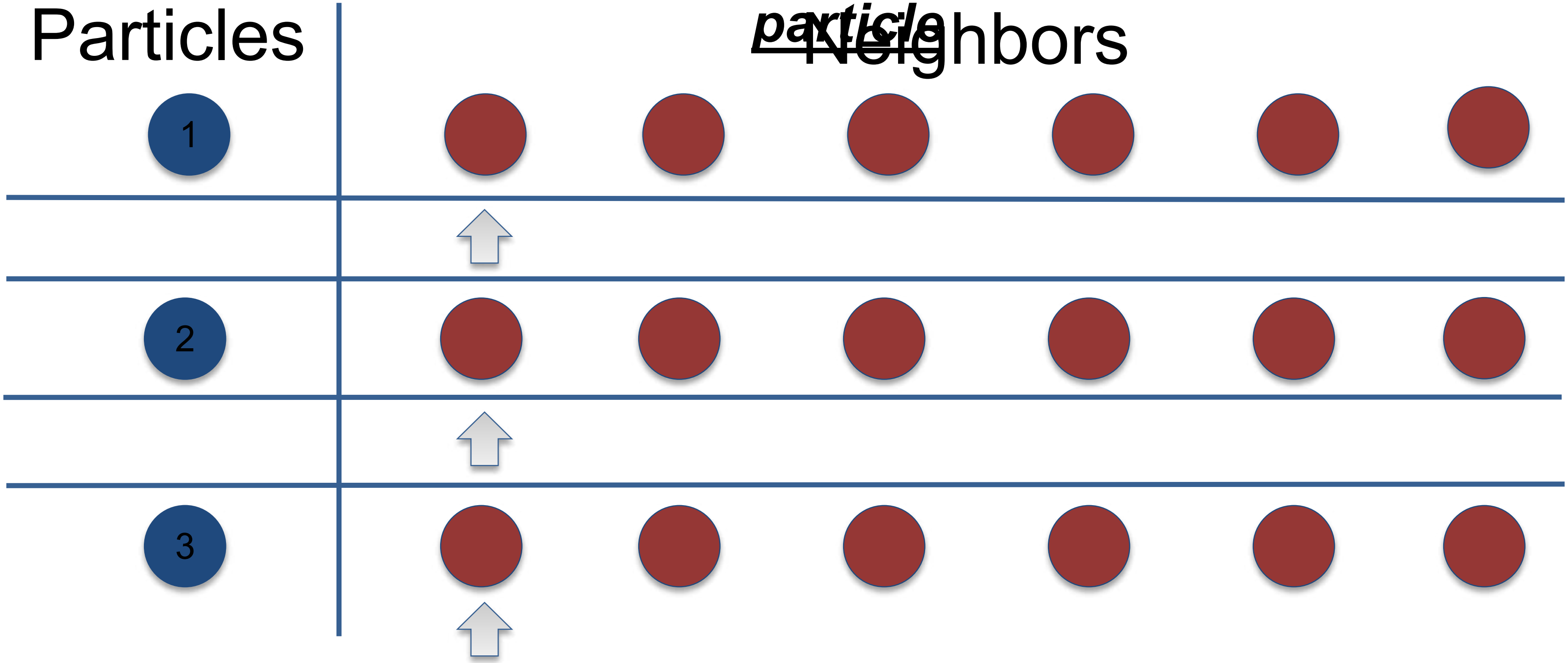
3 particles: ● each w/ 6 neighbo●s: On a "toy" GPU w/ 3 Threads:



Thread->data iteration strategy

3 particles: ● each w/ 6 neighbors: ● GPU w/ 3 ⇧ Threads:

Strategy 1: Assign 1 thread per particle

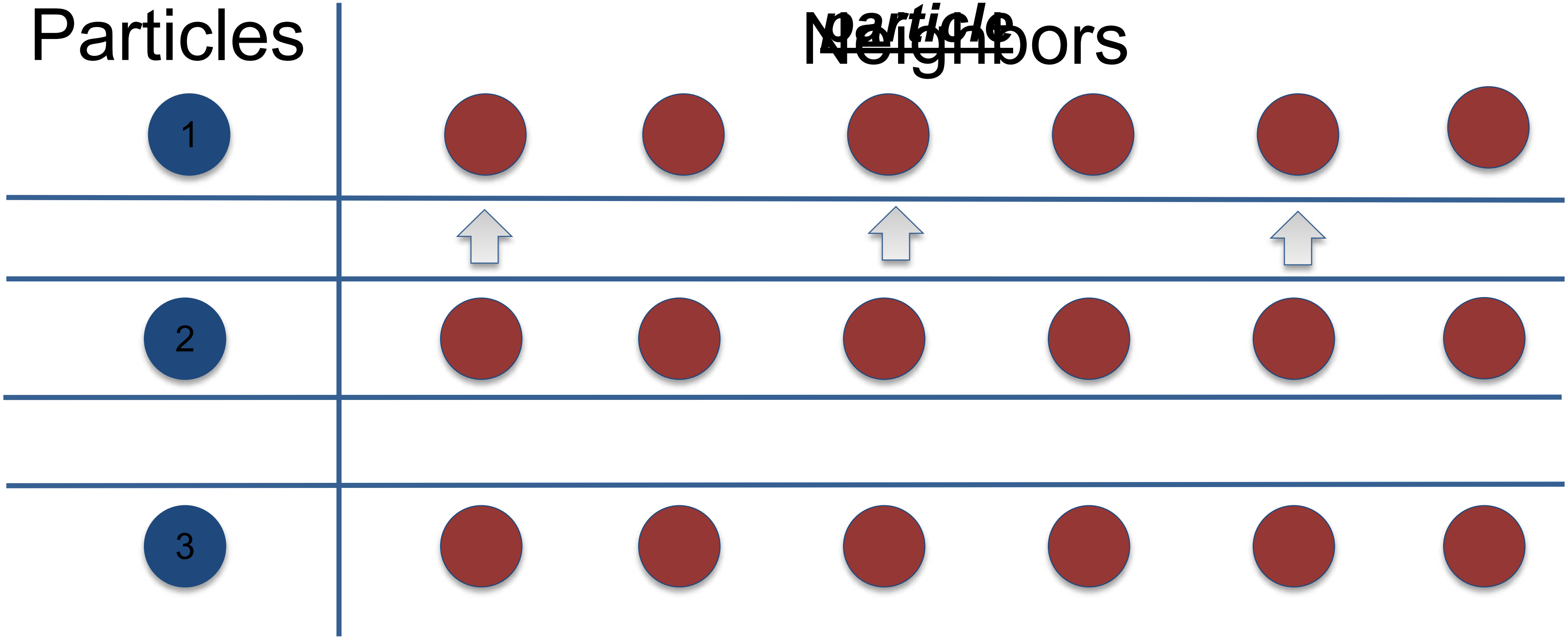


Result: Bad Caching eg each thread reading different parts of memory

Thread->data iteration strategy

3 particles: ● each w/ 6 neighbors: ● GPU w/ 3 ⇧ Threads:

Strategy 2: Assign multiple threads per particle

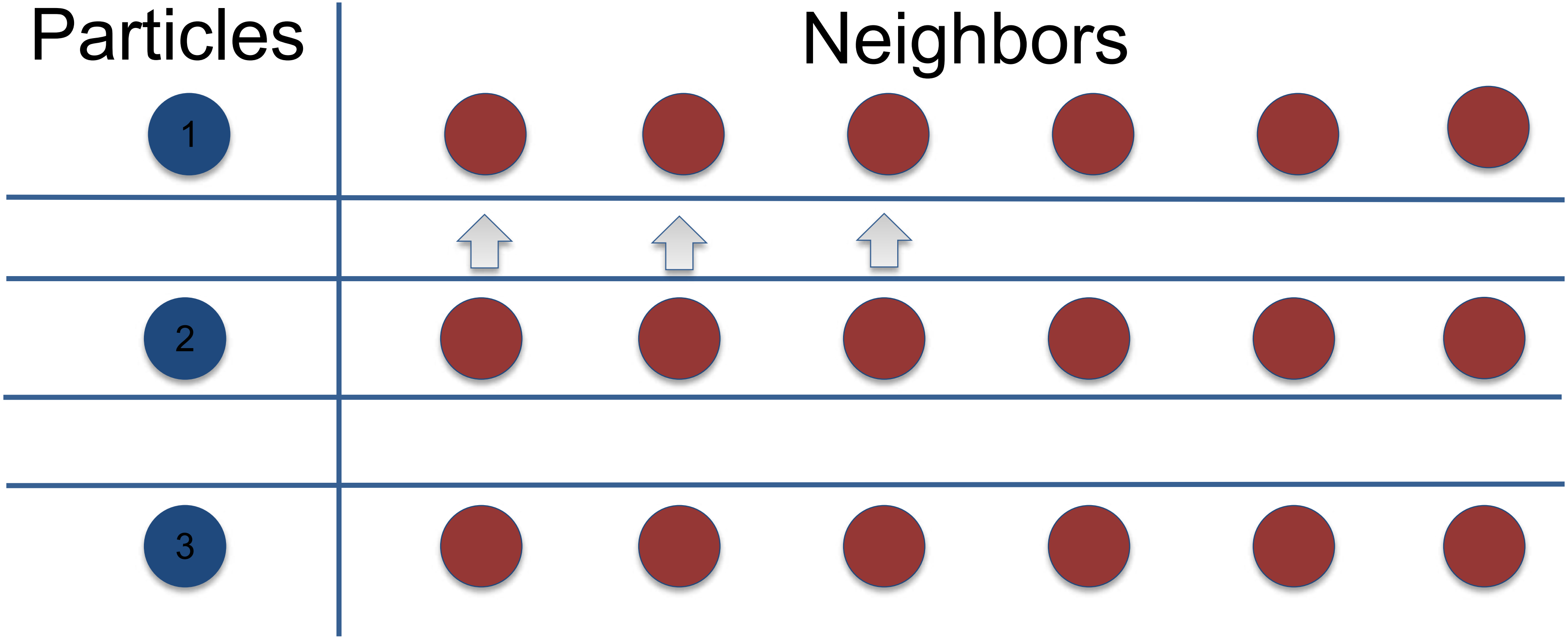


Result: Better Caching (& more threads, if resources not already maxed out). But threads not reading contiguous memory/coalesced

Thread->data iteration strategy

3 particles: ● each w/ 6 neighbors: ● GPU w/ 3 Threads: ⇧

Strategy 3: Assign multiple threads/particle, coalesced



Result: Better Caching, more threads, coalesced accesse

also % peak performance increases significantly with bigge

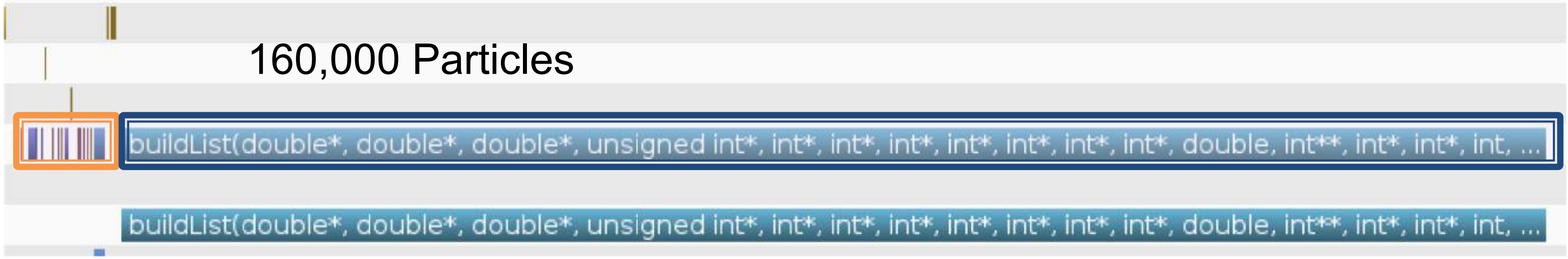
Thread->data iteration strategy recap

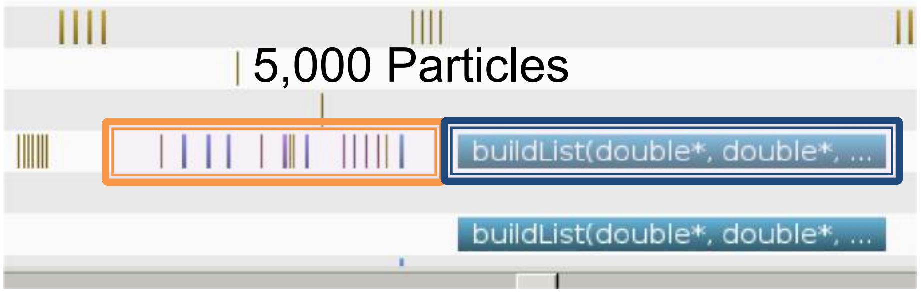
- Memory
  - Spawn multiple threads per particle
    - A team of N threads to process all of a particle's neighbors
    - Better locality, more threads
    - 3x performance increase
  - Use of shared memory
    - Scratch space
    - 30% performance increase vs global memory
  - Optimize memory access pattern & striding
    - Understand which threads in your block are actually warp contiguous
    - avoid bank conflicts & achieve better locality
    - another 50% perf increase

- Sharing values across threads
  - Can store in shared memory (threads in same block).
  - But can also shuffle sync to keep memory in REGISTERS
    - Great for reductions
    - Can double performance of non-neighbor list kernel when coupled with good caching strategies

Strong scaling optimizations

- 160,000 particles/GPU → 5,000 particles (strong scaling)
- The bars within the colored boxes = kernel runtime
- Empty space in colored boxes = kernel initialization


160,000 Particles
buildList(double*, double*, double*, unsigned int*, int*, int*, int*, int*, int*, int*, int*, double, int**, int*, int*, int, ...
buildList(double*, double*, double*, unsigned int*, int*, int*, int*, int*, int*, int*, int*, double, int**, int*, int*, int, ...


5,000 Particles
buildList(double*, double*, ...
buildList(double*, double*, ...

Strong scaling optimizations

- 160,000 particles/GPU → 5,000 particles
- …No longer bottlenecked by kernel run time (minimal)
- Kernel "launch" time matters
  - Cuda 8: need a new kernel for every cross-block synchronization
  - More global synchs = more launch time
- Cuda 9 solution: Cooperative groups for cheaper cross block synchronizations
  - >~2x-4x faster
  - Limitation: your number of threads cannot exceed capacity of GPU

Focus on Molecular dynamics

Molecular dynamics II: Long-range electro-static interactions and CHARMM

Molecular dynamcs at 1µs per day

- Target is 1 million atoms at 1 µs/day
  - Estimate to achieve that on around 50 nodes of Sierra.
- Timestep is around 2 fs
  - Need 5 000 - 10 000 timesteps per second to reach goal
- Atoms interact with neighboring atoms through bond forces and van der Waal forces. These have limited range, and so incur a constant cost per atom to calculate.
- Atoms may also be charged, which is an interaction with infinite range. An efficient method is needed to avoid calculating distances between all pairs of atoms O(N^2) cost per timestep.

Long-range (Coulomb) interactions

- Traditional Coulomb solvers (e.g. PME, P^3M) use Fourier methods
  - High global communication demads
  - Multiple communications / transposes of all data each timestep
- The fast multipole method (FMM) is a hierarchical method that achieves calculation of all N^2 interactions in O(N) time with some prescribed accuracy
- FMM is computationally more intensive, but has sparse communication pattern
  - **Trade flops for communication!**
- It also allows a variety of boundary conditions to be applied in an efficient manner (e.g. supports periodic boundary conditions)
- Half the work is through cell-cell interactions, and half the work is in particle-particle interactions.

Design for Sierra

- Have done flop and bandwidth estimations of central operators:
  - Need about 100k flops per particle
  - Memory and L2 bandwidth on Pascal sufficient to support core operators at peak flops.

- Have looked at efficiency in scalar fortran code:
  - Single precision and expansion to order 10 multipoles is sufficient for most calculations
  - Unoptimized fortran code runs at ~50% of scalar peak.
  - Exploit symmetry in direct pair kernel, still runs at >50% of peak flops.
  - Most expensive pair kernel, multipole-to-local conversion, runs at ~35% of peak.
  - Overall code runs at more the 40% of peak performance.

- Have tested communication pattern on Ray:
  - Can send and receive data for ~20k particles in <100us using one communication task per node
  - That means ~10k timesteps per second is achievable from a latency perspective

Communication

- Important to overlap communication and computation

- The FMM method with 512 cells and 20k particles per node allows:
  - About 60% of work can be performed before communication completes
  - Only small amount of work needs to be done in each timestep before communication begins

- We expect to be able to overlap a substantial amount of communication wait times with computational work.

FMM optimizations

- Use O(p^3) translation operators, instead of naïve O(p^4) operators
- Exploit real nature of diagonal M2L operator
- Use symmetry to avoid complex-complex multiplications in rotation operators
- Our O(p^3) operator uses 6 times fewer flops than optimized O(p^4) operators, even for expansion order 10.

Running on GPU’s

- Particle-particle interactions has lots of data-reuse and is computationally dense.
- There are 512 x 13 particle-particle calculation batches, each which could be made to use at least one warp
- There are 512 x 216 cell-cell interactions per node per timestep
- Each cell-cell interaction operator can utilize a full warp, with less than 15% of padding (i.e. calculations on zeros / wasted flops)
- Total of over 100k warps to issue per timestep should allow good occupancy on Pascal/Volta GPUs (4 per node, so 50k warps per GPU).

GPU performance on Pascal and Maxwell

- Direct Coulomb pair-kernel runs at >50% of peak performance, even while exploiting symmetry
- Most expensive translation operator runs at >35% of peak
- Overall force evaluation runs at about 40% of peak.

- Network can sustain ~8,000 timesteps per second in a scalable fashion.

- Scaling studies on to 16 nodes on pre-Sierra hardware (Pascal GPU's) show good weak and strong scaling.

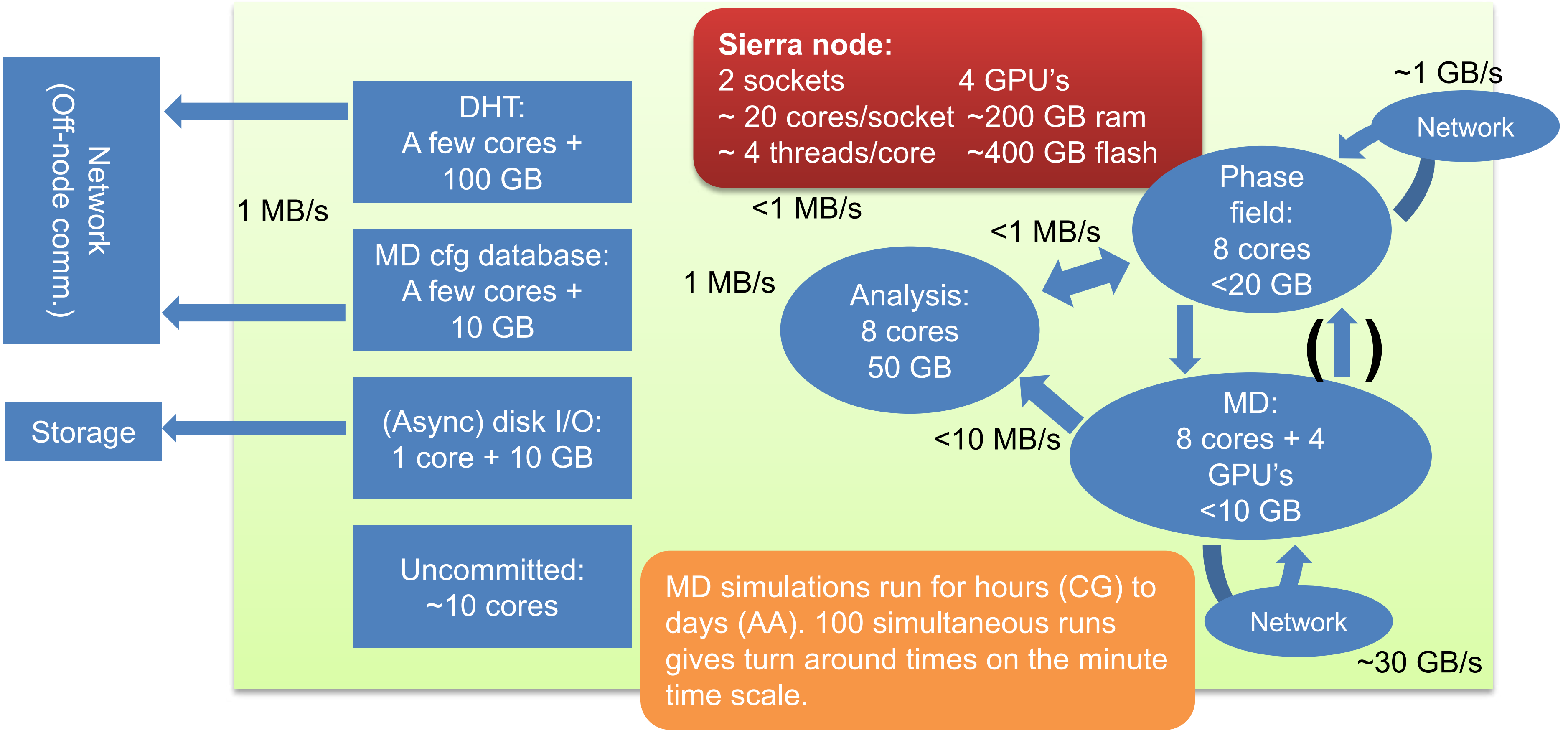
Fast molecular dynamics summary

- Theoretical estimates and code benchmarks indicate we should be able to achieve good utilization and high efficiency running molecular dynamics on the Sierra nodes
  - Dense algebra kernels with good data reuse
  - Many independent calculations gives good pipelining
  - Can overlap most of the communication with computation
  - Network fast enough to support target of 10 000 timesteps per second

- Sparse / local communication pattern should allow efficient scaling to full machine

- With this performance <10% of Sierra can outperform the Anton-2 machine

Application hardware layout



Network (Off-node comm.)

Storage

1 MB/s

DHT:
A few cores +
100 GB

MD cfg database:
A few cores +
10 GB

(Async) disk I/O:
1 core + 10 GB

Uncommitted:
~10 cores

Sierra node:

2 sockets	4 GPU's
~ 20 cores/socket	~200 GB ram
~ 4 threads/core	~400 GB flash

<1 MB/s

1 MB/s

Analysis:
8 cores
50 GB

<1 MB/s

Phase
field:
8 cores
<20 GB

~1 GB/s

Network

MD:
8 cores + 4
GPU's
<10 GB

<10 MB/s

Network

~30 GB/s

MD simulations run for hours (CG) to days (AA). 100 simultaneous runs gives turn around times on the minute time scale.

Lawrence Livermore
National Laboratory