Introduction to
NAMD 3.0

Rafael C. Bernardi

Department of Physics at Auburn University
NIH Center for Macromolecular Modeling and Visualization
rcbernardi@auburn.edu
@rafaelcbernardi

COLLEGE OF SCIENCES
AND MATHEMATICS

What is new in NAMD 3.0

- Released June 14, 2024

- NAMD 3.0 has many advantages over NAMD 2.14, including:

  - GPU-resident mode providing very fast dynamics:
    Achieves 2x or more speedup on single GPU versus GPU-offload mode, and
    7x or more speedup for multi-GPU scaling on NVIDIA DGX-A100 versus GPU-offload mode
    Supports single GPU and single-node multi-GPU scaling for tightly coupled GPUs
    GPU-accelerated alchemical free energy methods (FEP & TI) for both GPU-offload and GPU-resident modes

  - HIP kernel improvements for better performance on AMD GPUs
  - CPU-vectorization mode compatible with Intel and AMD CPU models that support AVX-512 instructions
  - Achieves speedup of up to 1.8x on Intel Xeon over AVX2 builds
  - Update Colvars to version 2024-06-04
  - Fix several long-standing issues
  - Fix CPU memory leaks
  - Fix using advanced features together with GPU atom migration

  ...

https://www.ks.uiuc.edu/Research/namd/3.0/announce.html

Rafael C. Bernardi
rcbernardi@auburn.edu

What is Molecular Dynamics

$$V = V_{str} + V_{bend} + V_{tors} + V_{oop} + V_{vdW} + V_{es}$$

$$V_{es,ij} = Q_i Q_j / \varepsilon_r R_{ij}$$

$$V_{vdW,ij} = \varepsilon_{IJ}\left[\left(\frac{R_{IJ}}{R_{ij}}\right)^{12} - 2\left(\frac{R_{IJ}}{R_{ij}}\right)^{6}\right]$$

$$V_{oop} = \frac{1}{2} k_{oop} \omega_{oop}^2$$

$$V_{tors,ijkl} = \frac{1}{2}\left[V_1(1+\cos\varphi) + V_2(1-\cos 2\varphi) + V_3(1+\cos 3\varphi) + \ldots\right]$$

$$V_{bend,ijk} = \frac{1}{2} k_{IJK}\left(\theta_{ij} - \theta^0{}_{IJK}\right)^2$$

$$V_{str,ij} = \frac{1}{2} k_{IJ}\left(l_{ij} - l^0{}_{IJ}\right)^2$$

$$\vec{F}_i = -dV / d\vec{r}_i$$

$$\vec{a}_i = \vec{F}_i / m$$

$$\vec{s}_i = \vec{s}_{0i} + \vec{v}_i t + \frac{1}{2}\vec{a}_i t^2$$

Rafael C. Bernardi
rcbernardi@auburn.edu

What is Molecular Dynamics

$$V = V_{str} + V_{bend} + V_{tors} + V_{oop} + V_{vdW} + V_{es}$$

$$V_{str,ij} = \frac{1}{2} k_{IJ} \left(l_{ij} - l^0_{IJ}\right)^2$$

$$V_{bend,ijk} = \frac{1}{2} k_{IJK} \left(\theta_{ij} - \theta^0_{IJK}\right)^2$$

$$V_{tors,ijkl} = \frac{1}{2}\left[V_1(1+\cos\varphi) + V_2(1-\cos 2\varphi) + V_3(1+\cos 3\varphi) + \ldots\right]$$

$$V_{oop} = \frac{1}{2} k_{oop} \omega^2_{oop}$$

$$V_{vdW,ij} = \varepsilon_{IJ}\left[\left(\frac{R_{IJ}}{R_{ij}}\right)^{12} - 2\left(\frac{R_{IJ}}{R_{ij}}\right)^{6}\right]$$

$$V_{es,ij} = Q_i Q_j / \varepsilon_r R_{ij}$$

$$\vec{F}_i = -dV / d\vec{r}_i$$

$$\vec{a}_i = \vec{F}_i / m$$

$$\vec{s}_i = \vec{s}_{0i} + \vec{v}_i t + \frac{1}{2}\vec{a}_i t^2$$

CPU

Rafael C. Bernardi
rcbernardi@auburn.edu

What is a GPU-resident software

$$V = V_{str} + V_{bend} + V_{tors} + V_{oop} + V_{vdW} + V_{es}$$

$$V_{es,ij} = Q_i Q_j / \varepsilon_r R_{ij}$$

$$V_{vdW,ij} = \varepsilon_{IJ}\left[\left(\frac{R_{IJ}}{R_{ij}}\right)^{12} - 2\left(\frac{R_{IJ}}{R_{ij}}\right)^{6}\right]$$

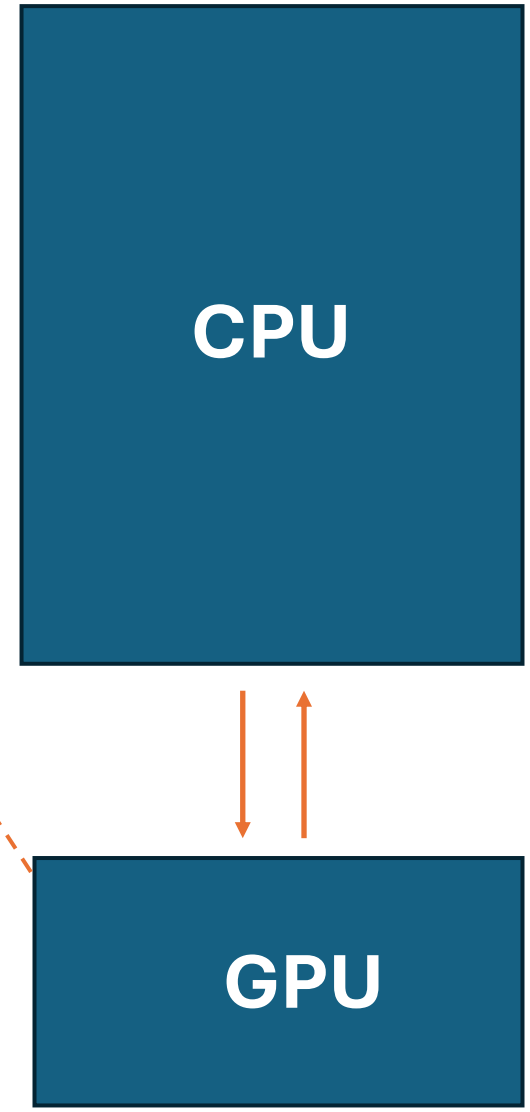
$$V_{oop} = \frac{1}{2} k_{oop} \omega_{oop}^2$$

$$V_{tors,ijkl} = \frac{1}{2}\left[V_1(1+\cos\varphi) + V_2(1-\cos 2\varphi) + V_3(1+\cos 3\varphi) + \ldots\right]$$

$$V_{bend,ijk} = \frac{1}{2} k_{IJK}\left(\theta_{ij} - \theta^0{}_{IJK}\right)^2$$

$$V_{str,ij} = \frac{1}{2} k_{IJ}\left(l_{ij} - l^0{}_{IJ}\right)^2$$

GPU offload



Rafael C. Bernardi
rcbernardi@auburn.edu

What is a GPU-resident software

$$V = V_{str} + V_{bend} + V_{tors} + V_{oop} + V_{vdW} + V_{es}$$

$$V_{es,ij} = Q_i Q_j / \varepsilon_r R_{ij}$$

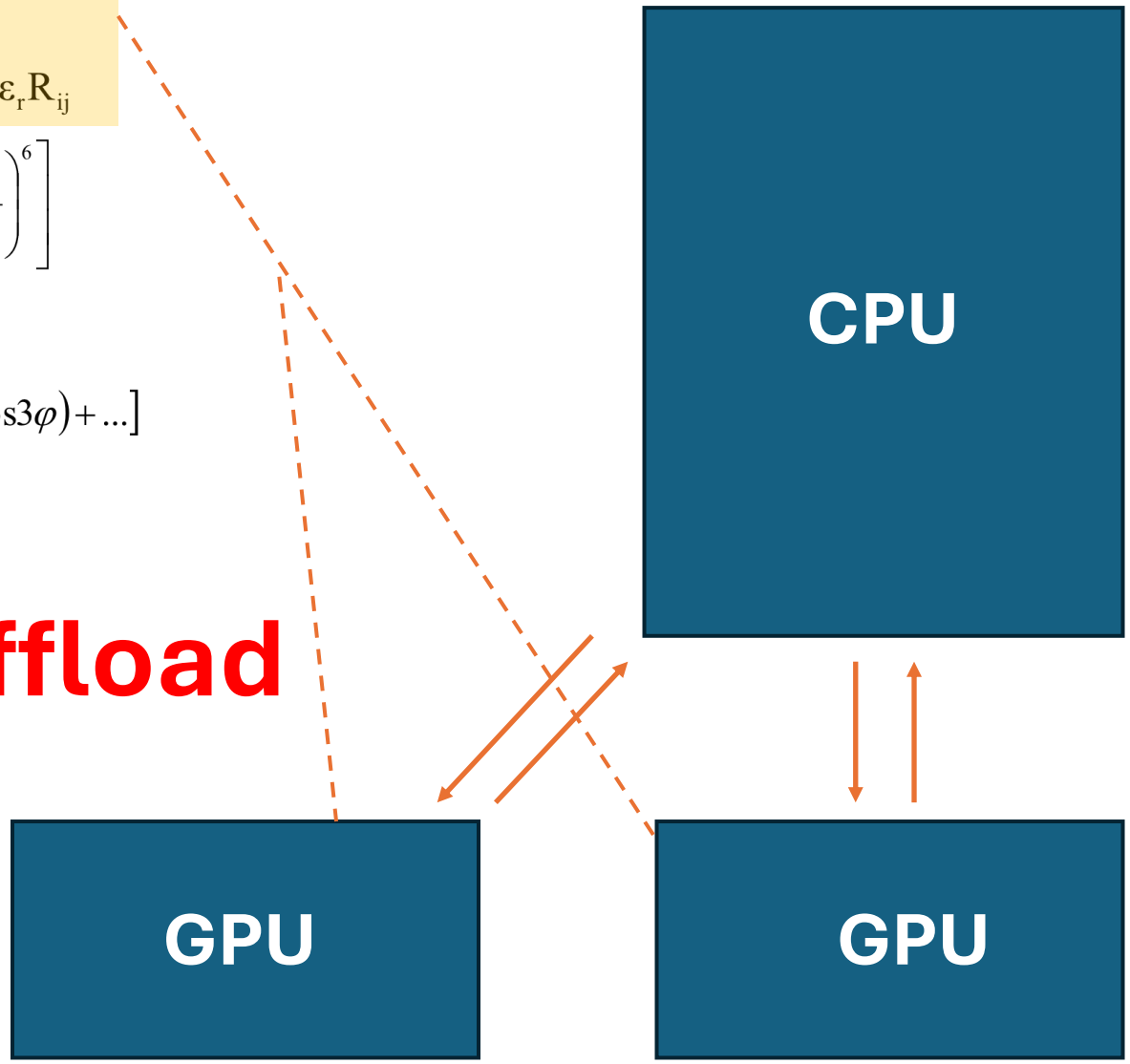
$$V_{vdW,ij} = \varepsilon_{IJ}\left[\left(\frac{R_{IJ}}{R_{ij}}\right)^{12} - 2\left(\frac{R_{IJ}}{R_{ij}}\right)^{6}\right]$$

$$V_{oop} = \frac{1}{2} k_{oop} \omega_{oop}^2$$

$$V_{tors,ijkl} = \frac{1}{2}\left[V_1(1+\cos\varphi) + V_2(1-\cos 2\varphi) + V_3(1+\cos 3\varphi) + \ldots\right]$$

$$V_{bend,ijk} = \frac{1}{2} k_{IJK}\left(\theta_{ij} - \theta^0{}_{IJK}\right)^2$$

$$V_{str,ij} = \frac{1}{2} k_{IJ}\left(l_{ij} - l^0{}_{IJ}\right)^2$$



Rafael C. Bernardi
rcbernardi@auburn.edu

What is a GPU-resident software

GPUs got faster

$$V = V_{str} + V_{bend} + V_{tors} + V_{oop} + V_{vdW} + V_{es}$$

$$V_{es,ij} = Q_i Q_j / \varepsilon_r R_{ij}$$

$$V_{vdW,ij} = \varepsilon_{IJ}\left[\left(\frac{R_{IJ}}{R_{ij}}\right)^{12} - 2\left(\frac{R_{IJ}}{R_{ij}}\right)^{6}\right]$$

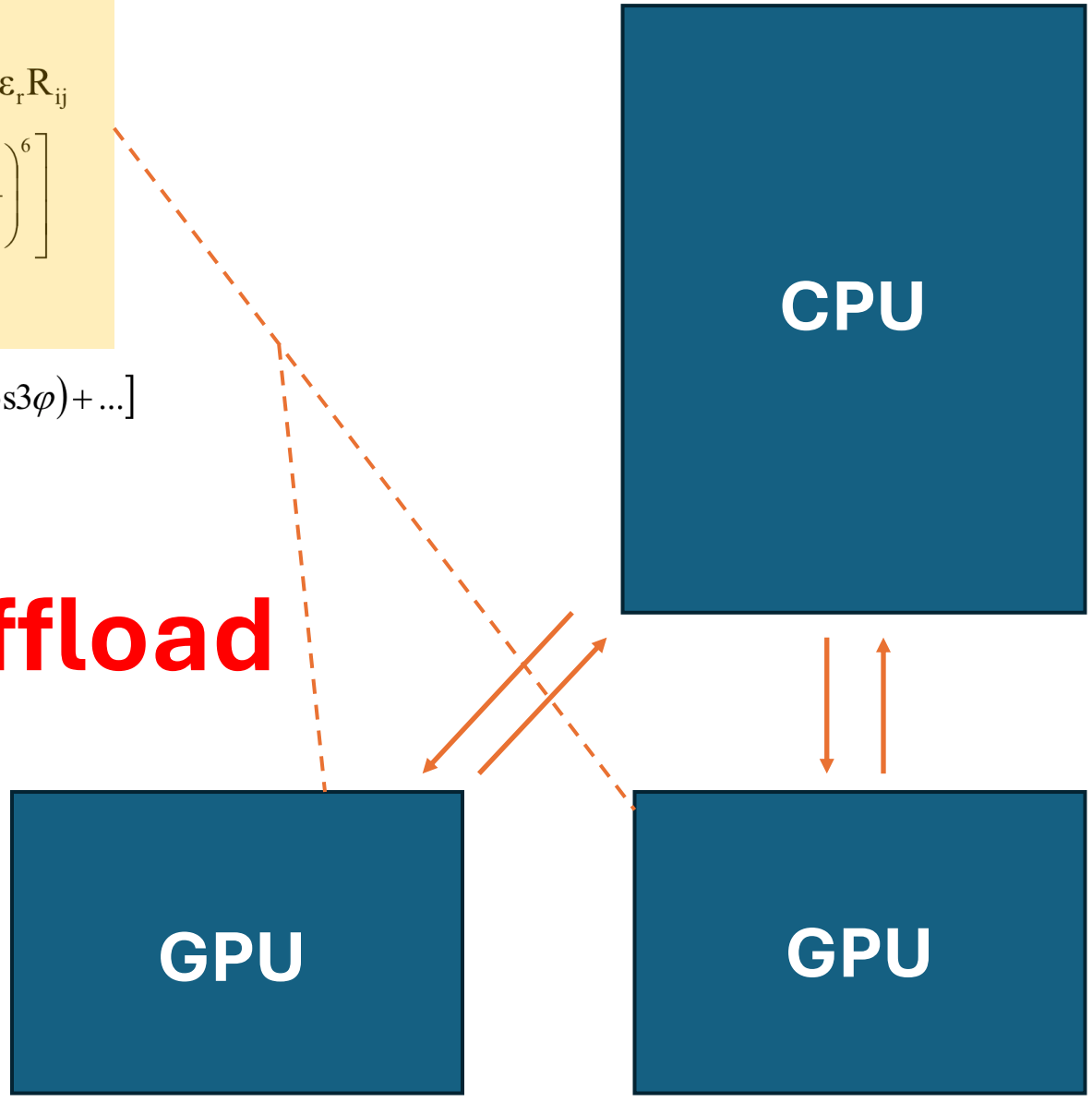
$$V_{oop} = \frac{1}{2} k_{oop} \omega_{oop}^2$$

$$V_{tors,ijkl} = \frac{1}{2}\left[V_1(1+\cos\varphi) + V_2(1-\cos 2\varphi) + V_3(1+\cos 3\varphi) + \ldots\right]$$

$$V_{bend,ijk} = \frac{1}{2} k_{IJK}\left(\theta_{ij} - \theta^0{}_{IJK}\right)^2$$

$$V_{str,ij} = \frac{1}{2} k_{IJ}\left(l_{ij} - l^0{}_{IJ}\right)^2$$

GPU offload



Rafael C. Bernardi
rcbernardi@auburn.edu

What is a GPU-resident software

$$V = V_{str} + V_{bend} + V_{tors} + V_{oop} + V_{vdW} + V_{es}$$

$$V_{es,ij} = Q_i Q_j / \varepsilon_r R_{ij}$$

$$V_{vdW,ij} = \varepsilon_{IJ}\left[\left(\frac{R_{IJ}}{R_{ij}}\right)^{12} - 2\left(\frac{R_{IJ}}{R_{ij}}\right)^{6}\right]$$

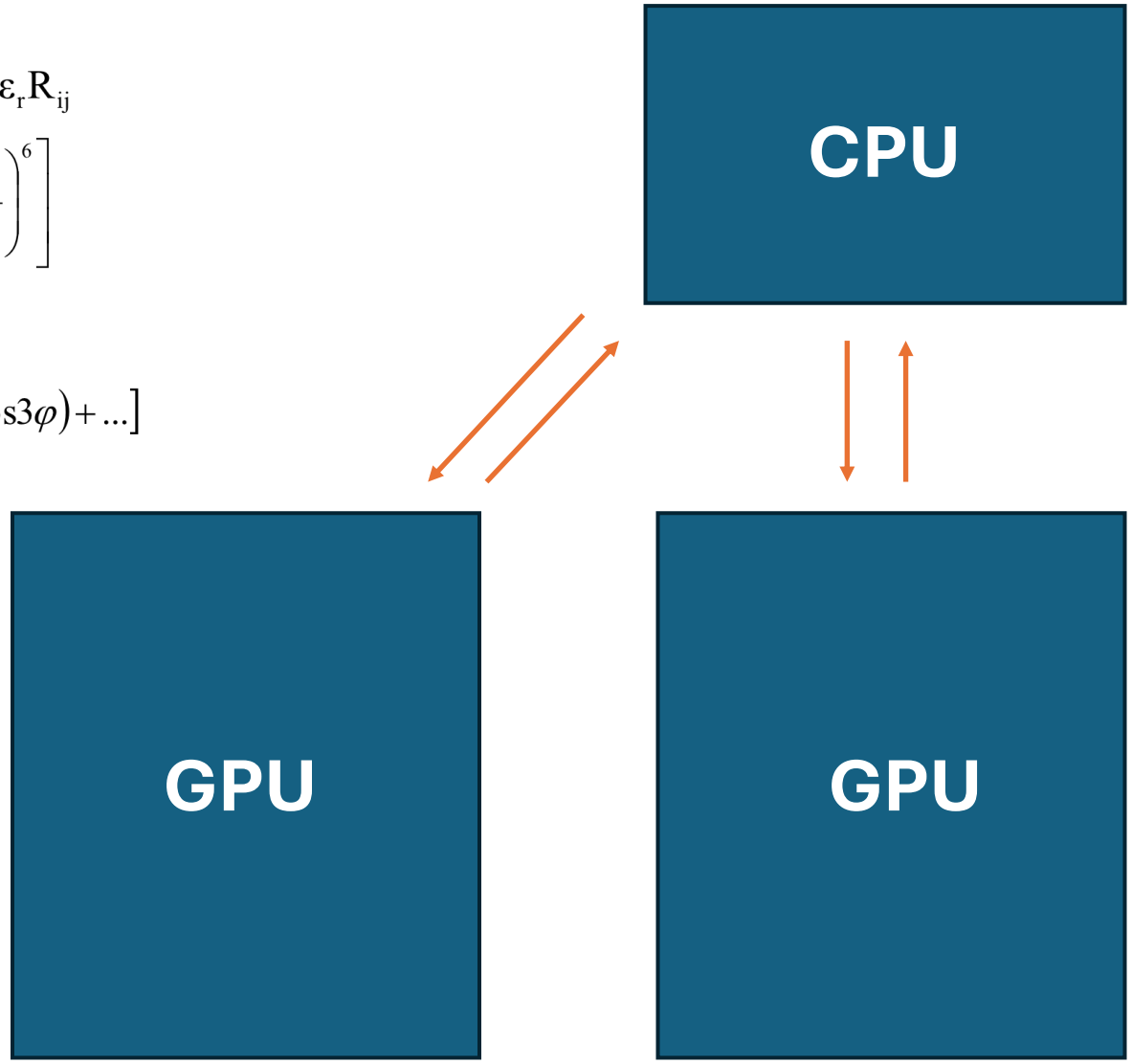
$$V_{oop} = \frac{1}{2} k_{oop} \omega_{oop}^2$$

$$V_{tors,ijkl} = \frac{1}{2}\left[V_1(1+\cos\varphi) + V_2(1-\cos 2\varphi) + V_3(1+\cos 3\varphi) + \ldots\right]$$

$$V_{bend,ijk} = \frac{1}{2} k_{IJK}\left(\theta_{ij} - \theta^0{}_{IJK}\right)^2$$

$$V_{str,ij} = \frac{1}{2} k_{IJ}\left(l_{ij} - l^0{}_{IJ}\right)^2$$

After NVIDIA Volta



Rafael C. Bernardi
rcbernardi@auburn.edu

What is a GPU-resident software

What is the solution?

What is a GPU-resident software

$$V = V_{str} + V_{bend} + V_{tors} + V_{oop} + V_{vdW} + V_{es}$$

$$V_{es,ij} = Q_i Q_j / \varepsilon_r R_{ij}$$

$$V_{vdW,ij} = \varepsilon_{IJ}\left[\left(\frac{R_{IJ}}{R_{ij}}\right)^{12} - 2\left(\frac{R_{IJ}}{R_{ij}}\right)^{6}\right]$$

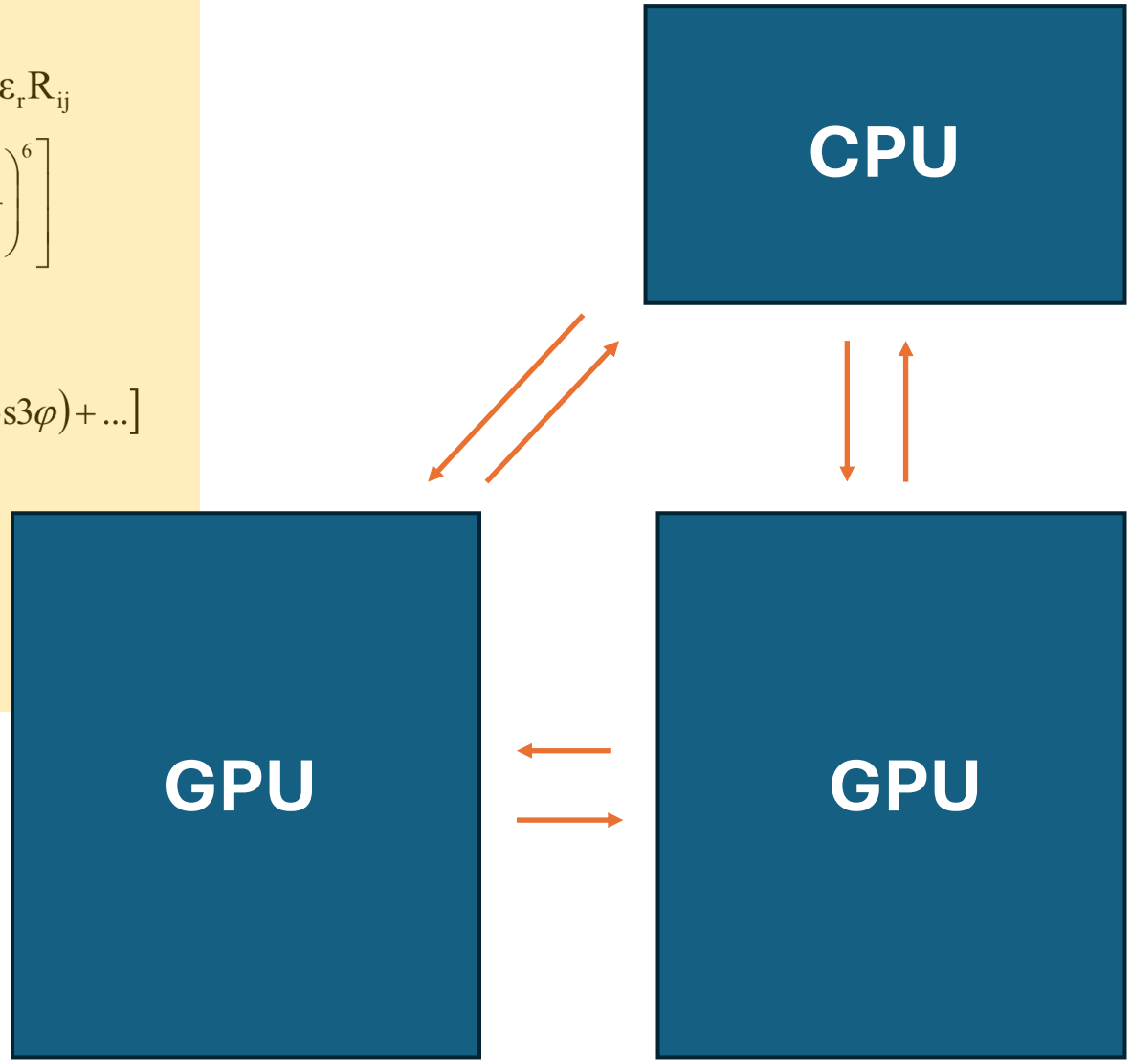
$$V_{oop} = \frac{1}{2} k_{oop} \omega_{oop}^2$$

$$V_{tors,ijkl} = \frac{1}{2}\left[V_1(1+\cos\varphi) + V_2(1-\cos 2\varphi) + V_3(1+\cos 3\varphi) + \ldots\right]$$

$$V_{bend,ijk} = \frac{1}{2} k_{IJK}\left(\theta_{ij} - \theta^0{}_{IJK}\right)^2$$

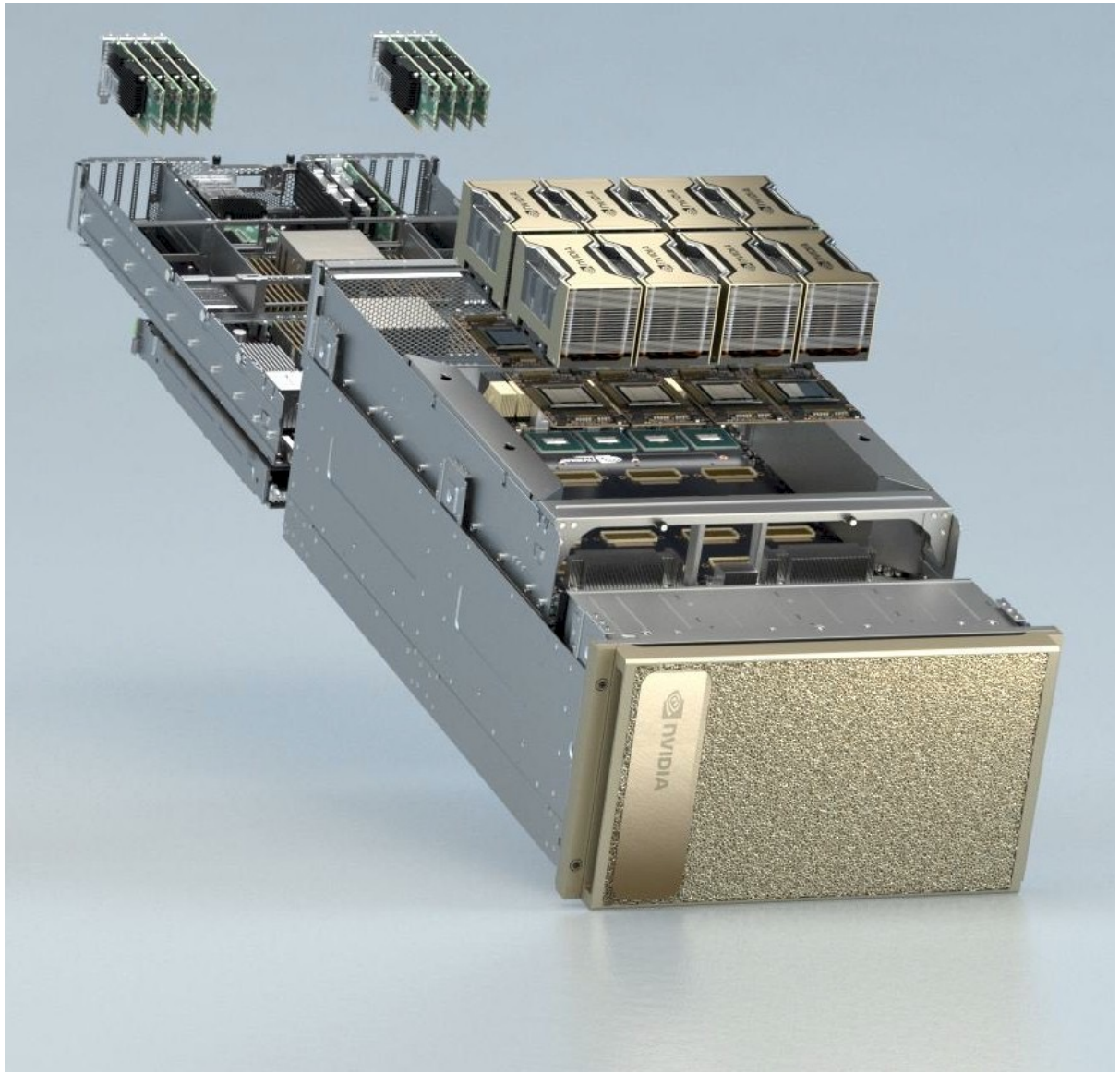
$$V_{str,ij} = \frac{1}{2} k_{IJ}\left(l_{ij} - l^0{}_{IJ}\right)^2$$

GPU-resident



Rafael C. Bernardi
rcbernardi@auburn.edu

What is a GPU-resident software



Tightly coupled GPUs

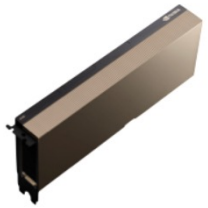
Rafael C. Bernardi
rcbernardi@auburn.edu

What is a GPU-resident software

What are limitations?

How fast is NAMD 3.0

Single GPU performance improvements



A100

NVE simulation (constant energy):

- DHFR: AMBER-like force field (9 Å cutoff), HMR with 4 fs time step, PME, rigid bond constraints, "margin" 2 Å, two-away-Z.
- ApoA1: CHARMM force field (12 Å cutoff), multiple time stepping with 2 fs time step and 4 fs PME, rigid bond constraints.
- STMV: CHARMM force field (12 Å cutoff), multiple time stepping with 2 fs time step and 4 fs PME, rigid bond constraints.
- Spike ACE-2: CHARMM force field (12 Å cutoff), multiple time stepping with 2 fs time step and 4 fs PME, rigid bond constraints.
- Each measurement calculates the average ns/day running dynamics for 3 minutes of wall clock time.

Platform details:

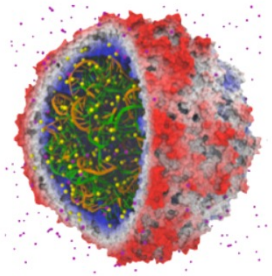
- 1 GPU and CPU cores from HGX-A100 (4x A100-SXM4-40GB, NVLink, 2x AMD EPYC 74F3 (Milan) 24-core processor)
- GPU-offload performs best for each system using **all** 48 cores.
- GPU-resident: DHFR — 2 cores, ApoA1 — 4 cores, STMV — 8 cores, Spike ACE-2 — 8 cores.

	GPU-resident ns/day	GPU-offload ns/day	speedup
DHFR (23.6k)	1174.0	330.4	3.55x
ApoA1 (92.2k)	190.4	63.88	2.98x
STMV (1.06M)	16.64	7.547	2.20x
Spike ACE-2 (8.56M)	1.875	0.7711	2.43x

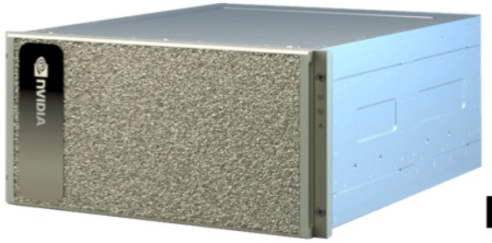
Rafael C. Bernardi
rcbernardi@auburn.edu

How fast is NAMD 3.0

Single-node multi-GPU scaling of STMV



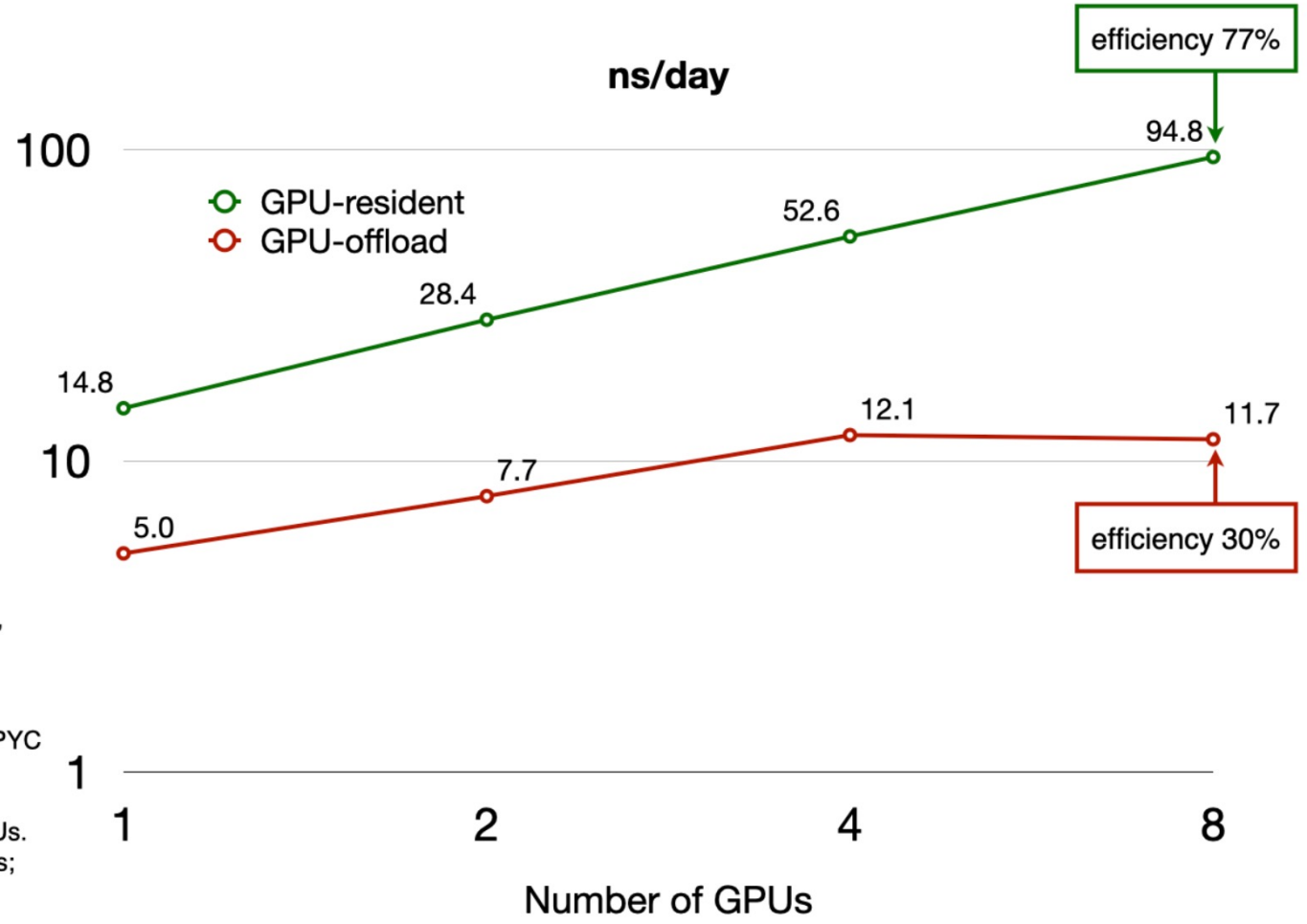
STMV
1.06M atoms



NVIDIA
DGX-A100

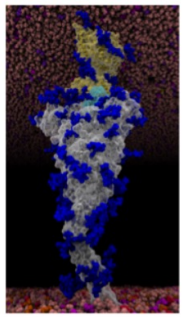
NVE simulation (constant energy):
- CHARMM force field (12 Å cutoff), rigid bond constraints, multiple time stepping with 2 fs time step and 4 fs PME.

Platform details:
- HGX-A100 (8x A100-SXM4-40GB, NVSwitch, 2x AMD EPYC 7742 (Rome) 64-core processor)
- GPU-resident uses 8 cores per non-PME GPU and sets PME-PEs to 7 for 2 GPUs, 5 for 4 GPUs, and 1 for 8 GPUs.
- GPU-offload uses 16 cores per GPU for 1, 2, 4, GPU runs; uses 8 cores per GPU for 8 GPU run.
- (Unable to run GPU-offload using all 128 cores.)

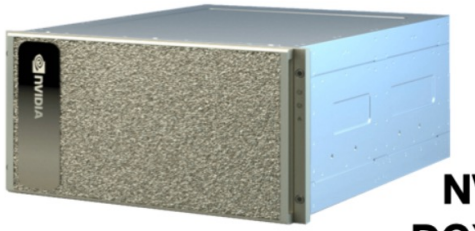


Rafael C. Bernardi
rcbernardi@auburn.edu

How fast is NAMD 3.0



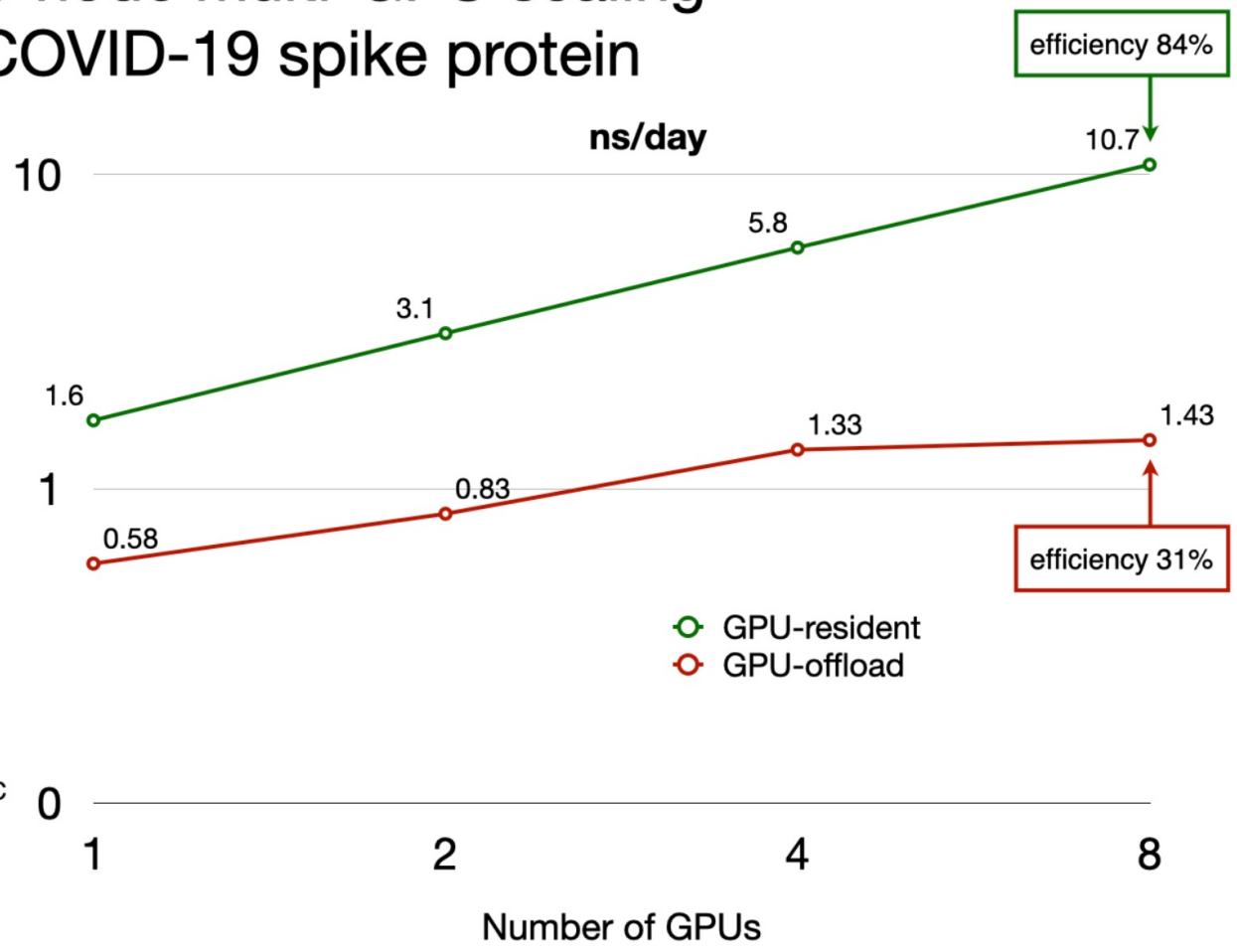
Spike-ACE2
8.56M atoms



NVIDIA
DGX-A100

NVE simulation (constant energy):
- CHARMM force field (12 Å cutoff), rigid bond constraints, multiple time stepping with 2 fs time step and 4 fs PME.

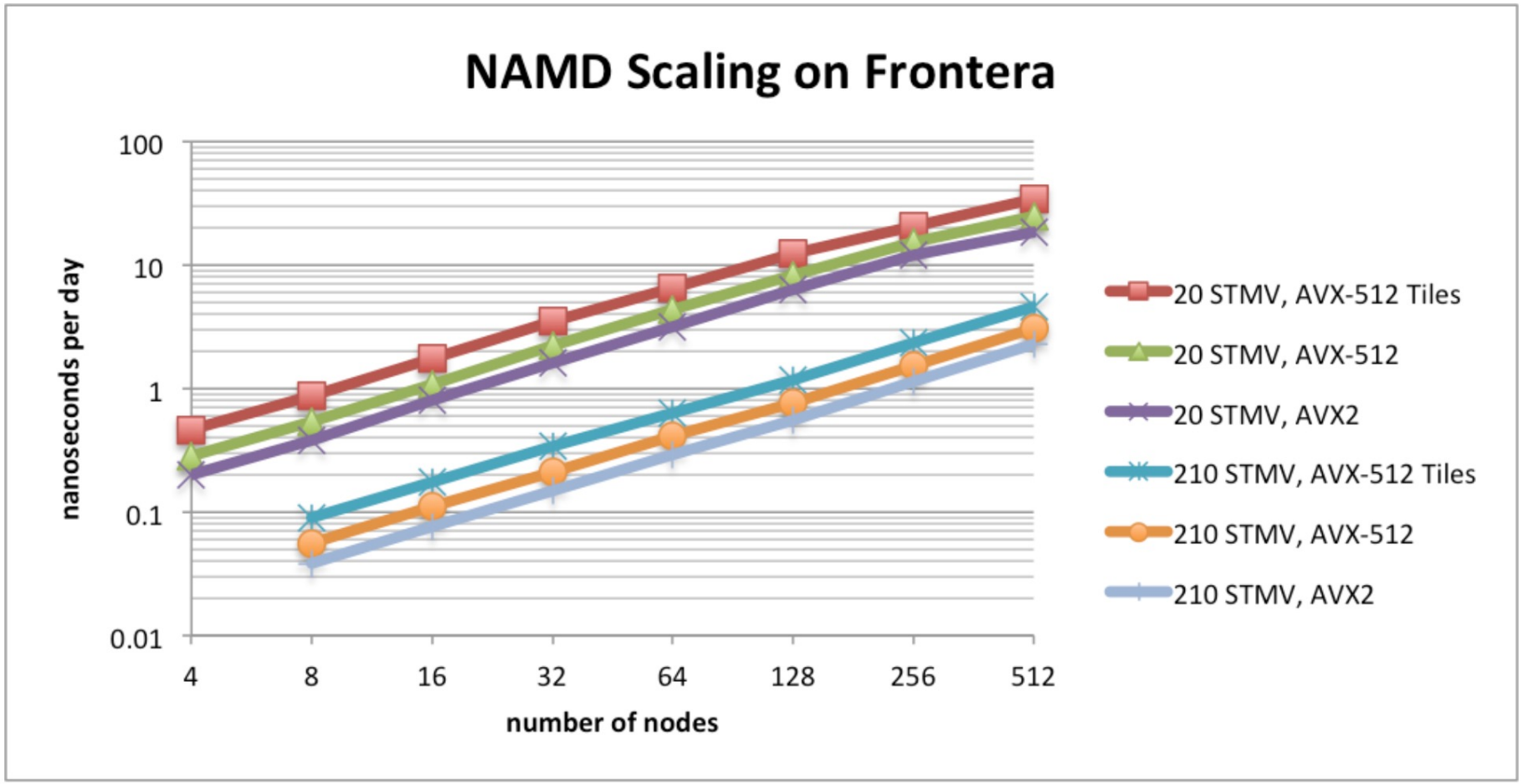
Platform details:
- HGX-A100 (8x A100-SXM4-40GB, NVSwitch, 2x AMD EPYC 7742 (Rome) 64-core processor)
- GPU-resident uses 8 cores per non-PME GPU and sets PME-PEs to 7 for 2 GPUs, 5 for 4 GPUs, and 1 for 8 GPUs.
- GPU-offload uses 16 cores per GPU for 1, 2, 4, GPU runs; uses 8 cores per GPU for 8 GPU run.
- (Unable to run GPU-offload using all 128 cores.)



Rafael C. Bernardi
rcbernardi@auburn.edu

How scalable is NAMD 3.0

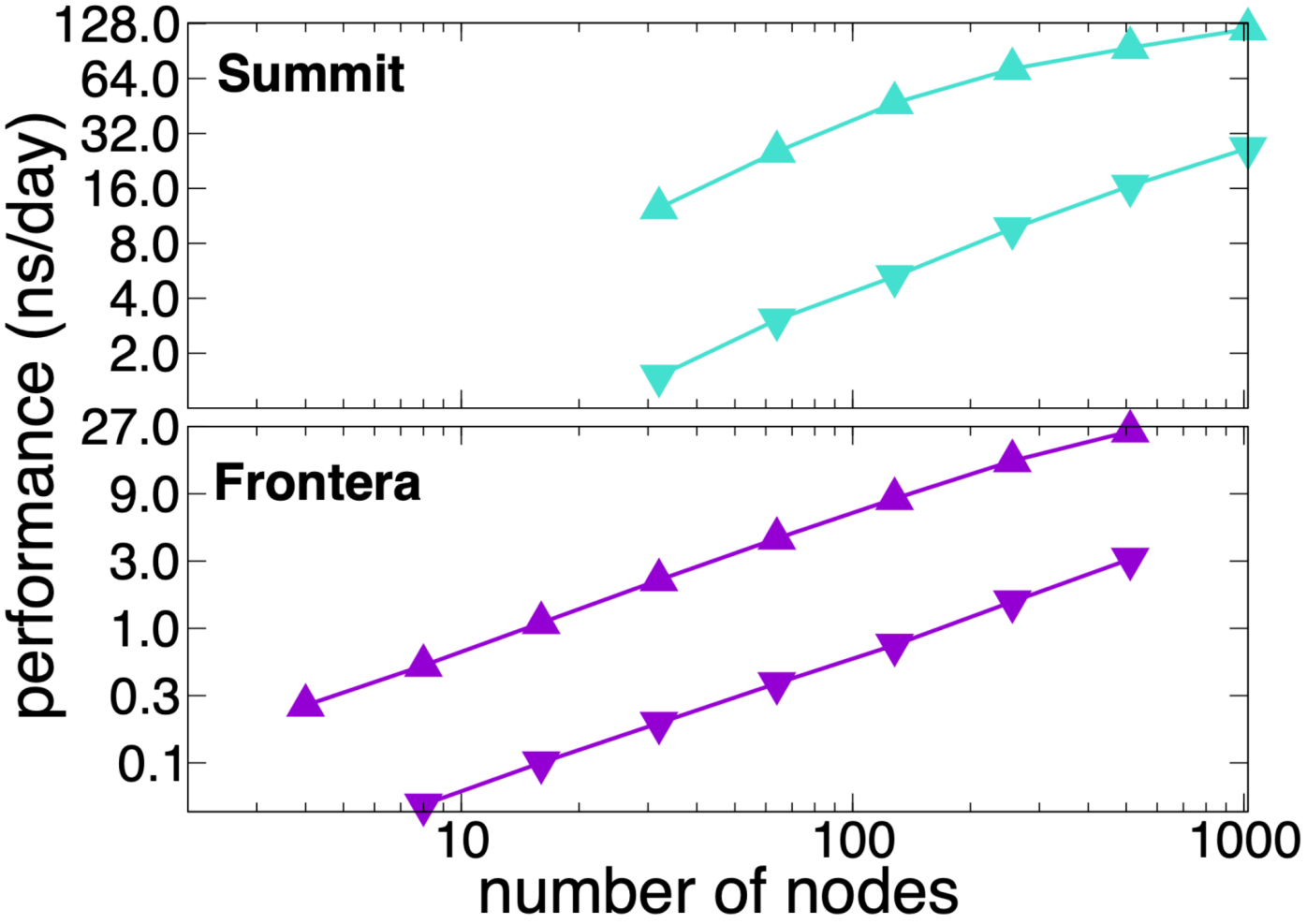
Benchmarking STMV matrix systems (NPT, 2fs timestep, 12A cutoff + PME every 3 steps) on TACC Frontera



Rafael C. Bernardi
rcbernardi@auburn.edu

How scalable is NAMD 3.0

Benchmarking STMV matrix systems (NPT, 2fs timestep, 12A cutoff + PME every 3 steps) on Summit and Frontera



NAMD 3.0 still supports the multi-node scaling available in release 2.14. The plots show performance results on OLCF Summit and TACC Frontera scaling the synthetic benchmark systems created by tiling the periodic STMV (1M atoms) cell in arrays of 5x2x2 (21M atoms, upward pointing triangles) and 7x6x5 (224M atoms, downward pointing triangles). The simulations use Langevin piston and Langevin damping to impose pressure and temperature control (NPT) with a 2fs time step, 12A cutoff, and PME every 3 steps. Multi-node scaling for GPU-based computers like Summit is supported by NAMD's GPU-offload mode.

Rafael C. Bernardi
rcbernardi@auburn.edu

Thank you

Main development team

David Hardy

Eric Bohm

Haochuan Chen

Jim Phillips

Julio Maia

Many others

Rafael C. Bernardi
rcbernardi@auburn.edu

NIH Resource for Macromolecular Modeling and Visualization (NIH R24-GM145965)