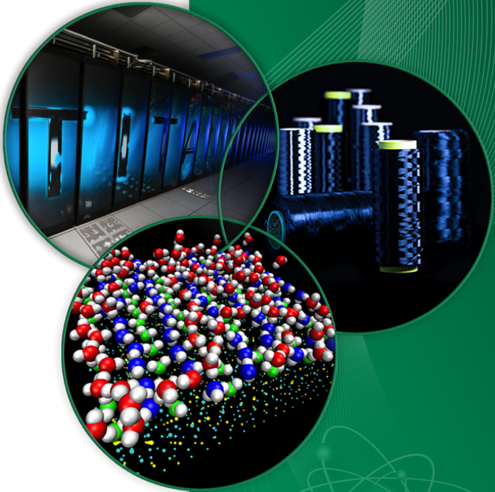
Parallel Algorithms for Monte Carlo Linear Solvers



Stuart Slattery
Steven Hamilton
Tom Evans

Oak Ridge National Laboratory

March 17, 2015

Acknowledgements

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.

Motivation

- As we move towards exascale computing, the rate of errors is expected to increase dramatically
  - The probability that a compute node will fail during the course of a large scale calculation may be near 1
- Algorithms need to not only have increased concurrency/scalability but have the ability to recover from hardware faults
  - Lightweight machines
  - Heterogeneous machines
  - Both characterized by low power and high concurrency

Towards Exascale Concurrency and Resiliency

- Two basic strategies:

  1. State 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 a component of the tally variance
     - Hard failures potentially mitigated by replication
     - Concurrency enabled by several levels of parallelism

Outline

- Monte Carlo Linear Solvers
- Domain Decomposition and Replication
- Scaling Studies
- Matrix-Free and Stochastic Approximate Inverse Algorithms

Monte Carlo Methods

Monte Carlo for Linear Systems

- Suppose we want to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$
- 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

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

- Define a sequence of state transitions

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

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 $\mathbf{x} \approx \mathbf{M_{MC}}\mathbf{b}$

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

Solving the Heat Equation: Forward Method

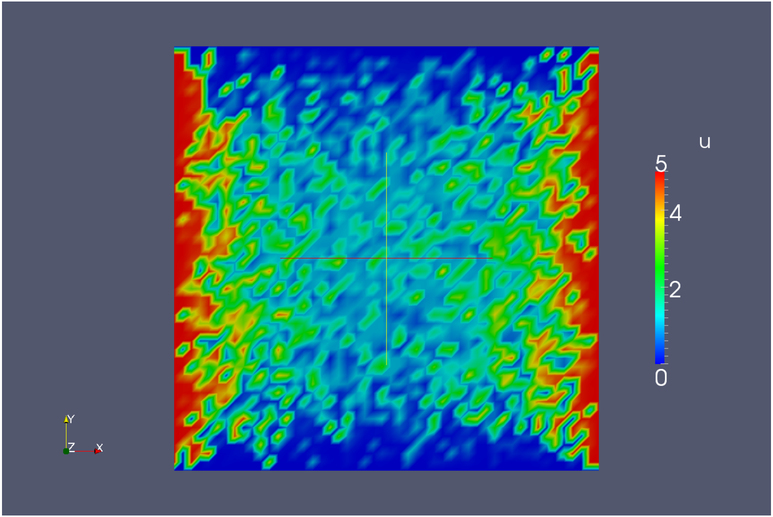


Figure : **Forward solution.** 2.5×10^3 *total histories.*

OAK RIDGE National Laboratory

Solving the Heat Equation: Forward Method

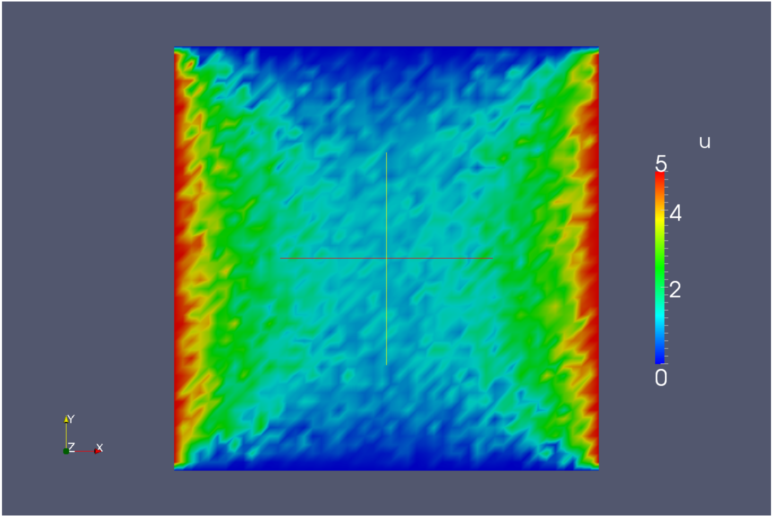


Figure : **Forward solution.** 2.5×10^4 *total histories.*

Solving the Heat Equation: Forward Method

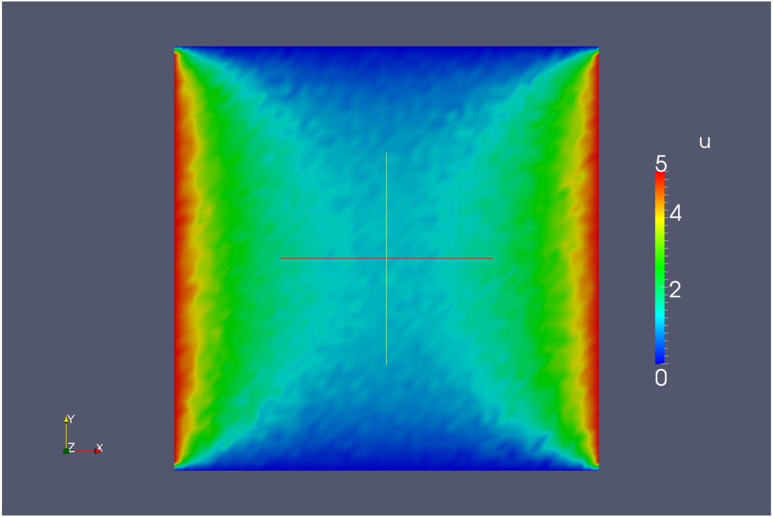


Figure : **Forward solution.** 2.5×10^5 *total histories.*

Oak Ridge National Laboratory

Solving the Heat Equation: Forward Method

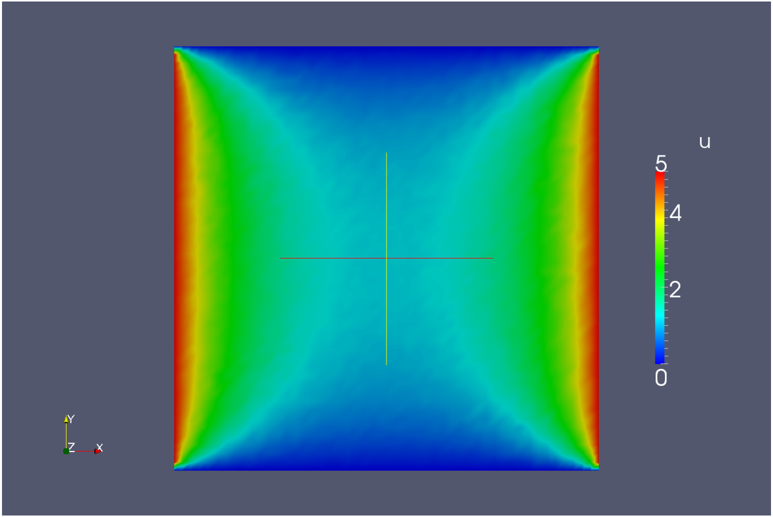


Figure : **Forward solution.** 2.5×10^6 *total histories.*

OAK RIDGE National Laboratory

Monte Carlo Synthetic Acceleration

- Devised by Evans and Mosher in the 2000's as an acceleration scheme for radiation diffusion problems (LANL)
- Can be abstracted as a general linear solver with Monte Carlo as a preconditioner
- Combine with Richardson iteration as a "smoother" in between Monte Carlo steps:

$$
\begin{aligned}
\mathbf{r}^k &= \mathbf{b} - \mathbf{A}\mathbf{x}^k \\
\mathbf{x}^{k+1/2} &= \mathbf{x}^k + \mathbf{r}^k \\
\mathbf{r}^{k+1/2} &= \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} \\
\mathbf{x}^{k+1} &= \mathbf{x}^{k+1/2} + \mathbf{M_{MC}}\mathbf{r}^{k+1/2}
\end{aligned}
$$

Domain Decomposition and Replication

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

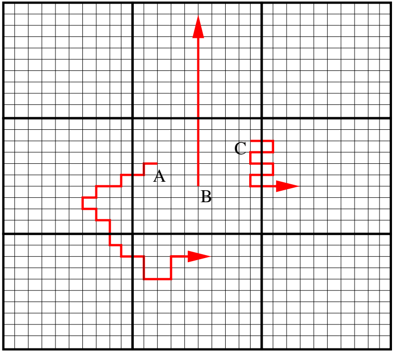
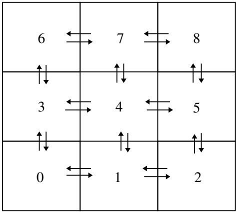
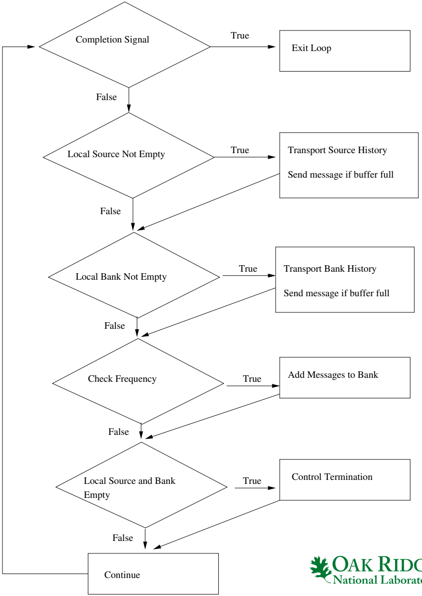


Figure : Domain decomposition example illustrating how domain-to-domain transport creates communication costs.

Asynchronous Monte Carlo Transport Kernel

- General extension of the Milagro algorithm (LANL)
- Asynchronous nearest neighbor communication of histories
- System-tunable communciation 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

Stopping the Asynchronous Kernel

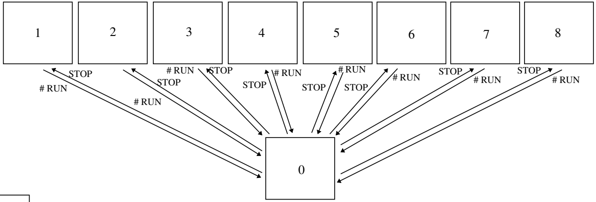


Figure : Master/Slave

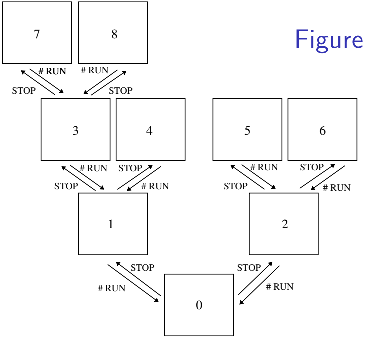


Figure : Binary Tree

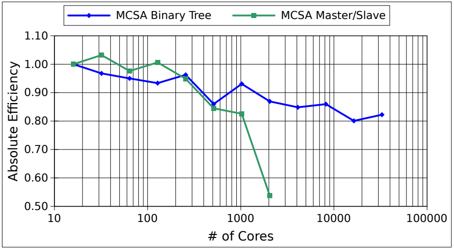


Figure : Weak scaling absolute efficiency

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.

Scaling Studies

Parallel Test Application – Nuclear Reactor Analysis

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

$$\hat{\Omega} \cdot \vec{\nabla} \psi(\vec{r}, \hat{\Omega}, E) + \sigma(\vec{r}, E) \psi(\vec{r}, \hat{\Omega}, E) = \iint \sigma_s(\vec{r}, E' \rightarrow E, \hat{\Omega}' \cdot \hat{\Omega}) \psi(\vec{r}, \hat{\Omega}', E') d\Omega' dE' + q(\vec{r}, \hat{\Omega}, E) \quad (1)$$

$$-\nabla \cdot \left[\frac{n}{2n+1} \frac{1}{\Sigma_{n-1}} \nabla \Big(\frac{n-1}{2n-1} \phi_{n-2} + \frac{n}{2n-1} \phi_n \Big) + \frac{n+1}{2n+1} \frac{1}{\Sigma_{n+1}} \nabla \Big(\frac{n+1}{2n+3} \phi_n + \frac{n+2}{2n+3} \phi_{n+2} \Big) \right] + \Sigma_n \phi_n = q \delta_{n0} \qquad n = 0, 2, 4, \cdots, N \quad (2)$$

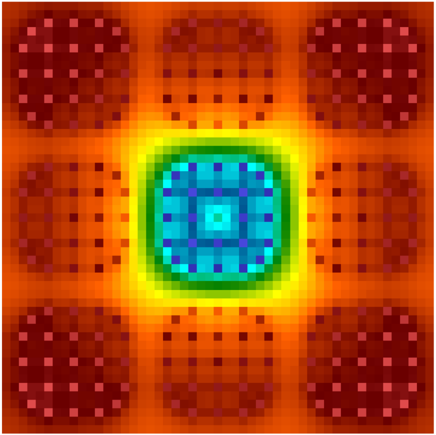
$$-\nabla \cdot \mathbb{D}_n \nabla \mathbb{U}_n + \sum_{m=1}^{4} \mathbb{A}_{nm} \mathbb{U}_m = \frac{1}{k} \sum_{m=1}^{4} \mathbb{F}_{nm} \mathbb{U}_m \qquad n = 1, 2, 3, 4 \quad (3)$$

OAK RIDGE National Laboratory

SP_N Assembly Problem

Test problem – 3×3 array of fuel assemblies with control rod in center location (Profugus)

- 23 energy groups, 2 angular moments, 25M degrees of freedom
- 1,000 computational cores via domain decomposition
- We are interested in solving generalized eigenvalue problem, for this study we use Arnoldi as the eigensolver and compare different methods for solving linear systems



Monte Carlo Communication Parameters

Method	Total Linear Iteration	Setup Time (s)	Solve Time (s)
GMRES-ILUT	1675	0.7	18.4
GMRES-AMG	626	0.7	46.0
GMRES-MGE	498	1.5	33.7
Richardson-AINV	5208	20.6	52.0
MCSA-AINV	1268	25.5	46.6

- ILUT preconditioning is winner here, but known to have issues with parallel scaling on large core counts
- Solve times for MCSA are competitive, but setup times are very large due to construction of sparse approximate inverse

Monte Carlo Strong Scaling

Monte Carlo Weak Scaling

Matrix-Free and Stochastic Approximate Inverse Algorithms

Alternative Parallelism – Additive Schwarz

- Instead of performing Monte Carlo on full problem, another possibility is to apply Monte Carlo as an additive Schwarz approach
- Decompose problem into (possibly overlapping) domains
- Perform Monte Carlo on individual subdomains
  - No communication costs in Monte Carlo problem!
- With domain decomposed Monte Carlo, iteration counts are effectively independent of the number of processors
- In an additive Schwarz approach, the preconditioner will become less effective as processor counts grow – algorithmic scalability may be an issue
- **Replication for resiliency and performance**

More Parallelism – Threading

- Within a Monte Carlo solve, every history is independent of other histories – great potential for highly concurrent hardware (GPU, Xeon Phi)
- Polynomial formulation enables a priori determination of operation counts per thread
- Memory locality an issue due to random access via random walks (block formulation?)
- Early experiments using the Trilinos Kokkos library show promising performance for multi-core CPUs
- Team members recently took part in OLCF “Hackathon” in late October to begin implementing computation kernels in OpenACC to allow for GPU capability on Titan with early results indicating **1.3-9.4x** speedup for MCSA (largely dependent on random number generation)

Conclusions

- Monte Carlo methods offer great potential for both resilient and highly parallel solvers
- For certain classes of problems, Monte Carlo methods can be competitive with leading modern solvers
- Extending methods to broader problem areas is significant challenge and an attractive area for continued research
- Performance modeling and resiliency simulations this FY