NAVAL NUCLEAR LABORATORY | Technology

Accelerating Transport Codes on Newer Architectures

Alicia Klinvex, David Griesheimer, and Steven Douglass

The Naval Nuclear Laboratory is operated for the U.S. Department of Energy by Fluor Marine Propulsion, LLC,
a wholly owned subsidiary of Fluor Corporation.

What is the problem we're trying so solve?

- We want to speed up cross-section (XS) lookups
  - Can be over 70% of runtime
- On-the-fly Doppler broadening is computationally expensive and would negatively impact the runtime

- We want to go faster so more/better science can be done in the same amount of time
- …operating within lab constraints (staffing, budget, etc)

- We will discuss two codes: MC21 and Bengal/Hobbes

NAVAL NUCLEAR LABORATORY | Technology

How do we plan to go faster?

- General purpose Graphical Processing Units (GP-GPUs)
  - Originally made for rendering graphics (movies/video games)
  - Accidentally very good at parallel computing
  - Commonly found on fastest high performance computers (HPCs) in the world
  - Frequently provide the highest floating point operations per second (FLOPs) per dollar

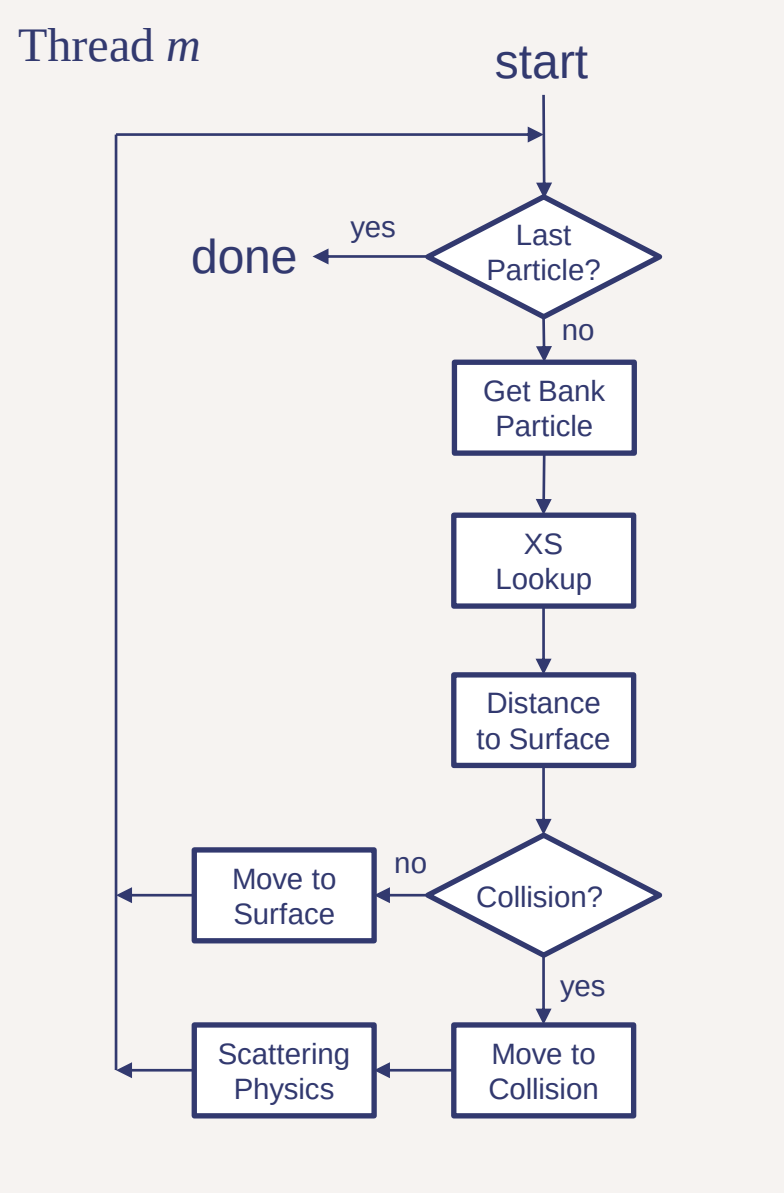
- The GPU Catch-22
  - Why buy GPUs when no codes use them?
  - Why write code for GPUs when we don't own them?
    - This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

NAVAL NUCLEAR LABORATORY | Technology

Why is this hard?

- GPU code looks different from CPU code
  - Cuda, Kokkos/Raja, HIP, OneAPI/DPC++/SyCL, OpenMP offloading
- GPUs introduce a separate memory space
  - Requires communication like message passing interface (MPI)
  - “Automagic” data movement offers terrible performance
  - Have relatively small memory
    - Common advice is to go all-GPU or no-GPU because of communication costs
    - May not be able to port entire code to GPU
    - Asynchronous computing (Cuda streams) is not supported by all programming models
- Useless without a single instruction multiple data (SIMD) algorithm
  - Need to bundle a bunch of similar operations together, then send it to the GPU
  - *This has been the hardest part for us*

NAVAL NUCLEAR LABORATORY | Technology

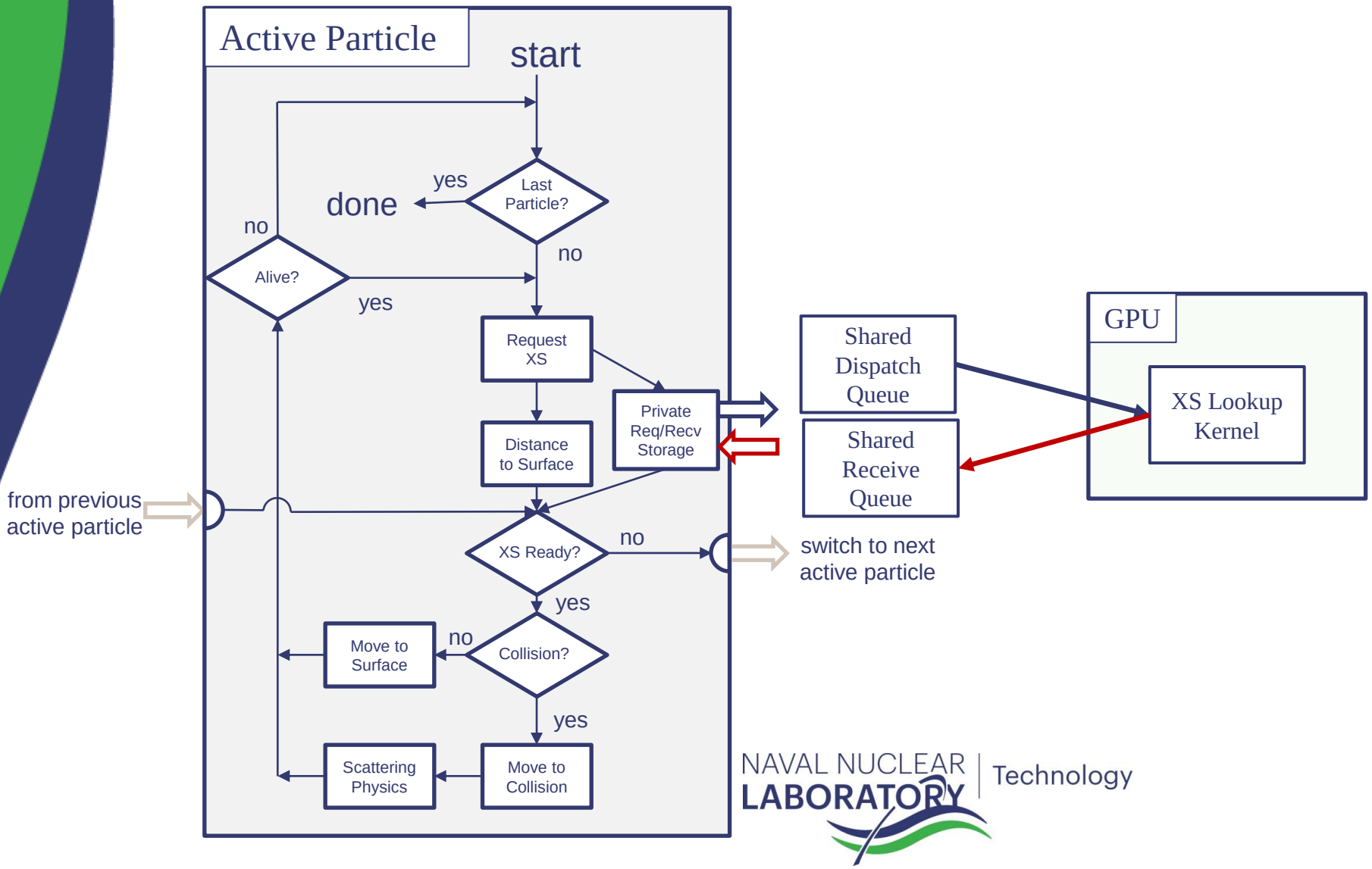


The algorithm we mean to optimize

- MC21 uses MPI+OpenMP
  - 1 MPI process per compute node
  - 1 thread per CPU core
- Particles are divided among available threads on all nodes.
- Each thread processes particles *independently*, following standard Monte Carlo (MC) process.
- Cross section lookup is *independent* of the distance to surface calculation

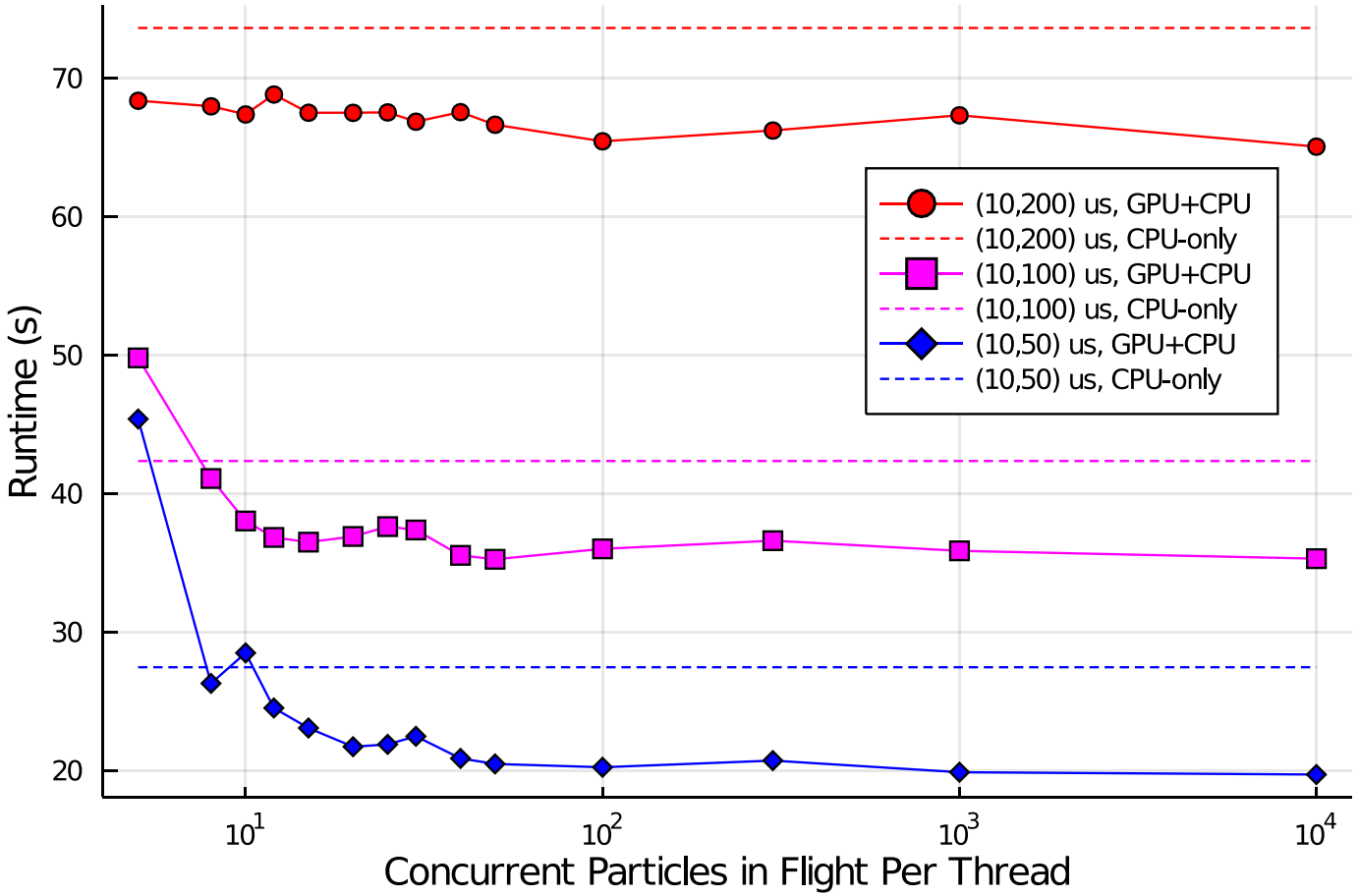
NAVAL NUCLEAR LABORATORY | Technology

The GPU port



NAVAL NUCLEAR LABORATORY | Technology

But *is it fast?*



NAVAL NUCLEAR LABORATORY | Technology

What did we learn?

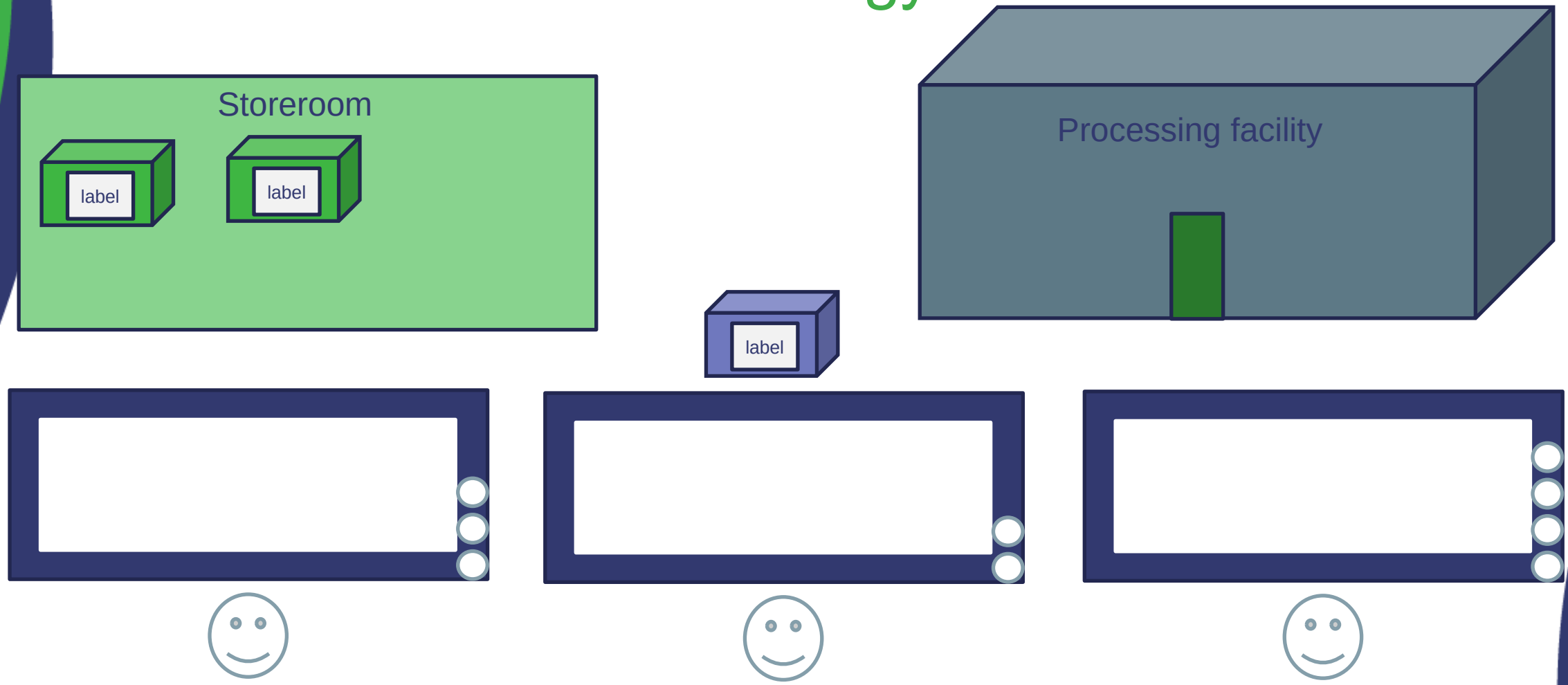
- GPUs can make SIMD parallel algorithms fast, but if you don't have one of those you're out of luck
  - The hardest part for us was batching the cross section lookups to throw on the GPU
  - The Cuda programming was comparatively easy
- If you can move all operations to a GPU, you probably should
- The easier a programming model is to use, the worse performance you can expect
- You should use abstraction layers if you can, but not if you need async operations
  - Maybe someday, Kokkos…

NAVAL NUCLEAR LABORATORY | Technology

Backup slides

NAVAL NUCLEAR LABORATORY | Technology

An analogy…



NAVAL NUCLEAR LABORATORY | Technology

Self-Shielding at Scale in Many-Core Architectures

- Macroscopic cross section generation is essentially the repeated application of:

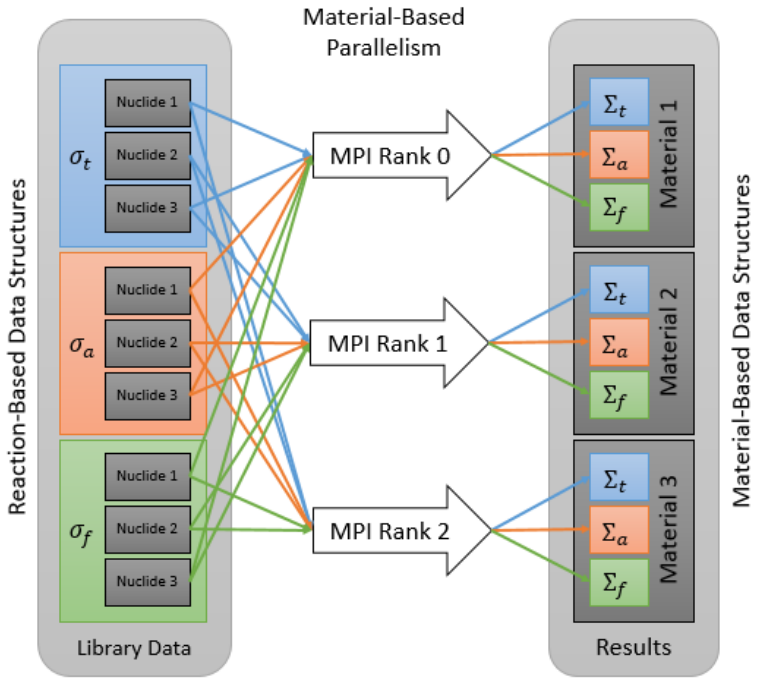
$$\Sigma_m^G \mathrel{+}= N_i^m \times \sigma_i^g \times w_g^m, \qquad g \in G$$

$$\begin{array}{rl} O(100) & \text{Nuclides} \\ \times\, O(100) & \text{Groups} \\ \times\, O(1 \times 10^7) & \text{Materials} \\ \hline O(1 \times 10^{11}) & \text{Per Reaction} \end{array}$$

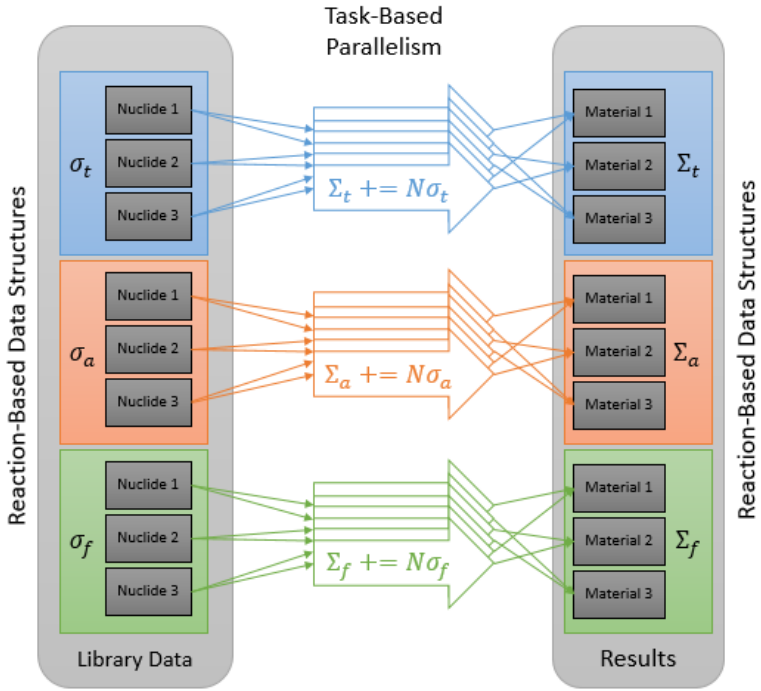
NAVAL NUCLEAR LABORATORY | Technology

Self-Shielding at Scale in Many-Core Architectures

"Physics" Approach



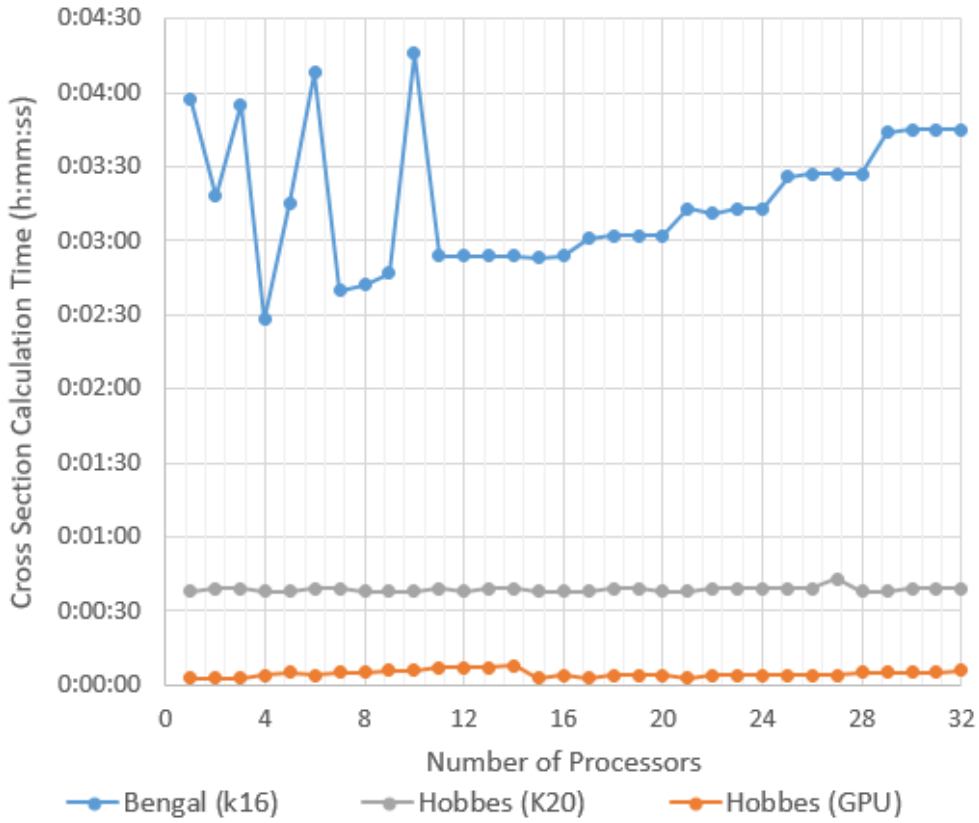
SIMD Approach



NAVAL NUCLEAR LABORATORY | Technology

Self-Shielding at Scale in Many-Core Architectures

- Weak Scaling Study
  - Approx. 2000 materials / rank
  - GPU calculation for equivalent number of materials
- Optimized memory access pattern leads to improved scaling and performance even in distributed MPI
- GPU performance is *incredibly* promising



NAVAL NUCLEAR LABORATORY | Technology