NAMD: GPU-Accelerated Petascale Molecular Dynamics Simulations on Titan and Blue Waters

James Phillips
Beckman Institute, University of Illinois
http://www.ks.uiuc.edu/Research/namd/
NAMD: Scalable Molecular Dynamics

2002 Gordon Bell Award

60,000 Users, 3000 Citations

Blue Waters Target Application

ATP synthase

PSC Lemieux

Computational Biophysics Summer School

GPU Acceleration

Illinois Petascale Computing Facility

NVIDIA Tesla

NCSA Lincoln

2013 GPU Programming for Molecular Modeling Workshop

Biomedical Technology Research Center for Macromolecular Modeling and Bioinformatics

http://www.ks.uiuc.edu/
Larger machines enable larger simulations
NIH BTRC for Macromolecular Modeling and Bioinformatics 1990-2017

Beckman Institute
University of Illinois at Urbana-Champaign
Collaborative Driving Projects

1. Ribosome
   R. Beckmann (U. Munich)
   J. Frank (Columbia U.)
   T. Ha (UIUC)
   K. Fredrick (Ohio state U.)
   R. Gonzalez (Columbia U.)

2. Blood Coagulation Factors
   J. Morrissey (UIUC)
   S. Sligar (UIUC)
   C. Rienstra (UIUC)
   G. Gilbert (Harvard)

3. Whole Cell Behavior
   W. Baumeister (MPI Biochem.)
   J. Xiao (Johns Hopkins U.)
   C.N. Hunter (U. Sheffield)
   N. Price (U. Washington)

4. Biosensors
   R. Bashir (UIUC)
   J. Gundlach (U. Washington)
   G. Timp (U. Notre Dame)
   M. Wanunu (Northeastern U.)
   L. Liu (UIUC)

5. Viral Infection Process
   J. Hogle (Harvard U.)
   P. Ortoleva (Indiana U.)
   A. Gronenborn (U. Pittsburgh)

6. Integrin
   T. Ha (UIUC)
   T. Springer (Harvard U.)

7. Membrane Transporters
   H. Mchaourab (Vanderbilt U.)
   R. Nakamoto (U. Virginia)
   D.-N. Wang (New York U.)
   H. Weinstein (Cornell U.)
Physics of in vivo Molecular Systems

Biomolecular interactions span many orders of magnitude in space and time.

Center software provides multi-scale computational modeling.

femtoseconds
Ångstrom

Potential-based
all atom
Configuration sampling

Potential-based
coarse grained

Nonspecific interactions

Probability-based

BD: Brownian Dynamics

RDME: Reaction-diffusion master equation
Parallel Programming Lab
University of Illinois at Urbana-Champaign

http://charm.cs.illinois.edu/

2013 GPU Programming for Molecular Modeling Workshop
Biomedical Technology Research Center for Macromolecular Modeling and Bioinformatics
http://www.ks.uiuc.edu/
Develop abstractions in context of full-scale applications

Quantum Chemistry (QM/MM)

Computational Cosmology

NAMD: Molecular Dynamics

STM virus simulation

Parallel Objects, Adaptive Runtime System

Libraries and Tools

Rocket Simulation

Dendritic Growth

Space-time meshes

Protein Folding

Crack Propagation
Charm++ Used by NAMD

• Parallel C++ with *data driven* objects.
• Asynchronous method invocation.
• Prioritized scheduling of messages/execution.
• Measurement-based load balancing.
• Portable messaging layer.
NAMD Hybrid Decomposition


- Spatially decompose data and communication.
- Separate but related work decomposition.
- “Compute objects” facilitate iterative, measurement-based load balancing system.
NAMD Overlapping Execution

Phillips et al., SC2002.

Objects are assigned to processors and queued as data arrives.

2013 GPU Programming for Molecular Modeling Workshop

Biomedical Technology Research Center for Macromolecular Modeling and Bioinformatics

http://www.ks.uiuc.edu/

Beckman Institute, UIUC
Overlapping GPU and CPU with Communication

GPU
- Remote Force
- Local Force

CPU
- Remote
- Local
- Update

Other Nodes/Processes

One Timestep

Phillips et al., SC2008

2013 GPU Programming for Molecular Modeling Workshop
Beckman Institute, UIUC
“Remote Forces”

- Forces on atoms in a local patch are “local”
- Forces on atoms in a remote patch are “remote”
- Calculate remote forces first to overlap force communication with local force calculation
- Not enough work to overlap with position communication

Work done by one processor

Phillips et al., SC2008
Actual Timelines from NAMD
Generated using Charm++ tool “Projections” http://charm.cs.uiuc.edu/
NAMD 2.9 on Keeneland ID
(12 Intel cores and 3 Telsa M2070 GPUs per node)

STMV (1M atoms) s/step

GPUs = 8x nodes

2013 GPU Programming for Molecular Modeling Workshop
Biomedical Technology Research Center for Macromolecular Modeling and Bioinformatics
http://www.ks.uiuc.edu/
Beckman Institute, UIUC
Know Your (Cray) Supercomputers

**Titan**
- Funded by DOE
- Allocated by INCITE, etc.
- NCCS (Oak Ridge)
- 18,688 XK7 compute nodes
- In full production

**Blue Waters**
- Funded by NSF
- Allocated by PRAC
- NCSA (U. Illinois)
- 22,000 XK6 compute nodes
  + 3,000 XK7 compute nodes
- Closed for remodeling and expansion (+ 1152 XK7)
Trends Affecting Performance

- GPU performance increasing
  - Performance limit will be code on CPU
  - Most highly tuned CPU code moved to GPU
  - Remaining CPU code is also less efficient
  - Therefore CPU must run serial code well

- CPU serial performance static

- CPU core counts increasing
Suggested Strategy

• Focus on CPU-side code
  – Port to GPU or optimize/parallellize on CPU
  – Stream results off GPU to increase overlap
  – Use CPUs with best single-thread performance

• Focus on communication
  – Reduce communication overhead on CPU
  – Deal with multithreaded MPI issues
  – General parallel scalability improvements
Cray Gemini Optimization

- The new Cray machine has a better network (called Gemini)
- MPI-based NAMD scaled poorly
- BTRC implemented direct port of Charm++ to Cray
  - \textit{uGNI} is the lowest level interface for the Cray Gemini network

ApoA1 (92,000 atoms) on Cray Blue Waters prototype

Gemini provides at least 2x increase in usable nodes for strong scaling
100M Atoms on Titan vs Jaguar

Sun et al., SC2012

New Optimizations
- Charm++ uGNI port
- Node-aware optimizations
- Priority messages in critical path
- Persistent FFT messages for PME
- Shared-memory parallelism for PME

5x5x4 STMV grid
PME every 4 steps
Streaming GPU Results to CPU

- Allows incremental results from a single grid to be processed on CPU before grid finishes on GPU
- **GPU side:**
  - Write results to host-mapped memory
  - `__threadfence_system()` and `__syncthreads()`
  - Atomic increment for next output queue location
  - Write result index to output queue
- **CPU side:**
  - Poll end of output queue (int array) in host memory
TitanDev Strong Scaling

4 timesteps = 4982ms = 1.24 s / step

1.01 s for non-PME step  1.83 s for PME step

100 stmv
32 nodes

comm thread

4 timesteps = 336ms = 0.084 s / step

0.046 s for non-PME step  0.185 s for PME step

100 stmv
768 nodes

comm
TitanDev Weak Scaling

4 timesteps = 231 ms = 0.057 s / step

0.049 s for non-PME step 0.076 s for PME step

4 stmv
30 nodes

comm thread

4 timesteps = 336 ms = 0.084 s / step

0.046 s for non-PME step 0.185 s for PME step

100 stmv
768 nodes

comm thread
PME delays – tracing data needed for one ungrid calculation

4 timesteps = 336ms = 0.084 s / step

0.046 s for non-PME step

0.185 s for PME step

0.046s delay for 100KB message

0.042s delay for 10 KB message
Strategy to improve scalability

• Coarsen PME grid with higher-order interpolation
  – Reduces communication (factor of 8)
  – Does not increase short-range work or communication
• Start GPU work sooner
  – Currently waiting for all coordinate receives
  – Break up (now longer PME) to avoid delays
• Fix issues with communication
  – Improved mapping of patches to nodes
• Push PME work to the GPU
  – Charge gridding overlaps coordinate receive
Effect of Coarsening PME Grid
GPU invocation delayed by PME! Break up PME to allow interleaving.
June 2013: Improved (but not torus-aware) patch placement.
Aug 2013: after torus-aware patch placement.
NAMD PME CUDA Kernel

- CPU may be bottleneck for higher-order PME
  - Especially once the Kepler non-bonded kernel is finished…

- Target Kepler, test new features

- Simplest design that might possibly work:
  - One stream per host PE (preserve control flow)
  - One atom per warp with warp-synchronous programming
  - Atomics to accumulate charge grid in global memory
    - One per thread so accesses coalesce
    - Also build “used” flags arrays for x-y pencils and z plane
NAMD PME CUDA Kernel

Stream in

Ensure ordering

Overlap non-bonded

Stream out
A Smaller Driving Project: The Ribosome

Target of over 50% of antibiotics

Many related diseases. e.g. Alzheimer’s disease due to dysfunctional ribosome (J. Neuroscience 2005, 25:9171-9175)

Localization failure of nascent chain lead to neurodegenerative disease (Mol. Bio. of the Cell 2005, 16:279-291)
NAMD 2.9 Scalable Replica Exchange

- Easier to use *and* more efficient:
  - Eliminates complex, machine-specific launch scripts
  - Scalable pair-wise communication between replicas
  - Fast communication via high-speed network

- Basis for many enhanced sampling methods:
  - Parallel tempering (temperature exchange)
  - Umbrella sampling for free-energy calculations
  - Hamiltonian exchange (alchemical or conformational)
  - Finite Temperature String method
  - Nudged elastic band

- Great power *and* flexibility:
  - **Enables petascale simulations of modestly sized systems**
  - Leverages features of Collective Variables module
  - Tcl scripts can be highly customized and extended

Released in NAMD 2.9
NAMD 2.10 Scalable Replica Exchange

• More general Charm++ integration:
  – NAMD 2.9 used MPI communicator splitting
  – NAMD 2.10 splits replicas in Converse low-level runtime (LRTS)
  – LRTS underlies MPI, Cray (uGNI), and BlueGene/Q (PAMI) implementations

• Basis for many enhanced sampling methods:
  – Parallel tempering (temperature exchange)
  – Umbrella sampling for free-energy calculations
  – Hamiltonian exchange (alchemical or conformational)
  – Finite Temperature String method
  – Nudged elastic band

• Better scaling for individual replicas:
  – Cray uGNI layer essential for multi-node GPU replicas
  – IBM BlueGene/Q will benefit similarly from PAMI layer
  – Porting native InfiniBand (ibverbs) layer to LRTS
## NAMD 2.9: What is accelerated?

<table>
<thead>
<tr>
<th>Accelerated</th>
<th>Not Accelerated</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Short-range non-bonded</td>
<td>• Bonded terms</td>
</tr>
<tr>
<td>– Cutoff or with PME</td>
<td>• PME reciprocal sum</td>
</tr>
<tr>
<td>– w/ or w/o energy calculation</td>
<td>• Integration</td>
</tr>
<tr>
<td>• Implicit solvent</td>
<td>• Rigid bonds</td>
</tr>
<tr>
<td>• NVIDIA GPUs only</td>
<td>• Grid forces</td>
</tr>
<tr>
<td></td>
<td>• Collective variables</td>
</tr>
<tr>
<td></td>
<td>• Etc.</td>
</tr>
</tbody>
</table>
# NAMD 2.9: What is disabled?

<table>
<thead>
<tr>
<th>Disabled</th>
<th>Not Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Alchemical (FEP and TI)</td>
<td>• Memory optimized builds</td>
</tr>
<tr>
<td>• Locally enhanced sampling</td>
<td>• Conformational free energy</td>
</tr>
<tr>
<td>• Tabulated energies</td>
<td>• Collective variables</td>
</tr>
<tr>
<td>• Drude (nonbonded Thole)</td>
<td>• Grid forces</td>
</tr>
<tr>
<td>• Go forces</td>
<td>• Steering forces</td>
</tr>
<tr>
<td>• Pairwise interaction</td>
<td>• Almost everything else</td>
</tr>
<tr>
<td>• Pressure profile</td>
<td></td>
</tr>
</tbody>
</table>
NAMD 2.9: What is different?

• **Forces**
  – Slightly less accurate than CPU
    • Different interpolation scheme, single precision
    • Also affects pressure calculation

• **Energies**
  – Don’t match forces as closely as on CPU
  – Constant for interactions less than 1 Angstrom
    • Seems to be causing minimizer issues
    • Velocity-quenching minimizer should work better
NAMD 2.9: Performance

What to expect

• 1 GPU = ~24 CPU cores
  – Depending on CPU and GPU
• Scaling to 10K atoms/GPU
  – Assuming fast network
• Must use smp/multicore
  – Many cores share each GPU
  – Use multicore for single node
  – At most one process per GPU

Why it may be worse

• Weak GPU (e.g., laptop)
• Too few CPU cores used
• Coarse-grained simulation
• Too few atoms per GPU
• Limited by network
• Limited by MPI (use ibverbs)
• Limited by special features
NAMD 2.10: Kepler optimization

- Current NAMD kernel design is from 2007
- Kepler is current NVIDIA GPU architecture
  - Used on Titan and Blue Waters
  - Adds new capabilities relevant for MD codes
- Designing and optimizing new kernel for Kepler
  - New version should be faster and scale better
  - Force/energy interpolation closer to CPU version
    * Minimizer should work as well as on CPU
  - Backport to Fermi, possibly earlier if possible
Thanks to: NIH, NSF, DOE, NCSA, NVIDIA (Sarah Tariq, Sky Wu, Justin Luitjens, Nikolai Sakharnykh), Cray (Sarah Anderson, Ryan Olson), NCSA (Robert Brunner), PPL (Eric Bohm, Yanhua Sun, Gengbin Zheng, Nikhil Jain) and 18 years of NAMD and Charm++ developers and users.

James Phillips
Beckman Institute, University of Illinois
http://www.ks.uiuc.edu/Research/namd/