Title: Development of a Library for Computing Monte Carlo Tallies on Heterogeneous Systems

Author(s): Burke, Timothy Patrick
Brown, Forrest B.

Intended for: ANS Winter Meeting 2018, 2018-11-11 (Orlando, Florida, United States)

Issued: 2018-11-01

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Development of a Library for Computing Monte Carlo Tallies on Heterogeneous Systems

Timothy P. Burke, Forrest Brown

XCP-3 Monte Carlo Codes, Methods, and Applications
Los Alamos National Laboratory

ANS Winter 2018

Motivation

Several Monte Carlo codes at LANL

- MCNP
- MCATK
- Jayenne
- Mini-apps

...many more elsewhere

All codes have tally capabilities duplicated

- Eg. mesh tallies, cell and energy tallies, etc.

Tally capabilities heavily integrated into transport code

Motivation

Goals of Tally LIbrary:

1. Code-reuse among various Monte Carlo codes
   - Code maintainability
   - Unified I/O
   - New methods readily transferred between codes
2. Export tallies to GPUs
   - CPU generates particle tracks and collisions, GPU conducts tallies

Talon has been in development for ~1 year, with plans for open source distribution in the future

- C++ based
- CUDA for GPUs, using same algorithms and functions as CPU.

Talon Capabilities

Linked to MCNP, MCATK, and OpenMC

- C/Fortran interface for MCNP and OpenMC

Tallies:

- Mesh Tallies
- Surface-crossing Flux Estimators
- Functional Expansion Tallies
- Kernel Density Estimators
- Sensitivities to cross-section and geometric perturabations
  - Differential Operator Sampling (DOS)
  - Iterated Fission Probability (IFP)

Filters:

- Cell, Energy, Surface, etc.

Talon Input

Tallies defined at compile time or run time

- Compile-time optimization
  - Arrays instead of vectors for filter bins
  - Container of tallies free of virtual function performance penalties
- XML based input

```
<tallies>

  <tally id="2">
    <type> collision </type>
    <dimension> 1 </dimension>
    <architecture> cpu </architecture>
    <precision> double </precision>
    <scores> flux </scores>
    <filters> 2 3 </filters>
  </tally>

  <filter id="1" type="energy" bins="0. 10.0e6" />
  <filter id="2" type="energy" bins="0. 0.5e6 1.0e6 10.0e6" />
  <filter id="3" type="cell" bins="1 2" />
  <filter id="4" type="surface" bins="7" />

</tallies>
```

Talon Interface

C/Fortran interface

- initialize tallies / read tally input file
- initialize tally batch
- compute particle track / collision contribution
- handle banking and unbanking of secondary particles
- accumulate batch statistics
- finalize tallies / compute tally uncertainties

Talon also needs data from transport codes

- cell and surface lookups
- cross section lookups

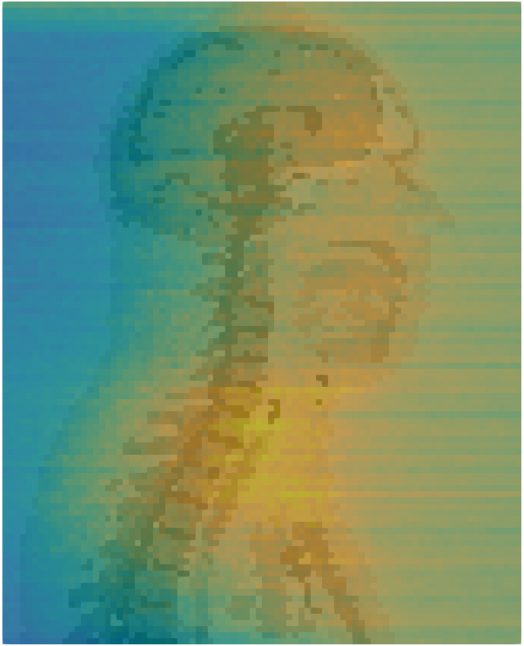
Either pass a function to Talon, or define C-callable routines in transport code

Tally visualization

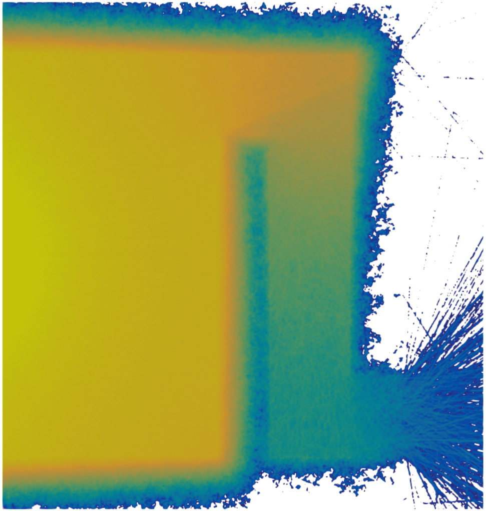
Plotting done via ParaView

Flux from a 3-D track-length histogram at the center of VIP man with an overlay of material density in greyscale at 20% opacity.

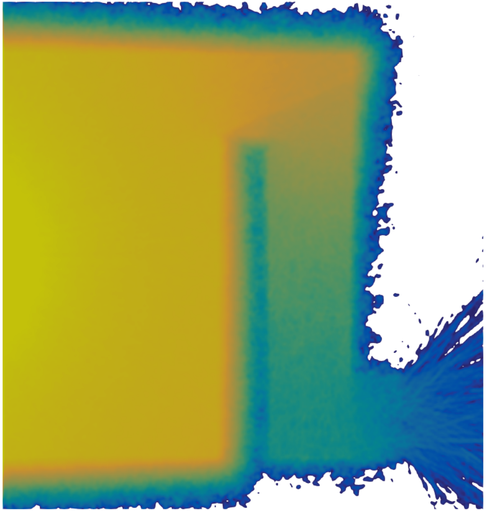
- 1.3×10^6 voxels, 2 mm cubes
- Mono-directional beam of gamma rays
- 10^7 histories using MCNP
- Geometry plotted as voxelized representation of density (Credit: Joel Kulesza)



KDE vs Mesh Tally in Concrete Treatment Room



Mesh



KDE

- Isotropic point source of 1 MeV neutrons, 5×10^5 particles using MCNP
- 950×950 mesh elements, each 1 cm $\times$ 1 cm
- Concrete treatment room 7.5 x 7.5 m x 3.8 m, with maze
- Delta-scattering XS of 2 cm^{-1}

Parallelization

CPU parallelization determined in the transport code

- Compatible with OpenMP and MPI
  - MPI tested with MCATK, OpenMC, and MCNP
  - OpenMP tested with MCNP

Can also specify tallies to run on GPUs

- Compatible with CUDA-aware MPI, threading should work but is untested

Tallies on GPUs

Clusters are moving towards GPUs or Co-processors in the age of exascale computing

- Summit, 6 GPUs / 2 CPUs
- Sierra, 4 GPUs / 2 CPUs
- Desktops, 1 GPU / 1 CPU

GPUs sit unused in most Monte Carlo codes

- Changing in recent years, MCNP still does not use GPUs

Export tallies to GPU while transport is being done on CPU

Design Talon to write one algorithm that can be used on CPUs or GPUs - “performance portable”

- Uses same ideology as Raja or Kokkos

Concept has been proven w/ KDEs, volumetric ray-casting estimators

Tallies on GPUs

CPU creates particle tracks and collisions, event data passed to Talon

- Talon stores particle data (position, angle, distance, energy, cell, material, etc.) in batches (49152 samples)
- Looks up / computes additional XS data for reaction rates and stores it
- Once filled, data asynchronously sent to GPU for processing
- GPU tally kernels launched asynchronously (CUDA streams)
- CPU immediately returns and begins filling another set of events
  - Waits if GPU hasn't finished processing the array it's about to fill

Can also process events as they happen w/o storage

GPU / CPU code re-use

CUDA code must be decorated with __device__ qualifiers

- CUDA code cannot use containers / methods from the standard library
- array, vector, map, function, iterators, etc

CUDA code examples often contain raw pointers

Talon uses Kokkos containers and overloaded versions of std::vector

- Enables CUDA kernels that look like normal CPU code

GPU / CPU Code re-use

Tallies are templated on an execution policy

- What data structures to use
- How to allocate and access memory
- Where to run tally calculations
- What atomic operations to use

Tally methods written as functors or C++ lambdas

- Same algorithm executed on CPU and GPU
- Data layouts may be different (AoS vs SoA)

Talon Performance

Performance tested on a 3-D tally in a 1-D slab problem

- Transport and tally calculations overlap when using GPU
- Simple transport calculation puts larger burden on tallies

Results are shown for tally-only time

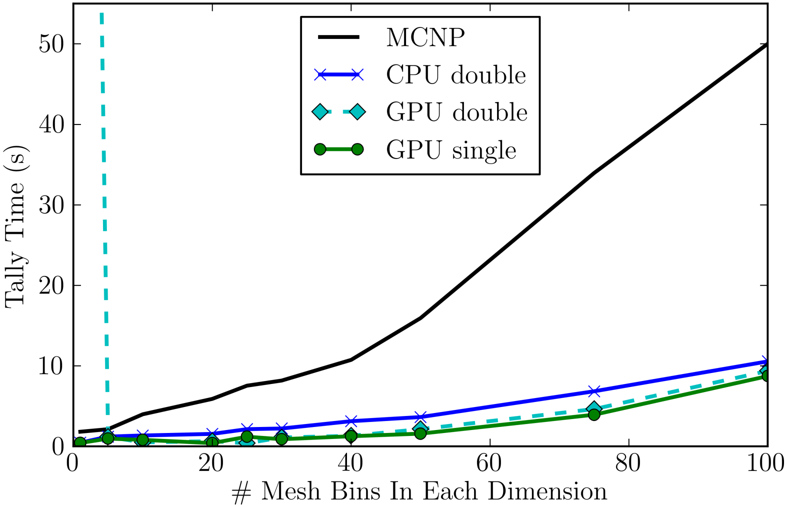
- Average of 5 runs w/ tallies minus average of 5 runs w/o tallies
- 5 inactive, 15 active, 10^6 particles/generation

One compute node, two Intel Xeon E5-2660 V3 processors and one NVIDIA GTX TitanX GPU

Timing obtained for single-precision and double-precision Talon tallies and double-precision MCNP tallies

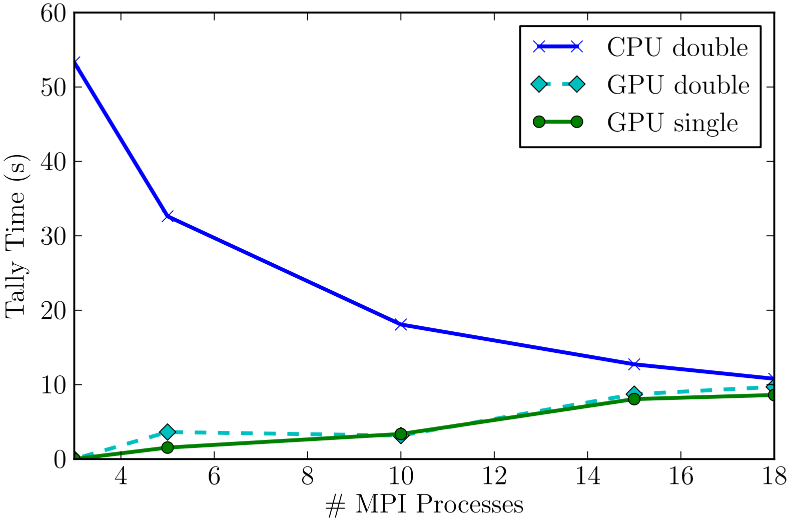
- GTX TitanX single-precision FLOPS ~$30\times$ larger than double-precision

Mesh Tally Scaling with Mesh Size



- All runs use 18 MPI processes
- 15.3 seconds w/o tallies

Mesh Tally Scaling with MPI Processes



Tally time versus number of MPI processes for a $100 \times 100 \times 100$ mesh tally.

Summary and Future Work

- Demonstrated GPU tally capabilities
- Method / code re-use between 3 MC transport codes
  - Derivatives/sensitivities in OpenMC & MCNP (ANS Winter 2018)
  - Derivatives with respect to geometric perturbations in MCNP (RPSD 2018)

Future work

- Python API
- Refactor code base
- Open-source release
- Distributed tallies / tally server

Acknowledgements

This work was supported by the DOE-NNSA Advanced Simulation and Computing and Nuclear Criticality Safety Programs.

Special thanks to Joel Kulesza for MCNP Geometry plotting in ParaView and Austin McCartney for C++ guidance.

Questions?

Development of a Library for Computing Monte Carlo Tallies on Heterogeneous Systems

Timothy P. Burke, Forrest Brown

XCP-3 Monte Carlo Codes, Methods, and Applications
Los Alamos National Laboratory

ANS Winter 2018