CESAR Update

Andrew Siegel, ANL

CESAR Director

Outline

- Simulating a reactor core
- CESAR co-design process
- Exemplar: Monte Carlo Neutronics

CESAR Mission

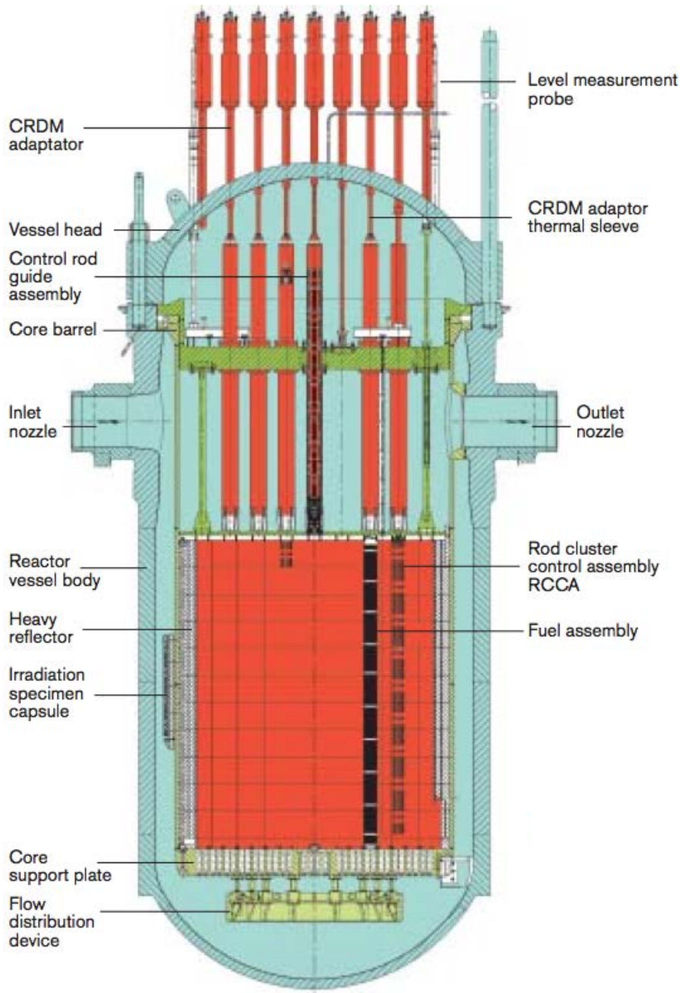
- To work with industry and DOE research partners to influence the design of future hardware architecture, system software, and applications based on the key algorithms underlying computational nuclear engineering.

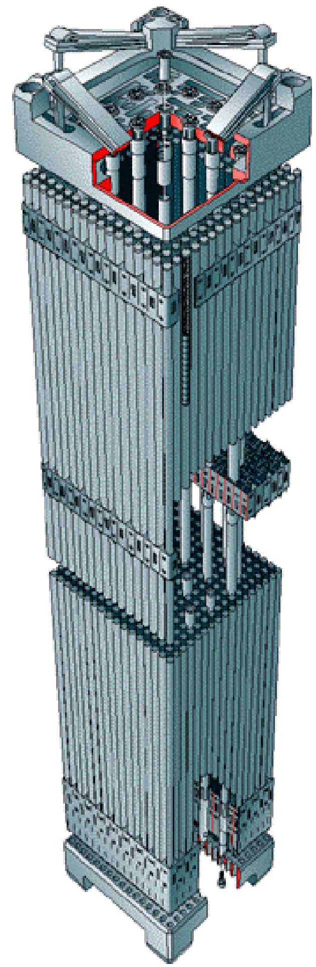
CFD + neutron transport optimized in parameter regime relevant to reactor simulation

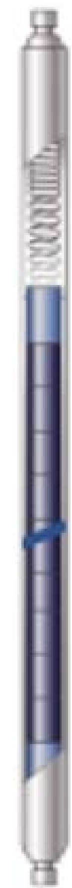
- To develop a new generation of underlying algorithms that enable the solution of significant outstanding nuclear engineering problems by leveraging exascale resources.

REACTOR CORE SIMULATION

CESAR's Nuclear Reactor Coupled Neutronics/Hydraulics Problem







Vessel →	Core →	Fuel Assembly →	Fuel Rod →	Nozzles/Spacer →	Fuel Pellet
(14 m x 4.5 m)	(4 m x 4 m)	(4 m x 20 cm)	(4 m x 1 cm)	(20 cm x 4 cm)	(1 cm x 1.5 cm)

Multi-Physics Spatial Resolution Requirements/Challenges

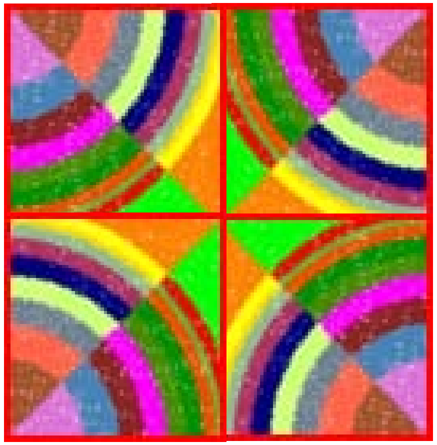
Neutronics

200 assemblies/core
264 pins/assembly
500 pellets/pin
10 rings/pellet

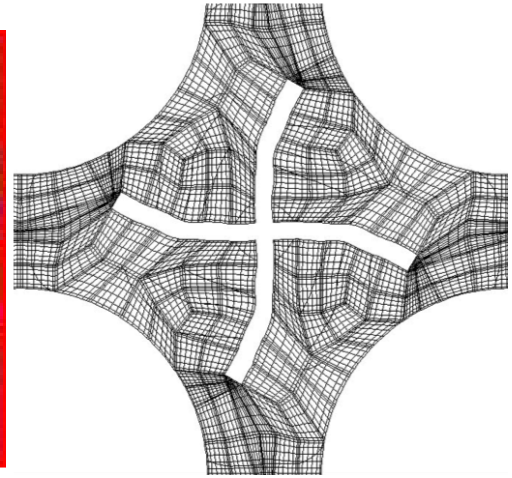
300 Million Regions

400 isotopes/ring

1 T-byte of memory



CFD (LES)



200 assemblies/core
10 spacers/assembly
2B grid points/spacer

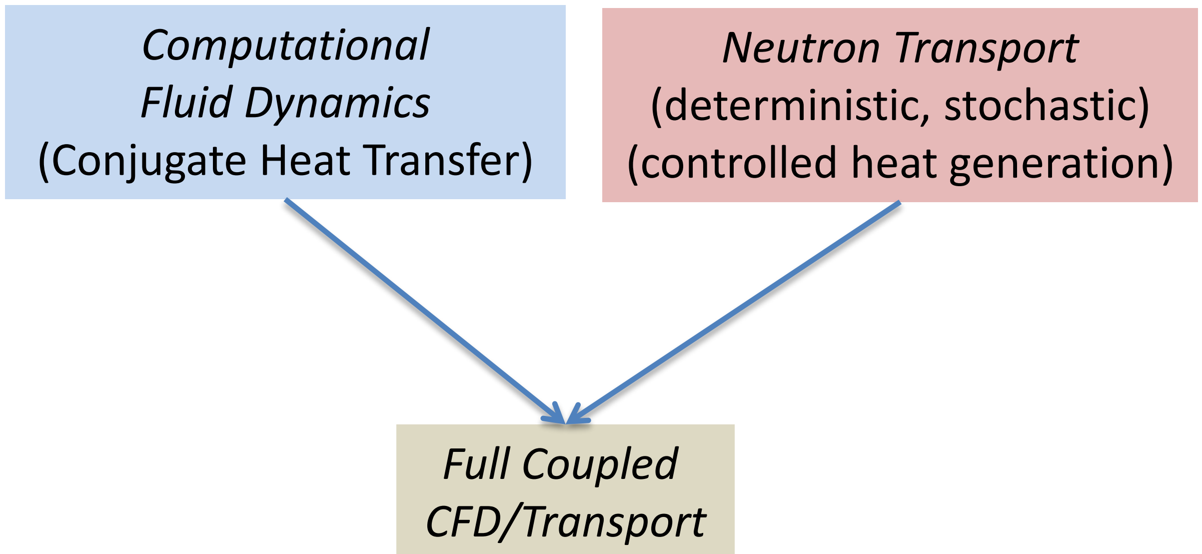
4 Trillion Grid Points

5 unknowns/grid point

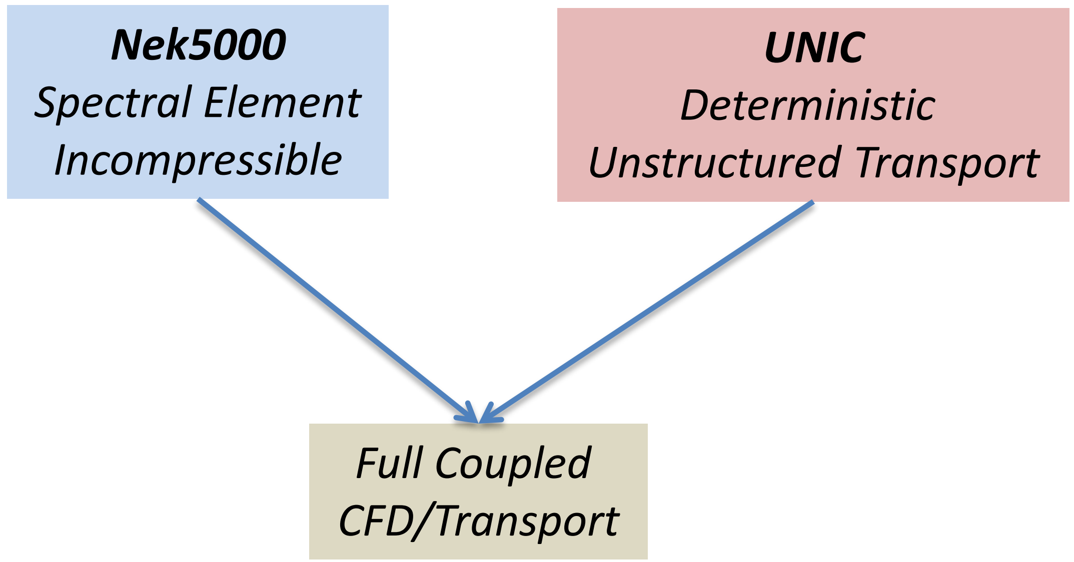
200 T-byte of memory

Grids mismatched 10,000 to 1 at spacers: major challenge for coupling data transfer

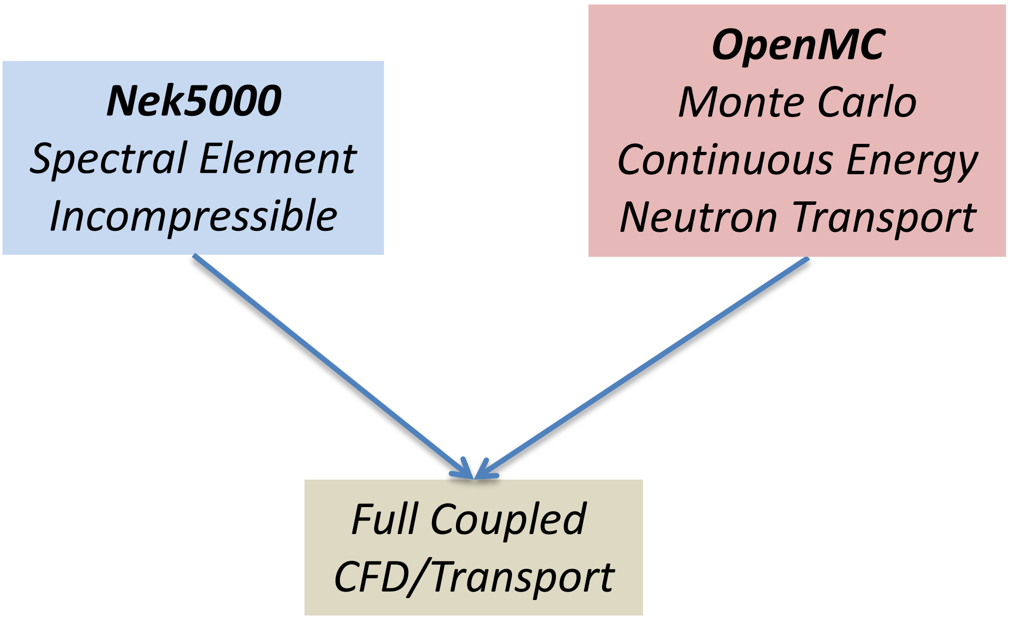
CESAR Physics



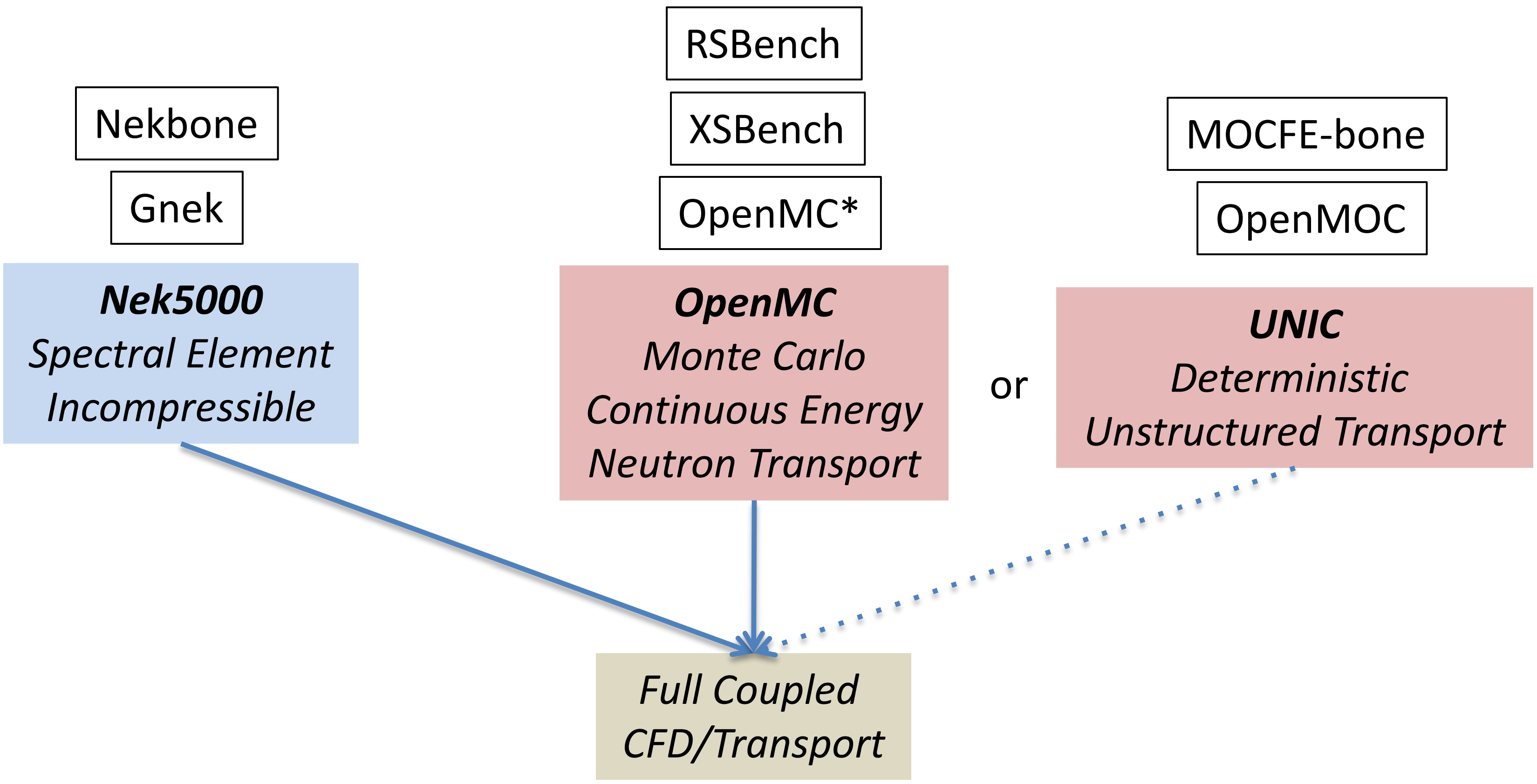
CESAR Foundational Codes



CESAR Foundational Codes



CESAR Principal Mini-apps



* A highly constrained version of the full app serves as mini-app

CESAR CO-DESIGN PROCESS

CESAR Co-design process

1. Work with community and vendors to develop/leverage exascale straw-man models/hardware simulators

1. Identify application bottlenecks in depth
   - On current platforms
   - Basic extrapolations to exascale-type architectures

1. Develop proxy applications
   - Abstract key performance features
   - Must continuously evolve
   - Must continuously verify
   - Identify/document verification bounds
   - Make user friendly
   - Distribute robustly to community
   - Work with vendors to port to simulators, guide design choices

Process is not stagnant

- Do not simply “throw over the fence”
- Participate actively in algorithmic co-design
- Three critical issues
  - As bottlenecks are encountered, identify where algorithmic compromises can be made
  - VERIFY the relevance of results for the full application
  - Be open to new mathematical formulations

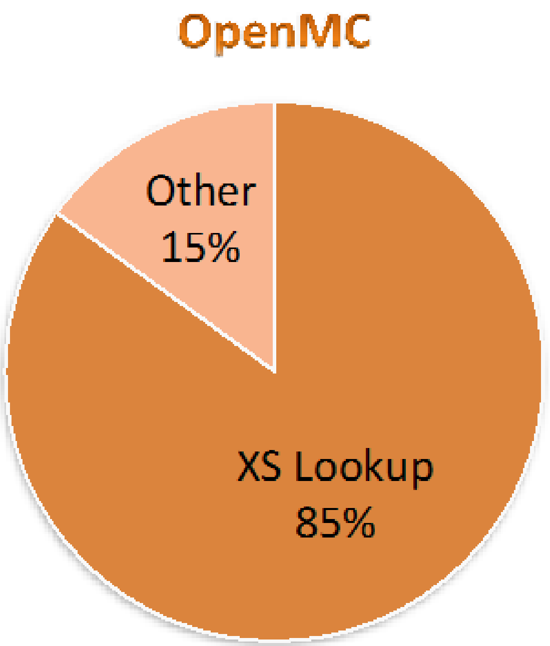
EXEMPLAR: MONTE CARLO NEUTRON TRANSPORT

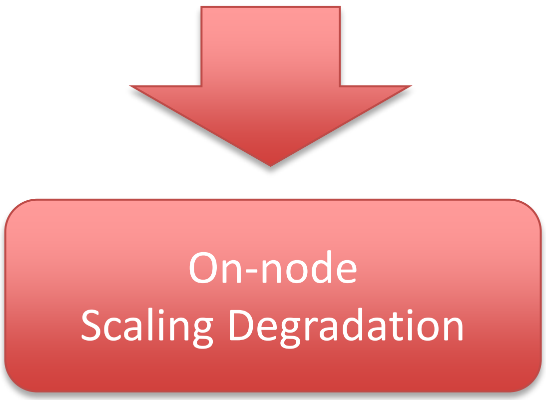
Monte Carlo Pseudo-code

```
Initialize initial neutron positions
for each batch
  for each particle in batch
    while (not absorbed)
      move particle to next interaction point
      lookup material at collision point
      for each nuclide in material
        for each reaction type
          look up micro cross-section
          build macro cross section
      sample reaction    // either collision or absorption
    end
    sample if fission occurred //guaranteed absorbed here
    if fission
      - tally //one type of tally, others possible
      - add new source sites
  end
  resample source sites //for steady state calculation
  estimate eigenvalue
end
```

On-node performance characteristics

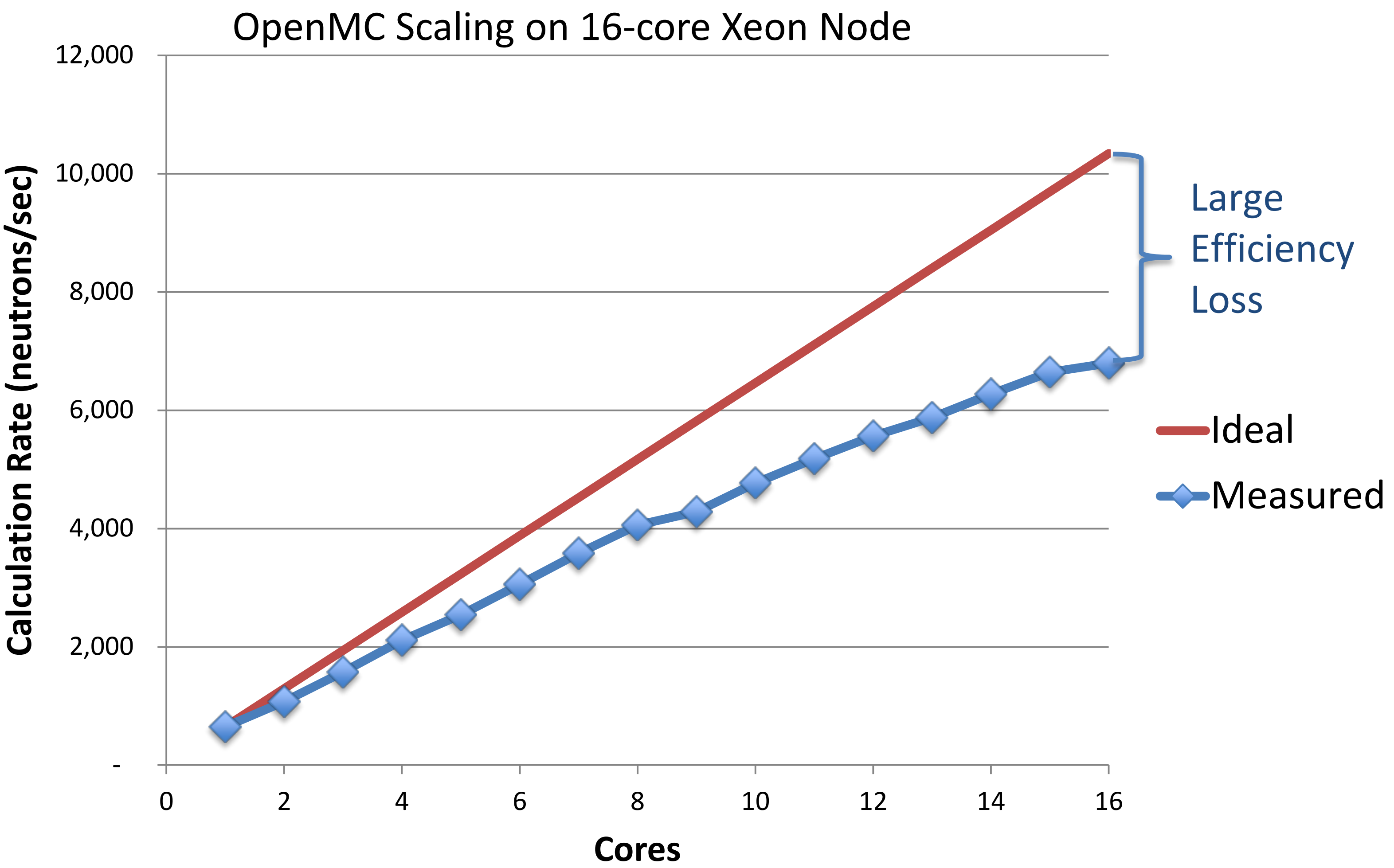
- In OpenMC, 85% of runtime is spent calculating macroscopic cross section (XS) data
- XS calculation requires many random reads from memory
- Memory subsystems are overloaded as cores are added in multi-core systems





What are implications for next-generation node architectures?

One implication: erosion of multi-core scaling



Develop Mini-app: XSBench

- Isolates key macroscopic XS algorithm from OpenMC.
- Measures the number of macroscopic XS lookups per second.
- Simple. ~1,000 lines of C.
- Easily run. Easily instrumented. Easily hacked-up.
- Cross section lookups multi-threaded with OpenMP and GPU (NVIDIA)
- Data parameters similar to those required for running a large Hoogenboom-Martin model reactor.

Role of XSBench mini-app

- If XSBench runs well, so will OpenMC

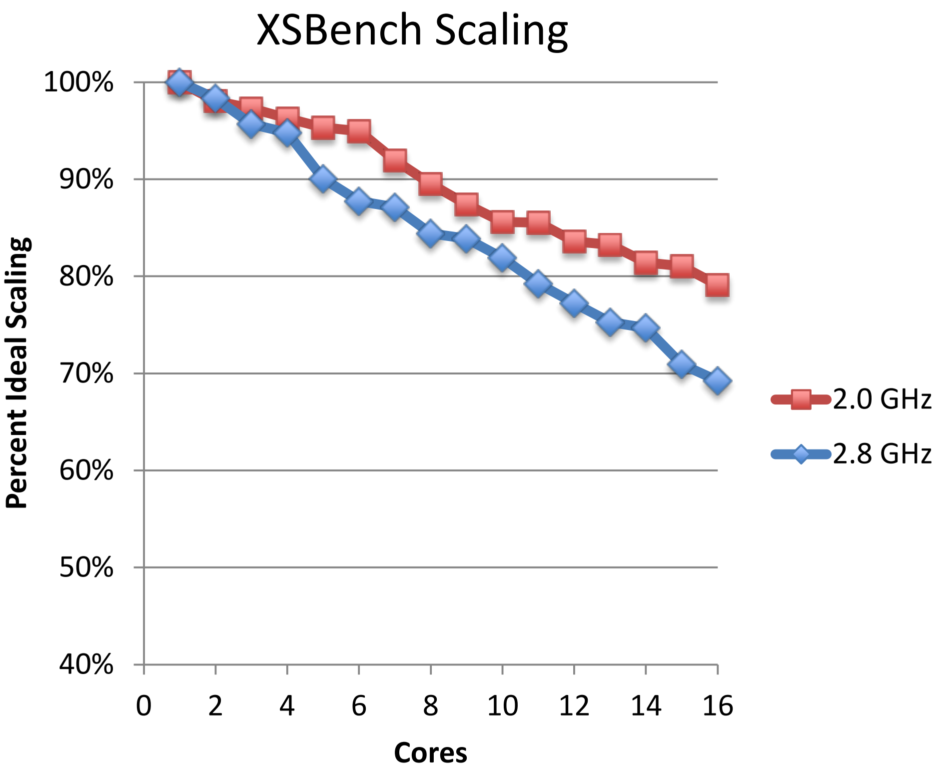
→ Optimizations to XSBench will work for OpenMC too

- Mimics real world computation without the burden of large data files and obfuscating “last-mile” science

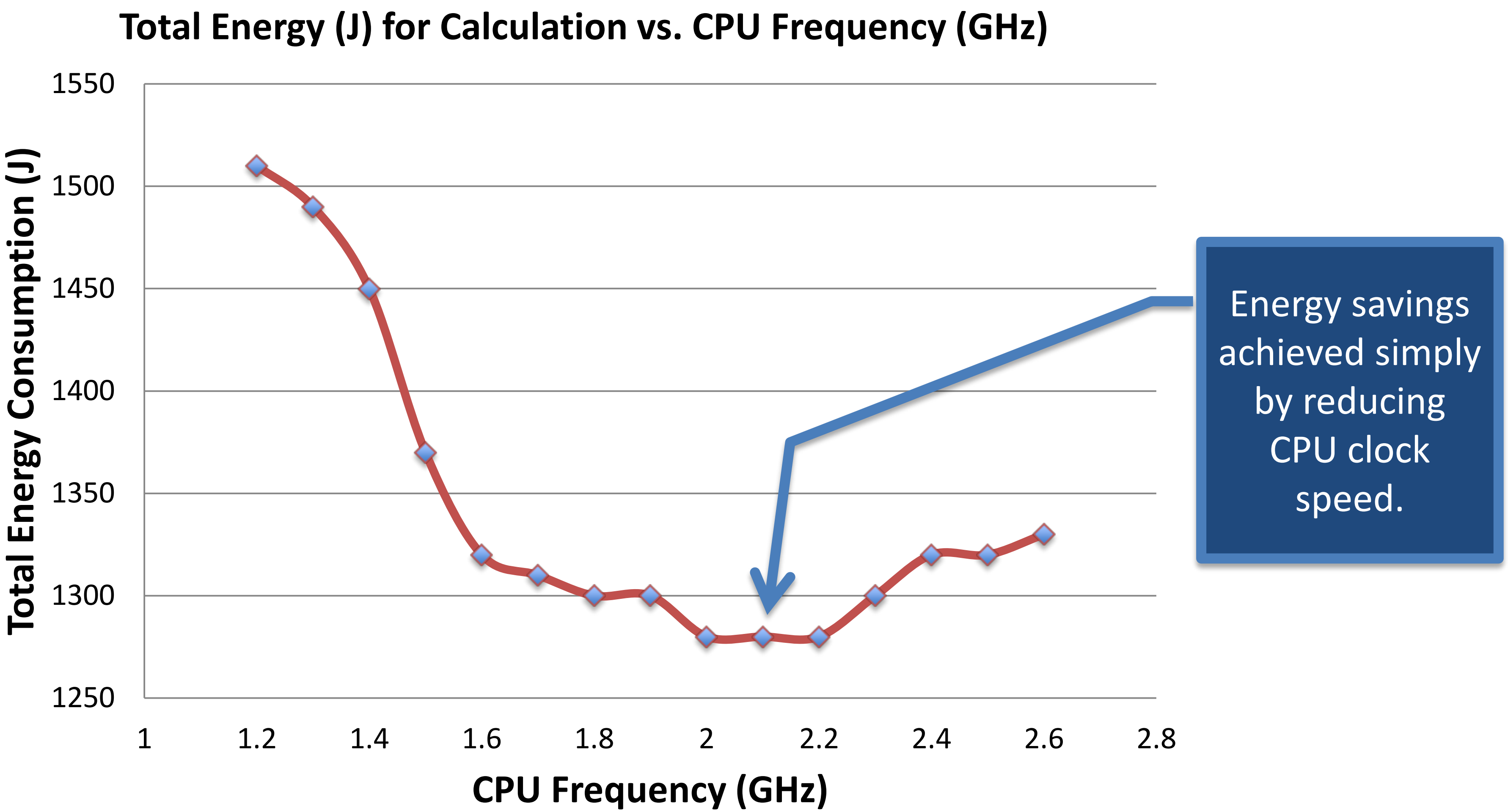
- This provides a simplified basis for:
  1. Assessing/driving exascale architectures
  2. Refining and optimizing the existing XS lookup algorithm
  3. Facilitating development of heterogeneous XS algorithms

Role of Processor Speed

- The faster the processor, the more CPU cycles will be wasted while waiting for outstanding reads.
- Calculation only slowed down by 7% while the processor speed was reduced by over 28% (clock reduced via BIOS).
- Slowing down the clock speed of a processor may:
  - Reduce bandwidth requirements
  - Improve scalability
  - Improve power efficiency



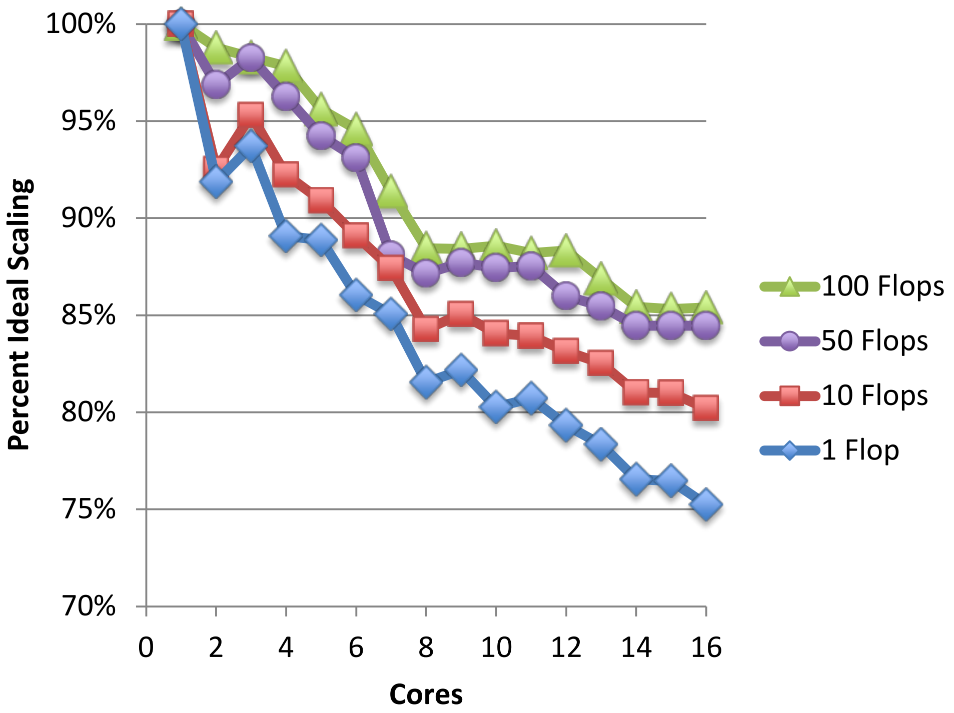
Effects of CPU Speed on Energy



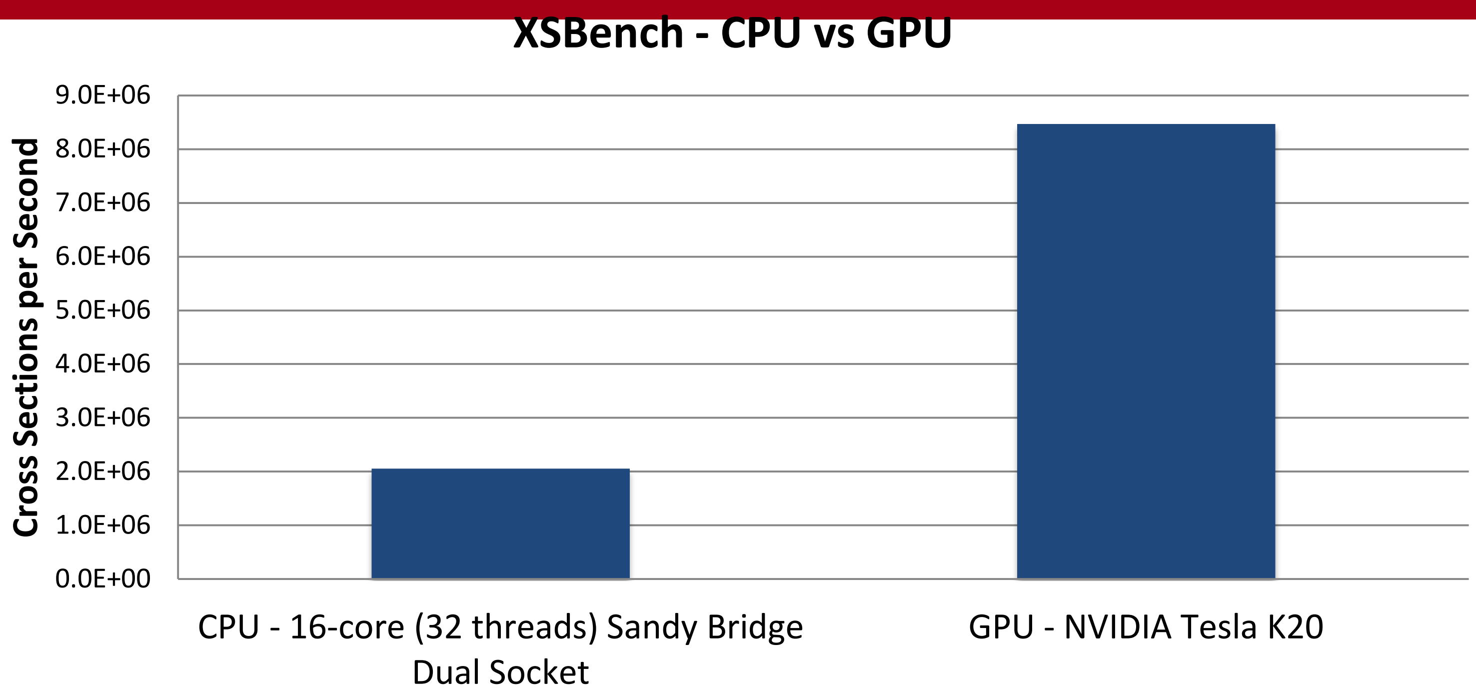
This power analysis and data provided by **Kazutomo Yoshii**

Impact of Altering FLOP/Load Ratio

- Ratio altered by adding “dummy flops” to every microscopic cross section lookup.
- Addition of 50 FLOPs per micro-XS lookup increases scaling from 75% to 85%.
- Useful if meaningful work can be done with those FLOPs, such as:
  - On-the-fly Doppler broadening
  - Cross section data decompression



Co-Design: XSBench on GPUs



- Anthony Scudiero of NVIDIA ported XSBench to CUDA – to be run on their current generation GPGPUs
- Carried out “naïve”, “mainline”, and “export” ports”
- Factor > 10x speedup, bandwidth of 128.5GB/s
- Recently ran everything on Gryphon simulator, very interesting results indicating platform may be easier to optimize than anticipated
- Sergey Blagodurov of AMD showed that using TLMSim can more than double bandwidth by identifying good data structures for caching.

Co-Design: XSBench on NVRAM

XSBench Model Size	Data Requirements (GB)
Small	0.24
Large (Default)	5.68
XL	120
XXL	252

Used for testing NVRAM prototypes

- For full scale reactor simulation, 100GB of cross section data may be required.
- XSBench has XL and XXL modes that model XS grids of this size.
- IBM and AMD have used this capability to test node prototypes that include NVRAM, to fit >100GB of data on node.
- AMD also has expressed interest in evaluating feasibility of storing tally data (over a terabyte in size) in NVRAM as well, requested a mini-app as a basis for testing

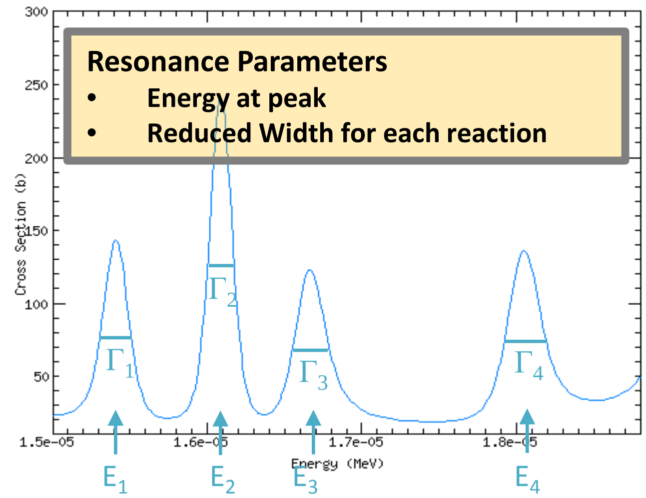
Algorithmic co-design

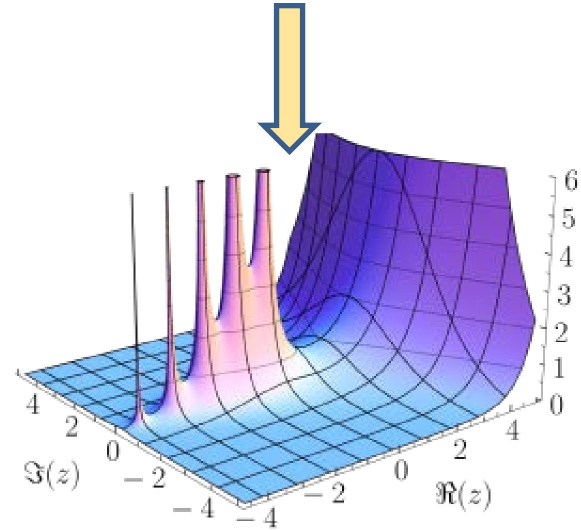
- All of this work led us to consider
  - Are there fundamental algorithmic changes that would be a better fit for exascale-type nodes?
  - “On-the-fly” cross-section construction using a small number of resonance parameters
  - Large number of FLOP/s replace random memory lookups
  - A losing proposition on single-core traditional nodes, but what about candidate exascale-type nodes?

"Multipole" Resolved Resonance Method

- **Transform resonance parameters to multipole representation for analytical Doppler integration (Hwang, 1987)**
  - **s-wave has 2 poles**
  - **p-wave has 4 poles**
- **Store only poles and residues for each resonance (~100 Mb for 400 nuclides)**
- **Cross sections are generated on-the-fly, for each neutron energy during Monte Carlo simulation**
- **Exact broadening by summing contributions of all poles at energies, E_i, to cross section at energy E**
- **Must evaluate Faddeeva function for each pole (at local temperature) by quadrature, interpolation, or C++ function evaluations**

$$W(z) = \exp(-z^2)\,\mathrm{erfc}(-iz)$$





Poles and Residues

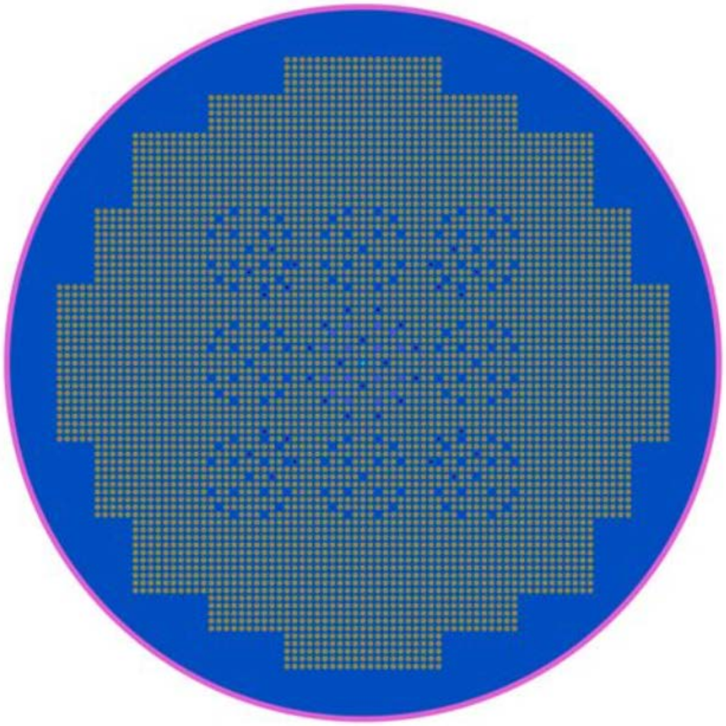
OpenMC Multipole Performance Testing

- Implemented for two uranium isotopes (U-235 and U-238)
- Tested on B&W critical core at room temperature

Method	k	Neutrons/s
ACE	1.00123 ± 0.00024	21,200
MULTIPOLE	1.00137 ± 0.00025	18,900

U-238 Absorption Rates

Lower E bound	Abs. Rate	Rel. Err. (%)
5 eV	3.5738e-02 $\pm$ 4.3e-5	-0.01 $\pm$ 0.17
11.46 eV	1.9556e-02 $\pm$ 2.9e-5	-0.22 $\pm$ 0.21
26.27 eV	1.5363e-02 $\pm$ 2.4e-5	0.15 $\pm$ 0.22
60.20 eV	1.5910e-02 $\pm$ 2.3e-5	0.26 $\pm$ 0.20
137.97 eV	9.6001e-03 $\pm$ 1.6e-5	0.24 $\pm$ 0.22
316.23 eV	7.3483e-03 $\pm$ 1.2e-5	-0.16 $\pm$ 0.23
724.78 eV	6.8581e-03 $\pm$ 8.6e-6	0.17 $\pm$ 0.18
1661.2 eV	5.5835e-03 $\pm$ 5.2e-6	-0.20 $\pm$ 0.13
3807.3 eV	4.6548e-03 $\pm$ 3.6e-7	-0.10 $\pm$ 0.11
8726.2 eV	4.0504e-03 $\pm$ 2.7e-7	-0.01 $\pm$ 0.10



RSBench: New CESAR mini-app

- Developed the new mini-app RSBench to abstract key performance characteristics of multipole on the fly method

- Results presented Wed. at EASC2014:

"Performance Analysis of a Reduced Data Movement Algorithm for Neutron Cross Section Data in Monte Carlo Simulations"