Experiences with CUDA Streaming in Teton's Linear Sweep

Lawrence Livermore National Laboratory

Steven Rennich*, Teresa Bailey, Aaron Black, Peter Brown, Robert Chen, Terry Haut, Adam Kunen, John Loffeld, Paul Nowak, Bujar Tagani, Ben Yee

Lawrence Livermore National Laboratory, *NVIDIA

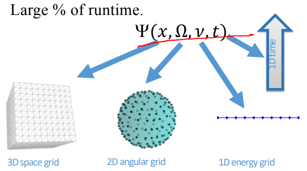
Introduction

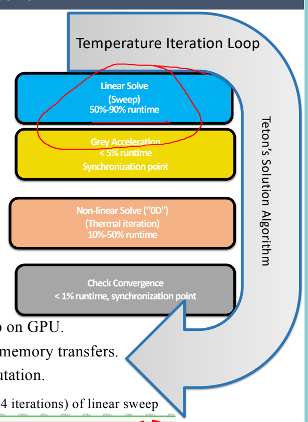
While GPUs are enabling the scaling of new performance peaks, their limited memory sizes are still an obstacle to codes using large quantities of data. Teton, a thermal radiative transfer code, reads and writes data many times the size of GPU memory, which results in more memory transfer overhead. To alleviate this overhead, the Teton team prototyped a CUDA-C streaming version of its linear sweep, that enables more data to be processed than the GPU can usually hold. This capability allows Teton to run larger problem sizes, and increases performance by overlapping memory transfers with computation.

Background

Linear sweep characteristics:

- Large number of unknowns.
- Dependencies between variables.
- Large % of runtime.





Fortran Implementation:

- Uses OpenMP 4.5 target offload to run sweep on GPU.
- "Mapped" host-to-device and device-to-host memory transfers.
- Memory transfers not overlapped with computation.

Figure 1. Nvidia Performance Profile of one cycle (with 4 iterations) of linear sweep

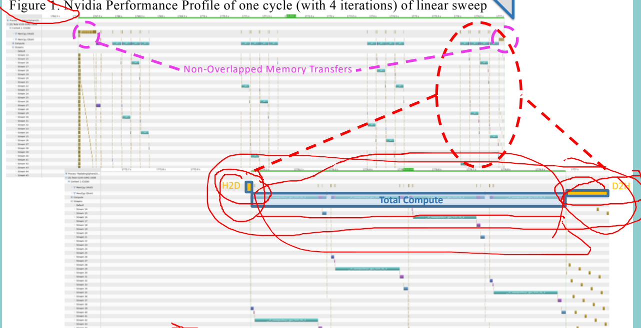


Figure 2. Nvidia Performance Profile of last iteration of linear sweep

Standard Linear Sweep (OpenMP Target Offload) Last Cycle Timings (seconds)					
	iter 1	iter 2	iter 3	iter 4	subtotals
H2D	0.44304131	0.003409	0.00337925	0.0040936	0.45392316
Total Compute	0.9736103	0.86929491	0.86959245	0.91361693	3.62611459
D2H	0.00324177	0.00312492	0.00253443	0.12504085	0.13394197

Cycle Statistics:
% Overhead of H2D+D2H vs. Total Compute = 16.21%

Modifications for Streaming

Streaming code design:

- CUDA-C conversion with Fortran interface.
- Sweep variable sets within an SM to minimize synchronization overhead.
- Asynchronously stream data through GPU to minimize transfer overhead.
- As much as possible, keep sweep logic intact.

Implementation details:

- Each sweep kernel computes 1 angle, and each angle is associated with a specified number of groups (groups per block).
- The number of groups per block needs to be scaled by the user such that GPU memory usage is not exceeded.
- Each hyperplane contains a zone loop.
- Zones are batched to scale with the number of groups per block, into chunks. The sweep kernel logic iterates over these chunks.
- A small amount of data is directly copied to the GPU per kernel launch. Although this is not optimal, the copies are small enough to be overlapped with computation. Further optimization or caching of these data structures would have a modest impact on performance.
- Converted initialization, update, and 2 helper functions into separate streaming kernels.

	Fortran OpenMP 4.5	Streaming CUDA-C
Shared memory usage	8.6KB	96KB (max)
Register count per GPU thread	128	76
Threads per block	512	128

Notes:

- Specifying shared memory variables in OpenMP 4.5 is difficult. Can be specified in 5.1.
- Streaming register count could be reduced further with more optimization of temporary variables.
- Streaming threads per block was capped at 128, but could be increased to ~800.
- Streaming sweep algorithm based on original non-parallelized version of sweep.

Problem Configuration

3D radiating sphere:

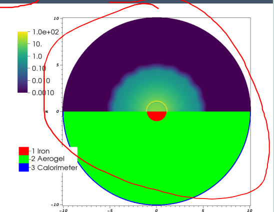
- 100K zones, 16 groups, 80 angles

Mapping to GPU:

- 4 energy groups/SM
- 20 simultaneous angles (kernels)

1 node:

- 2 IBM POWER9 CPUs, 256GB memory
- 4 NVIDIA Volta GPUs, 16GB memory



Results

Figure 3. Nvidia Performance Profile of one cycle (with 4 iterations) of streaming linear sweep

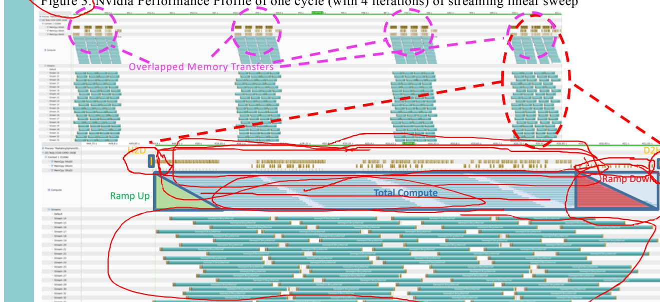


Figure 4. Nvidia Performance Profile of last iteration of streaming linear sweep

Terminology:
Ramp = time during computation when not all kernels active
Total Compute includes Ramp Up and Ramp Down

Cycle Statistics:
% Overhead of H2D+D2H vs. Total Compute = 0.81%
% Overhead of Ramp Up vs. Total Compute = 12.92%
% Overhead of Ramp Down vs. Total Compute = 14.25%
% Overhead of Ramp Up + Ramp Down vs. Total Compute = 27.97%

Streaming Linear Sweep (CUDA-C) Last Cycle Timings (seconds)					
	iter 1	iter 2	iter 3	iter 4	subtotals
H2D	0.00719266	0.00674312	0.00719266	0.00674312	0.02787156
Ramp Up	0.14340368	0.14879817	0.14205505	0.14924771	0.58350461
Total Compute	1.1576	1.08924	1.08654	1.18454	4.51792
Ramp Down	0.14025689	0.14879817	0.14250459	0.2121835	0.64374315
D2H	0.00179817	0.00179817	0.00179817	0.00314679	0.0085413

- Ramp time scales to the amount of GPU memory used (time required to fill GPU memory), and is the same regardless of problem size.
- Ramp down can be considered the streaming overhead (14.25% per cycle) for this problem configuration, which is similar to the OpenMP overhead (16.21%).

Scaling Configurations (3D, 16 groups, 4 ranks)				
	150k zones	300k zones	600k zones	1.2M zones
80 SMs used	8 groups/SM 40 angles	4 groups/SM 20 angles	2 groups/SM 10 angles	1 group/SM 5 angles
40 SMs used			4 groups/SM 10 angles	2 groups/SM 5 angles
20 SMs used				4 groups/SM 5 angles

GPU Memory Usage Per Configuration				
	150k zones	300k zones	600k zones	1.2M zones
80 SMs used	8.7 GB	8 GB	8.3 GB	8.5 GB
40 SMs used			8.3 GB	8.5 GB
20 SMs used				8.5 GB

Grind Time Per Configuration (unknowns/second)				
	150k zones	300k zones	600k zones	1.2M zones
80 SMs used	60160000	63333333	56655737	41103230
40 SMs used			43746835	34912985
20 SMs used				25125240

Figure 5. Scaling study of streaming linear sweep, maintaining constant GPU memory usage, while increasing zone count

- Streaming implementation is bounded by system memory, rather than GPU memory.
- 4+ groups/SM enables coalesced memory accesses for better performance.
- Spreading problem over more SMs begets better throughput.
- Can fit problems onto GPU by scaling SM use (# angle kernels), and/or groups per SM.

Future Work

- Find limit of maximum number of zones which can be handled by streaming.
- Integrate streaming sweep as option into main codebase.
- Implement Fortran OpenMP 5.1 sweep with streaming and shared memory.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.