A Transport Sweep MiniApp:

Users Manual

1 Introduction

DFEComm is a mini-app developed for the Center for Exascale Simulation of Advanced Reactors (CESAR) co-design center. The mini-app is purposed with simulating the performance characteristics of software that utilizes transport sweeps to solve the neutron transport equation.

This user manual will briefly illustrate how parallel transport sweeps are performed, detail the input file format used in DFEComm, and describe the options available to the user. The Future Work section will outline features to be implemented in future versions of DFEComm.

2 Parallel Transport Sweeps

The S_N, or discrete ordinates, form of the neutron transport equation is:

$$\hat{\Omega}_m \cdot \nabla \psi_m^g(\mathbf{r}) + \sigma_t^g(\mathbf{r}) \psi_m^g(\mathbf{r}) = q_{tot}^g(\mathbf{r}, \hat{\Omega}_m), \tag{1}$$

where $\psi_m^g(\mathbf{r})$ is the angular flux, $\sigma_t^g(\mathbf{r})$ is the macroscopic total cross section, $q_{tot}^g(\mathbf{r}, \hat{\Omega}_m)$ is the total neutron source, and $\hat{\Omega}_m$ is the m^{th} quadrature direction. The total source provides the only coupling between quadrature directions. If the source is lagged, the equation for each angle can be solved independently of the others. A *sweep* solves Eq. 1 with a given iterate for the total source for all directions and energies.

Equation 1 is solved in each spatial cell through an upwinding scheme in which information from cells *upstream* of the current cell is used as a boundary condition. Here, *upstream* is defined as in the direction opposite the current quadrature direction, and *upstream cells* are those for which the solution has already been calculated. Once the equation is solved in the cell, the solution is then transmitted to its *downstream* neighbor. In this way, the method *sweeps* through the problem.

Take for example a 1D mesh, seen in Figure 1. Given a sweep direction in the positive x direction, cell 1 must be solved first, using information from the left boundary of the problem. Once the angular flux in cell 1 has been computed, cell 2 may be solved using the solution in cell 1. Then cell 3 can be computed using cell 2's solution, etc. For this sweep direction, cell 1 is considered upstream of cell 2, and cell 2 is downstream of cell 1. Likewise, if the sweep direction is in the negative x direction, cell 8 must be solved first using information from the right boundary, followed by 7, then 6, etc. In this case, cell 1 would be downstream of cell 2, and cell 2 would be upstream of cell 1.

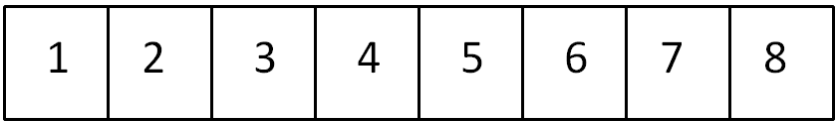

Figure 1: 1D mesh

When solving Eq. 1 using multiple processes, the spatial mesh is *partitioned* among the shared memory locations, or SMLs. A shared memory location is a unit of computational hardware which has a shared block of memory. For example, this could be a single MPI process, or a node with multiple threads. Details on the partitioning process are described below in Section 3. Given a partitioning, the spatial cells, quadrature directions, and energy groups are *aggregated* into cellsets, anglesets, and groupsets, respectively. A triplet of cellset, angleset, and groupset define a *task*. A *task* is then defined as the amount of work an SML must perform before communicating to its downstream neighbors. A

simple, reasonably good model for parallel sweep efficiency is:

$$E \approx \frac{1}{\left[1 + \frac{N_{idle}}{N_{tasks}}\right]\left[1 + \frac{T_{comm}}{T_{task}}\right]} \tag{2}$$

where N_{idle} is the number of idle stages each process has, N_{tasks} is the number of tasks each SML performs, T_{comm} is the communication time, and T_{task} is the time it takes to perform a task. The aggregation of the problem allows the parallel efficiency to be optimized. For example, choose a large number of tasks minimizes the first bracketed term, but maximizes the second, since the time per task is approximately inversely proportional to the number of tasks.

3 Partitioning

In their 2009 presentation at the Mathematics, Computational Methods, and Reactor Physics conference, Bailey and Falgout presented a generalized method for partitioning the spatial mesh among multiple processes. The terminology in that paper is used in DFEComm, and is briefly described below. The reader is referred to [1], [2], and [3] for more detailed discussions on the theory behind and analysis of these partitioning methods as they are applied to transport sweeps.

Each dimension (in SML space) is given one of two *distribution functions*: round robin or blocked. Round robin refers to a sequence that increments and then repeats, e.g. 1 2 3 4 1 2 3 4. Blocked refers to a sequence that repeats and then increments, e.g. 1 1 2 2 3 3 4 4. Each dimension is also given an *overload factor*, which is the number of spatial cellsets given to each SML. In the previous examples, the overload factor is 2 (each number is repeated twice). Finally, each dimension is given a total number of spatial cellsets that the mesh is broken up into. In both previous examples, the number of cellsets is 8 (there are 8 numbers). The quotient of the total number of cellsets and the overload factor is the number of SMLs in that dimension. The total number of SMLs is the product of the number of SMLs in each dimension.

In DFEComm, there are several preset partitioning schemes available to the user: Koch-Baker-Alcouffe (KBA), Hybrid-KBA, Volumetric, and Volumetric with Round Robin. In addition, the user is allowed to define their own partitioning scheme using partition functions, overload factors, and total number of cellsets. Table 1 defines the partitioning parameters for these pre-defined schemes. The implementation of these preset partitions are found in *Problem_Input.cc* in the *GetPartitionParameters()* function, if the user would like to define their own. As a note, the division by 2 in the number of cellsets for partition types 0 and 2 are removed if the problem is run on 1 SML.

Partition Type	Description	Partition Function	Overload Factor	Number of Cellsets (in x,y, and z)
0	Default Hybrid-KBA	Blocked	x = 1, y = 1, z = *z_planes*	x = $\sqrt{\frac{\#\text{ SML}}{2}}$ y = $\sqrt{\frac{\#\text{ SML}}{2}}$ z = $\frac{z\_planes}{2}$
1	KBA	Blocked	x = user defined y = user defined z = *z_planes*	x = $\sqrt{\frac{\#\text{ SML}}{\text{overload}}}$ y = $\sqrt{\frac{\#\text{ SML}}{\text{overload}}}$ z = *z_planes*
2	Hybrid-KBA	Blocked	x = user defined y = user defined z = *z_planes*	x = $\sqrt{\frac{\#\text{ SML}}{2*\text{overload}}}$ y = $\sqrt{\frac{\#\text{ SML}}{2*\text{overload}}}$ z = $\frac{z\_planes}{2}$

4	Volumetric	Blocked	x = 1
			y = 1
			z = 1
				x = # SML$^{1/3}$
				y = # SML$^{1/3}$
				z = # SML$^{1/3}$

5	Volumetric with Round Robin	Round Robin	x = user defined y = user defined z = user defined	x = $\left(\frac{\#\text{ SML}}{\text{overload}}\right)^{1/3}$ y = $\left(\frac{\#\text{ SML}}{\text{overload}}\right)^{1/3}$ z = $\left(\frac{\#\text{ SML}}{\text{overload}}\right)^{1/3}$
6	User Defined Partitioning	User Defined	x,y, z = user defined	x,y, z = user defined

Table 1: Partition Types

3.1 Auto-partitioning

Currently, DFEComm has the ability to auto-partition the mesh. Given an SML count, the code attempts to find the optimal layout of SMLs. The ideal case is a hybrid-KBA layout with 2 SMLs in the z dimension, and $\sqrt{\frac{\#\text{SML}}{2}}$ SMLs in both the x and y dimensions. If $\sqrt{\frac{\#\text{SML}}{2}}$ is not an integer, a function computes the prime factorization of $\sqrt{\frac{\#\text{SML}}{2}}$ and chooses an SML layout closest to a square, assigning the larger SML count to the x dimension. For example, given 12 SMLs, the auto-partition function would provide a layout of 4 SMLs in x, 3 in y, and 2 in z. An even number of SMLs is required so that the z-dimension can always be divided into 2 and that hybrid-KBA can be used. Currently auto-partitioning is available only for the preset problem sizes, defined in Section 5.1.

4 Compiling and Running DFEComm

DFEComm is available for download on GitHub. To clone the repository on GitHub:

```
>$ git clone https://github.com/cnmcgraw/DFEComm.git
```

Once cloned, the code can be updated to the newest version using the following command (when in the DFE_Comm directory):

```
>$ git pull
```

DFEComm is built upon MPI and has been tested using mpi++ as its compiler. To compile DFEComm, use the following command:

```
>$ make
```

To run DFEComm, use the following command:

```
>$ mpirun -n [# SML] ./DFEComm -f [Input file]
```

Here the input file must be a text file in the format described in the next section. The number of SMLs indicated at run time must match that found in the input file.

5 Input File Structure

The input file is a text file with a listing of case sensitive keywords separated from their values by an "=" surrounded by at least one set of spaces. Lines beginning with "#" followed by at least one space are comments. The file format can take multiple formats, and blank lines and the order of keywords do not matter. A listing of all available keywords, their definitions, and their available values is seen in Table 2. Several example inputs are available with the DFEComm source code to assist users.

Input Parameter	Values Allowed	Default Value	Definition
problem_size	0,1,2,3,4	0	Predefined Problem Size per SML
num_pin_x	Positive Integer	None	Number of fuel pins in the x-dimension
num_pin_y	Positive Integer	None	Number of fuel pins in the y-dimension
refinement	Positive Integer	None	Number of 'rings' in each fuel pin
z_planes	Positive Integer	None	Number of mesh cells in the z-dimension
bc	1,2	2	Boundary Conditions, 1 = Reflecting, 2 = Incident Flux
num_polar	Positive Even Integers	None	Total number of polar angles
num_azim	Positve Integers Divisible by 4	None	Total number of azimuthal angles
angle_aggregation	1,2,3	None	Angle Aggregation, 1 = Single, 2 = Polar, 3 = Octant
num_groups	Positive Integer, Divisible by num_groupsets	None	Total number of energy groups
num_groupsets	Positive Integer	None	Number of energy groupsets
num_SML	Positive Integer	None	Number of Shared Memory Locations
partition_type	0,1,2,3,4,5,6	0	Type of spatial partitioning, options explained in Table 1
func_{x,y,z}	1,2	None	Partition function to use in {x,y,z}, 1 = blocked, 2 = round robin
overload_{x,y,z}	Positive Integer	None	Overload factor in {x,y,z}
overload	Positive Integer	None	Total overload factor, used with partition_type = 1,2, and 4
num_cellsets_{x,y,z}	Positive Integer	None	Total number of cellsets to divide the problem into in {x,y,z}
num_sweeps	Positive Integer	None	Number of transport sweeps to perform and average over

Table 2: All Available Input Parameters

An example of the most verbose form of the input file can be seen in Figure 2. This form allows the user complete control over the definition of their problem. This example defines a problem with 2 cellsets, each with a 16x16x16 mesh layout. Each of the two SMLs specified have one cellset. There are 80 total angles, and octant aggregation is used. There is one groupset, with one energy group. Finally, 10 sweeps will be performed and the average time per sweep will be reported. This example is found in *Example_Input.txt* in the DFEComm folder.

```
# Space
num_pin_x = 1
num_pin_y = 1
refinement = 8
z_planes = 32
bc = 2

# Angle
num_polar = 10
num_azim = 8
angle_aggregation = 3

# Energy
num_groups = 1
num_groupsets = 1

# Partitioning
num_SML = 2
partition_type = 6

func_x = 1
overload_x = 1
num_cellsets_x = 1
func_y = 1
overload_y = 1
num_cellsets_y = 1
func_z = 1
overload_z = 1
num_cellsets_z = 2

num_sweeps = 10
```

Figure 2: Example Verbose Input File

On the other end the spectrum, a much simpler form of the input can be seen in Figure 3. This form is intended to allow the user to define a problem using the least amount of keywords. DFEComm has as a collection of preset problems which are defined per SML, specified with only one keyword, *problem_size*. The specifics of these predefined problems are detailed in the next section. For this input format, only the number of SMLs and the number of sweeps to perform are further required. DFEComm uses the auto-partitioning function and hybrid-KBA to divide the mesh among the SMLs. This example input is found in *Example_Input_ProblemSize.txt* in the DFEComm folder.

```
# Simplified Input
problem_size = 1
num_SML = 2
num_sweeps = 5
```

Figure 3: Example Simplified Input File

An example of an input using one of the predefined partitioning schemes in seen in Figure 4. The *overload* keyword, required for *partition_types* 1, 2, and 4, defines the overload factor for both the x and y dimensions.

```
# Space
num_pin_x = 1
num_pin_y = 1
refinement = 8
z_planes = 32
bc = 2

# Angle
num_polar = 10
num_azim = 8
angle_aggregation = 3

# Energy
num_groups = 1
num_groupsets = 1

# Partitioning
num_SML = 2
partition_type = 2
overload = 1

num_sweeps = 10
```

Figure 4: Example Predefined Partitioning Input File

5.1 Pre-defined Problems

Four predefined problems are available in DFEComm to enable weak scaling studies. Each problem size, defined in Table 3, is defined per SML. The angular quadrature is aggregated by octant, and the energy groups are aggregated into one groupset. Currently, the spatial mesh is aggregated into one cellset per SML. This will be optimized in future versions of DFEComm. The largest of these problems (4), is intended to test the limits of the machine DFEComm is run on. It is also meant to be extended to an on-node mini-app in which the quadrature and energy groups are split up over threads on a SML.

Problem Size ID	Spatial Mesh	Quadrature (# Polar × # Azimuthal)	Number of Energy Groups	Unknowns per Task per SML
1	2 x 2 x 10	4 x 4	10	800
2	10 x 10 x 100	4 x 16	50	4M
3	10 x 10 x 200	16 x 16	100	64M
4	10 x 10 x 200	16 x 64	500	1.2B

Table 3: Pre-defined Problem Definitions

5.2 Future Work

DFEComm is an ongoing research effort, and therefore is still under active development. As such, a few desired features not yet implemented into DFEComm are set to become available soon. This includes:

- Reflecting Boundary Conditions
- Allowing the number of SMLs to be defined via the command line
- Defining a default number of sweeps to perform
- Multiple Cellsets per SML
- Optimization of data movement to improve weak scaling efficiency

References

[1] 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).

[2] W. Daryl Hawkins, Timmie Smith, Michael P. Adams, Lawrence Rauchwerger, Nancy M. Amato, and Marvin L. Adams, "Efficient Massively Parallel Transport Sweeps," *Trans. Amer. Nucl. Soc.*, **107**, pp. 477-481 (2012).

[3] M.P. Adams, M.L. Adams, W.D. Hawkins, T. Smith, L. Rauchwerger, N.M. Amato, T.S. Bailey, 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, Idaho, USA, May 5-9, CD-ROM (2013).