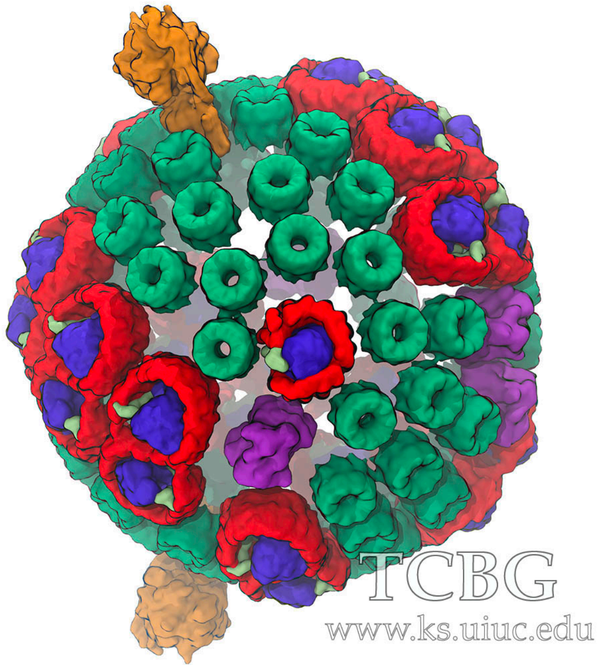
NAMD
Scalable Molecular Dynamics



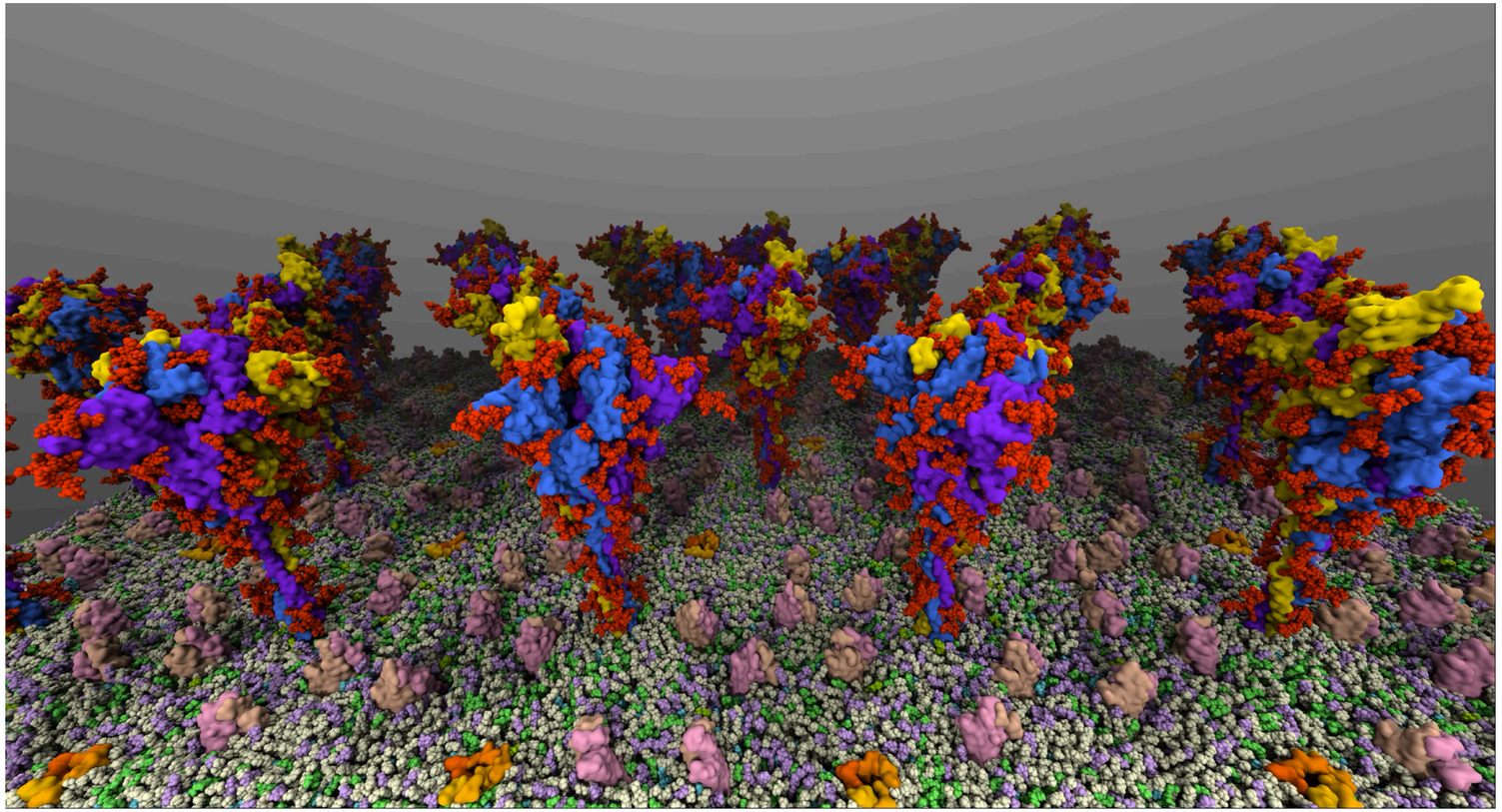
NAMD 3.0 Features, Performance, and Capabilities

David J. Hardy
Senior Research Programmer
University Of Illinois at Urbana-Champaign

NIH National Institutes of Health

NAMD: Scalable Molecular Dynamics

- Popular parallel MD code capable of scaling to tens of thousands of CPU cores and thousands of GPUs
- Developed and trusted by scientists since the mid '90s
- Written in C++ with CUDA and using Charm++ parallel objects
- Full-featured MD application with many advanced features:
  - Free energy methods
  - Enhanced sampling methods
  - Built-in collective variables (Colvars) module
  - Customizable user scripting with Tcl and Python



Investigations of coronavirus (SARS-CoV-2) spike dynamics.
Credit: Tianle Chen, Karanpal Kapoor, Emad Tajkhorshid (UIUC).
Simulations with NAMD, movie created with VMD.

Phillips, et al. *J. Comput. Chem.* 26, 1781-1802 (2005)
Phillips, et al. *J. Chem. Phys.* 153, 044130 (2020)

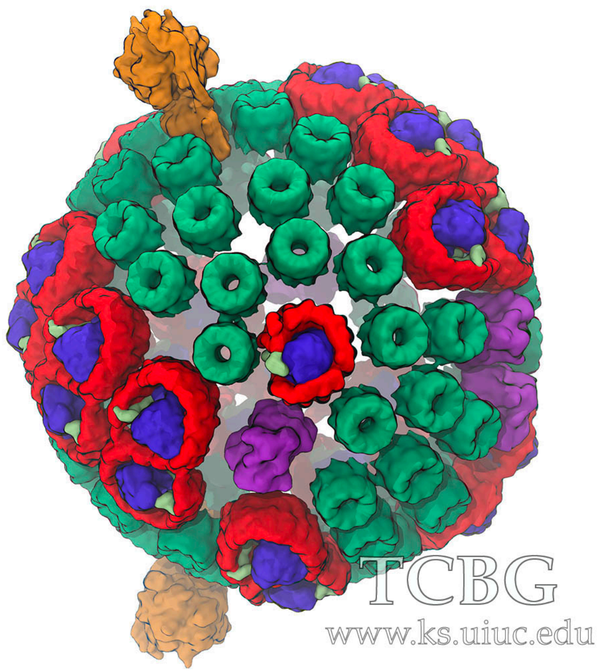
NIH National Institutes of Health

MD Simulation and Parallel Scaling

- **Strong scaling**: Single simulation scaled across multiple computational resources
  - CPU-based — 2-10k atoms per core
  - GPU-based — 20-100k atoms per device
- **Weak scaling**: Multi-copy / replica-exchange simulation in which the total number of simulations scales with the computational resources
  - Determine most efficient use of resources for your simulation (e.g., one copy per node or per GPU device)
  - Scale your total number of copies accordingly (up to whatever makes sense for your scientific investigation and resource allocation)

NAMD force field and modeling support

- CHARMM force field
- AMBER (file and force field support)
- GROMACS (some file support)
- Drude polarizable force field
- Water models: TIP3P, TIP4P, SWM4-NDP (Drude)
- MARTINI residue-based coarse-grained (limited support)
- File support:
  - PDB files (reads ATOM records)
  - PSF files (CHARMM and X-PLOR formats)
  - Force field files (CHARMM19, 22, 27, etc.)
  - DCD trajectory files
  - NAMD binary files



NAMD standard features (equilibrium simulation)

Features less common or distinctive among MD codes denoted by →

- Constant energy
- Temperature control
  - Langevin thermostat
  - Stochastic velocity rescaling
  - Berendsen heat bath (tCouple)
- Pressure control
  - → Langevin piston
  - Berendsen pressure bath
- Periodic boundary conditions
- Non-periodic with spherical or cylindrical BCs

- Long-range electrostatics
  - Particle-mesh Ewald (for PBCs)
  - → Multilevel summation method (for non-periodic or semi-periodic BCs)
- Rigid bond constraints for hydrogen
- → Multiple time stepping
- → Conserve momentum while still conserving energy (zeroMomentum)
- Energy minimization
  - Conjugate gradient and velocity quenching

NAMD advanced features (non-equilibrium simulation)

- Enhanced sampling methods
  - Apply external forces:
    - Harmonic restraints, fixed atoms, external electric field, steered MD, interactive MD, grid forces
  - Boost or modify interaction potentials to flatten the energy landscape:
    - Accelerated MD, Gaussian-accelerated MD, solute scaling and REST2, replica-exchange MD
- Collective variables (Colvars) module
- Alchemical free energy methods
  - Free energy perturbation (FEP)
  - Thermodynamic integration (TI)
- Constant pH simulation
- Hybrid QM/MM simulation
- Tcl (and Python) scripting interface accessed through the NAMD configuration file

NAMD 3.0 beta released

- New **GPU-resident mode** for very fast dynamics:
  - Achieves 2x or more speedup on single GPU versus GPU-offload simulation
  - Efficient single-node multi-GPU scaling for tightly coupled GPU architectures (e.g. DGX)
  - Supports both NVIDIA and AMD GPUs
- GPU support for **alchemical free energy methods** (FEP and TI)
- GPU-resident provides some advanced feature support:
  - harmonic restraints, external electric field, steered MD, REST2, replica-exchange MD
  - **Monte Carlo barostat**
  - **group position restraints**

available only for GPU-resident mode

- New CPU vectorization mode supporting AVX-512 instructions (Intel Xeon and AMD Zen4)

NAMD hardware and feature support

- CPU-based (x86, ARM, Power, KNL, ...)
  - AVX-512 accelerated — implements non-bonded tiles optimization from CUDA
- GPU-accelerated (download builds **-CUDA** for NVIDIA and **-HIP** for AMD)
  - GPU-offload — force calculation offloaded to GPU device
  - GPU-resident — (almost) all calculation performed by GPU device, atom data resides on device between time steps
    - Available in **multicore-CUDA** and **netlrts-smp-CUDA**
    - Enable mode with config file keyword: **CUDASOAintegrate on**
- Advanced feature support is limited for accelerated modes
  - Greater acceleration provides less feature support

Molecular Dynamics Simulation

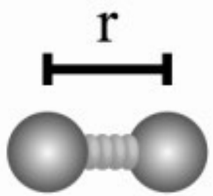
Integrate Newton's equations of motion:

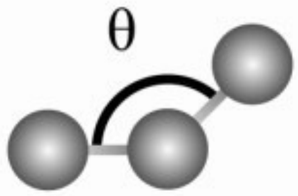
$$m_i \frac{d^2 \vec{r}_i}{dt^2} = \vec{F}_i = -\vec{\nabla} U(\vec{R})$$

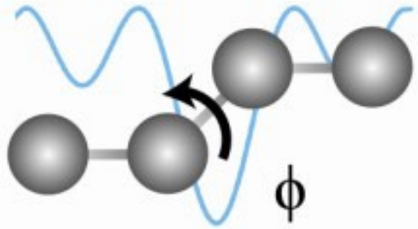
Integrate for millions of time steps

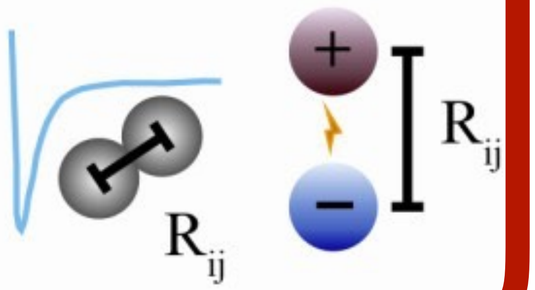
Bonded **Non-bonded**

$$E_{total} = \sum_{bonds} K_r (r - r_{eq})^2 + \sum_{angles} K_\theta (\theta - \theta_{eq})^2 + \sum_{dihedrals} \frac{V_n}{2} [1 + \cos(n\phi - \gamma)] + \sum_{i<j} \left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\epsilon R_{ij}} \right]$$





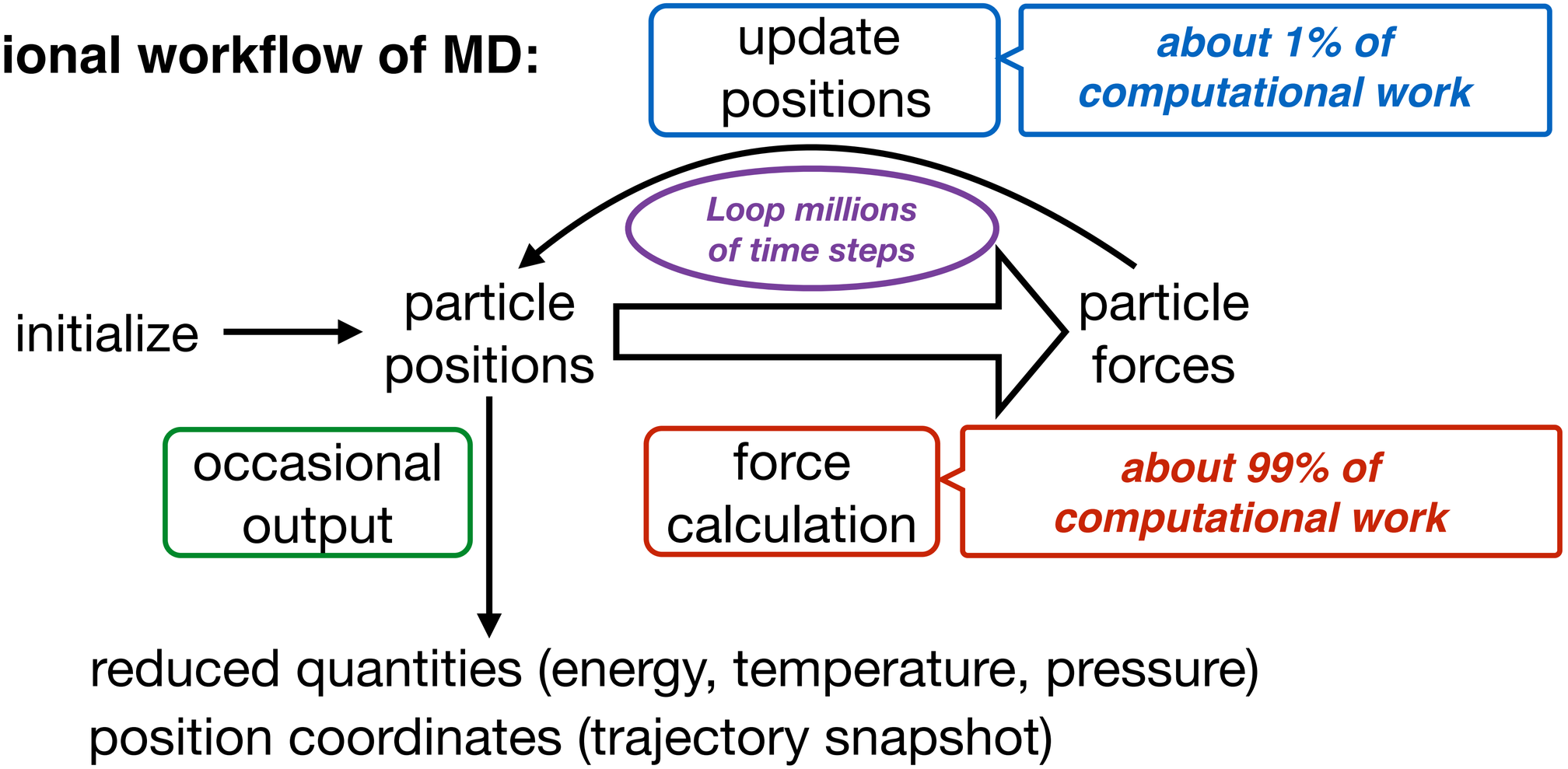




Most computationally intensive part

Parallelism for MD simulation limited to each time step

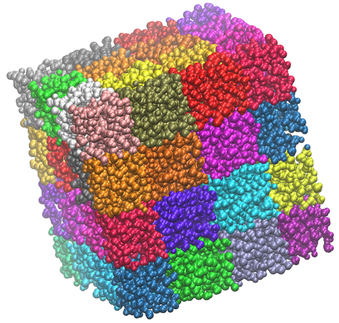
Computational workflow of MD:



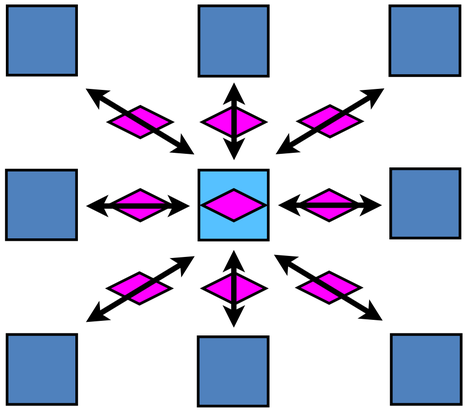
Decomposition of data **and compute objects**

- Decompose atoms into equal volume ***patches***
- Calculate pairwise forces between atoms, treat as interactions between neighboring patches
- Decompose patch-patch interaction ***compute objects***
- Moving atoms: update spatial decomposition by ***migrating atoms*** between adjacent patches
- Load balancing: update work decomposition by ***migrating compute objects*** to keep processors consistently occupied

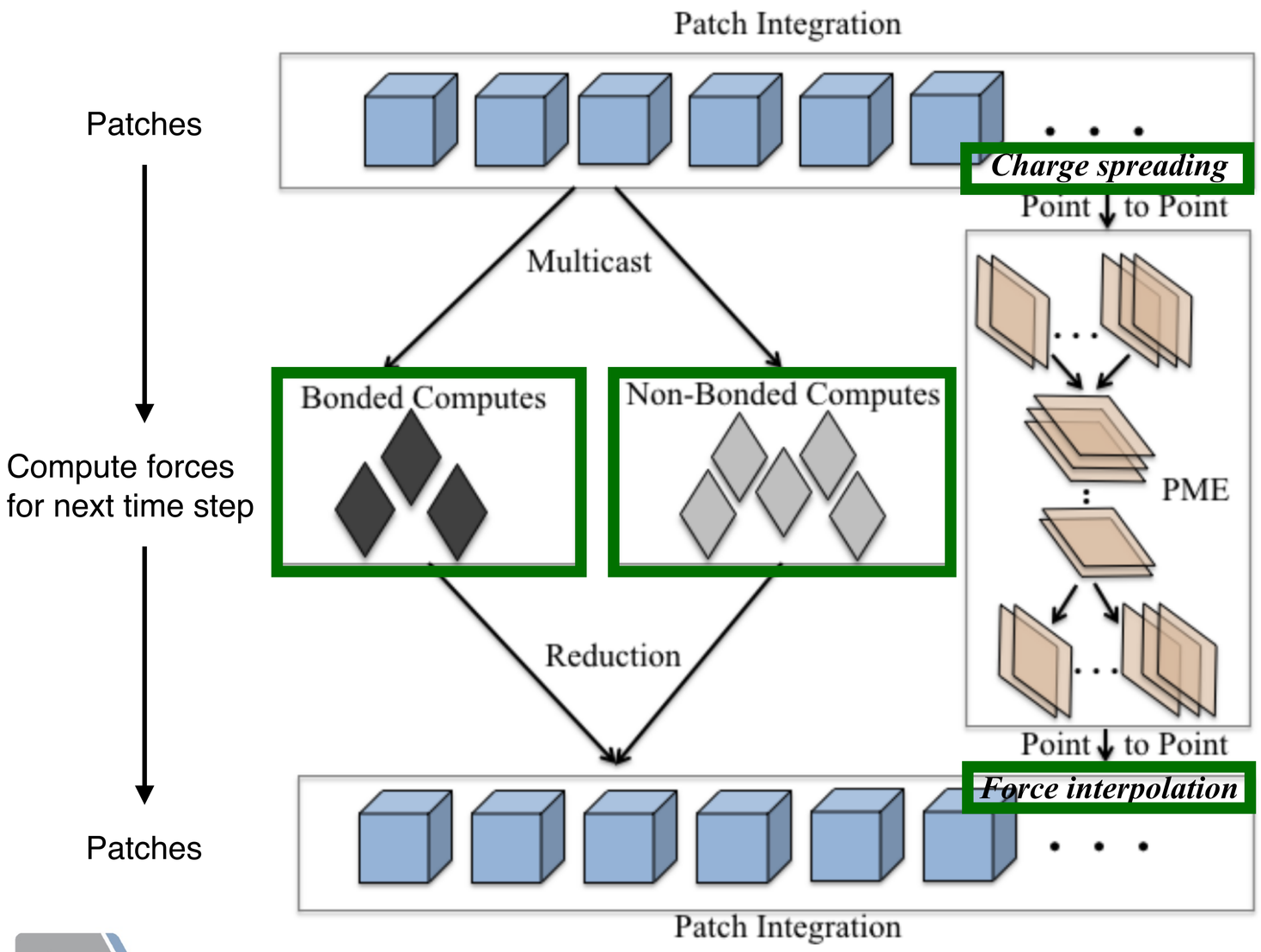
Spatial decomposition of atoms into patches



Work decomposition of patch-patch interactions into migratable compute objects



Using GPU-offload approach for multi-node simulation

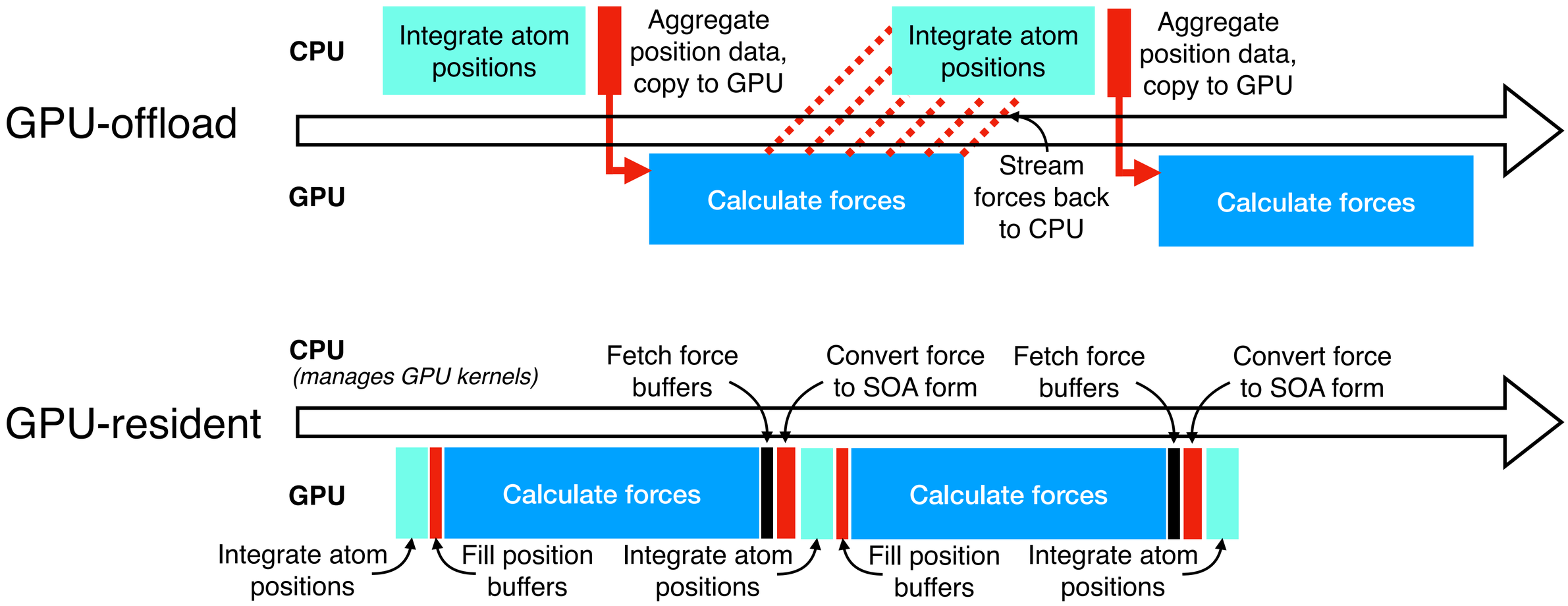


Offload force compute to GPU

Must aggregate positions

New GPU-resident approach

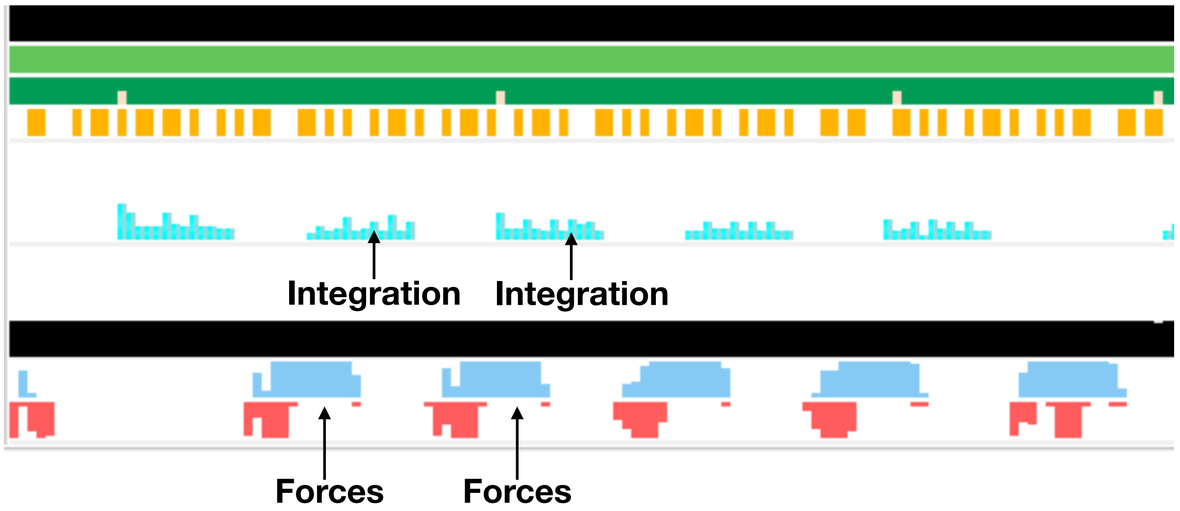
Move integrator to GPU and maintain data between time steps



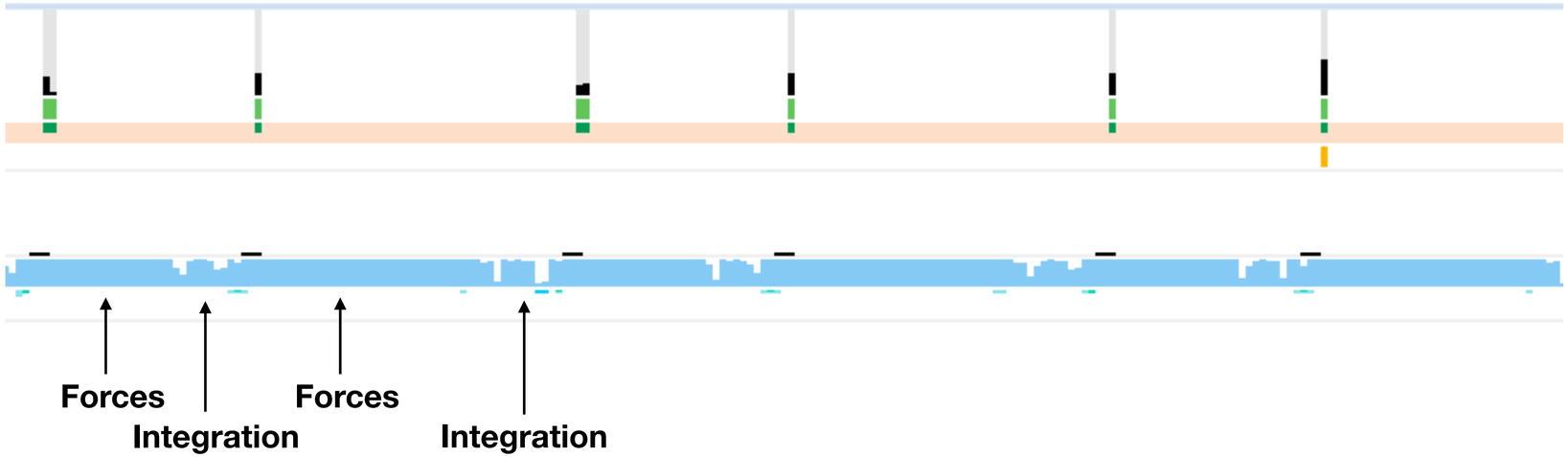
New GPU-resident approach

Profiling shows new approach fully utilizes GPU, no more CPU bottleneck

Before (GPU-offload):



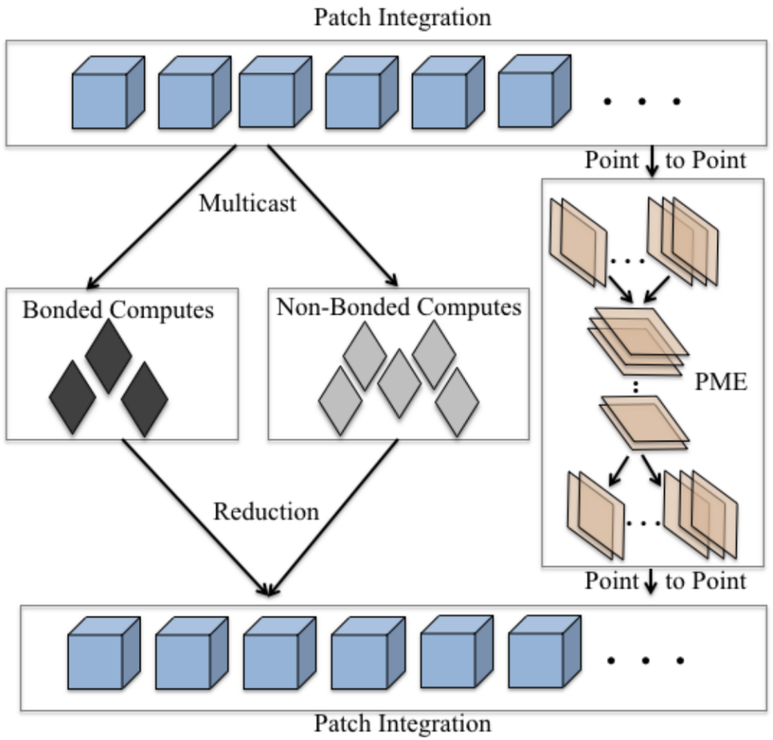
After (GPU-resident):

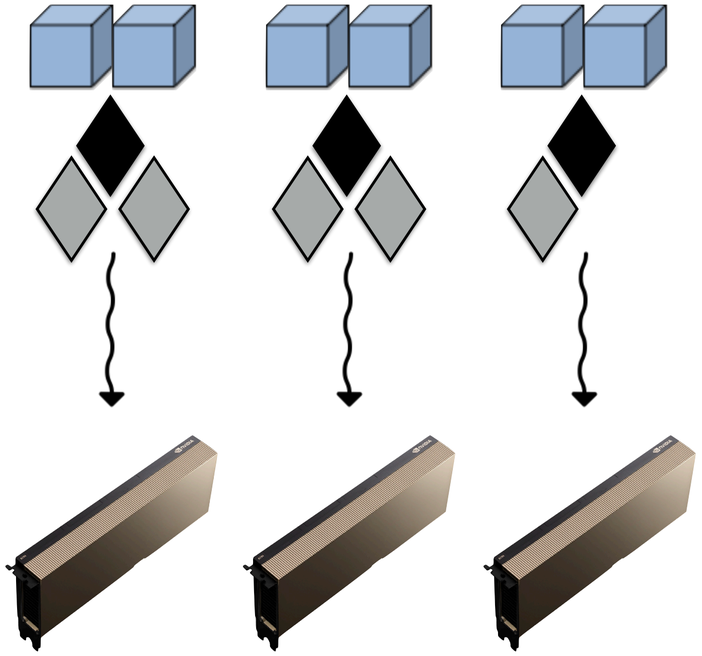


Adapting parallel scaling to GPU-resident approach

Apply similar decomposition of data and work among GPUs

- Take a conceptually similar approach, except we **must have SoA** (structure-of-arrays) data layout for performance
- Each CPU thread binds to a particular GPU
- Aggregate compute and patch data per thread to launch integration and force kernels
- Exploit tightly coupled (peered) GPUs (NVLink, PCIe, …)



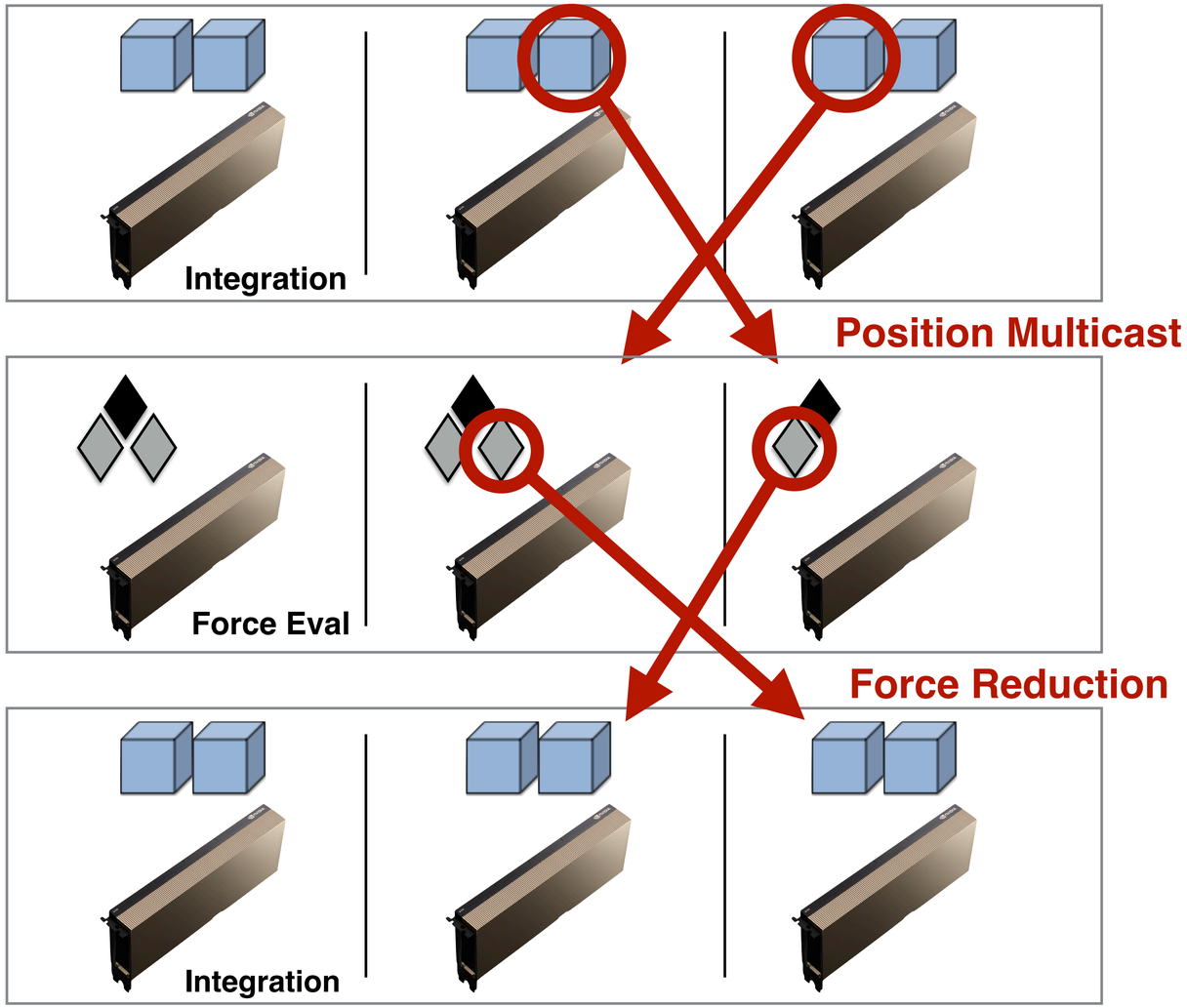


Adapting parallel scaling to GPU-resident approach

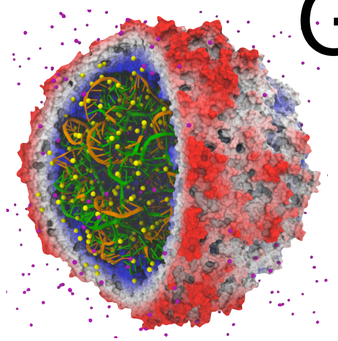
Some communication required: multicasts and reductions

- Update atom positions in each patch during integration
- Perform **position multicast** into compute objects
- Compute new forces
- Perform **force reduction** back to patches
- GPUs need load-store memory access between different devices ***within every time step***, with data sizes on the order of 8KB per access

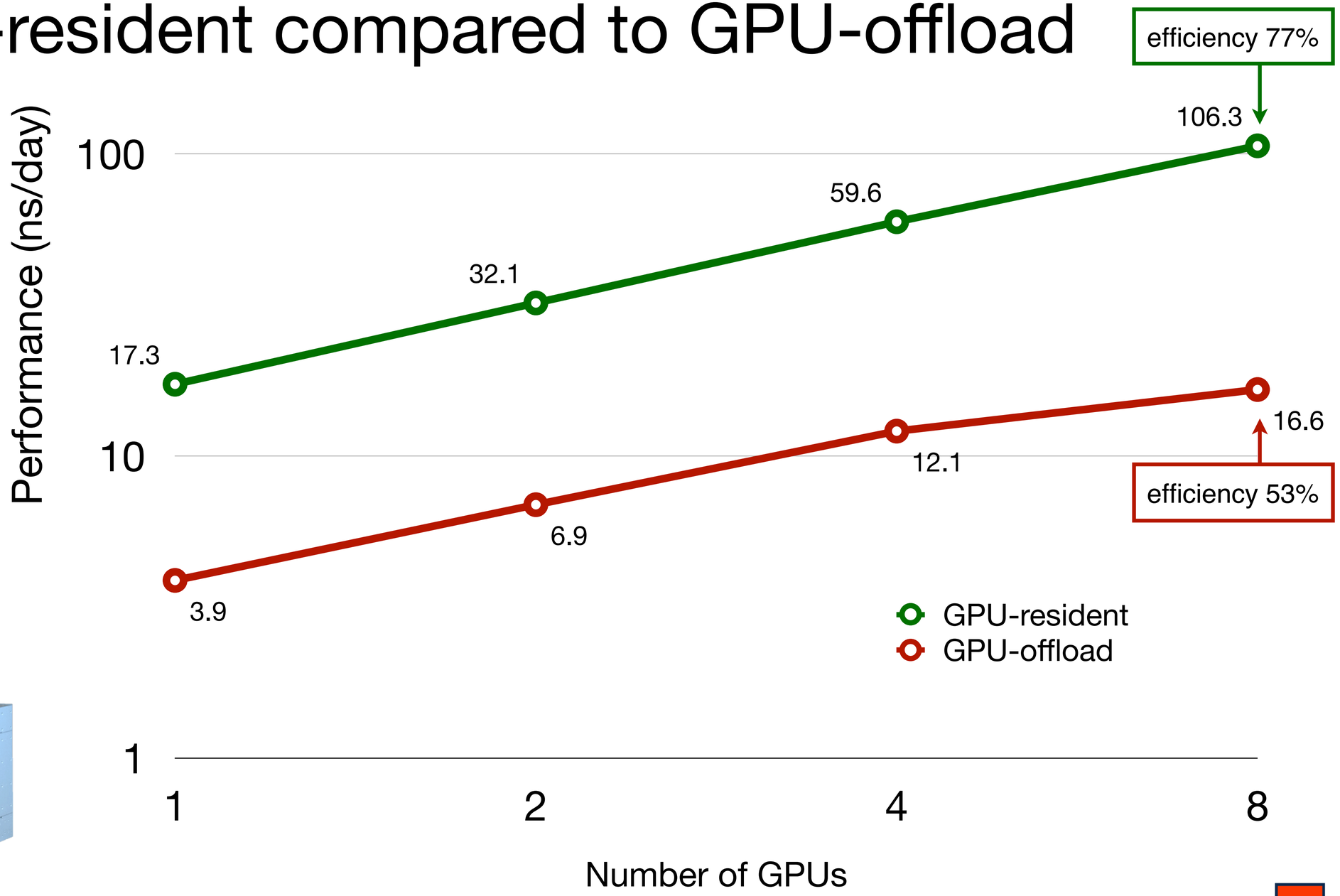
See past NVIDIA GTC talks for more details: s31529, s41378, s51693



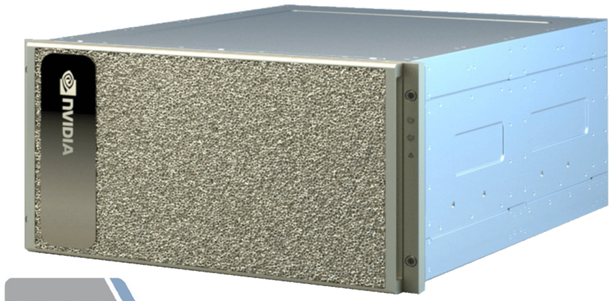
GPU-resident compared to GPU-offload



STMV
1.06M atoms



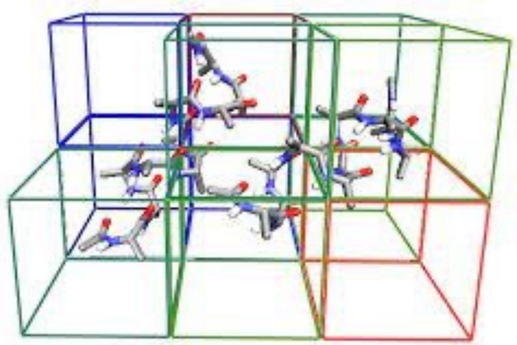
DGX-A100



NIH National Institutes of Health

Improving performance for smaller systems

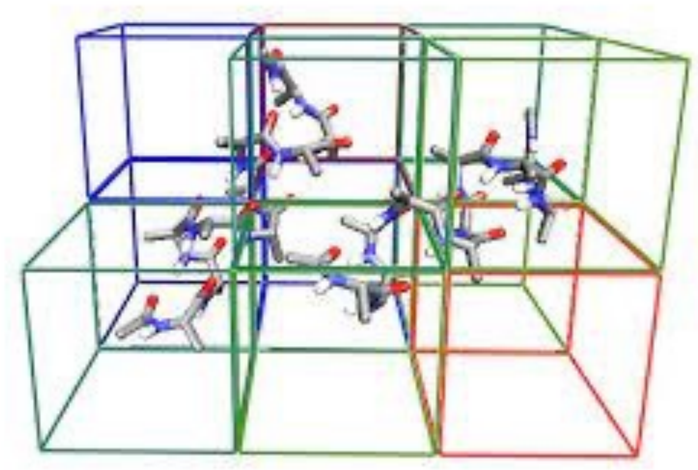
- Perform atom migration on GPU
  - Removes the biggest remaining CPU bottleneck



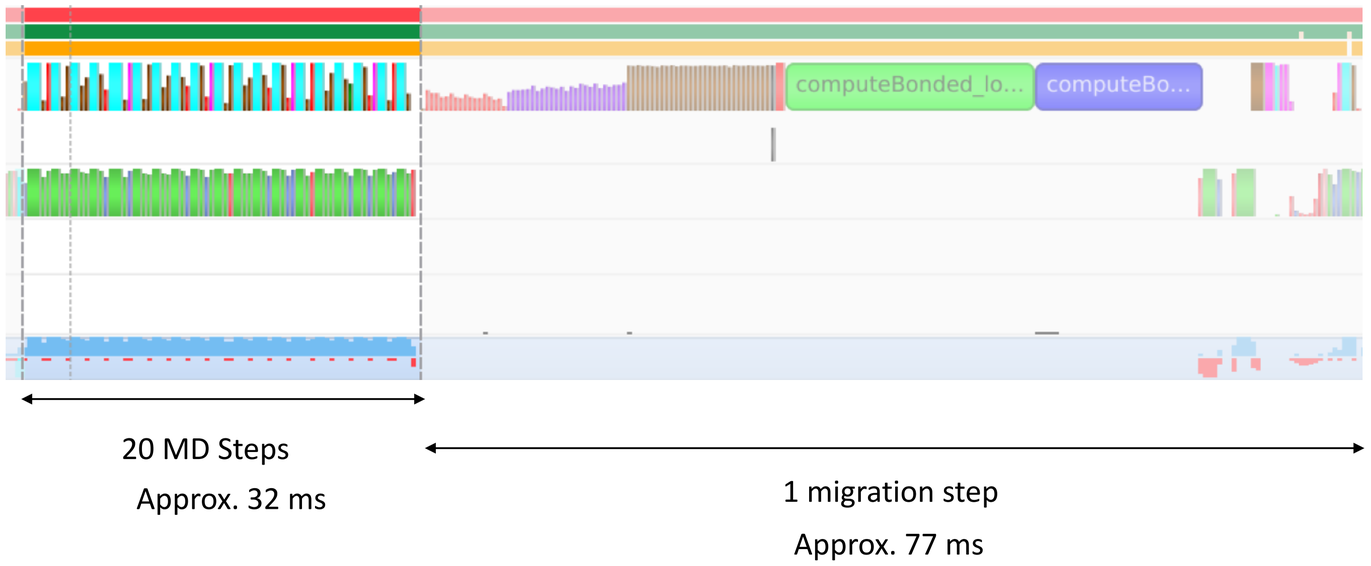
- Exploit "two-away" patch splitting option
  - Create more finer-grained work units
  - Provides more work to schedule across SMs (streaming multiprocessors) of each GPU
- Use MPS (Multi-Process Server) to co-schedule multiple jobs per GPU when running multiple simulations for ensemble sampling
  - Maximizes overall throughput by keeping GPUs fully occupied

Atom migration introduces overhead

- Earlier profiling showed the excessive cost of atom migration



ApoA1 (92k atoms) profiling on single GPU

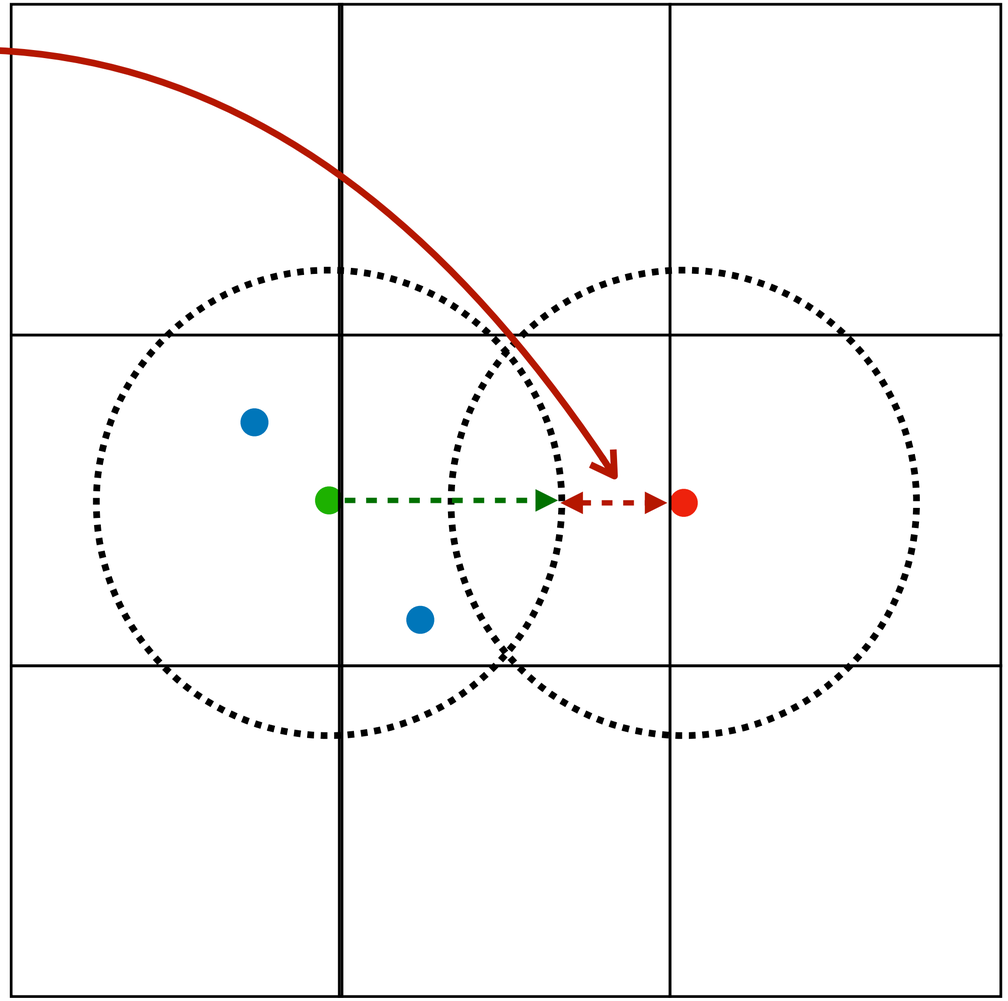


Time for atom migration is equal to 48 MD steps

NAMD's default is 20 steps per migration

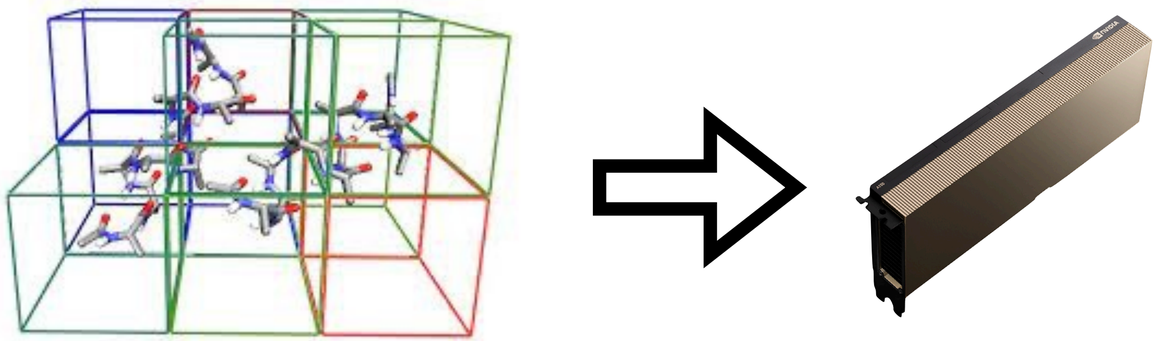
Mitigating cost of atom migration

- Extend default *patch margins* to permit more steps between migration
- Monitor atom movements to perform migration only when needed
- Utilize multiple CPU cores per device to decrease migration cost



Porting atom migration to GPU

- Requires extra data structures on GPU
  - Topology data to update bonded terms
  - Extra buffer space to receive atoms from other GPUs
  - Maintain copy of full atom data in AoS (array-of-structures) form
- Implementation in two main stages
  - Refactor the device buffers and data structures
  - Introduce kernels for performing migration



- Benefits all GPU-resident simulation, especially for smaller systems
  - Having less computational work available to smaller systems exposes a greater penalty from CPU migration

Single GPU performance improvements

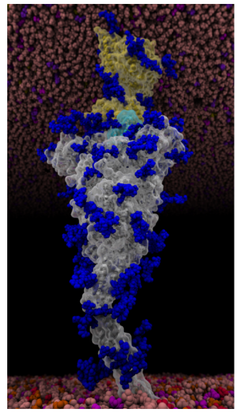
A100

Simulation details:
Spike ACE-2: NPT, 1 bar, 310 K, CHARMM force field, cutoff distance 12 Å, MTS with 2 fs time step and 4 fs PME, rigid bond constraints.
STMV, ApoA1: NVE, CHARMM force field, cutoff distance 12 Å, MTS with 2fs time step and 4 fs PME, rigid bond constraints.
Spike, STMV, ApoA1: Performance tuning parameter "margin" set to 8 Å for older versions, 4 for new version.
DHFR: NVE, CHARMM force field, cutoff distance 9 Å, HMR with 4 fs time step, PME, rigid bond constraints, "margin" 2 Å, two-away-Z.
https://www.ks.uiuc.edu/Research/namd/benchmarks/

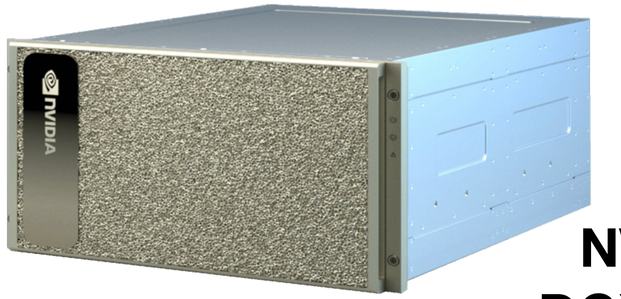
	Optimized version (Mar 2022) ns/day	GPU atom migration (Mar 2023) ns/day	% improvement
Spike ACE-2 (8.56M)	1.72	1.81	4.9%
STMV (1.06M)	15.87	17.20	8.4%
ApoA1 (92.2k)	182.0	190.7	4.8%
DHFR (23.6k)	903.1	1102.0	**22%**

NIH National Institutes of Health

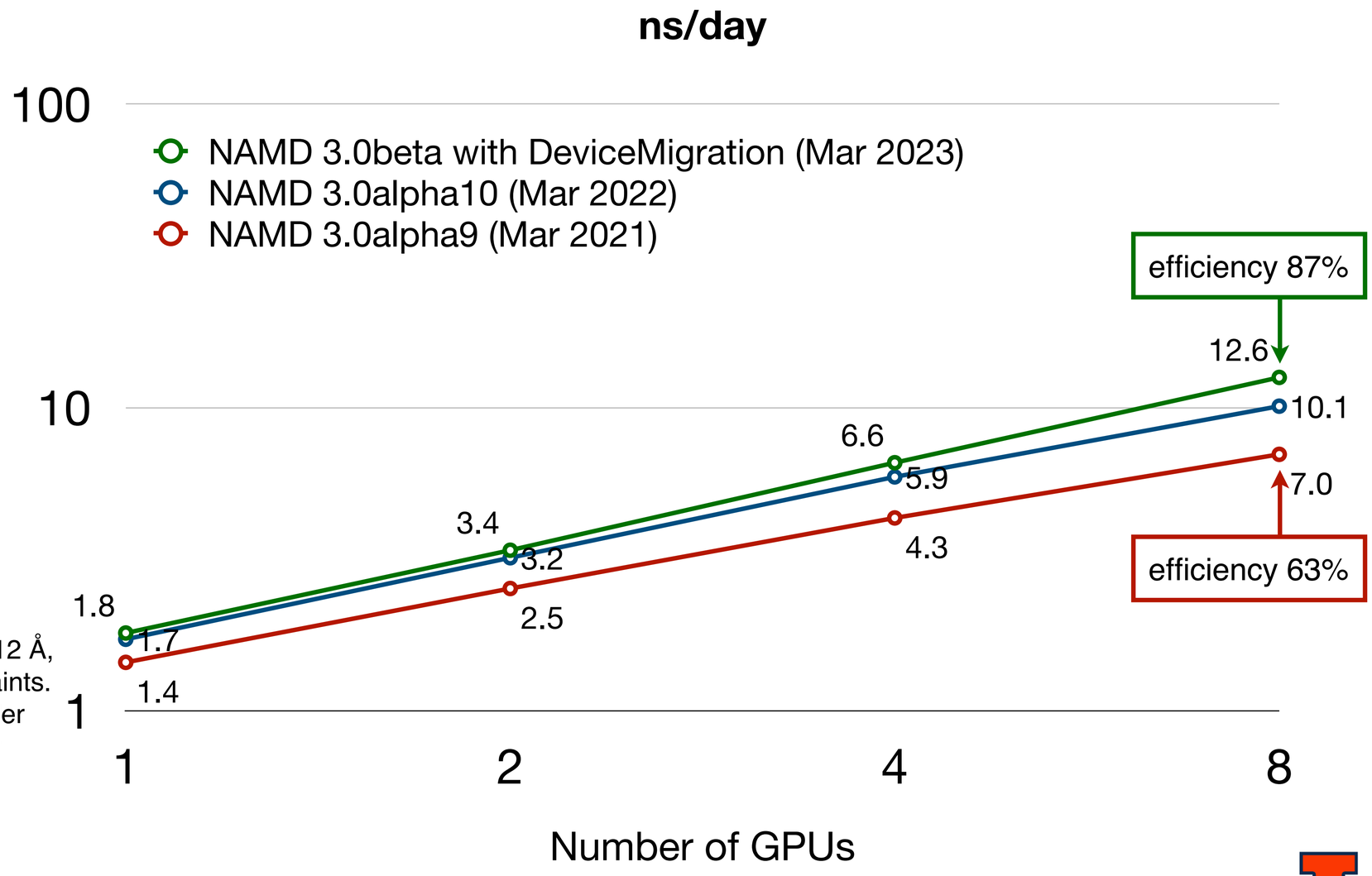
GPU-resident multi-GPU scaling of COVID-19 spike protein



Spike-ACE2
8.56M atoms



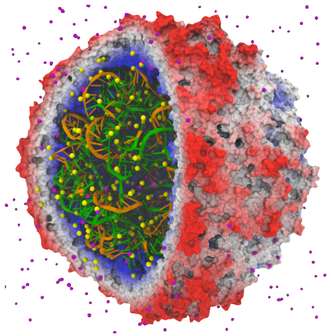
NVIDIA
DGX-A100



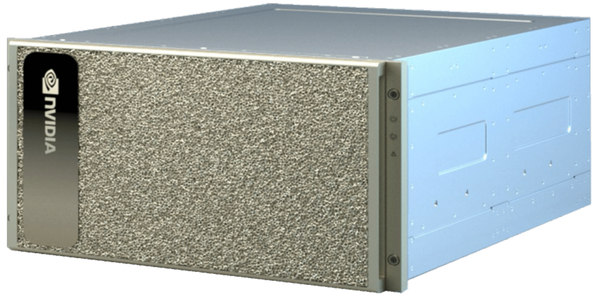
Simulation details:
NPT, 1 bar, 310 K, CHARMM force field, cutoff distance 12 Å, MTS with 2 fs time step and 4 fs PME, rigid bond constraints. Performance tuning parameter “margin” set to 8 Å for older versions, 4 for new version. PME PEs set to 8, 7, 5, 1 for numbers of GPUs 1, 2, 4, and 8, respectively, for all.

NIH National Institutes of Health

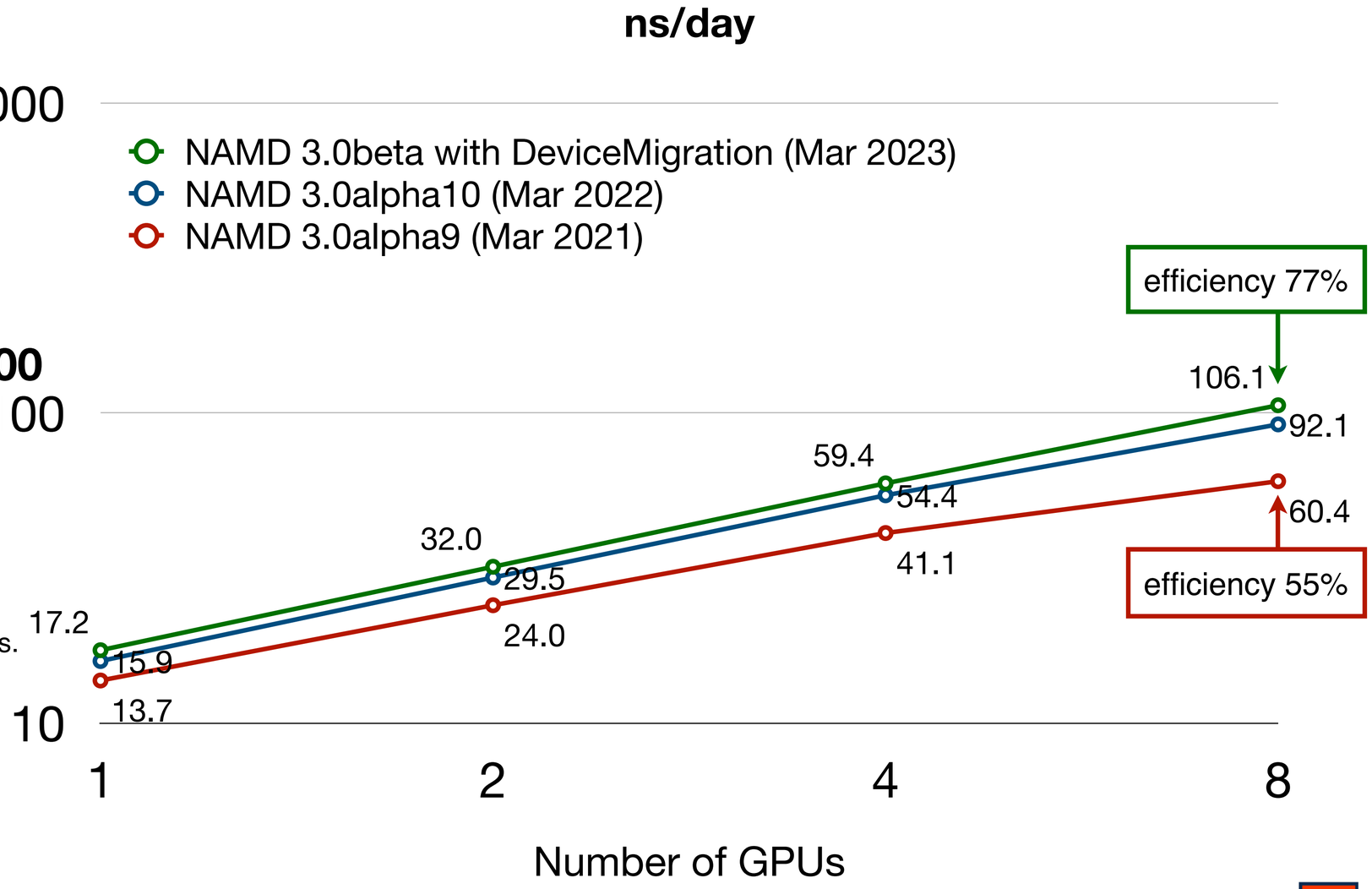
GPU-resident multi-GPU scaling of STMV



STMV
1.06M atoms



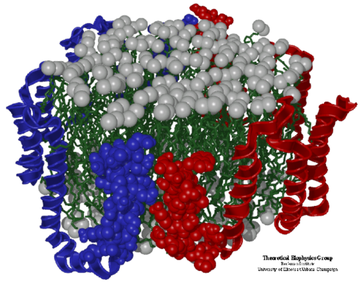
DGX-A100



Simulation details:
NVE, CHARMM force field, cutoff distance 12 Å, MTS with 2 fs time step and 4 fs PME, rigid bond constraints. Performance tuning parameter “margin” set to 8 Å for older versions, 4 Å for new version. PME PEs set to 8, 7, 5, 1 for numbers of GPUs 1, 2, 4, and 8, respectively, for all.

NIH National Institutes of Health

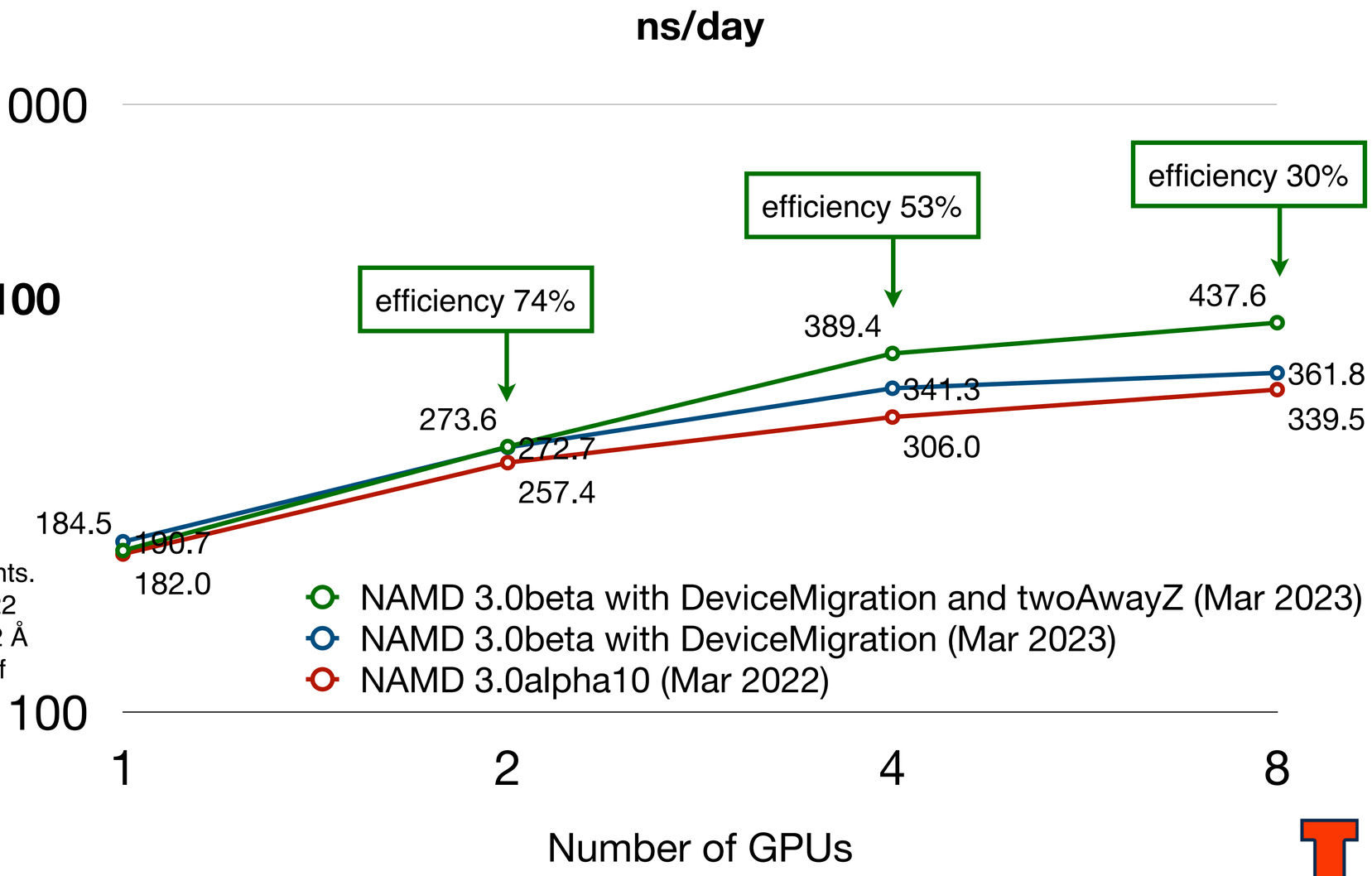
GPU-resident multi-GPU scaling of ApoA1



ApoA1
92.2k atoms

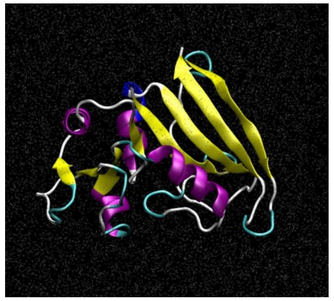
DGX-A100

Simulation details:
NVE, CHARMM force field, cutoff distance 12 Å, MTS with 2 fs time step and 4fs PME, rigid bond constraints. Performance tuning parameter “margin” set to 4 Å for 2022 version with 1 GPU and 0 Å for 2, 4, and 8 GPUs; set to 2 Å for 2023 version. PME PEs set to 8, 7, 5, 1 for numbers of GPUs 1, 2, 4, and 8, respectively for both versions.



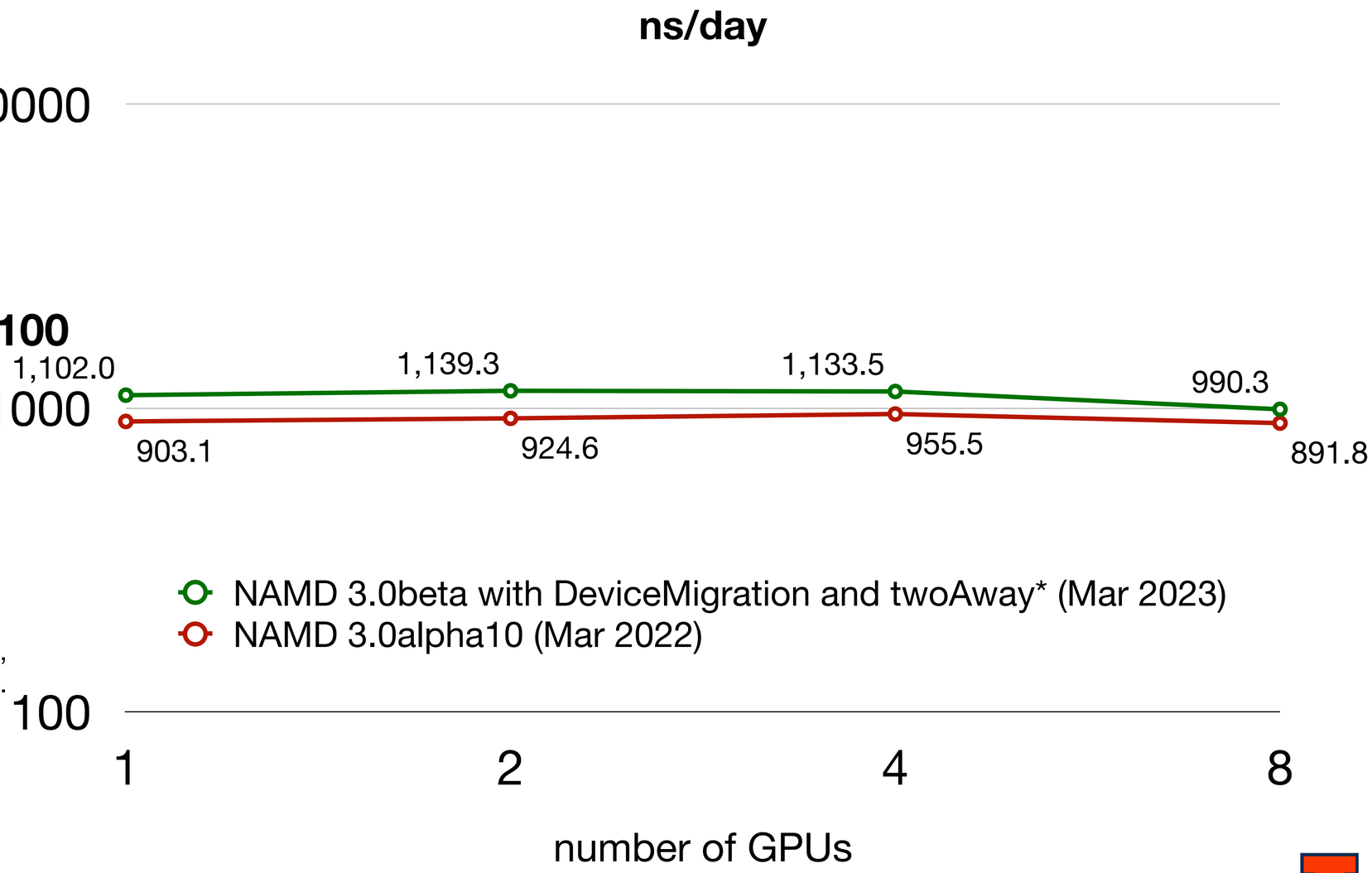
National Institutes of Health

GPU-resident multi-GPU scaling of DHFR



DHFR
23.6k atoms

DGX-A100



Simulation details:
NVE, CHARMM force field, cutoff distance 9 Å,
HMR with 4 fs time step, PME, rigid bond constraints.
Performance tuning parameter “margin” set to 2 Å.
*GPU atom migration uses two-away-Z for 1 and 2 GPUs,
two-away-YZ for 4 GPUs, and two-away-XYZ for 8 GPUs.
https://www.ks.uiuc.edu/Research/namd/benchmarks/

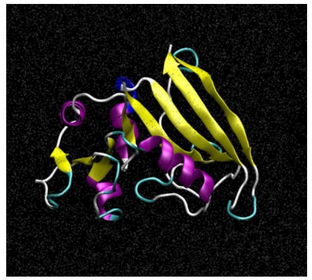
NIH National Institutes of Health

Improving throughput for ensemble sampling

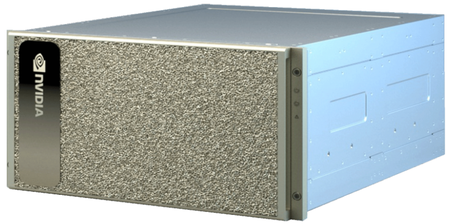
- Time sampling (single sequential simulation of phase space) offers limited parallelization — especially for smaller systems

- Ensemble sampling (many independent simulations of phase space) can provide better statistics with less overall computational cost
  - Good approach: running one simulation per GPU
  - Possibly better approach: **using MPS (Multi-Process Service) to run multiple simulations per GPU**
    - Keep SMs on all GPUs fully occupied
    - Although no single simulation finishes as fast as running one simulation per GPU, the aggregate ns/day achieved by all simulations is higher

DHFR ensemble sampling



DHFR
23.6K atoms



DGX-A100

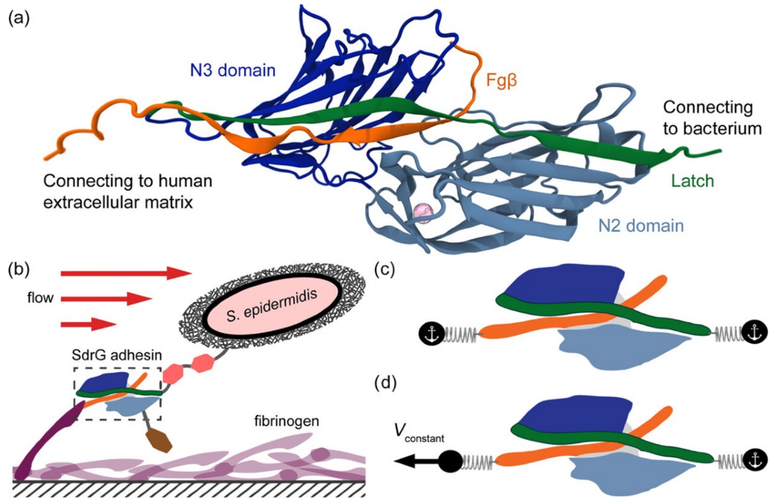
Simulation details:
Same as before for version 2023 on 1 GPU.
Each simulation is 500,000 steps total
NVE, CHARMM force field, cutoff distance 9 Å,
HMR with 4 fs time step, PME, rigid bond constraints.

Simulations per GPU	Total time steps	Total run time (sec)	Aggregate ns/day	Per GPU ns/day
1	4,000,000	197.7	6,993	874*
2	8,000,000	297.2	9,302	1,163
4	16,000,000	539.2	10,256	1,282
8	32,000,000	1,063.8	**10,396**	**1,299**

**Note: single simulation per GPU is lower than before due to including full execution time, including startup and ending file I/O*

SdrG ensemble sampling

Melo, Gomes, Bernardi. *J. Am. Chem. Soc.* 145, 1, 70-77 (2023)



- *Staphylococcus epidermidis* is major cause of infection in medical implants
- SdrG adhesin protein binds to human fibrinogen during infection
- Understand molecular origins of stabilizing forces underlying strong bindings

SdrG
240k atoms

DGX-A100

Simulation details:
NPT, 1 bar, 300 K, CHARMM force field, cutoff distance 11 Å, HMR, MTS with 4 fs time step and 8 fs PME, rigid bond constraints. Production runs in paper used SMD.

Simulations per GPU	Total jobs	Aggregate ns/day	Per GPU ns/day
1	8	734.23	91.78
2	**16**	**781.30**	**97.66**
3	24	754.04	94.26
4	32	764.00	95.50

NIH National Institutes of Health

Using MPS with NAMD

- Begin launch script with the following:

```
export CUDA_MPS_PIPE_DIRECTORY=/tmp/nvidia-mps
export CUDA_MPS_LOG_DIRECTORY=/tmp/nvidia-log
nvidia-cuda-mps-control -d
```

- Launch NAMD jobs in the background (using '&') and wait on the job PIDs

- Caveat: superuser access is required to run "nvidia-cuda-mps-control" but should work if computing center whitelists command or when using containers

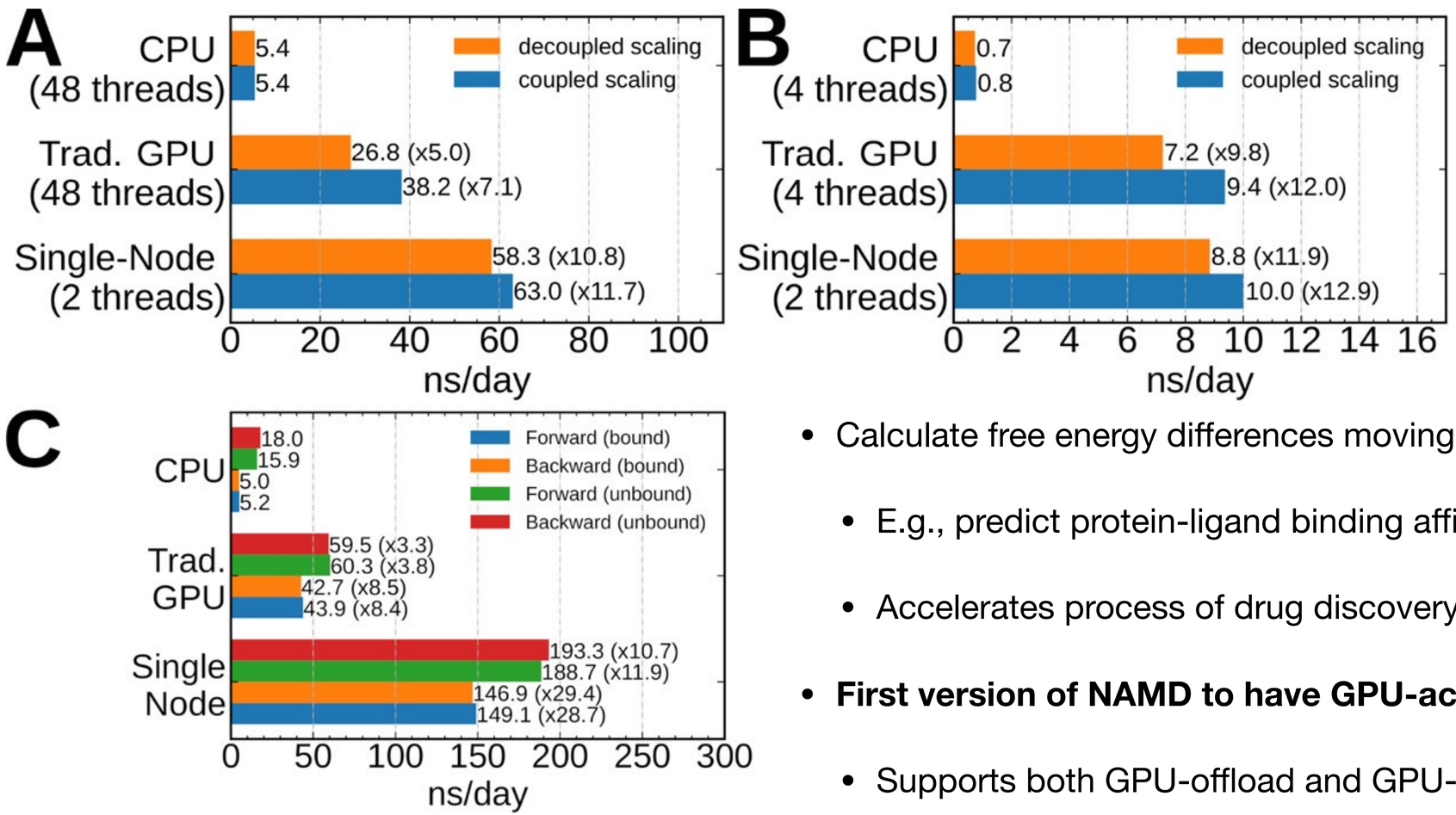
See: https://docs.nvidia.com/deploy/mps/index.html

Advanced features supported by GPU-resident

- Replica-exchange MD
- ✓ Alchemical free energy methods:
  FEP (free energy perturbation) and TI (thermodynamic integration)
- REST2 (replica-exchange solute scaling)
- Harmonic restraints
- External electric field
- SMD (steered MD)
- ✓ Monte Carlo barostat (faster than Langevin piston)
- ✓ Group position restraints (replaces Colvars common use case)

Alchemical free energy methods

Chen, et al. *J. Chem. Inf. Model.* 60, 5301-5307 (2020)



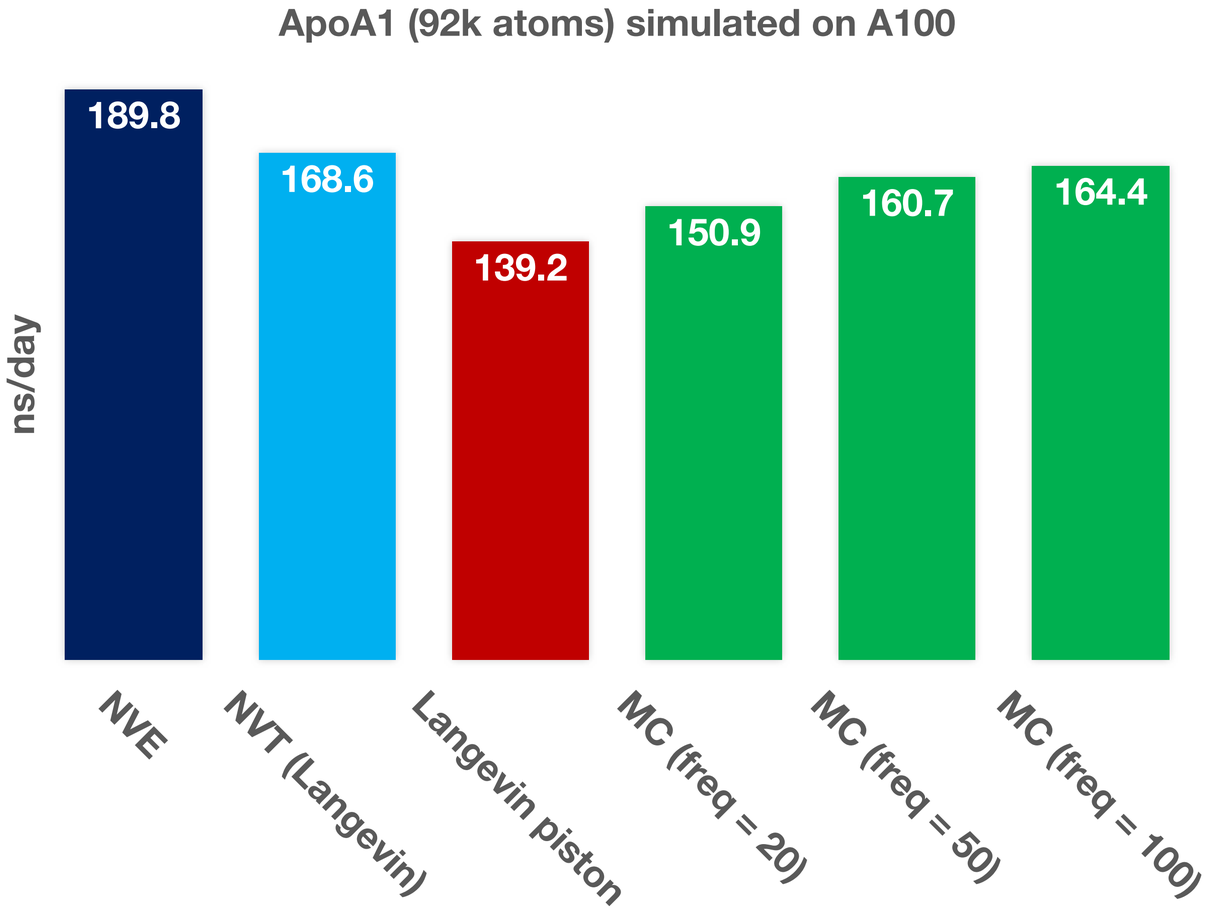
- Calculate free energy differences moving between two different chemical states
  - E.g., predict protein-ligand binding affinity, determine solvation free energies
  - Accelerates process of drug discovery
- **First version of NAMD to have GPU-accelerated FEP and TI**
  - Supports both GPU-offload and GPU-resident, up to 30x speedup over CPU-only
  - Compatible with multi-GPU scaling

National Institutes of Health

Monte Carlo barostat

- Rescales periodic cell at fixed step intervals, accept or reject based on MC acceptance of new energy
- Faster than Langevin piston due to avoiding virial calculations
- Rescaling is performed on geometric centers of molecules
- Only for GPU-resident, would require extra communication for multi-node

$$p_{\mathrm{acc}} = \min\left\{1, \left(\frac{V_{\mathrm{new}}}{V_{\mathrm{old}}}\right)^N e^{-\beta(U_{\mathrm{new}}-U_{\mathrm{old}})-\beta P(V_{\mathrm{new}}-V_{\mathrm{old}})}\right\}$$

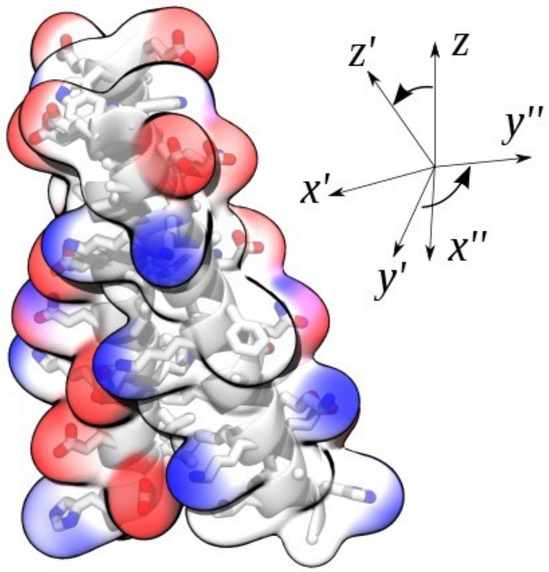


Faller, de Pablo. *J. Chem. Phys.* 116, 55 (2002)

Colvars Module for NAMD

Fiorin, Klein, Hénin. *Mol. Phys.* 111, 3345-3362 (2013) https://colvars.github.io/

- Modular framework for expressing and manipulating collective variables that reduce the large number of degrees of freedom for a molecular system down to its essentials
  - Monitor statistics in situ during simulation
  - Apply biasing forces for enhanced sampling
- Provides flexible and deep interface for specifying user-defined potentials assembled from atom position and force data
- **Is the most requested feature missing from GPU-resident mode**
- Externally developed library that is CPU-based, with deep object hierarchy, single-threaded (but with some OpenMP directives), making it difficult to port to GPU
- Initial testing to incorporate Colvars Module together with GPU-resident code path slows down performance to almost same as GPU-offload
- GPU port will require co-development with Colvars developers, streamlining atom data movement into Colvars and perhaps porting essential compute routines to GPU



Showing tilt and spin angle orientations of a molecule

Colvars host-device data transfer bottleneck

Multiple time stepping showing a non-PME step

Copy forces from device to host

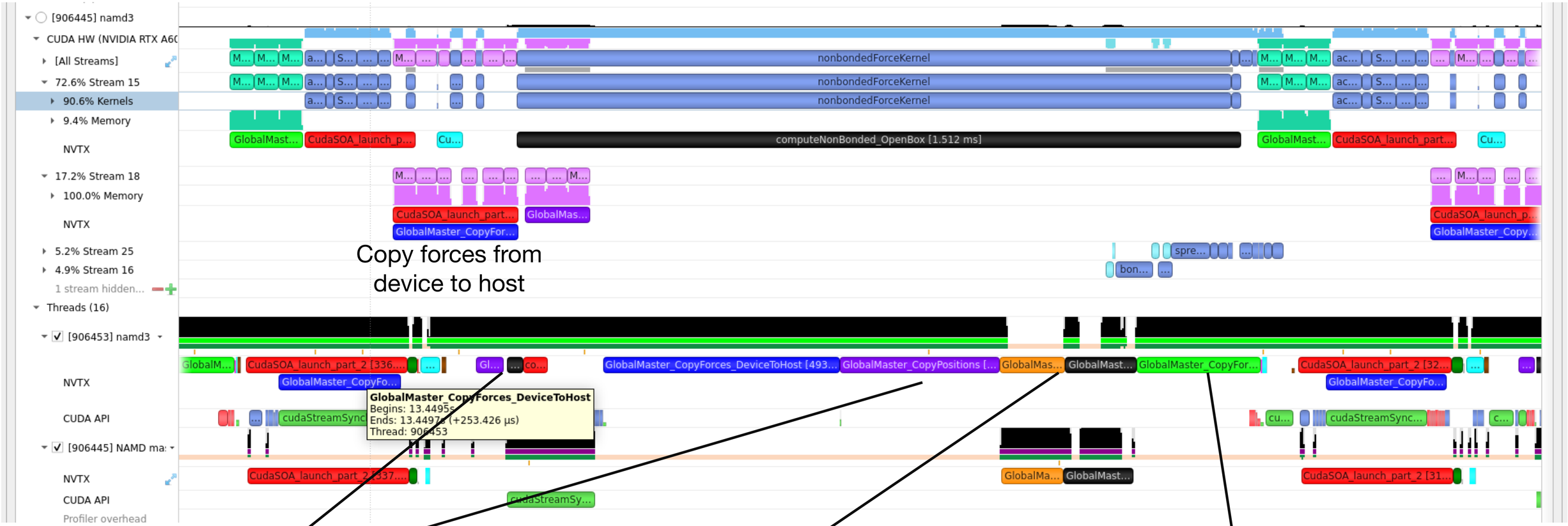
Copy positions from device to host

Initialize force and calculate GlobalMaster force

Copy new forces from host to device

Colvars overlapping host-device data transfer

Multiple time stepping showing a PME step



Copy forces from device to host

Copy positions from device to host

Initialize force and calculate GlobalMaster force

Copy new forces from host to device

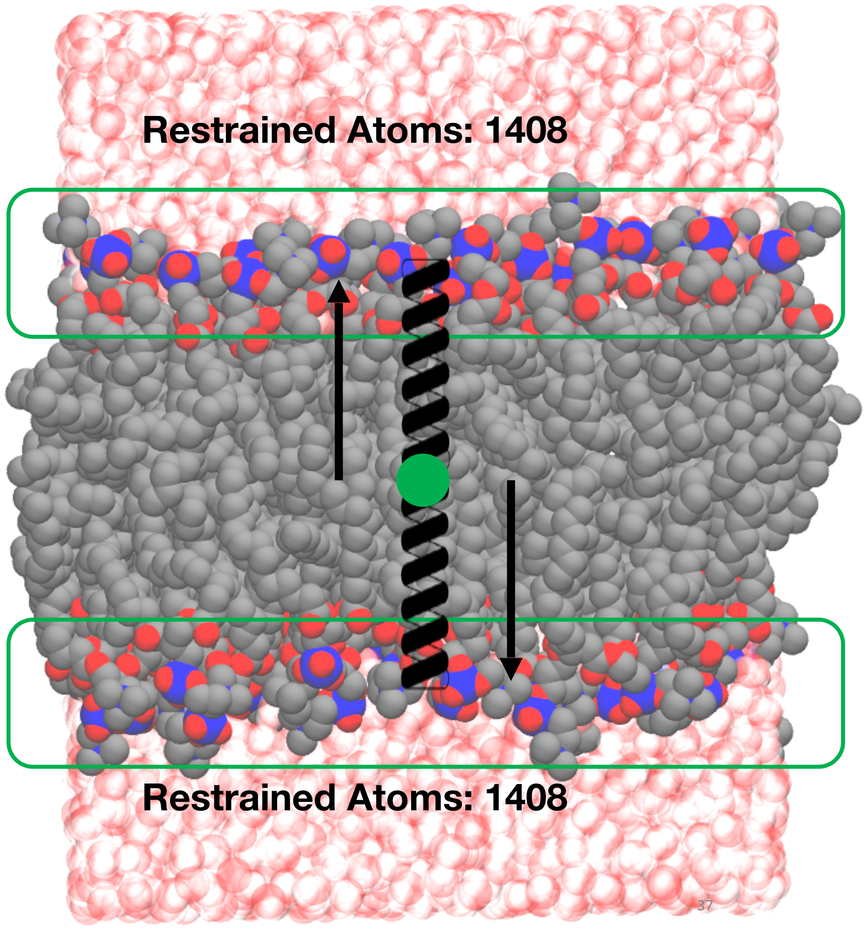
Initial results interfacing Colvars to GPU-resident

Benchmark: NVT POPC Equilibration

CPU: 32 cores AMD 3975WX 3.50GHz
GPU: 1 RTX A6000 GPU
Timestep: 2 fs
fullElectFreq: 2,4 fs

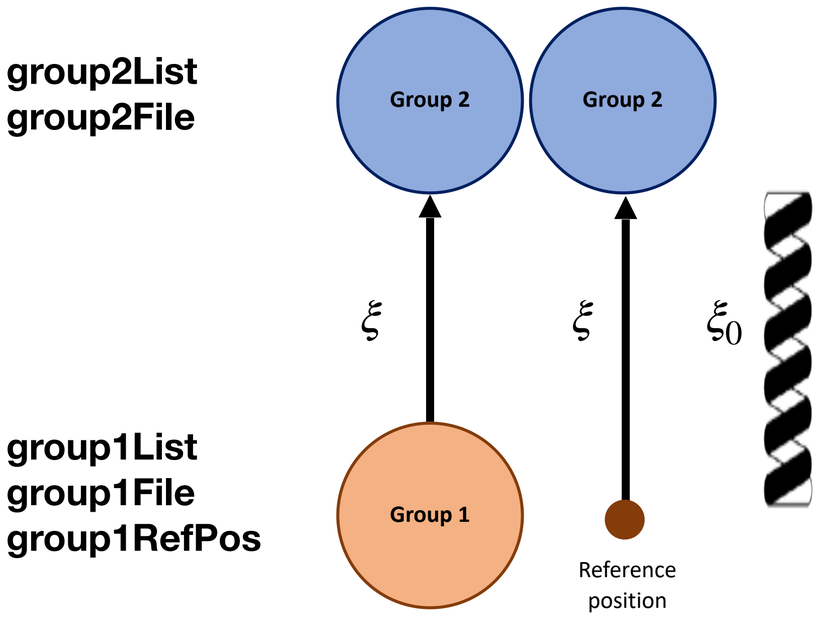
	Simulation (ns/day)		%Performance Lost
NAMD Version	**Without colvars**	**With colvars**	**(WO-W)/WO Restraints**
NAMD 3	77	58.3	**25%**
NAMD 3 MTS	89.7	58.9	**34%**

Total Atoms: 132k
Total Restrained Atoms: 2816



Group position restraints

- User-defined groups of atoms, with centers of mass connected by harmonic restraints
- Provides native support for a common collective variable use case with Colvars Module
- Only for GPU-resident, would require extra communication for multi-node



groupResUseMagnitude

$$E = k\left(\xi - \xi_0\right)^n$$

$$\xi_0 = \vec{r}^{\,\text{center}}_{\text{restraint}}$$

$$\xi = \vec{r}_2^{\,\text{COM}} - \vec{r}_1^{\,\text{COM}}$$

$$\xi_0 = \left|\vec{r}^{\,\text{center}}_{\text{restraint}}\right|$$

$$\xi = \left|\vec{r}_2^{\,\text{COM}} - \vec{r}_1^{\,\text{COM}}\right|$$

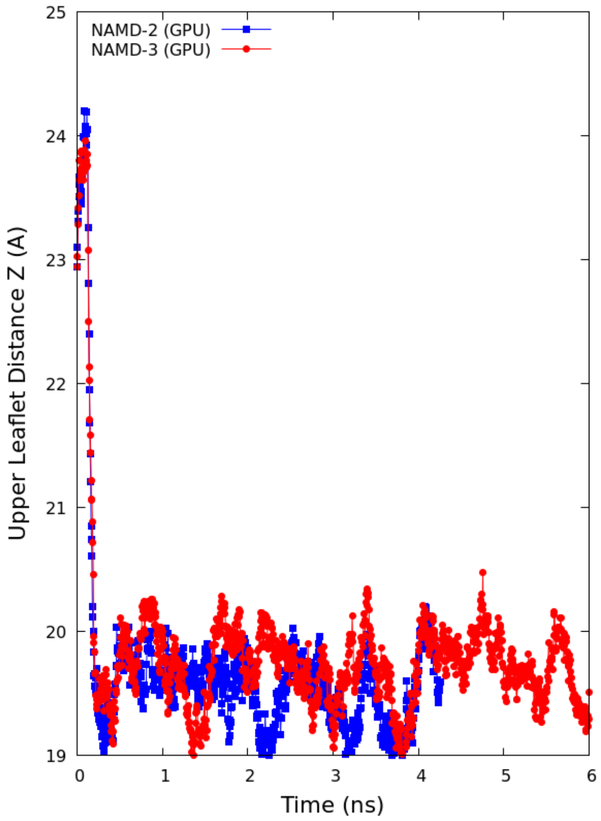
Group position restraints validation

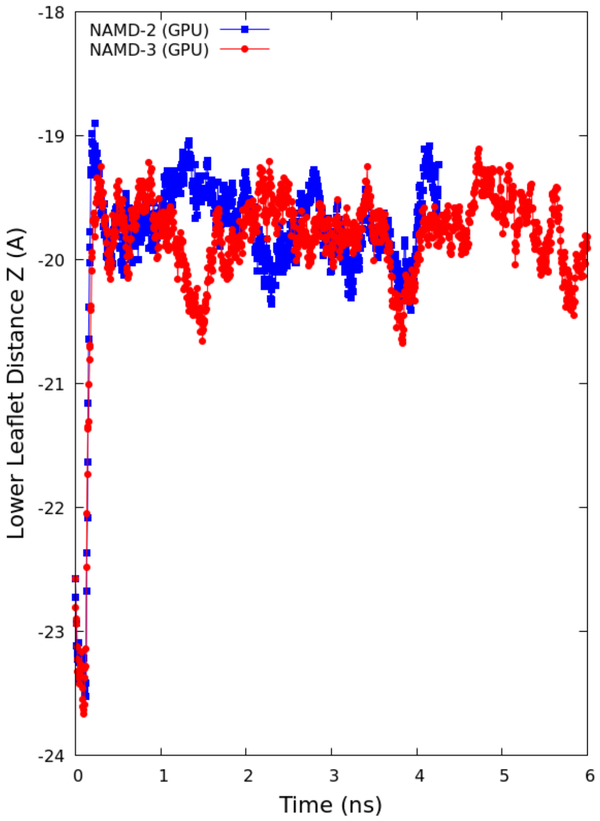
Benchmark: NVT POPC Equilibration

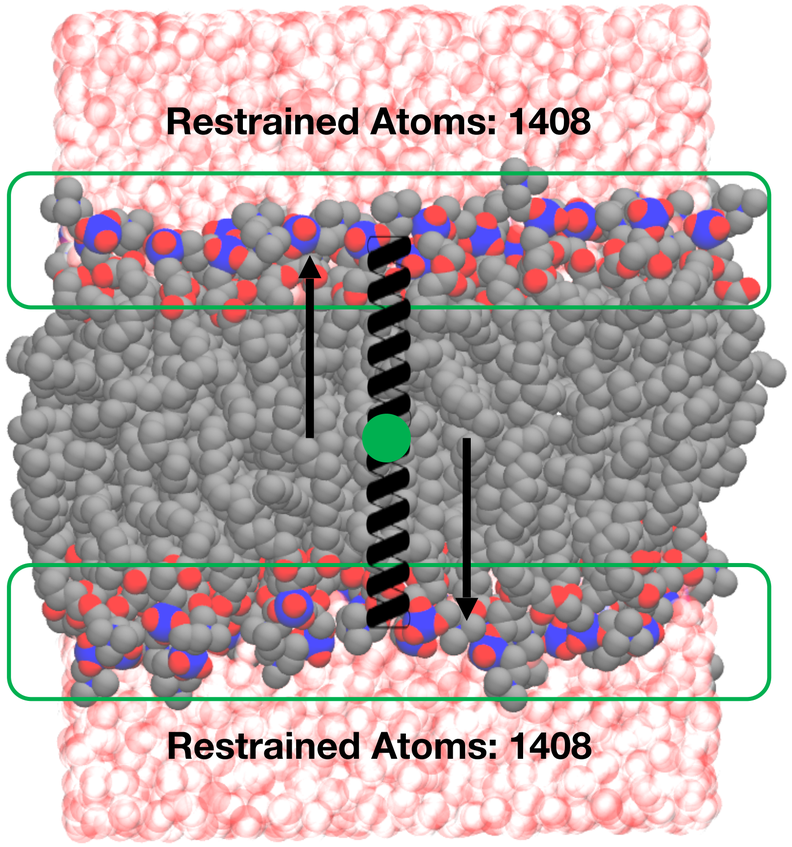
Restraint force: 2.5 (kcal/mol/Å)
Restraint distance: 19 Å

Timestep: 1 fs
fullElectFreq: 1 fs

Total Atoms: 132k
Total Restrained Atoms: 2816







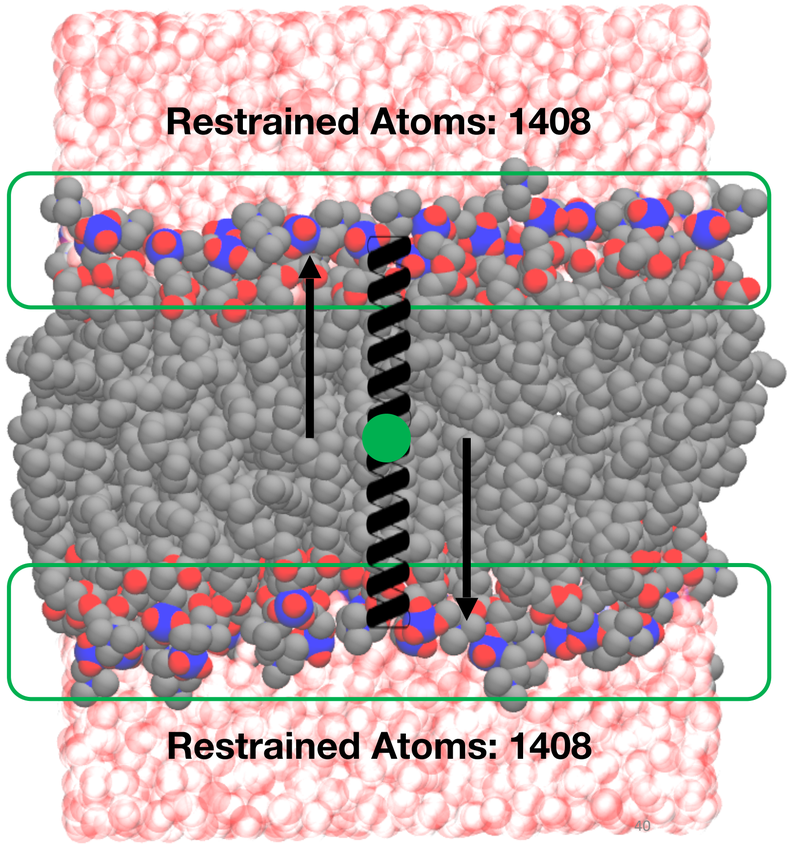
Group position restraints performance

Benchmark: NVT POPC Equilibration

CPU: 16 cores Intel Xeon E5-2650 v2 @ 2.60GHz
GPU: 1 TITAN V GPU
Timestep: 2 fs
fullElectFreq: 4 fs

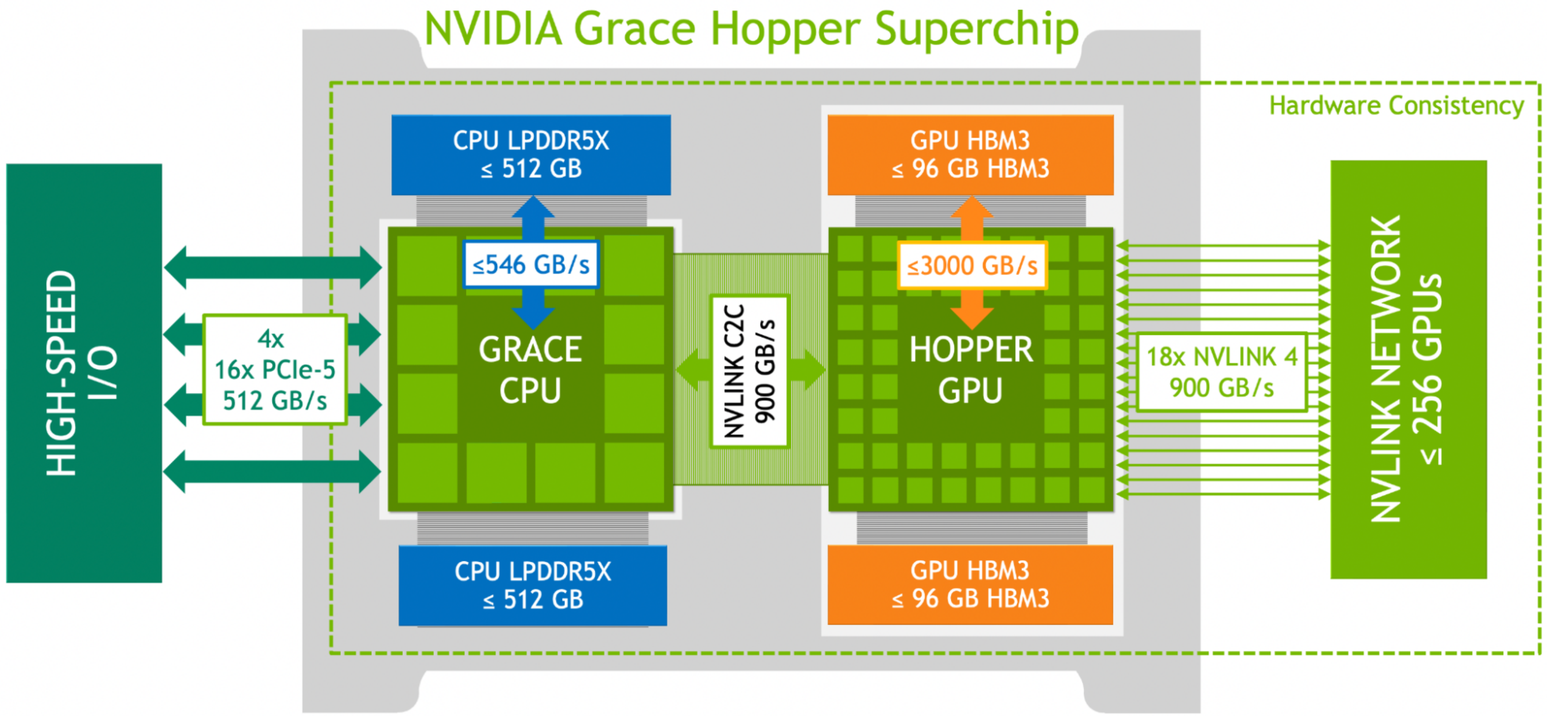
Total Atoms: 132k
Total Restrained Atoms: 2816

	Simulation (ns/day)		%Performance Lost
NAMD Version	**Without Restraints**	**With Restraints**	**(WO-W)/WO Restraints**
NAMD 2.14	25.2	23.5	6.75%
NAMD 3	69.0	67.6	2.03%
Speedup	**2.74**	**2.88**	



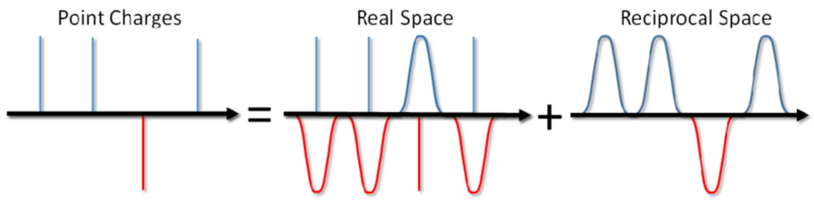
Leveraging Grace Hopper architecture

- Enables fast, low-latency communication between CPU and GPU via NVLink
- Provides memory coherency between host and device
- Has much higher CPU memory bandwidth per GPU than x86
- Expected to greatly reduce CPU-side bottlenecks, such as using Colvars with GPU-resident simulation



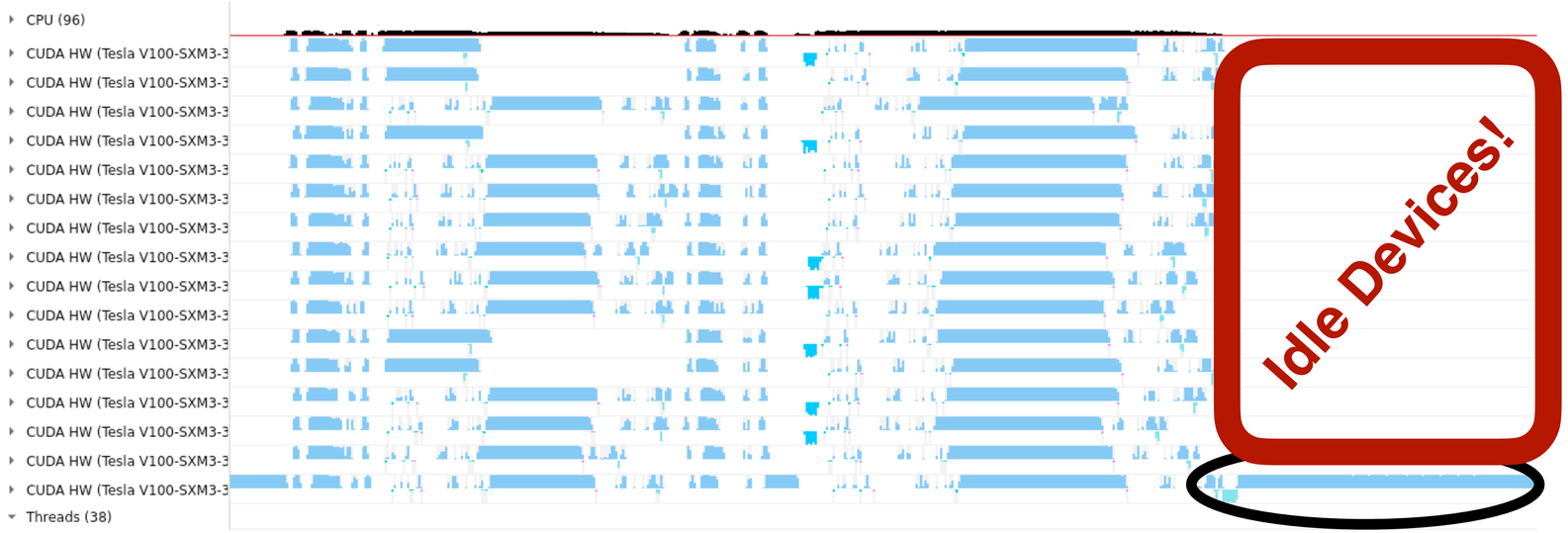
https://developer.nvidia.com/blog/nvidia-grace-hopper-superchip-architecture-in-depth/

Overcome Scaling Bottleneck From PME Long-Range Electrostatics



- PME (particle-mesh Ewald) requires calculating FFT
  - 3D FFTs for PME can be too small to parallelize effectively on GPUs
  - Too much latency is introduced with slab or pencil decomposition
- Assign PME to a single device
  - But over assignment can cause load imbalance

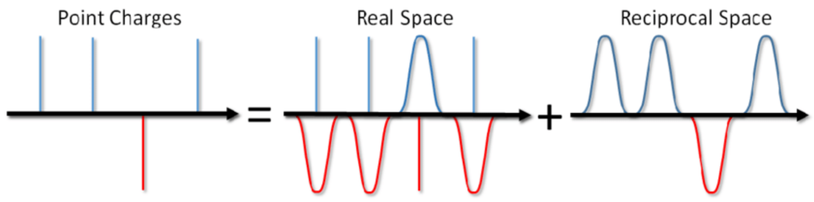
STMV NVE on DGX-2



PME Evaluation

Must make sure that PME device is not overloaded!

Overcome Scaling Bottleneck From PME Long-Range Electrostatics



- Exploit task-based parallelism
  - Use one device for PME
  - Reduce other force calculation on that device by **restricting number of CPU cores** assigned to it
    - Utilize NAMD's existing patch and compute object distribution across CPU cores
  - Much better to underload one GPU than to overload one GPU!

Nsight Systems profile shows algorithmic phases:

- Integration
- Non-PME force step
- Integration
- PME force step

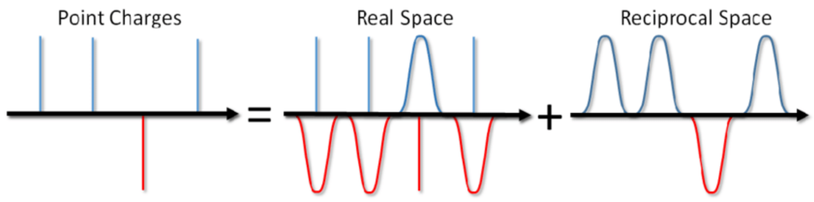
First GPU is PME device; fully utilized on PME step without creating bottleneck for the other GPUs

8.56M atom Spike-ACE2 on DGX-A100

- 1 CPU core for PME device
- 8 CPU cores for other 7 devices

Expand second GPU to understand compute details; remaining GPUs are similar

Overcome Scaling Bottleneck From PME Long-Range Electrostatics



- Optimal number of CPU cores assigned to PME device depends on how many GPUs we are scaling across
  - Need to determine for good load balancing
- Alternative approach under investigation
  - Parallelize the scalable parts of PME (*charge spreading* and *force gathering*) across all GPUs
  - Use task-based parallelism for just the FFTs — much smaller serial bottleneck than entire PME
  - Problem: overall bandwidth requirements **double** for sending grid points versus sending atoms

Determine optimal number of CPU cores for PME device for each number of GPUs

STMV NVE on DGX-A100 all optimizations enabled

PME cores	PME device work	GPUs 1	2	4	8
1	12.5%		18.4501	48.5648	92.0631
2	25.0%		20.4685	50.3603	91.2143
3	37.5%		22.4732	52.7799	87.9946
4	50.0%		24.4936	54.2095	86.5377
5	62.5%		26.6304	54.3703	82.6978
6	75.0%		28.6954	53.4331	79.9226
7	87.5%		29.5140	51.9185	77.8757
8	100.0%	15.8655	29.4101	50.6227	76.6106

National Institutes of Health

Obtaining and Running GPU-Resident NAMD

- Website: https://www.ks.uiuc.edu/Research/namd/
- On the "Software Download" page, choose "Version 3.0b3" (or later)
- GPU-resident mode is support by **multicore-CUDA** and **netlrts-smp-CUDA** builds
- Source code is available as tar ball, access through GitLab repository available by request: https://www.ks.uiuc.edu/Research/namd/development.html
- Config file parameter to enable GPU-resident mode: **`CUDASOAintegrate on`**
- Run NAMD from a terminal command line; restrict PME cores as follows (DGX-A100, 8-GPU):

```
./namd3 +p57 +pmepes 1 +setcpuaffinity +devices 0,1,2,3,4,5,6,7 myconf.namd
```

"+p" needs to be **total number of PEs** (CPU-threads) set to: 7*8 + {#pmepes}

"myconf.namd" refers to the NAMD config file

Acknowledgments

- Haochuan Chen (UIUC), Antti-Pekka Hynninen (NVIDIA, ORNL), Julio Maia (AMD, UIUC), Jim Phillips (UIUC), Mohammad Soroush Barhaghi (Schrödinger, UIUC), John Stone (NVIDIA, UIUC), Peng Wang (NVIDIA)
- NIH grants P41-GM104601 and R24-GM145965

NIH Center for Macromolecular Modeling and Bioinformatics (2019)
Beckman Institute, University of Illinois at Urbana-Champaign

National Institutes of Health