Addressing the Challenges of Heterogeneous Computing in NAMD

David J. Hardy
Senior Research Programmer

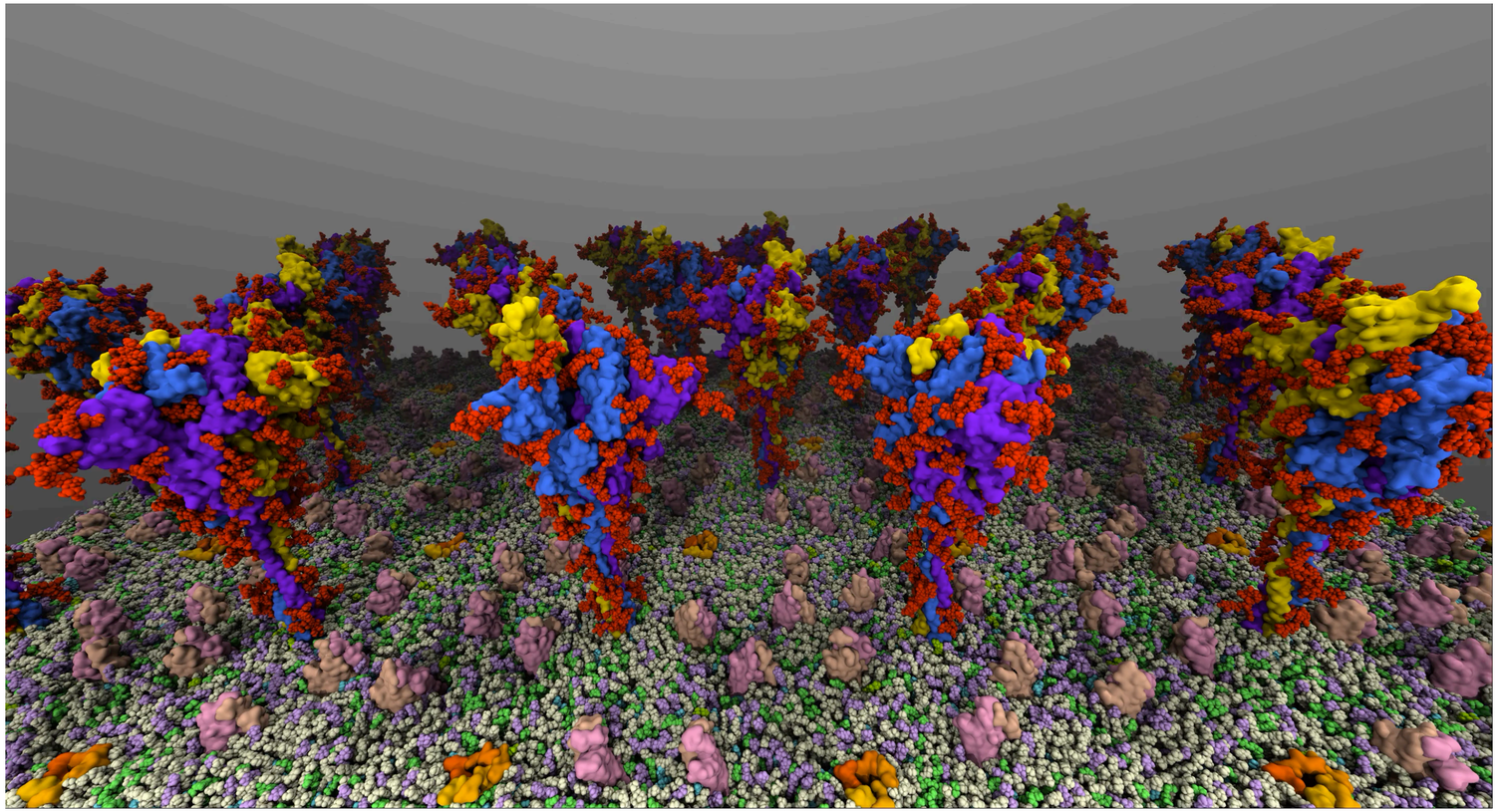
Theoretical and Computational Biophysics Group
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign

Charm++ Workshop
October 18-19, 2021

NAMD: Nanoscale Molecular Dynamics

https://www.ks.uiuc.edu/Research/namd/
Phillips, et al. *J. Chem. Phys.* 153, 044130 (2020)

- Parallel molecular dynamics application written in C++ with Charm++ objects
- Runs on all major operating systems, on laptops up through supercomputers
- Specializes in parallel scaling of large biomolecular simulations
- Many advanced features:
  - Enhanced sampling methods
  - Alchemical free energy methods
  - Collective variables module (Colvars)
  - TCL and Python scripting
- Over 25,000 registered users
- Over 15,000 citations of our NAMD reference papers (Google Scholar)



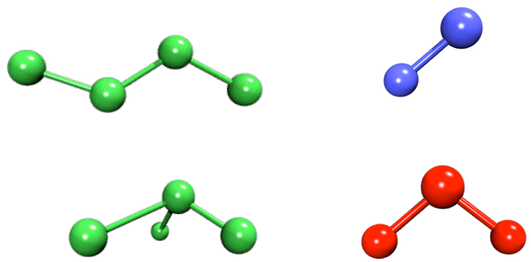
Investigations of coronavirus (SARS-CoV-2) spike dynamics.
Credit: Tianle Chen, Karan Kapoor, Emad Tajkhorshid (UIUC).
Simulations with NAMD, movie created with VMD.

NIH National Institutes of Health

Molecular Dynamics Simulation

- Most fundamentally, 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 up to billions of time steps



$$U(\vec{R}) = \underbrace{\sum_{bonds} k_i^{bond}(r_i - r_0)^2}_{U_{bond}} + \underbrace{\sum_{angles} k_i^{angle}(\theta_i - \theta_0)^2}_{U_{angle}} +$$

$$\underbrace{\sum_{dihedrals} k_i^{dihe}[1 + \cos(n_i\phi_i + \delta_i)]}_{U_{dihedral}} +$$

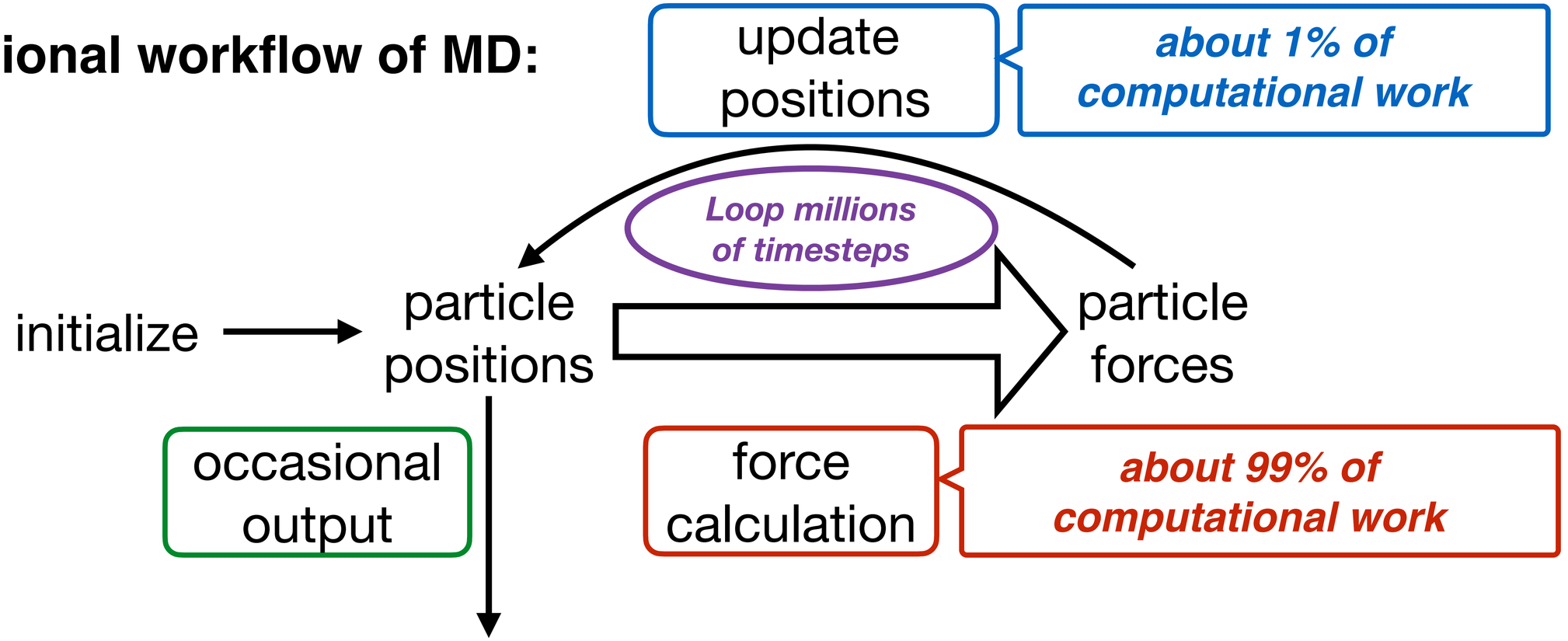
most of the computational work ⟶

$$\underbrace{\sum_i \sum_{j \neq i} 4\epsilon_{ij} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \sum_i \sum_{j \neq i} \frac{q_i q_j}{\epsilon r_{ij}}}_{U_{nonbond}}$$

(Lennard-Jones) (electrostatics)

Parallelism for MD Simulation Limited to Each Timestep

Computational workflow of MD:



reduced quantities (energy, temperature, pressure)

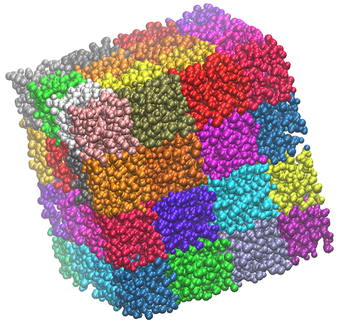
position coordinates (trajectory snapshot)

Hybrid Decomposition of Data *and* Work

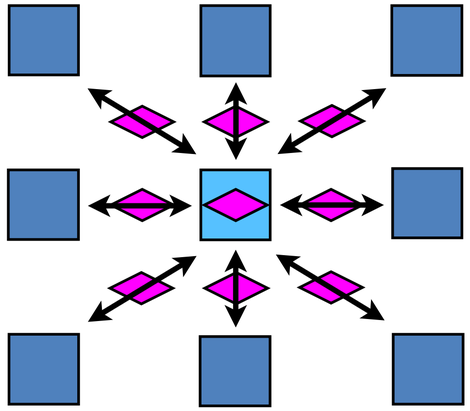
Kale et al., *J. Comp. Phys.* 151:283-312, 1999

- Atoms are decomposed into fixed volume ***patches*** within the system
- Forces that move atoms are calculated in parallel at each step between adjacent patches
- Work decomposition into ***compute objects*** creates much greater amount of parallelization, facilitates measurement-based load balancing with Charm++
- Migrate atoms to adjacent patches, updating domain decomposition after every ***cycle*** (e.g. 20 steps)

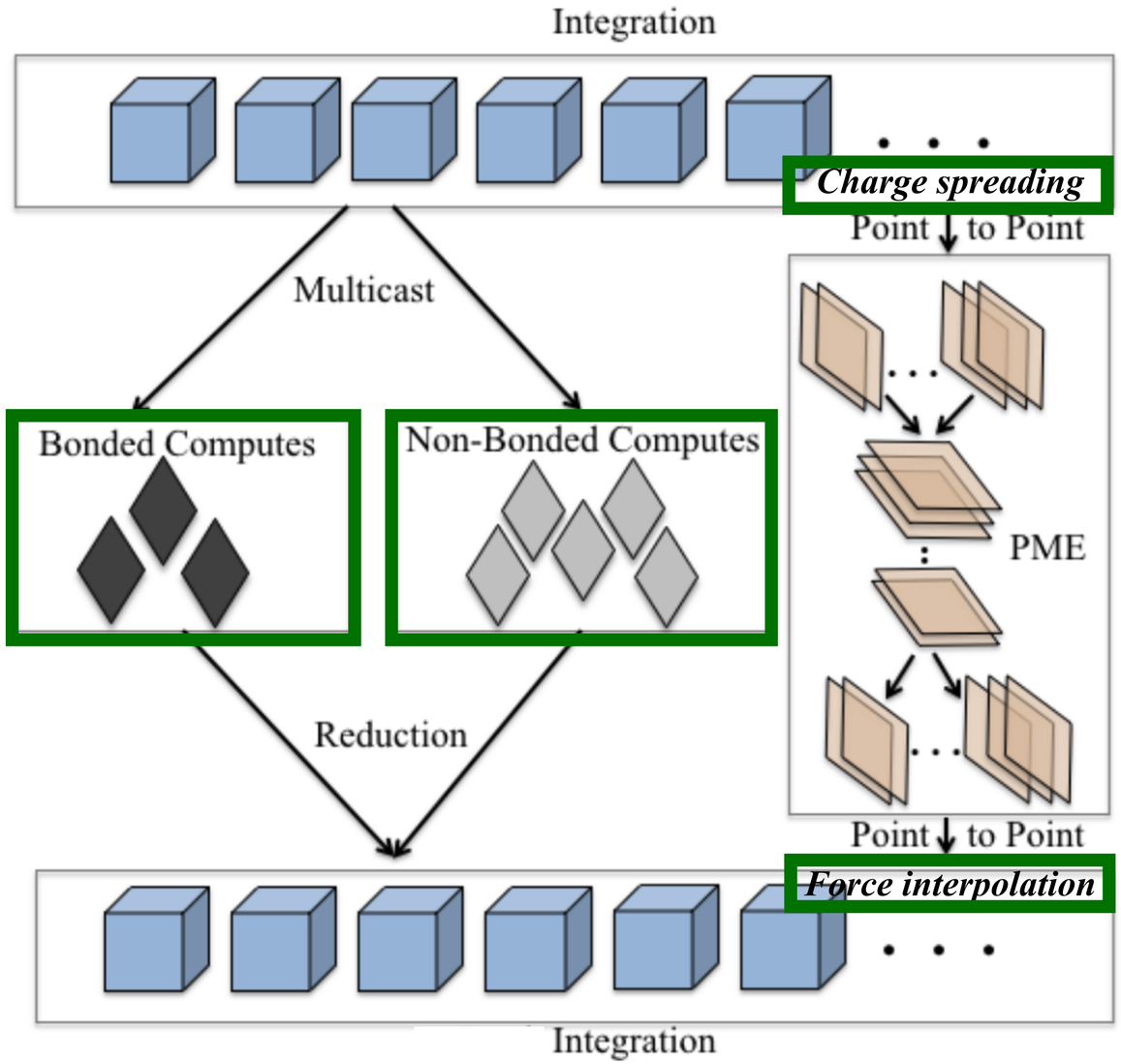
Spatial decomposition of atoms into patches



Work decomposition of patch interactions

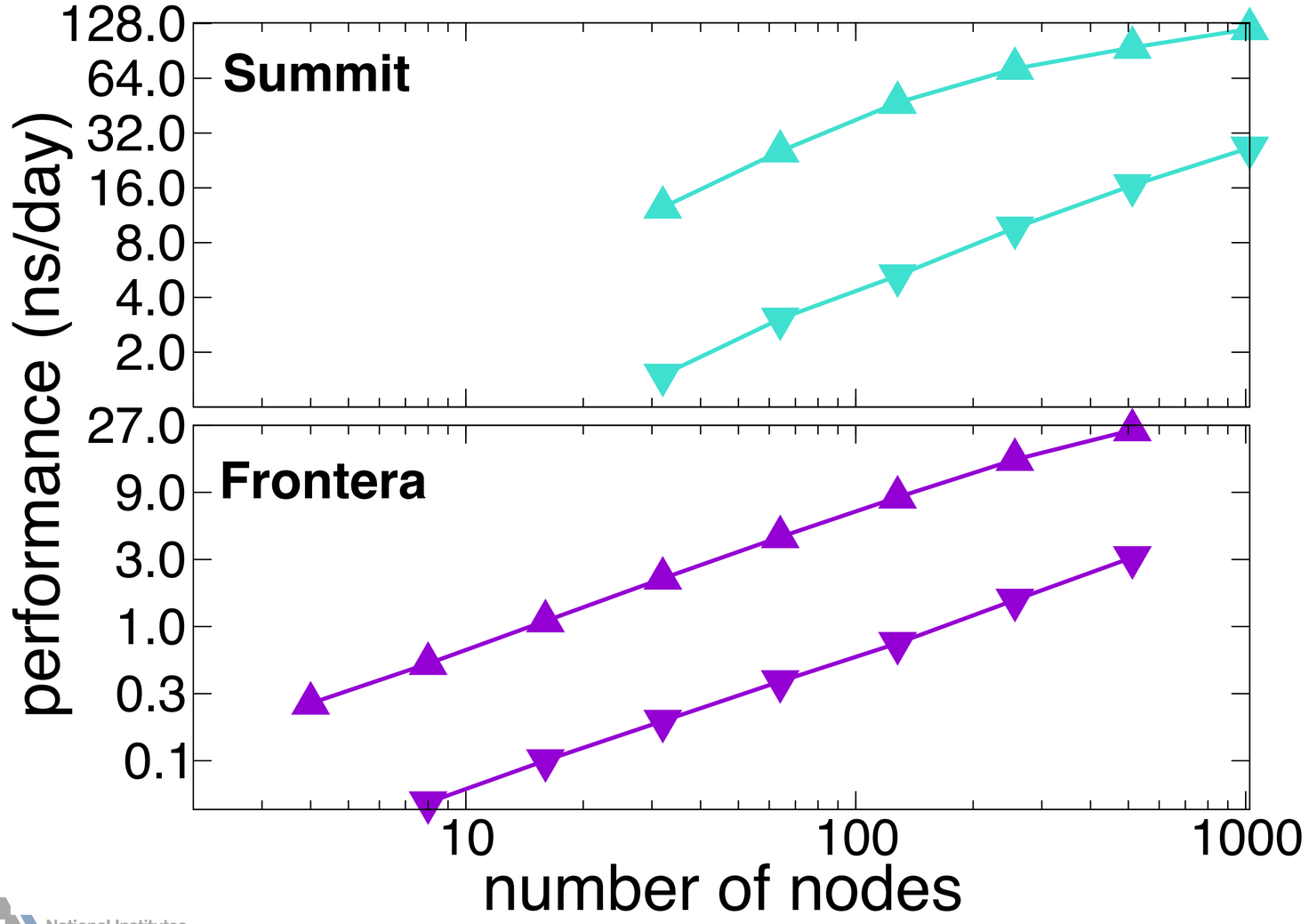


NAMD Decomposes Force Terms into Fine-Grained Objects for Scalability

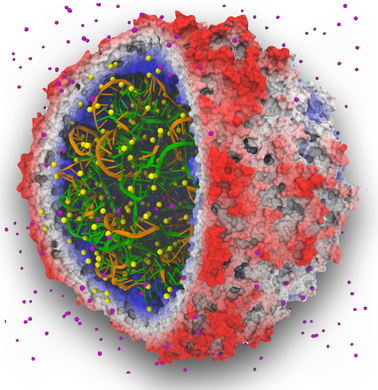


Offload forces to GPU

NAMD Scaling on CPUs and GPUs



Replications of the Satellite Tobacco Mosaic Virus (STMV)



▲ = 5x2x2 grid = 21M Atoms

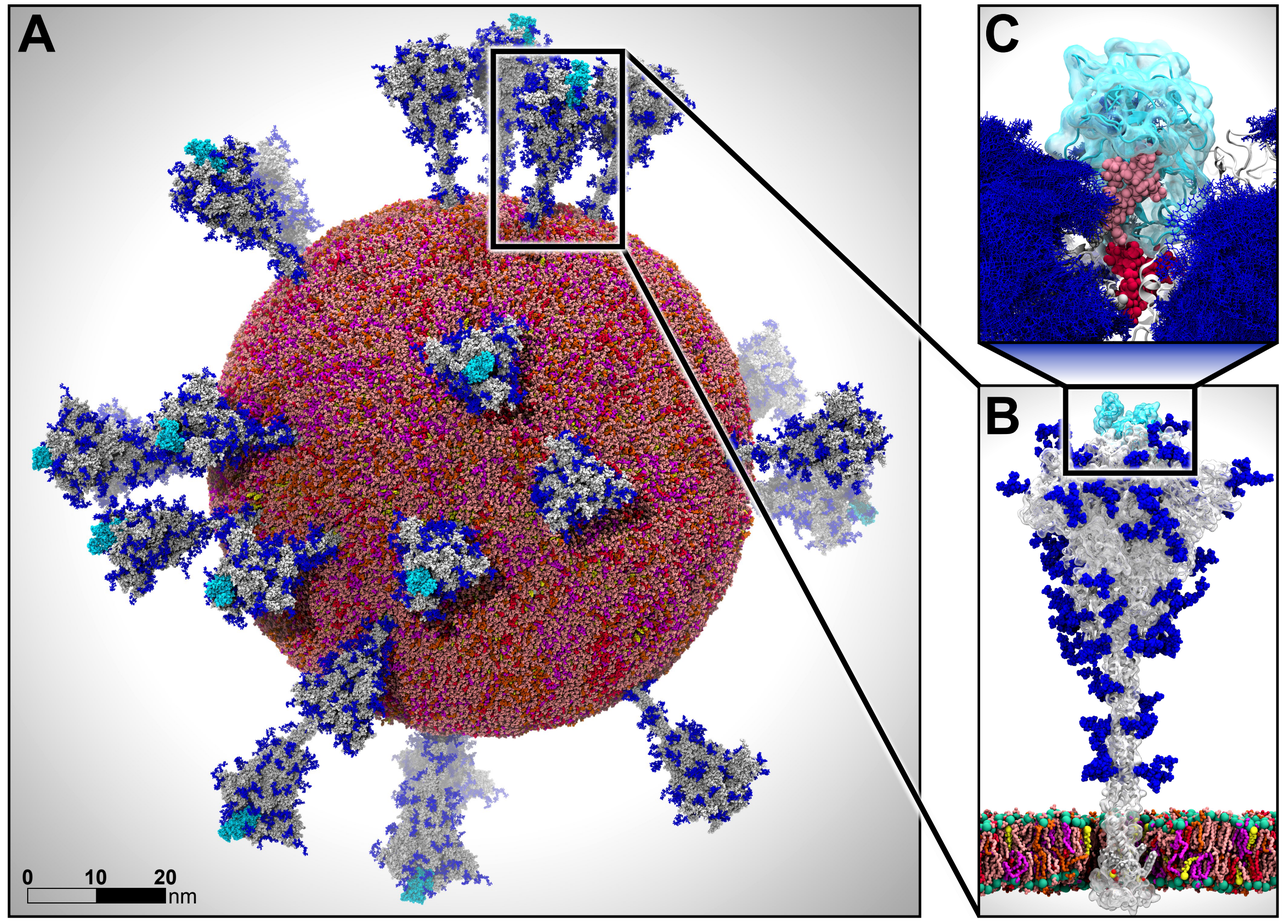
▼ = 7x6x5 grid = 224M Atoms

NIH National Institutes of Health

NAMD Simulating SARS-CoV-2 on Frontera and Summit

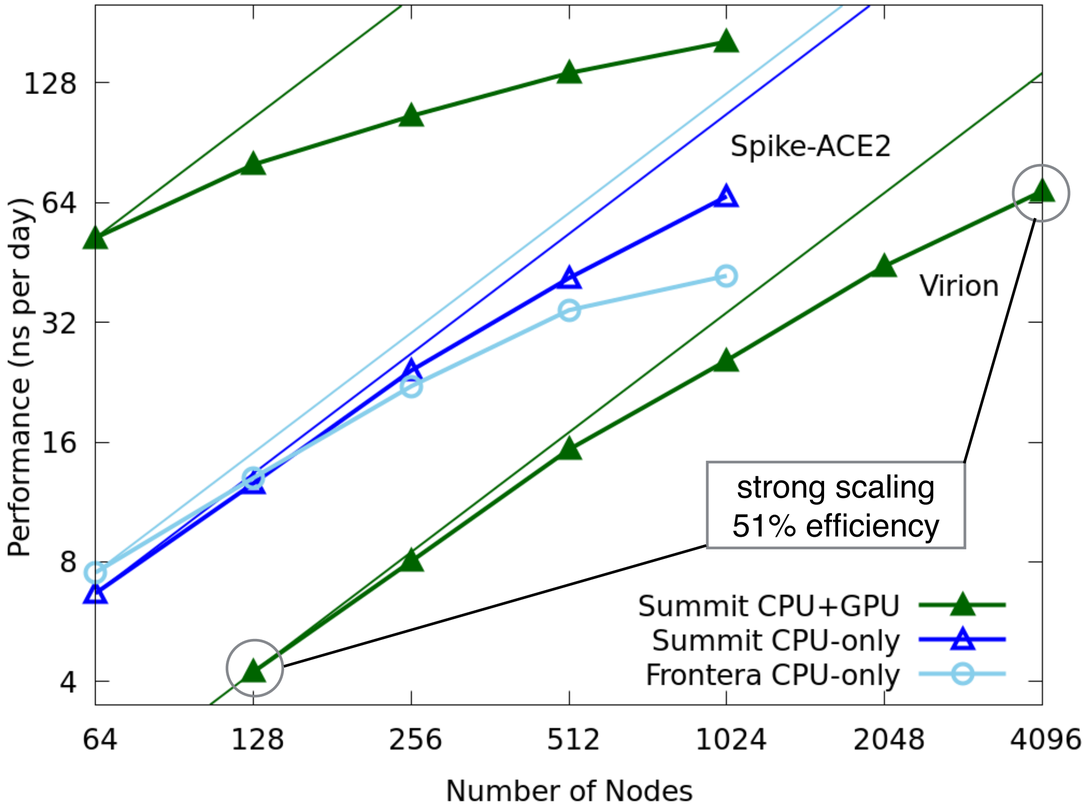
Collaboration with Amaro Lab at UCSD, images rendered by VMD
Winner of Gordon Bell Special Prize at SC20, project involved overall 1.13 Zettaflops of NAMD simulation

(A) Virion, (B) Spike, (C) Glycan shield conformations



Scaling performance:

- ~305M atom virion
- ~8.5M atom spike



Casalino, et al. *IJHPCA*, 2021 https://doi.org/10.1177%2F10943420211006452

NIH National Institutes of Health

Original GPU-Offload Scheme

Partition work between CPU and GPU

Showing approximate percentage of total work per step:

force calculation on GPUs

- **Short-range non-bonded forces (90%)**
- **Long-range PME electrostatics (5%)**
- **Bonded forces (2%)**
- **Corrections for excluded interactions (2%)**

update coordinates on CPUs

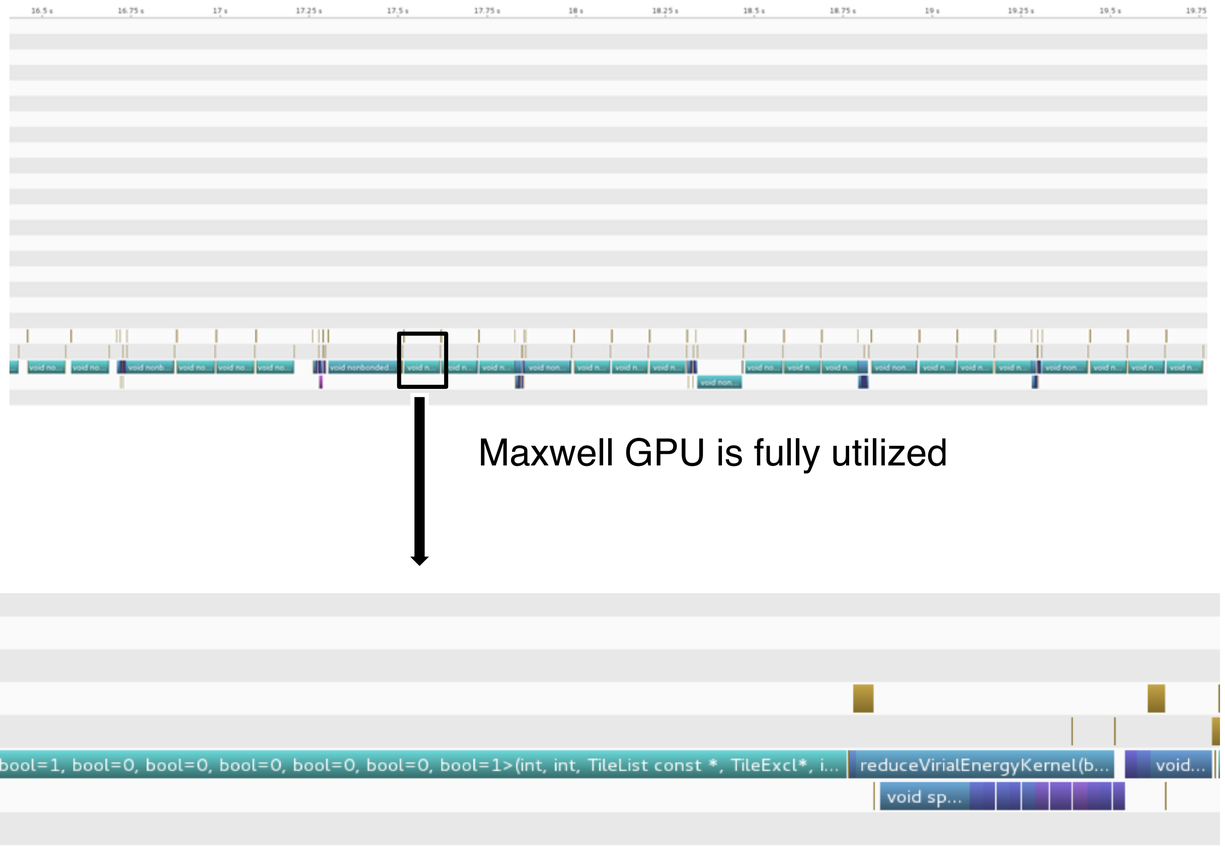
- **Integrator, rigid bond constraints (1%)**

Enhanced sampling methods: additional forces, grid potentials, collective variables

Original GPU-Offload Scheme

Good enough until Pascal (2016)

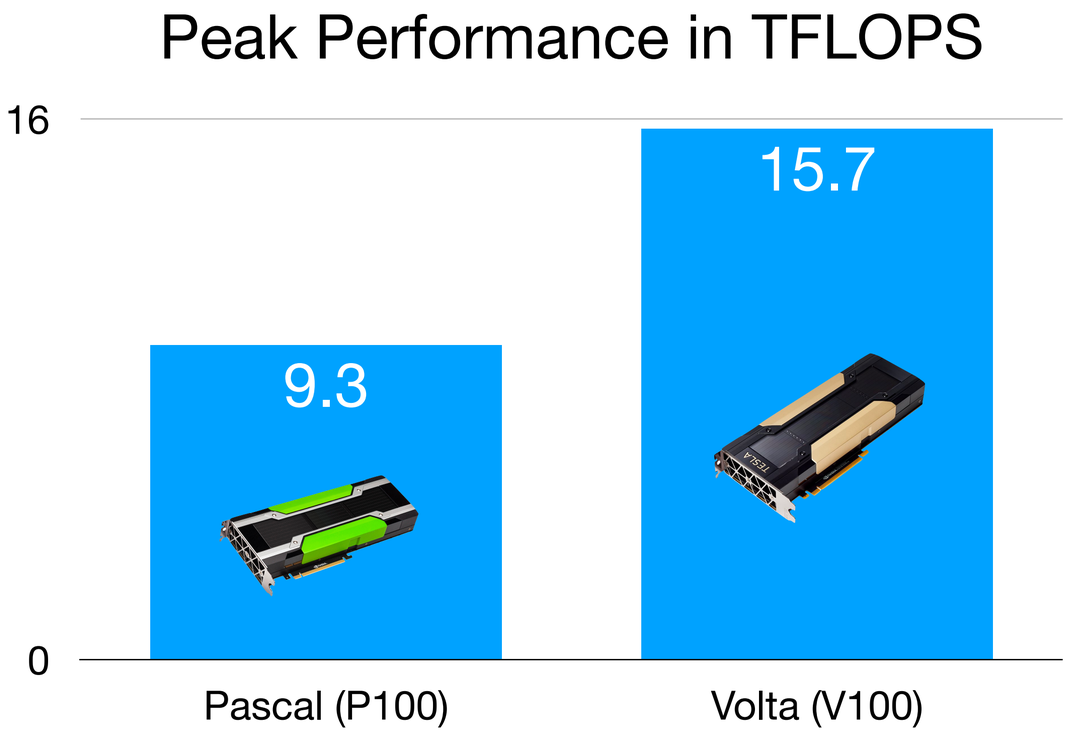
- GPUs weren't **that** fast back then
- Profiling shows GPUs are fully occupied by forces - no idle time
- Streaming forces allows overlap of CPU and GPU computation



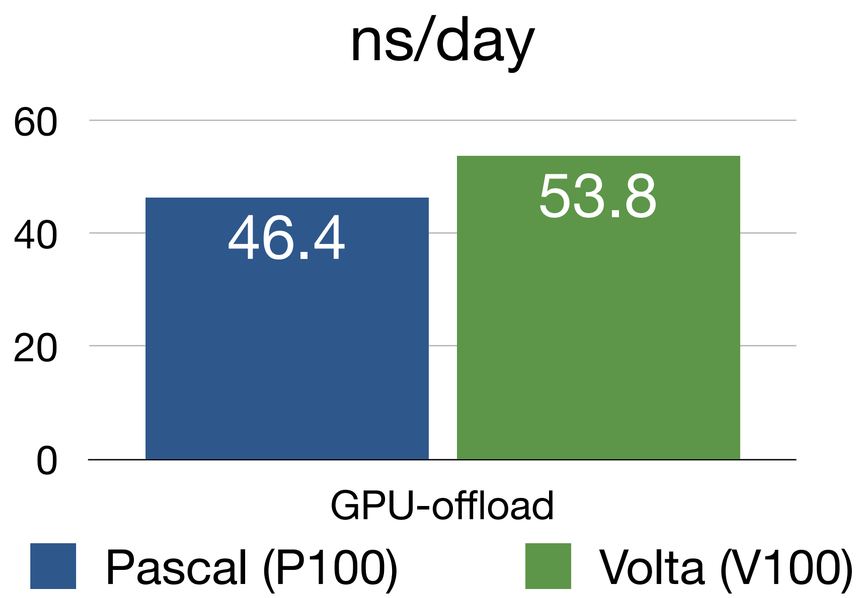
Original GPU-Offload Scheme

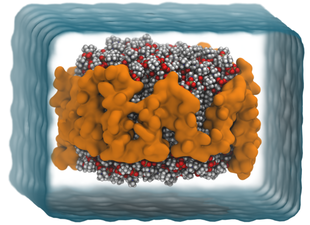
Benchmarking on newer GPUs revealed problems

Hardware has ~70% perf improvement!



NAMD (in 2018) less than 20% perf improvement!





ApoA1
92k atoms

Simulation details:
NVE, CHARMM force field, cutoff distance 12Å,
MTS with 2fs time step and 4fs PME, rigid bond constraints.
https://www.ks.uiuc.edu/Research/namd/benchmarks/

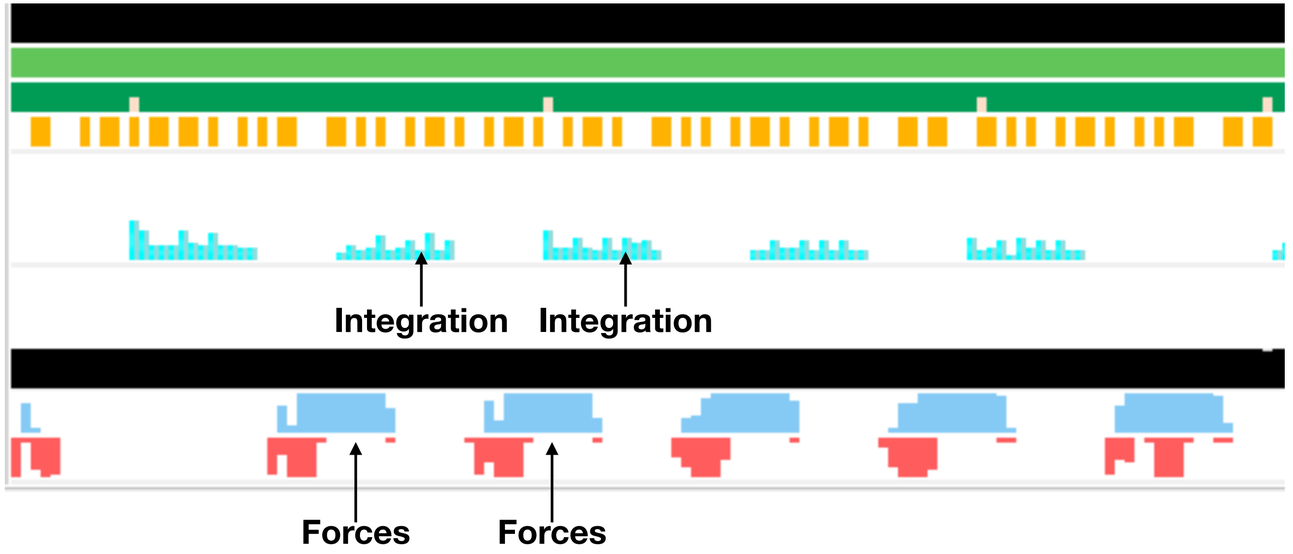
NIH National Institutes of Health

Original GPU-Offload Scheme

CPU-bound on Volta and beyond

- GPUs became **much** faster!
- Attempt to overlap CPU and GPU causes performance bottleneck
- Unable to fully utilize GPU

Profile using **Nsight Systems** with NVTX tags to trace execution of CPU kernels:



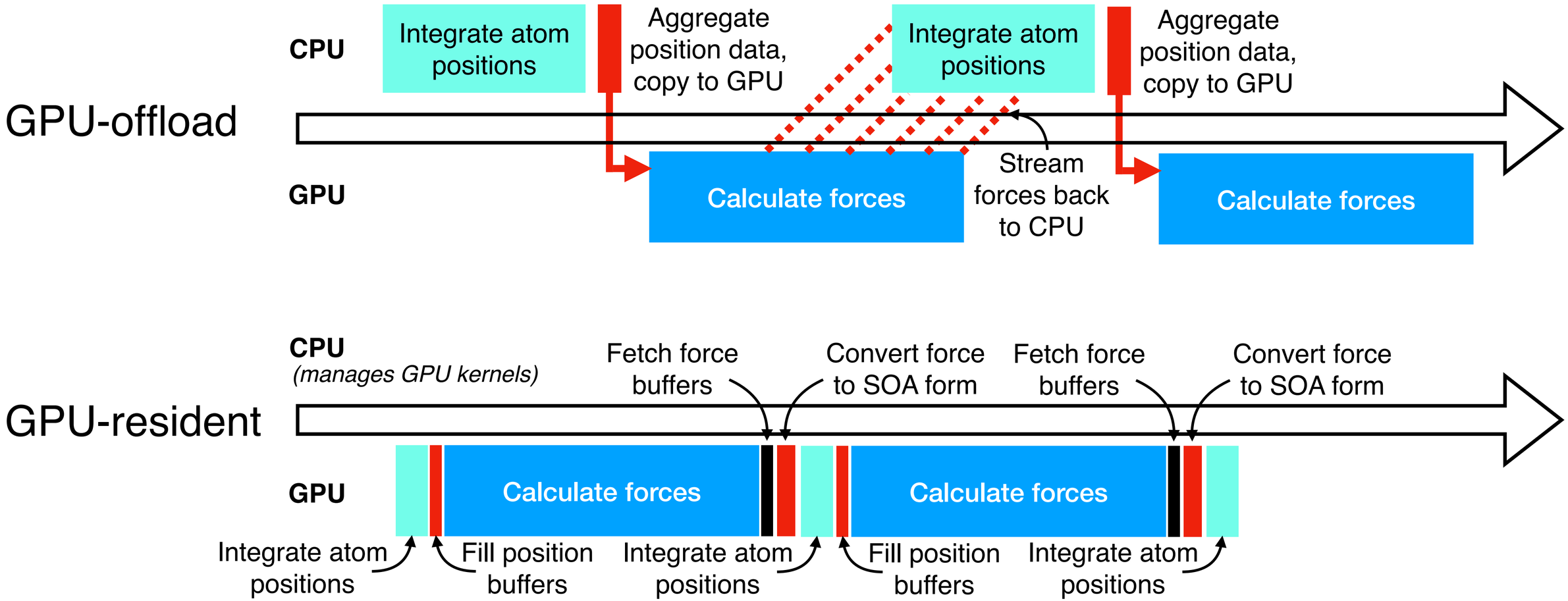
Original GPU-Offload Scheme

Performance limitations on modern GPUs

- Offloading force calculation is not enough!
- Overall utilization of modern GPUs is limited by remaining CPU work
- We want better GPU performance
  - Strong scaling of small- to medium-sized systems is not well served by traditional supercomputers
  - Majority of MD users run system sizes < 1M atoms that are suitable for a single GPU
- Must transition from **GPU-offload** to **GPU-resident**!

New GPU-Resident Scheme

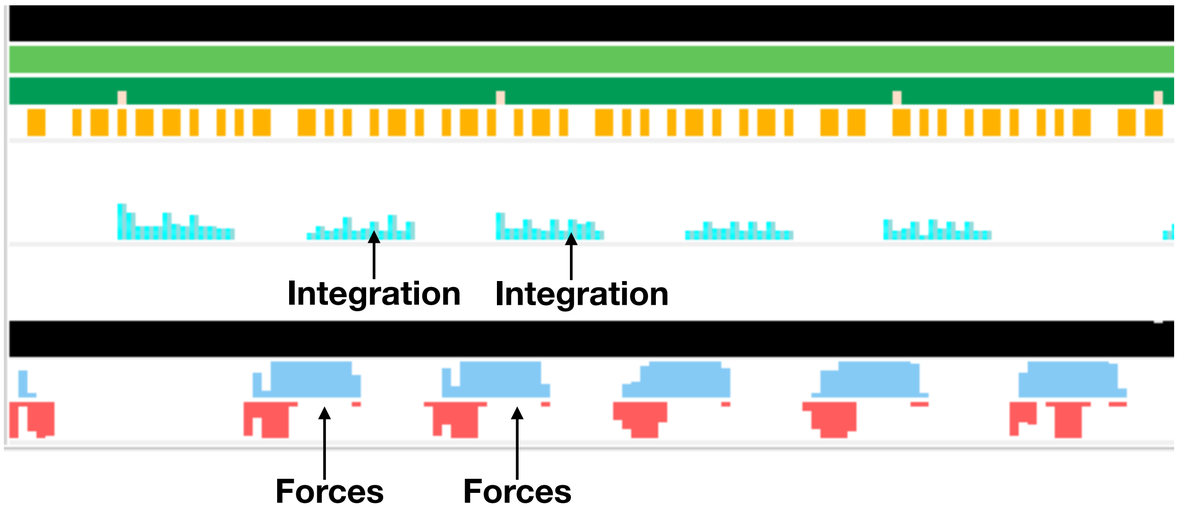
Move integrator to GPU and maintain data between time steps



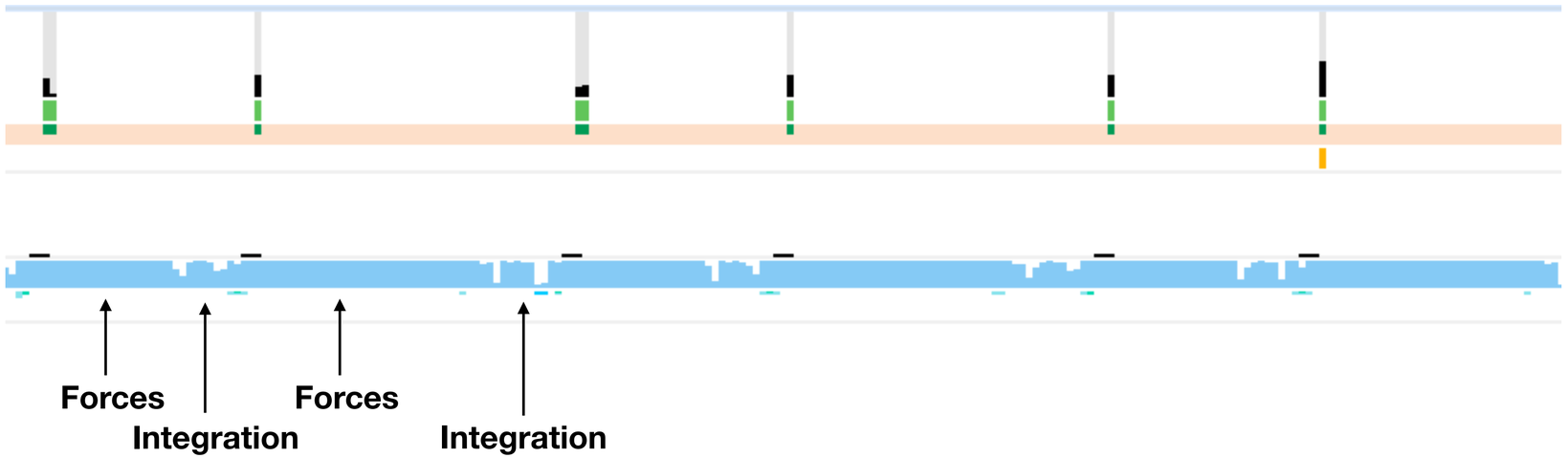
New GPU-Resident Scheme

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

Before (GPU-offload):



After (GPU-resident):

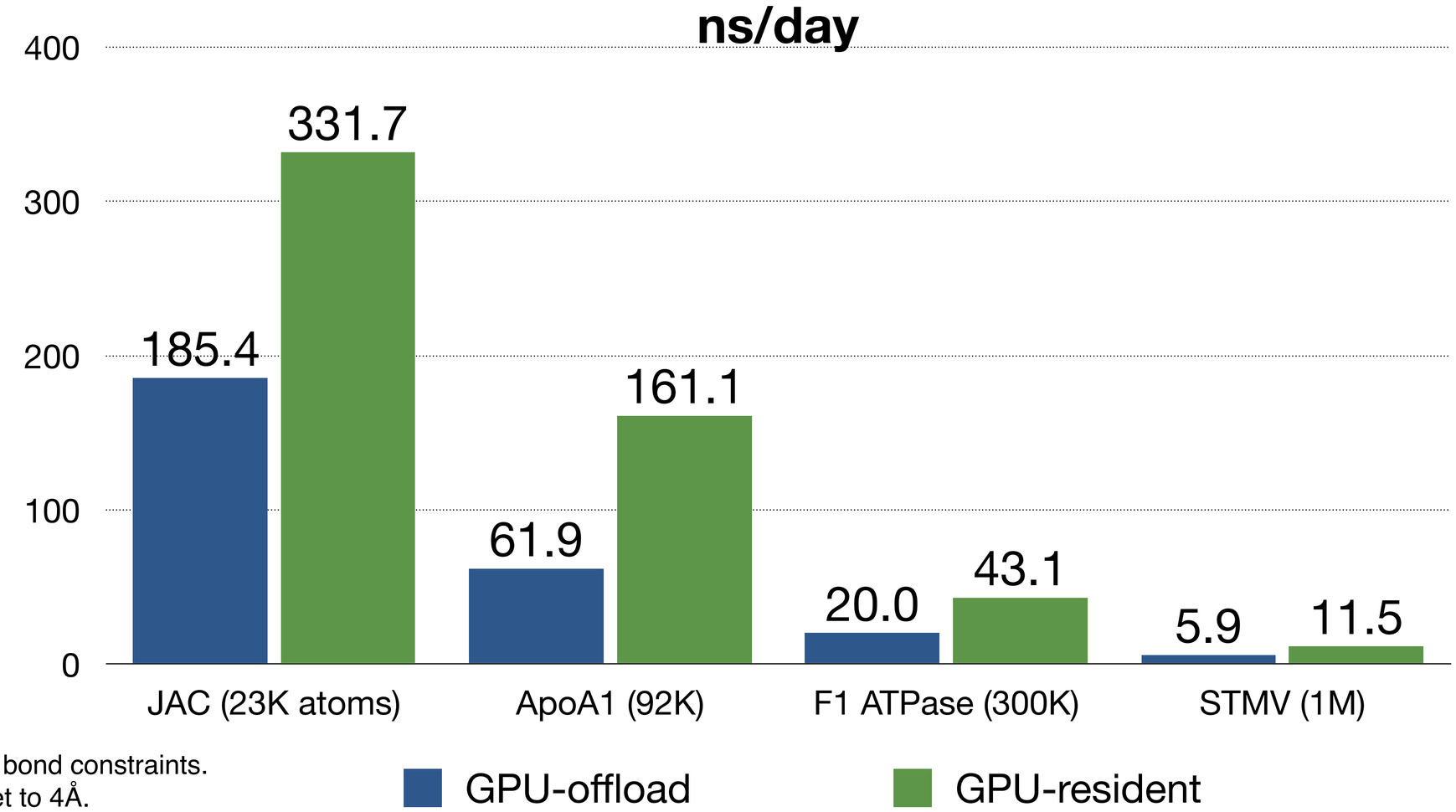


New GPU-Resident Scheme

Performance for constant energy (NVE) simulation on single GPU

Intel Xeon Gold 6134 @ 3.2 GHz

NVIDIA A100-PCIe

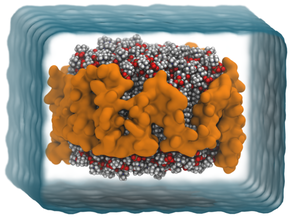


Simulation details:
CHARMM force field, cutoff distance 12Å,
MTS with 2fs time step and 4fs PME, rigid bond constraints.
Performance tuning parameter “margin” set to 4Å.
https://www.ks.uiuc.edu/Research/namd/benchmarks/

NIH National Institutes of Health

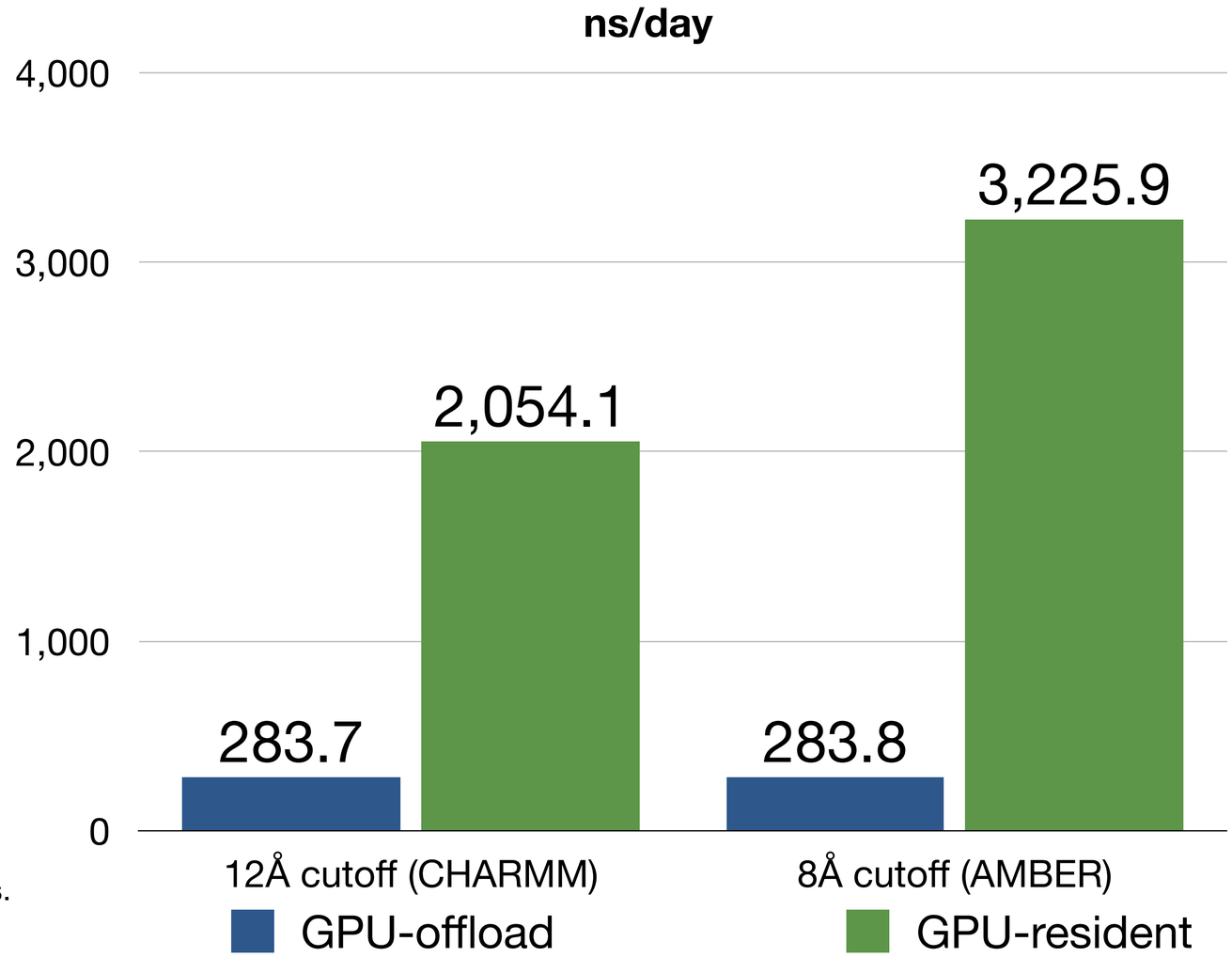
New GPU-Resident Scheme

Aggregate throughput for constant energy (NVE) simulation on GPU-dense hardware



ApoA1
92k atoms

DGX-2 running 16 replicas, one for each NVIDIA V100



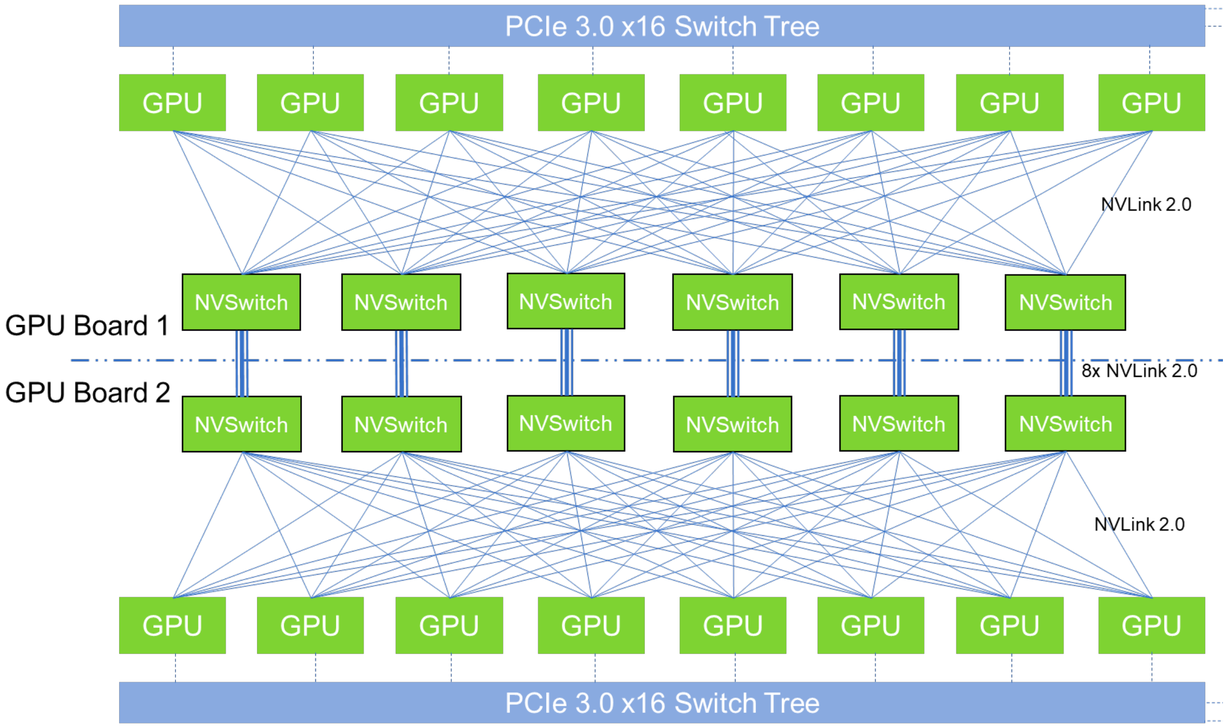
Simulation details:
CHARMM force field uses cutoff distance 12Å,
AMBER force field uses cutoff distance 8Å,
MTS with 2fs time step and 4fs PME, rigid bond constraints.
Performance tuning parameter “margin” set to 4Å.
https://www.ks.uiuc.edu/Research/namd/benchmarks/

NIH National Institutes of Health

Goal: Support GPU-Dense Architectures

Scaling a single simulation across multiple GPUs

- Extend GPU-resident implementation for many interconnected GPUs on a single-node
- Upcoming leadership class supercomputers will also have many GPUs per node



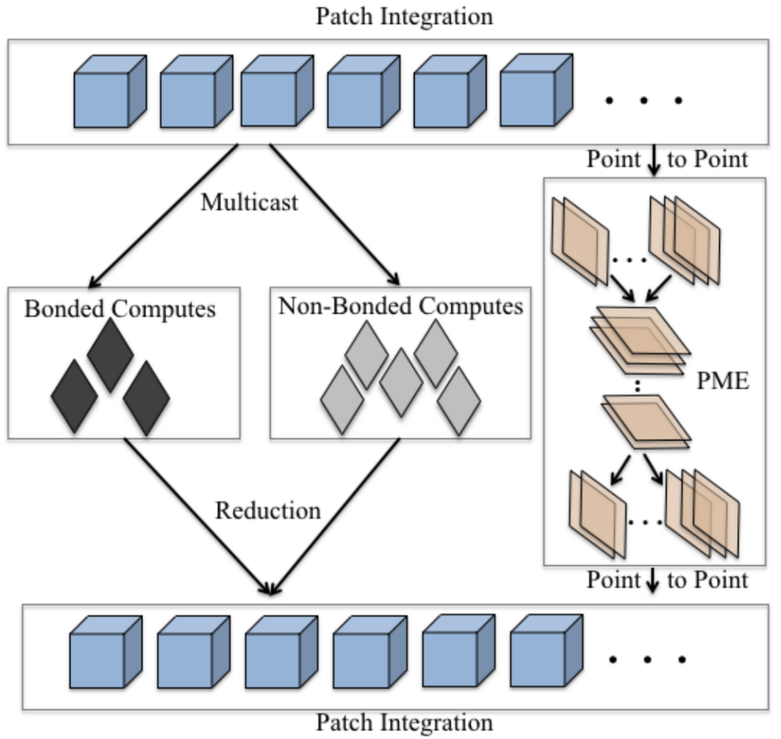
summit

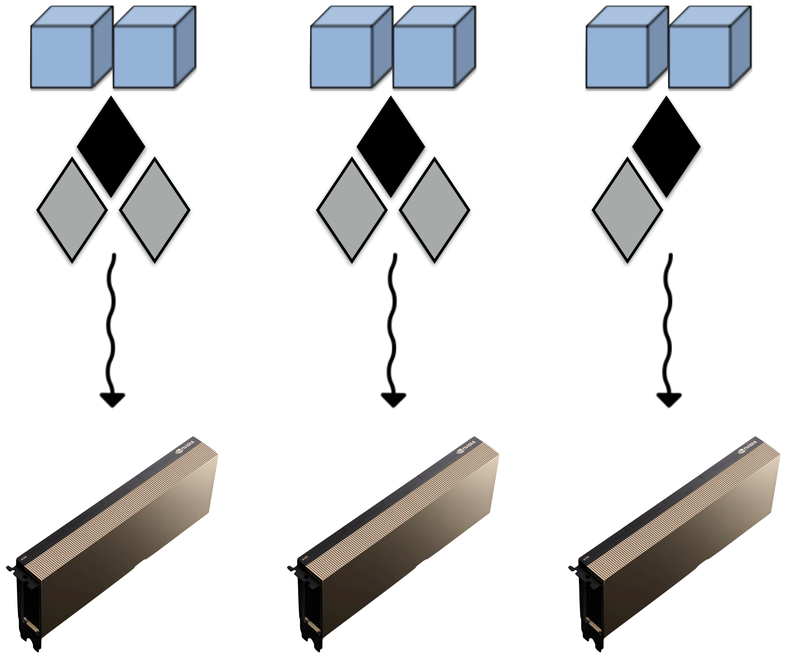
NIH National Institutes of Health

Adapting NAMD's Scalability to GPU-Resident Version

Apply similar decomposition of data and work among GPUs

- Each CPU thread binds to a particular GPU
- Aggregate compute and patch data per thread to launch integration and force kernels
- Maintain a single thread per GPU to make things easier
- Exploit tightly coupled (peered) GPUs (NVLink, PCIe, …)

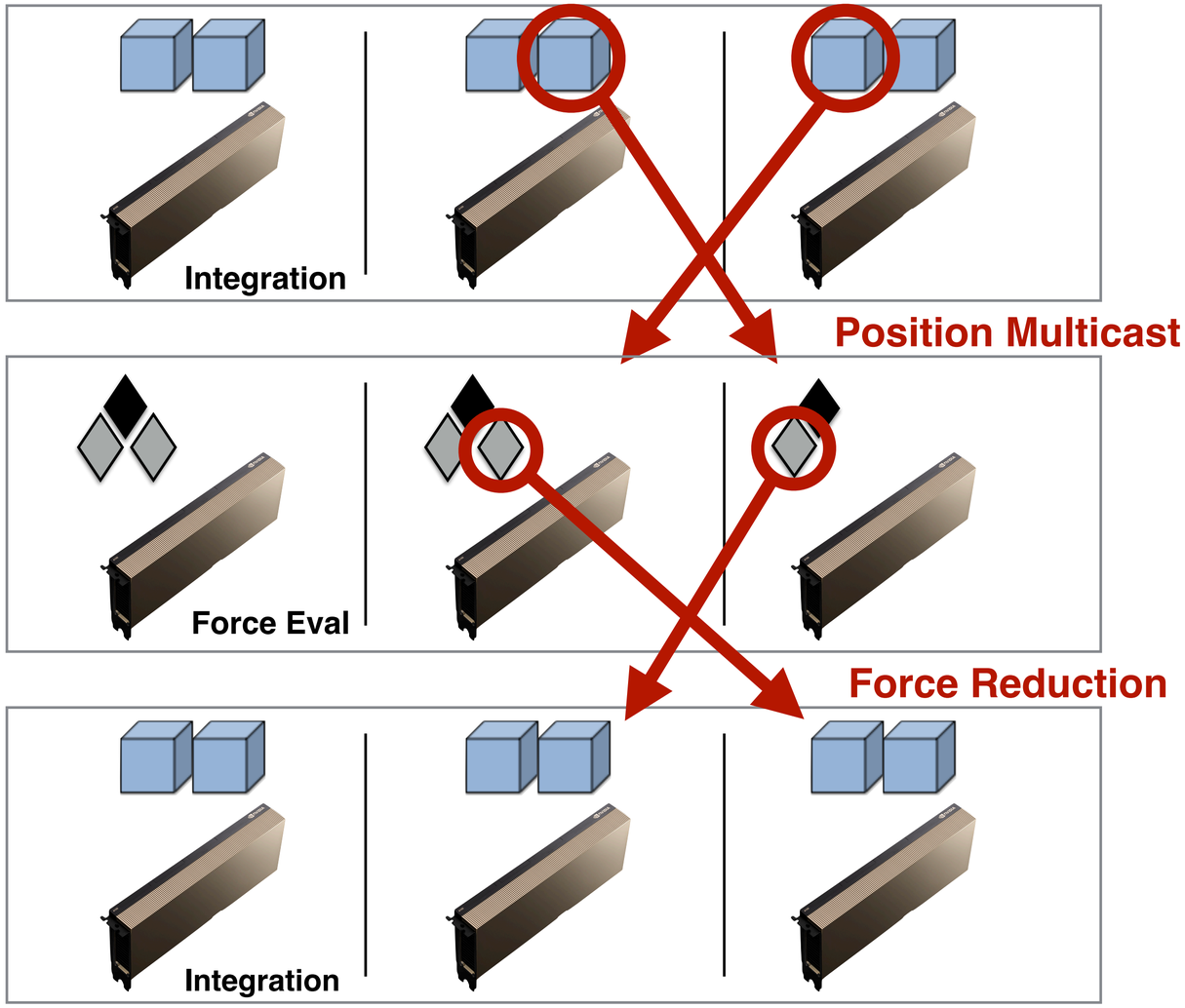




Adapting NAMD's Scalability to GPU-Resident Version

Some communication required: multicasts and reductions

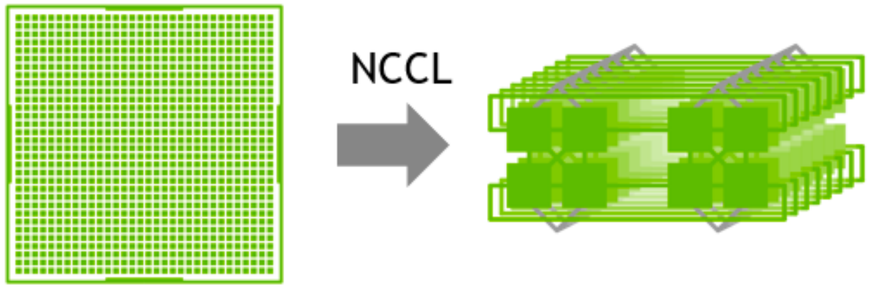
- GPUs need to communicate information among themselves during simulation
- Update atom positions in each patch during integration
- Perform **position multicast** to compute objects
- Compute new forces
- Perform **force reduction** back to patches

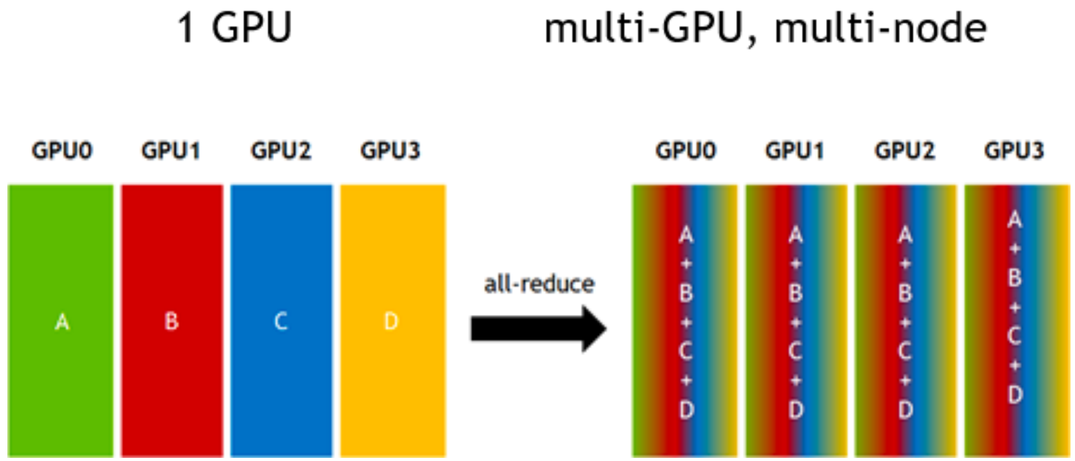


Adapting NAMD's Scalability to GPU-Resident Version

Rapid prototyping using NCCL for collectives

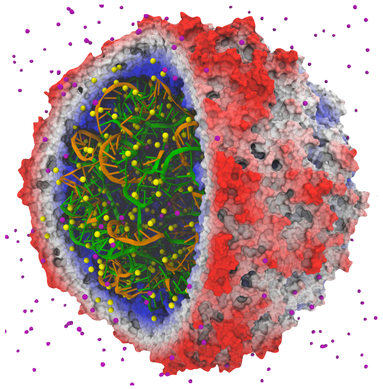
- The multicast and reduction operations can be expressed as ALLREDUCE primitives
- Use NCCL (NVIDIA Collective and Communications Library) to get it working quickly
- *NCCL* also abstracts the underlying GPU topology, while still achieving high bandwidth
- However, using ALLREDUCE results in wasted communication
  - Too hard to use scatter/gather because our data lacks regularity
  - Better performance expected if we do our own communication



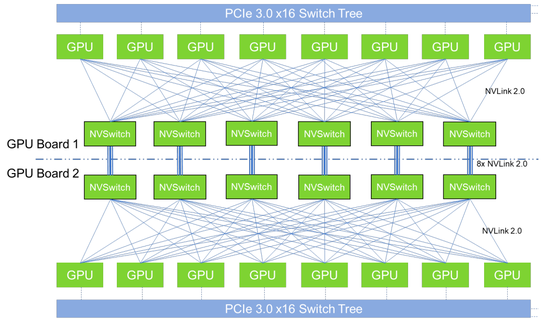


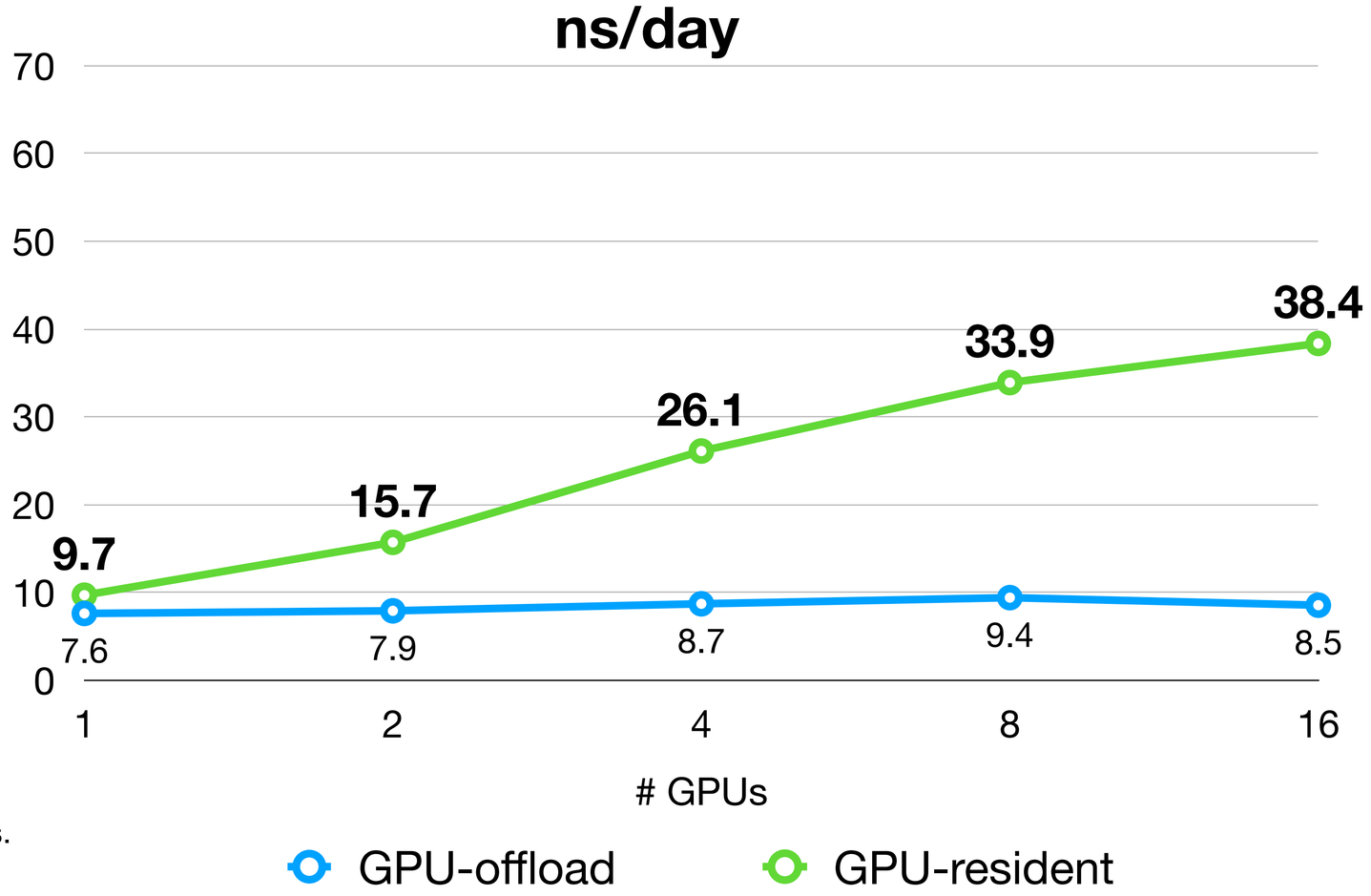
Multi-GPU Scaling on DGX-2

Using NCCL for communication collectives



STMV
1.06M atoms
2fs timestep





Simulation details:
NVE, CHARMM force field, cutoff distance 12Å,
MTS with 2fs time step and 4fs PME, rigid bond constraints.
Performance tuning parameter “margin” set to 4Å.
https://www.ks.uiuc.edu/Research/namd/benchmarks/

NIH National Institutes of Health

Multi-GPU Scaling on DGX-2

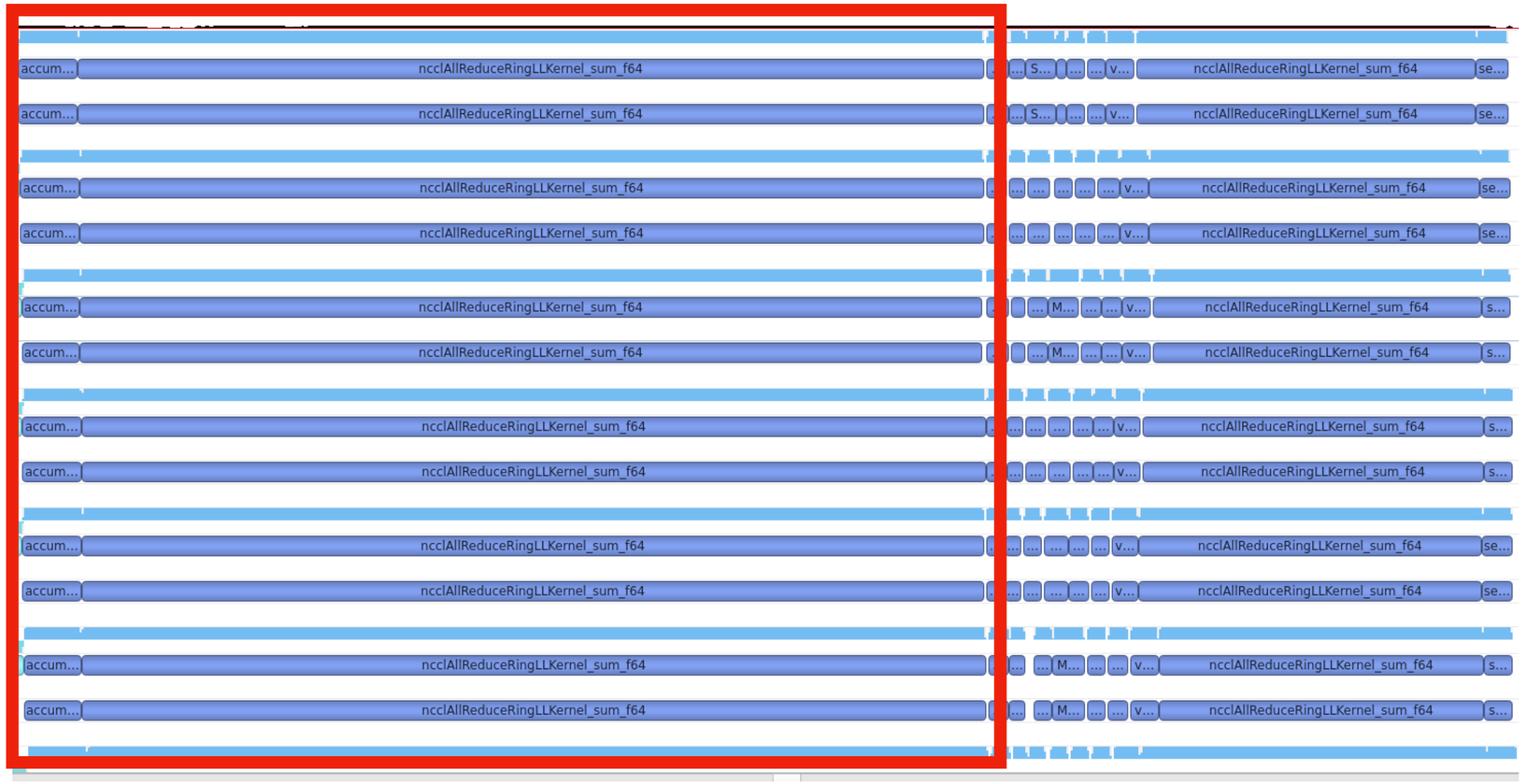
How to improve scalability on large GPU counts?

NCCL's ALLREDUCE accounts for 40% of GPU kernel execution time on 8 GPUS!

Multi-GPU Scaling on DGX-2

Replacing NCCL: Point-to-point force sum reductions

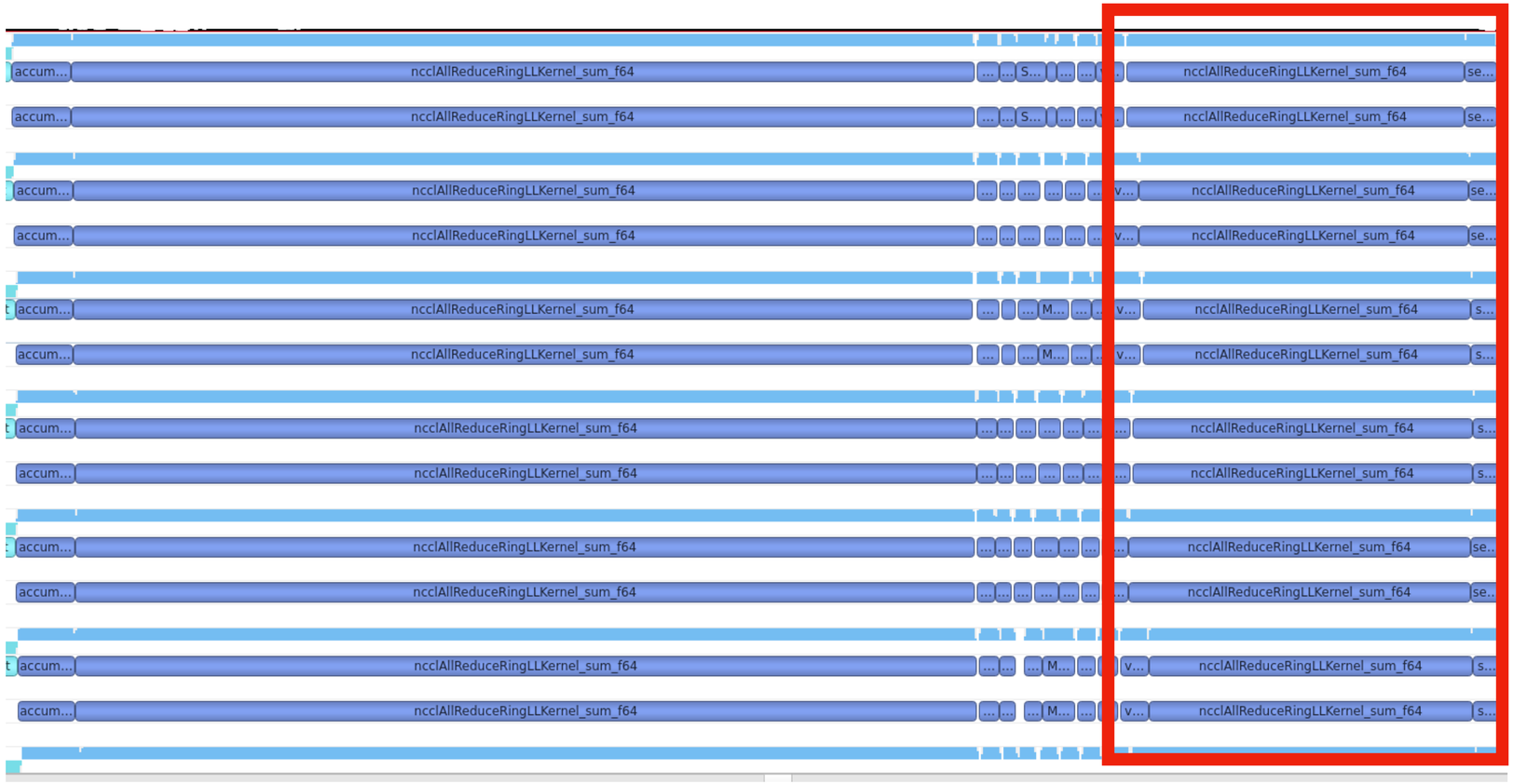
- Force sum reductions take longer than position multicasts
- Need to replace them with a specialized kernel for our problem, since not all values need to be reduced!
- Use CUDA peer-to-peer functionalities for better performance



Multi-GPU Scaling on DGX-2

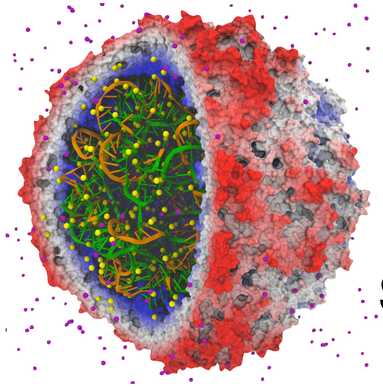
Replacing NCCL: Point-to-point position multicasts

- Same logic as before: Replace ALLREDUCE with a specialized kernel
- Each GPU fills its positions by accessing its peers' memories (better performance with NVLINK)
- Retrieve only those positions owned by that GPU for better scaling

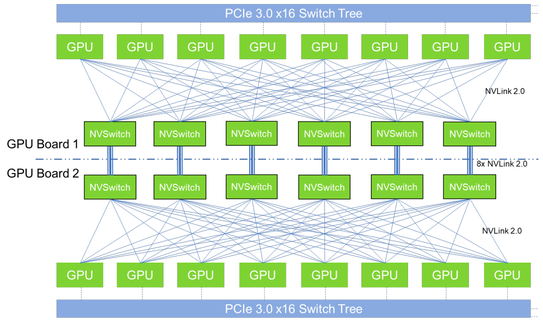


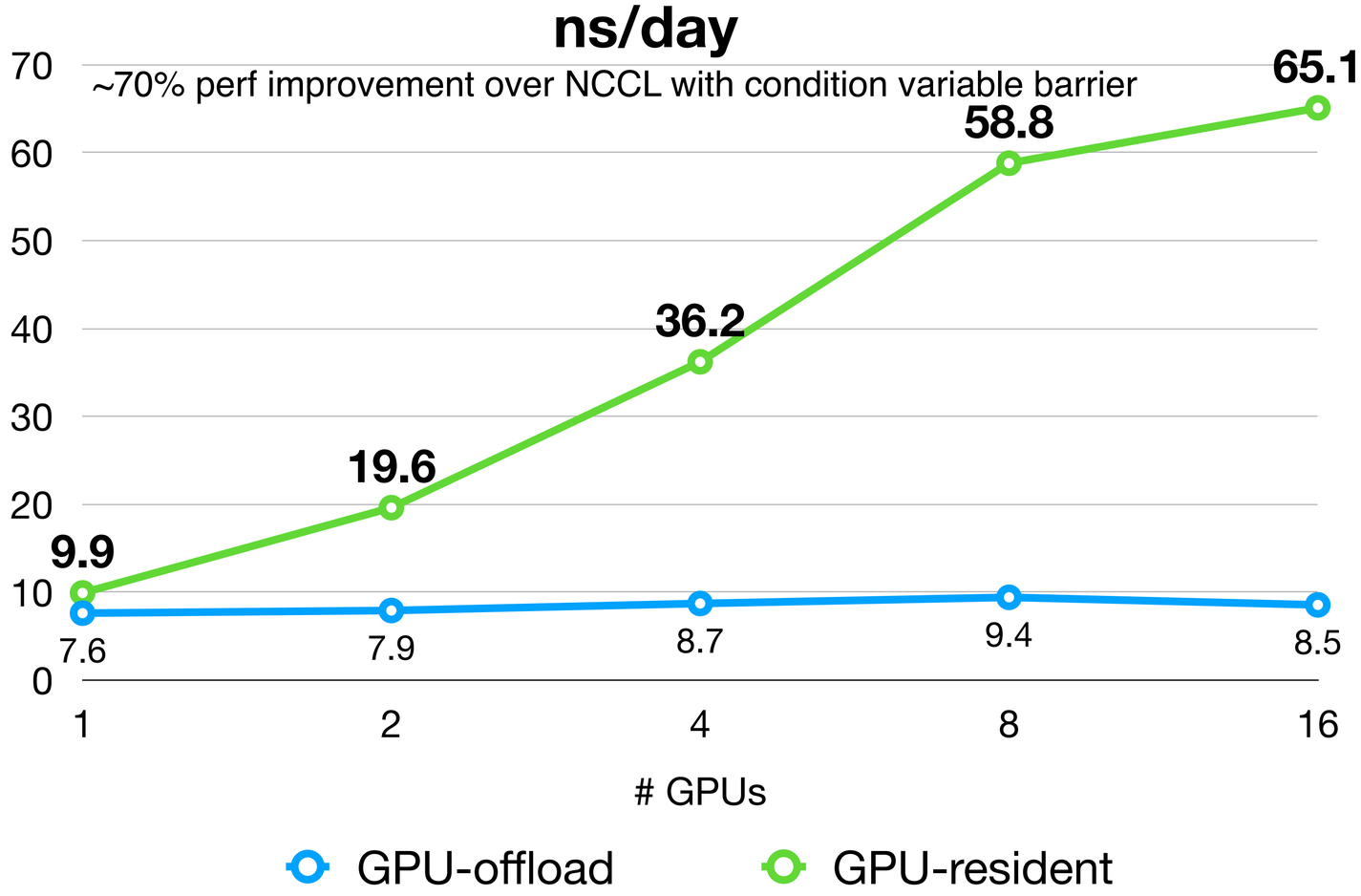
Multi-GPU Scaling on DGX-2

Overcoming CPU thread synchronization latency with spinlock barrier



STMV
1.06M atoms
2fs timestep
Spinlock barrier

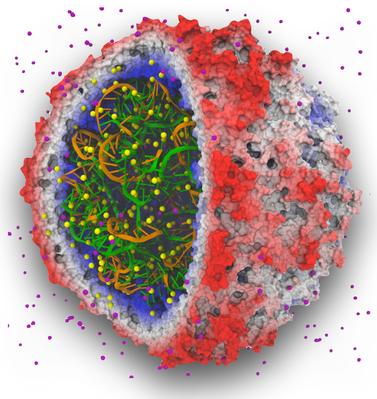




Simulation details:
NVT 300K, CHARMM force field, cutoff distance 12Å,
with 2fs time step, without PME, rigid bond constraints.
Performance tuning parameter “margin” set to 4Å.
https://www.ks.uiuc.edu/Research/namd/benchmarks/

NIH National Institutes of Health

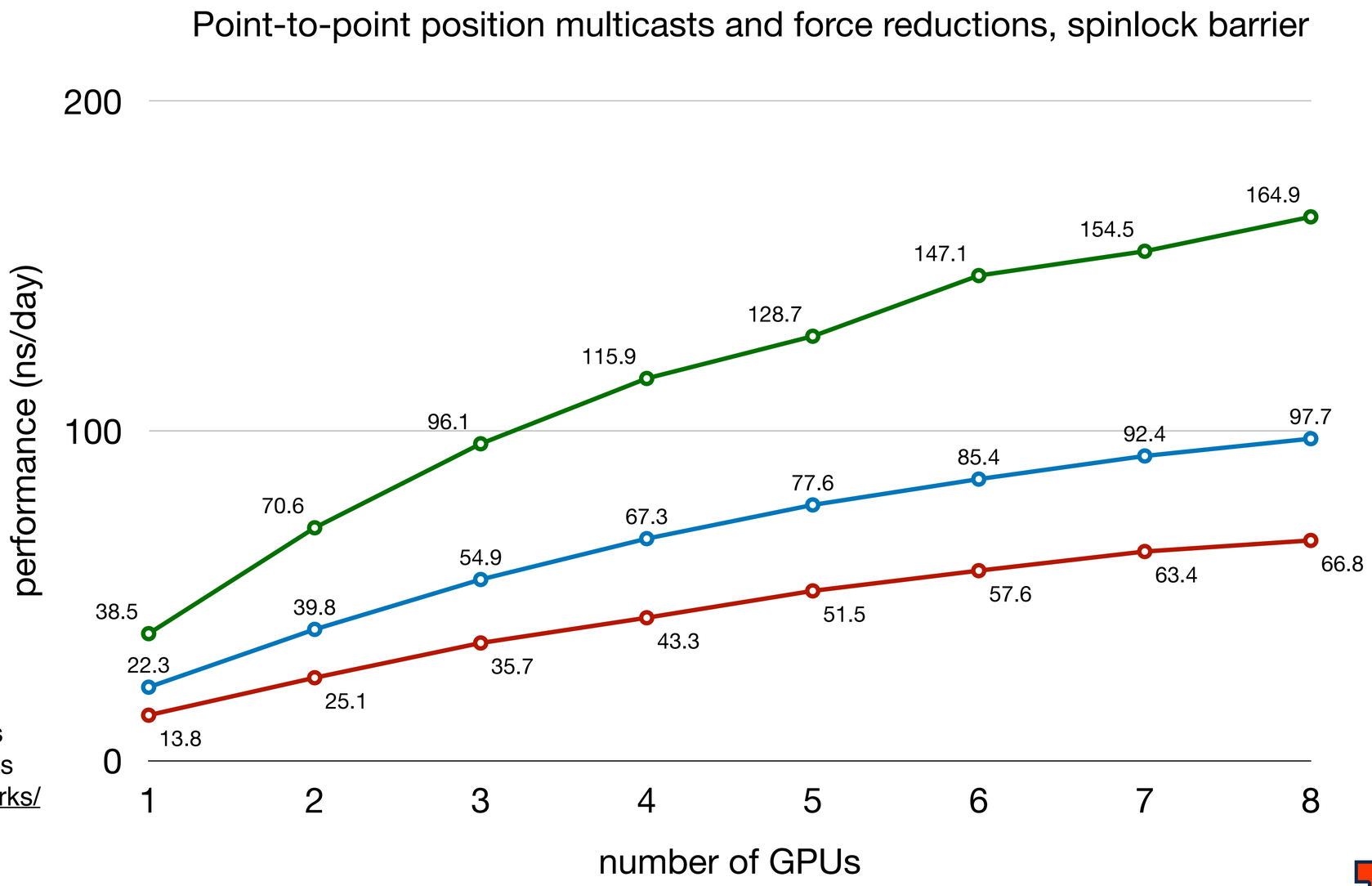
Multi-GPU Scaling on DGX-A100



STMV
1.067M atoms

DGX-A100

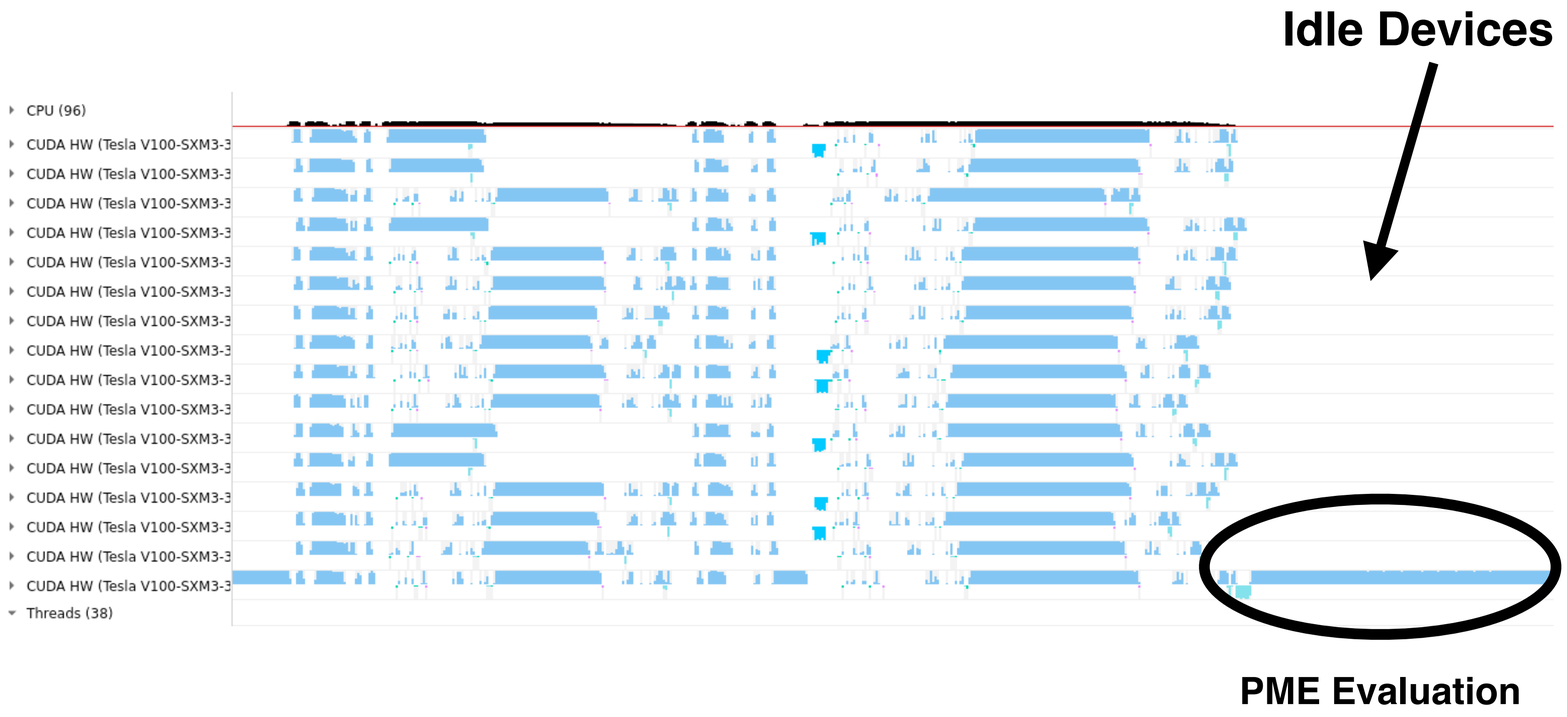
Simulation details:
Green line: 8Å cutoff, HMR, 4fs PME, rigid bonds
Blue line: 8Å cutoff, MTS 2fs, 4fs PME, rigid bonds
Red line: 12Å cutoff, MTS 2fs, 4fs PME, rigid bonds
https://www.ks.uiuc.edu/Research/namd/benchmarks/



Future Improvements to GPU-Resident

Overcoming scaling bottleneck from PME long-range electrostatics

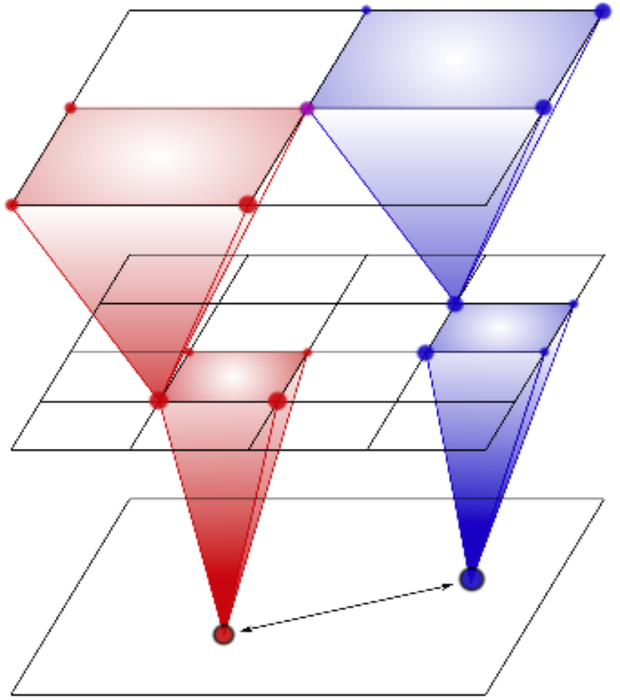
- PME (particle-mesh Ewald) requires calculating FFT
  - 3D FFTs for PME are too small to parallelize effectively with cuFFT
  - Too much latency is introduced with slab or pencil decomposition
- Assign PME to a single device
  - But over assignment can cause load imbalance



Future Improvements to GPU-Resident

Replacing PME with better scaling MSM algorithm

- MSM (multilevel summation method) provides a better scaling alternative to PME
- Hierarchical grid calculation offers tree-like work decomposition, similar in structure to FMM (fast multipole method)
- Localized 3D convolutions involving nearest neighbor communication are well suited to GPU computation
- Coarsest level grid has 3D FFT, but it can be made as small as desired



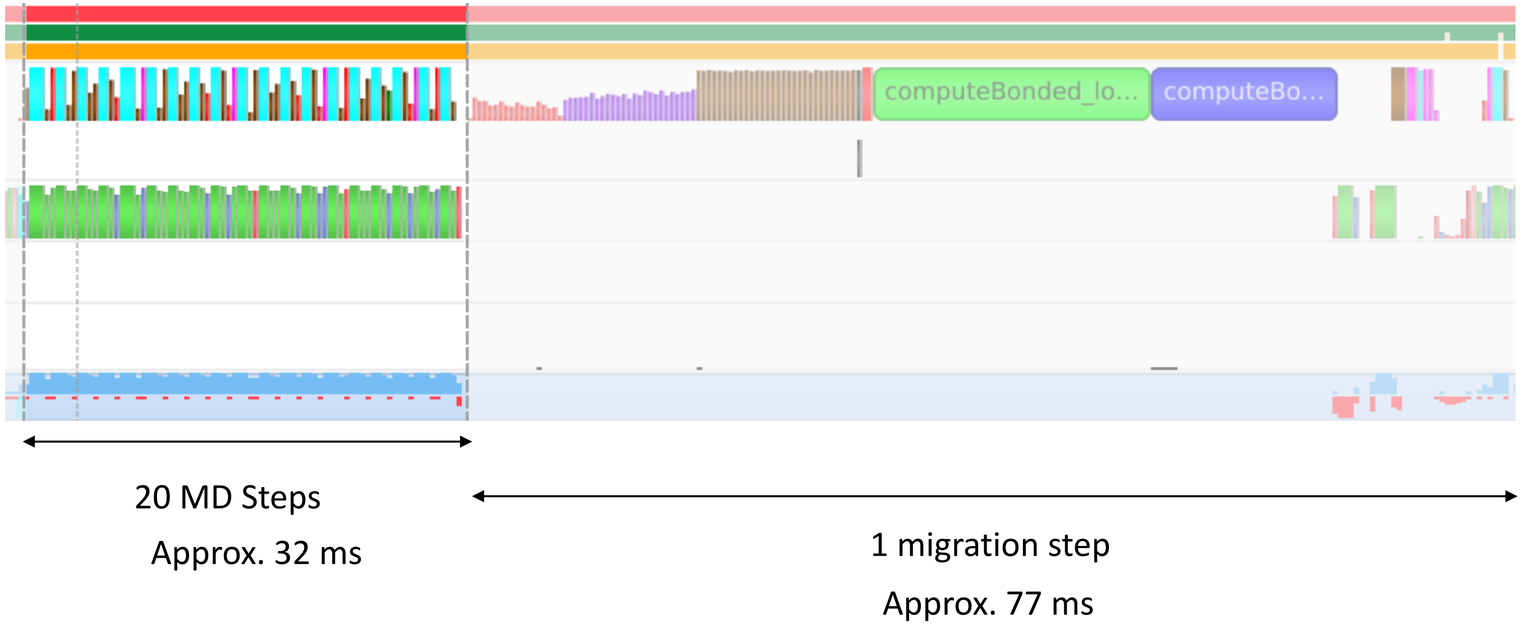
Hardy, Stone, Schulten. *J. Paral. Comp.* (2009)
Hardy, Wu, et al. *J. Chem. Theory Comput.* (2015)
Hardy, Wolff, et al. *J. Chem. Phys.* (2016)
Kaya, Hardy, Skeel. *J. Chem. Phys.* (2021)

Future Improvements to GPU-Resident

Offload domain decomposition (atom migration) to GPU

- Domain decomposition (atom migration) presently poses a large CPU bottleneck, mitigated by:
  - Increasing patch margin
  - Performing only when necessary
- Bypass CPU thread synchronization and data buffering
- Multi-GPU implementation will require additional communication
- Restructure fundamental data structures for fast recalculation of force auxiliary arrays

Profiling on single GPU



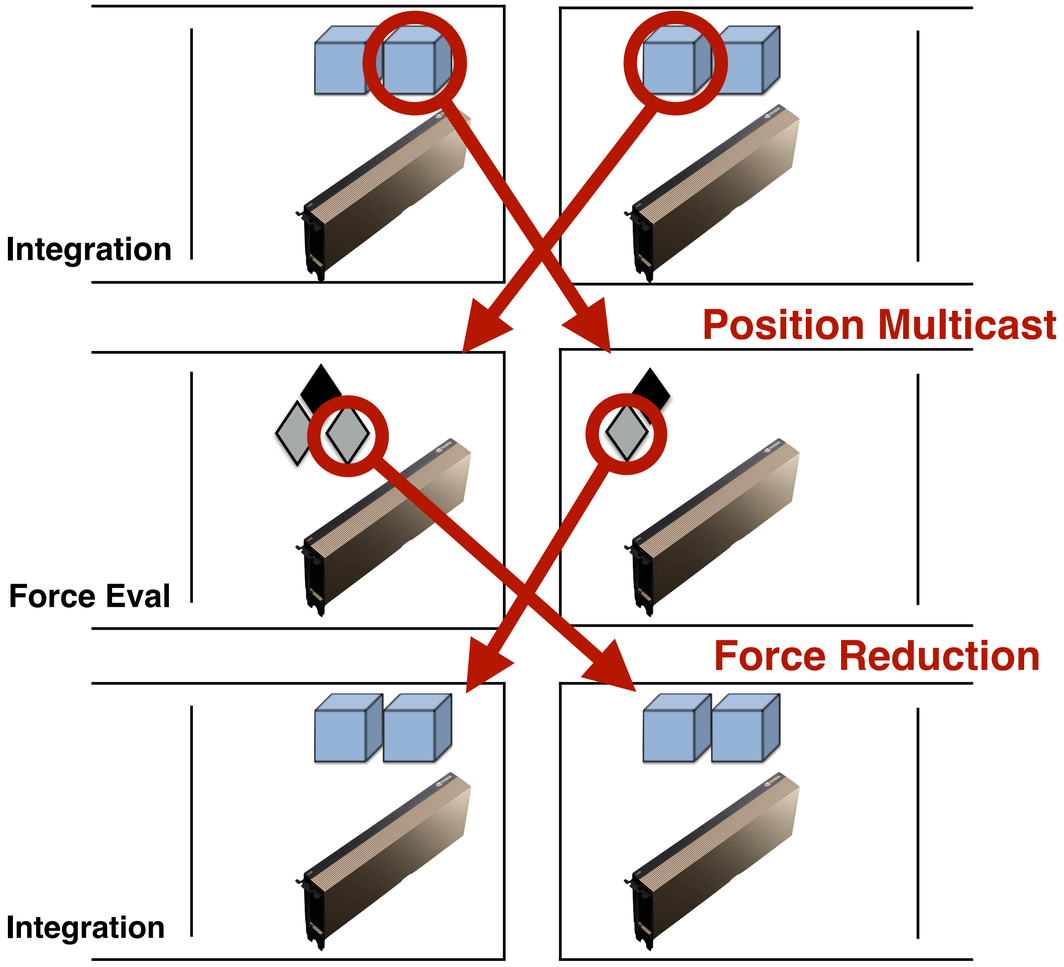
Time for atom migration is equal to 48 MD steps

NAMD's default is 20 steps per migration

Future Improvements to GPU-Resident

Multi-node scaling

- Minimize multi-node communication latency between GPU-resident NAMD processes running on different nodes
- Exploit Infiniband-connected DGX-like nodes using RDMA hardware acceleration
  - Use fabric-based (switch, NIC) collective operations and reductions to avoid host CPU involvement
- Ultimately, Charm++ will be needed for large scale runs
  - Use GPU-direct communication
  - Load balancing needs to understand GPU workloads



Challenge: Support for New GPUs

- AMD GPU support (Josh Vermaas, Julio Maia)
  - Use Hipify to translate CUDA to HIP, no need for direct HIP implementation
  - Few additional tweaks required:
    - HIP wavefront size 64 vs CUDA warp size 32
    - Need some macro definitions in extra header file
    - Workaround for texture memory interpolation
  - GPU-offload already available, Julio is developing GPU-resident

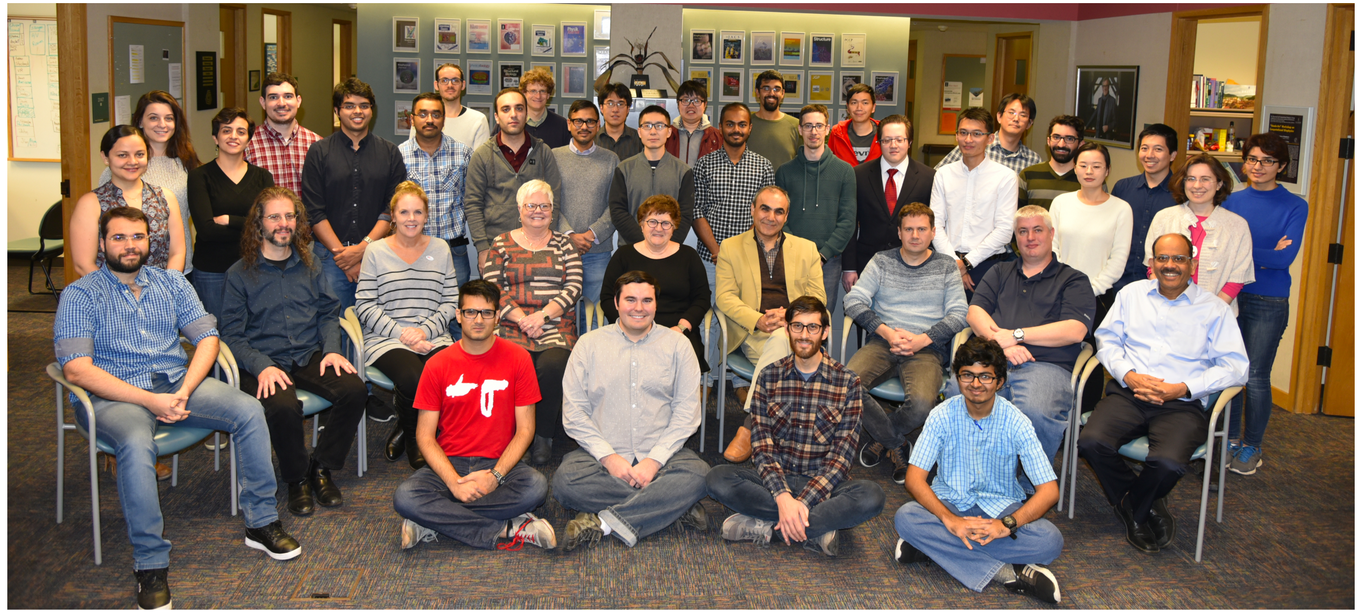
- Intel GPU support (Tareq Malas, Jaemin Choi)
  - DPC++ is significantly different from CUDA, requires its own implementation
  - Assisted by a conversion tool
  - Significant modifications required after conversion
  - Recently have working force kernels for GPU-offload

Challenge: GPU-Resident Feature Support

- Essential standard integration methods supported
  - Constant energy, constant temperature (Langevin damping and stochastic rescaling), constant pressure (Langevin piston)
  - Multiple time stepping
  - Rigid bond constraints
- Some advanced features already supported
  - Alchemical free energy methods FEP and TI (Haochuan Chen, Julio Maia)
    Chen, et al. *J. Chem. Inf. Model.* 60 (11), 5301-5307 (2020)
  - Multi-copy simulation (e.g. replica-exchange)
  - External electric field
- Other features require extensive porting to GPU
  - Colvars (collective variables) module

Acknowledgments

- NAMD development is funded by NIH P41-GM104601
- DPC++ porting is funded in part by Intel
- GPU programming team: Julio Maia (AMD); David Clark (NVIDIA); Tareq Malas (Intel); Jaemin Choi, John Stone (UIUC)



NIH Center for Macromolecular Modeling and Bioinformatics
Beckman Institute, University of Illinois at Urbana-Champaign (2018)

NIH National Institutes of Health