Validation of Full-Domain Massively Parallel Transport Sweep Algorithms

W. Daryl Hawkins[1], Teresa S. Bailey[2], Marvin L. Adams[1], Peter N. Brown[2], Adam J. Kunen[2], Michael P. Adams[1], Timmie Smith[3], Nancy Amato[3], Lawrence Rauchwerger[3]

[1]*Department of Nuclear Engineering, Texas A&M University, College Station, TX 77843-3133*
[2]*Lawrence Livermore National Laboratory, Livermore CA, 94551*
[3]*Department of Computer Science and Computer Engineering, Texas A&M University, College Station, TX 77843-3112*

INTRODUCTION

Recent publications have predicted excellent scaling of sweep algorithms based on parallel performance models. These performance models have yet to be completely validated with computational results. In this paper, we compare two parallel performance models for sweep algorithms to computational performance using distinctly different discrete ordinates transport codes. We have run both codes beyond 1Million MPI ranks on Lawrence Livermore National Laboratory's (LLNL) BGQ machines (SEQUOIA and VULCAN) to validate our model predictions.

PARALLEL SWEEPS

The full-domain "sweep," in which all angular fluxes in a problem are calculated given previous-iterate values only for the volumetric source, forms the foundation for many iterative methods that have desirable properties [1]. The sweep algorithms considered here partition the spatial domain D into $P*N_{over}$ "cellsets" and distribute them across the machine. Here each of the P "ranks" owns a "subdomain," which is a collection of N_{over} cellsets, where N_{over} is an "overload factor." Cellset boundary dependencies are satisfied by boundary conditions or by solutions from up-wind neighboring cellsets. The sweep solution is complicated by the dependency of a given cellset on its upstream neighbors. Figure 1 illustrates how sweeps progress from cellset to cellset, showing that the cellsets on "diagonals" can be processed in parallel and

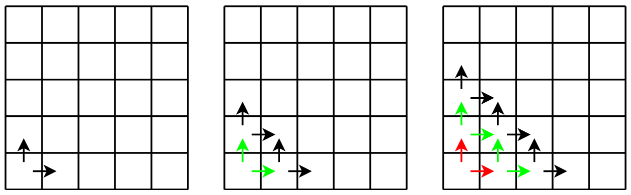
Fig. 1. Dependency relationships form "sweeps" across the spatial domain. Multiple directions starting from the same corner (black, green, red) can be pipelined.

that multiple angle sweeps can be pipelined. Each picture in the figure depicts a "stage" in the sweep algorithm. At the beginning of a sweep, some processors are idle. We call this idle time *pipe fill*. There is similar *pipe empty* idle time at the end. For good parallel performance it is important to reduce pipe fill and empty time.

The sweep for a given direction originates from one of the 2^d corners of the spatial domain and completes at the opposite corner (where d is the number of spatial dimensions). If we initiate sweeps from all corners simultaneously our pipe fill time is reduced, but we introduce collisions (shown as gray cellsets in Figure 2), which must be resolved. A collision occurs when a

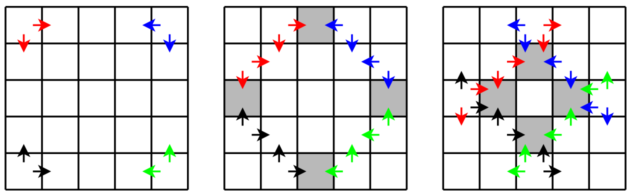
Fig. 2. Simultaneous sweeps from all corners reduce pipe fill, but create collisions (gray domains).

cellset has more than one angle that is ready for computation, and a decision must be made about which to compute first. Sweep algorithms are therefore composed of two main components: decomposition ("partitioning") and "scheduling" to define how the collisions are resolved. The goal of the scheduling algorithm is to resolve wave-front collisions in a way that minimizes total stage count for the full sweep.

The Koch-Baker Alcouffe (KBA) algorithm [2], the first widely adopted parallel sweep algorithm, introduced two ideas that reduce pipe fill and pipe empty time. KBA decomposes the spatial domain into P columnar subdomains, thus decomposing d−space into d−1 dimensions. Each columnar subdomain is then divided into cellsets, each an integral number of axial cell-planes. Because each rank's subdomain contains more than one cellset, we say the decomposition is overloaded. The KBA decomposition can also be described as decomposing the spatial domain into cellsets, then assigning all contiguous cellsets in a spatial column to a given MPI rank. The standard KBA algorithm begins the sweep at only one of the eight available corners, which eliminates wave-front collisions from the schedule.

Dorr and Still [3] and Clouse [4] applied a volumetric spatial domain decomposition, which decomposes the spatial domain in all dimensions into cube-shaped cellsets. Dorr and Still assign one cellset to each MPI rank, while Clouse allowed for cellset overloading. These decompositions allow sweeps to begin at all eight corners. Dorr and Still noted that there would be wave-front collisions, but they did not discuss strategies for resolving the collisions. Clouse developed a schedule that resolved collisions by using powers-of-2 valued decomposition in space and delays in the schedule. Collision-resolving schedules as outlined in [6,9,10] can be applied to both the KBA-style and the volumetric decompositions. Our results show that these scheduling rules lead to predictable, efficient scaling behavior for two very different parallel and solution strategies.

ARDRA

ARDRA [5] is a research code developed at LLNL to study parallel discrete ordinates transport. The code applies a general framework to domain decompose the angle, energy and spatial unknowns among available parallel ranks. Typically, problems run with ARDRA are only decomposed in space (volumetrically) and energy. Spatial overloading is not currently supported. ARDRA's default spatial discretization is diamond difference (finite difference), with one spatial unknown per mesh cell.

ARDRA solves the discrete form of the transport problem via source iteration with optional DSA acceleration. This iterative method has the form: For i = 0, 1, · · · , until convergence, solve:

$$H\Psi^{i+1} = L^{+}\Sigma_{s} L\Psi^{i} + Q \tag{1}$$

where Ψ is the discrete flux, H is the discrete form of the streaming plus collision operator $\Omega\cdot\nabla + \sigma$, Q is the discrete source, L is the discrete moment operator, and the matrix Σ_s represents the discretized scattering operator. ARDRA is unusual in that the code stores the entire angular flux vector and builds the RHS of (1) before the sweep has occurred. On each cellset, the basic sweep algorithm in ARDRA has the form given in Figure 3. The model of the time to completion for this algorithm is:

$$T = GS(T_{task} + T_{comm}) + T_{RHS} = S\tilde{T} + T_{RHS} \tag{2}$$

with G being the number of groups, S being the number of sweep stages, $\tilde{T} = G(T_{task} + T_{comm})$ the time to calculate a single angle on a cellset and communicate downstream data and T_{RHS} the time to calculate the RHS. We now define parallel efficiency in Eq. 3, where T_{ref} is the calculation time for a reference number of processors, and T_P is the calculation time for P processors to compare to the reference calculation time.

$$\varepsilon = \frac{T_{ref}}{T_P} = \frac{S_{ref}\tilde{T} + T_{RHS}}{S_P\tilde{T} + T_{RHS}} = \frac{S_{ref} + T_{RHS}/\tilde{T}}{S_P + T_{RHS}/\tilde{T}}. \tag{3}$$

```
while not converged do
  Build RHS = $L^{+}\Sigma_{s} L\Psi^{i} + Q$ from $\Psi^{i}$
  for $g$ = 1:G do
    for $m$ = 1:M do
      Wait for incoming boundary data
      Choose angle ready to be swept
      for $z$ = 1:number spatial cells do
        Solve for $\Psi^{i+1}_{g,m,z}$
      end for
      Communicate out-going $\Psi^{i+1}$
    end for
  end for
end while
```

Fig. 3. ARDRA Solution Algorithm

PDT

PDT [6] is a massively parallel discrete ordinates transport code under development at Texas A&M University and built on the Standard Template Adaptive Parallel Library (STAPL) [7], also under development at Texas A&M University. PDT most often utilizes a hybrid KBA-Volumetric decomposition with cellset overloading, where one dimension has two-way spatial parallelism and the other two dimensions subdivide the remaining processors in a square fashion. PDT simultaneously sweeps from all eight corners using an optimal schedule [6]. For this paper, PDT uses a piecewise linear discontinuous finite element spatial discretization with eight spatial unknowns per cell.

PDT solves the discrete form of the transport problem via source iteration and variants, including Krylov methods with and without diffusion preconditioners. This requires repeated solution of the sweep equation:

$$H\Psi^{i+1} = L^{+}\Sigma_{s}\Phi^{i} + L^{+}Q \tag{4}$$

where $\Phi^{i} = L\Psi^{i}$ contains the moments of the angular flux and L^{+} is the moments-to-discrete operator. For this, PDT executes the algorithm in Figure 4. A sweep task is the execution of the sweep operator on a cellset for a set of quadrature directions (an "angleset") and a set of energy groups (a "groupset"). The sizes of cellsets, anglesets, and groupsets are determined by aggregation parameters, which can be user-defined or code-selected at runtime. The time to completion model is:

$$T_{sweep} = N_{stages} \quad T_{comm} + T_{wu} + \\ N_{dof} A_c \left[T_c + A_m \quad T_m + A_g T_g \quad \right] \tag{5}$$

where T_{comm} is the communication time, T_{wu} is the time per degree of freedom spent getting into a work function, N_{dof} is the number of spatial degrees of freedom per cell, A_c is the number of cells in a cellset, T_c is the time per degree of freedom spent in the work-function cell loop outside of the angle loop, A_m is the number of angles in an angleset, T_m is the time per degree of freedom spent in the work-function angle loop outside of the group loop, A_g is the number of groups in a groupset, and T_g is the time per degree of freedom spent in the work-function group loop. Parallel efficiency is defined as

$$\varepsilon = \frac{T_{sweep,ref}}{T_{sweep,P}} \tag{6}$$

PDT's basic solver algorithm is shown in Figure 4.

```
while not converged do
  Build B = $\Sigma_s \Phi^i + Q$
  for t = 1:Ntask do
    Obtain next task ID from schedule
    Wait for incoming boundary data
    for z = 1:cells in cellset do
      Choose next cell from graph
      for m = 1:angles in angleset do
        for g = 1:groups in groupset do
          Build $RHS_{m,g,z}$ from $B_{g,z}$ using $L^+$
          Solve for $\Psi^{i+1}_{m,g,z}$
          Accumulate $\Psi^{i+1}_{m,g,z}$ into $\Phi^{i+1}_{m,g,z}$
        end for
      end for
    end for
    Communicate out-going $\Psi^{i+1}_{m,g,z}$
  end for
end while
```

Fig. 4. PDT Solution Algorithm

RESULTS

We present results from ARDRA and PDT, and compare them to predictions from the previously defined parallel performance models (Eq. 3 and 6). Both transport codes have successfully demonstrated the ability to perform the full-domain transport sweep using more than 1Million MPI ranks.

ARDRA

ARDRA's scaling problem is based on the Jezebel criticality experiment. We ran this problem in 3D with all vacuum boundary conditions, 48 energy groups and three quadrature sets: S_8, S_{12}, and S_{16}. We perform two weak scaling studies: one with spatial parallelism only, and the second with a mixture of energy and spatial parallelism. We ran standard power iteration for k-effective, stopping the run at 11 iterations for the purpose of throughput. Both of our weak scaling studies start with one node of Sequoia, using 16 MPI ranks, with 1 rank per CPU core. Both studies have an initial 48 × 24 × 24 spatial mesh, but decompose the problem differently among the 16 ranks. In our first weak scaling study we decompose the problem into 12 × 12 × 12 cells per rank, with the resulting spatial decomposition of Px = 4, Py = 2, and Pz = 2. Our second study uses 16-way on-node energy decomposition, with each rank having the entire 48 × 24 × 24 spatial mesh, but only 3 energy groups, resulting in Px = Py = Pz = 1. Weak scaling is achieved by increasing the number of spatial cells proportional to increasing the processor count. ARDRA's largest run is at SEQUOIA's full scale, which is 37.5 trillion unknowns using 1,572,864 MPI ranks, resulting in 71% parallel efficiency using energy and spatial parallelism. Results are shown in Figures 5 and 6.

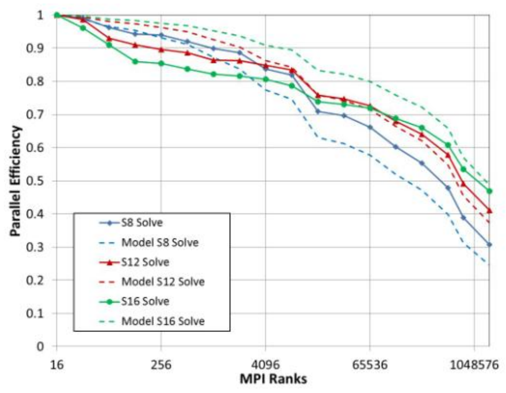

Fig. 5. ARDRA Weak Scaling (Spatial Parallelism)

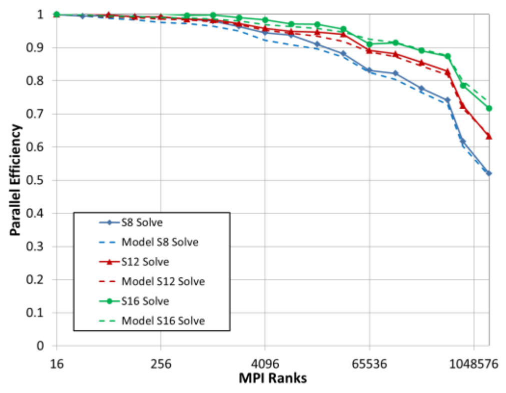

Fig. 6. ARDRA Weak Scaling (Spatial and Energy Parallelism)

PDT

PDT's weak-scaling study was performed using a 3D problem introduced by Zerr and Azmy [8] with a single energy group, S_8 level-symmetric quadrature, vacuum boundaries, 4096 cells/core, and spatial parallelism only. The cell and sub-domain sizes remain constant as the number of MPI ranks is increased because the volume of the problem domain is increased in proportion to number of ranks. Figure 7 shows our model prediction and data out to 384K MPI ranks on the SEQUOIA machine. Each case was run using 1 MPI rank/core. We have also demonstrated PDT's ability to run with 512K and 1M MPI ranks on the VULCAN machine. In these cases, each job was run with 4 MPI ranks/core (one rank per hardware thread). Based on previous experience, the deviation of the data from the model at high rank counts indicates a code issue or inefficiency that needs to be resolved rather than a deficiency in the model itself.

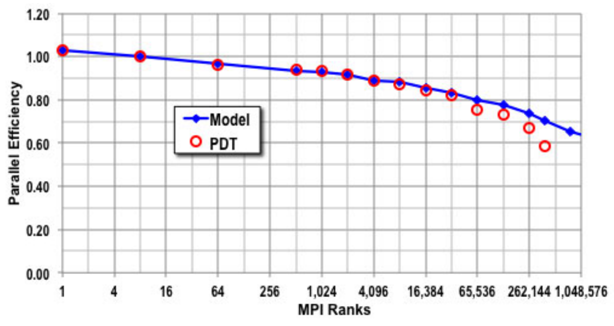

Fig. 7. PDT Weak Scaling (Spatial Parallelism)

CONCLUSION

These computational results validate the predictions made by the sweep algorithm performance models. Codes based on sweep algorithms and domain decompositions implemented as described here and in the referenced publications will scale to millions of MPI ranks and potentially beyond. The strength of this study is the demonstration of the validity of sweep algorithm scaling predictions using two vastly different codes with markedly different domain decomposition and spatial discretization choices. Additional research into algorithms and strategies, as well as code improvements, will almost certainly result in improved efficiencies.

ACKNOWLEDGMENTS

Part of this work was performed under the auspices of the U.S. Department of Energy (DOE) by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Part was funded under a collaborative research contract from Lawrence Livermore National Security, LLC. Part was performed under the Center for Exascale Radiation Transport at Texas A&M University, which is funded by the DOE National Nuclear Security Agency under award number DE-NA0002376.

REFERENCES

1. M. L. ADAMS and E. W. LARSEN, "Fast iterative methods for discrete-ordinates particle transport calculations," *Prog. Nucl. Energy*, **40**, No. 1, pp. 3-159, (2002).

2. R. S. BAKER and K. R. KOCH, "An Sn Algorithm for the Massively Parallel CM-200 Computer," *Nucl. Sci. Eng*., **128**, p. 312 (1998); dx.doi.org/10.13182/NSE98-1.

3. M. R. DORR and C. H. STILL, "Concurrent Source Iteration in the Solution of Three-dimensional, Multigroup, Discrete Ordinates Neutron Transport," *Nucl. Sci. Eng*., **122**(3), pp. 287-308 (1996).

4. C. CLOUSE, "Parallel Deterministic Neutron Transport with AMR," *Proc. Computational Methods in Transport Workshop*, Tahoe City, CA, September 11-16, 2004.

5. U. HANEBUTTE AND P. N. BROWN, "ARDRA, Scalable Parallel Code System to Perform Neutron and Radiation Transport Calculations," *Lawrence Livermore National Laboratory Technical Report UCRL-TB-132078*, February (1999).

6. M.P. ADAMS, M.L. ADAMS, W.D. HAWKINS, T. SMITH, L. RAUCHWERGER, N.M. AMATO, T.S. BAILEY, and R. D. FALGOUT, "Provably Optimal Parallel Transport Sweeps on Regular Grids*," Proc. International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering*, Sun Valley, ID, May 5-9, CDROM (2013).

7. A. BUSS, HARSHVARDAN, I. PAPADOPOULOS, O. PEARCE, T. SMITH, G. TANASE, N. THOMAS, X. XU, M. BIANCO, N. M. AMATO, L. RAUCHWERGER, "STAPL: Standard Template Adaptive Parallel Library," *SYSTOR*, Haifa, Israel, June 4-6, 2010, ACM, pp.1–10, http://doi.acm.org/

8. R. J. ZERR and Y. Y. AZMY, "Solution of the Within- Group Multidimensional Discrete Ordinates Transport Equations on Massively Parallel Architectures," *Trans. Amer. Nucl. Soc*., **105**, 429 (2011).

9. T. S. BAILEY and R. D. FALGOUT, "Analysis Of Massively Parallel Discrete-Ordinates Transport Sweep Algorithms With Collisions," *Proc. International Conference on Mathematics*, *Computational Methods & Reactor Physics*, Saratoga Springs, May 3-7, CDROM (2009).

10. W. D. HAWKINS, T. SMITH, M. P. ADAMS, L. RAUCHWERGER, N. M. AMATO, and M. L. ADAMS, "Efficient Massively Parallel Transport Sweeps," *Trans. Amer. Nucl. Soc*., **107**, pp. 477-481 (2012).