LA-UR-16-25273

Abstract

Neutral particle transport, which includes the transport of neutrons, photons, and neutrinos, is an important phenomenon in determining the behavior of nuclear reactor cores and astrophysical processes. One deterministic method of simulating the transport of neutral particles is discrete-ordinates, which involves solving the radiative transfer equation (RTE) for a finite number of discrete solid angles. Parallelization of discrete-ordinates methods for use on supercomputing clusters can be difficult, especially for 3D, unstructured, tetrahedral meshes. In this project, the weak-scaling of various parallelization schemes for 3D discrete-ordinates methods were compared using Tycho2, a discontinuous Galerkin linear elements code developed at Los Alamos National Lab. Parallelization was done in conjunction with three algorithms: a parallel sweep in the direction of transport, Parallel Block Jacobi, and a Schur-Complement parallel Krylov subspace method. All three algorithms were run on the Cray XC30 system, Edison, from NERSC for various numbers of energy groups, numbers of solid angles, and domain sizes. MPI and OpenMP were used to implement the parallel algorithms.

Background

The behavior of neutral particles, such as neutrons, photons, and neutrinos, is important in the analysis of nuclear reactors and astrophysical phenomena. Kinetic transport of neutral particles is governed by the Radiative Transfer Equation

$$\Omega \cdot \nabla_x \Psi(x,\Omega,E) + \sigma_t \Psi(x,\Omega,E) = \frac{\sigma_s}{4\pi}\int_{\mathbb{S}^2} \Psi(x,\Omega',E)d\Omega' + Q(x,\Omega,E) \tag{1}$$

which balances material for a given $x \in D \subset \mathbb{R}^3$, $\Omega \in \mathbb{S}^2$, $E \in \mathbb{R}^{\geq 0}$. The function Ψ is the unknown, Q is a known source, and σ_t and σ_s are the total and scattering cross-sections (with $\sigma_t > \sigma_s$). Though E is coupled, it is typical to solve a simpler problem and discretize E into energy groups. Furthermore, discrete ordinates (S_N) methods can be used to discretize angle based on the quadrature of the sphere. Using discretization in E and Chebyshev-Legendre quadrature on the sphere for discrete ordinates, Equation (1) becomes

$$\Omega_q \cdot \nabla_x \Psi_{qg}(x) + \sigma_t \Psi_{qg}(x) = \frac{\sigma_s}{4\pi}\sum_{q'=0}^{2N^2-1} w_{q'}\Psi_{q'g}(x) + Q_{qg}(x) \tag{2}$$

where g is the index of the energy groups, q is the quadrature index, and the nodes and weights of the quadrature are Ω_q and w_q.

To deal with the integral on the RHS of equation (2), the data from the integral can be lagged. This is called source iteration

$$\Omega \cdot \nabla_x \Psi^{k+1} + \sigma_t \Psi^{k+1} = \frac{\sigma_s}{4\pi}\int_{\mathbb{S}^2} \Psi^k d\Omega' + Q. \tag{3}$$

Since source iteration can be used to determine the RHS of equation (2), the scattering term can be absorbed in the source term to form a simplified equation

$$\Omega_q \cdot \nabla_x \Psi_{qg}(x) + \sigma_t \Psi_{qg}(x) = Q_{qg}(x) \tag{4}$$

Linear elements are used to discretize spatial variables. An example quadrature of the sphere is depicted in Figure 1.

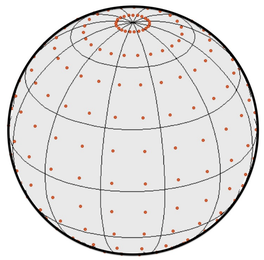


Figure 1: Chebyshev-Legendre quadrature

Tycho2

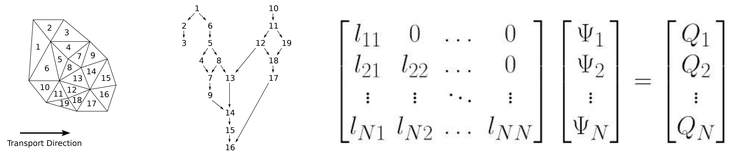
Tycho2 is an open-source, discontinuous Galerkin linear elements code that solves the RTE using discrete ordinates on 3D, unstructured tetrahedral meshes. It can solve the RTE in parallel using several different algorithms. The code is available at https://github.com/losalamos/tycho2.

Acknowledgements

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Additionally, the author would like to acknowledge the help of the mentors involved in the 2016 Parallel Computing Summer Research Internship, especially Charles K. Garrett.

Algorithms



$$\begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{N1} & l_{N2} & \dots & l_{NN} \end{bmatrix} \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_N \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_N \end{bmatrix}$$

Figure 2: (Left) 2D unstructured mesh and chosen transport direction for sweep method. (Center) Directed acyclic graph produced as a result of the mesh. (Right) Performing the sweep is equivalent to inverting a sparse lower triangular matrix. Note that the figure depicts a 2D example, but Tycho2 uses a 3D tetrahedral mesh.

Sweep:

As depicted in Figure 2, the method of sweeps involves selecting a transport direction and generating a list of priorities for computing each of the cells. This is the same as forming a directed acyclic graph (DAG) of cell dependencies, or inverting a sparse lower triangular matrix.

Optimal scheduling for a variable number of processors and a general DAG is NP-Complete [1]. However, Tycho2 supports several known heuristics for scheduling [2]. For the purposes of this poster, the heuristic used was Breadth First Descendant Seeking (see [3]).

Parallel Block Jacobi:

With the Parallel Block Jacobi (PBJ) algorithm, each MPI rank has a guess for its local boundary data and sweeps are independently performed on each partition of the mesh. Then, boundary values are communicated between all of the ranks and sweeps are repeated. This process continues until Ψ has converged to a given tolerance. Parallel Block Jacobi is equivalent to performing a fixed-point iteration on a linear system with a diagonal block matrix.

$$\begin{bmatrix} A_{11} & 0 & \dots & 0 \\ 0 & A_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{NN} \end{bmatrix} \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_N \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_N \end{bmatrix}$$

Figure 3: Block Triangular Matrix

Schur-Complement Krylov:

With the Schur-Complement Krylov algorithm, the mesh is partitioned into regions (similar to the PBJ partitioning) and a boundary partition. Each MPI rank iteratively solves the local boundary of one of the partitions using a Krylov solver. Communication occurs between each rank until the global boundary partition has converged to a given tolerance. Then, each rank performs a sweep in parallel to solve the internal values for each partition.

$$\begin{bmatrix} A_{11} & 0 & \dots & 0 & A_{1\Gamma} \\ 0 & A_{22} & \dots & 0 & A_{2\Gamma} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & A_{NN} & A_{N\Gamma} \\ A_{\Gamma 1} & A_{\Gamma 2} & \dots & A_{\Gamma N} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_N \\ \Psi_\Gamma \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_N \\ Q_\Gamma \end{bmatrix} \implies \begin{bmatrix} A_{11} & 0 & \dots & 0 & A_{1\Gamma} \\ 0 & A_{22} & \dots & 0 & A_{2\Gamma} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & A_{NN} & A_{N\Gamma} \\ 0 & 0 & \dots & 0 & A_{\Gamma\Gamma} - \sum_i A_{\Gamma i}A_{ii}^{-1}A_{i\Gamma} \end{bmatrix} \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \vdots \\ \Psi_N \\ \Psi_\Gamma \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_N \\ Q_\Gamma - \sum_i A_{\Gamma i}A_{ii}^{-1}Q_i \end{bmatrix}$$

Figure 4: System with boundary partition and corresponding Schur-Complement

The Schur-Complement Krylov method is equivalent to rearranging the lower triangular matrix used in the sweep to include a boundary region (Γ), taking its Schur-Complement, solving the boundary, and then solving the other blocks in parallel. Figure 4 depicts the rearrangement of a lower triangular matrix into the appropriate form of the algorithm as well as the Schur-Complement of said matrix.

Therefore, to solve the system, the equation

$$(A_{\Gamma\Gamma} - \sum_i A_{\Gamma i}A_{ii}^{-1}A_{i\Gamma})\Psi_\Gamma = Q_\Gamma - \sum_i A_{\Gamma i}A_{ii}^{-1}Q_i \tag{5}$$

can be solved first using a Krylov subspace method such as GMRES, allowing the remaining equations

$$A_{ii}\Psi_i = Q_i - A_{\Gamma N}\Psi_\Gamma \tag{6}$$

to be solved in parallel to determine the remainder of the system. In Tycho2, the PETSC library [4] was used to implement matrix-free operators that solve equations (7) and (8). It can be shown that $A_{\Gamma\Gamma}$ is the identity, $A_{i\Gamma}$ corresponds to removing all values from the input vector except the boundary values corresponding to the ith processor, A_{ii}^{-1} is equivalent to performing a sweep on partition i, and $A_{\Gamma i}$ is the same as communicating the outgoing boundary values to corresponding incoming boundaries.

Implementation and Results

To investigate the weak-scaling of the various algorithms implemented in Tycho2, cube-shaped meshes were divided into various numbers of cells. The cells were then partitioned into columns such that the size of the columns remained constant at 1032 cells/column. Scaling was performed using Edison, a Cray XC30 cluster at the National Energy Research Scientific Computing Center. Each node on the supercomputer has 2 sockets, each populated with a 12-core Intel "Ivy Bridge" processor with hyper-threading. One MPI rank was used per core, and the number of cores was varied from 4 to 2048. Figure 5 gives the results for weak scaling of the Sweep, PBJ, and Schur-Complement Krylov algorithms.

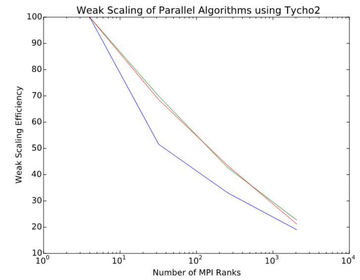


Figure 5: Weak Scaling Efficiency vs. Number of Partitions. Green is PBJ, blue is Schur-Complement Krylov, and red is the parallel sweep. The Schur-Complement Krylov algorithm approaches the scaling efficiency of the other algorithms at 2048 cores, and may surpass them for higher numbers of cores.

To investigate the strong scaling of the algorithms implemented in Tycho2, a mesh with 264192 tetrahedral elements was created. This mesh was then divided into 4, 16, 256, 1024, or 2048 partitions, and each of the three algorithms was used with a number of MPI ranks equal to the number of partitions to solve the RTE. The results of these experiments are shown in Figure 6.

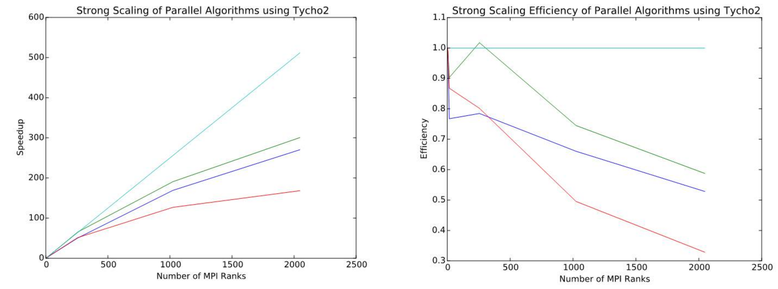


Figure 6: (Left) Speedup vs. Number of Partitions. (Right) Strong Scaling Efficiency vs. Number of Partitions. Cyan is ideal scaling, green is PBJ, blue is Schur-Complement Krylov, and red is the parallel sweep. PBJ seems to exhibit the best strong scaling, followed by Schur-Complement, followed by the parallel sweep algorithm.

Conclusion

In conclusion, it appears that the Schur-Complement Krylov algorithm is comparable in weak-scaling efficiency to PBJ and parallel sweeps for 2048 cores. Further investigation must be undertaken to determine relative scaling efficiency for the various algorithms at higher core counts and for variations in parameters such as scattering.

References

[1] Convolbo, M. W., and Chou, J. (2016). Cost-aware DAG scheduling algorithms for minimizing execution cost on cloud resources. The Journal of Supercomputing, 72(3), 985-1012. doi:10.1007/s11227-016-1637-7.

[2] Garrett, Charles K., Tycho2, (2016), GitHub repository, https://github.com/losalamos/tycho2.

[3] Pautz, S. (2002). An Algorithm for Parallel Sn sweeps on Unstructured Meshes. NSE Nuclear Science and Engineering, 111-136. doi:10.13182/nse02-1.

[4] Balay, et al. (1997). Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. *Modern Software Tools in Scientific Computing*. Ed. E. Arge and A. M. Bruaset and H. P. Langtangen. 163-202. Birkhäuser Press.