Evaluation of Emerging Energy-Efficient Computing Platforms for Biomolecular and Cellular Simulation Workloads

John E. Stone, Michael J. Hallock, James C. Phillips, Joseph R. Peterson, Zaida Luthey-Schulten, Klaus Schulten

NIH Center of Macromolecular Modeling and Bioinformatics
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign

http://www.ks.uiuc.edu/
http://www.scs.illinois.edu/schulten/lm/

25th Heterogeneity in Computing Workshop
IEEE International Symposium on Parallel and Distributed Processing
Chicago, IL, May 23, 2016
Biomolecular and Cellular Simulation

- **LM: Lattice Microbes**
  - Lattice-based simulations of bacterial cells using reaction diffusion models

- **NAMD: Molecular Dynamics**
  - Classical mechanics simulation of proteins, biomolecular complexes, viruses, organelles

- **VMD: Visual Molecular Dynamics**
  - Visualization and analysis of molecular and lattice cell simulations

- E.coli cell
- HIV-1 capsid and nuclear pore complex
- Molecular orbitals, vibrating $C_{60}$
Goal: A Computational Microscope

Study the molecular machines in living cells

Ribosome: target for antibiotics  Poliovirus
LM, NAMD, and VMD Use GPU Accelerators and Petascale Computing to Meet Computational Biology’s Insatiable Demand for Processing Power

Growth in size of simulated molecular complexes 1986-2014

- Lysozyme
- ApoA1
- ATP Synthase
- STMV
- Ribosome
- HIV capsid

Number of atoms

NAMD Titan XK7 Performance August 2013

HIV-1 Trajectory: ~1.2 TB/day @ 4096 XK7 nodes
Continued Growth in Simulation Performance Requires Increased Energy Efficiency

- GPUs have revolutionized MD+cell sim., 2007-present
- Kernel perf. increases of 5x to 10x are common
- Amdahl’s law pushes apps to leverage GPUs to an increasing degree for best performance
- Codes such as LM and HOOMD solely for GPU-accelerated platforms, CPU is just doing bookkeeping!
- State-of-the-art GPUs are now often thermally limited
- Questions:
  - Are emerging ARM+GPU platforms competitive for MD+cell sim?
  - Are they more energy efficient than conventional x86?
  - Why, why not?
## NCSA AC Cluster GPU Performance and Power Efficiency Results (2010)

<table>
<thead>
<tr>
<th>Application</th>
<th>GPU speedup</th>
<th>Host watts</th>
<th>Host+GPU watts</th>
<th>Perf/watt gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAMD</td>
<td>6</td>
<td>316</td>
<td>681</td>
<td>2.8</td>
</tr>
<tr>
<td>VMD</td>
<td>25</td>
<td>299</td>
<td>742</td>
<td>10.5</td>
</tr>
<tr>
<td>MILC</td>
<td>20</td>
<td>225</td>
<td>555</td>
<td>8.1</td>
</tr>
<tr>
<td>QMCPACK</td>
<td>61</td>
<td>314</td>
<td>853</td>
<td>22.6</td>
</tr>
</tbody>
</table>

Emerging ARM+GPU Platforms

A) CARMA: Tegra3 + Quadro 1000M GPU
B) KAYLA: Tegra3 + PCIe 2.0 x4 Discrete GPU
C) Jetson TK1: Tegra K1 (iGPU)
D) Jetson TX1: Tegra X1 (iGPU)
E) APM X-Gene: X-Gene + PCIe 2.0 x8 Tesla K20c GPU
ARM Platform Porting + Eval Challenges

- ARM platform Linux differences:
  - ARM Linux much less standardized than x86
  - Kernel scheduler **DVFS response time to load variation appears much longer than x86**
  - Kernel scheduler **powers off entire cores** and/or migrates processes between perf. and energy efficiency-optimized cores
  - **Dynamically varying number of available cores breaks conventional CPU work scheduler approaches** e.g. as in TBB, OpenMP – at startup they only see one available CPU core
  - **We modified apps to cope with varying core counts at runtime**

- ARM GPU drivers don’t (yet) support DVFS on-par with x86, **neither platform supports app-controlled GPU DVFS in user-mode processes**

- ARM platforms tested lacked power monitoring APIs/hw – we used external instrumentation for all reported tests
### VMD C₆₀ Molecular Kernel Orbital Performance

<table>
<thead>
<tr>
<th>Platform</th>
<th>C₆₀ MO Kernel Execution Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARMA Tegra 3 + Quadro 1000M</td>
<td>2.170 s</td>
</tr>
<tr>
<td>Jetson TK1 Tegra K1</td>
<td>2.020 s</td>
</tr>
<tr>
<td>Jetson TX1 Tegra X1 (Beta sw)</td>
<td>1.210 s</td>
</tr>
<tr>
<td>KAYLA Tegra 3 + GeForce 640</td>
<td>0.989 s</td>
</tr>
<tr>
<td>KAYLA Tegra 3 + GeForce Titan</td>
<td>0.396 s</td>
</tr>
<tr>
<td><strong>APM X-Gene + Tesla K20c</strong></td>
<td><strong>0.243 s</strong></td>
</tr>
<tr>
<td>i7-3960X + Tesla K20c</td>
<td>0.208 s</td>
</tr>
<tr>
<td>I7-3960X + GeForce Titan</td>
<td>0.157 s</td>
</tr>
</tbody>
</table>

For VMD MO kernels which are GPU-dominant, ARM platforms using comparable GPUs can approach conventional x86 platforms.
Power Monitoring Instrumentation: Commercial Kill-a-Watt

NCSA “AC” GPU cluster and Tweet-a-watt wireless power monitoring device. 0.2% accuracy w/ standard device

Power Monitoring Instrumentation: ACS712 Current Sensor + LabJack ADC
Power Monitoring Instrumentation: CARMA Attached to ACS712+DSO
Power Monitoring Instrumentation: CARMA Attached to ACS712+ADC
NAMD Simulation Performance

<table>
<thead>
<tr>
<th>Platform</th>
<th>GPU</th>
<th>Time/step</th>
<th>Power</th>
<th>Steps/kJ</th>
<th>GPU Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARMA</td>
<td>Quadro 1000M</td>
<td>0.350 s</td>
<td>34 W</td>
<td>84</td>
<td>4.3x</td>
</tr>
<tr>
<td>KAYLA</td>
<td>GeForce Titan</td>
<td>0.283 s</td>
<td>93 W</td>
<td>38</td>
<td>5.9x</td>
</tr>
<tr>
<td>i7-3960X</td>
<td>GeForce Titan</td>
<td>0.0185 s</td>
<td>444 W</td>
<td>122</td>
<td>5.8x</td>
</tr>
</tbody>
</table>

- NAMD perf. and efficiency on ARM GPU platforms far below that of x86
- ARM+GPU system software lacks host-mapped GPU memory, preventing GPU from “streaming” output to host
- NAMD sensitive to CPU-GPU small-transfer overheads
- ARM+GPU platform CPU-GPU transfer perf. Is low…
i7-3960X+Titan
fastest x86 platform

ARM +
integrated GPU
## LM Simulation Performance

<table>
<thead>
<tr>
<th>Lattice Size</th>
<th>Particles</th>
<th>Carma Steps/s</th>
<th>W</th>
<th>Efficiency</th>
<th>Kayla+ GeForce Titan Steps/s</th>
<th>W</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>32³</td>
<td>2K</td>
<td>726</td>
<td>31W</td>
<td>23.4 steps/J</td>
<td>1304</td>
<td>137W</td>
<td>9.5 steps/J</td>
</tr>
<tr>
<td>128³</td>
<td>256K</td>
<td>21.7</td>
<td>34W</td>
<td>0.64 steps/J</td>
<td>253</td>
<td>203W</td>
<td>1.2 steps/J</td>
</tr>
<tr>
<td>256³</td>
<td>512K</td>
<td>3.0</td>
<td>35W</td>
<td>0.09 steps/J</td>
<td>40.4</td>
<td>212W</td>
<td>0.19 steps/J</td>
</tr>
</tbody>
</table>

X-Gene: Efficiency winner in all but one LM test case, perf is competitive with x86

<table>
<thead>
<tr>
<th>Lattice Size</th>
<th>Particles</th>
<th>Intel i7-3960X+Tesla K20c Steps/s</th>
<th>W</th>
<th>Efficiency</th>
<th>APM X-Gene+Tesla K20c Steps/s</th>
<th>W</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>32³</td>
<td>2K</td>
<td>5463</td>
<td>226W</td>
<td>24.2 steps/J</td>
<td>4638</td>
<td>142W</td>
<td>32.6 steps/J</td>
</tr>
<tr>
<td>128³</td>
<td>256K</td>
<td>305</td>
<td>266W</td>
<td>1.2 steps/J</td>
<td>300</td>
<td>189W</td>
<td>1.6 steps/J</td>
</tr>
<tr>
<td>256³</td>
<td>512K</td>
<td>48.3</td>
<td>270W</td>
<td>0.18 steps/J</td>
<td>47.7</td>
<td>195W</td>
<td>0.24 steps/J</td>
</tr>
</tbody>
</table>
ARM + integrated GPU

X-Gene+GPU: fastest ARM platform

Note comparative performance for transfers ranging from 2KB to 16KB

i7-3960X+Titan fastest x86 platform
Digging Deeper Into CPU-GPU Transfer Performance Issues

- ARM consistently underperformed vs. x86 comparison cases
- ARM PCIe interfaces run at lower rates: x4 or x8 vs. x16 on x86
- ARM perf. for small-sized transfers is very low, even w/ integrated GPUs:
  - ARM cores lack sophistication in single-thread code paths, no out-of-order, etc
  - ARM arch has complex procedure for VM ops, e.g., **TLB shootdown**; tree-like sync of all CPU cores
  - Lower clock rates
  - Driver stack less mature than x86
Improving VMD C_{60} MO Perf. on KAYLA ARM with Optimized CPU-GPU Transfers

<table>
<thead>
<tr>
<th>C_{60} MO Algorithm</th>
<th>Perf. (FPS)</th>
<th>Power (W)</th>
<th>Energy Efficiency (frames/kJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>2.12 FPS</td>
<td>88 W</td>
<td>24 frames/kJ</td>
</tr>
<tr>
<td>New Transfer-optimized</td>
<td>3.89 FPS</td>
<td>89 W</td>
<td>49 frames/kJ</td>
</tr>
</tbody>
</table>

- Eliminate copy of MO wavefunction densities from GPU to CPU
- Perform intermediate marching cubes step in-place on GPU
- **1.8x performance increase on ARM**
- On x86 strategy is usually but not universally better, closer to break-even point, multi-GPU, etc.
Future Work

- Ongoing study of sources of PCIe CPU-GPU transfer overheads, schemes to mitigate performance loss on ARM or other platforms
- Direct link high-freq. power monitoring instrumentation to conventional performance instrumentation tools
- Develop new GPU algorithms that are tolerant of low-perf. CPU-GPU transfers: new kernels, even simple ones, that eliminate small transfers when possible
- Re-test platforms reported on here with new and improved compilers, kernels, drivers, an other system software expected to become available later this year
- Develop new low-cost instrumentation schemes that separate GPU and CPU power on commodity x86
- Compare w/ POWER8, other x86 platforms, Xeon Phi, etc.
Acknowledgements

• Theoretical and Computational Biophysics Group, University of Illinois at Urbana-Champaign

• NVIDIA GPU Center of Excellence, University of Illinois at Urbana-Champaign

• NVIDIA CUDA team

• NCSA

• Funding:
  – DOE INCITE, ORNL Titan: DE-AC05-00OR22725
  – NSF Blue Waters:  
    NSF OCI 07-25070, PRAC “The Computational Microscope”,  
    ACI-1238993, ACI-1440026
  – NIH support: 9P41GM104601, 5R01GM098243-02, PHS 5 T32 GM008276
NIH BTRC for Macromolecular Modeling and Bioinformatics

Beckman Institute
University of Illinois at Urbana-Champaign

1990-2017
Related Publications
http://www.ks.uiuc.edu/Research/gpu/


• **High Performance Molecular Visualization: In-Situ and Parallel Rendering with EGL.** John E. Stone, Peter Messmer, Robert Sisneros, and Klaus Schulten. High Performance Data Analysis and Visualization Workshop, IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2016. (In-press)


• **Chemical Visualization of Human Pathogens: the Retroviral Capsids.** Juan R. Perilla, Boon Chong Goh, John E. Stone, and Klaus Schulten SC’15 Visualization and Data Analytics Showcase, 2015.
Related Publications
http://www.ks.uiuc.edu/Research/gpu/


• **Unlocking the Full Potential of the Cray XK7 Accelerator.** M. D. Klein and J. E. Stone. Cray Users Group, Lugano Switzerland, May 2014.


Related Publications
http://www.ks.uiuc.edu/Research/gpu/


Related Publications

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


Related Publications
http://www.ks.uiuc.edu/Research/gpu/


Related Publications

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

- **Adapting a message-driven parallel application to GPU-accelerated clusters.**

- **GPU acceleration of cutoff pair potentials for molecular modeling applications.**

- **GPU computing.**

- **Accelerating molecular modeling applications with graphics processors.**

- **Continuous fluorescence microphotolysis and correlation spectroscopy.**