Python-metaprogramming in MC/DC: identifying and analyzing development paths

Jackson P. Morgan[a], Ilham Variansyah[b], Kyle Niemeyer[a]

Center for Exascale Monte-Carlo Neutron Transport (CEMeNT), [a]Oregon State University, [b]University of Notre Dame

Objective

Identify and perform preliminary analysis of Python-based methods to implement fine-grain parallelism on MC/DC

Introduction

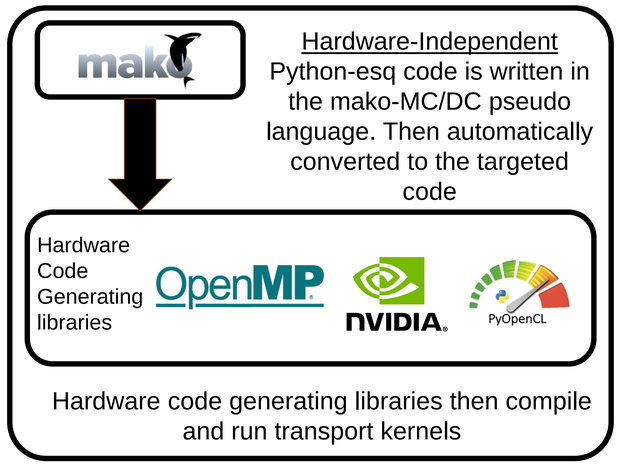
- HPC environment is growing more complex and individual machines are more heterogeneous
- Can we use Python metaprogramming techniques to get comparable runtimes to traditionally implemented codes?
- Coarse grain parallelism gained through MPI4Py

Metrics

- Has it been implemented before?
- How difficult to implement?
- What architectures can it target?
- Can it dynamically switch between targeted architectures?

Methods

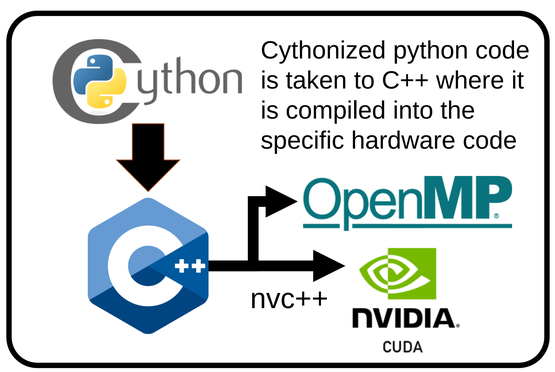
Templating Engine (PyFR [1])



Pros
- Successfully implemented at peta-scale (PyFR)
- Can target any architecture we have code generating libraries for

Cons
- High initial development costs
- Relies on multiple components which are subject to change
- Requires special development knowledge for collaborators

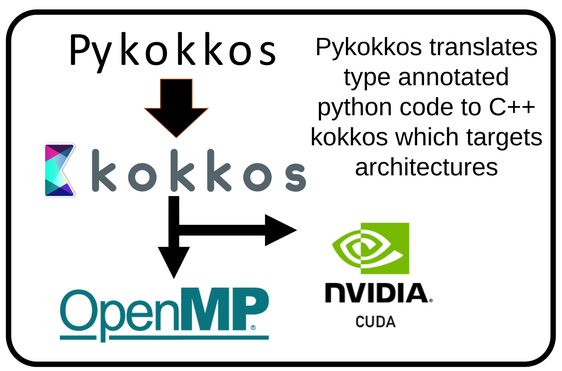
Cython [2]



Pros
- Easy to implement

Cons
- Still requires some C/C++ code for OpenCL

PyKokkos [3]



Pros
- Use DOE coding infrastructure
- Targets specific DOE machines

Cons
- Currently under development
- Does not target OpenCL

Conclusions

Templating engine method seems to be the most viable although testing is required to form performance-based conclusions

Future Work

- Implement each method with a transient test case (with pop control) and examine performance and difficulty of development

References

1. PyFR: An Open Source Framework for Solving Advection-Diffusion Type Problems on Streaming Architectures using the Flux Reconstruction Approach. F. D. Witherden, A. M. Farrington, P. E. Vincent. Computer Physics Communications, Volume 185, Pages 3028-3040, 2014.
2. R. Bradshaw, S. Behnel, D. S. Seljebotn, G. Ewing, et al., The Cython compiler, http://cython.org.
3. Al Awar, N., Zhu, S., Biros, G., & Gligoric, M. (2021). A Performance Portability Framework for Python. In International Conference on Supercomputing (pp. To appear).

Acknowledgements

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.