USING NSIGHT TOOLS TO OPTIMIZE THE NAMD MOLECULAR DYNAMICS SIMULATION PROGRAM

Robert (Bob) Knight - Software Engineer, NVIDIA
David Hardy - Senior Research Programmer, U. Illinois at Urbana Champaign
March 19, 2019
Rebalancing Your Investment Mix

Even during a strong stock market, you should evaluate risks and returns.

It’s Time to Rebalance
WHY REBALANCE?

GPU Motivation: Performance Trends

Peak Double Precision FLOPS

Peak Memory Bandwidth
BECAUSE PERFORMANCE MATTERS

DELIVERING MOST OF THE NEW COMPUTING PERFORMANCE

NVIDIA GPUs Share of New FLOPS on Top 500 Systems

- 2015 Tesla K80: 11%
- 2017 Tesla P100: 25%
- 2018 Tesla V100: 56%
HOW?
NSIGHT SYSTEMS & NSIGHT COMPUTE

- Nsight Systems
  - Focus on the application’s algorithm - a unique perspective
  - Rebalance your application’s compute cycles across the system’s CPUs & GPUs

- Nsight Compute
  - CUDA kernel profiling

Workflow
Start Here
NAMD - NANOSCALE MOLECULAR DYNAMICS

- 25 years of NAMD
- 50,000+ Users
- Awards: 2002 Gordon Bell, 2012 Sidney Fernbach
- Solving Important Biophysics/Chemistry Problems
- Focused on scaling across GPUs - Biggest Bang for Their Compute $
NAMD & VISUAL MOLECULAR DYNAMICS COMPUTATIONAL MICROSCOPE

Enable researchers to investigate systems at the atomic scale

NAMD - molecular dynamics simulation
VMD - visualization, system preparation and analysis

Ribosome  Neuron  Virus Capsid
NAMD OVERVIEW

- Simulate the physical movement of atoms within a molecular system
- Atoms are organized in fixed volume *patches* within the system
- Forces that move atoms are calculated at each timestep
- After a *cycle* (e.g. 20 timesteps), atoms may migrate to an adjacent patch
- Performance measured as ns/day - the number of nanoseconds of simulation that could be calculated in one day of running the workload (higher is better)
PARALLELISM IN MOLECULAR DYNAMICS LIMITED TO EACH TIMESTEP

Computational workflow of MD

*Iterate for billions of time steps*

1. **Initialize** coordinates
2. **Force calculation**
3. **Update coordinates**

- **About 1% of computational work**
- **About 99% of computational work**
TIMESTEP COMPUTATIONAL FLOP COST

- force calculation
  - 90% — short-range non-bonded forces
  - 5% — long-range PME electrostatics
  - 2% — bonded forces
  - 2% — corrections for excluded interactions
  - 1% — numerical integration

Start applying GPU acceleration to most expensive parts
BREAKDOWN A WORKLOAD
NVIDIA TOOLS EXTENSION (NTVX) API

- Instrument application behavior
  - Supported by all NVIDIA tools
- Insert markers, ranges
- Name resources
  - OS thread, CUDA runtime
- Define scope using *domains*
NAMD ALGORITHM SHOWN WITH NVTX

- Zoom Out
- Distinct Phases of NAMD Become Visible

NVTX Ranges

Initialization

Setup

Simulation
- Zoom In
- 20 Timesteps followed by Atom Migration
ONE NAMD TIMESTEP

- Zoom In

  2197 patch updates

  calculate forces
TIMESTEP - SINGLE PATCH

- Zoom In
- Patches are implemented as user-level threads

~88us
NSIGHT SYSTEMS

- User Instrumentation
- API Tracing
- Backtrace Collection
- Custom Data Mining
- Nsight Compute Integration
API TRACING

- Process stalls on file I/O while waiting for 160ms mmap64 operation

- Thread communicates over socket

- APIs: CUDA, cuDNN, cuBLAS, OSRT (OS RunTime), OpenGL, OpenACC, DirectX 12, Vulkan*  
  * Available in next release
Filter By Selection shows a specific thread’s activity.

Blocked State Backtrace shows the path leading to an OS runtime library call.
REPORT NAVIGATION DEMO
CUSTOM DATA MINING

- nsys-exporter*
  - QDREP->SQLite

- Use Cases
  - Outlier Discovery
  - Regression Analysis
  - Scripted Custom Report Generation

Kernel Statistics - all times in nanoseconds

<table>
<thead>
<tr>
<th>minimum</th>
<th>maximum</th>
<th>average</th>
<th>kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1729557</td>
<td>5347138</td>
<td>2403882.7</td>
<td>nonbondedForceKernel</td>
</tr>
<tr>
<td>561821</td>
<td>631674</td>
<td>581409.6</td>
<td>batchTranspose_xyz_yzx_kernel</td>
</tr>
<tr>
<td>474173</td>
<td>574618</td>
<td>489148.1</td>
<td>batchTranspose_xyz_zxy_kernel</td>
</tr>
<tr>
<td>454621</td>
<td>593402</td>
<td>465637.6</td>
<td>spread_charge_kernel</td>
</tr>
<tr>
<td>393470</td>
<td>676060</td>
<td>420914.9</td>
<td>gather_force</td>
</tr>
<tr>
<td>52288</td>
<td>183455</td>
<td>116258.2</td>
<td>bondedForcesKernel</td>
</tr>
</tbody>
</table>

... nonbondedForceKernel

duration    start        stream      context      GPU
----------  -----------  ----------  ----------  ----------
5347138     35453528745  133         7            1
5245934     39527523457  132         8            0
5076271     41048810842  132         8            0

The longest nonBondedForceKernel is at 35.453s on GPU1, stream 133

* Available in next release
KERNELSTATS SCRIPT

#!/bin/bash

sqlite3 $DB "ALTER TABLE CUPTI_ACTIVITY_KIND_KERNEL ADD COLUMN duration INT;"

sqlite3 $DB "UPDATE CUPTI_ACTIVITY_KIND_KERNEL SET duration = end-start;"

// add duration column, set duration column’s value

sqlite3 $DB "CREATE TABLE kernelStats (shortName INTEGER, min INTEGER, max INTEGER, avg INTEGER);"

sqlite3 $DB "INSERT INTO kernelStats SELECT shortName, min(duration), max(duration), avg(duration) FROM CUPTI_ACTIVITY_KIND_KERNEL GROUP BY shortName;"

// create new table, insert kernel name IDs, min, max, avg into it

sqlite3 -column -header $DB SELECT min as minimum, max as maximum, round(avg,1) as average, value as kernel FROM kernelStats INNER JOIN StringIds ON StringIds.id = kernelStats.shortName ORDER BY avg DESC;“

// print formatted min, max, avg, and kernel name values, order by descending avg
NSIGHT COMPUTE INTEGRATION

Right click on kernel, select Analyze…

Copy suggested Compute command line, profile it…
DATA COLLECTION

Host – Target Remote Collection

Command Line Interface

- No root access required
- Works in Docker containers
- Interactive Mode
  - Supports `cudaProfilerStart/Stop` APIs
What about NAMD...

Profiling Simulations of Satellite Tobacco Mosaic Virus (STMV)
~ 1 Million Atoms
V2.12 TIMESTEP COMPUTATIONAL FLOP COST

- **force calculation**
  - 90% — short-range non-bonded forces
  - 5% — long-range PME electrostatics
  - 2% — bonded forces
  - 2% — corrections for excluded interactions
  - 1% — numerical integration

- **update coordinates**
  - GPU
  - CPU
NAMD V2.12

Profiling STMV with nvprof - Maxwell GPU Fully Loaded
NAMD V2.12
Volta GPU Severely Underloaded

- Bonded forces and exclusions calculations
- ~23.4ms
- ~22.7ms
- Bonded forces and exclusions 25.8%
### NAMD PERFORMANCE

<table>
<thead>
<tr>
<th>GPUs</th>
<th>Architecture</th>
<th>V2.12 Nanoseconds/Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Maxwell</td>
<td>0.65</td>
</tr>
<tr>
<td>1</td>
<td>Volta</td>
<td>5.34619</td>
</tr>
<tr>
<td>2</td>
<td>Volta</td>
<td>5.45701</td>
</tr>
<tr>
<td>4</td>
<td>Volta</td>
<td>5.35999</td>
</tr>
<tr>
<td>8</td>
<td>Volta</td>
<td>5.31339</td>
</tr>
</tbody>
</table>

Volta (2018) delivers ~10x performance boost relative to Maxwell (2014)

Failure to scale is caused by unbalanced resource utilization
V2.13 Timestep Computational FLOP Cost

- **Force Calculation**
  - 90% — short-range non-bonded forces
  - 5% — long-range PME electrostatics
  - 2% — bonded forces
  - 2% — corrections for excluded interactions

- **Update Coordinates**
  - 1% — numerical integration

- **GPU**
  - Force Calculation
  - Update Coordinates

- **CPU**
  - Force Calculation
  - Update Coordinates
NAMD V2.13
Moving all force calculations to GPU shrinks timeline gap

~18.5ms
~15.5ms
## NAMD PERFORMANCE

<table>
<thead>
<tr>
<th>GPUs (Volta)</th>
<th>V2.12 Nanoseconds/Day</th>
<th>V2.13 Nanoseconds/Day (% Gain vs 2.12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.34619</td>
<td>5.4454 (1.2%)</td>
</tr>
<tr>
<td>2</td>
<td>5.45701</td>
<td>5.97838 (9.5%)</td>
</tr>
<tr>
<td>4</td>
<td>5.35999</td>
<td>7.49265 (39.8%)</td>
</tr>
<tr>
<td>8</td>
<td>5.31339</td>
<td>7.55954 (42.3%)</td>
</tr>
</tbody>
</table>
Force calculation

- 90% — short-range non-bonded forces
- 5% — long-range PME electrostatics
- 2% — bonded forces
- 2% — corrections for excluded interactions
- 1% — numerical integration

Update coordinates

GPU

CPU
### Bonded Kernel Optimization

Host-side post-processing of bonded forces still a significant bottleneck.

<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>Self, %</th>
<th>Module Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence:submitReductions_SOA()</td>
<td>19.08%</td>
<td>projects/shardy/namd/Linux-x86_64-icc-smp/CUDA_devel/nAMDd</td>
</tr>
<tr>
<td>Sequence:submitHofstep_SOA()</td>
<td>16.59%</td>
<td>projects/shardy/namd/Linux-x86_64-icc-smp/CUDA_devel/nAMDd</td>
</tr>
<tr>
<td>Sequence: addForceToMomentum_SOA(d, d, d, d)</td>
<td>6.40%</td>
<td>projects/shardy/namd/Linux-x86_64-icc-smp/CUDA_devel/nAMDd</td>
</tr>
<tr>
<td>CmRdfAllTimer</td>
<td>5.55%</td>
<td>projects/shardy/namd/Linux-x86_64-icc-smp/CUDA_devel/nAMDd</td>
</tr>
<tr>
<td>homePatch:doMarginCheck_SOA()</td>
<td>4.18%</td>
<td>projects/shardy/namd/Linux-x86_64-icc-smp/CUDA_devel/nAMDd</td>
</tr>
<tr>
<td>Or7/Ifes1fee2f</td>
<td>3.86%</td>
<td>[CUDA]</td>
</tr>
<tr>
<td>ComputeBondedCUDA:FinishPatchesOnP1</td>
<td>3.53%</td>
<td>projects/shardy/namd/Linux-x86_64-icc-smp/CUDA_devel/nAMDd</td>
</tr>
<tr>
<td>CmGettonLocal</td>
<td>3.52%</td>
<td>projects/shardy/namd/Linux-x86_64-icc-smp/CUDA_devel/nAMDd</td>
</tr>
<tr>
<td>ComputePrivateCUDA:sendAtoms()</td>
<td>2.93%</td>
<td>projects/shardy/namd/Linux-x86_64-icc-smp/CUDA_devel/nAMDd</td>
</tr>
<tr>
<td>CodaCallbacks</td>
<td>2.42%</td>
<td>projects/shardy/namd/Linux-x86_64-icc-smp/CUDA_devel/nAMDd</td>
</tr>
</tbody>
</table>

**Cache locality optimized - type conversion on GPU, loop rearranged**

**ns/day gain 0%**

Not on critical path. Will be future benefit in multi-GPU environment.
NAMD NEXT

CPU integrator causing bottleneck

1% of computation is now ~50% of timestep work. Amdahl’s Law strikes again.

Data parallel calculation for GPU!
NAMD NEXT

Integrator Development Phases

CPU vectorization improvements

CUDA integrator per patch

CUDA integrator per CPU core

Manageable Changes
Validate Each Step

CUDA integrator per system (upcoming)
CPU vectorization – arrange data into SOA (structure of arrays) form

Speedup calculated via custom SQL-based script, NVTX ranges
Speedups: 26.5% for integrate_SOA_2, 52% for integrate_SOA_3
NAMD NEXT

Integrator - Phase 2

Per **Patch** Integrator Offload - Zoom In

Memory Transfer **Hell**

GPU Underutilized

Each Kernel Handles ~500 atoms

STMV includes 1M atoms

2200 *more* streams
NAMD NEXT

*Integrator - Phase 3*

Per **CPU Core** Integrator Offload - Zoom In

- Improved GPU Utilization
- Each Kernel Handles ~33K atoms
- STMV includes 1M atoms

**CPU Utilization**

**GPU timeline**

filing with integrator work

**CPU Utilization**

Drops Dramatically
Small memory copy operations should be avoided by grouping data together.

Improve memory access performance by using Pinned memory.
## NAMD PERFORMANCE

<table>
<thead>
<tr>
<th>GPUs</th>
<th>V2.12 Nanoseconds/Day</th>
<th>V2.13 Nanoseconds/Day (% Gain vs 2.12)</th>
<th>NEXT Nanoseconds/Day SOA + Incomplete Integrator Phase 3 (% Gain vs 2.12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.34619</td>
<td>5.4454 (1.2%)</td>
<td>4.1451 (-22.5%)</td>
</tr>
<tr>
<td>2</td>
<td>5.45701</td>
<td>5.97838 (9.5%)</td>
<td>5.76149 (5.6%)</td>
</tr>
<tr>
<td>4</td>
<td>5.35999</td>
<td>7.49265 (39.8%)</td>
<td>7.11889 (32.8%)</td>
</tr>
<tr>
<td>8</td>
<td>5.31339</td>
<td>7.55954 (42.3%)</td>
<td>8.10406 (52.5%)</td>
</tr>
</tbody>
</table>

**Integrator Phase 3 Optimization Development In Progress**
NAMD NEXT

What? Charm++ Runtime using 21.3% on gettimeofday()!

Replace with x86_64 RDTSC instruction, avoid unnecessary calculations.
NAMD NEXT

Charm++ Runtime Lock Optimization

Replace `pthread_mutex_lock` with `pthread_spin_lock`.
## NAMD PERFORMANCE

<table>
<thead>
<tr>
<th>GPUs</th>
<th>V2.12 Nanoseconds/Day</th>
<th>V2.13 Nanoseconds/Day (% Gain vs 2.12)</th>
<th>NEXT Nanoseconds/Day SOA+Runtime (% Gain vs 2.12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.34619</td>
<td>5.4454 (1.2%)</td>
<td>6.7032 (25.4%)</td>
</tr>
<tr>
<td>2</td>
<td>5.45701</td>
<td>5.97838 (9.5%)</td>
<td>7.24473 (32.8%)</td>
</tr>
<tr>
<td>4</td>
<td>5.35999</td>
<td>7.49265 (39.8%)</td>
<td>8.95371 (67.0%)</td>
</tr>
<tr>
<td>8</td>
<td>5.31339</td>
<td>7.55954 (42.3%)</td>
<td>9.26881 (74.4%)</td>
</tr>
</tbody>
</table>

Integrator Phase 3 Optimization Development In Progress
Nsight Systems guides development process

- Estimated best case performance: 10 to 12 nanoseconds/day (single GPU)
- Data transfer activity constraining performance

- CPU vectorization improvements
- CUDA integrator per patch
- CUDA integrator per CPU core
- CUDA integrator per system (upcoming)

CUDA integrator minimizing data transfer with overlapped computation
COMMON OPTIMIZATION OPPORTUNITIES

- **CPU**
  - Thread Synchronization
  - Algorithm bottlenecks starve the GPU(s)

- **Multi GPU**
  - Communication between GPUs
  - Lack of Stream Overlap in memory management, kernel execution

- **Single GPU**
  - Memory operations - blocking, serial, unnecessary
  - Too much synchronization - device, context, stream, default stream, implicit
  - CPU GPU Overlap - avoid excessive communication
NSIGHT PRODUCT FAMILY

- **Nsight Systems** - Analyze application algorithm system-wide
- **Nsight Compute** - Debug/optimize CUDA kernel
- **Nsight Graphics** - Debug/optimize graphics workloads
ACKNOWLEDGMENTS

- U. of Illinois
  - Julio Maia, Ronak Buch, John Stone, Jim Phillips

- NVIDIA
  - Daniel Horowitz, Antoine Froger, Sneha Kottapallli, Peng Wang
THANK YOU!

Visit us at the NVIDIA booth for a live demo!

Download latest public version
https://developer.nvidia.com/nsight-systems

Also available in CUDA Toolkit (v10.1 and later)

Forums: https://devtalk.nvidia.com

Email: nsight-systems@nvidia.com
DEVELOPER TOOLS AT GTC19

Talks

- S9345: CUDA Kernel Profiling using NVIDIA Nsight Compute
- S9661: Nsight Graphics - DXR/Vulkan Profiling/Vulkan Raytracing
- S9751: Accelerate Your CUDA Development with Latest Debugging and Code Analysis Developer Tools
- S9866: Optimizing Facebook AI Workloads for NVIDIA GPUs
- S9339: Profiling Deep Learning Networks

Demos of DevTools products on Linux, DRIVE AGX, & Jetson AGX at the showfloor

- Wednesday @12-7
- Thursday @11-1
DEMO BACKUP
CORRELATE ACTIVITY

Selecting one highlights both cause and effect, i.e. dependency analysis.
FINDING A CORRELATION