Rapid Development of a High-Performance Application Using Python

HPC Best Practices Webinar

Wednesday September 2nd, 2026

Joanna Piper Morgan
Nuclear Criticality Safety Division, Lawerence Livermore National Laboratory

Kyle E. Niemeyer
School of Mechanical Industrial and Manufacturing Engineering Oregon State University

Prepared by LLNL under Contract DE-AC52-07NA27344.

The Problem

When developing HPC applications the:

Subject area expert (physicist/engineer)

Numerical methods developer

HPC specific software engineer

Are all the same person

"Still the magnitude of the endeavor to compute on massively parallel devices must not be underestimated. Some of the tools and techniques needed are: A high-level language and new architecture able to deal with the demands of such a sophisticated language (to the relief of the user);...and A fresh look at numerical analysis and the preparation of new algorithms..."

—Nicolas Metropolis, *1987*

Goals

We need to write a new Monte Carlo application specifically for time dependent methods, specifically for heterogeneous HPCs

Goals:

- ❑Write one piece of code and execute on GPUs CPUs and whatever
- ❑Code base specifically designed for rapid methods development
- ❑Have good coding practices for open-source development

The Impact of Large Language Model AI

AI has changed the useability component of this research is dramatic

A human still needs to read the code (in my opinion)

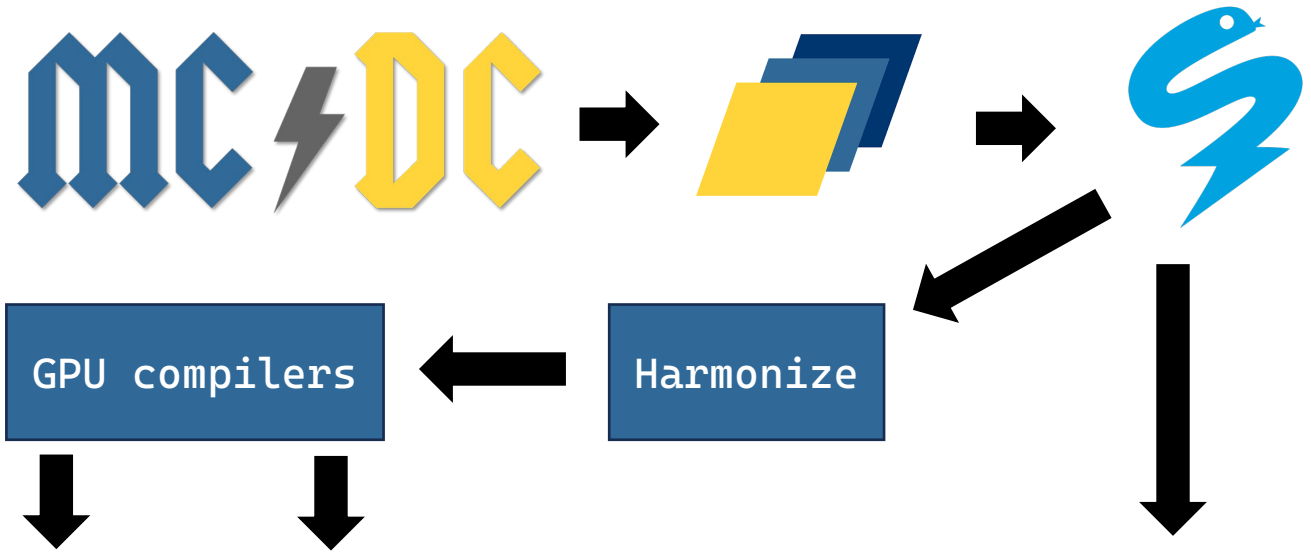
Having human readable code is only going to get more important as time goes on

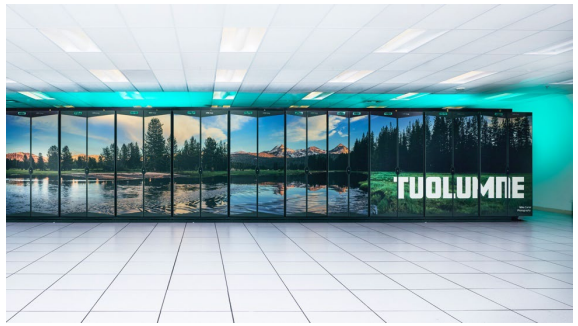
What others do

- Python based:
  - Templating Engines
  - pyKokkos
  - looPy
- Honorable Mentions:
  - Kokkos/Raja other portability layer
  - Julia

Broad Overview of Software Design of MC/DC

- Written entirely in Python and initial development can be done in pure Python (Python as a DSL)
- MC/DC compiles with Numba to approach C-like speeds
- Numba-CUDA/Numba-HIP produces device functions
- Device functions then compiled with the Harmonize GPU runtime for optimized execution
- Just in time compiled





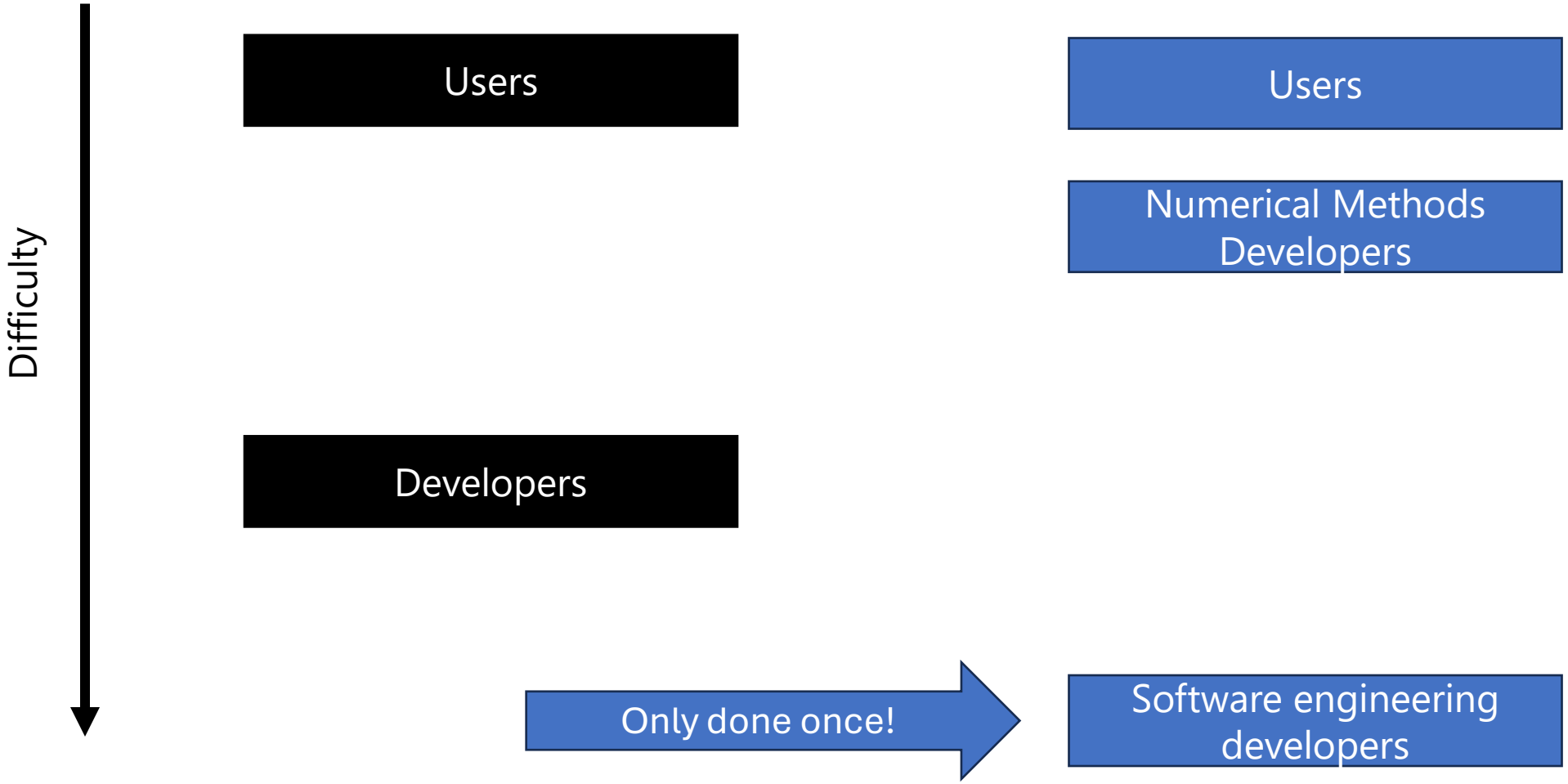
Tuolumne (AMD GPUs)

Lassen (Nvida GPUs)

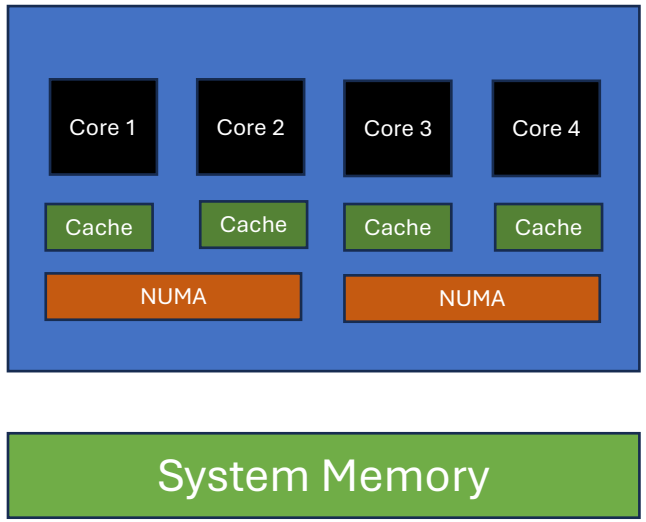
Dane (CPUs)

For the Numerical Methods Developer Python-Numba acts as a DSL

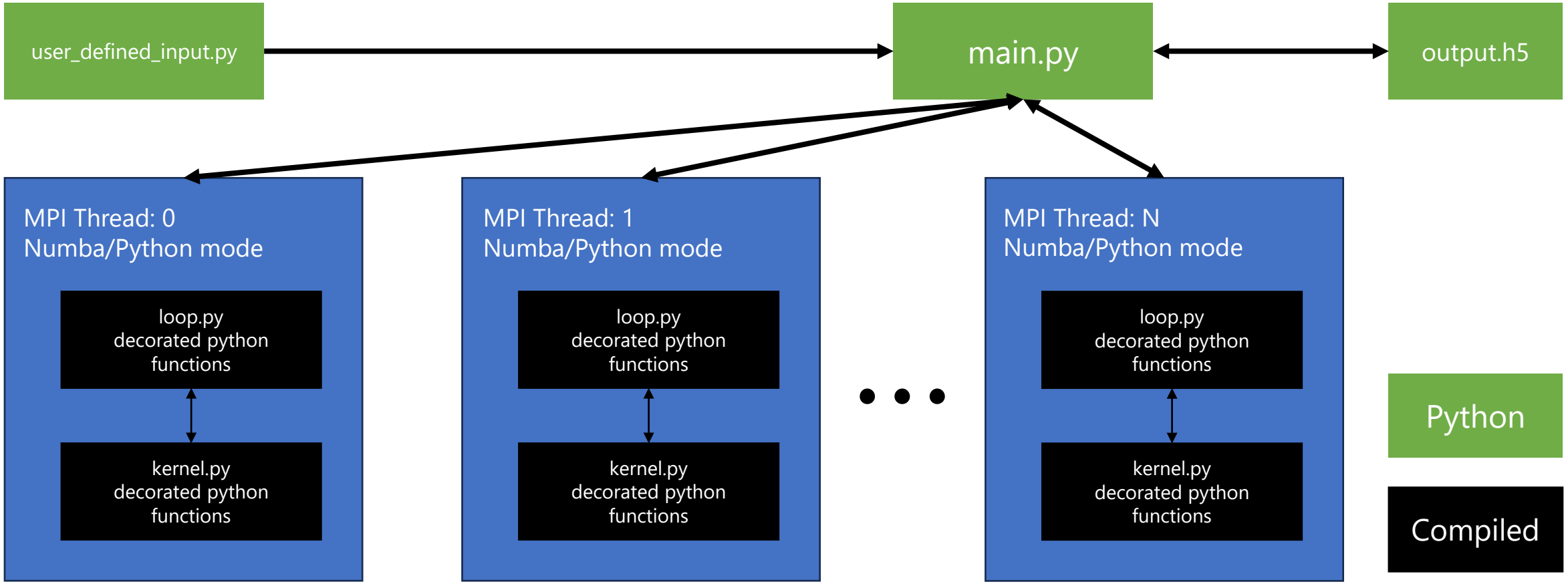
- Domain specific languages can ease support
- The restrictions on Python when compiling in Numba CPU and or GPU mode act like a DSL
  - Off the shelf
  - Easy to set up
  - Works for most scientific applications
- Can be used with other HPC libraries


"Normal" Codes
MC/DC
Difficulty
Users
Developers
Users
Numerical Methods Developers
Only done once!
Software engineering developers

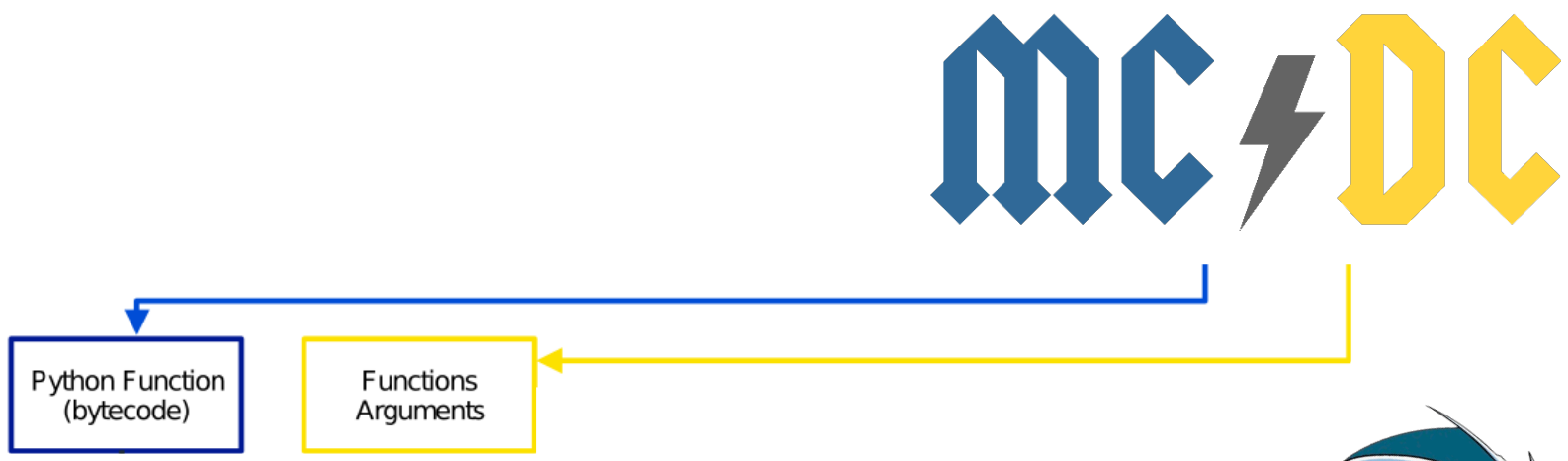
Parallelism on a CPU



MPI Parallelism

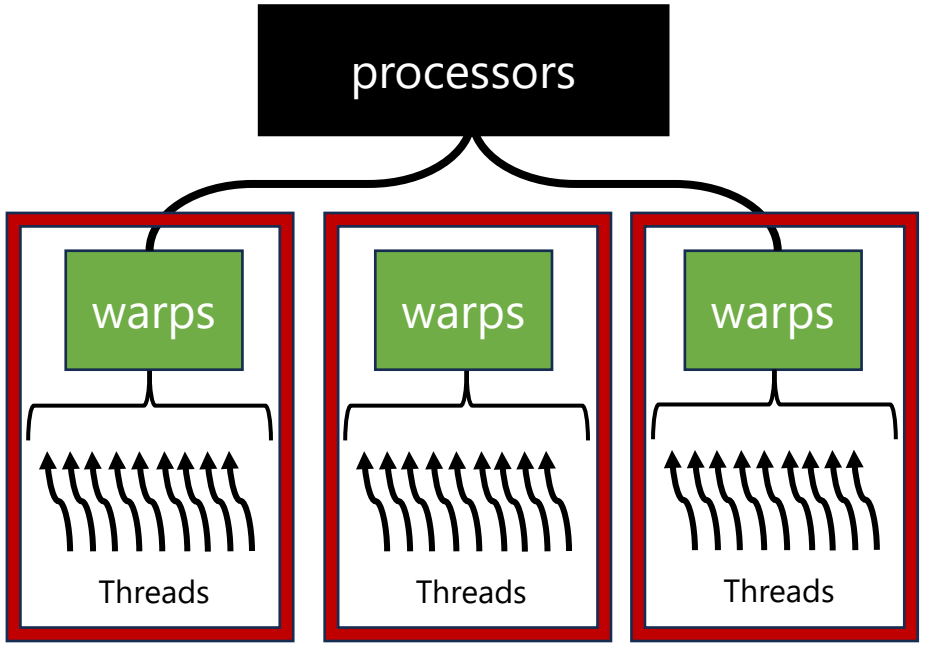


MC/DC on CPUs (x86, ARM)



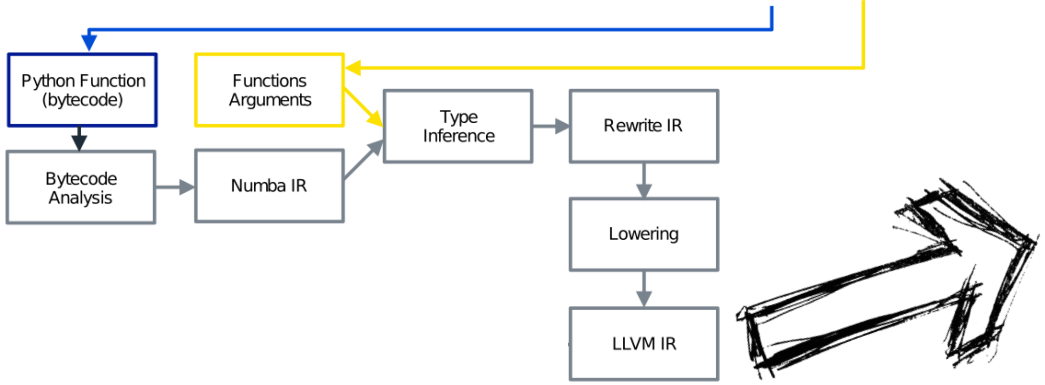
Graham Markall. *The Life of a Numba Kernel: A Compilation Pipeline Taking User Defined Functions in Python to CUDA Kernels*. Online. Medium. Jun 16, 2020.

Parallelism on a GPU



Numba Just in Time (jit)

```
@nb.njit
def add_one(value):
    return value + 1
```



add_one.inspect_llvm()

```
define i32 @add_one(double* noalias nocapture
writeonly %retptr, { i8*, i32, i8*, i8*, i32
}** {

entry:
    %.6 = fadd double %arg.value, 1.000000e+00
    store double %.6, double* %retptr, align 8
    ret i32 0

}
```

```
numba.cuda.compile_for_current_device()
numba.hip.compile_llvm_ir_for_current_device()
```

Compilation to an Outside Library

```python
# Python device function
def add_one(value):
    return value + 1
```

AMD LLVM-IR or Nvidia PTX

AMD ROCm

NVIDIA CUDA

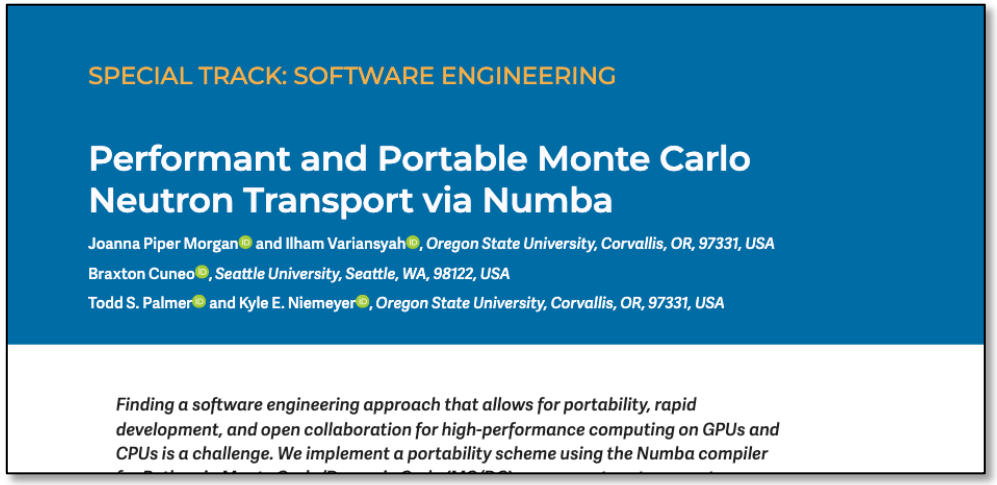
Shared Compiled Object Library

```
# extern declaration (eventually from python source)
extern "C" __device__ int add_one(int value);

__global__ void increment_kernel(int *array, int size) {
    int count = gridDim.x * blockDim.x;
    int tid = blockIdx.x * blockDim.x + threadIdx.x;
    for (int i=tid(); i<size; i+=count) {
        array[i] = add_one(array[i]);
    }
}
```

Executable with Host and Device Code

General overview of the code base



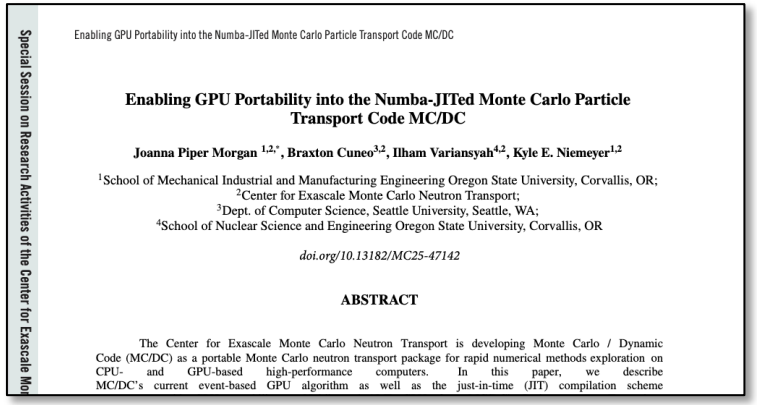
SPECIAL TRACK: SOFTWARE ENGINEERING

Performant and Portable Monte Carlo Neutron Transport via Numba

Joanna Piper Morgan and Ilham Variansyah, *Oregon State University, Corvallis, OR, 97331, USA*
Braxton Cuneo, *Seattle University, Seattle, WA, 98122, USA*
Todd S. Palmer and Kyle E. Niemeyer, *Oregon State University, Corvallis, OR, 97331, USA*

Finding a software engineering approach that allows for portability, rapid development, and open collaboration for high-performance computing on GPUs and CPUs is a challenge. We implement a portability scheme using the Numba compiler

Description of the compilation methodology



Special Session on Research Activities of the Center for Exascale Mo

Enabling GPU Portability into the Numba-JITed Monte Carlo Particle Transport Code MC/DC

Enabling GPU Portability into the Numba-JITed Monte Carlo Particle Transport Code MC/DC

Joanna Piper Morgan [1,2,*], Braxton Cuneo[3,2], Ilham Variansyah[4,2], Kyle E. Niemeyer[1,2]

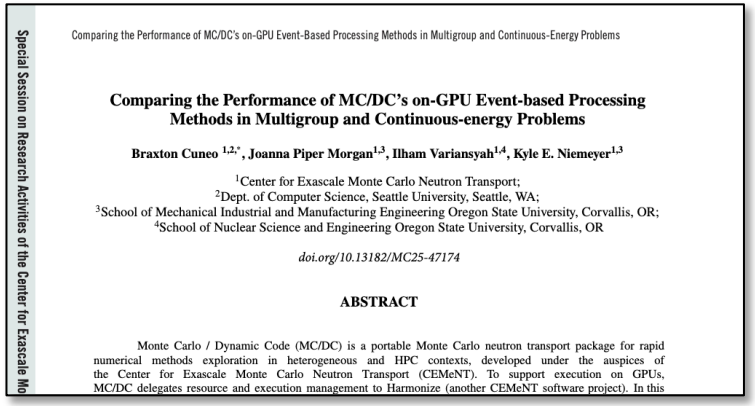
[1]School of Mechanical Industrial and Manufacturing Engineering Oregon State University, Corvallis, OR;
[2]Center for Exascale Monte Carlo Neutron Transport;
[3]Dept. of Computer Science, Seattle University, Seattle, WA;
[4]School of Nuclear Science and Engineering Oregon State University, Corvallis, OR

doi.org/10.13182/MC25-47142

ABSTRACT

The Center for Exascale Monte Carlo Neutron Transport is developing Monte Carlo / Dynamic Code (MC/DC) as a portable Monte Carlo neutron transport package for rapid numerical methods exploration on CPU- and GPU-based high-performance computers. In this paper, we describe MC/DC's current event-based GPU algorithm as well as the just-in-time (JIT) compilation scheme

Description of the asynchronous hardware code generation



Special Session on Research Activities of the Center for Exascale Mo

Comparing the Performance of MC/DC's on-GPU Event-Based Processing Methods in Multigroup and Continuous-Energy Problems

Comparing the Performance of MC/DC's on-GPU Event-based Processing Methods in Multigroup and Continuous-energy Problems

Braxton Cuneo [1,2,*], Joanna Piper Morgan[1,3], Ilham Variansyah[1,4], Kyle E. Niemeyer[1,3]

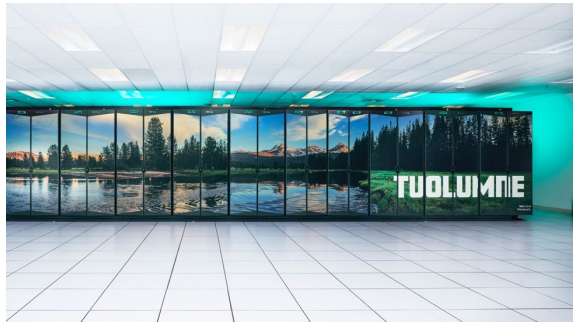
[1]Center for Exascale Monte Carlo Neutron Transport;
[2]Dept. of Computer Science, Seattle University, Seattle, WA;
[3]School of Mechanical Industrial and Manufacturing Engineering Oregon State University, Corvallis, OR;
[4]School of Nuclear Science and Engineering Oregon State University, Corvallis, OR

doi.org/10.13182/MC25-47174

ABSTRACT

Monte Carlo / Dynamic Code (MC/DC) is a portable Monte Carlo neutron transport package for rapid numerical methods exploration in heterogeneous and HPC contexts, developed under the auspices of the Center for Exascale Monte Carlo Neutron Transport (CEMeNT). To support execution on GPUs, MC/DC delegates resource and execution management to Harmonize (another CEMeNT software project). In this

Test Beds



Tuolumne
or Tioga

APU: 4X 24-core AMD EPYC @2.0GHz with an MI300a and 512 GB of **unified storage**

Lassen

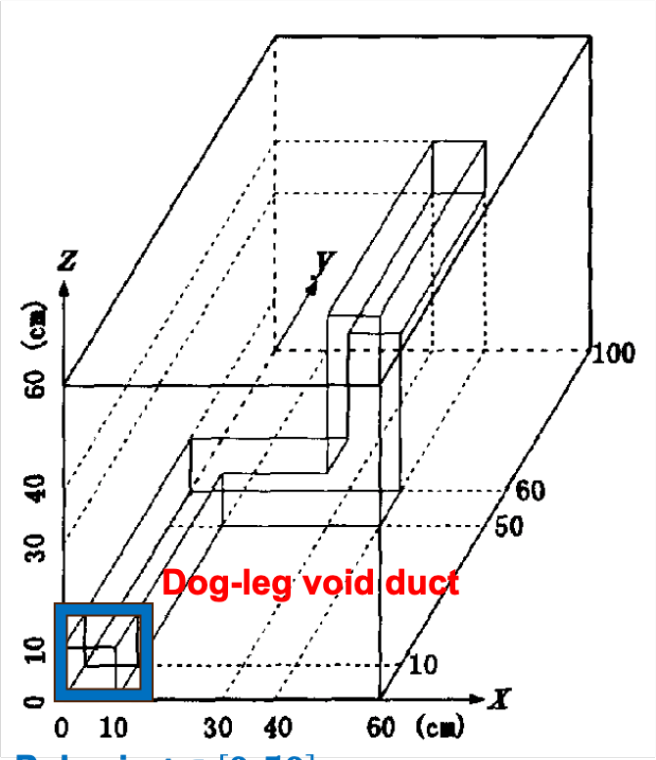
CPU: 2X 22-core IBM POWER9 @3.5 GHz and 256 GB

GPU: 4X Nvidia Tesla V100, 16GB/GPU

Dane

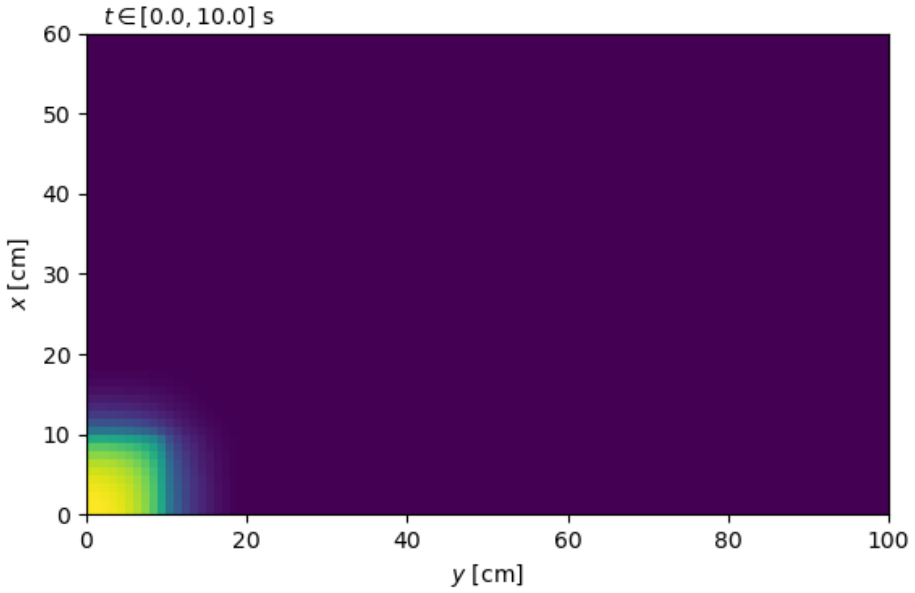
CPU: 2X 56-Core Intel Xeon Sapphire Rapids @2.0 GHz and 256 GB

Time-dependent Kobayashi dog-leg benchmark



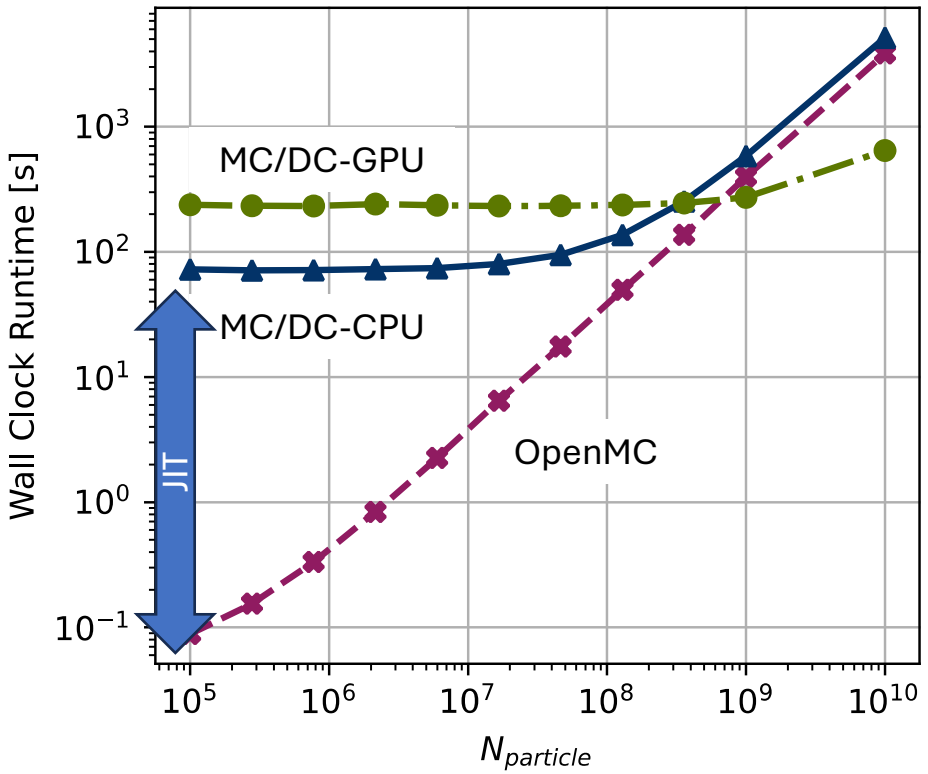
Pulse in $t \in [0, 50]$ s

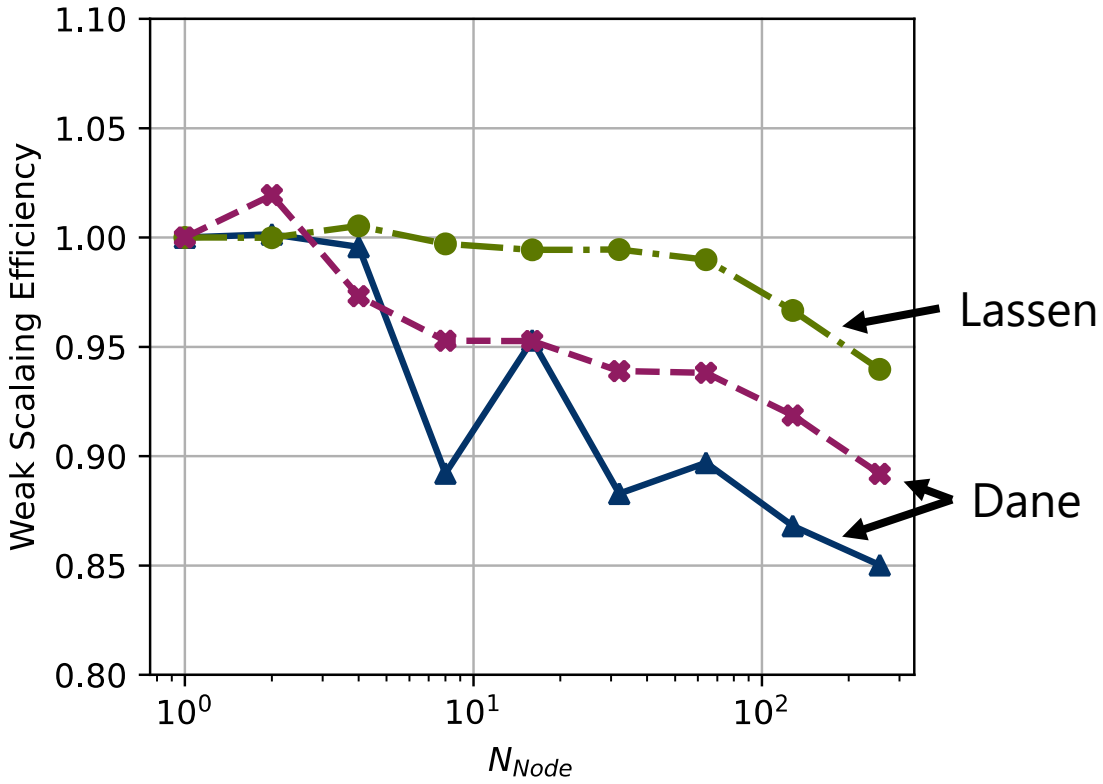
K. Kobayashi, "3-D Radiation Transport Benchmarks for Simple Geometries with Void Regions," OECD/NEA report



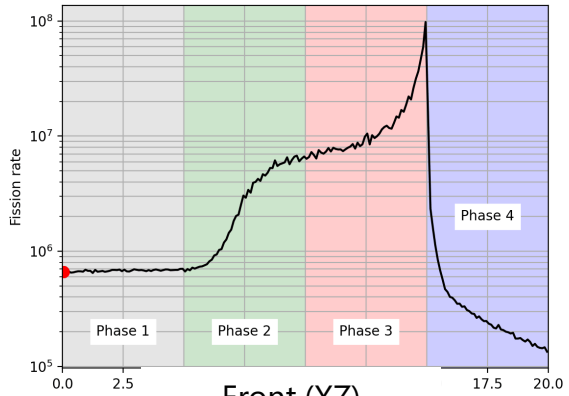
	Dane	Lassen	Tulamne-MI300a
Run time [s] (CPU/GPU)	1574 (-)	973.2 (1.6)	583.2 (2.7)
CPU core equiv	-	415	326

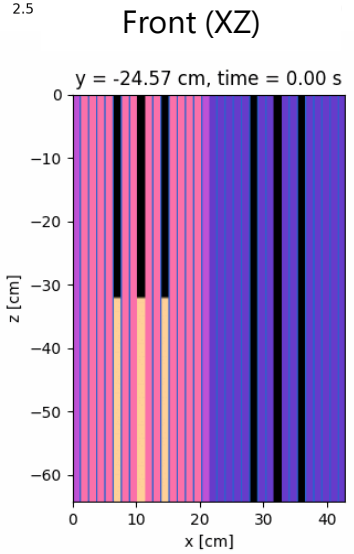
Kobayashi Scaling

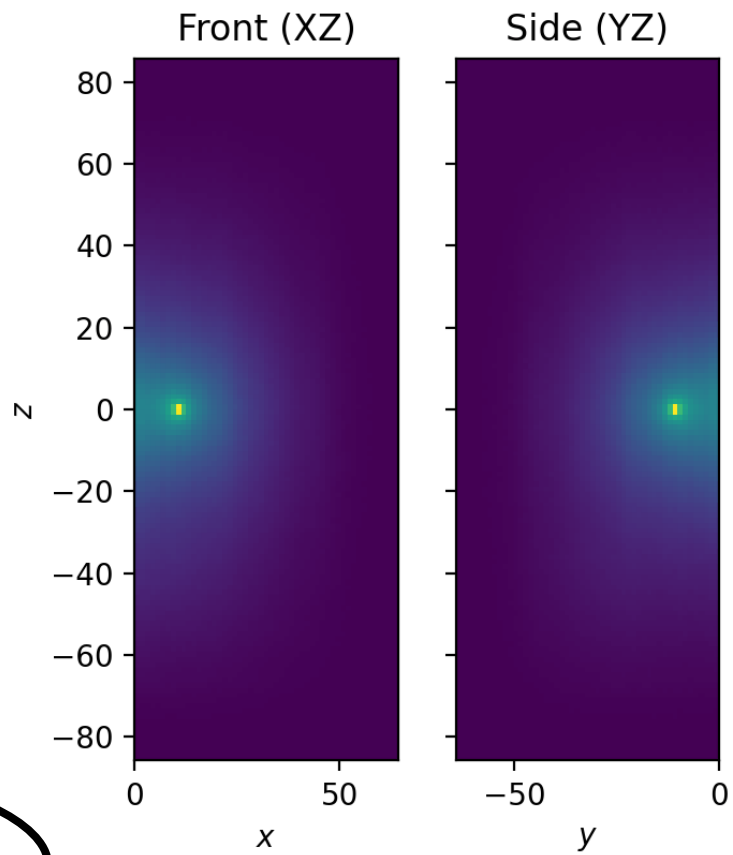


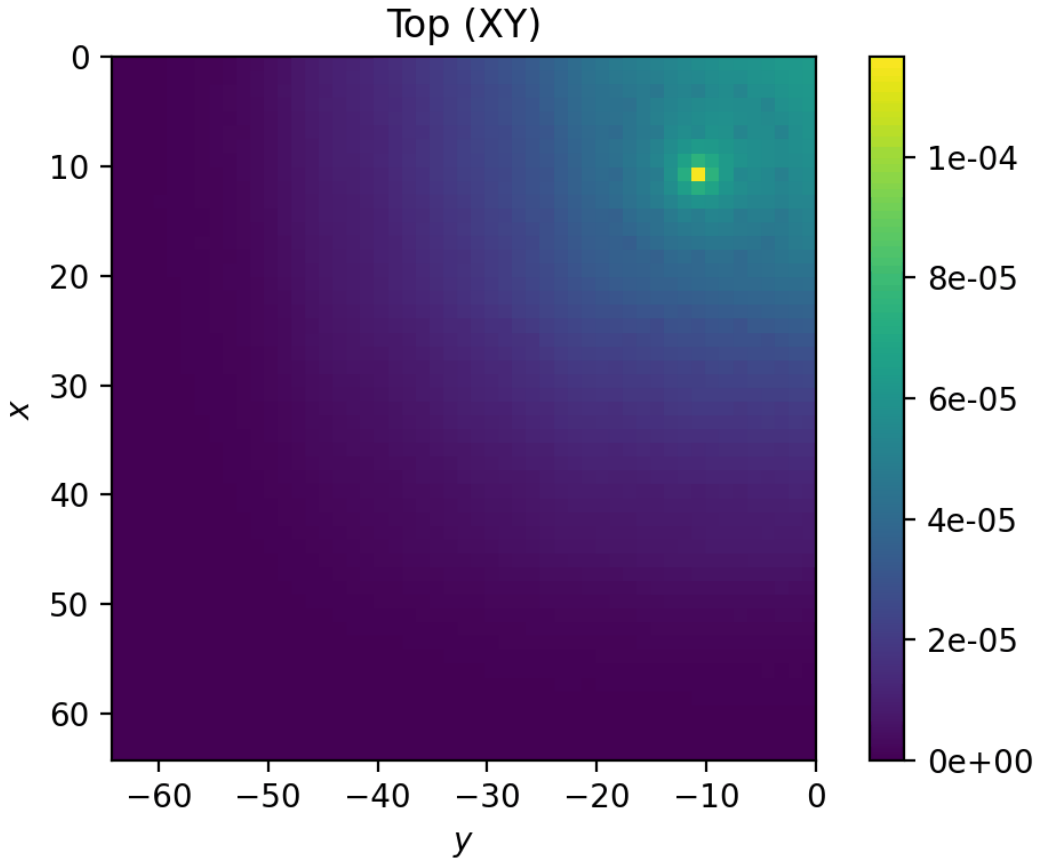


Continuous Energy Time Dependent Reactor Accident









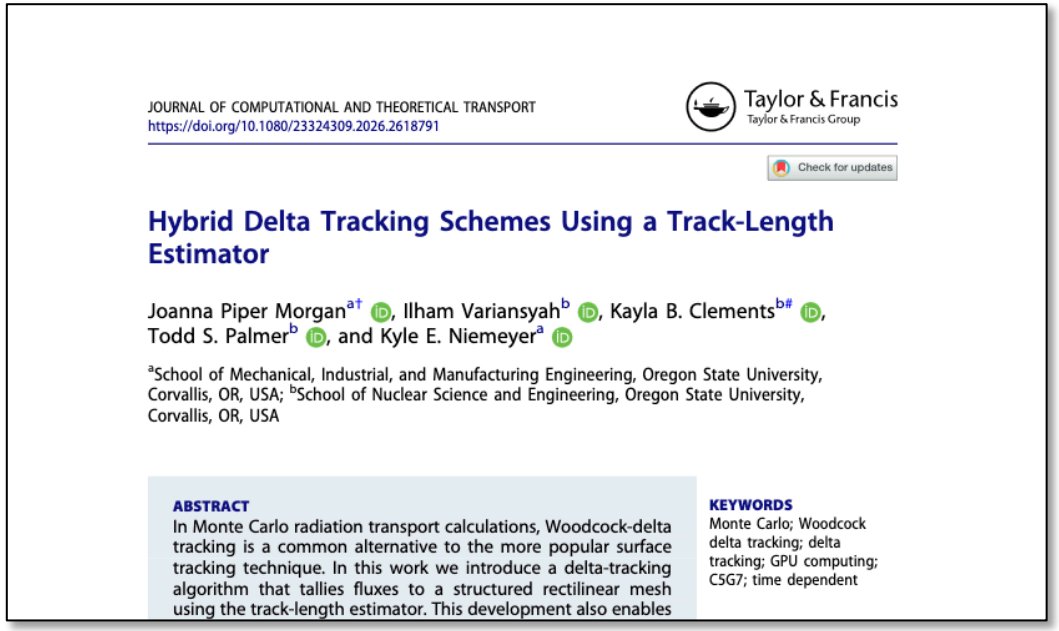
	Dane	Lassen
Run time [s] (CPU/GPU)	12996 (-)	5292 (2.5)
CPU core equiv	-	69

Goals

- ❑Write one piece of code and execute on GPUs CPUs and whatever
- ❑Code base specifically designed for rapid methods development
- ❑Have good coding practices for open-source development

Novel Method: Hybrid Tracking Method

- Let's implement a new method!
- Change the control flow of the algorithm dramatically
  - Hybrid surface-delta tracking algorithm
  - New tally structures pushing the data structures



JOURNAL OF COMPUTATIONAL AND THEORETICAL TRANSPORT
https://doi.org/10.1080/23324309.2026.2618791

Taylor & Francis
Taylor & Francis Group

Check for updates

Hybrid Delta Tracking Schemes Using a Track-Length Estimator

Joanna Piper Morgan[a†], Ilham Variansyah[b], Kayla B. Clements[b#], Todd S. Palmer[b], and Kyle E. Niemeyer[a]

[a]School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR, USA; [b]School of Nuclear Science and Engineering, Oregon State University, Corvallis, OR, USA

ABSTRACT

In Monte Carlo radiation transport calculations, Woodcock-delta tracking is a common alternative to the more popular surface tracking technique. In this work we introduce a delta-tracking algorithm that tallies fluxes to a structured rectilinear mesh using the track-length estimator. This development also enables

KEYWORDS

Monte Carlo; Woodcock delta tracking; delta tracking; GPU computing; C5G7; time dependent

What we want

- ❑ Write one piece of code and execute on GPUs CPUs

- ❑ Code base specifically designed for rapid methods development
- ❑ Have good coding practices for open-source development

What we've done in MC/DC

- Portable
  - Non compiled: Anything that runs Python3
  - CPUs: x86, ARM, POWERPC9*
  - GPUs: Nvidia, AMD
- Performant
  - scales on hundreds of nodes
  - single node performance matches compiled Monte Carlo codes
- Rapid methods development test bed
  - 15 research papers and counting from the code base

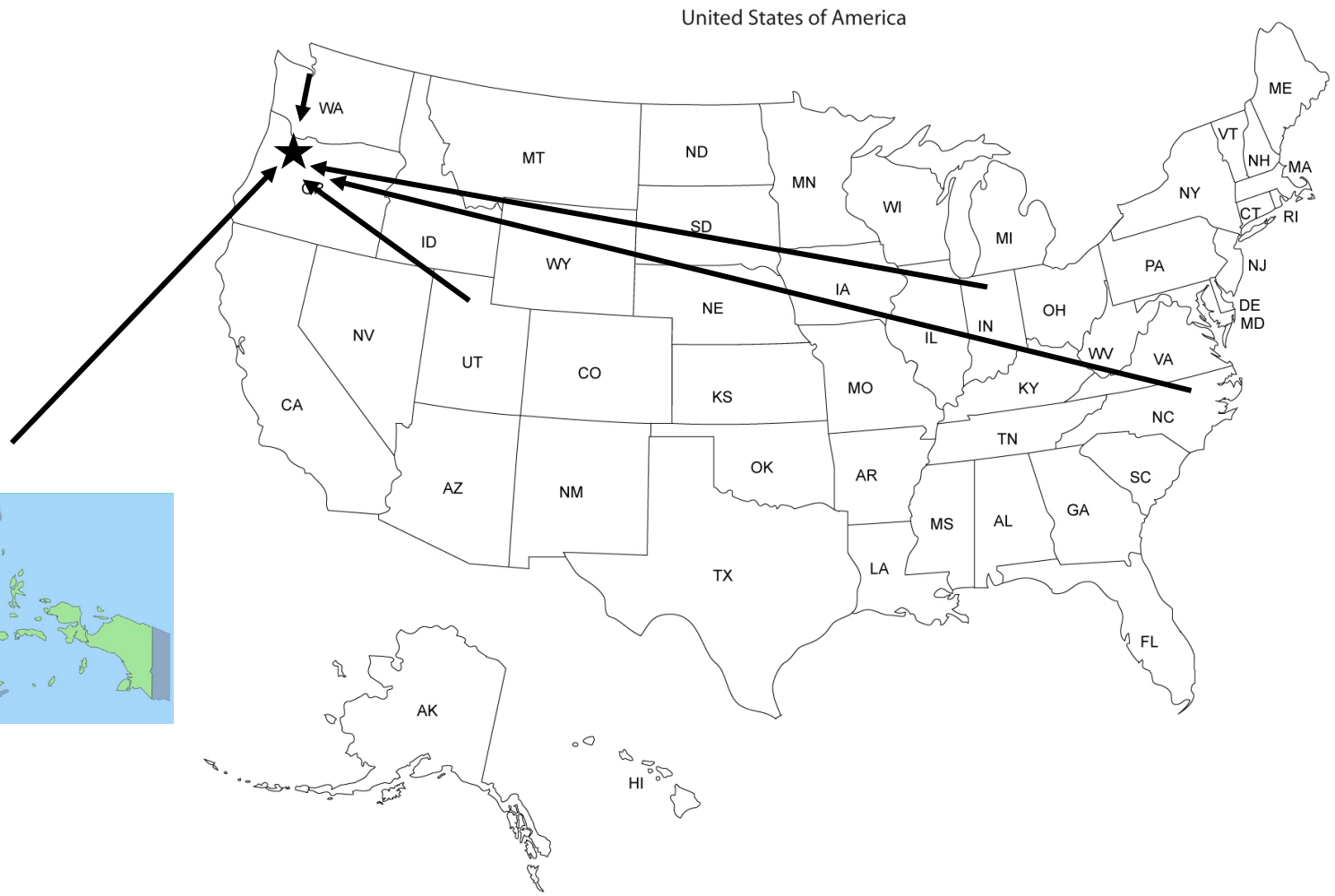
Technical Challenges

- Unsupported C-side functions (MPI, memalloc)
- Long JIT compile times
- Lacking ahead of time compilation
- Kernel profiling on CPUs or GPUs
- Package Instillation
- Cryptic compiler errors

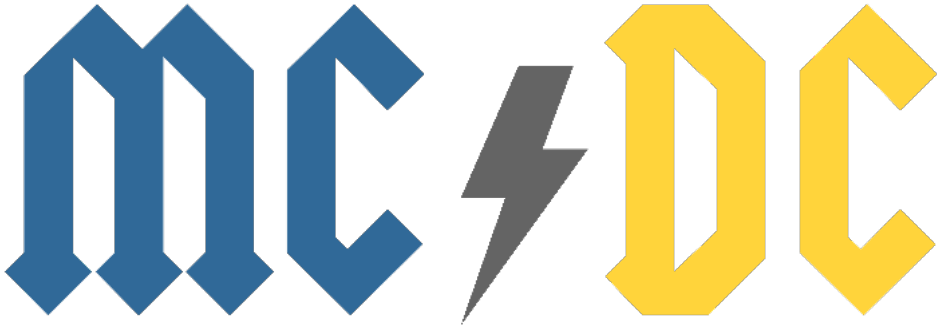
What can be Abstracted for Other Codes

- It is possible to us Python+Numba as a portability layer
- It does allow for rapid numerical methods development
- We suggest using two control flow sets (one for GPUs and one for CPUs)
- External scheduling is possible but still a research area
- Reach out for help!

Collaboration is Key: Hackathons



Thank you!



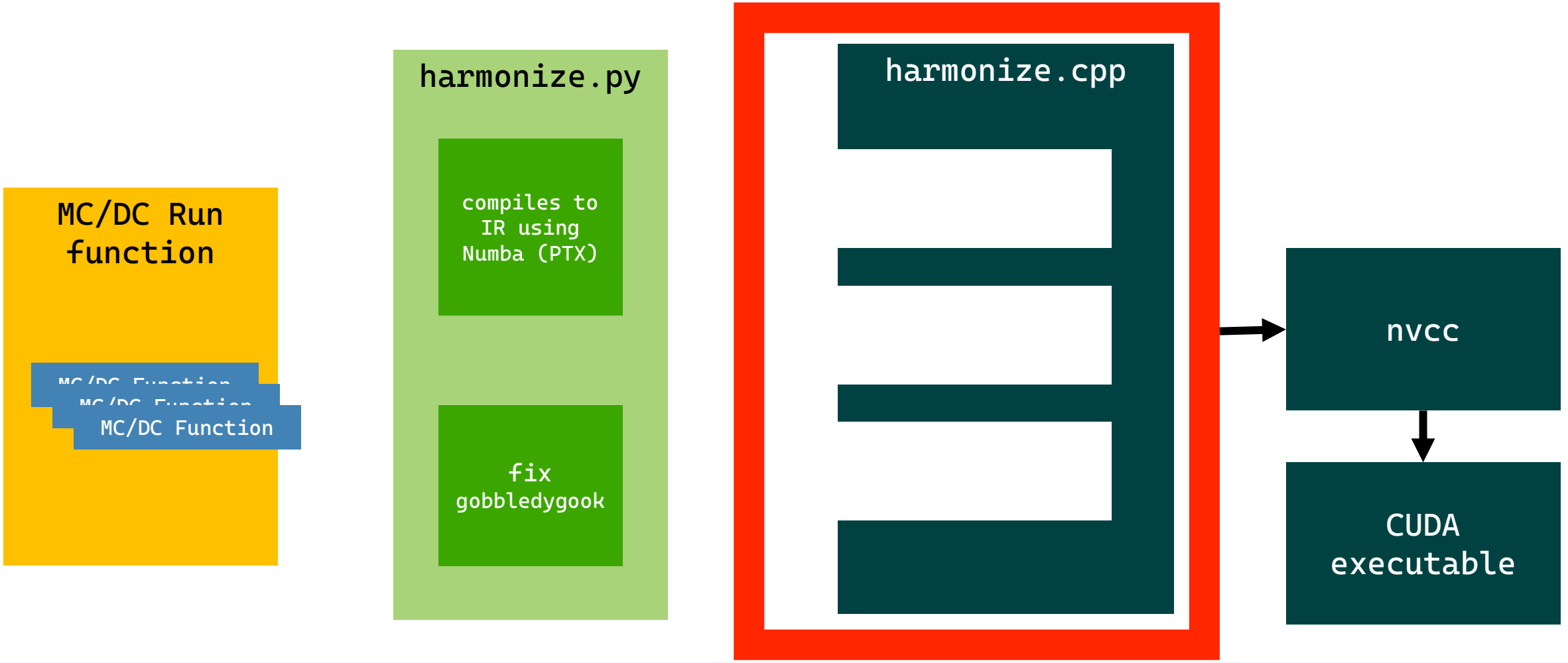
https://github.com/mcdc-project/mcdc

Harmonize

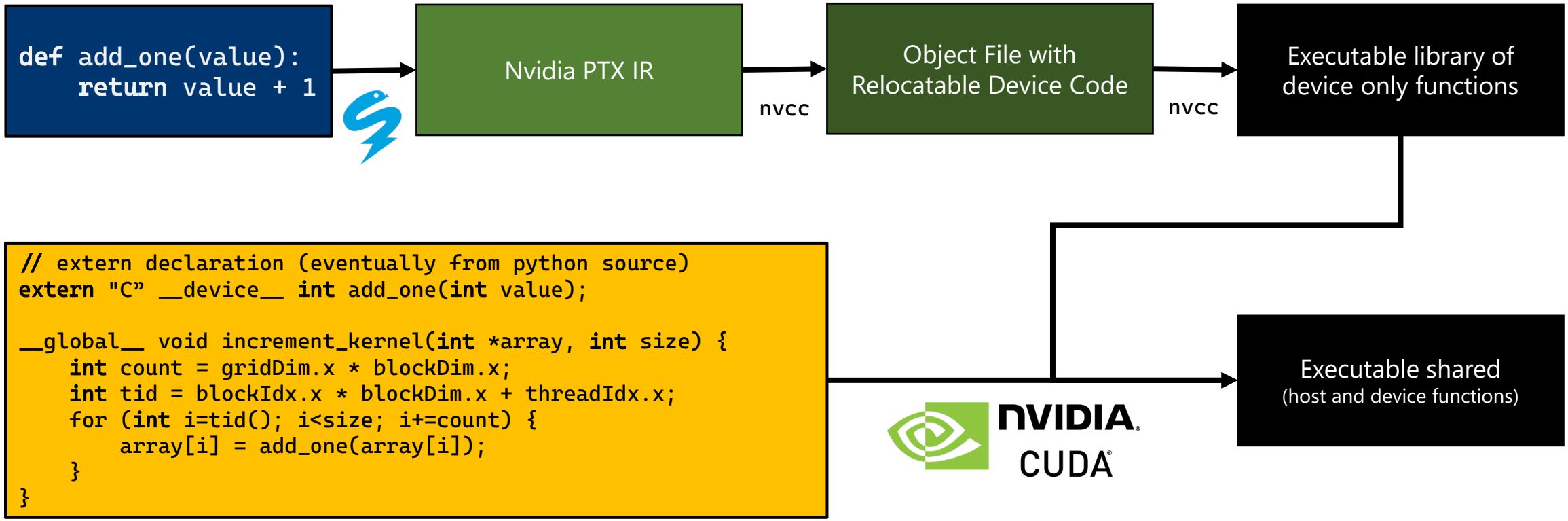
https://github.com/CEMeNT-PSAAP/harmonize

Backmatter

GPU device side async-scheduler



Nivida Support



AMD ROCm Support