LA-UR-23-32643

Approved for public release; distribution is unlimited.

Title:	Monte Carlo Transport Computational Summit-Kickoff
Author(s):	Long, Alex Roberts
Intended for:	Monte Carlo Computational Summit, 2023-10-25/2023-10-26 (South Bend, Indiana, United States)
Issued:	2023-11-06

Monte Carlo Transport Computational Summit—Kickoff

Alex Long, Los Alamos National Laboratory

10/25/2023

LA-UR-2023-XXXXX

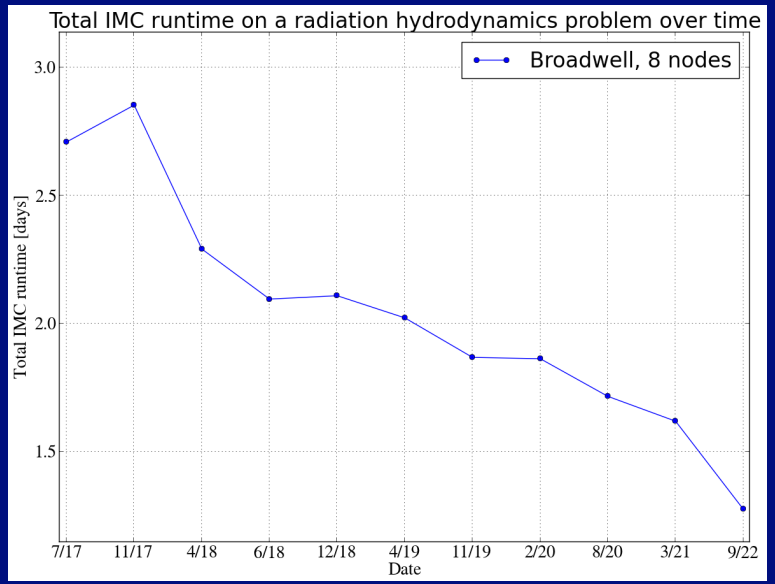
A hearty welcome to all!

- Thank you all for coming to the inaugural Monte Carlo Computational Summit!
- A brief history
  - Not many venues where DOE NNSA labs, DOE Office of Science labs and academia get to interact in a less formal (i.e. not a conference) environment
  - This meeting was suggested by Dave Richards pre-pandemic, here we are, finally
  - Thanks to Ryan McClarren and CEMENT PSAAP center for hosting
- This talk is meant to foster discussion--even if I am very wrong and you all hate this talk, I hope it will be a jumping off point

“What kind of computer would you like?”

- This is a question I’ve heard multiple times
- How close are we to answering it as a community and do we need to?

THIS SPACE INTENTIONALLY LEFT BLANK, SORRY GORDON



The Monte Carlo transport community continues to make impressive progress on performance

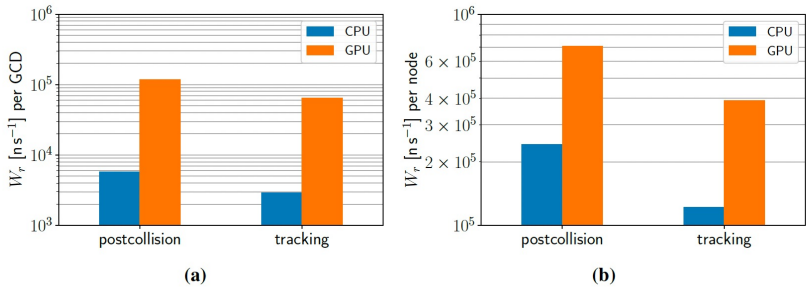


Figure 7: Work rate performance of SMR excore problem on Summit comparing (a) per GCD (core/GPU) and (b) per node.

Royston et al, 2023

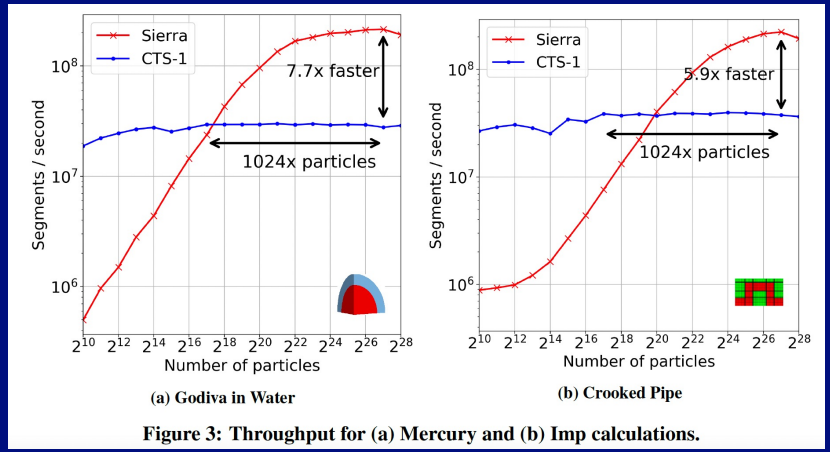


Figure 3: Throughput for (a) Mercury and (b) Imp calculations.

Pozulp et al, 2023

The Monte Carlo transport community continues to make impressive progress on performance

Table 2: Observed performance on the BEAVRS benchmark using the Intel Core i7-1260P processor for various thread configurations.

P-cores	E-cores	Threads/core	Throughput [particles/sec]
0	1	1	5334
0	4	1	19910
0	8	1	35437
0	8	2	34885
1	0	1	9316
4	0	1	27186
4	0	2	31939
4	0	4	31889
4	8	2/1	48962

Romano et al, 2023

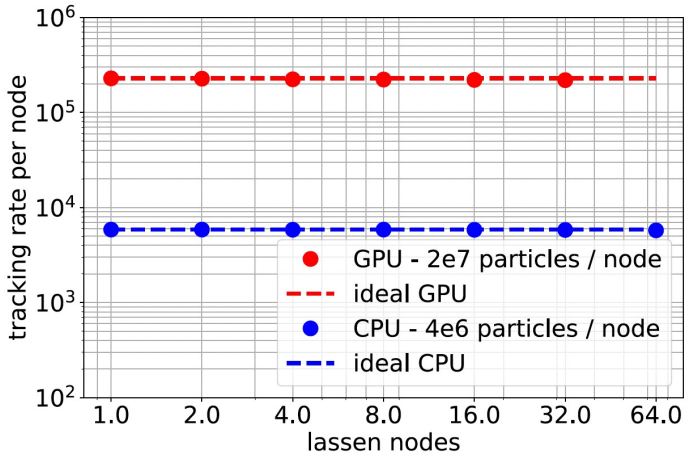


Figure 10: Neutron tracking rate on SMR problem.

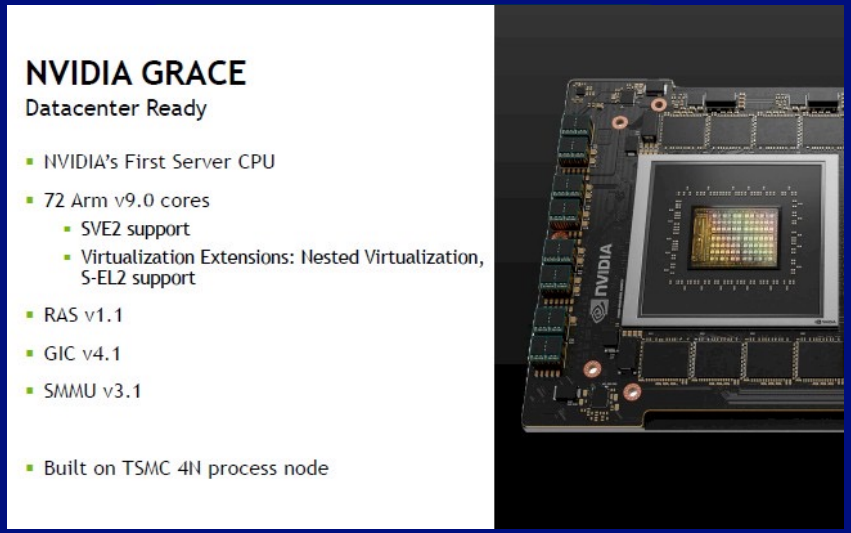
Reynolds et al, 2023

Have we found our computer in GPUs?

Maybe?

- MC transport codes reporting speedups of 20x
- Frontier is solving variety of problems 5.5x faster than Summit (Budiardja et al, 2023)
- CUDA/HIP/OneAPI interoperability
- True unified memory

Have we found our computer in GPUs?



Maybe not?

- Low-power CPUs run OpenMC at roughly 0.7x full-power CPUs
- Exciting early results on Grace CPUs
- Proliferation of "efficiency cores"?
- Scalable Vector Extension (SVE) instruction

Have we found our computer in GPUs?

Maybe?

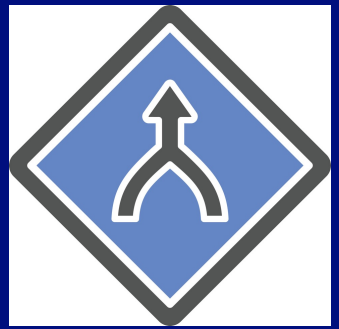
- MC transport codes reporting speedups of 20x
- Frontier is solving variety of problems 5.5x faster than Summit (Budiardja et al, 2023)
- CUDA/HIP/OneAPI interoperability
- True unified memory

Maybe not?

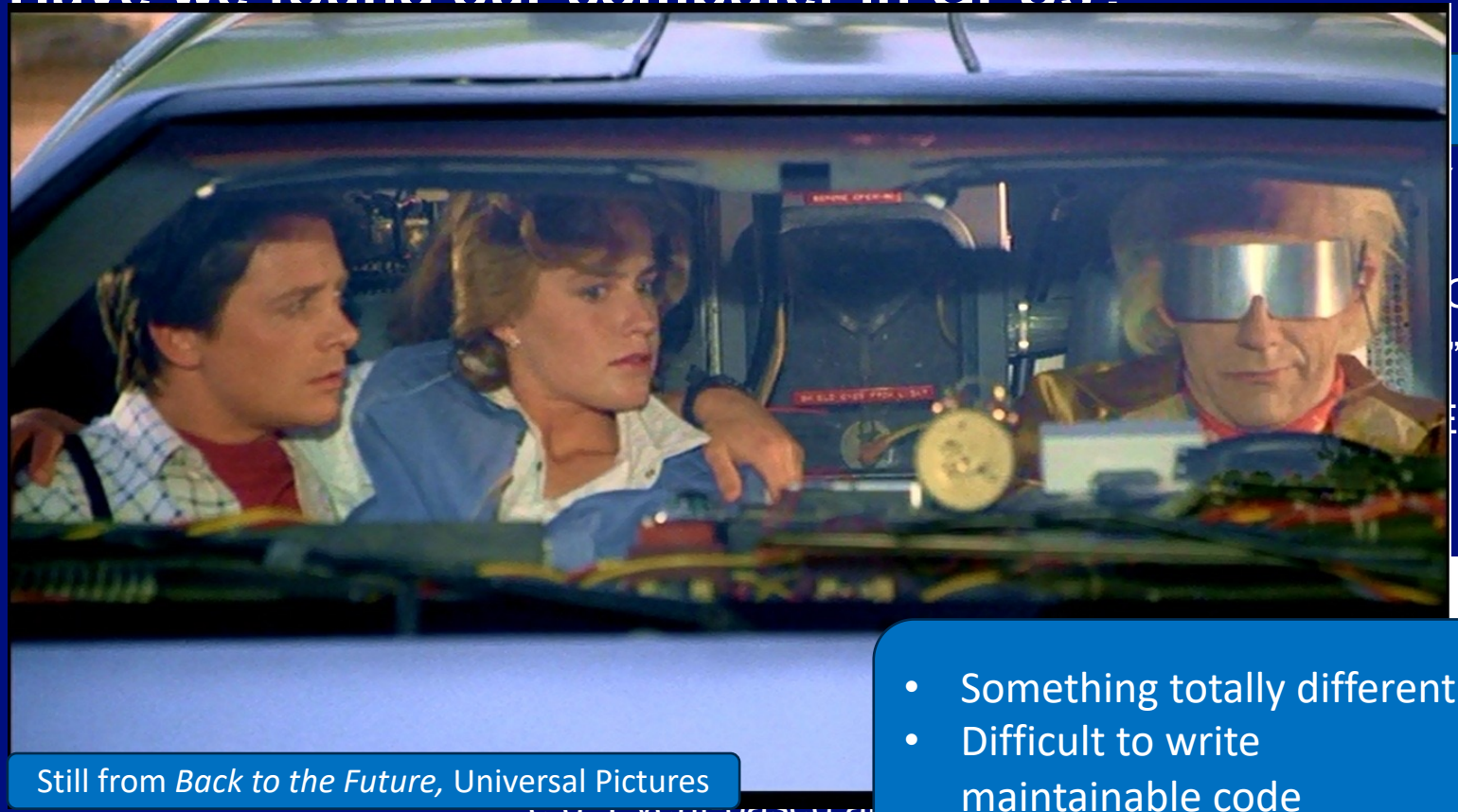
- Low-power CPUs run OpenMC at roughly 0.7x full-power CPUs
- Exciting early results on Grace CPUs
- Proliferation of “efficiency cores”?
- Scalable Vector Extension (SVE) instruction

Converging?

- Do wide vector lanes on CPUs look enough like SIMT GPUs for them to be the same?
- e.g. Event-based algorithms



Have we found our computer in GPUs?



Still from *Back to the Future,* Universal Pictures

problems

CPUs

"?

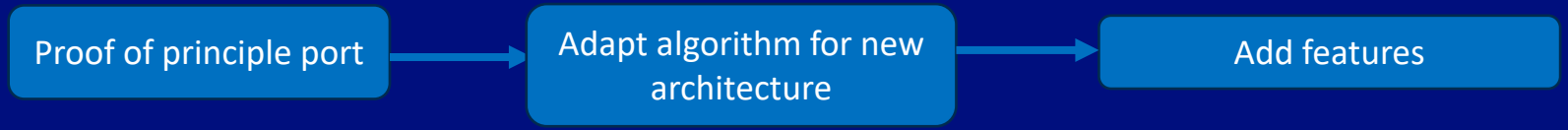
E)

- Something totally different!
- Difficult to write maintainable code

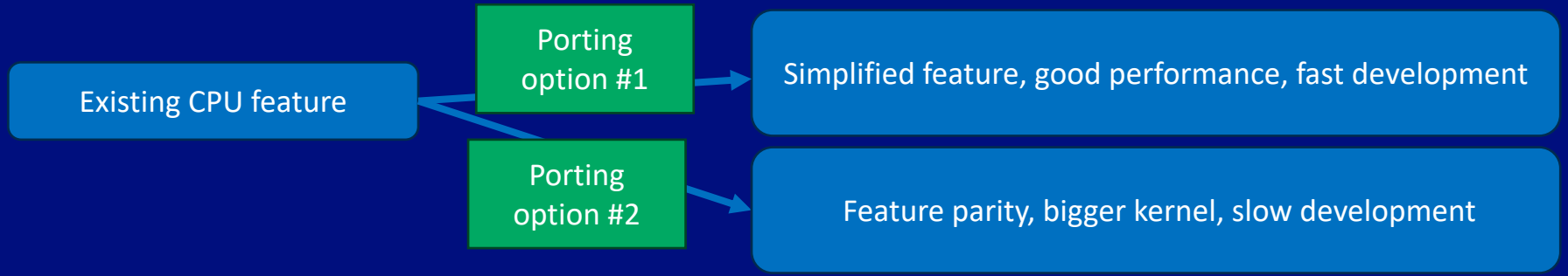
e.g. Event based a

Potential Threats—Feature Additions

- Often, GPU code seems to evolve like this:



- How does a feature get implemented while preserving performance?



- Do portability layers (e.g. Kokkos, RAJA, OpenMP) help with this?

Potential Threats—Maintainability

- Example: I have transport and acceleration method ported to the GPU
  - On the CPU, code acceleration routines live inside of the core transport loop
  - On the GPU, the best performance was obtained by splitting these in to two kernels
- Team has rightly raised concerns about duplication of transport code
- What's the best solution?

Where do we go?

- I’ve been disappointed by early performance of Crossroads
  - Big multicore machine, 3x the power, 3x the performance
  - Most of all, I don’t see where to go next on this machine
- Contrast to GPUs and potential paths forward
  - Reworking algorithms for smaller register footprint
  - Data structure optimizations
  - Streams, event-based, persistent thread approaches
- AI
- Reduced-precision
- Old/new methods
  - Quasi Monte Carlo
- Task-based frameworks

End matter

- Does DOE need more definitive proof or are they convinced GPUs are the wave of the future?
- What is the value in clearly demonstrating **why** Monte Carlo transport sees dramatic speedup on GPUs?
- Our actual resource is people who are interested in transport and computational performance