Monte Carlo/Dynamic Code: Performant and Portable High-Performance Computing at Scale via Python and Numba

Joanna Piper Morgan* & Kyle E. Niemeyer

The Center for Exascale Monte Carlo Neutron Transport (CEMeNT)† *at Oregon State University*

**morgajoa@oregonstate.edu*

†*https://cement-psaap.github.io/*

23rd Conference on Scientific Computing in Python (SciPy 2024)

Tacoma, WA, USA

Wednesday, July 9th, 2024

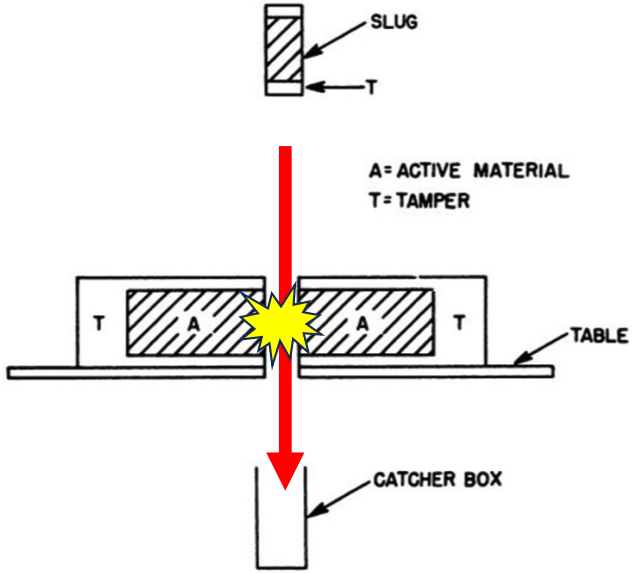
Oregon State University

This work was supported by the Center for Exascale Monte Carlo Neutron Transport (CEMeNT) a PSAAP-III project funded by the Department of Energy, grant number: DE NA003967.

CEMENT

Advancing the state of the art of **Monte Carlo neutronics calculations** particularly for solving **time-dependent transport problems** on **exascale computer architectures** in a **sustainable open-source community**

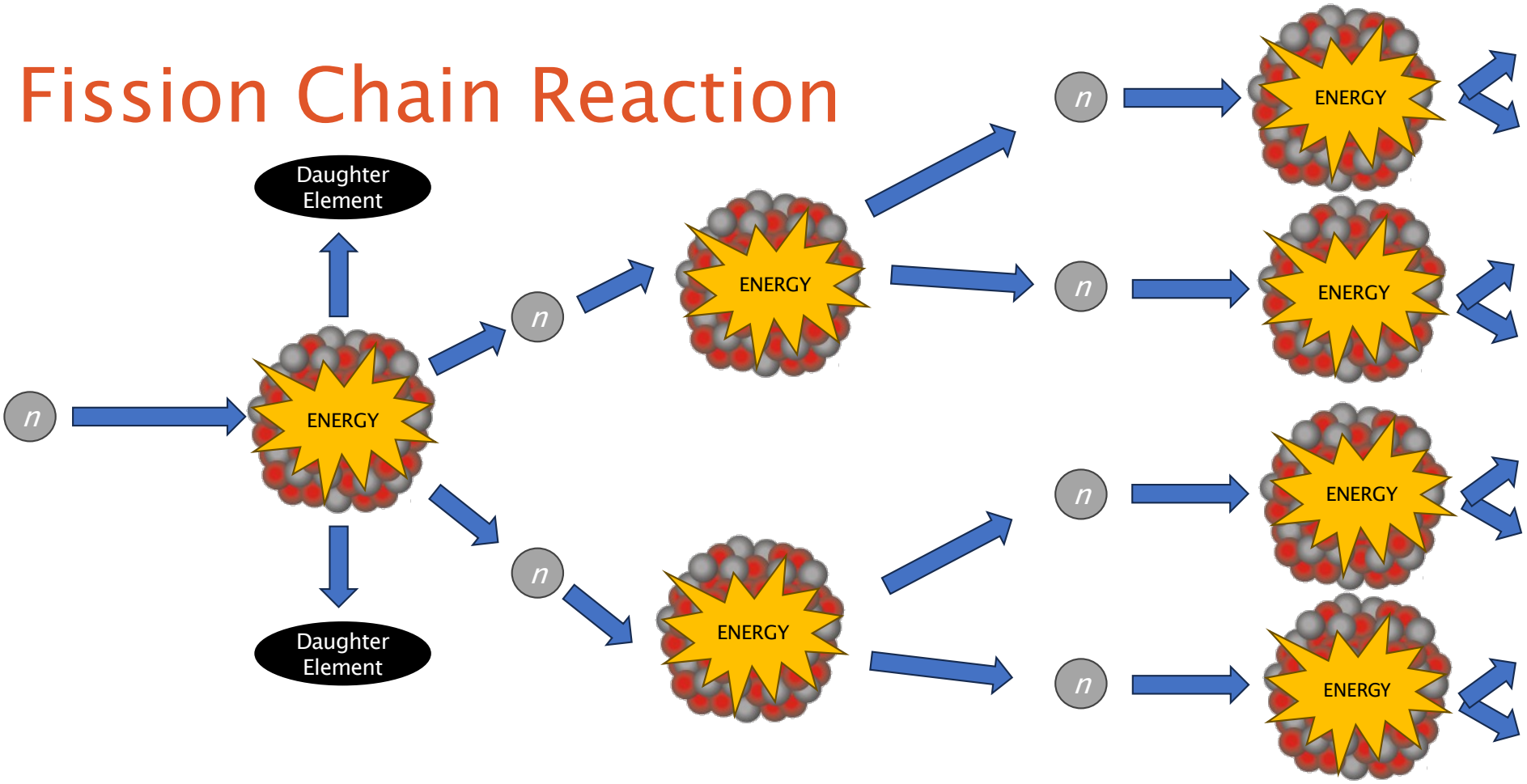
Dragon Burst Experiment



Kimpland, Robert, et al. "Critical assemblies: Dragon burst assembly and solution assemblies." *Nuclear Technology*207.sup1 (2021)

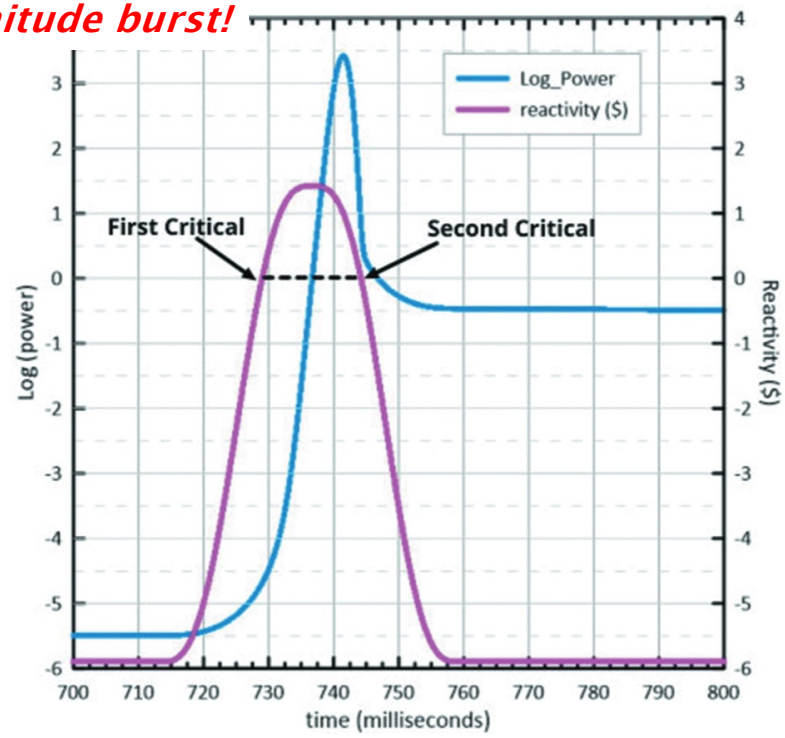
Slides courtesy of Ilham Variansyah

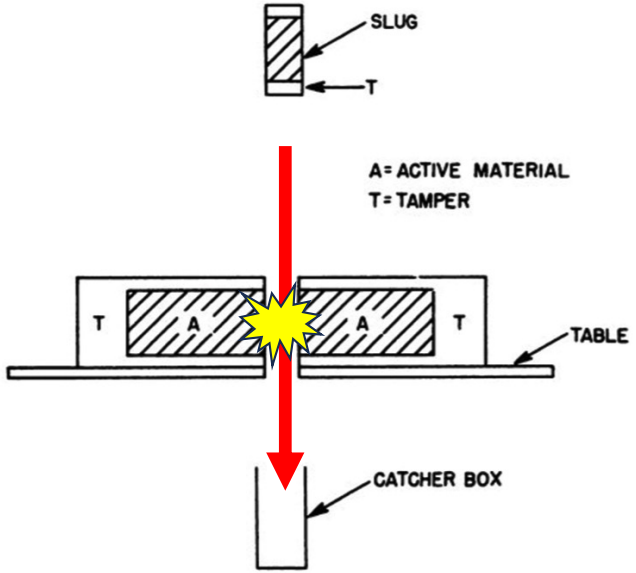
Fission Chain Reaction



Dragon Burst Experiment

Nine orders of magnitude burst!

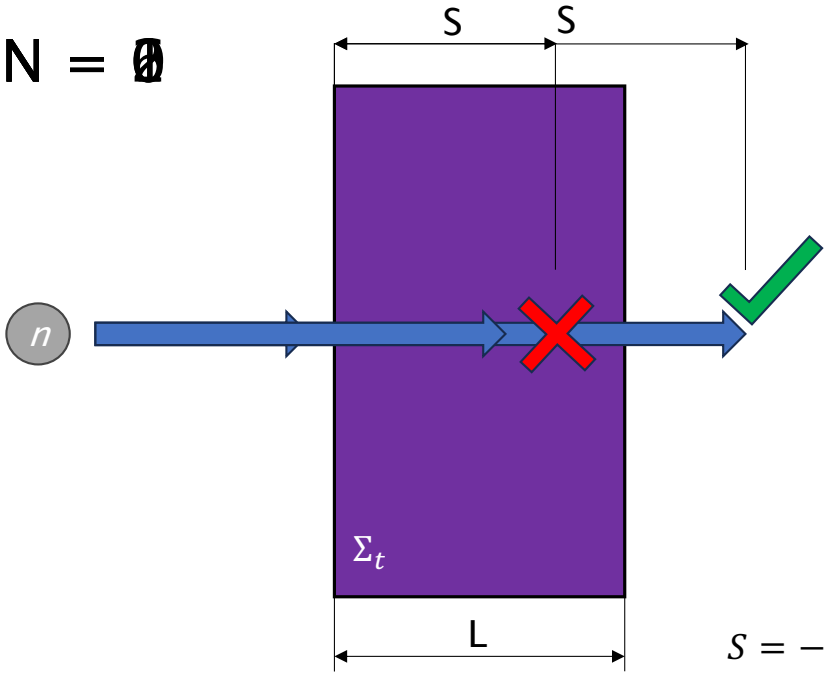




Kimpland, Robert, et al. "Critical assemblies: Dragon burst assembly and solution assemblies." *Nuclear Technology*207.sup1 (2021)

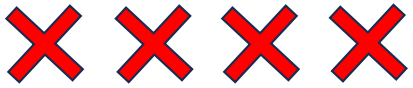
Slides courtesy of Ilham Variansyah

Monte Carlo Algorithm: Transmittance



Transmitted (S>L)

Absorbed (S<L)



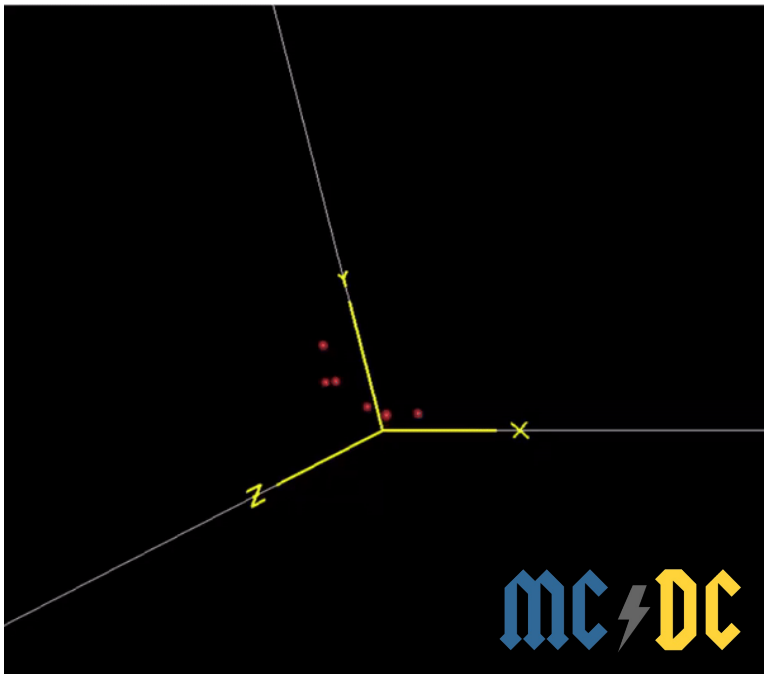
$$S = -\frac{\ln(\xi_N)}{\Sigma_t}$$

$$T = \frac{2}{6} = \frac{1}{3} \pm 0.221$$

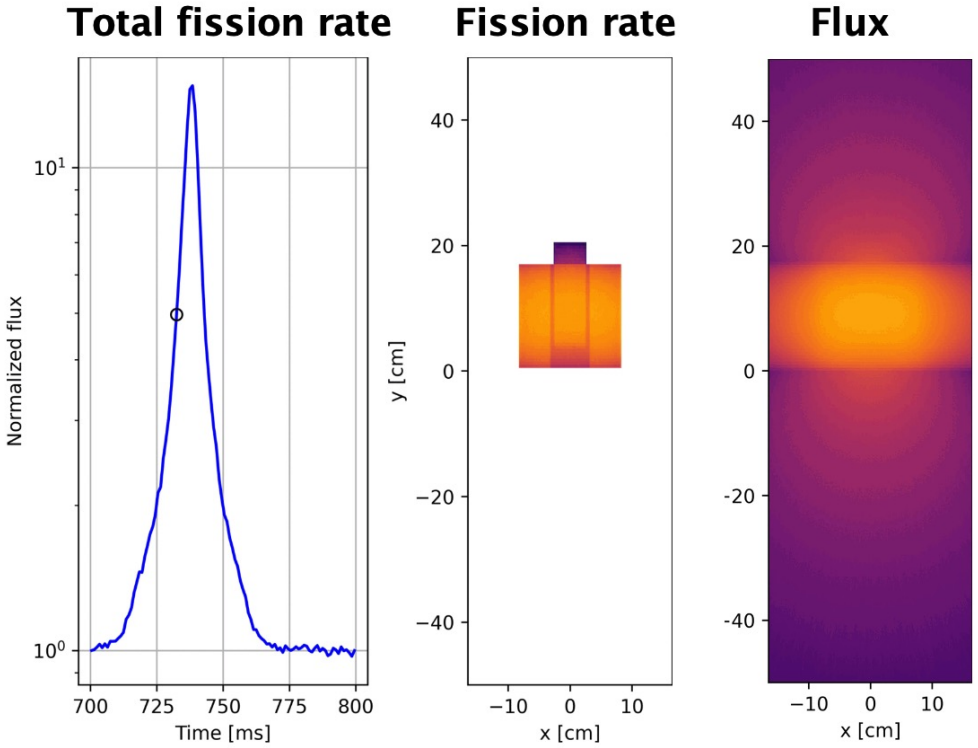
Monte Carlo / More Complexly

Imagine if:

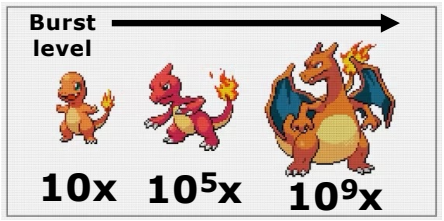
- neutrons could all be going at different speeds
- traveling in 3 spatial dimensions
- geometry changing thru time
- more complex tallies then transmittance
- neutrons produced from fission reactions or source regions
- multi-material systems
- error propagation
- temperature dependent systems



Dragon Burst Experiment



GIF removed for size constraint



Slides courtesy of Ilham Variansyah

Major take aways

No linear algebra, physics happens at the kernel

Each particle history is independent of every other particle

Solutions always have a statistical error

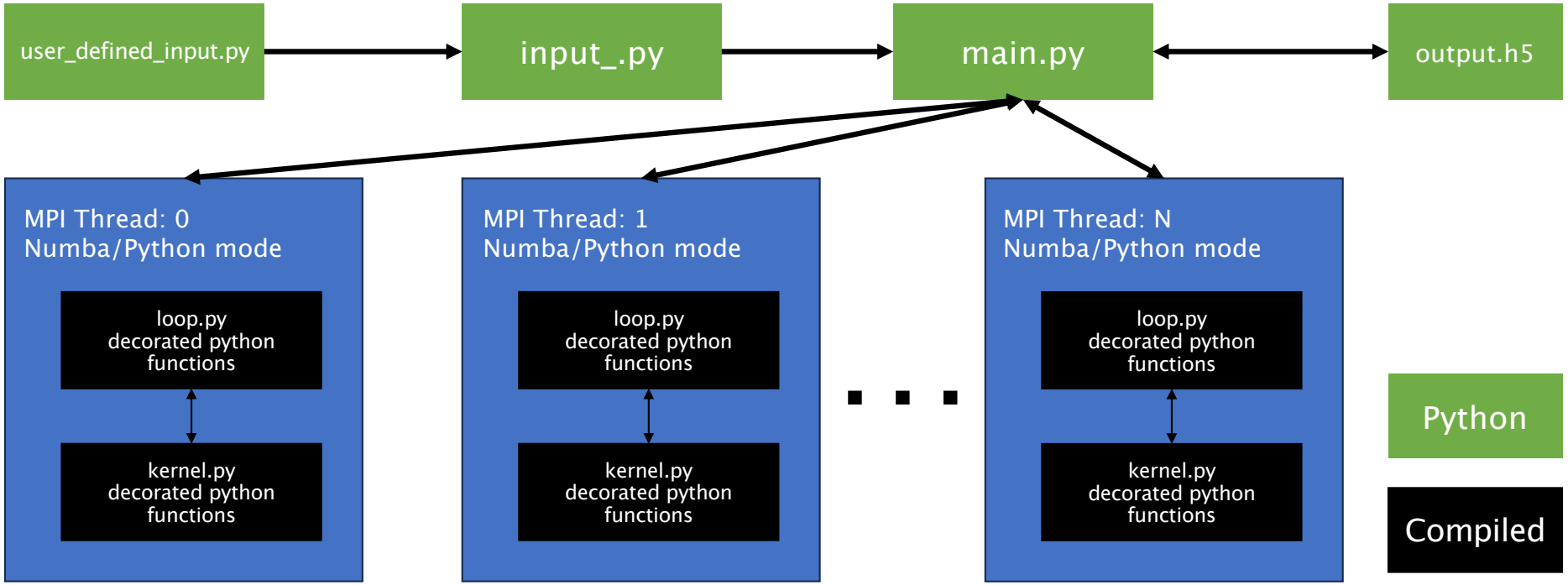
Converges very slowly

We'll need every FLOP we can get!

MC/DC at HPC

- Modern HPC's use lots of GPUs
- We need to produce compute kernels for both CPUs and GPUs
- Portability frameworks have entered the chat
  - Kokkos/Raja
  - DSLs + Python Glue
  - Julia*
  - Numba

MC/DC Layout


user_defined_input.py
input_.py
main.py
output.h5
MPI Thread: 0
Numba/Python mode
loop.py
decorated python
functions
kernel.py
decorated python
functions
MPI Thread: 1
Numba/Python mode
loop.py
decorated python
functions
kernel.py
decorated python
functions
MPI Thread: N
Numba/Python mode
loop.py
decorated python
functions
kernel.py
decorated python
functions
Python
Compiled

Monte Carlo/Dynamic Code (MC/DC)

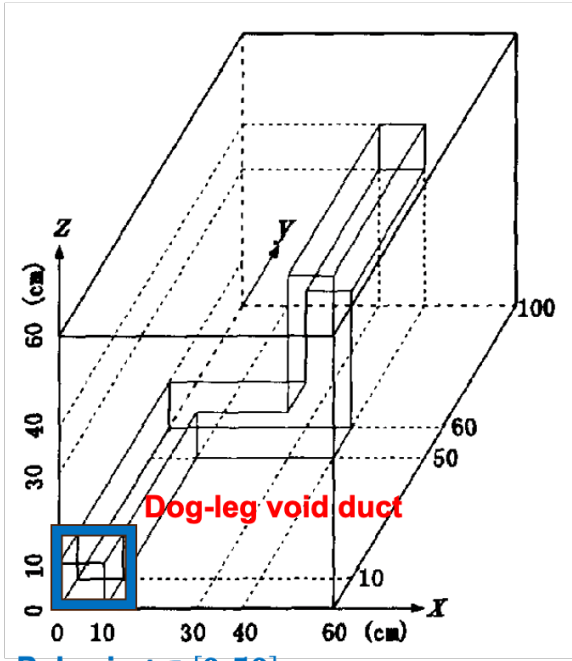
- Add @jit decorator to all functions
- Overhead is expected, but negligible for large problem
- Extensively use NumPy structured array for particle, cell, material, etc.
- Numpy structured scalar is used as global variable container

```
@njit
def move_particle(P, distance):
    P['x'] += P['ux']*distance
    P['y'] += P['uy']*distance
    P['z'] += P['uz']*distance
    P['t'] += distance/P['v']
```

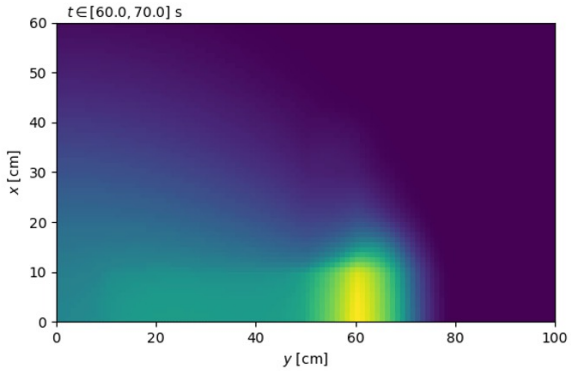
```
particle = np.dtype([
    ('x', float64),  ('y', float64),  ('z', float64),
    ('ux', float64), ('uy', float64), ('uz', float64)
```

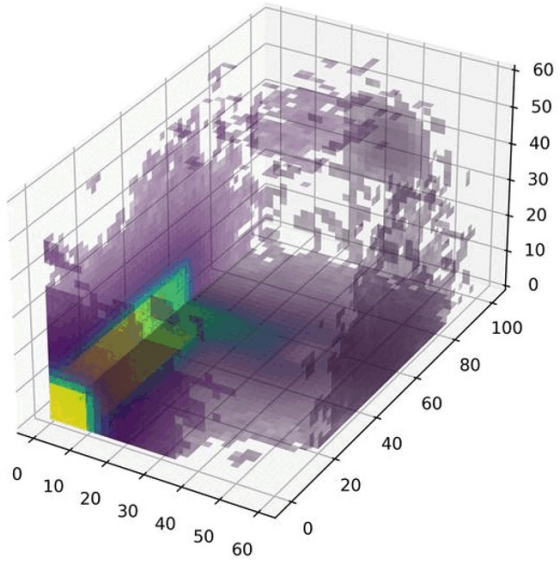
```
@njit
def distance_to_collision(P, mcdc):
    # Get total cross-section
    material = mcdc['materials'][P['material_ID']]
    SigmaT   = material['total'][P['g']]
```

Time-dependent Kobayashi dog-leg benchmark



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





GIF removed for size constraint

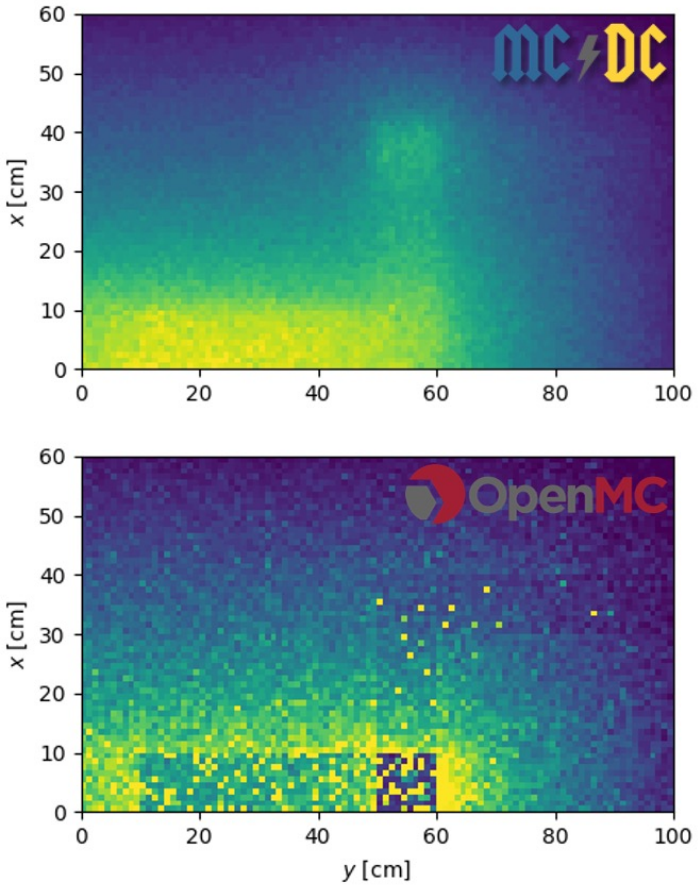
K. Kobayashi, "3-D Radiation Transport Benchmarks for Simple Geometries with Void Regions," OECD/NEA report

OpenMC vs MC/DC

	Runtime* [min]	FOM**
MC/DC	**22.27** (13.3x)	**2.24** (770x)
MC/DC (new)	**6.81** (4.1x)	**7.32** (2520x)
OpenMC	**1.67**	**0.0029**

*Run with 10 batches and 10^7 particles/batch on 36 cores
**Based on error 2-norm, with MC/DC 10^9 particles/batch as reference

GIF removed for size constraint



GPU Implementation of MC/DC

- Python abstractions to abstract hardware arch
- Turbocharging performance via Harmonize
  - A-sync GPU scheduler
  - Effectively on the fly event-based

Tomorrow, 16:30–17:00, Ballroom:
Dante's Externo: Injecting Python Functions into a Template-Driven CUDA C++ Framework, Braxton Cuneo

Harmonize Repo: github.com/CEMeNT-PSAAP/harmonize

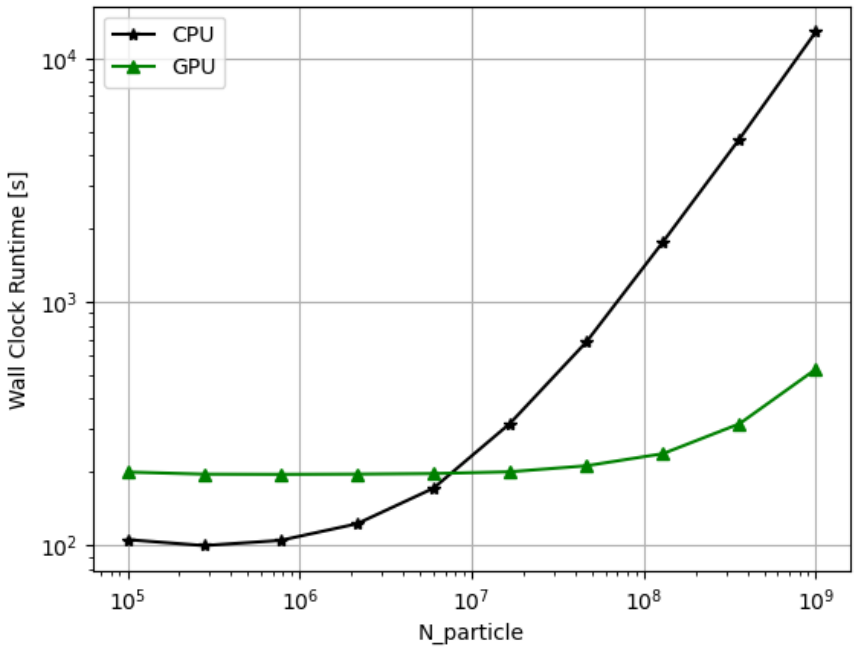
```python
@for_cpu()
def local_particle():
    return np.zeros(1, dtype=type_.particle)[0]

@for_gpu()
def local_particle():
    return cuda.local.array(1, dtype=type_.particle)[0]


@for_cpu()
def local_particle_record():
    return np.zeros(1, dtype=type_.particle_record)[0]

@for_gpu()
def local_particle_record():
    return cuda.local.array(1, dtype=type_.particle_record)[0]


@for_cpu()
def global_add(ary,idx,val):
    result    = ary[idx]
    ary[idx] += val
    return result

@for_gpu()
def global_add(ary,idx,val):
    return cuda.atomic.add(ary,idx,val)
```

MC/DC CPU v MC/DC GPU

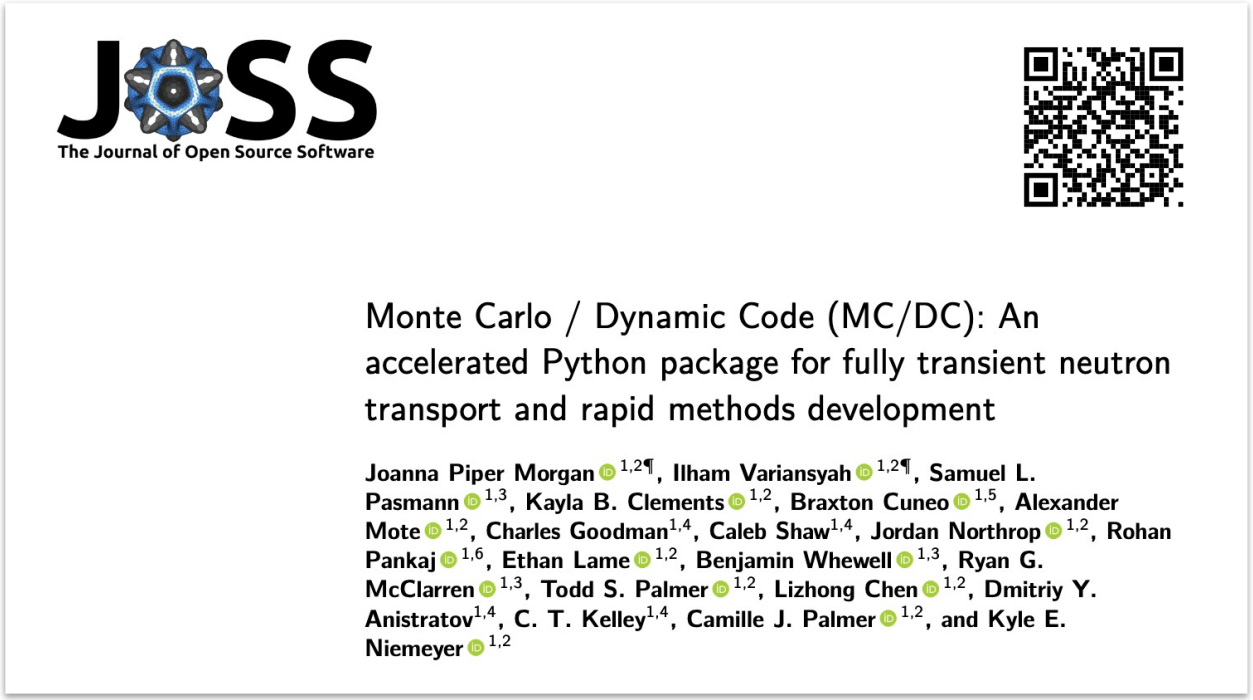
Kobayashi-monoenergetic
CPU Intel Xeon E5-2695,
36 cores/node

GPU 1 Nvidia Tesla V100

GPU speedup ~21-24 times



Publications that Force Good Development



DOI: 10.2205/joss.06415

Limitations of Numba

- Unsupported C-side functions (MPI, memalloc)
- Undocumented IR generation behavior
- Long compile times
- Lacking ahead of time compilation
- Lack of compiled kernel profiling on CPUs or GPUs
- Cryptic compiler errors

```
Traceback (most recent call last):
  File "/home/joamorga/workspace/MCDC/examples/fixed_source/slab_absorbium/input.py", line 48, in <module>
    mcdc.run()
  File "/home/joamorga/workspace/MCDC/mcdc/main.py", line 61, in run
    mcdc = prepare()
           ^^^^^^^^^
  File "/home/joamorga/workspace/MCDC/mcdc/main.py", line 191, in prepare
    build_gpu_progs()
  File "/home/joamorga/workspace/MCDC/mcdc/loop.py", line 1208, in build_gpu_progs
    process_sources = make_gpu_process_sources(False)
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/workspace/MCDC/mcdc/loop.py", line 1112, in make_gpu_process_sources
    spec = adapt.harm.RuntimeSpec(spec_name,adapt.state_spec,base_fns,async_fns)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/workspace/harmonize/harmonize.py", line 1100, in __init__
    self.generate_code()
  File "/home/joamorga/workspace/harmonize/harmonize.py", line 1564, in generate_code
    ptx_text  = extern_device_ptx(fn,self.type_map)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/workspace/harmonize/harmonize.py", line 636, in extern_device_ptx
    ptx_text, res_type = device_ptx(func)
                         ^^^^^^^^^^^^^^^^
  File "/home/joamorga/workspace/harmonize/harmonize.py", line 121, in device_ptx
    ptx, res_type = cuda.compile_ptx_for_current_device(func,fn_arg_ano(func),device=True,debug=DEBUG,opt=(not DEBUG))
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/hip/compiler.py", line 423, in compile_llvm_ir_for_current_device
    return compile_llvm_ir(
           ^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_lock.py", line 35, in _acquire_compile_lock
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/hip/compiler.py", line 354, in compile_llvm_ir
    cres: CompileResult = compile_hip(
                          ^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_lock.py", line 35, in _acquire_compile_lock
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/hip/compiler.py", line 262, in compile_hip
    cres = compiler.compile_extra(
           ^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 770, in compile_extra
    return pipeline.compile_extra(func)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 461, in compile_extra
    return self._compile_bytecode()
           ^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 529, in _compile_bytecode
    return self._compile_core()
           ^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 508, in _compile_core
    raise e
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 495, in _compile_core
    pm.run(self.state)
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_machinery.py", line 368, in run
    raise patched_exception
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_machinery.py", line 356, in run
    self._runPass(idx, pass_inst, state)
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_lock.py", line 35, in _acquire_compile_lock
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_machinery.py", line 311, in _runPass
    mutated |= check(pss.run_pass, internal_state)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_machinery.py", line 273, in check
    mangled = func(compiler_state)
              ^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typed_passes.py", line 110, in run_pass
    typemap, return_type, calltypes, errs = type_inference_stage(
                                            ^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typed_passes.py", line 91, in type_inference_stage
    errs = infer.propagate(raise_errors=raise_errors)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typeinfer.py", line 1086, in propagate
    raise errors[0]
numba.core.errors.TypingError: Failed in hip mode pipeline (step: nopython frontend)
No implementation of function Function(<function do_nothing_12 at 0x7fd11f3d02c0>) found for signature:
 >>> do_nothing_12(nestedarray(int64, (1,)), Literal[int](0), Literal[int](1))
There are 2 candidate implementations:
  - Of which 1 did not match due to:
  Overload in function 'jit_func': File: ../../../../../../workspace/MCDC/examples/fixed_source/slab_absorbium/<string>: Line 0.
    With argument(s): '(nestedarray(int64, (1,)), int64, int64)':
   Rejected as the implementation raised a specific error:
     TypingError: Failed in hip mode pipeline (step: nopython frontend)
   No implementation of function Function(<class 'numba.hip.typing_lowering.hipdevicelib.hipsource.add'>) found for signature:
    >>> add(nestedarray(int64, (1,)), int64, int64)
   There are 2 candidate implementations:
     - Of which 2 did not match due to:
     Type Restricted Function in function 'add': File: unknown: Line unknown.
       With argument(s): '(nestedarray(int64, (1,)), int64, int64)':
      No match for registered cases:
       * (int32,) -> UniTuple(int32 x 2)
       * (uint32,) -> UniTuple(uint32 x 2)
       * (uint64,) -> UniTuple(uint64 x 2)
       * (uint64,) -> UniTuple(uint64 x 2)
       * (float32,) -> UniTuple(float32 x 2)
       * (float64,) -> UniTuple(float64 x 2)
   During: resolving callee type: Function(<class 'numba.hip.typing_lowering.hipdevicelib.hipsource.add'>)
   During: typing of call at /home/joamorga/workspace/MCDC/mcdc/adapt.py (386)
   File "../../../mcdc/adapt.py", line 386:
   def global_add(ary,idx,val):
       return cuda.atomic.add(ary,idx,val)
       ^
  raised from /home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typeinfer.py:1086
  - Of which 1 did not match due to:
  Overload in function 'jit_func': File: ../../../../../../workspace/MCDC/examples/fixed_source/slab_absorbium/<string>: Line 0.
    With argument(s): '(nestedarray(int64, (1,)), Literal[int](0), Literal[int](1))':
   Rejected as the implementation raised a specific error:
     TypingError: Failed in hip mode pipeline (step: nopython frontend)
   No implementation of function Function(<class 'numba.hip.typing_lowering.hipdevicelib.hipsource.add'>) found for signature:
    >>> add(nestedarray(int64, (1,)), Literal[int](0), Literal[int](1))
   There are 2 candidate implementations:
      - Of which 2 did not match due to:
      Type Restricted Function in function 'add': File: unknown: Line unknown.
        With argument(s): '(nestedarray(int64, (1,)), int64, int64)':
       No match for registered cases:
        * (int32,) -> UniTuple(int32 x 2)
        * (uint32,) -> UniTuple(uint32 x 2)
        * (uint64,) -> UniTuple(uint64 x 2)
        * (uint64,) -> UniTuple(uint64 x 2)
        * (float32,) -> UniTuple(float32 x 2)
        * (float64,) -> UniTuple(float64 x 2)
   During: resolving callee type: Function(<class 'numba.hip.typing_lowering.hipdevicelib.hipsource.add'>)
   During: typing of call at /home/joamorga/workspace/MCDC/mcdc/adapt.py (386)
   File "../../../mcdc/adapt.py", line 386:
   def global_add(ary,idx,val):
       return cuda.atomic.add(ary,idx,val)
```

```
^
  raised from /home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typeinfer.py:1086
During: resolving callee type: Function(<function do_nothing_12 at 0x7fd11f3d02c0>)
During: typing of call at /home/joamorga/workspace/MCDC/mcdc/loop.py (966)
File "../../../mcdc/loop.py", line 966:
    def make_work(prog: nb.uintp) -> nb.boolean:
        <source elided>
        idx_work = adapt.global_add(mcdc["mpi_work_iter"],0,1)
```

Numba-Python v Others

- HIP, CUDA, Julia, and Kokkos may have superior performance on CPUs and GPUs across supported precisions for certain workflows (unoptimized gemm kernel)

Godoy, W. F., et. Al. (2023) Evaluating performance and portability of high-level programming models: Julia, Python/Numba, and Kokkos on exascale nodes; *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* DOI: 10.1109/IPDPSW59300.2023.00068

- Julia GPU support started after we started our work

- Taking full Python HPC as a DSL is doable but there might be better options

Current and Ongoing Work

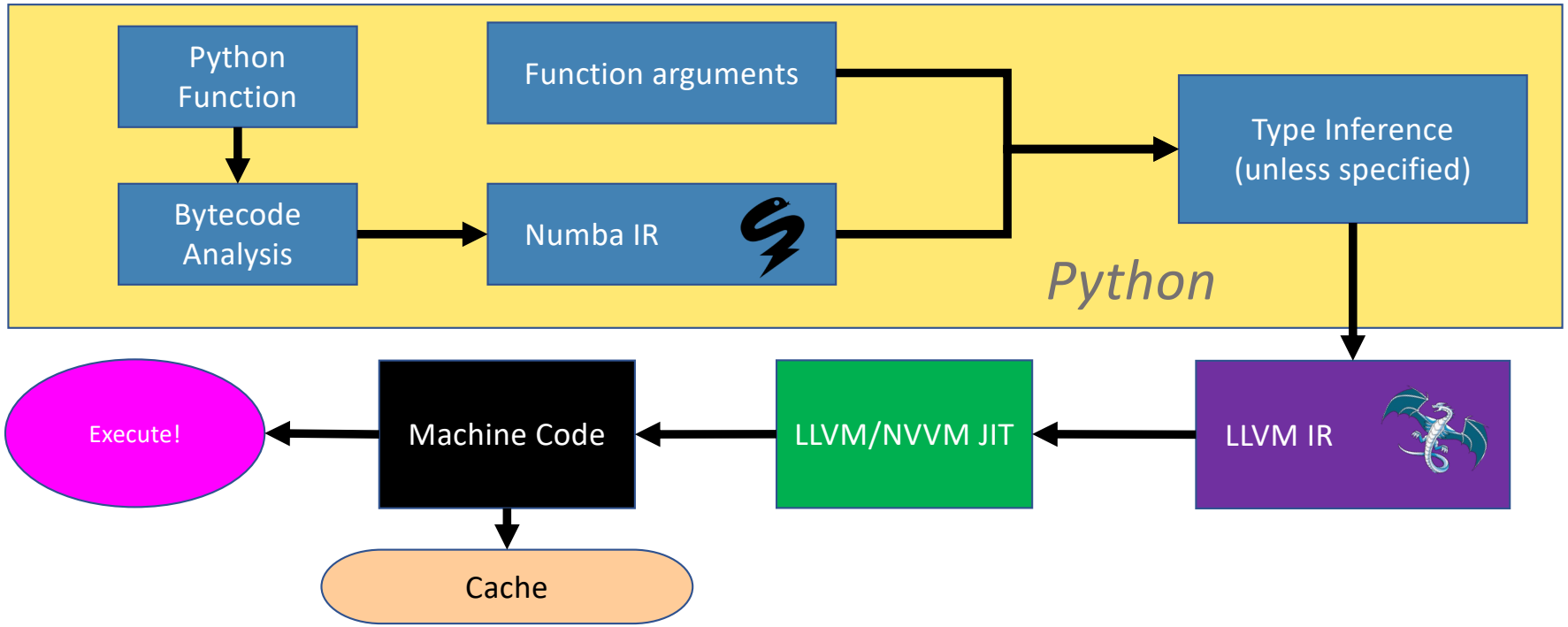
- Building out documentation, focus on users
- Full supporting AMD GPUs
- Profiling for both GPU and CPU systems (profila)
- Remove as much object modding as possible (MPI-Numba)

Conclusions

Advancing the state of the art of **Monte Carlo neutronics calculations** particularly for solving **time-dependent transport problems** on **exascale computer architectures** in a **sustainable open-source community**

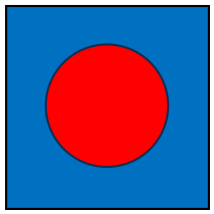
- Monte Carlo neutron transport is hard
- Exa-class computing is hard
- Performance portability using Numba for us seems to make things easier for developers enabling rapid numerical methods development

Numba Compilation

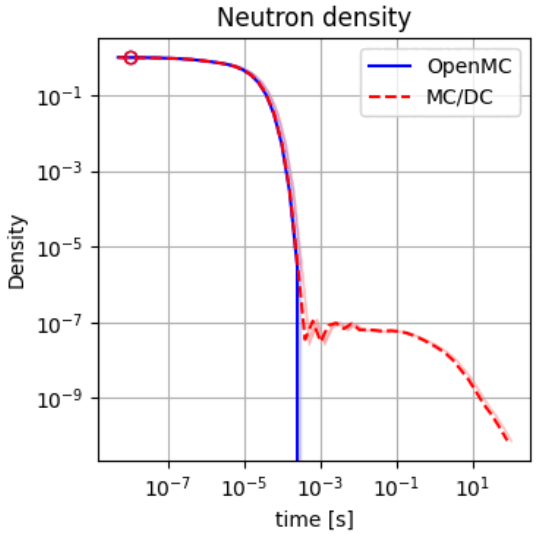


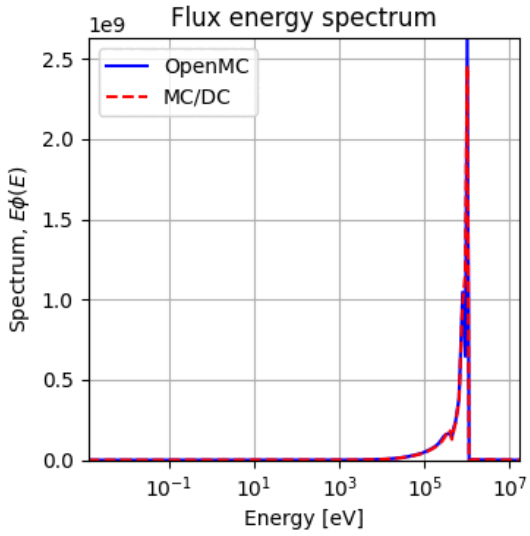
MC/DC Verification: Continuous energy physics

- **High-energy pulse in LEU pin cell**
  → neutron slowing-down wave
- Compared to OpenMC result
- Disagreement due to the missing
  - some inelastic reactions, and
  - high-fidelity scattering models [S(alpha, beta), …] in MC/DC.
  - OpenMC seems not tracking delayed neutrons yet.



UO_2 (2.4% enrichment)
H_2O + Boron





```
Traceback (most recent call last):
  File "/home/joamorga/workspace/MCDC/examples/fixed_source/slab_absorbium/input.py", line 48, in <module>
    mcdc.run()
  File "/home/joamorga/workspace/MCDC/mcdc/main.py", line 61, in run
    mcdc = prepare()
           ^^^^^^^^^
  File "/home/joamorga/workspace/MCDC/mcdc/main.py", line 191, in prepare
    build_gpu_progs()
  File "/home/joamorga/workspace/MCDC/mcdc/loop.py", line 1208, in build_gpu_progs
    process_sources = make_gpu_process_sources(False)
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/workspace/MCDC/mcdc/loop.py", line 1112, in make_gpu_process_sources
    spec = adapt.harm.RuntimeSpec(spec_name,adapt.state_spec,base_fns,async_fns)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/workspace/harmonize/harmonize.py", line 1100, in __init__
    self.generate_code()
  File "/home/joamorga/workspace/harmonize/harmonize.py", line 1564, in generate_code
    ptx_text  = extern_device_ptx(fn,self.type_map)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/workspace/harmonize/harmonize.py", line 636, in extern_device_ptx
    ptx_text, res_type = device_ptx(func)
                         ^^^^^^^^^^^^^^^^
  File "/home/joamorga/workspace/harmonize/harmonize.py", line 121, in device_ptx
    ptx, res_type = cuda.compile_ptx_for_current_device(func,fn_arg_ano(func),device=True,debug=DEBUG,opt=(not DEBUG))
                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/hip/compiler.py", line 423, in compile_llvm_ir_for_current_device
    return compile_llvm_ir(
           ^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_lock.py", line 35, in _acquire_compile_lock
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/hip/compiler.py", line 354, in compile_llvm_ir
    cres: CompileResult = compile_hip(
                          ^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_lock.py", line 35, in _acquire_compile_lock
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/hip/compiler.py", line 262, in compile_hip
    cres = compiler.compile_extra(
           ^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 770, in compile_extra
    return pipeline.compile_extra(func)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 461, in compile_extra
    return self._compile_bytecode()
           ^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 529, in _compile_bytecode
    return self._compile_core()
           ^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 508, in _compile_core
    raise e
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler.py", line 495, in _compile_core
    pm.run(self.state)
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_machinery.py", line 368, in run
    raise patched_exception
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_machinery.py", line 356, in run
    self._runPass(idx, pass_inst, state)
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_lock.py", line 35, in _acquire_compile_lock
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_machinery.py", line 311, in _runPass
    mutated |= check(pss.run_pass, internal_state)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/compiler_machinery.py", line 273, in check
    mangled = func(compiler_state)
              ^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typed_passes.py", line 110, in run_pass
    typemap, return_type, calltypes, errs = type_inference_stage(
                                            ^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typed_passes.py", line 91, in type_inference_stage
    errs = infer.propagate(raise_errors=raise_errors)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typeinfer.py", line 1086, in propagate
    raise errors[0]
numba.core.errors.TypingError: Failed in hip mode pipeline (step: nopython frontend)
No implementation of function Function(<function do_nothing_12 at 0x7fd11f3d02c0>) found for signature:
 >>> do_nothing_12(nestedarray(int64, (1,)), Literal[int](0), Literal[int](1))
There are 2 candidate implementations:
  - Of which 1 did not match due to:
  Overload in function 'jit_func': File: ../../../../../../workspace/MCDC/examples/fixed_source/slab_absorbium/<string>: Line 0.
    With argument(s): '(nestedarray(int64, (1,)), int64, int64)':
   Rejected as the implementation raised a specific error:
     TypingError: Failed in hip mode pipeline (step: nopython frontend)
   No implementation of function Function(<class 'numba.hip.typing_lowering.hipdevicelib.hipsource.add'>) found for signature:
    >>> add(nestedarray(int64, (1,)), int64, int64)
   There are 2 candidate implementations:
     - Of which 2 did not match due to:
     Type Restricted Function in function 'add': File: unknown: Line unknown.
       With argument(s): '(nestedarray(int64, (1,)), int64, int64)':
      No match for registered cases:
       * (int32,) -> UniTuple(int32 x 2)
       * (uint32,) -> UniTuple(uint32 x 2)
       * (uint64,) -> UniTuple(uint64 x 2)
       * (uint64,) -> UniTuple(uint64 x 2)
       * (float32,) -> UniTuple(float32 x 2)
       * (float64,) -> UniTuple(float64 x 2)
   During: resolving callee type: Function(<class 'numba.hip.typing_lowering.hipdevicelib.hipsource.add'>)
   During: typing of call at /home/joamorga/workspace/MCDC/mcdc/adapt.py (386)
   File "../../../mcdc/adapt.py", line 386:
   def global_add(ary,idx,val):
       return cuda.atomic.add(ary,idx,val)
       ^
  raised from /home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typeinfer.py:1086
  - Of which 1 did not match due to:
  Overload in function 'jit_func': File: ../../../../../../workspace/MCDC/examples/fixed_source/slab_absorbium/<string>: Line 0.
    With argument(s): '(nestedarray(int64, (1,)), Literal[int](0), Literal[int](1))':
   Rejected as the implementation raised a specific error:
     TypingError: Failed in hip mode pipeline (step: nopython frontend)
   No implementation of function Function(<class 'numba.hip.typing_lowering.hipdevicelib.hipsource.add'>) found for signature:
    >>> add(nestedarray(int64, (1,)), Literal[int](0), Literal[int](1))
   There are 2 candidate implementations:
       - Of which 2 did not match due to:
       Type Restricted Function in function 'add': File: unknown: Line unknown.
         With argument(s): '(nestedarray(int64, (1,)), int64, int64)':
        No match for registered cases:
         * (int32,) -> UniTuple(int32 x 2)
         * (uint32,) -> UniTuple(uint32 x 2)
         * (uint64,) -> UniTuple(uint64 x 2)
         * (uint64,) -> UniTuple(uint64 x 2)
         * (float32,) -> UniTuple(float32 x 2)
         * (float64,) -> UniTuple(float64 x 2)
   During: resolving callee type: Function(<class 'numba.hip.typing_lowering.hipdevicelib.hipsource.add'>)
   During: typing of call at /home/joamorga/workspace/MCDC/mcdc/adapt.py (386)
   File "../../../mcdc/adapt.py", line 386:
   def global_add(ary,idx,val):
       return cuda.atomic.add(ary,idx,val)
       ^
  raised from /home/joamorga/miniconda3/envs/hip/lib/python3.11/site-packages/numba/core/typeinfer.py:1086
During: resolving callee type: Function(<function do_nothing_12 at 0x7fd11f3d02c0>)
During: typing of call at /home/joamorga/workspace/MCDC/mcdc/loop.py (966)
File "../../../mcdc/loop.py", line 966:
        def make_work(prog: nb.uintp) -> nb.boolean:
            <source elided>
            idx_work = adapt.global_add(mcdc["mpi_work_iter"],0,1)
```

MC/DC current core capabilities

- **Multigroup physics**
  - ✓ Capture
  - ✓ Isotropic scattering
  - ✓ Fission (prompt and delayed)
- **Continuous energy physics** [new!]
  - ✓ NJOY generated point-wise data,
    - Room temperature
    - Assumed linear interpolation
  - ✓ Capture (MT=102-117)
  - ✓ Fission (prompt and delayed)
  - ✓ Scattering (non-capture & non-fission)
    - Isotropic elastic scattering in COM
    - Free gas, constant XS model for thermal scattering
  - ✓ Support almost all nuclides
- **Geometry**
  - ✓ Surface-tracking
  - ✓ Quadric CSG surface
  - ✓ Multi-level lattice
  - ✓ Time-dependent planar surfaces
- **Simulation modes**
  - ✓ Fixed-source (time-dependent)
  - ✓ k-Eigenvalue
- **Running modes:** Python, Numba
- **Parallel support**
  - ✓ MPI
  - ✓ Numba-CUDA (via ***Harmonize***)
  - ✓ Domain decomposition
  - ✓ Reproducibility (via hash-based RNG seeding)