HPC-BP Webinar Q&A

Rapid Development of a High-Performance Applications Using Python

Date: 2 September 2026
Presented by: Joanna Piper Morgan (Nuclear Criticality Safety Division, Lawrence Livermore National Lab)

(The slides are available via the link in the page's sidebar.)

Papers:

- J. P. Morgan, I. Variansyah, B. Cuneo, T. S. Palmer, and K. E. Niemeyer. (2025) Performant and Portable Monte Carlo Neutron Transport via Numba. *Computing in Science and Engineering* 27(1) pp. 57-65. doi 10.1109/MCSE.2025.3550863. arXiv:2409.04668.
- J. P. Morgan, I. Variansyah, S. Pasmann, K. B. Clements, B. Cuneo, A. Mote, C. Shaw, J. Northrop, R. Pankaj, E. Lame, B. Whewell, R. McClarren, T. Palmer, L. Chen, D. Anistratov, C. T. Kelley, C. Palmer, and K. E. Niemeyer. (2024) Monte Carlo / Dynamic Code (MC/DC): An accelerated Python package for fully transient neutron transport and rapid methods development. *Journal of Open Source Software*. 9(96), 6415. doi 10.21105/joss.06415
- Braxton Cuneo and Mike Bailey. 2024. Divergence Reduction in Monte Carlo Neutron Transport with On- GPU Asynchronous Scheduling. ACM Trans. Model. Comput. Simul. 34, 1, Article 2 (January 2024), 25 pages. https://doi.org/10.1145/3626957
- J. P. Morgan, B. Cuneo, I. Variansyah, K. E. Niemeyer. Enabling GPU portability into the Numba-JITed Monte Carlo particle transport code MC/DC. (2025). *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (ANS M&C 2025).* Denver, CO, USA. doi 10.13182/MC25-47142. arXiv:2501.05440
- B Cuneo, J. P. Morgan, I. Variansyah, K. E. Niemeyer. Comparing the Performance of MC/DC's on-GPU Event-based Processing Methods in Multigroup and Continuous-energy Problems. *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (ANS M&C 2025).* Denver, CO, USA. doi 10.13182/MC25-47174. arXiv:2506.00263

Project Repos:

- https://github.com/mcdc-project/mcdc
- https://mcdc.readthedocs.io/en/stable/index.html
- https://github.com/CEMeNT-PSAAP/harmonize
- https://carre-psaapiv.org/

Q: Have you considered/tried NumPy-like array libraries such as Pytorch, JAX, or cuPyNumeric to provide the compilation and GPU capabilities instead of Numba? These rely on vectorized operations for performance, does MC/DC rely heavily on explicit loops, or is it primarily vectorized?

A: Yes we did look into that however the limitations with posing Monte Carlo neutronics as vectorized operations were considered to be non-intunitive for standard kernel development. We decided to use a tool that placed as little restrictions on the methods development as possible so we could pose individual per particle operations. There is nothing that would stop you from doing this if you considered an event-based approach for particle scheduling.

Q: Can you name/link to some of the Numba extensions or Numba-like projects you alluded to in the presentation?

A: Yes!

- https://github.com/numba-mpi/numba-mpi/
- https://github.com/pythonspeed/profila
- https://github.com/numba/numba-scipy
- https://github.com/CEMeNT-PSAAP/harmonize
- https://github.com/ROCm/numba-hip
- https://github.com/IntelPython/numba-dpex

Q: How relevant and beneficial might be the offloading the library selection and compilation steps to a RAG-based LLM? Alternatively, how tedious is figuring out the optimization flags, troubleshooting the compilation steps manually for different DOE or national HPC systems? Did you containerize the workflow instead?

A: Deployments are still difficult on HPCs. For CPU things are fairly straightforward—given a compatible mpi4py distribution with your system's MPI. On

GPU things can be a bit more hacky with required package compilation and patch-fixes, but we also saw significant improvement over the duration of the project on distributions. Getting support from HPC staff was also pretty crucial for the viability of this project, shout out to Livermore Computing Services

Q: Can you cache the JITting away, e.g., avoiding that 10k GPU host ranks do the same JIT in parallel on first access? [Axel]

A: yes caching does exist in Numba and with Harmonize, however we ran into issues requiring us to clear the cache frequently.

Q: This is more of a comment. There are several Julia packages, such as Dagger.jl and JACC.jl, that allow the user to “write-once, run-anywhere”. Have you considered comparing the Python and Julia versions of your code in terms of development time and performance?

A: This project was started in 2020 before broad GPU support on Julia. Python solutions seemed to have more stable releases and support from GPU vendors at the time which was one of the reasons we went with a Python based approach. However as Julia has grown in popularity it has become an increasingly compelling platform. If we were to do the apples to apples comparison today I think the outcome would reflect that.

Q: Do you have any research ideas / directions on speeding up first-time JIT compilation?

A: Not currently but I would think the process could be parallelized or done with a “magic zero problem” touching all functions to be compiled then cached then JIT compiled. Also maybe some kind of a machine learning model to predictively guess which functions will need to be executed. This could also be coupled to some other understanding about execution times on a function by function basis to determine if there is even a benefit to compiling a given JIT function.

Q: Did cython exist when you started this work? Did you consider it rather than Numba? Do you have any thoughts about it today?

A:Yes, cython did exist we did but we thought Numba had a cleaner high-level language for our application and had a clearer path to GPUs with that high-level language. At first

we also tried to get away with not-statically typing (taking advantage of the Python-y-ness) values but found it impossible so the code is currently statically typed in the Numpy-ndarray. This effectively places all the limitations as a Cython implementation

Q: You mentioned there are open source projects bridging SciPy into C FFI calls or even GPU device function calls. What projects are those? [Axel]

A: Here!

- https://github.com/numba-mpi/numba-mpi/
- https://github.com/pythonspeed/profila
- https://github.com/numba/numba-scipy

Q: In event-based mode (I guess it is supported in MCDC), how do you handle cache misses, especially on GPU, since memory adjacent particles may not do the same event at once?

A: We have two strategies for this. The first uses the Harmonize asynchronous event scheduler so that like-operations across particles are binned and executed together. This is done on-device on-the-fly so that no user or numerical methods developer logic must be supplied other than the –hardware_target=gpu. The second is an event based algorithm so that we organize and sort particles into buffers to complete the same event in unison.
Some publications:

- Braxton Cuneo and Mike Bailey. 2024. Divergence Reduction in Monte Carlo Neutron Transport with On- GPU Asynchronous Scheduling. ACM Trans. Model. Comput. Simul. 34, 1, Article 2 (January 2024), 25 pages. https://doi.org/10.1145/3626957
- J. P. Morgan, B. Cuneo, I. Variansyah, K. E. Niemeyer. Enabling GPU portability into the Numba-JITed Monte Carlo particle transport code MC/DC. (2025). *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (ANS M&C 2025)*. Denver, CO, USA. doi 10.13182/MC25-47142. arXiv:2501.05440
- B Cuneo, J. P. Morgan, I. Variansyah, K. E. Niemeyer. Comparing the Performance of MC/DC's on-GPU Event-based Processing Methods in Multigroup and Continuous-energy Problems. *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and*

Engineering (ANS M&C 2025). Denver, CO, USA. doi 10.13182/MC25-47174. arXiv:2506.00263

Finding profilers for this application type to binge to explore cache misses was also quite difficult.

Q: Do you support unstructured meshes as a tally filter?

A: Not currently but there is no reason why this couldn't be added. May require some funky indexing to do. It's an open source project and I would implore you to give it a shot!