
# Workshop

Sterling Harper, Travis Labossiere-Hickman,
Luke Eure, Patrick White,
and Benoit Forget

April 6 2016

# OpenMC

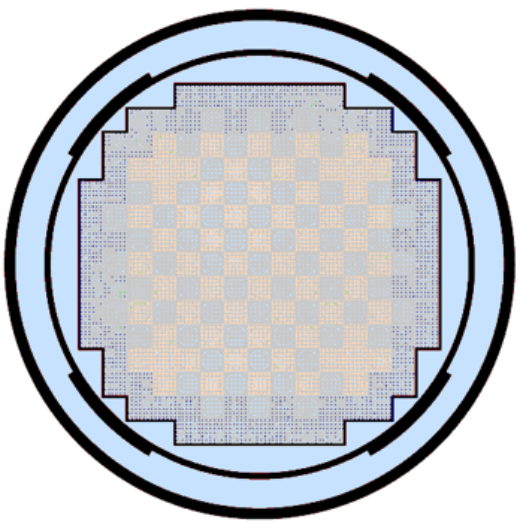
OpenMC is an open source Monte Carlo code that has been developed at MIT by the CRPG group since 2010.

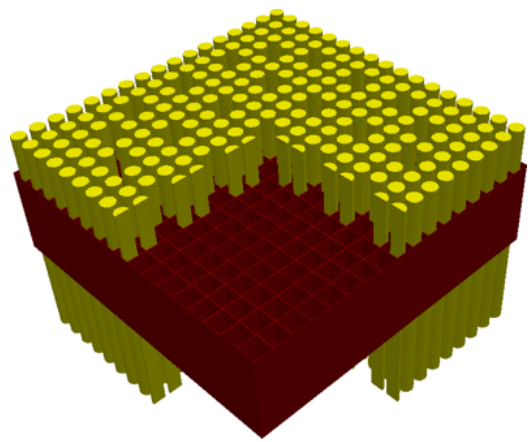


- Developed as part of a PhD thesis in order to resolve scalability issues on leadership class computing platforms.
- Features were added to truly test parallel algorithm to a point where results became realistic.
- Serves as an essential research element of the current CPRG research

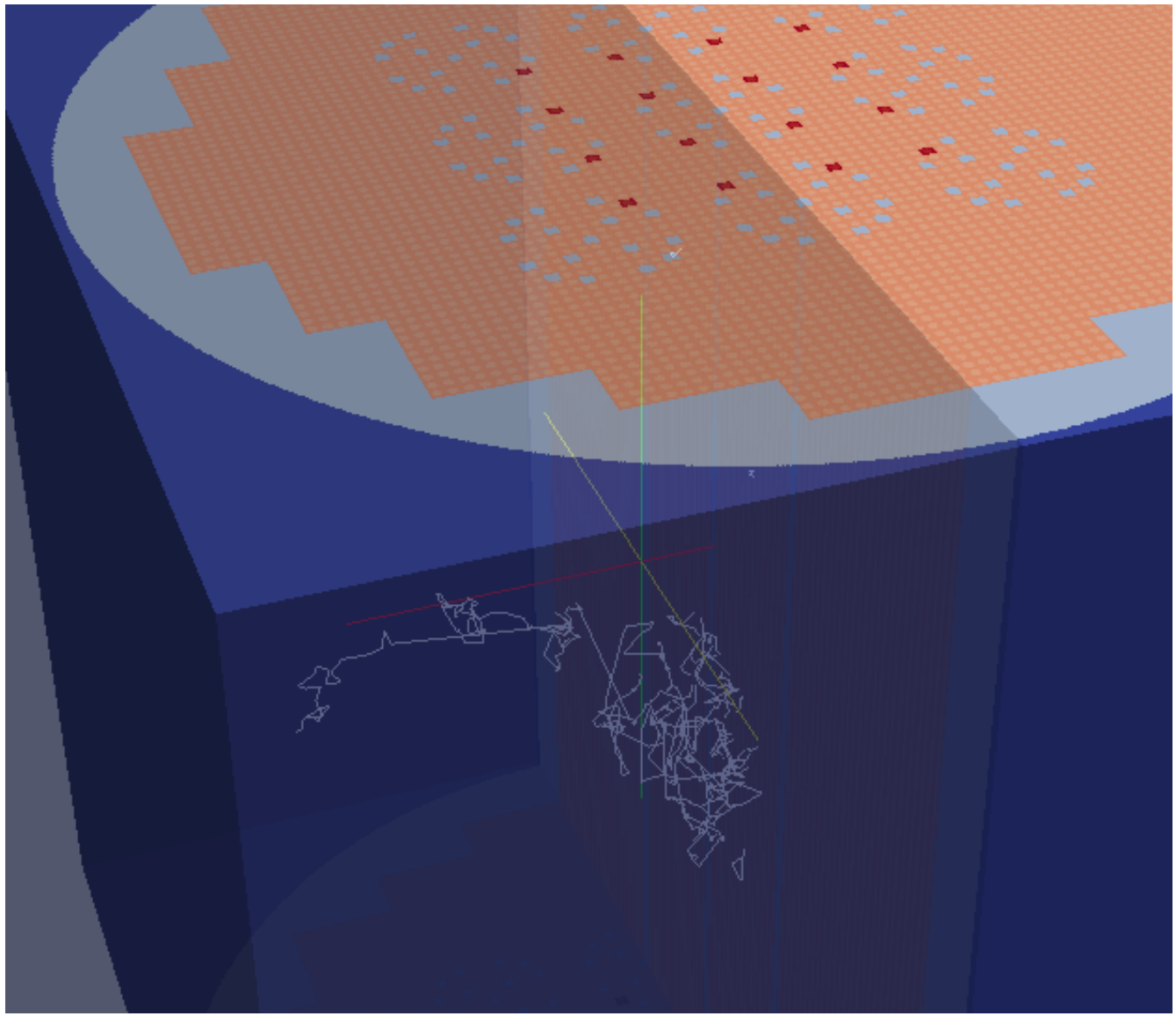
# Development Team

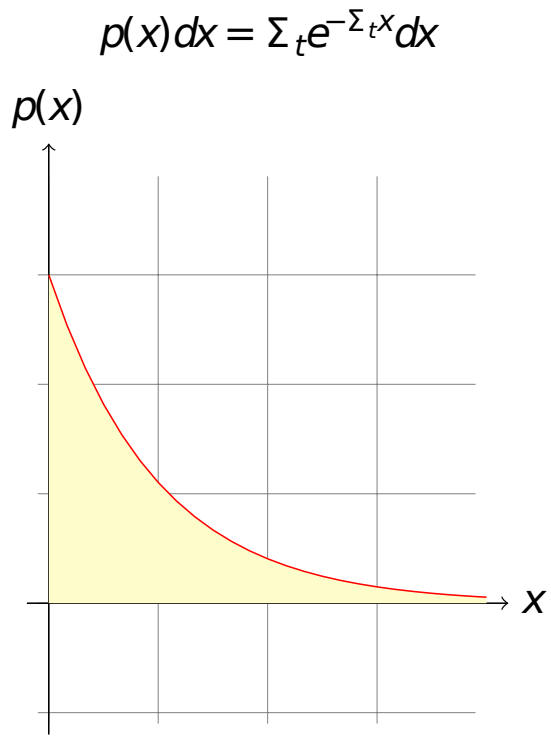
- Paul Romano (MIT grad, Lead developer, ANL)
- Nicholas Horelik (MIT grad, startup)
- Bryan Herman (MIT grad, KAPL)
- Adam Nelson (UM grad, Naval Reactors)
- Jon Walsh (MIT grad, LLNL)
- Will Boyd (MIT grad, BCG)
- Lulu Li (MIT grad, Google)
- Sterling Harper (MIT)
- Matt Ellis (MIT)
- Sam Shaner (MIT)
- Colin Josey (MIT)
- Travis Labossiere-Hickman (MIT)
- Jingang Liang (Tsinghua grad, MIT postdoc)
- Xingjie Peng (Tsinghua grad, MIT postdoc)





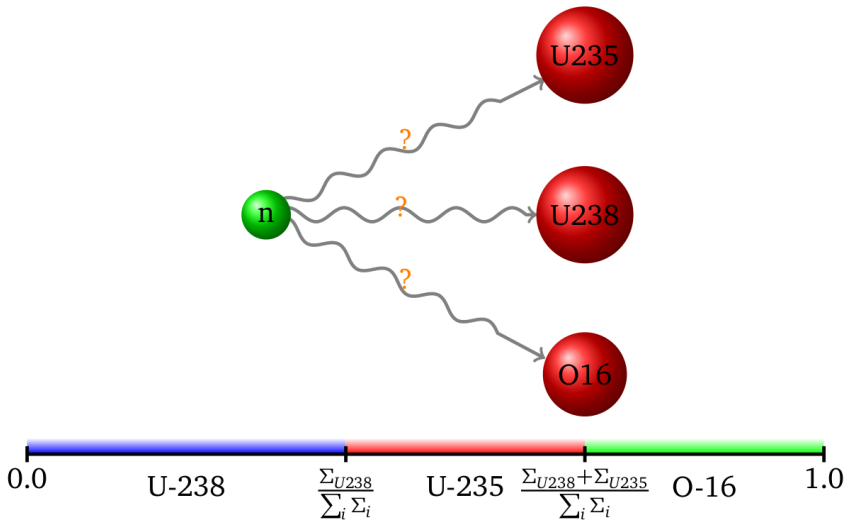
# Monte Carlo simulates the movement and reactions of individual neutrons



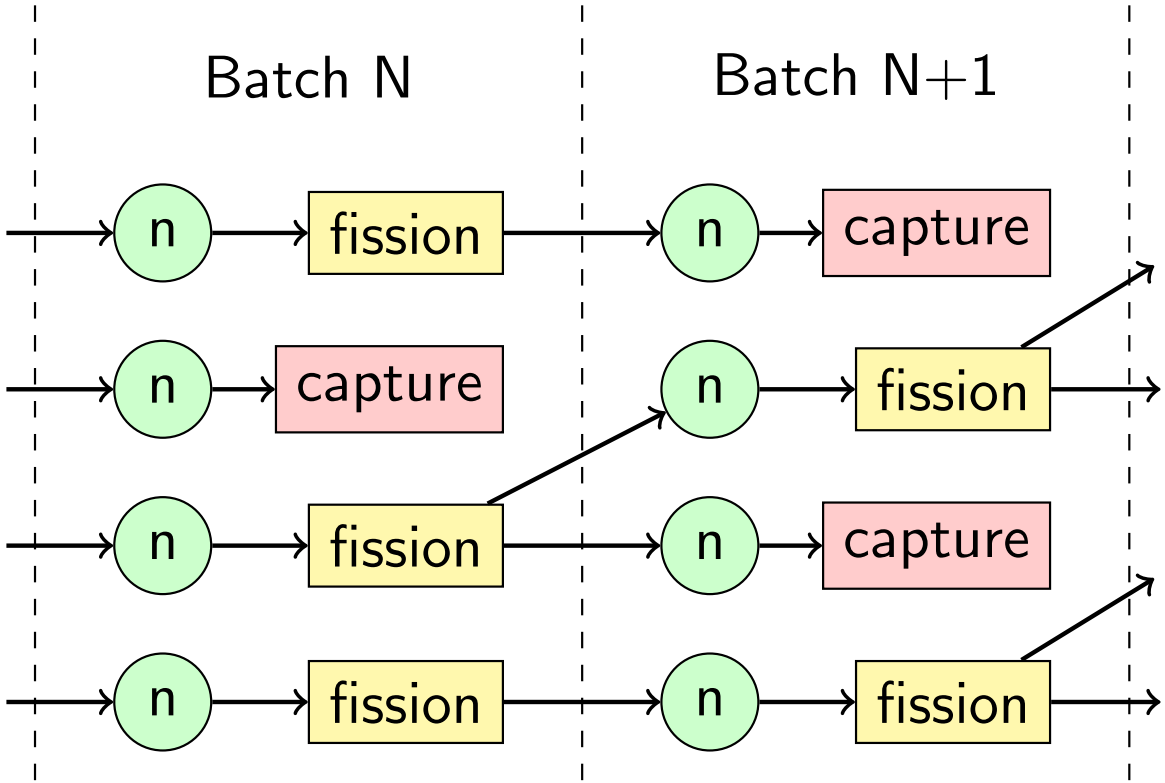



Monte Carlo randomly samples continuous distributions

For example, we sample an exponential to determine how far a neutron travels before the next collision




We compare cross sections to determine which nuclide a neutron collides with and which reaction happens

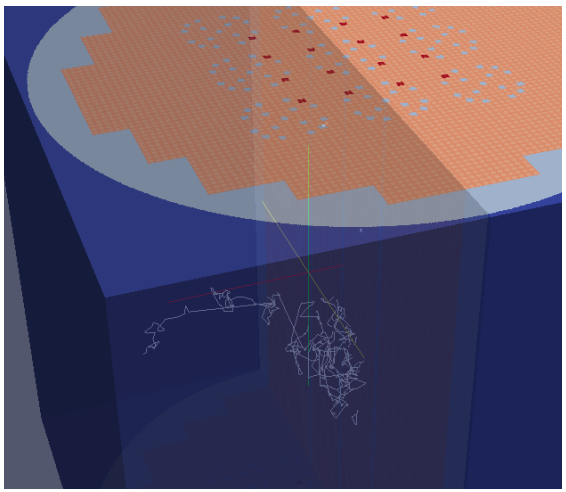



MC simulations are divided into batches for statistical reasons

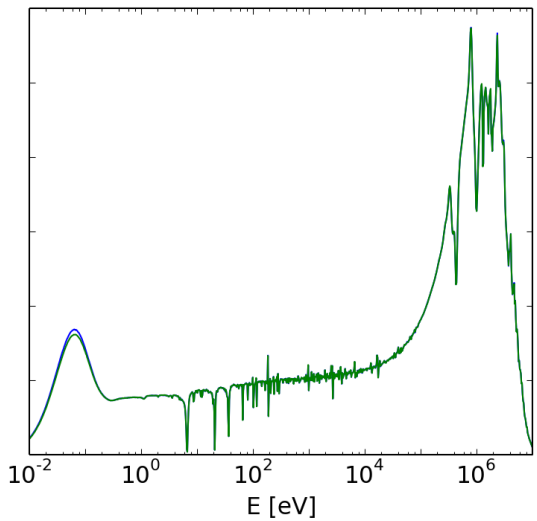
Neutrons are born from fission sites in the last batch

# Monte Carlo is precise because it uses

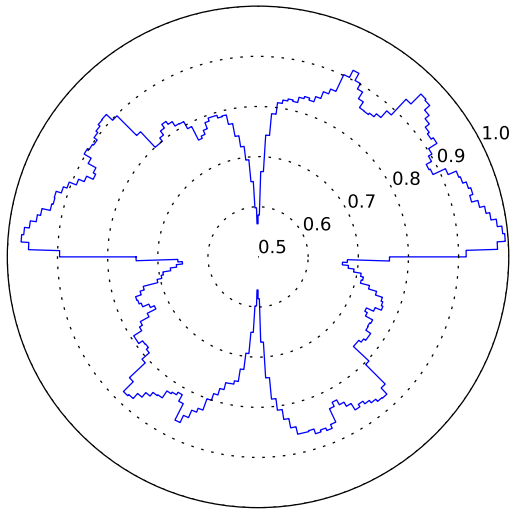
## 3D Space



## Continuous Energy

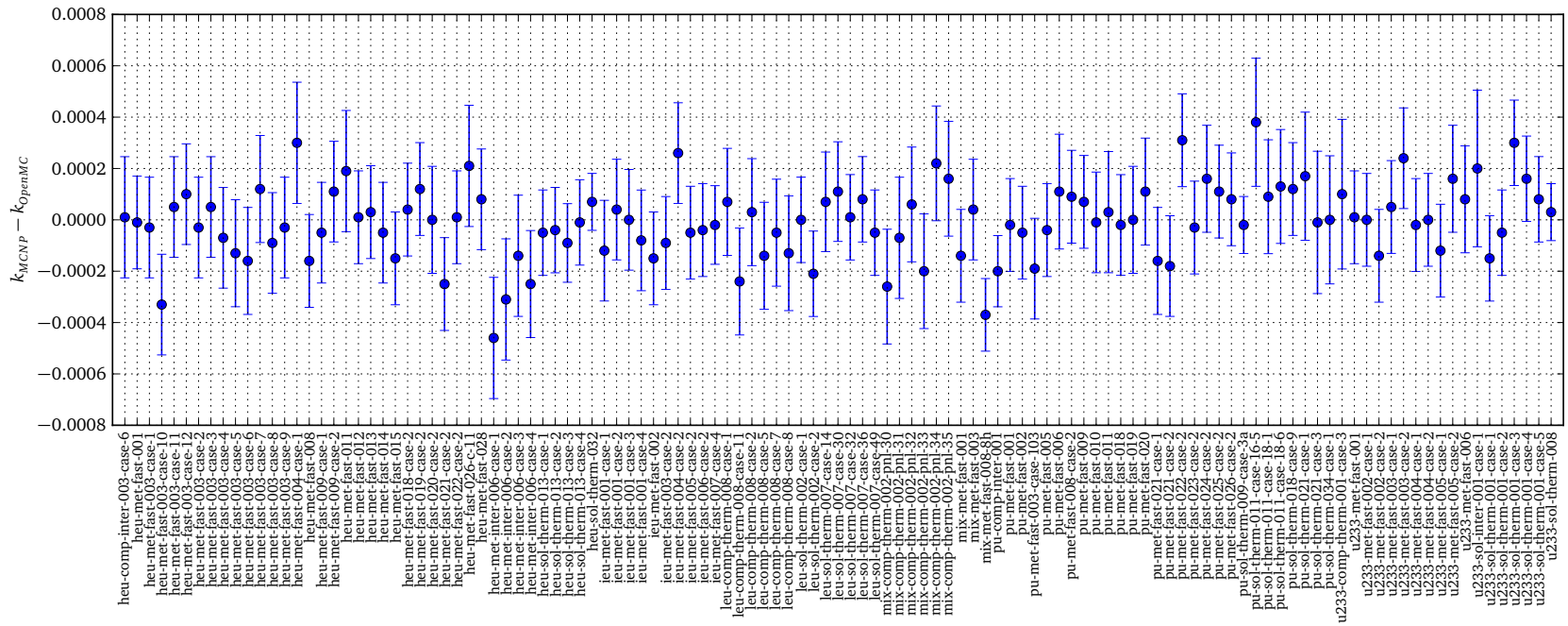


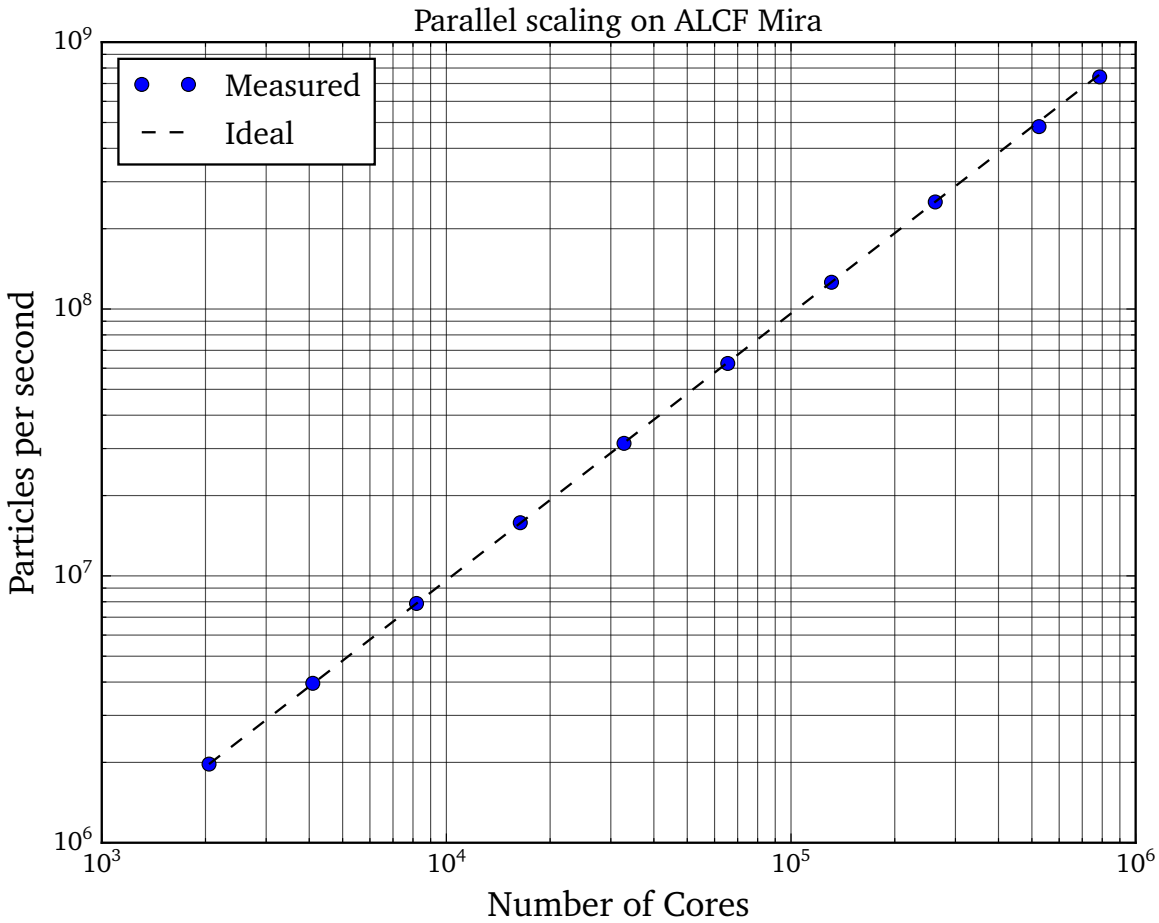

## Continuous Angle



# OpenMC is validated against MCNP Criticality Benchmark Suite

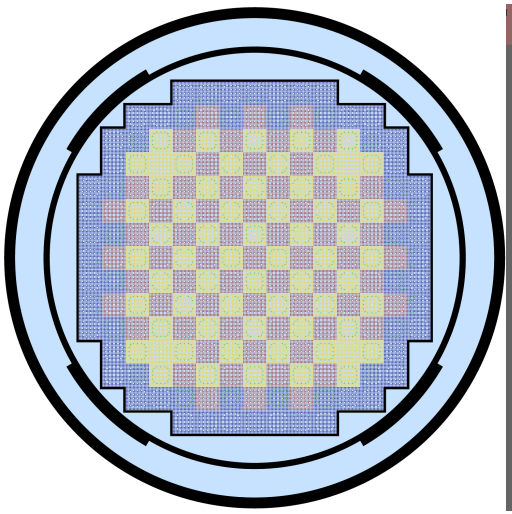
117 configurations with different spectra, materials, enrichment

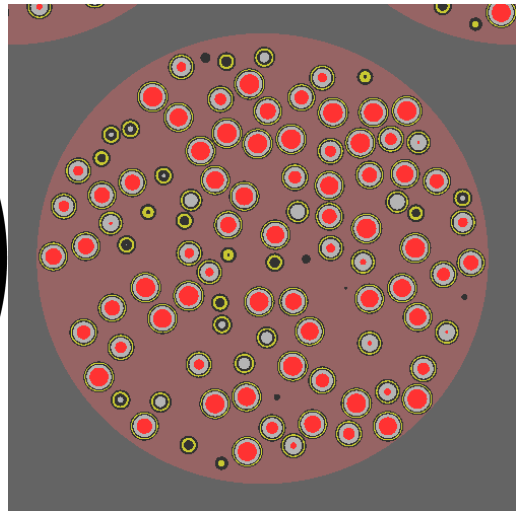


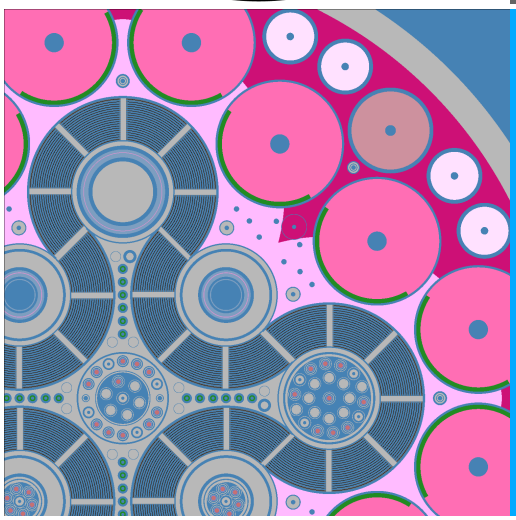


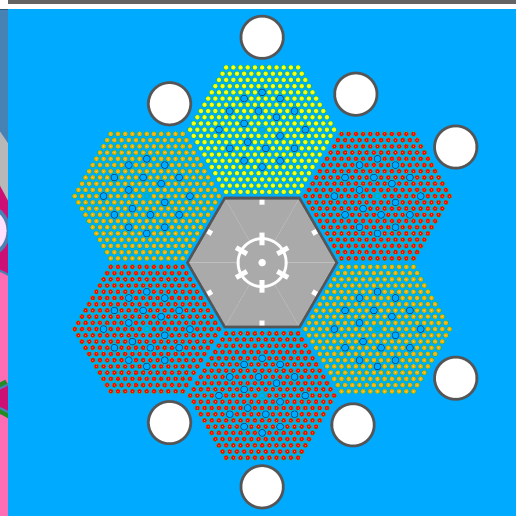

OpenMC scales linearly up to $\infty$ processors

(786,000 cores, 3,150,000 threads on Mira supercomputer)









OpenMC can model a wide variety of geometries, for example:

Full-core PWR

TRISO particles

ATR

What makes OpenMC special is its Python-powered input generation and post-processing


```
>>> import openmc
```

OpenMC is 46,000 lines of F90 and 46,000 lines of Python

## The OpenMC workflow:

1. Write Python code describing the problem.
2. Use `.export_to_xml()` to create XML files.
3. Run OpenMC (using Python or shell). This creates `tallies.out` and `statepoint.h5` output files.
4. Read `tallies.out` with a text editor or read `statepoint.h5` with Python.

# OpenMC Features

**Public Release**

- **Modes:** Fixed source and k-eigenvalue
- **Geometry:** CSG second order, universes, translations, rotations, rectangular and hexagonal lattices
- **Cross Sections:** HDF5-ACE, MG, WMP-beta
- **Physics:** neutron transport, S($\alpha$, $\beta$) tables, URR probability tables, free gas scattering, resonance upscattering, ...
- **Acceleration:** CMFD
- **Parallelism:** Distributed/shared memory via MPI/OpenMP, replication
- **Input:** XML or Python API
- **Output:** HDF5
- **Diagnostics:** Shannon entropy, iso-in-lab scattering, particle tracking files

**Private Branches**

- **URR:** Equiprobable surfaces for temperature interpolation, on-the-fly URR
- **Parallelism:** Domain decomposition, tally servers
- **Physics:** Resonance upscattering with WMP, depletion, photon transport, continuous material tracking, multigrid time dependence
- **Tallies:** Functional expansion tallies
- **Acceleration:** Low order transport
- **Input:** VERAin converter (CASL)
- **UQ:** IFP, Clutch, Differential tallies
- **Diagnostics:** Center-of-mass variance