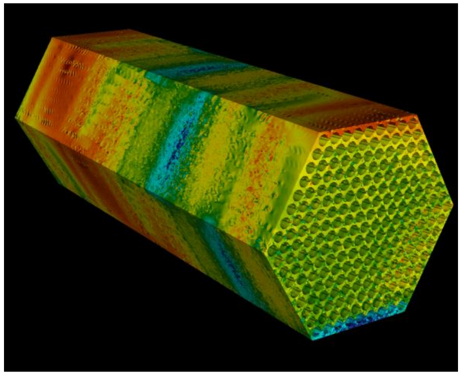
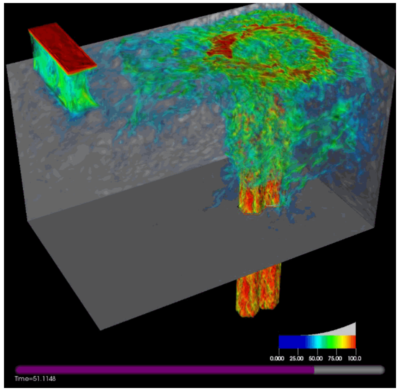
CSGF Fellows Talk

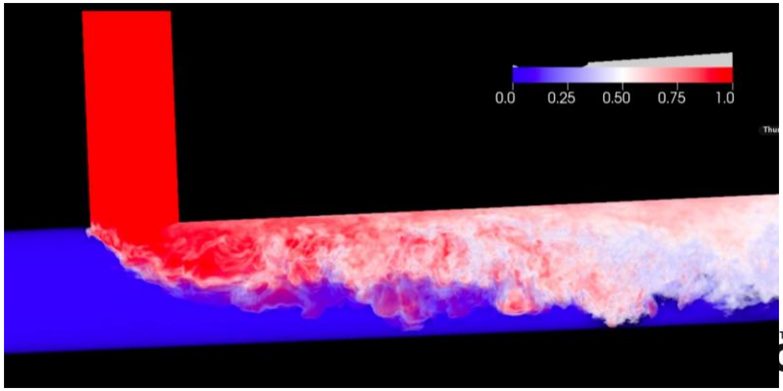
Trends in next generation HPC architectures and their impact on computational methods for nuclear reactor analysis

Andrew Siegel

Argonne National Laboratory







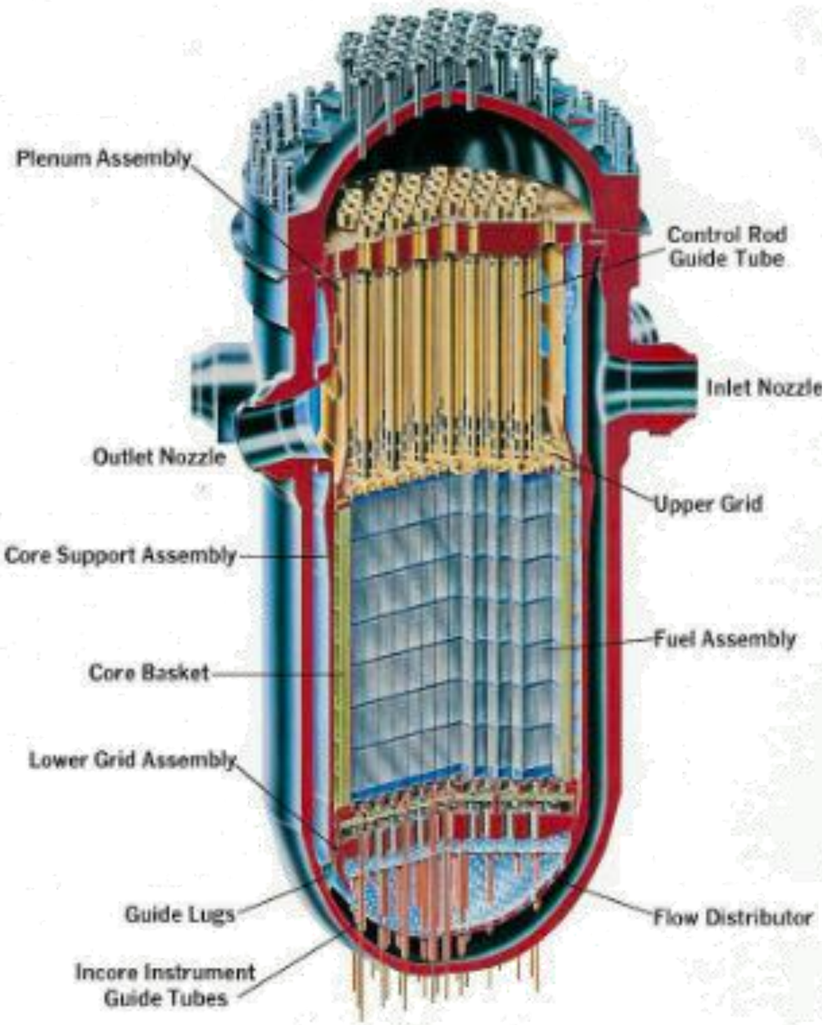
Outline

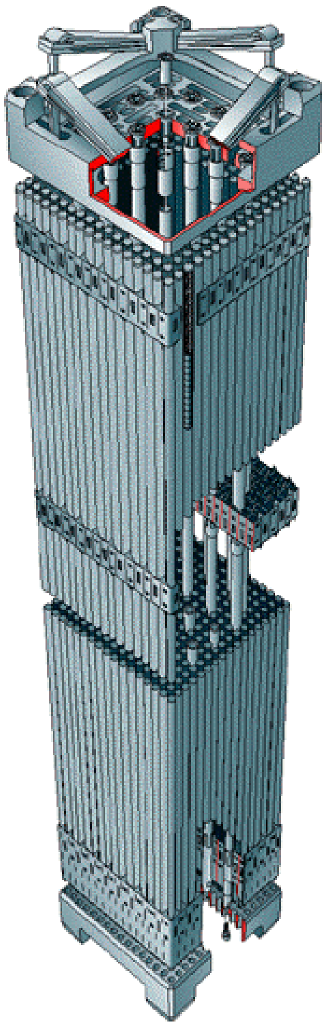
- Nuclear Reactor Simulation
- Next-generation High Performance Computing
- Three algorithmic exemplars
  - Particle vs. classic PDE-based methods
    - extreme concurrency
  - Cross section lookup vs. on-the-fly reconstruction
    - memory access vs. FLOP/s
  - Reduction of synchronicity in timestepping algorithms
    - Inherent machine-induced load imbalances

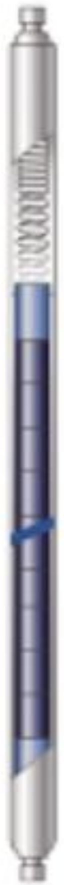
Outline

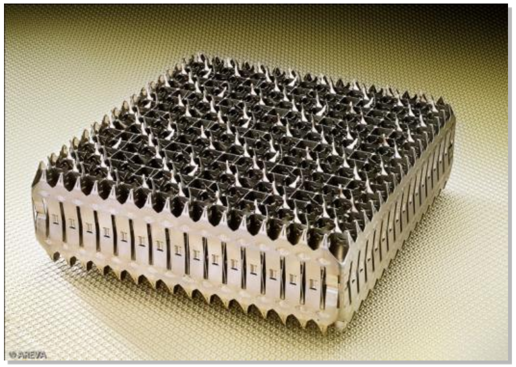
- Nuclear Reactor Simulation
- Next-generation High Performance Computing
- Three algorithmic exemplars
  - Particle vs. classic PDE-based methods
    - extreme concurrency
  - Cross section lookup vs. on-the-fly reconstruction
    - memory access vs. FLOP/s
  - Reduction of synchronicity in timestepping algorithms
    - Inherent machine-induced load imbalances

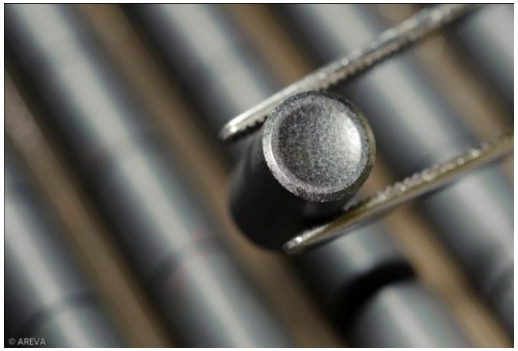
Nuclear Reactor Coupled Neutronics/Hydraulics









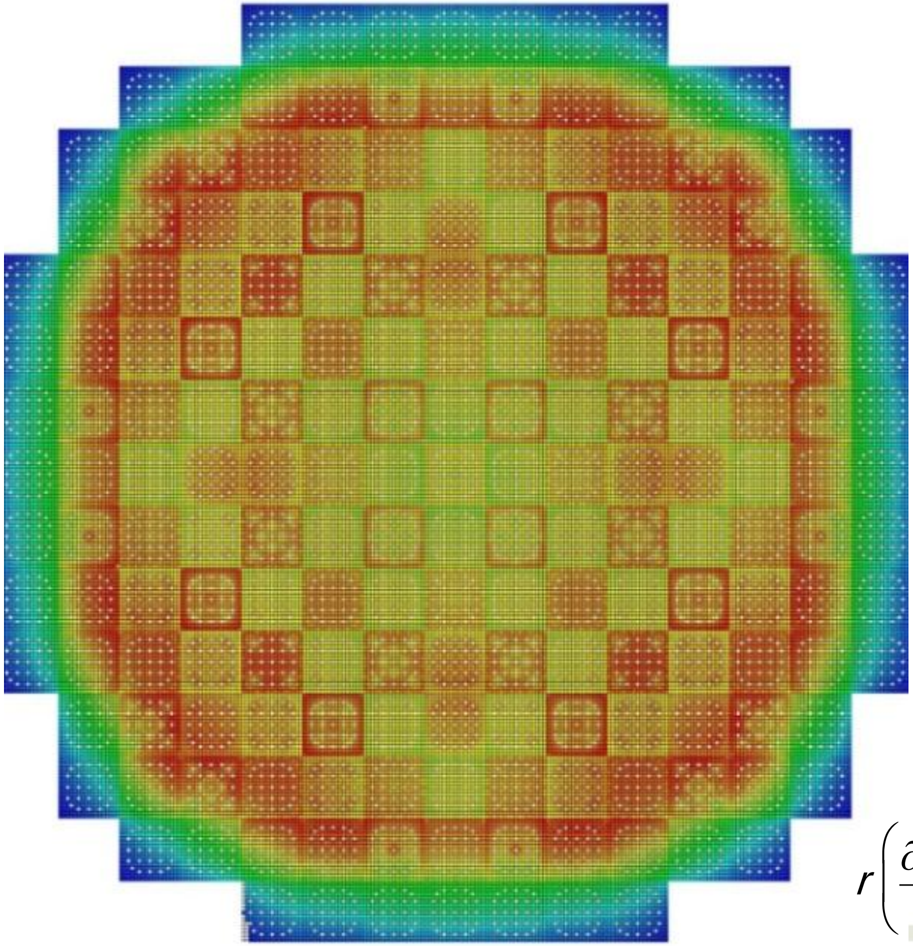


Vessel (14 m x 4.5 m) → Core (4 m x 4 m) → Fuel Assembly (4 m x 20 cm) → Fuel Rod (4 m x 1 cm) → Nozzles/Spacer (20 cm x 4 cm) → Fuel Pellet (1 cm x 1.5 cm)

Boltzmann Neutron Transport coupled to Incompressible Navier-Stokes

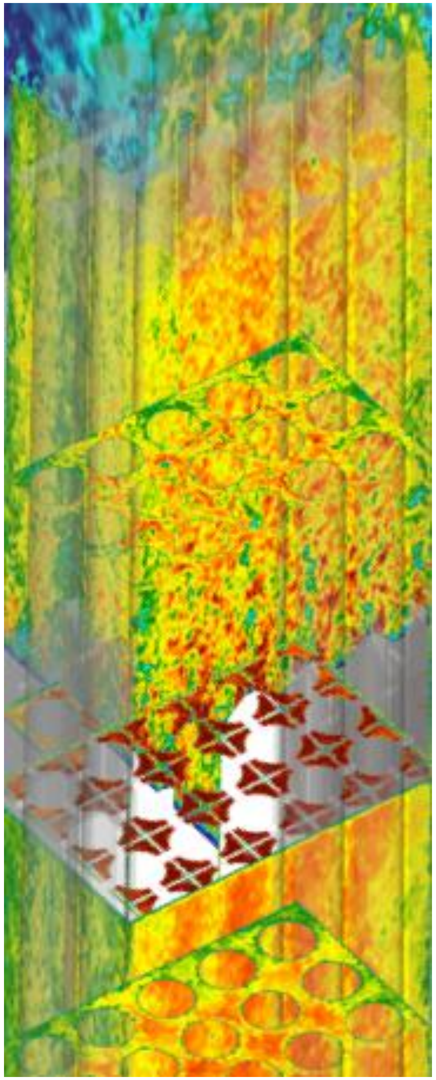
$$\frac{1}{v(E)}\frac{\partial}{\partial t}y(\vec{r},E,\vec{W},t)+\vec{W}\cdot\nabla y(\vec{r},E,\vec{W},t)+S_t(\vec{r},E,t)y(\vec{r},E,\vec{W},t)$$

$$=\int dE'\int d\vec{W}'\left[\left(\frac{c(E)}{k_{eff}}nS_f(\vec{r},E',t)+S_s(\vec{r},\vec{W}\rightarrow\vec{W},E'\rightarrow E,t)\right)y(\vec{r},E',\vec{W},t)\right]$$



$$r c_p(T)\frac{\P T(\vec{r},t)}{\P t}=$$

$$\nabla\cdot k(T)\nabla T(\vec{r},t)+\dot{q}(\vec{r},t)$$



$$\nabla\cdot\vec{v}(\vec{r},t)=0$$

$$r\left(\frac{\partial\vec{v}(\vec{r},t)}{\partial t}+\vec{v}(\vec{r},t)\cdot\nabla\vec{v}(\vec{r},t)\right)=-\nabla p(\vec{r},t)+m\nabla^2\vec{v}(\vec{r},t)+\vec{f}(\vec{r},t)$$

Spatial Resolution Requirements

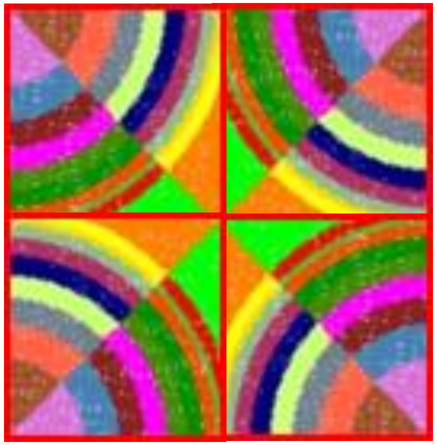
Neutronics

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

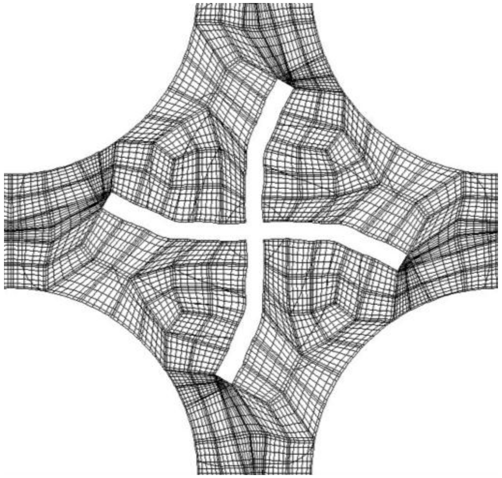
300 Million Regions

400 isotopes/ring

1 T-byte of memory



CFD (LES)



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

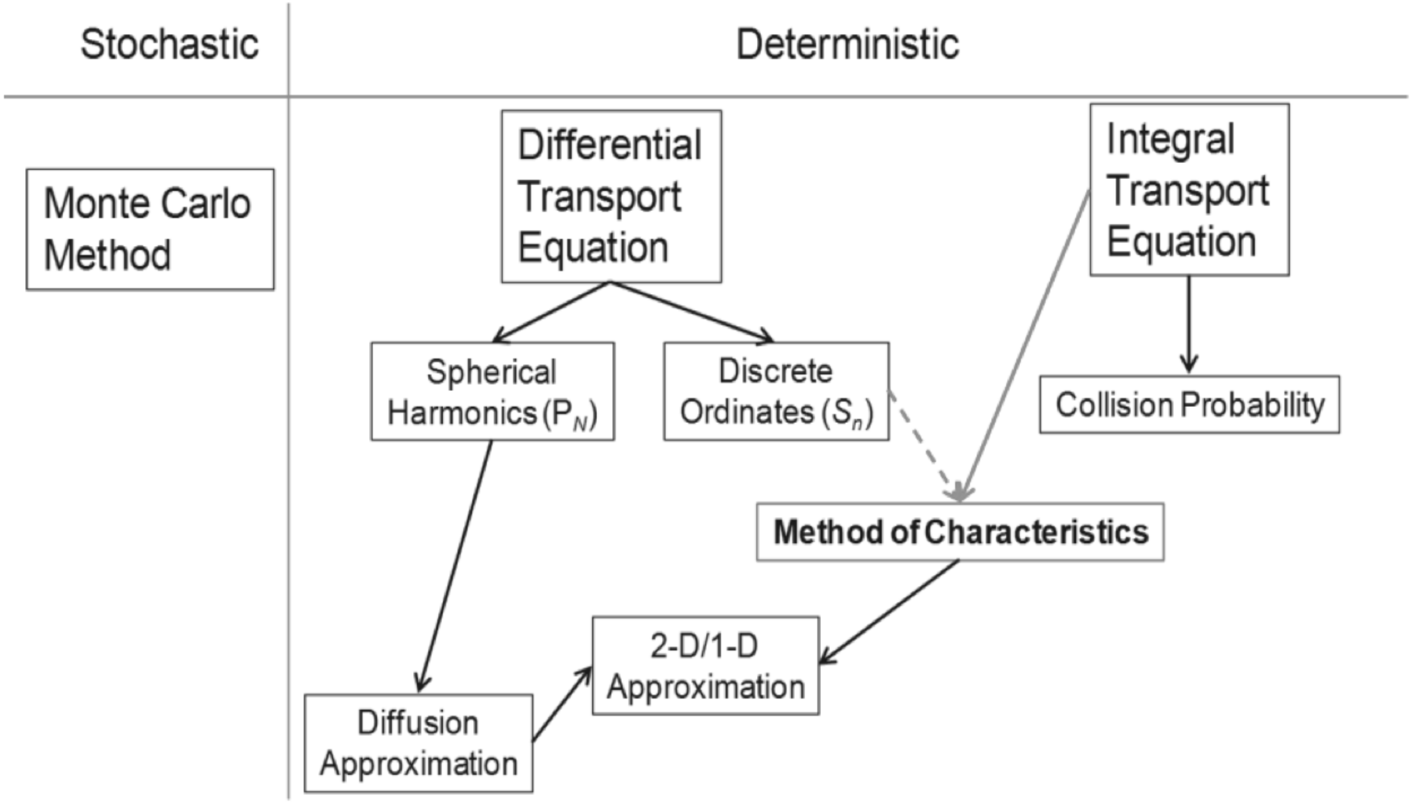
4 Trillion Grid Points

5 unknowns/grid point
200 words storage

~5 PB of memory

Grids are mismatched > 10,000-to-1 at spacers.

Taxonomy of methods for transport equation



Stochastic
Deterministic
Monte Carlo Method
Differential Transport Equation
Integral Transport Equation
Spherical Harmonics (P_N)
Discrete Ordinates (S_n)
Collision Probability
Method of Characteristics
2-D/1-D Approximation
Diffusion Approximation

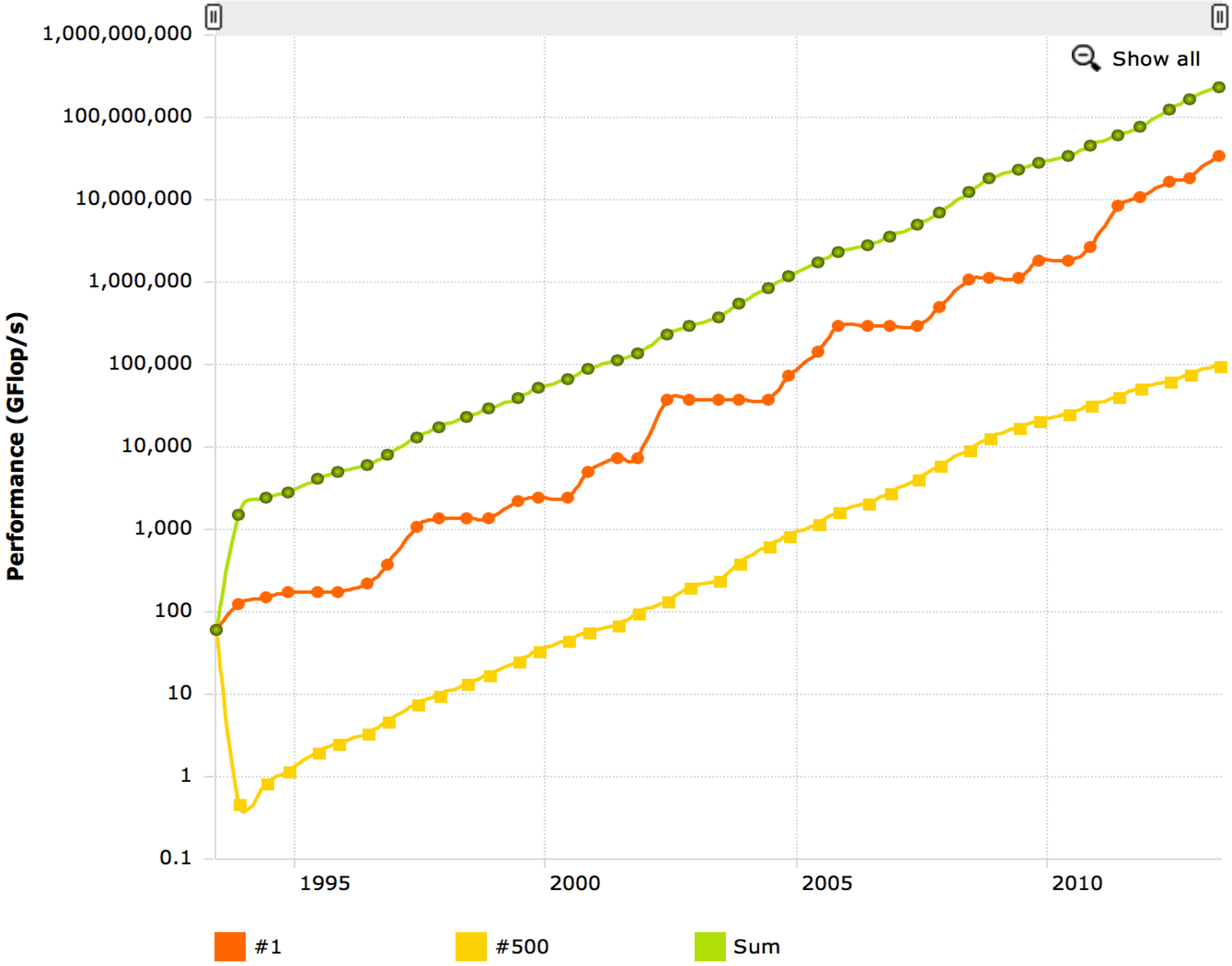
From *Brendan Kochunas, Ph.D. Thesis*

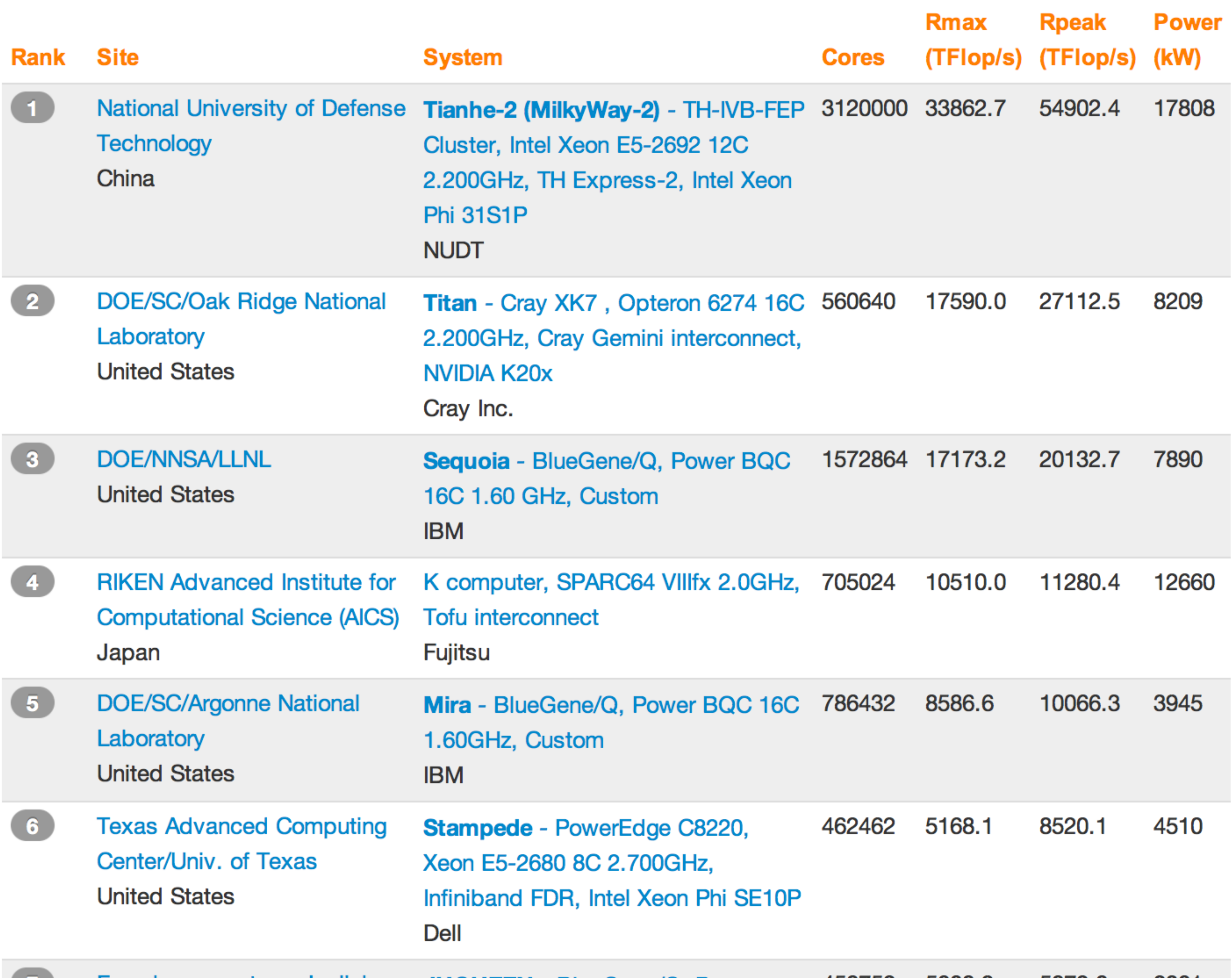
Outline

- Nuclear Reactor Simulation
- Next-generation High Performance Computing
- Three algorithmic exemplars
  - Particle vs. classic PDE-based methods
    - extreme concurrency
  - Cross section lookup vs. on-the-fly reconstruction
    - memory access vs. FLOP/s
  - Reduction of synchronicity in timestepping algorithms
    - Inherent machine-induced load imbalances

Where are we now?

Top500 Historical Performance



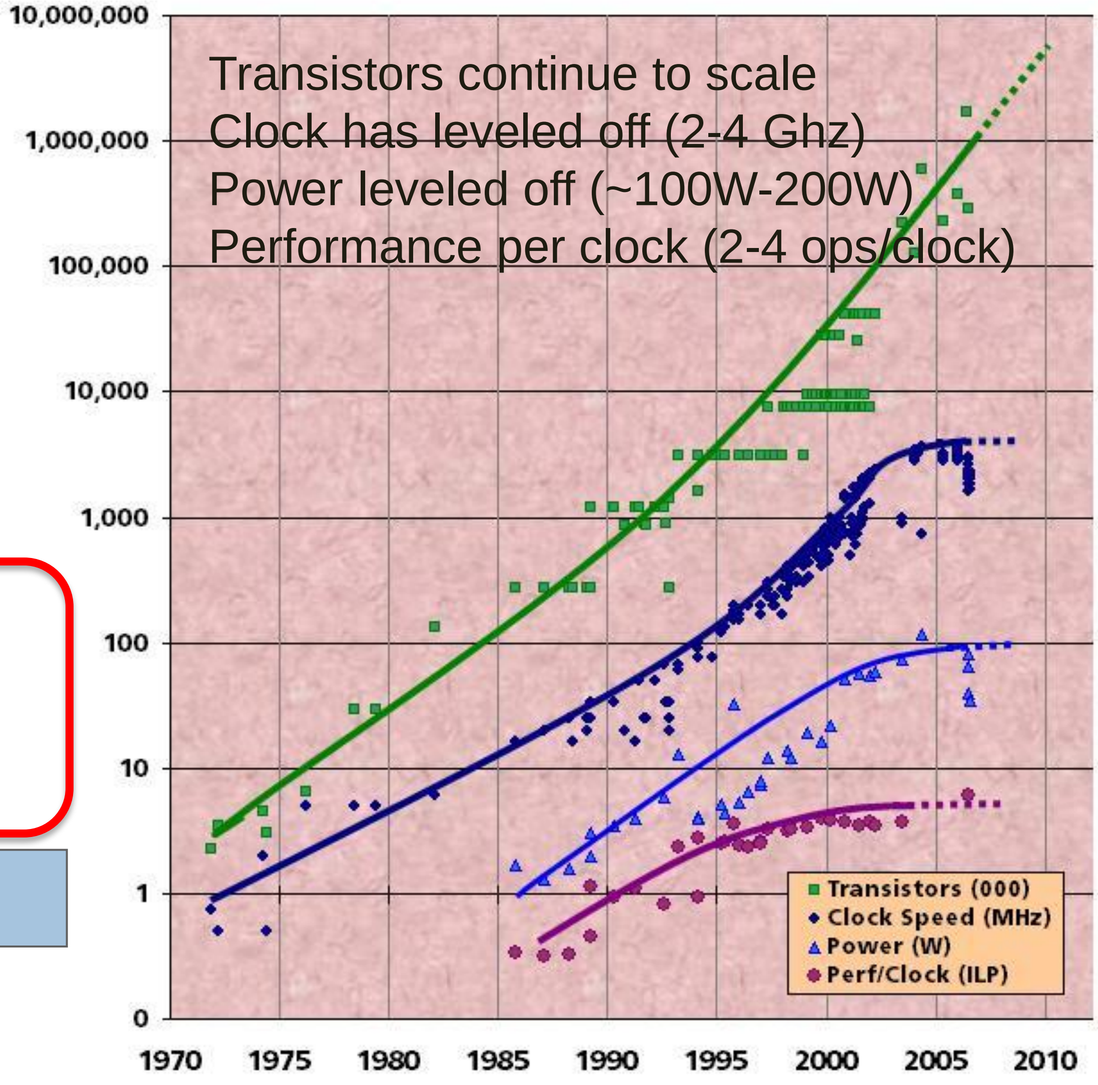


Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National University of Defense Technology China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
6	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462462	5168.1	8520.1	4510

Biggest current change at ~10 PF: on-node parallelism

- New Constraints
  - 15 years of *exponential* clock rate growth has ended
- Moore's Law reinterpreted:
  - How do we use all of those transistors to keep performance increasing at historical rates?
  - Industry Response: #cores per chip doubles every 18 months *instead* of clock frequency!

Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith


Transistors continue to scale
Clock has leveled off (2-4 Ghz)
Power leveled off (~100W-200W)
Performance per clock (2-4 ops/clock)

What are key issues moving forward?

From Shalf et. al.

		ExaNode1	ExaNode2
Flops	TF/node	10	10
Num. cores	Cores/chip	1024	1024
Num. chips	Chips/node	1	1
Mem BW	TB/s/node	1	4
Mem Cap	GB/node	256	32
2nd Mem BW	TB/s/node	NA	0.1
2nd Mem Cap	GB/node	NA	1024
L1 cache	KB/core	16	16
NIC BW	GB/s	100	400
NIC Latency	Microsec	.4	.02

- From architecture perspective

  power + cost
  - 20-40MW
  - $100-150M

- From application perspective:

  programmability
  - Far greater overall concurrency
  - 1000-way shared memory
  - Power-aware → reduced data movement → programmable memory hierarchies
  - Efficient use of instruction level parallelism
  - Efficient use of hyperthreading
  - Much less memory core → harder to hide communication costs
  - Much less bandwidth per core → data locality critical
  - Programmer-aware fault tolerance characteristics
  - Inherent processor variability + cost of global sync →movement away from BSP

This motivates application research in several areas

- Can we extract billion-way concurrency from our applications?
- Can we achieve ***on-node scalability*** on shared memory architectures?
- Can we increase computational intensity in ***data-movement*** intensive areas of apps?
- Can we minimize ***bulk synchronization*** and make applications robust to inherent variability?
- Can we mask cost of data movement in ***low memory per core*** systems?

How do these changes impact our applications?

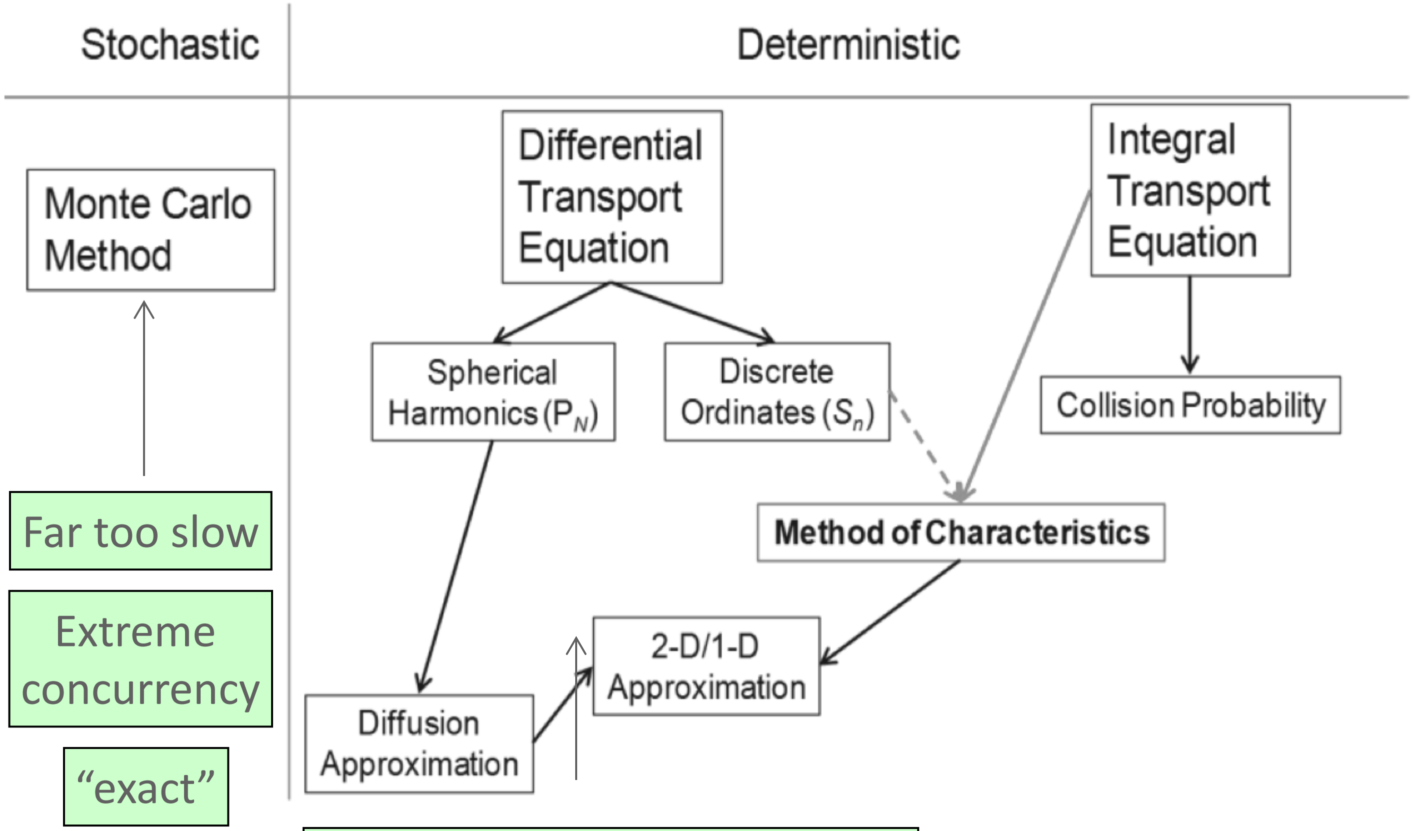
Outline

- Nuclear Reactor Simulation
- Next-generation High Performance Computing
- Three algorithmic exemplars
  - Particle vs. classic PDE-based methods
    - extreme concurrency
  - Cross section lookup vs. on-the-fly reconstruction
    - memory access vs. FLOP/s
  - Reduction of synchronicity in timestepping algorithms
    - Inherent machine-induced load imbalances

Outline

- Nuclear Reactor Simulation
- Next-generation High Performance Computing
- Three algorithmic exemplars
  - Particle vs. classic PDE-based methods
    - extreme concurrency
  - Cross section lookup vs. on-the-fly reconstruction
    - memory access vs. FLOP/s
  - Reduction of synchronicity in timestepping algorithms
    - Inherent machine-induced load imbalances

Revisiting particle methods for reactor analysis



Stochastic

Deterministic

Monte Carlo Method

Differential Transport Equation

Integral Transport Equation

Spherical Harmonics (P_N)

Discrete Ordinates (S_n)

Collision Probability

Method of Characteristics

2-D/1-D Approximation

Diffusion Approximation

Far too slow

Extreme concurrency

“exact”

PDE-based, bandwidth limited

Fast

Approximate

Do exascale machines favor Monte Carlo methods?

At high level MC algorithm very simple

```
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
```

At high level MC algorithm very simple

```
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
```

Perfectly parallel

At high level MC algorithm very simple

```
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
```

Perfectly parallel

Problem? Read-dominated

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

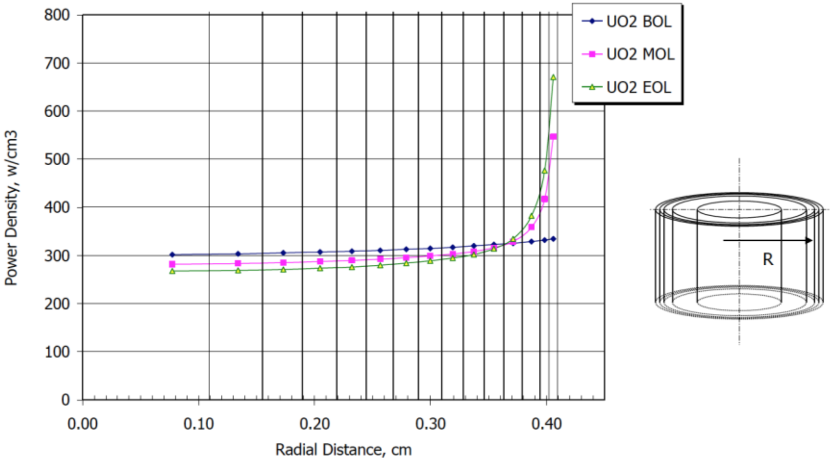
- This goes hand and hand with data decomposition choices
  - Potentially longer tracking times

- Scalable algorithms/hardware for on-node parallelism critical to success of Monte Carlo at exascale

Estimate of size	Quantity
<= 1.0%	Statistical uncertainty (2-sigma) of tallies
~ 10-20	Outer iterations (batches)
~ 300	Tracking rate (particles/sec) with current algorithms
~ 25,000,000,000	Particles simulated per batch
~ 100,000,000,000	Bytes of cross section data to access
~ 1Million	Core-hours to calculate one state point with current methods

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



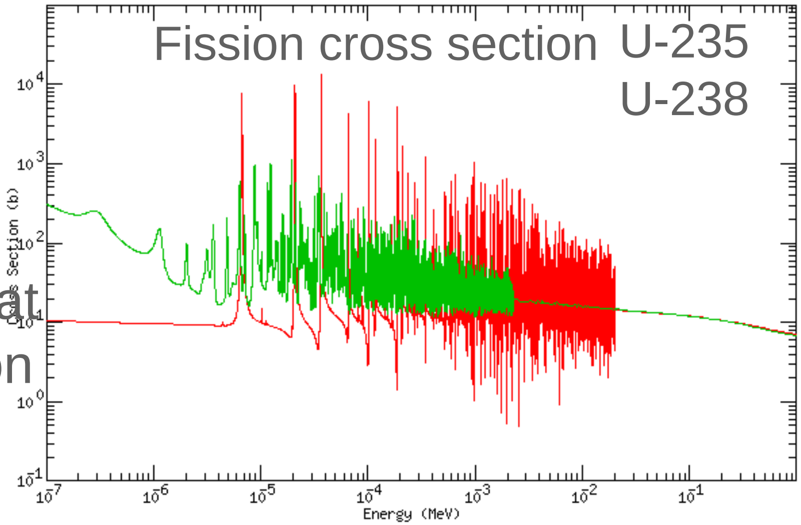
Estimate of size	Property
~200	Fuel assemblies
~700,000	Discrete fuel pins
~35,000,000	Discrete fuel pellets
~350,000,000	Discrete depletion zones
~1,000,000,000,000	Bytes of tally data for 300 nuclides
~100,000,000,000,000	Bytes of tally data for fuel history

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

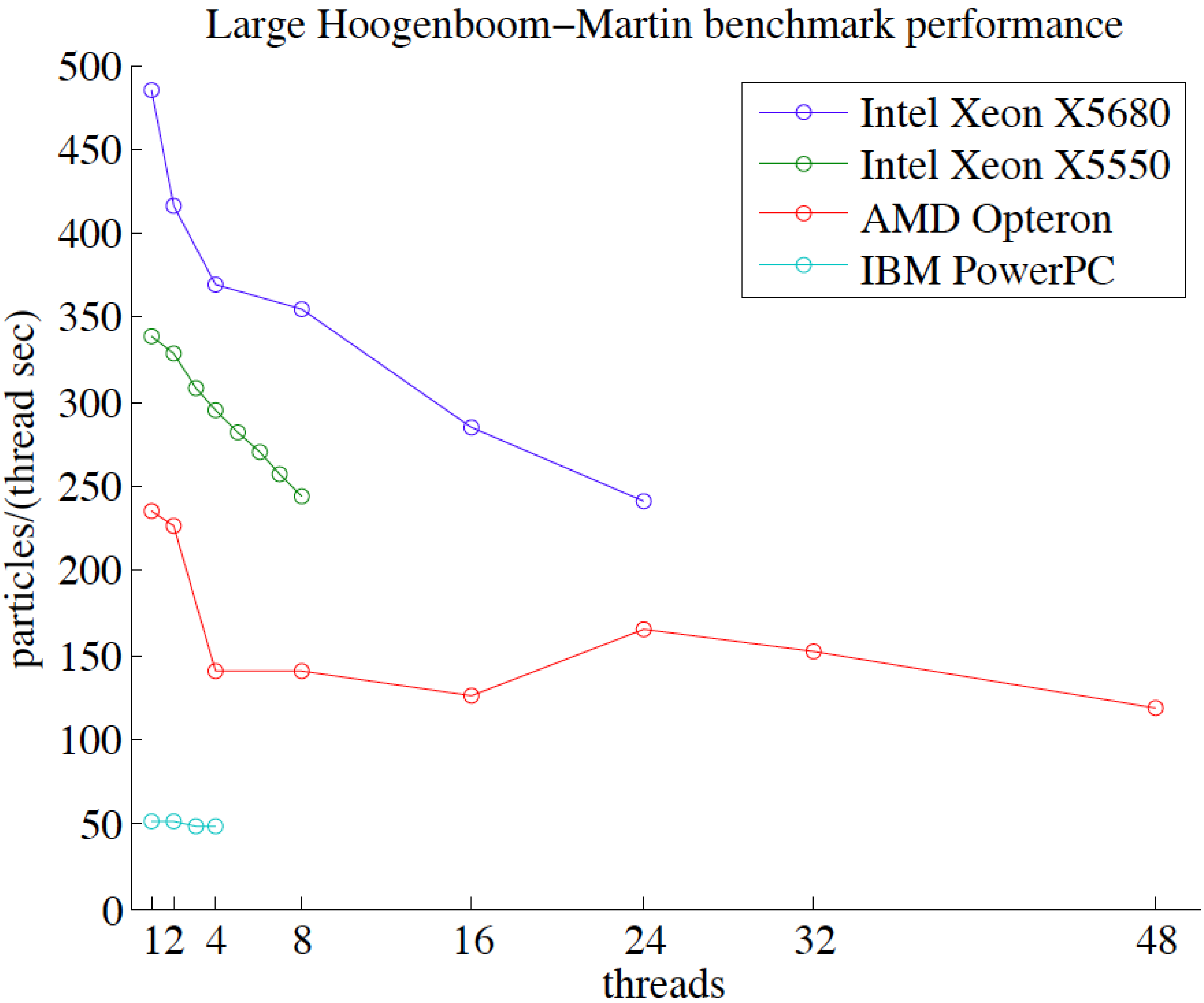
- Particle tracking requires cross-section lookup at each interaction or change of material region

- 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)



Estimate of size	Property
~100,000	Cross section energy levels
300-400	Nuclides in fuel region
~50-100	Discrete temperature values
5-10	Reaction types
~300,000,000,000	Bytes of cross section data


Large Hoogenboom–Martin benchmark performance
particles/(thread sec)
0
50
100
150
200
250
300
350
400
450
500
threads
1
2
4
8
16
24
32
48
Intel Xeon X5680
Intel Xeon X5550
AMD Opteron
IBM PowerPC

Outline

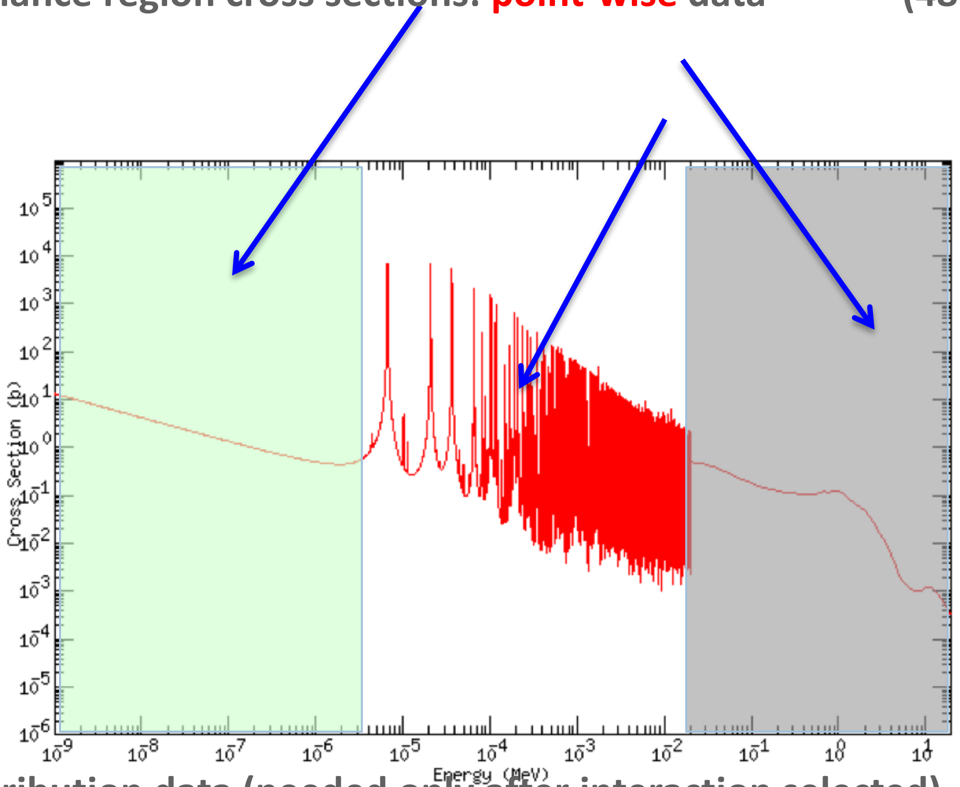
- Nuclear Reactor Simulation
- Next-generation High Performance Computing
- Three algorithmic exemplars
  - Particle vs. classic PDE-based methods
    - extreme concurrency
  - Cross section lookup vs. on-the-fly reconstruction
    - memory access vs. FLOP/s
  - Reduction of synchronicity in timestepping algorithms
    - Inherent machine-induced load imbalances

Replacing loads with FLOP/s

- Wall clock time can potentially be solved by concurrency
- Tally memory can potentially be solved by domain decomposition (or tally servers)
- Cross section memory more problematic
- Idea: if FLOP/s are cheap on next-generation machines, can we compute data “on-the-fly”?

Monte Carlo Cross Section Representations

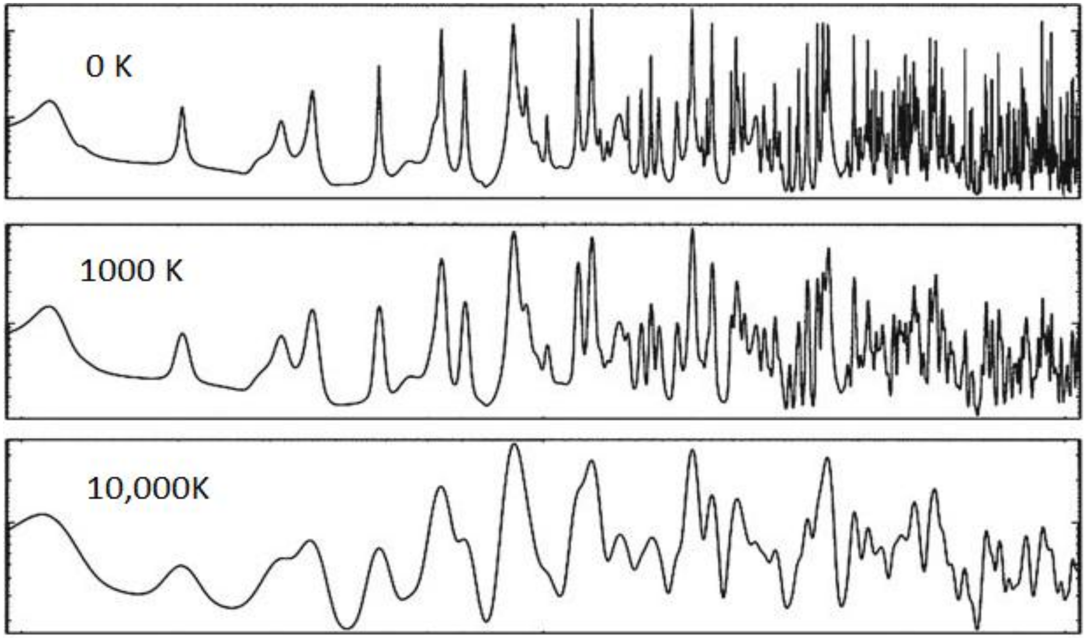
- Interaction cross section data fall into three categories:
  - Bound thermal scattering: **S(α,β) tables** vs. momentum, energy (0.5% of data)
  - Unresolved resonance region: **Probability tables** vs. energy (1.5% of data)
  - Resolved resonance region cross sections: **point-wise** data (48.0% of data)



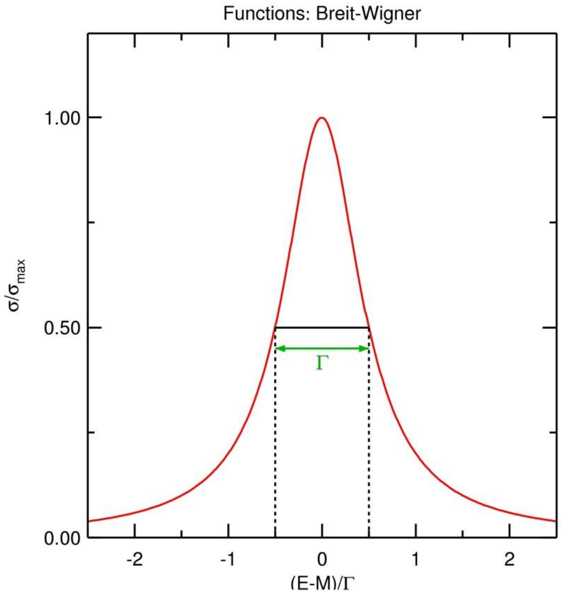
- Secondary distribution data (needed only after interaction selected) (50% of data)
- **Point-wise data evaluation totally dominates Monte Carlo run-time considerations**

Current Cross Section Representation for Monte Carlo

- Continuous piece-wise linear function from 1.e-5 eV to 20.0 MeV are generated using NJOY
- Reconstructed with an accuracy of < 0.1% relative to experimental physics data
- Linear tables permit rapid cross section interpolation at run-time
- Evaluations of cross sections consumes > 90% of Monte Carlo execution time (with 400 isotopes)
- Single-temperature data library for 400 isotopes is ~ 1 G-byte
- Reactors require temperatures from 0 to 3000K



U^{235} cross-section vs. neutron energy for 3 temperatures



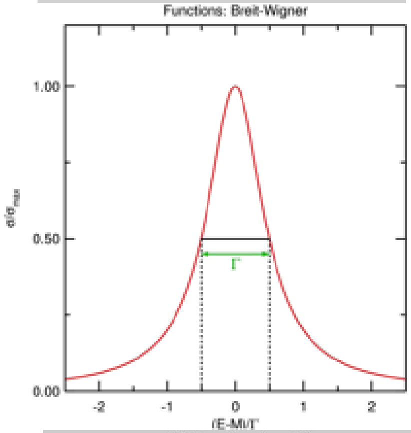
- Thermal motion causes a "smearing" of cross sections at high temperature

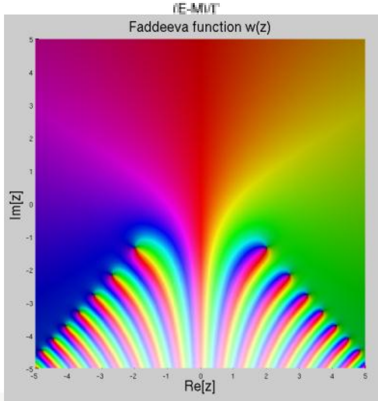
Using Physics and FLOPS to Reduce Resonance Data Movement

- We are developing an alternative physics-based method designed to permit:

1. Transformation of resonance data into “Generalized Multi-Pole” form (Huang 1987, achieves **factor 20 data reduction**)
2. **Store only 0K data** (temperature dependence is on-the-fly)
3. Point-wise data library is never constructed; rather data is generated on-the-fly for any each interacting neutron’s energy
4. Represent all reactions of each isotope with the same Poles (eliminate cache misses associated with looping over 3 reactions)
5. Exact Doppler broadening is achieved with psi/chi-like functions (evaluated directly from Faddeeva functions)

$E_t^r, E_t^i, G_t^r, G_t^i, G_c^r, G_c^i, G_f^r, G_f^i$





Forget, Xu, and Smith. Annals of Nuclear Energy (under review)

Multi-Pole Resonance Modeling

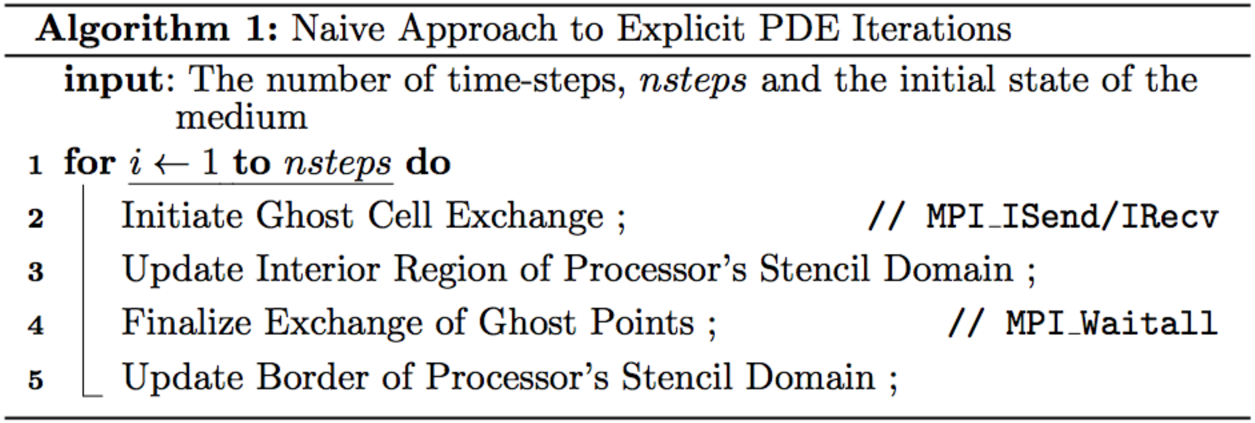
- Example of very complicated isotope U^{238}
  - Today's linear data requires about 150,000 energy points (150,000 x 3 reaction types x 2 data elements x 8 bytes) = 7 M-bytes
- Multi-pole U^{238}
  - Data has 11,500 Poles [(1 real + 1 imaginary) x (2*l* + 1) for ~3300 resonances] (11,500 x 3 reaction types x 8 bytes) = 0.25 M-bytes
- Reduce-Pole Representation of Huang (1992) treats smooth poles by regression:
  - Data has 3,500 Poles [(1 real + 1 imaginary) x (2*l* + 1) for ~3300 resonances] (3,500 x 3 reaction types x 8 bytes) = 0.08 M-bytes
- Massive data reduction will drastically reduce data movement and improve cache performance
- Small amount of data will also make GPU-like applications much more attractive.
- Tradeoffs of FLOPS for Memory
  - ~10-20 resonances contribute to each point-wise cross section
  - Faddeeva function must be evaluated for each resonance

Outline

- Nuclear Reactor Simulation
- Next-generation High Performance Computing
- Three algorithmic exemplars
  - Particle vs. classic PDE-based methods
    - extreme concurrency
  - Cross section lookup vs. on-the-fly reconstruction
    - memory access vs. FLOP/s
  - Reduction of synchronicity in timestepping algorithms
    - Inherent machine-induced load imbalances

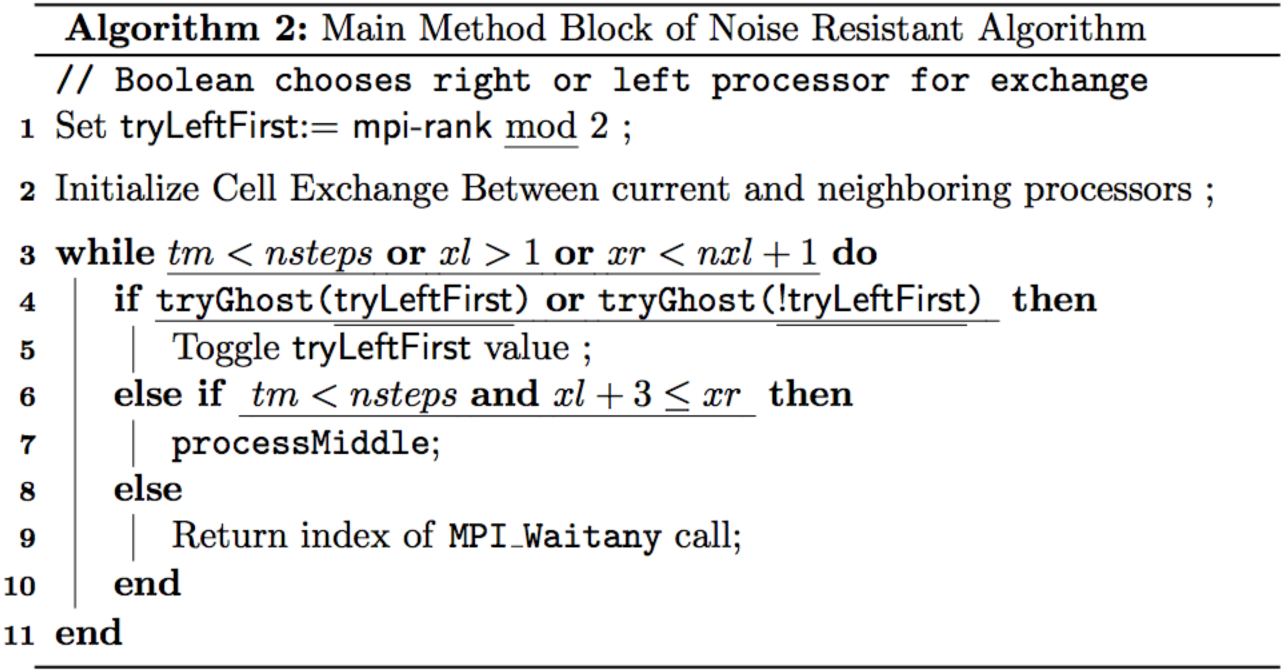
Bulk syncrhonicity

- Equal work != equal time on next generation machines
- Removal of bulk synchronization points will be one key to getting good scalability
- Lots of work in this area
  - e.g. Demmel on aasychronous Krylov solvers.
- Out initial interest also includes explicit timestepping methods



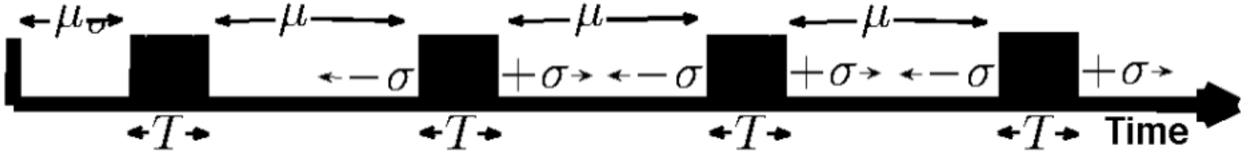
Algorithm 1: Naive Approach to Explicit PDE Iterations

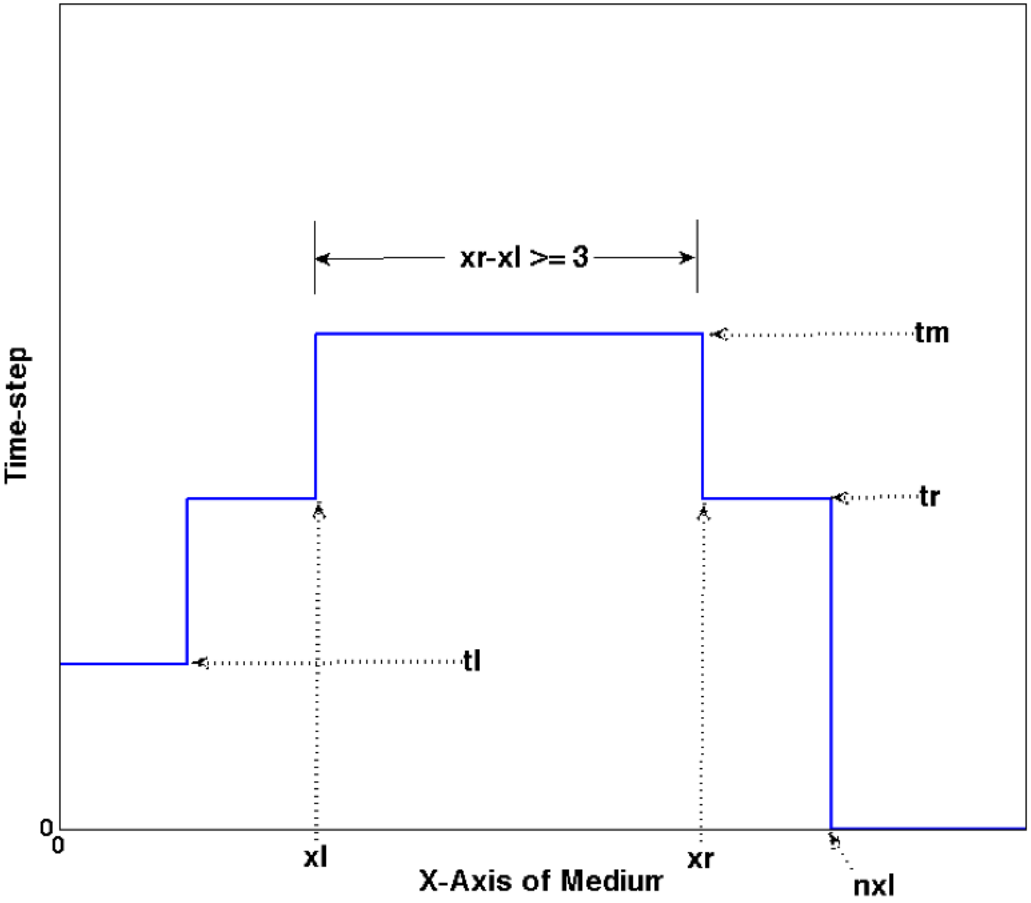
```
input: The number of time-steps, $nsteps$ and the initial state of the
       medium
for $i \leftarrow 1$ to $nsteps$ do
    Initiate Ghost Cell Exchange ;                       // MPI_ISend/IRecv
    Update Interior Region of Processor's Stencil Domain ;
    Finalize Exchange of Ghost Points ;                  // MPI_Waitall
    Update Border of Processor's Stencil Domain ;
```

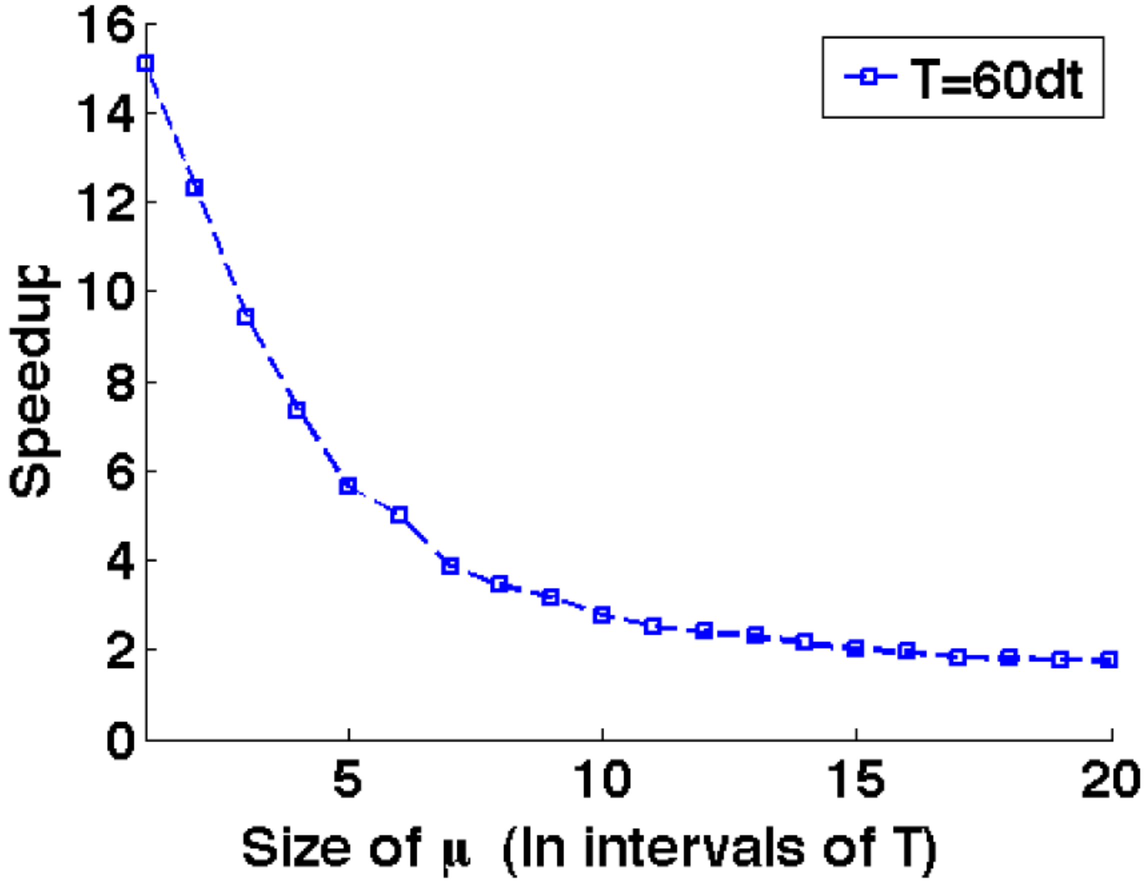


Algorithm 2: Main Method Block of Noise Resistant Algorithm

```
// Boolean chooses right or left processor for exchange
Set tryLeftFirst:= mpi-rank mod 2 ;
Initialize Cell Exchange Between current and neighboring processors ;
while $tm < nsteps$ or $xl > 1$ or $xr < nxl + 1$ do
    if tryGhost(tryLeftFirst) or tryGhost(!tryLeftFirst) then
        Toggle tryLeftFirst value ;
    else if $tm < nsteps$ and $xl + 3 \leq xr$ then
        processMiddle;
    else
        Return index of MPI_Waitany call;
    end
end
```


μ
σ
$-\sigma$
$+\sigma$
T
Time


xr-xl >= 3
tm
tr
tl
Time-step
0
0
xl
xr
X-Axis of Mediur
nxl



16
14
12
10
8
6
4
2
0
Speedup
5
10
15
20
Size of μ (In intervals of T)
T=60dt

- Fundamental architecture changes on path to exascale

- Will become increasingly difficult to make efficient use of leadership class machines

- These are forcing communities to consider fundamental new approaches
  - Not simply a matter of recoding existing algorithms

- Extreme concurrency + cost of power and thus data movement is the driving force to consider redundant re-computation vs. loads.

Extreme Concurrency

- Billion way concurrency potentially favors particle-based methods
- Every particle is tracked independently (neutrons)
- Tens of billions required for single depletion step
- Historically far too slow for required level of convergence
- Many open question, though.

Deterministic	Monte Carlo
Discretized Boltzmann	Discrete particle tracking
Computational mesh	Continuous space – tally regions
Multigroup energy	Continuous energy
Parallelization possible by energy, angle, space	Parallelization by particle, space(?), or data
Sparse PDEs – Krylov, sweeping with coarse-grid acceleration	