Space-filling Curves for Domain Decomposition in Scientific Simulations

Aparna Sasidharan, Marc Snir

***Abstract*—Space-filling curves(SFC) have been widely accepted as an easy technique to generate good quality mesh partitions. In this work, we discuss the limitations of some of the existing methods for generating SFCs and propose a recursive algorithm for constructing a general SFC that works for a range of meshes in 2D and 3D. All of our test cases for the 2D SFC come from the meshes used in Community Earth System Model(CESM). We compare the quality of SFC partitions with the existing strategies in CESM in terms of load balance and communication cost.**

I. Introduction

A space-filling curve is essentially a mathematical function that generates a linear order out of points in higher dimensions with reasonably good locality. This linear list can then be sliced into p pieces, each with a roughly equal number of points, in order to partition the points among p processors. This can be used to partition meshes: each cell is represented by one point contained in the cell, such as its center of gravity, and the cells are partitioned by partitioning their representative points.

Several space-filling curves have been described in the past - Hilbert, Peano and Sierpinski curves are some of them [1]. The definitions of these curves are dependent on the geometry of the domain which makes it difficult to use them to partition arbitrary meshes. In this work, we describe a general SFC whose definition is independent of the shape and size of the domain. We use this generalized SFC to partition meshes used in the *Community Earth System Model* (CESM) [2] and present results on the quality of the resulting partitions. We use two measures of goodness for mesh partitions.

We use the following definitions:

- $Volume(i)$ to be the number of cells allocated to processor i; this is a measure of the computational load at that processor.
- $Comm(i,j)$ is the number of cells allocated to processor i that are adjacent to cells allocated to processor j. This is a measure of the communication from i to j. $Comm(i,j)$ and $Comm(j,i)$ are not necessarily equal.
- $Surface(i) = \sum_{j \neq i} Comm(i,j)$. This is a measure of the communication from processor i.

We consider two metrics:

- *Maximum Load* defined as $MaxLoad = \max_i Volume(i)$. This is a measure of maximum compute load allocated to a processor
- *Maximum Communication* defined as $MaxComm = \max_i Surface(i)$. This is a measure of the maximum communication performed by a processor.

We shall typically seek to have the maximum load very close to the average load, so that the computation is *load balanced.*

II. General Space-filling Curve

A space-filling curve in 2D space f provides a mapping from $[0,1]$ to $[0,1]^2$. f^{-1} converts points in 2D into a linear order, maintaining locality, i.e points that are close to each other in 2D are relatively close on the curve. If adjacent points on the curve are not neighbours in 2D, we describe that as a jump or a discontinuity in the order. The fewer the number of discontinuities in f^{-1}, the better the space-filling curve. We use recursion to construct the general SFC. Points in the d-dimensional domain are arranged in a kd-tree. f^{-1} is described by the order of traversal of the nodes in the kd-tree. The binary tree (k=2 for the kd-tree) is built in a top-down manner starting with all the points in the domain assigned to the root. At every node in the tree, the datapoints are separated based on the splitting dimension and split value, and assigned to its left($\leq$ split value) and right($>$ split value) children. The recursion ends when we are left with only one point in a node (leaf node). We construct a tight bounding box around the points in a node. A simple splitter function computes the midpoint of this bounding box along the dimension of maximum spread.

We generate the 2D SFC by traversing the binary tree based on a set of 4 rules and their reflections. Each node in the tree corresponds to a subdomain. So visiting a node is equivalent to entering and exiting a subdomain by the SFC. Since the points in a node are enclosed in their tightest bounding box (four-sided polygon),we have assigned four directions (sides) and four entry/exit points (corners) for each subdomain. Our rules are defined based on these directions and entry/exit points. Each rule in turn generates the order of traversal for the subcells of that node, depending on whether it is resolved along both dimensions or one of them.

All the rules are implemented as simple transformations of a base rule.

We use the same techniques to generate a 3D SFC. The generating rules in 3D are analogues to the 2D rules. We have a total of 12 different rules for the 3D SFC, corresponding to the number of edges of a cube.

The partitioning time of the SFC algorithm is $O(nlogn)$ and the memory consumption is $O(n)$ where n is the number of datapoints. This makes it much cheaper and faster than the multi-level partitioning algorithms employed by packages like Metis [3].

	Direct k-way		SFC	
$\#procs$	maxload	max.surf	maxload	max.surf
512	1751	258	1728	188
1000	905	207	885	202
1024	889	198	864	117
2000	455	130	443	145
2048	444	134	432	92
3000	303	108	294	113
4096	222	91	216	56
5000	182	86	177	91
6000	151	78	148	77
8192	111	64	108	44

TABLE I
QUALITY OF PARTITIONS FOR DIRECT K-WAY AND SFC ALGORITHMS FOR THE STRUCTURED 768X1152 GRID

III. EXPERIMENTS

We implemented our SFC algorithm in C++ and provide a framework for evaluating the quality of partitions with respect to other partitioning schemes. The code reads the input meshes as a vector of points, partitions and computes the maximum volume and surface area for a given partition. The communication pattern and weights (for a weighted mesh) have to be conveyed to the code. We also have a framework in OpenCV for visualizing the SFC and the partitions.

A. Structured $k1 \times k2$ meshes

The test cases in this category come from the atmosphere model of CESM. We used the latitude-longitude grids in the FV dycore as inputs. There could be other applications that use similar meshes and might benefit from our tests. We compared the load balance and surface area of the SFC partitions with the direct k-way algorithm of Metis. CESM uses a block decomposition to partition these meshes. Both Metis and SFC perform better than a naive 2D block decomposition in terms of load balance and surface area of partitions. We used the largest structured grid available in CAM-FV for our experiments. The grid has 1152 longitudes and 768 latitudes which makes the horizontal grid resolution 0.23x0.31.

Table I summarizes the quality of partitions generated by SFC and direct k-way algorithms. The SFC partitions seem to be better than the Metis partitions in terms of both load balance and surface area when the number of processors is a power of 2. For the remaining values the Metis partitions do slightly better in communication at the cost of worse load balance. In short, space-filling curves seem to be a very good choice for partitioning structured $k1 \times k2$ grid.

B. Unstructured meshes

1) Uniformly Refined: We used the unstructured meshes from MPAS-Atmosphere (MPAS-A) model as test cases. We first projected the points onto a 2D surface. To obtain good quality partitions, we chose a projection with minimum distortion. We split the sphere into two hemispheres(north and south), projected them separately and generated SFCs for each half.

	SFC		Direct k-way	
$\#procs$	maxload	max.surf	maxload	max.surf
512	1281	174	1308	172
1024	641	124	659	143
2048	321	90	329	88
4096	161	62	164	61
8192	81	43	82	39

TABLE II
QUALITY OF SFC AND DIRECT K-WAY PARTITIONS FOR 30KM UNSTRUCTURED MESH WITH 655362 CELLS

	SFC		Direct k-way(Metis)	
$\#procs$	maxload	max.surf	maxload	max.surf
512	4096	1352	4219	2064
1024	2048	872	2109	1291
2048	1024	520	1054	758
4096	512	296	527	437
8192	256	184	263	243
16384	128	104	131	130
32768	64	56	65	65

TABLE III
QUALITY OF SFC AND DIRECT K-WAY PARTITIONS FOR THE STRUCTURED $128 \times 128 \times 128$ 3D GRID

We compared the quality of SFC partitions against their corresponding Metis partitions with increasing number of processors. Table II summarizes the maximum load and boundary length for the partitions generated by SFC and direct k-way algorithms for the same unstructured mesh. The results in table II show that the SFC partitions match upto the Metis partitions. In some cases, the SFC partitions are even better than Metis in terms of both load balance and surface area.

C. 3D Grids

1) Structured Grids: CESM does not use any 3D meshes, but we include some preliminary results in this section for completeness. For the structured category, we generated a large dataset of $128 \times 128 \times 128$ equi-distant points in 3D and compared the quality of partitions with those generated by the direct k-way algorithm of Metis. Like in the 2D case, SFC partitions are better than the Metis generated ones.

Table III provides a comparison of the SFC and Metis generated partitions. Although this might be an ideal dataset since the dimensions are powers of 2, and the points are equidistant, we expect the SFC algorithm to do well for other cases as well, like in 2D.

IV. ONGOING WORK

We are currently working on applying the SFC algorithm to adaptive meshes in 2D and 3D. We intend to supplement this work with refined performance models for typical applications and use the models to obtain better partitions.

REFERENCES

[1] H. Saagan, *Space-Filling Curves*. Springer-Verlag, 1994.
[2] NCAR, www2.cesm.ucar.edu.
[3] G. Karypis and V. Kumar, *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*, September 1998.