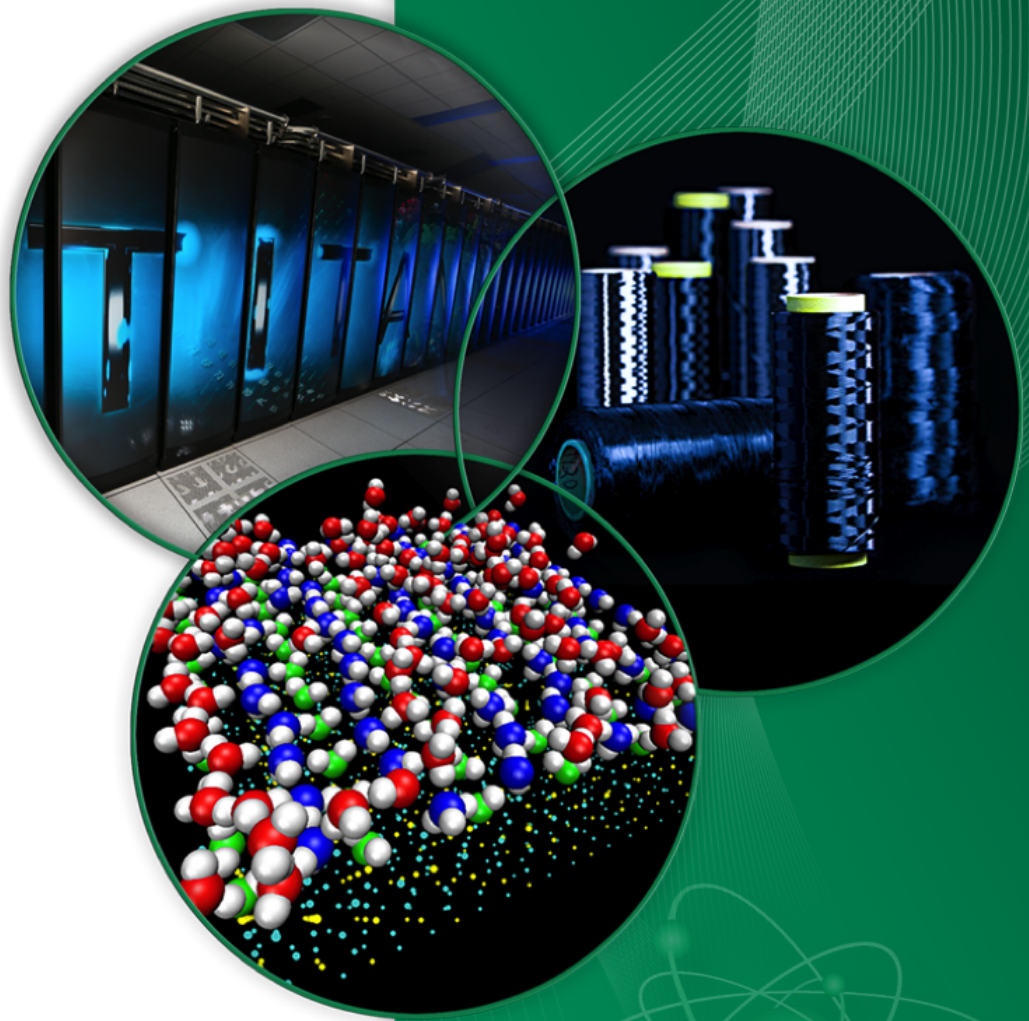
# Parallel Algorithms for Monte Carlo Linear Solvers



Stuart Slattery
Steven Hamilton
Tom Evans (PI)

Oak Ridge National Laboratory

March 17, 2015

# Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research program.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

OAK RIDGE National Laboratory

1. Motivation
2. Monte Carlo Methods
3. Domain Decomposition and Replication
4. Scaling Studies
5. Current/Future Work and Conclusions

OAK RIDGE National Laboratory

# Motivation

- As we move towards exascale computing, the rate of errors is expected to increase dramatically
- Algorithms need to not only have increased concurrency/scalability but have the ability to recover from hardware faults
- Two basic strategies:
  1. Start with current "state of the art" methods and make incremental modifications to improve scalability and fault tolerance
     - Many efforts are heading in this direction, attempting to find additional concurrency to exploit
  2. Start with methods having natural scalability and resiliency aspects and work at improving performance (e.g. Monte Carlo)
     - Soft failures introduce an additional stochastic error component
     - Hard failures potentially mitigated by replication
     - Concurrency enabled by several levels of parallelism

OAK RIDGE National Laboratory

# Monte Carlo for Linear Systems

- Suppose we want to solve $\mathbf{Ax} = \mathbf{b}$
- If $\rho(\mathbf{I} - \mathbf{A}) < 1$, we can write the solution using the Neumann series

$$\mathbf{x} = \sum_{n=0}^{\infty} (\mathbf{I} - \mathbf{A})^n \mathbf{b} = \sum_{n=0}^{\infty} \mathbf{H}^n \mathbf{b}$$

  where $\mathbf{H} \equiv (\mathbf{I} - \mathbf{A})$ is the Richardson iteration matrix
- Build the Neumann series stochastically

$$x_i = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} h_{i,i_1} h_{i_1,i_2} \ldots h_{i_{k-1},i_k} b_{i_k}$$

- Define a sequence of state transitions

$$\nu = i \to i_1 \to \cdots \to i_{k-1} \to i_k$$

OAK RIDGE National Laboratory

# Forward Monte Carlo

- Choose a row-stochastic matrix $\mathbf{P}$ and weight matrix $\mathbf{W}$ such that $\mathbf{H} = \mathbf{P} \circ \mathbf{W}$
- Typical choice (Monte Carlo Almost-Optimal):

$$\mathbf{P}_{ij} = \frac{|\mathbf{H}_{ij}|}{\sum_{j=1}^{N} |\mathbf{H}_{ij}|}$$

- To compute solution component $\mathbf{x}_i$:
  - Start a history in state $i$ (with initial weight of 1)
  - Transition to new state $j$ based probabilities determined by $\mathbf{P}_i$
  - Modify history weight based on corresponding entry in $\mathbf{W}_{ij}$
  - Add contribution to $\mathbf{x}_i$ based on current history weight and value of $\mathbf{b}_j$
- A given random walk can only contribute to a single component of the solution vector with $\mathbf{x} \approx \mathbf{M_{MC}b}$

OAK RIDGE National Laboratory

# Sampling Example (Forward Monte Carlo)

- Suppose

$$\mathbf{A} = \begin{bmatrix} 1.0 & -0.2 & -0.6 \\ -0.4 & 1.0 & -0.4 \\ -0.1 & -0.4 & 1.0 \end{bmatrix} \rightarrow \mathbf{H} \equiv (\mathbf{I} - \mathbf{A}) = \begin{bmatrix} 0.0 & 0.2 & 0.6 \\ 0.4 & 0.0 & 0.4 \\ 0.1 & 0.4 & 0.0 \end{bmatrix}$$

  then

$$\mathbf{P} = \begin{bmatrix} 0.0 & 0.25 & 0.75 \\ 0.5 & 0.0 & 0.5 \\ 0.2 & 0.8 & 0.0 \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} 0.0 & 0.8 & 0.8 \\ 0.8 & 0.0 & 0.8 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$$

- If a history is started in state $3$, there is a $20\%$ chance of it transitioning to state $1$ and an $80\%$ chance of moving to state $2$

OAK RIDGE National Laboratory

# Solving the Heat Equation: Forward Method

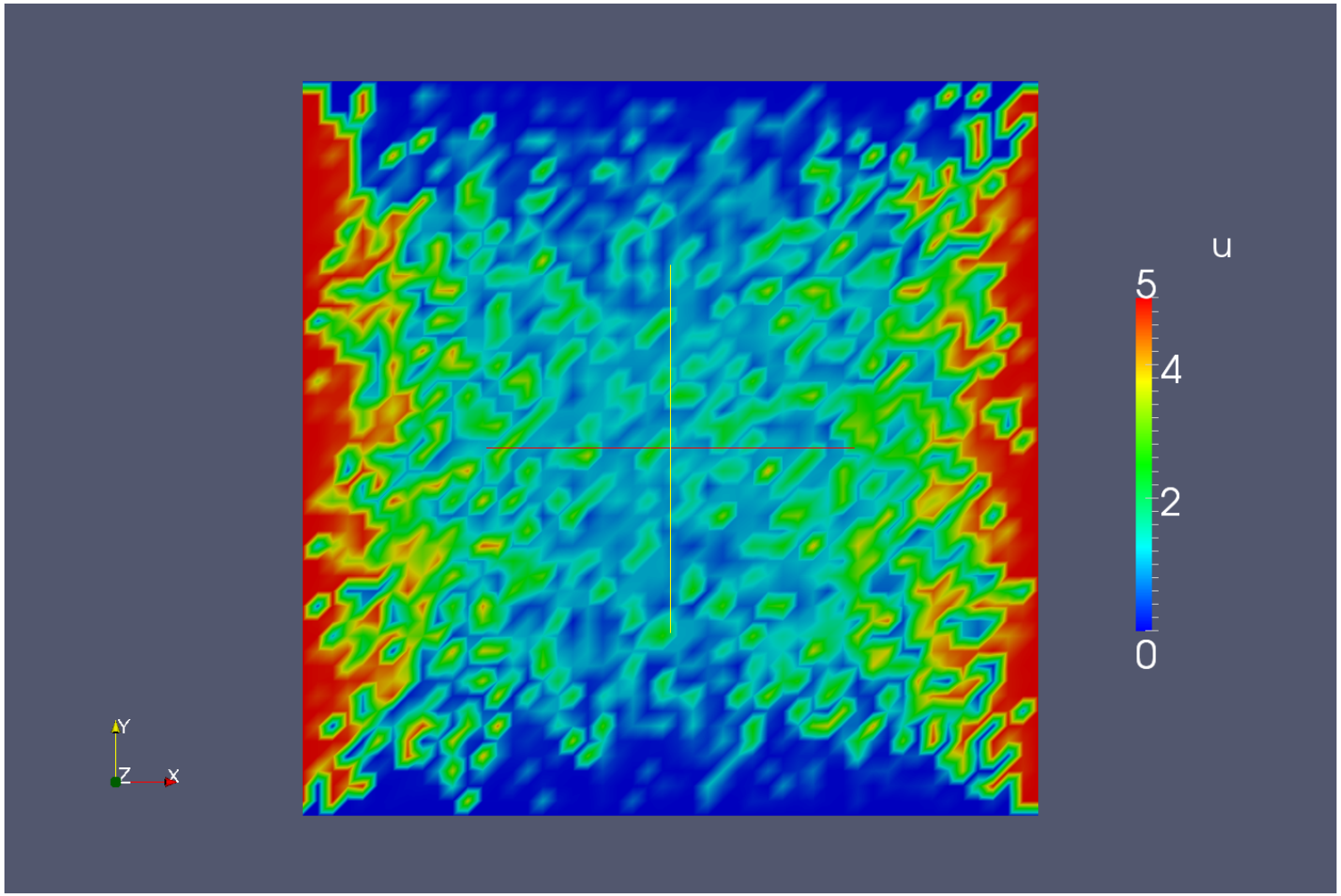



Figure : $2.5 \times 10^3$ total histories.

OAK RIDGE National Laboratory

# Solving the Heat Equation: Forward Method

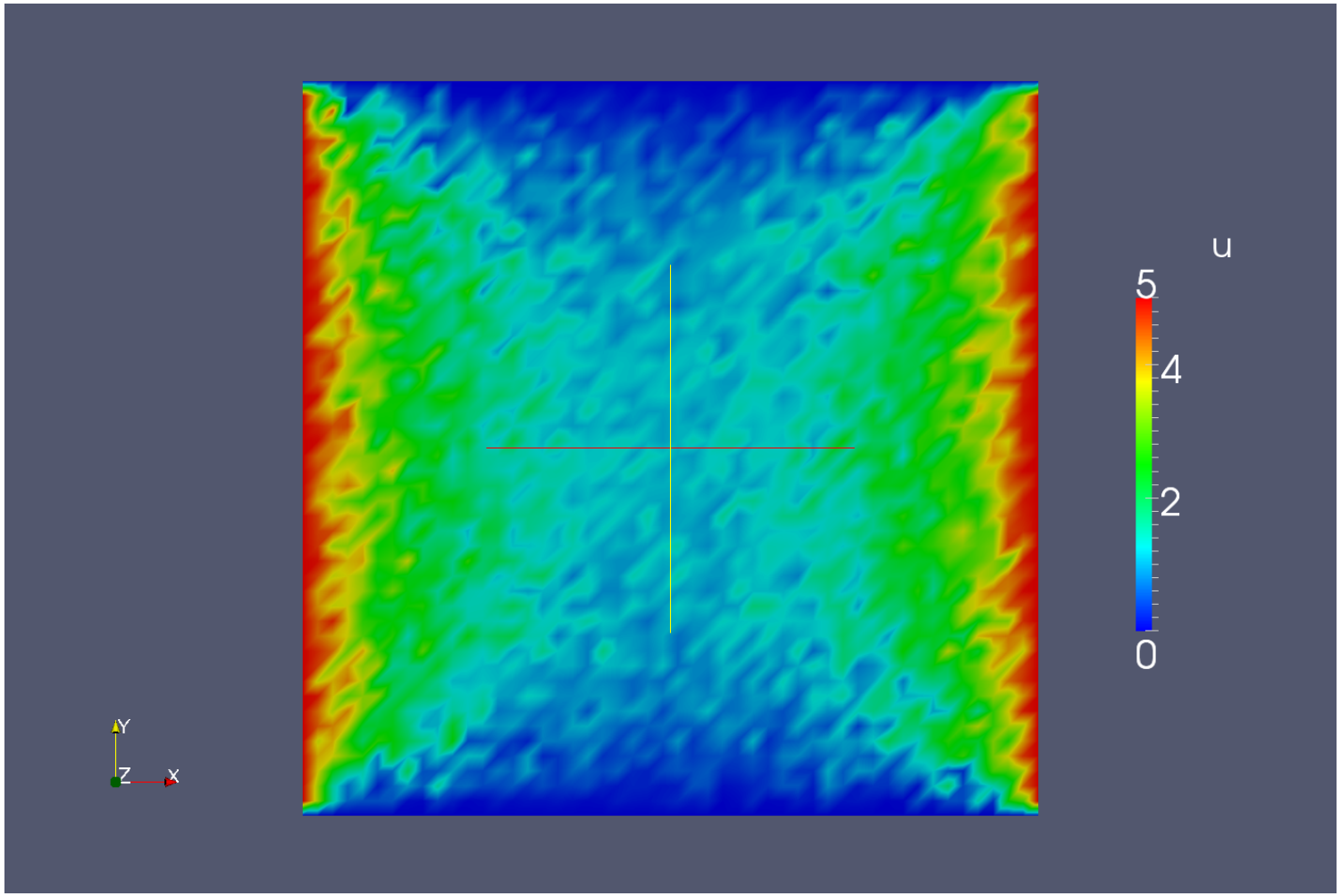



Figure : $2.5 \times 10^4$ total histories.

OAK RIDGE National Laboratory

# Solving the Heat Equation: Forward Method

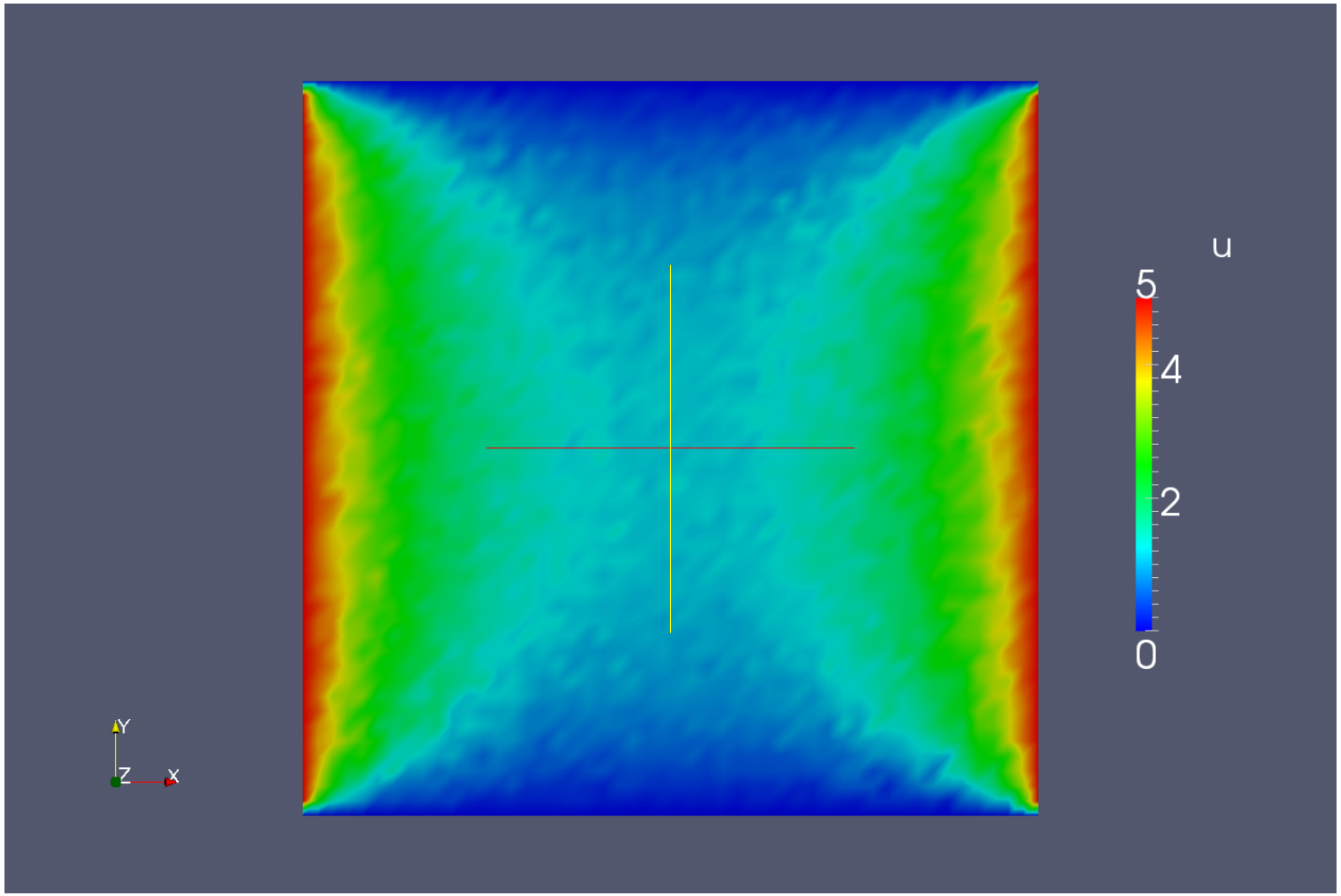



Figure : $2.5 \times 10^5$ total histories.

OAK RIDGE National Laboratory

# Solving the Heat Equation: Forward Method

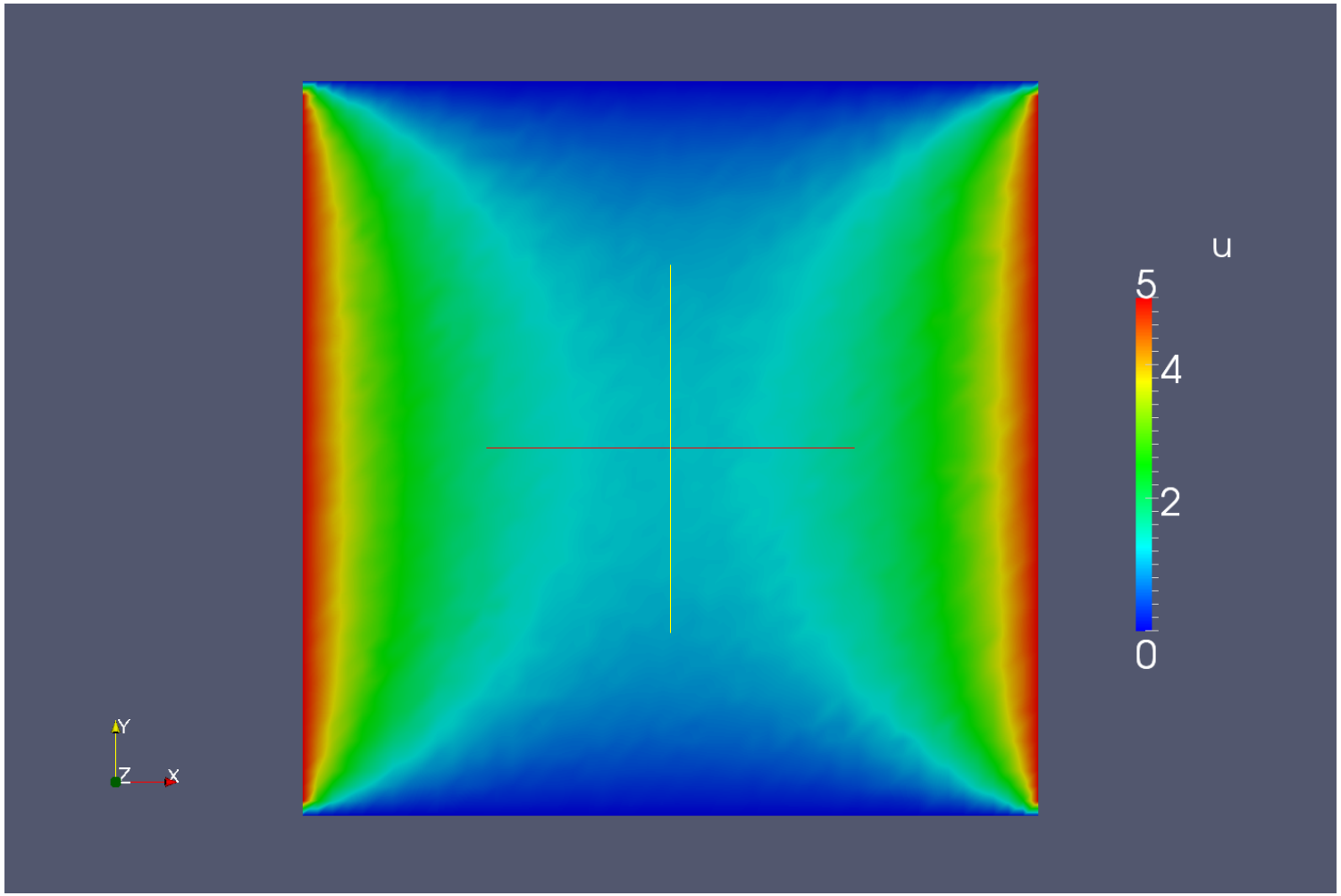



Figure : $2.5 \times 10^6$ total histories.

OAK RIDGE National Laboratory

# Domain Decomposed Monte Carlo

- Each parallel process owns a piece of the domain (linear system)
- Random walks must be transported between adjacent domains through parallel communication
- Domain decomposition determined by the input system
- Load balancing not yet addressed

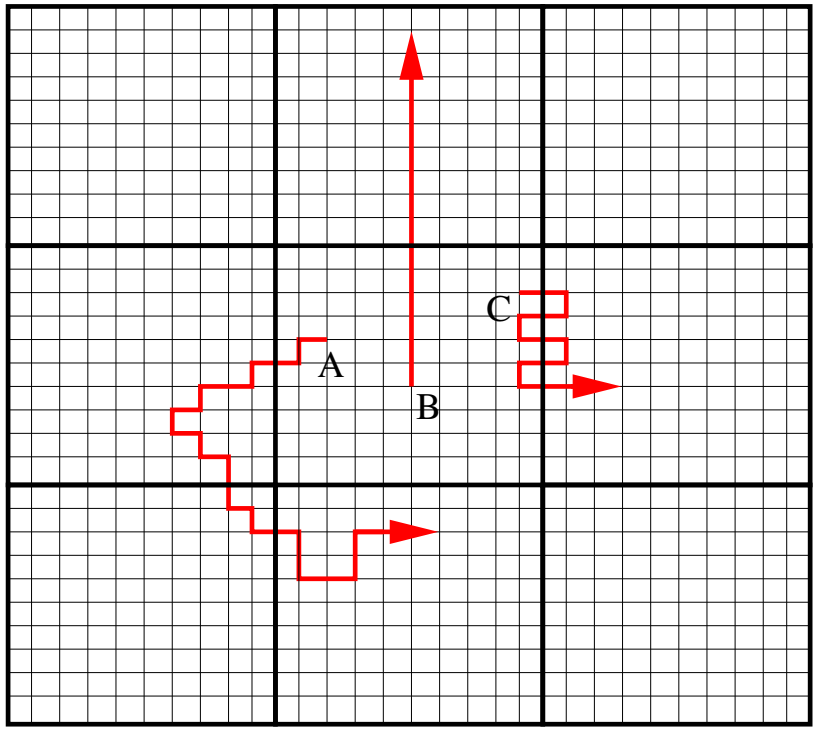
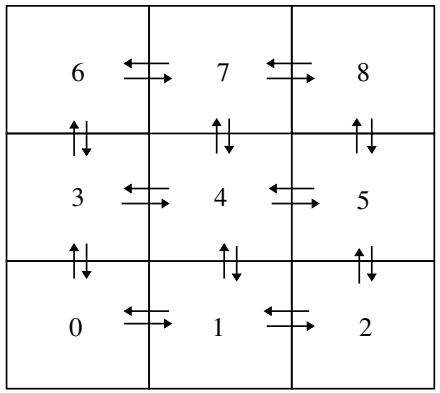


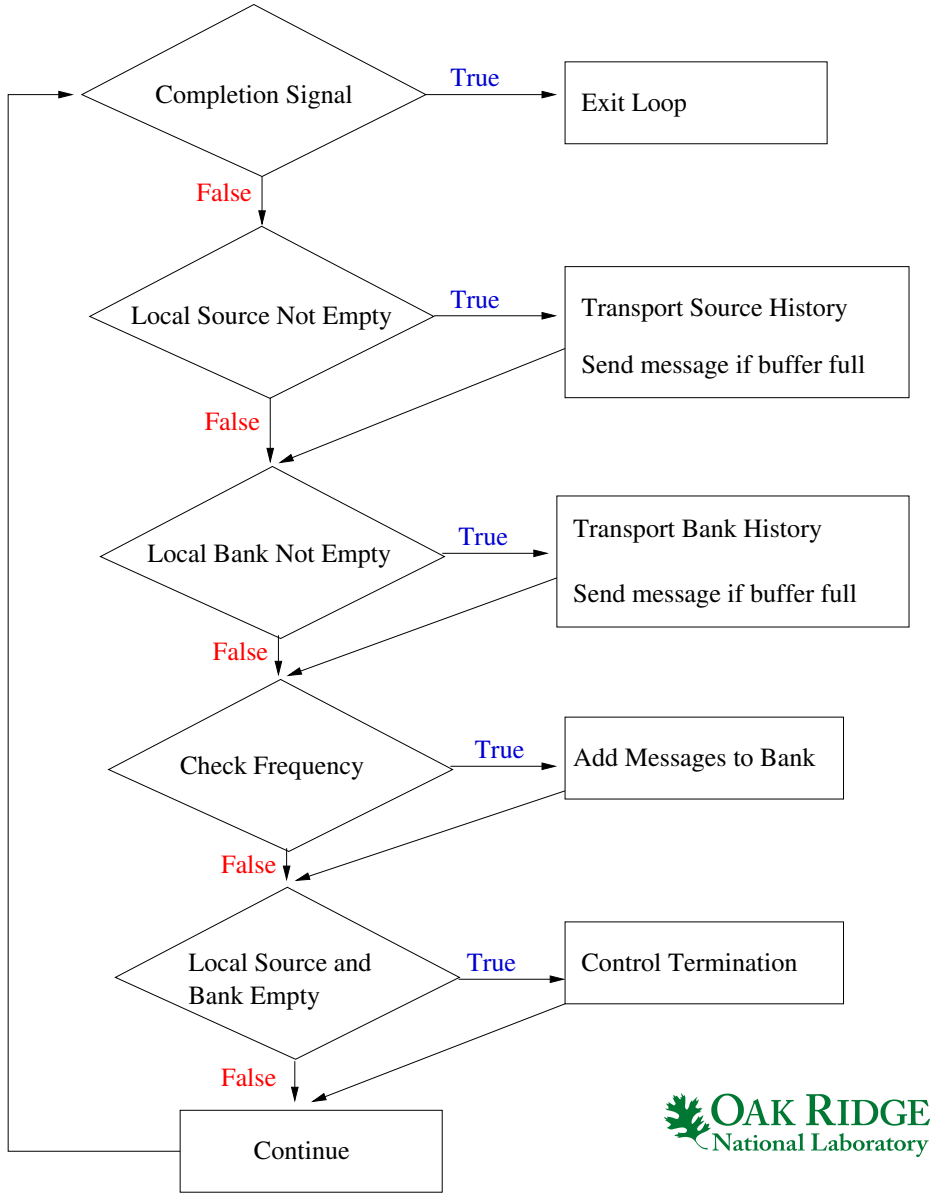

Figure : Domain decomposition example illustrating how domain-to-domain transport creates communication costs.

OAK RIDGE National Laboratory

# Asynchronous Monte Carlo Transport Kernel

- General extension of the Milagro algorithm (LANL)
- Asynchronous nearest neighbor communication of histories
- System-tunable communication parameters of buffer size and check frequency (performance impact)
- Need an asynchronous strategy for exiting the transport loop without a collective (running sum)




OAK RIDGE National Laboratory

# Exiting the Transport Loop without Collectives

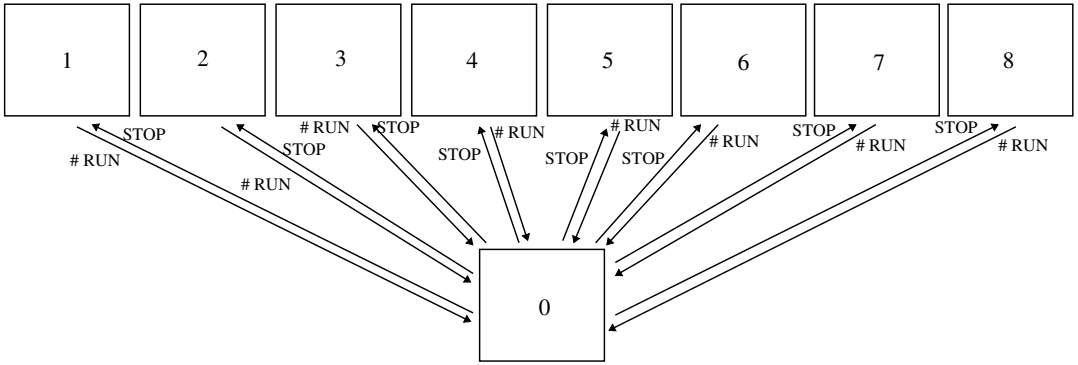



Figure : Master/Slave

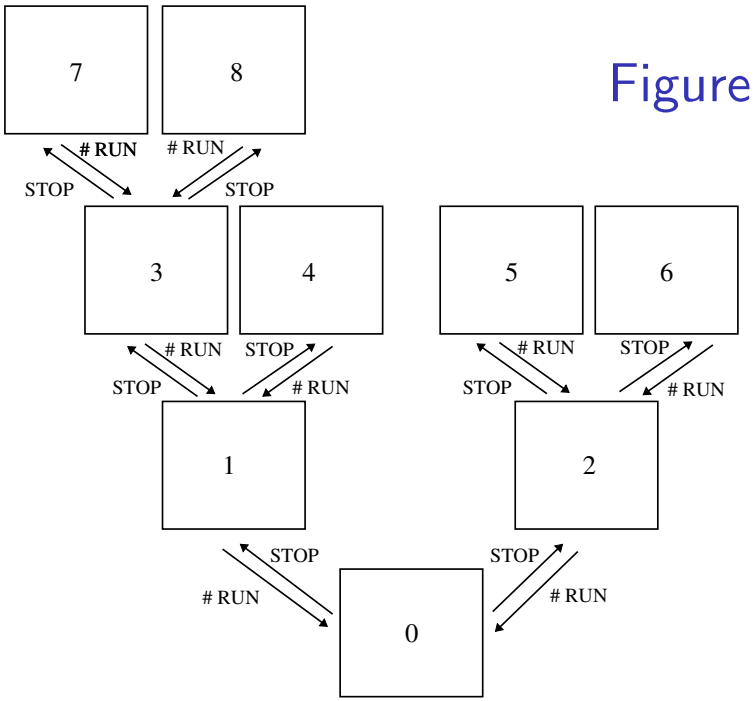



Figure : Binary Tree

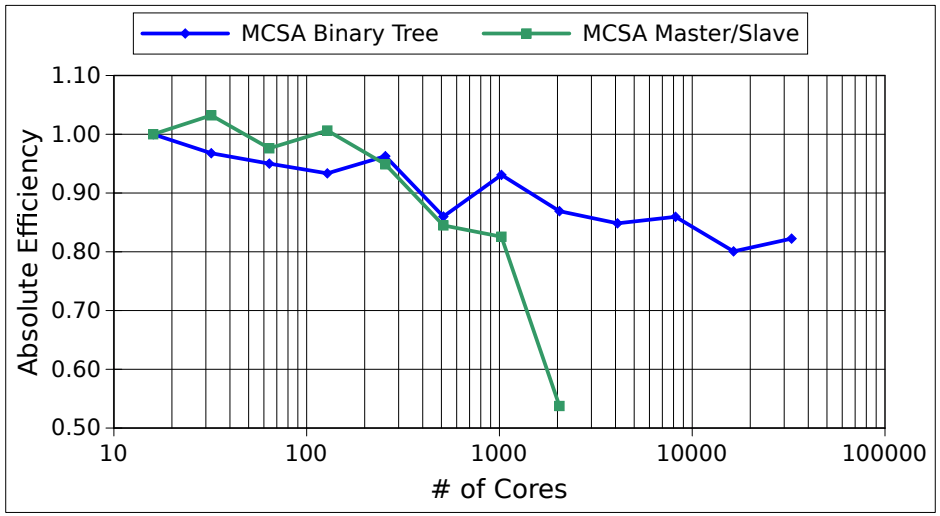



Figure : Weak scaling absolute efficiency

OAK RIDGE National Laboratory

# Replication

Different batches of Monte Carlo samples can be combined in summation via superposition if they have different random number streams. For two different batches:

$$\mathbf{M_{MC}x} = \frac{1}{2}(\mathbf{M_1} + \mathbf{M_2})\mathbf{x}$$

Consider each of these batches independent *subsets* of a Monte Carlo operator where now the operator can be formed as a general additive decomposition of $N_S$ subsets:

$$\mathbf{M_{MC}} = \frac{1}{N_S}\sum_{n=1}^{N_s}\mathbf{M_n}$$

We replicate the linear problem and form each subset on a different group of parallel processes. Applying the subsets to a vector requires an AllReduce to form the sum. Each subset is domain decomposed.

# Parallel Test - Simplified $P_N$ ($SP_N$) Assembly Problem

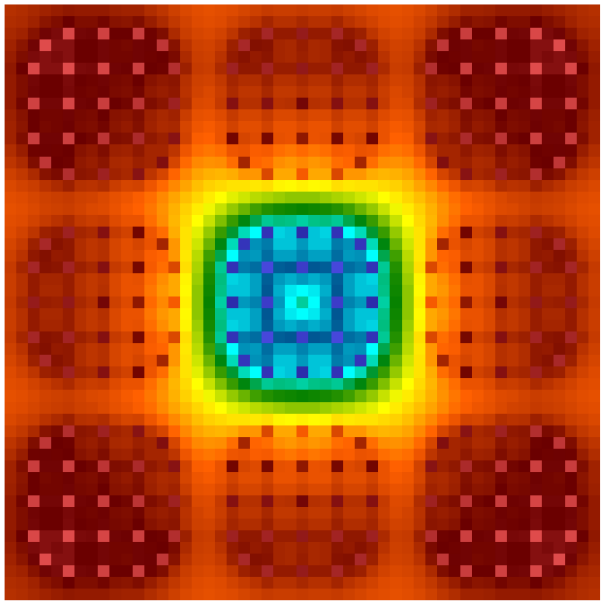


Figure : $SP_N$ solution example

The ($SP_N$) equations are an approximation to the Boltzmann neutron transport equation used to simulate nuclear reactors

$$-\nabla\cdot\mathbb{D}_n\nabla\mathbb{U}_n+\sum_{m=1}^{4}\mathbb{A}_{nm}\mathbb{U}_m=\frac{1}{k}\sum_{m=1}^{4}\mathbb{F}_{nm}\mathbb{U}_m$$

Scaling problem – $1\times1$ to $17\times17$ array of fuel assemblies with 289 pins each resolved by a $2\times2$ spatial mesh and 200 axial zones

- 7 energy groups, 1 angular moment, 1.6M to 273.5M degrees of freedom
- 64 to 10,800 computational cores via domain decomposition
- We are usually interested in solving generalized eigenvalue problem - we use the operator from these problems to test the kernel scaling

OAK RIDGE National Laboratory

# Monte Carlo Communication Parameters

| | | Message Check Frequency | | | |
|---|---|---|---|---|---|
| | | **128** | **256** | **512** | **1 024** |
| Message Buffer Size | **256** | 1.054 | 1.061 | 1.076 | 1.076 |
| | **512** | 1.103 | 1.146 | 1.211 | 1.270 |
| | **1 024** | 1.062 | 1.088 | 1.133 | 1.176 |
| | **2 048** | 1.030 | 1.042 | 1.072 | 1.107 |
| | **4 096** | 1.010 | 1.012 | 1.025 | 1.050 |
| | **8 192** | 1.001 | 1.000 | 1.008 | 1.018 |
| | **16 384** | 1.017 | 1.003 | 1.010 | 1.009 |

- OLCF Eos: 736-node Cray XC30, Intel Xeon E5-2670, 11,776 cores, 47 TB memory, Cray Aries interconnect
- 64 cores, 1.6M DOFs, history length of 15, 3 histories per DOF
- 27% decrease in runtime observed for bad parameter choices
- Worth the time to do this parameter study when running on new hardware

OAK RIDGE National Laboratory

# Domain Decomposition Scaling

| Cores | DOFs | DOFs/Core | Time Min (s) | Time Max (s) | Time Ave (s) | Efficiency |
|---|---|---|---|---|---|---|
| 256 | 273 509 600 | 1 068 397 | 515.51 | 515.52 | 515.52 | 1.00 |
| 1 024 | 273 509 600 | 267 099 | 122.76 | 122.77 | 122.76 | 1.05 |
| 4 096 | 273 509 600 | 66 775 | 27.96 | 27.97 | 27.96 | 1.15 |
| 7 744 | 273 509 600 | 35 319 | 17.72 | 17.72 | 17.72 | 0.96 |
| 10 816 | 273 509 600 | 25 288 | 13.72 | 13.72 | 13.72 | 0.89 |

Table : Strong Scaling

| Cores | DOFs | DOFs/Core | Time Min (s) | Time Max (s) | Time Ave (s) | Efficiency |
|---|---|---|---|---|---|---|
| 64 | 1 618 400 | 25 288 | 12.65 | 12.65 | 12.65 | 1.00 |
| 256 | 6 473 600 | 25 288 | 12.86 | 12.86 | 12.86 | 0.98 |
| 1 024 | 25 894 400 | 25 288 | 14.59 | 14.59 | 14.59 | 0.87 |
| 4 096 | 103 577 600 | 25 288 | 14.25 | 14.25 | 14.25 | 0.89 |
| 7 744 | 195 826 400 | 25 288 | 14.75 | 14.78 | 14.75 | 0.86 |
| 10 816 | 273 509 600 | 25 288 | 13.72 | 13.72 | 13.72 | 0.92 |

Table : Weak Scaling

- Titan Cray XK7 was used with no GPUs - used Eos parameters
- Results are consistent with transport literature
- History length of 15 and 3 histories per DOF
- Consider these effective operator apply times of 15 realizations of $\mathbf{H}$
- MC dominates run times in solver applications

OAK RIDGE National Laboratory

# Replication Scaling

| Subsets | Cores | DOFs | Time Min (s) | Time Max (s) | Time Ave (s) | Efficiency |
|---|---|---|---|---|---|---|
| 1 | 256 | 6 473 600 | 12.65 | 12.65 | 12.65 | 1.00 |
| 2 | 512 | 6 473 600 | 6.62 | 6.80 | 6.72 | 0.93 |
| 3 | 768 | 6 473 600 | 4.50 | 4.73 | 4.66 | 0.89 |
| 4 | 1 024 | 6 473 600 | 3.62 | 3.81 | 3.71 | 0.83 |

Table : Fixed algorithmic scaling

- Timings are Monte Carlo time plus AllReduce time
- 256 cores per subset with domain decomposition
- $1/N_S$ samples per subset gives flat algorithmic performance
- Two performance issues
  - Dividing up samples creates work starvation
  - AllReduce buffer size is constant regardless of work starvation
- Research is needed for a more performant subset combination

OAK RIDGE National Laboratory

# Current Work - Vectorization and Threading

- We have implemented a Monte Carlo kernel using the Kokkos threading model (Trilinos)
  - The kernel supports multi-threaded CPU, GPU, and Xeon Phi architectures with a single implementation
- Vectorization an area of active research with a focus on heterogeneous architectures (Titan and Summit)
  - Currently investigating event-based algorithms to enable vectorization
  - Event-based algorithm concepts are based on vectorized Monte Carlo algorithms from particle transport
- We are exploring an additive-Schwarz formulation to eliminate parallel communication in the Monte Carlo kernel
- Other threading models will be considered (e.g. HPX)

OAK RIDGE National Laboratory

# Conclusions and Future Work

- Monte Carlo methods offer great potential for both resilient and highly parallel solvers
  - A fully asynchronous algorithm provides scalability without collectives
  - Replication potentially offers resiliency with some overhead
- Extending methods to broader problem areas is a significant algorithmic challenge and an attractive area for continued research
  - Explicit preconditioners are required for all problems
- Performance modeling and resiliency simulations this FY
  - Fault injection studies using the xSim simulator

OAK RIDGE National Laboratory