Global Computing Lab

Towards Large-Scale Molecular Dynamics Simulations on Graphics Processors

Joe Davis

Michela Taufer

Sandeep Patel

Outline

- Introduction
- Brief overview on GPUs and GPU programming paradigm
- Running MD simulations on GPUs
- Case study I: Solvent Simulation
  - Water model
  - Performance
  - Result accuracy
- Case Study 2: Ionic solution
  - Ion Model
  - Performance
- Pitfalls
- Conclusions

Introduction (I)

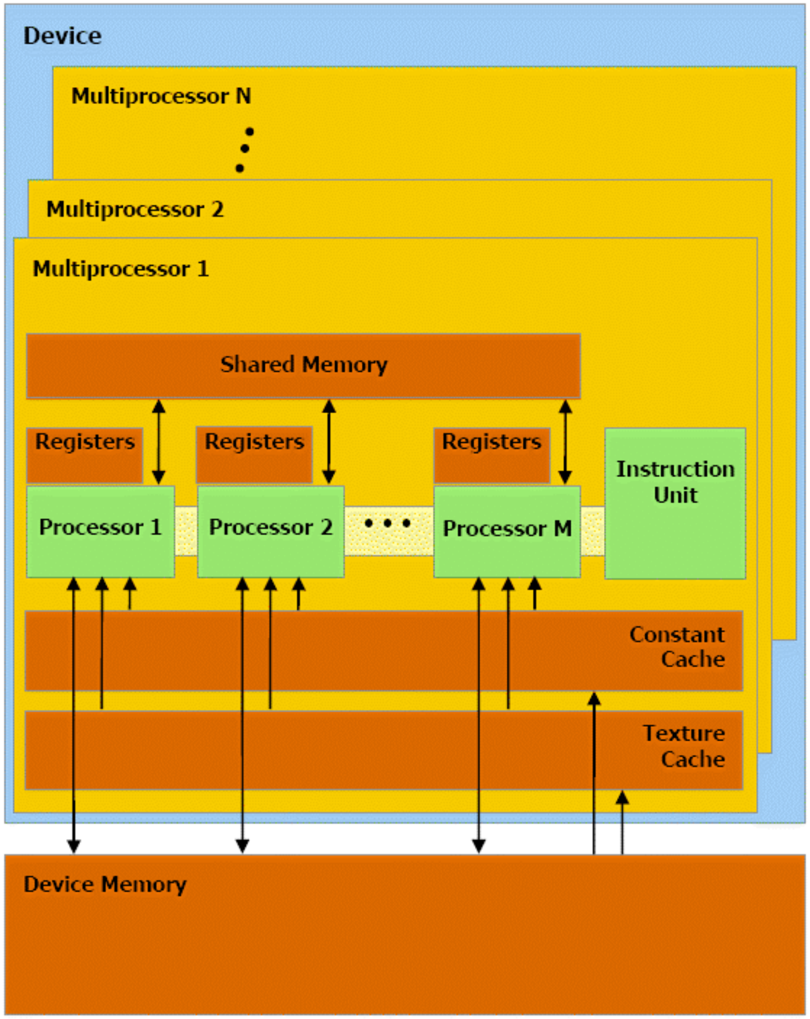
- Graphics Processing Units (GPUs) have been extensively used in graphics intensive applications
  - Development driven by economy, e.g., video game industry, motion picture
- The *inherent parallelization of GPUs* makes them suitable for scientific applications
- Recent exploration of potential of GPUs for *mathematics* and *scientific, and clinical* computing
  - Medical diagnostics:
    - GPUs coupled to MRI Hardware (Stone et al. Proc. of 2007 Computing Frontiers conference, 7-9 May, 2008)
  - Molecular modeling:
    - Electrostatic Potential Calculation (Stone et al. J. Comp. Chem. 28, #16, pp. 2618-2640)
    - Ion Placement (Stone et al. J. Comp. Chem. 28, #16, pp. 2618-2640)
    - Van der Waals Fluids / Polymers (Anderson et al. J. Comput. Physics 2008)

Introduction (II)

- Special purpose hardware: *specific types* of calculations
  - Protein Explorer systems and its LSI 'MDGRAPE-3 chip' (Taiji et al. in Proc. of 2003 ACM/IEEE Supercomputing Conference, 15-21 Nov. 2003)
  - Anton and its 12 identical MD-specific ASICs (Shaw et al. in Proc. of the 34th Annual International Symposium on Computer Architecture, *9-13 June, 2007)*
- General Purpose GPUs (or GPGPUs): *cost effective* and *readily available* in recent workstations
  - GeForce FX5600
    - 1.5GBytes memory
    - Cost $2,795

  - GeForce 9800 GX2
    - Dual GPU-based graphics card
    - 512MBytes memory per GPU
    - Cost $665

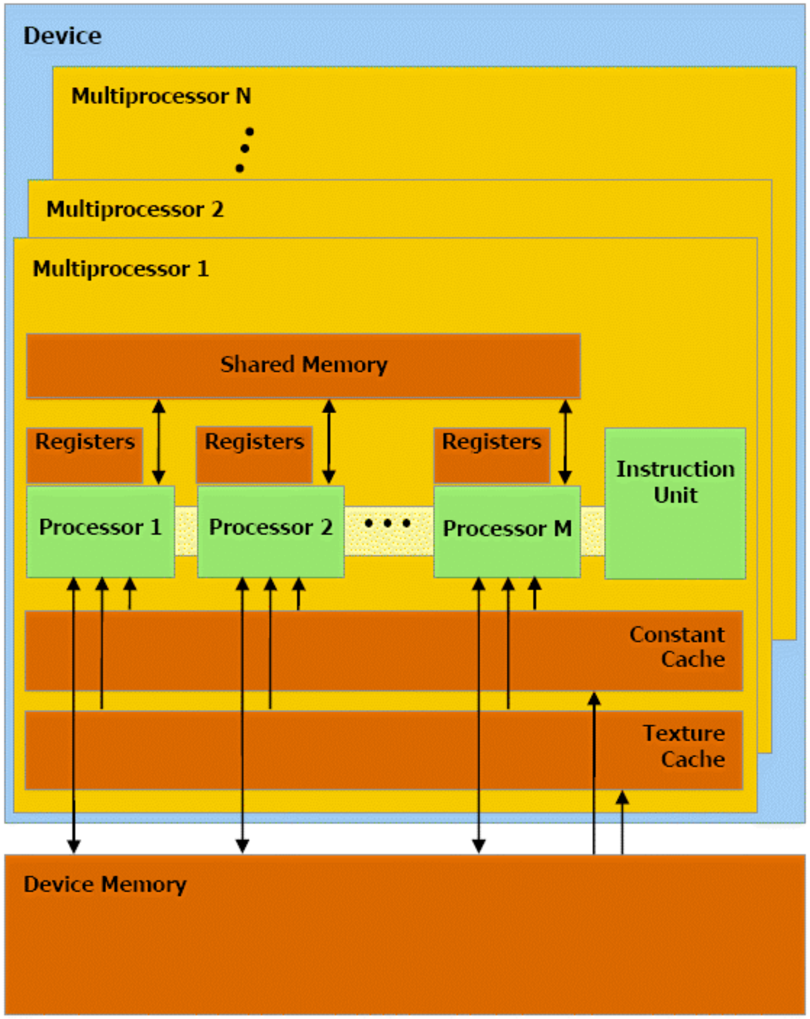
GPU Overview (I)



From CUDA Programming Guide, NVIDIA

- NVIDIA GeForce 8 Series:
  - 16 Streaming Multiprocessor (1-N)
  - 8 Scalar Processors/SM (1-M)
  - 16, 8-way SIMD cores = 128 PEs
- Massively parallel multithreaded
  - Up to 12,288 active threads handled by thread execution manager
- Actual application performance
  - Molecular dynamics -VMD ion placement: 290 GFLOPS
  - FFT: 52 benchFFT GFLOPS

GPU Overview (II)

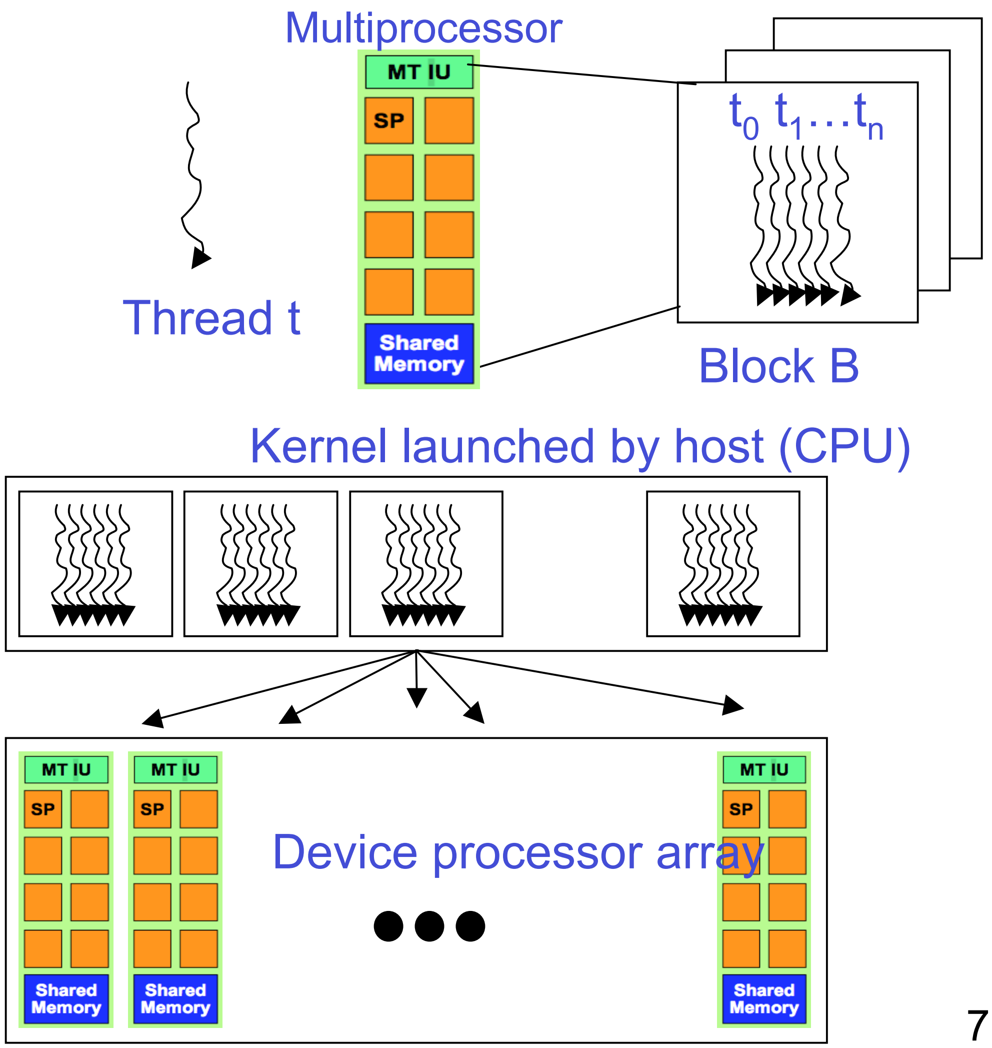


From CUDA Programming Guide, NVIDIA

- Memory types:
  - Read/write per thread
    - Registers
    - Local memory
  - Read/write per block
    - Shared memory
  - Read/write per grid
    - Global memory
  - Read-only per grid
    - Constant memory
    - Texture memory
- Communication among devices and with CPU
  - Through PCI bus

Programming Paradigm

- Program in C:
  - Serial program executed on CPU
  - Parallel kernels executed on GPU
- Parallel kernels composed of many threads
- Threads are grouped into thread blocks
  - Threads in the same block can cooperate
  - Thread block = a (data) parallel task (SIMD)
  - Same entry point but can execute any code - conditions are allowed in block threads
- Different blocks are independent
  - Several blocks = task parallelism

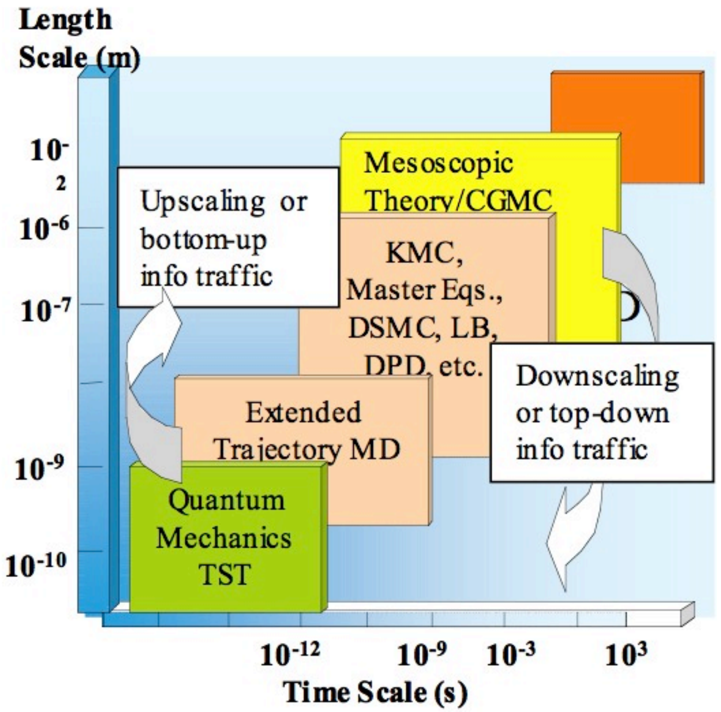


Programming GPUs

- Past: APIs originally through graphics interfaces e.g., OpenGL
  - Not easy to use for general usage: cast *computation* in terms of *graphics operations*:
    - Draw the calculation
    - Interpret “image” post-calculation
- Present: NVDIA CUDA (Compute Unified Device Architecture) language/library
  - Easy to use: CUDA provides minimal set of extensions necessary to expose power of GPGPUs
  - Includes C-compiler and development tools
- CUDA optimization strategy:
  - Maximize independent parallelism
  - Maximize arithmetic intensive computation
  - Take advantage of on-chip per-block shared memory
  - Do computation on the GPUs and avoid data transfer

MD on GPUs

- Why MD on GPU?
  - Non-bond expand scales of time and physical dimension (system complexity)
  - All-atom resolution (micro to milliseconds)
  - Course-graining (seconds)
  - Continuum physics with molecular detail?



- MD on GPU: Non-bond interactions (pair interactions)
  - Non-bond list is generated by checking all pair distances against the cut-off in parallel (efficient tiling approach)
  - A thread iterates through the non-bond list for a single atom and accumulates the non-bonded interactions

Water Simulations

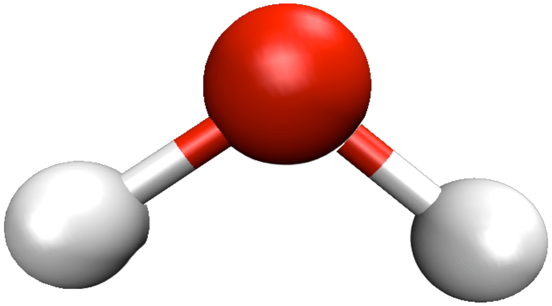
- Water simulations on GPU vs. on CPU
- CUDA code emulating the CHARMM molecular modeling package (Brooks, B. R. et al, *J. Comput. Chem.*, 1983, 4: 187)
- Reference simulation of CHARMM on Beowulf cluster
  - Intel Xeon 5150 2.66 GHz (Woodcrest)
- NVIDIA GPUs
  - Single precision Quadro FX 5600 (1.5GB memory)
  - Single precision GeForce 9800 GX2 (dual GPUs per card, 500MB memory)
  - Double precision GTX 280

Water Model

- Flexible Water SPC/Fw (Wu *et al*, J. Chem. Phys., 2006)
  - Intra-molecular potential:

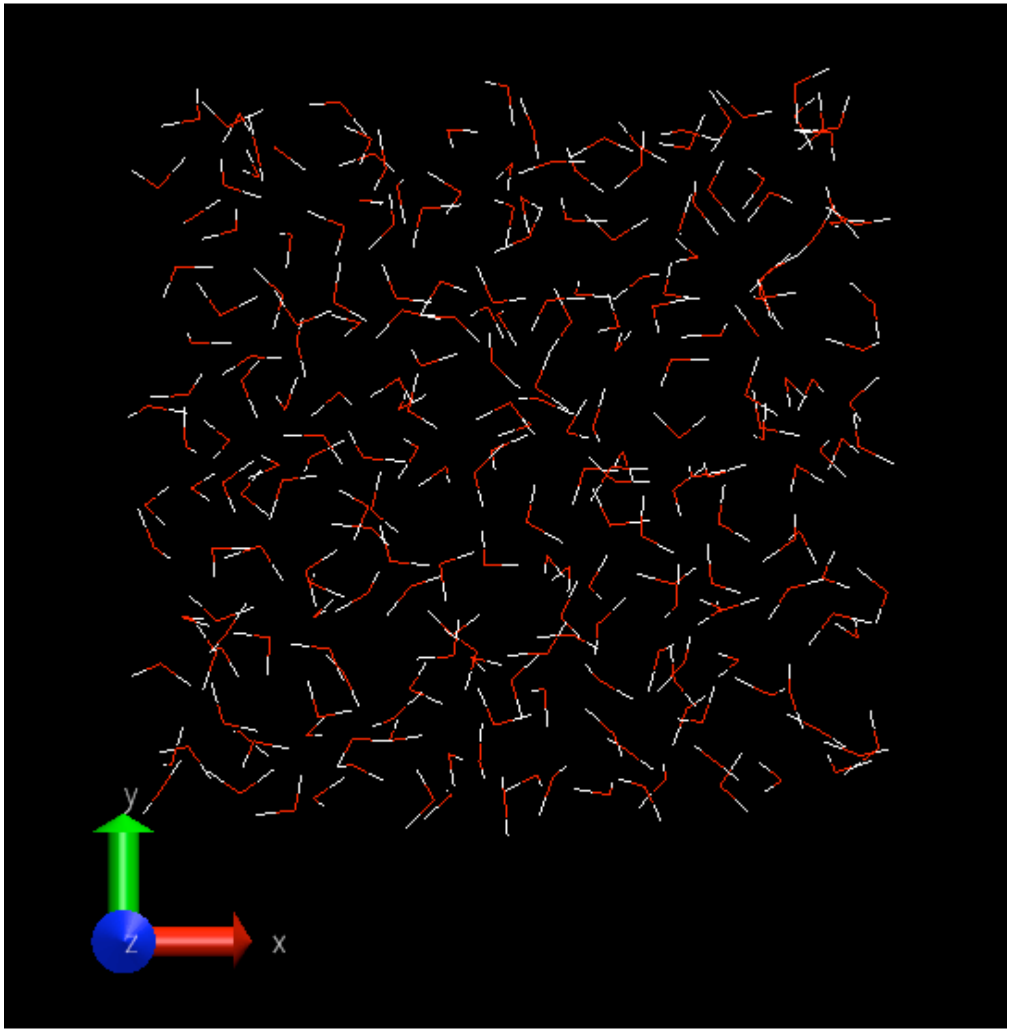
$$V^{\text{intra}} = \frac{k_b}{2}[(r_{\mathrm{OH}_1} - r^0_{\mathrm{OH}})^2 + (r_{\mathrm{OH}_2} - r^0_{\mathrm{OH}})^2] + \frac{k_a}{2}(\theta_{\angle\mathrm{HOH}} - \theta^0_{\angle\mathrm{HOH}})^2$$

    - Computed on GPU using lists (bonds/angles lists)
  - Non-bonded potential:
    - Lennard-Jones potential
    - Shifted-force electrostatics with cut-off only (no Ewald)
    - Computed on GPU using a list-based evaluation



System Parameters

- NVE
- Pre-equilibrated box
- PBC: Cubic
- Density = 1.012 g/mL
- Δt = 1 fs
- Integrator:
  - Verlet on GPU
  - Orig. Verlet with CHARMM on CPU

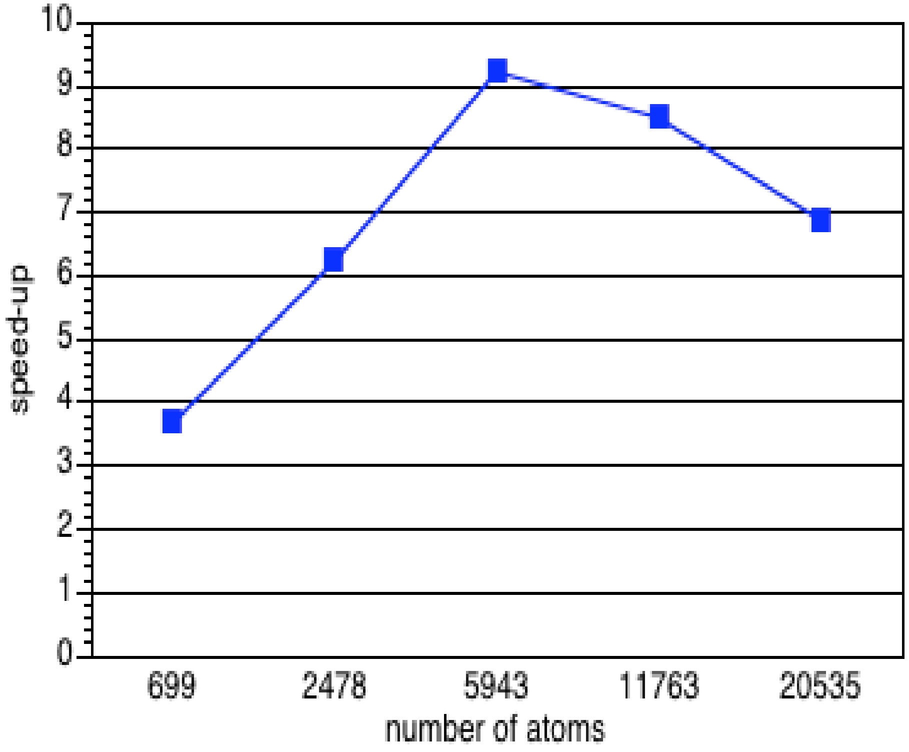


Performance

Performance metrics: number of MD steps in one second
GPU: single precision GeForce 9800 GX2 (dual GPUs per card, 500MB memory)

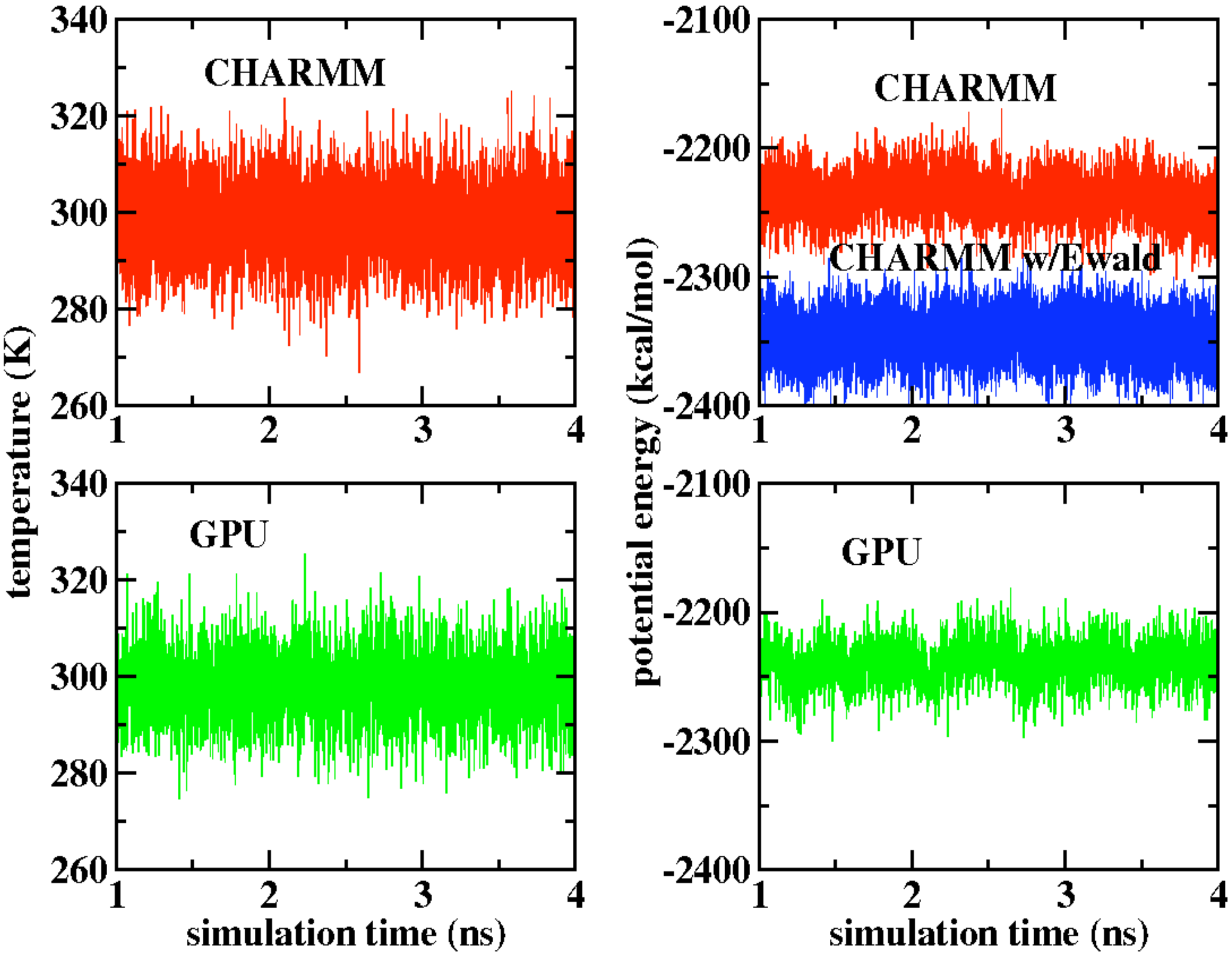
# of atoms	CHARMM (MD steps/sec)	GPU (MD steps/sec)
699	165.34	609.09
2478	43.49	271.2
5943	17.25	159.12
11763	8.52	72.32
20535	4.73	32.47

(data from 100,000 MD steps)



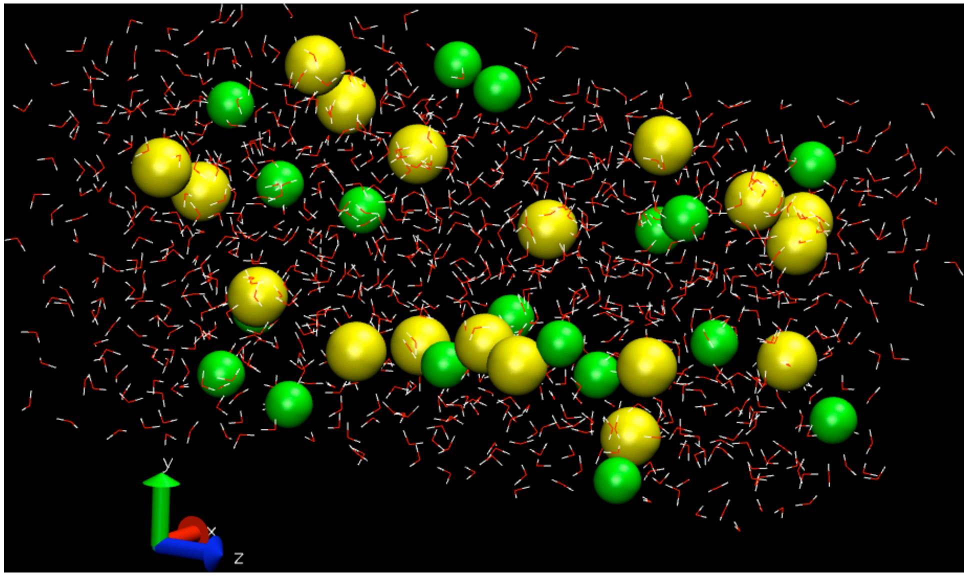
In average GPU is ~7x faster on average!

CHARMM on CPU and GPU



Ionic Solutions

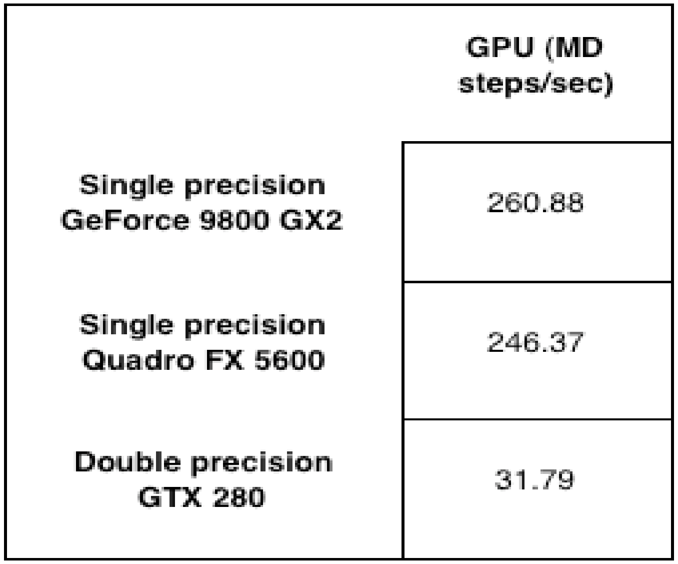
- Nonpolarizable ions with SPC/Fw water
- Liquid-vapor interface
- Ion model:
  - Electrostatic and van der Waals only
  - CHARMM parameters modified for more accurate interaction energies[1]



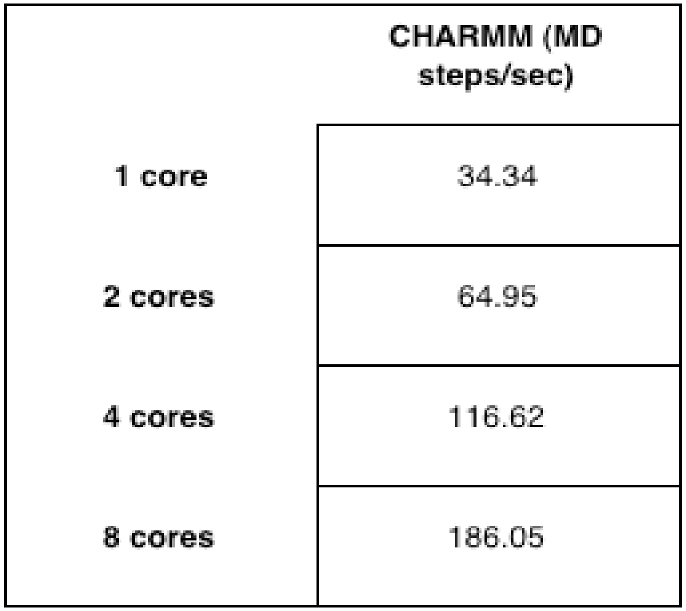
[1] Lamoureux, G. and Roux, B., J. *Phys. Chem. B*, 2006, 110: 3308.

Performance

- Performance metrics: number of MD steps in one second
- NVIDIA GPUs:
  - Single precision GeForce 9800 GX2 (dual GPUs per card, 500MB memory)
  - Single precision Quadro FX 5600 (1.5GB memory)
  - Double precision GTX 280



	GPU (MD steps/sec)
Single precision GeForce 9800 GX2	260.88
Single precision Quadro FX 5600	246.37
Double precision GTX 280	31.79



	CHARMM (MD steps/sec)
1 core	34.34
2 cores	64.95
4 cores	116.62
8 cores	186.05

Pitfalls

- Single or double precision?
  - G8x GPU FP is 32-bit, newer T10P GPU is 64-bit
  - Some 32-bit operations are not IEEE compliant
  - 64-bit arithmetic is more accurate, but more costly
- Fortran compilation?
  - Fortran compiler is forthcoming
  - Our team will be part of the alpha testers
- Code optimization is everything?
  - Targets: *memory access*, efficient list building/updating, loops, conditional, data structure, etc.
  - The optimization of our code is a work in progress
- Limited number of GPUs per card?
  - 870: board (1 GPU)
  - D870: deskside unit (2 GPUs)
  - S870: 1u server unit (4 GPUs)
  - We have workstations with two dual-GPU cards ready for testing

Related Work

- Yang et al[1]
  - Proof of concept
  - Limited by graphics-specific programming interface
- Stone et al[2]
  - Moved nonbond force calculation to GPU
  - Focus mostly on modeling applications
- Anderson et al[3]
  - MD running entirely on GPU (HOOMD)
  - Neighbor list implementation
- Van Meel et al[4]
  - Very similar to Anderson's study

[1] Yang, J. et al, *J. Comput. Phys.*, 2007, 221: 799.
[2] Stone, J. E. et al, *J. Comput. Chem.,* 2007, 28: 2618.
[3] Anderson, J. A. et al, *J. Comput. Phys.*, 2008, 227: 5342.
[4] Van Meel, J. A. et al, *Mol. Sim.*, 2008, 34: 259.

Conclusions

- Current achievements:
  - Implementation of a local version of MD code on current generation of GPUs
  - Straightforward, naive implementation
  - Promising results
- Work in progress:
  - Optimization and tuning of performance
  - Expand MD options (additional potentials, PME)
- Final goals:
  - Effective compilation of CHARMM on GPU
  - Study of large solvent systems for long simulation times, up to 100ns, with CHARMM

Acknowledgements

Collaborators:

- Adnan Ozsoy (University of Delaware)
- David Hoff, Sumit Gupta, Scott LeGrand (NVIDIA)
- Joshua Anderson (Iowa State University)

Sponsors:

- NVIDIA Professor Partnership program
- NSF OCI #0802650
- University of Delaware