SAND2018-XXXXX

SAND2018-11004C

Comparison of Kokkos and CUDA Programming Models for Key Kernels in the Monte Carlo Transport Algorithm

Kerry L. Bossler, PhD; Greg D. Valdez, PhD
Sandia National Laboratories

Introduction

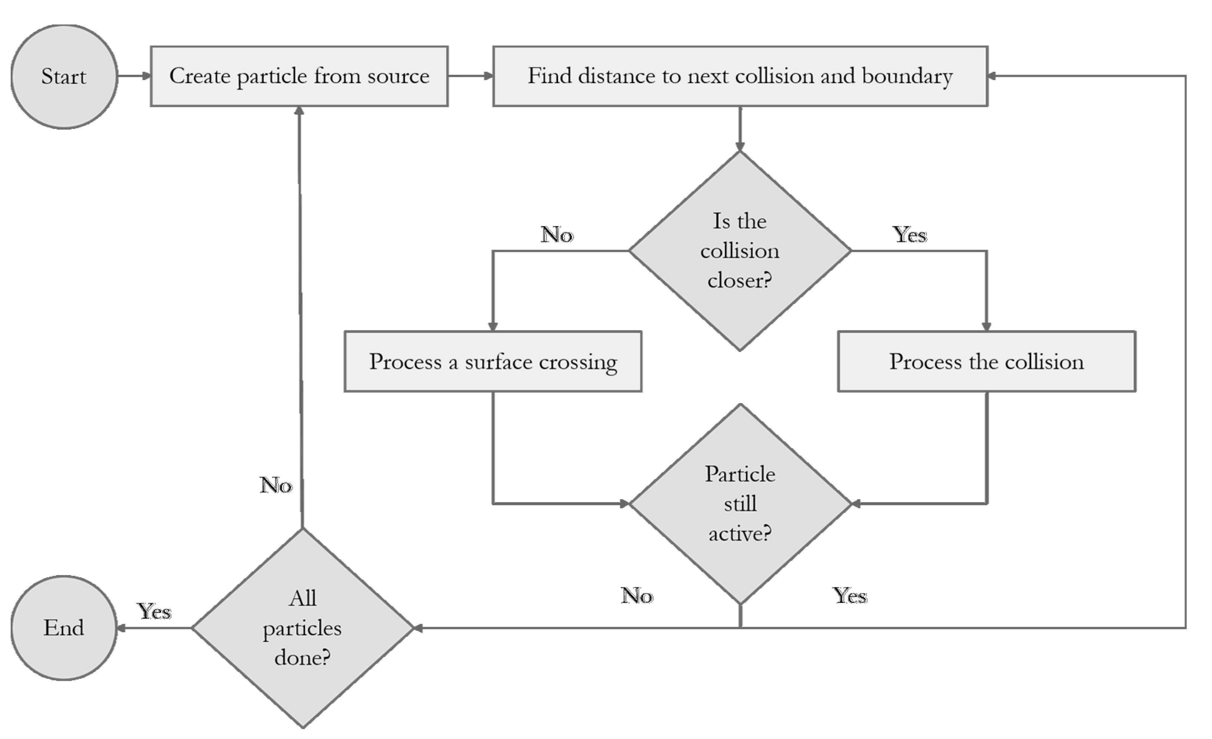
GOAL: Assess effectiveness of the Kokkos programming model for Monte Carlo particle transport on the GPU

- Kokkos is a programming model for performance portability across manycore devices including GPUs
- CUDA is a programming model explicitly designed for NVIDIA GPUs
- CUDA has direct access to features such as constant memory and warp shuffle that can improve performance

Monte Carlo Transport Algorithm

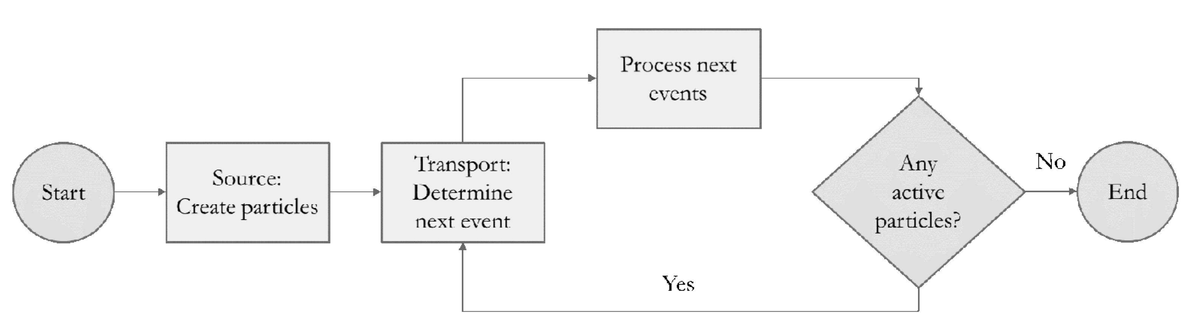
History-Based Transport Algorithm

- Traditional approach used by most production codes
- GPU implementation fully defined within one Big Kernel



Event-Based Transport Algorithm

- Novel approach that reduces divergence on GPUs
- GPU implementation requires multiple kernels



Event-Based Photon Attenuation

- Photon attenuation can be implemented with only three kernels as no events need to be processed

Source Kernel	• Creates all photons from a common source definition
Transport Kernel	• Identifies which photons are absorbed and which ones escape
Tally Kernel	• Counts the total number of photons that escaped

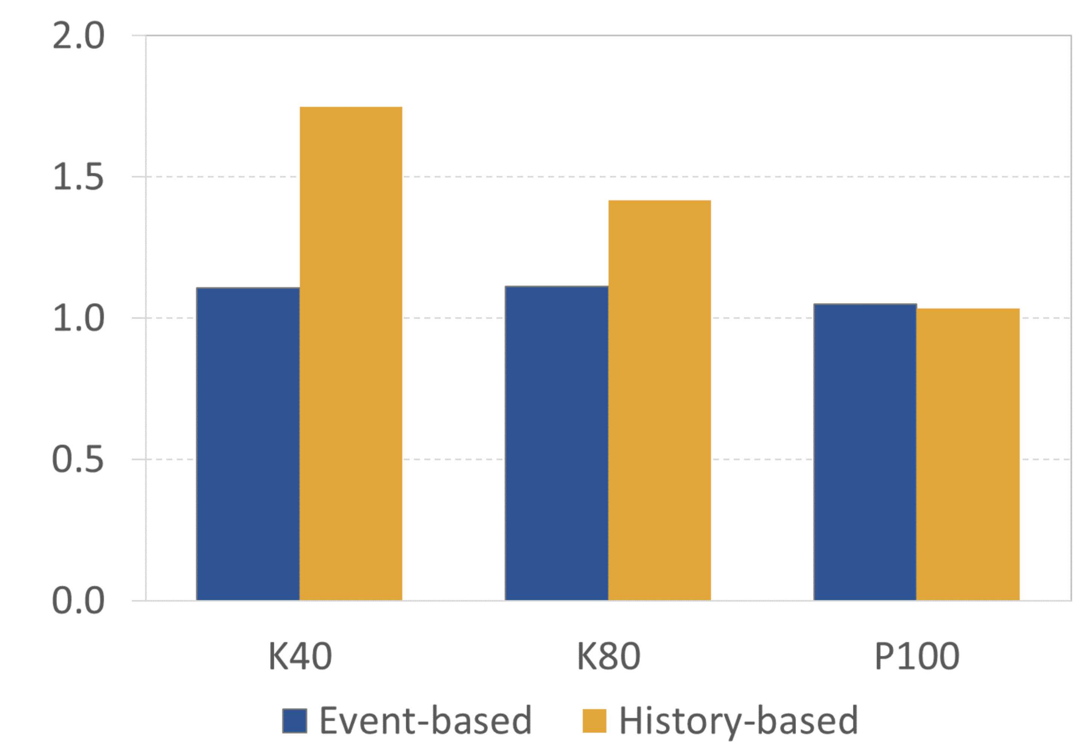
Performance Tests

- 1D photon attenuation problem with 32-bit integer escape tally for 10^8 particle histories
- Compared history-based and event-based algorithms
- Repeated tests on Tesla K40, Tesla K80, and Tesla P100
- All timing data is an average of ten independent runs

General Results

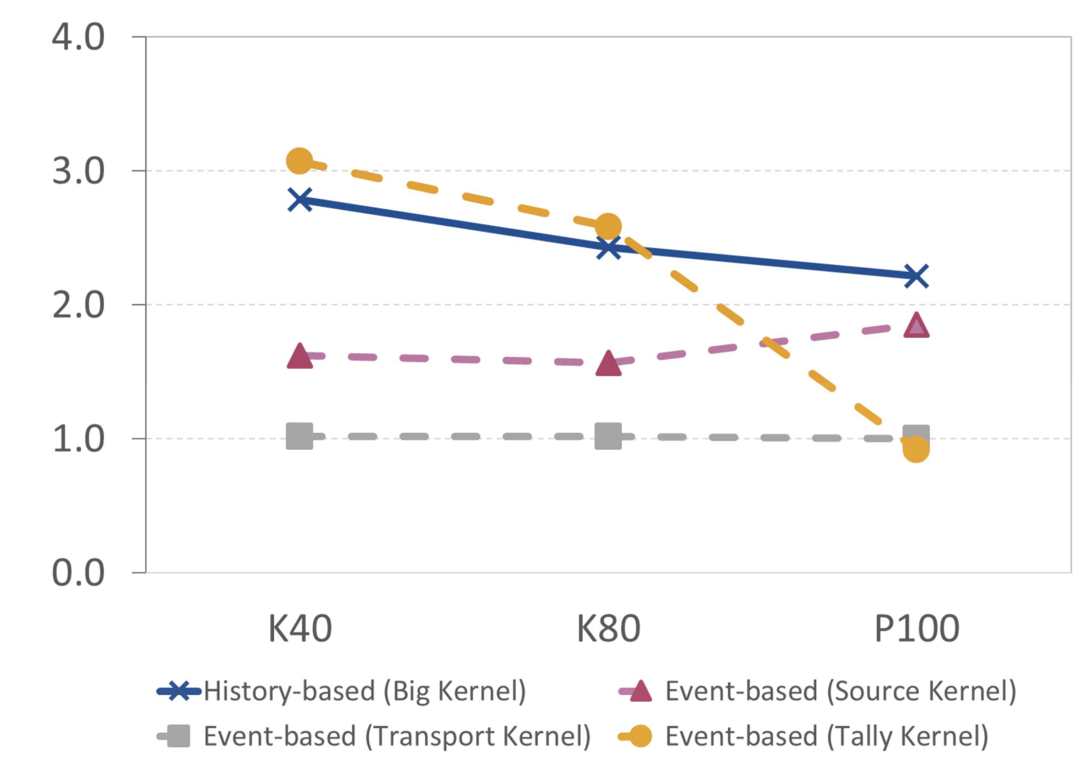
Average Speedup of CUDA over Kokkos (Total Runtime)

- Kokkos was more effective for event-based transport than for history-based transport



Average Speedup of CUDA over Kokkos (Key Kernels)

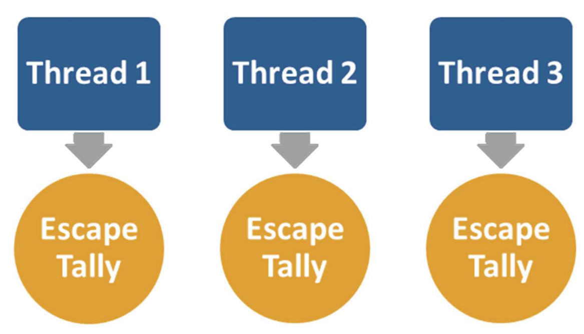
- Only the Transport Kernel had similar runtimes between the Kokkos and CUDA versions
- CUDA was consistently faster for the Source Kernel due to its effective use of constant memory



Tally Kernel

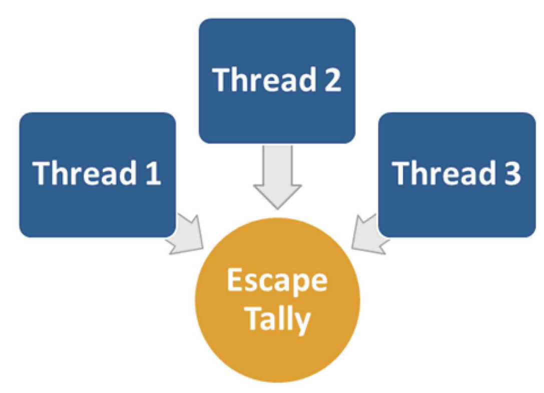
Tally Replication

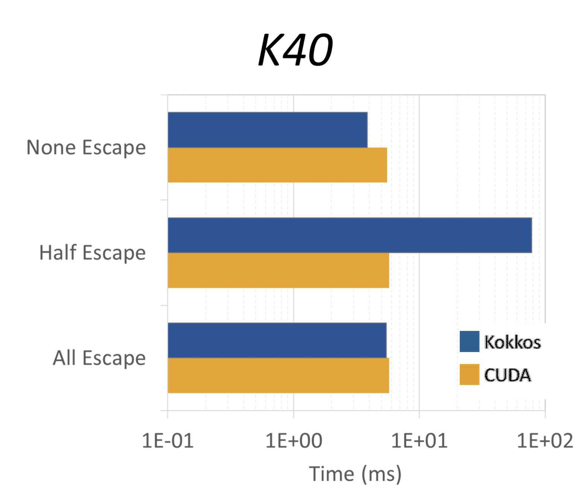
- Kokkos and CUDA had similar runtimes for the Tally Kernel when tallies were replicated



Atomics

- Kokkos with atomics was faster if no photons escaped because no atomic operations were needed
- CUDA's warp shuffle was 14x faster on the K40 when half the photons escaped



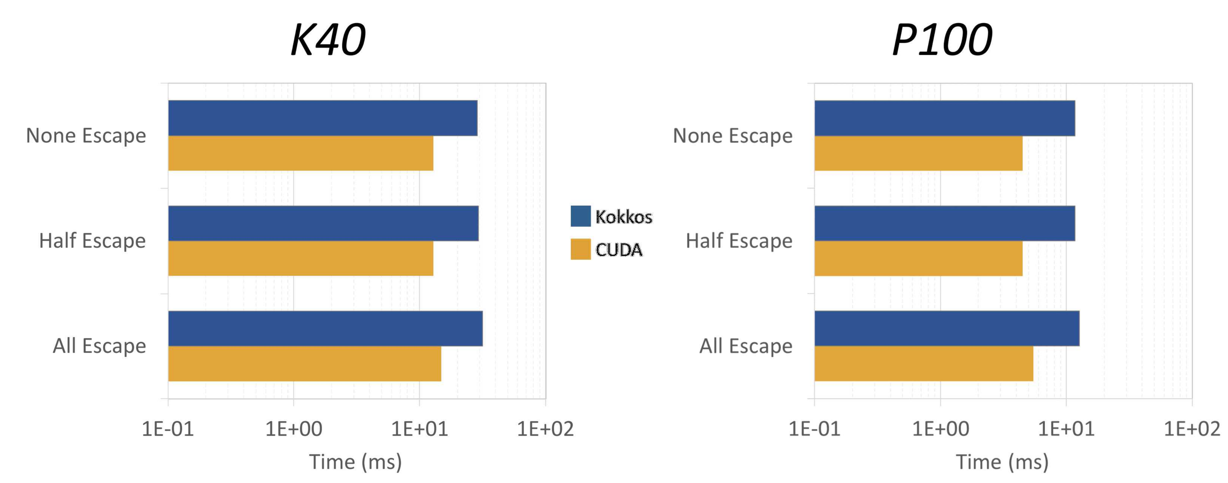


Big Kernel

- Kokkos version of the Big Kernel was slower than the CUDA version for all variations considered
- Direct access to CUDA features is more important for history-based transport

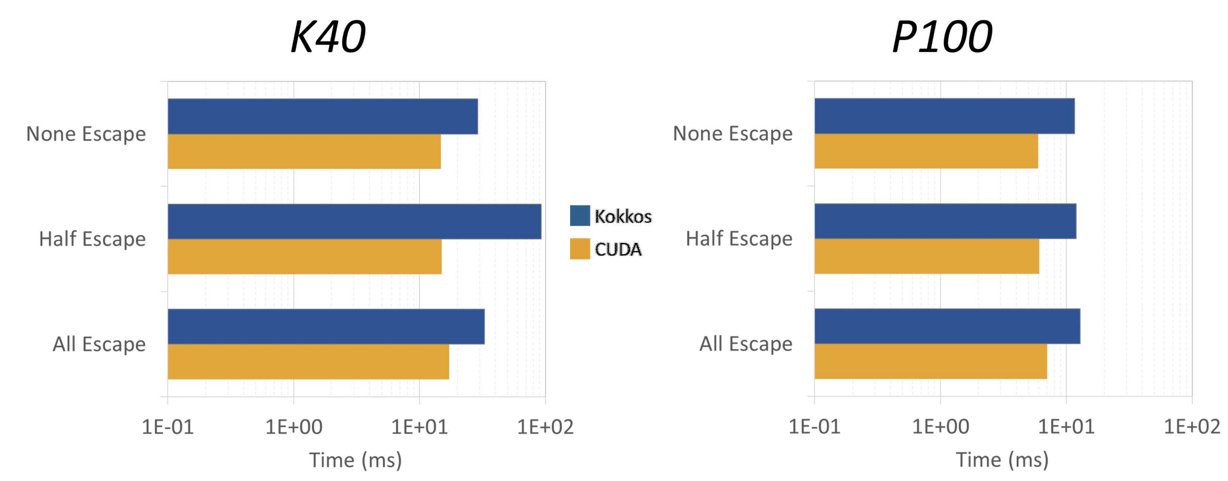
Big Kernel with Tally Replication

- Kokkos is slower because the Big Kernel needs too much data for Kokkos to use constant memory



Big Kernel with Atomics

- Kokkos was 6 times slower on the K40 when half the photons escape due to using atomics for tallying instead of warp shuffle



Conclusions

- Total runtime using Kokkos was competitive with CUDA in most cases – especially on the Tesla P100
- Direct access to CUDA's constant memory and warp shuffle features noticeably improved performance for the Big Kernel, Source Kernel, and Tally Kernel
- Kokkos performance could be improved by restructuring the code to get indirect access to CUDA features hidden under the abstraction
- Future work will compare Kokkos and CUDA on a more complex Monte Carlo transport problem

Acknowledgements

The authors would like to thank Frank Angers from the University of Michigan for obtaining all the performance results on the K40, K80, and P100.

This work was supported by the Advanced Technology Development and Mitigation (ATDM) and Laboratory Directed Research and Development programs at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.