Three key issues are focus of this tutorial

1. Efficient decomposition algorithms for tallies
2. Efficient decomposition algorithms for cross-section lookup tables
3. Efficient algorithms for on-node parallelism

- Each is explained in depth in subsequent slides!
The Scale of Monte Carlo LWR Problem – tally memory

- Detailed spatial tallies required to calculate fuel isotopic inventories
- For a robust reactor simulation, tally data for one fixed point calculation is ~1Tb
- Efficient decomposition methods are needed at exascale

<table>
<thead>
<tr>
<th>Estimate of size</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>~200</td>
<td>Fuel assemblies</td>
</tr>
<tr>
<td>~700,000</td>
<td>Discrete fuel pins</td>
</tr>
<tr>
<td>~35,000,000</td>
<td>Discrete fuel pellets</td>
</tr>
<tr>
<td>~350,000,000</td>
<td>Discrete depletion zones</td>
</tr>
<tr>
<td>~1,000,000,000,000</td>
<td>Bytes of tally data for 300 nuclides</td>
</tr>
<tr>
<td>~100,000,000,000,000</td>
<td>Bytes of tally data for fuel history</td>
</tr>
</tbody>
</table>

The Scale of Monte Carlo LWR Problem – cross-section memory

- Particle tracking requires cross-section lookup at each interaction or change of material region
- Cross-section value depends on energy, nuclide, reaction type, and temperature
- This results in very large lookup tables that need to be read per particle per interaction (tenths of milliseconds)

<table>
<thead>
<tr>
<th>Estimate of size</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>~100,000</td>
<td>Cross section energy levels</td>
</tr>
<tr>
<td>300-400</td>
<td>Nuclides in fuel region</td>
</tr>
<tr>
<td>~50-100</td>
<td>Discrete temperature values</td>
</tr>
<tr>
<td>5-10</td>
<td>Reaction types</td>
</tr>
<tr>
<td>~300,000,000,000</td>
<td>Bytes of cross section data</td>
</tr>
</tbody>
</table>

The Scale of Monte Carlo LWR Problem – tracking rate

- Target accuracy for reactor analysis requires billions of particles
- Thus, reducing time to solution at exascale is a critical focus area
- This goes hand and hand with data decomposition choices
  - Potentially longer tracking times
- Scalable algorithms/hardware for on-node parallelism critical to success of Monte Carlo at exascale

<table>
<thead>
<tr>
<th>Estimate of size</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= 1.0%</td>
<td>Statistical uncertainty (2-sigma) of tallies</td>
</tr>
<tr>
<td>~ 20</td>
<td>Outer iterations (batches)</td>
</tr>
<tr>
<td>~ 300</td>
<td>Tracking rate (particles/sec) with current algorithms</td>
</tr>
<tr>
<td>~ 25,000,000,000</td>
<td>Particles simulated per batch</td>
</tr>
<tr>
<td>~ 300,000,000,000</td>
<td>Bytes of cross section data to access</td>
</tr>
<tr>
<td>~ 500,000</td>
<td>Core-hours to calculate one state point</td>
</tr>
</tbody>
</table>

Comment on classic parallelism

- Independent trajectories makes algorithm nearly embarrassingly parallel
  - Romano et. al demonstrated scaling to 100K nodes
  - Synchronization of tally/source needs careful treatment
  - Also, load balancing considerations, small effect in general
- Scalability however assumes replication of tally/x-section data across all nodes!
  - This is in general not possible for full target simulation!!
Three major issues at exascale

- Efficient decomposition strategy for 1Tb tally data
- Efficient decomposition strategy for 300Gb cross section data
- Efficient on-node threading by particle history
  - or, for SIMD, algorithm to expose SIMD parallelism

Co-design for large tallies

- Spatial domain decomposition
  - Each spatial region stores tallies within its domain
  - Dramatically reduced memory footprint
  - Must move particles at processor boundaries
    - Nearest neighbor exchanges only
    - However, very high leakage rates
    - What are required network characteristics for efficient data exchange (next slide)
    - What is the impact of load imbalances on performance and cost of resolution on exascale-type machine (subsequent slide)

- Arbitrary decomposition
  - Write tally data to remote processor
  - “tally server” model one implementation
  - What are required interconnect characteristics? (subsequent slide)
  - Hardware support for fast non-blocking write operations?

Analysis of Inter-node communication requirements for domain decomposed model: **MCCK kernel**

- Modest effect for large sub-domains
- Predict penalty up to 30x for 1/9th assembly domain size

Analysis of impact of load imbalances on domain decomposed model


Measurements of overhead for tally server model


Three major issues at exascale

- Efficient decomposition strategy for 1Tb tally data
- Efficient decomposition strategy for 300Gb cross section data
- Efficient on-node threading by particle history – or, for SIMD, algorithm to expose SIMD parallelism

Efficient decomposition strategies for cross sections

- Cross section data size:
  - ~2 G-byte for 300 isotopes at one temperature
  - ~200 G-byte for tabulation over 300K-2500K in 25K intervals
    - Data is static during all calculations
    - Exceeds node memory of anticipated machines?
      - Especially when competing with other data structures

- NV-Ram Potential?
  - Data is static during all simulations
    - Size NV-RAM needed depends on data tabulation or expansion approach
    - Static data beckons for non-volatile storage to reduce power requirements
    - Access rate needs to be very high for efficient particle tracking

- Fully replace lookup with FLOP/s
  - Cullen’s method to compute cross section integral directly from 0\(^\circ\)K data, or
  - Stochastically sample thermal motion physics to compute broadened data
    - Never store temperature-dependent data, only the 0\(^\circ\)K data
    - Cache misses will be much smaller than with tabularized data
    - Flop requirement may be large, but it is easily vectorizable

- Data compression
  - U of Michigan has shown that 20-term expansion may be acceptable
  - ~40 G-byte for 300 isotopes
    - Large manpower effort to preprocess data
    - Many cache misses because data is randomly accessed during simulations
Efficient decomposition strategies for cross sections

- Energy domain decomposition?
  - Split energy range into a small number (~5-20) energy groups
  - Bank group-to-group scattering sites when neutrons leave a domain
  - Exhaust particle bank for one domain before moving to next domain
  - Use server nodes to move cross section only for the active domain
    - Modest effort to restructure simulation codes
    - Cache misses will be much smaller than with full range tabularized data
    - Communication requirements can be reduced by employing large particle batches

- EBMS kernel

Three major issues at exascale

- Efficient decomposition strategy for 1Tb tally data
- Efficient decomposition strategy for 300Gb cross section data
- Efficient on-node threading by particle history
  - or, for SIMD, algorithm to expose SIMD parallelism

Efficient on-node parallelism

- Both coarse and fine-grained threading possible
  - Coarse: thread particle loop
  - Fine: thread nuclide macro cross section loop
  - Hybrid: both in same situation (dynamic?)
- Algorithm is inherently scalable but
  - Current multicore machines with current programming models show scaling degradation (see paper and next slide)