

Partitioning with Space-Filling Curves on the Cubed-Sphere

John M. Dennis
Scientific Computing Division
National Center for Atmospheric Research
P.O. Box 3000
Boulder, CO 80307
dennis@ucar.edu

Abstract

Numerical methods for solving the systems of partial differential equations arising in geophysical fluid dynamics rely on a variety of spatial discretization schemes (e.g. finite difference, finite element). For parallel execution on distributed memory computers, the computational domain must be partitioned. The choice of partitioning algorithm can have a significant impact on the sustained floating point execution rate of an atmospheric model. The NCAR spectral element atmospheric model employs a gnomonic projection of a cube onto the surface of the sphere. The six cube faces are each subdivided into an array of quadrilateral spectral elements. When the cubed-sphere is partitioned using METIS, both computational load imbalance and communication requirements can lead to sub-optimal performance. In this paper, Hilbert, Peano, and nested Hilbert m -Peano space-filling curves are investigated as the basis of alternative partitioning algorithms. The resulting partitions allow a maximum 22% increase in the sustained floating point execution rate versus METIS on $\mathcal{O}(1000)$ processors, when running a relatively high resolution climate simulation.

1. Introduction

The solution of partial differential equations in geophysical fluid dynamics requires enormous computational resources. For example, climate simulation requires multiple, century long integrations of the equations governing the earth's atmosphere. The spectral element method is a promising horizontal discretisation scheme for the construction of dynamical cores of climate models. The computational domain is subdivided into spectral elements, where the model fields are approximated by a high order polynomials. C^0 continuity is imposed along element boundaries

that share degrees of freedom. In the spectral element atmospheric model (SEAM) originally developed by Taylor et al [9], the sphere is tiled with rectangular elements by subdividing the six faces of the cube, which circumscribes the sphere, and then a gnomonic projection maps these elements onto the surface of the sphere.

The cubed-sphere computational domain is displayed in Figure 1, where each cube face contains an array of $Ne \times Ne$ quadrilateral spectral elements. The total number of spectral elements is $K = 6 \times Ne \times Ne$. Parallel execution on distributed memory computers implies that the cubed-sphere must be partitioned across N_{proc} processors. For the purposes of partitioning we consider a spectral element to be indivisible, and represents the atomic data structure assigned to a processor. Communication between processors is determined by neighboring elements that share a boundary or corner point.



Figure 1. The cubed-sphere with continental outline for $Ne = 8$

Because climate modeling requires century long sim-

ulations, the spatial resolution is generally relatively coarse and the parallelism is relatively high, resulting in $\mathcal{O}(1)$ to $\mathcal{O}(10)$ elements per processor. Typical climate resolutions require anywhere from $K=384$ ($N_e = 8$) to $K=3456$ ($N_e = 24$) total spectral elements. Previously, it was observed that the floating point execution rate of SEAM, using partitioning algorithms provided by METIS [3], is adversely affected by computational load imbalance when running on $\mathcal{O}(1000)$ processors [4].

A promising technique for eliminating these load imbalances is based on space-filling curves. Space-filling curves (SFC) have been successfully applied in parallel adaptive mesh refinement strategies [1] [7] [2] [5]. It has not been widely recognized that these can be applied to static partitioning as well. Consider a square domain of size $P \times P$. A Hilbert curve can be used to partition this domain if $P = 2^n$, or a meandering Peano (m-Peano) if $P = 3^m$, where n and m are integers referring to the recursive level [8] of the Hilbert and m-Peano curve respectively. A new space-filling curve is introduced combining Hilbert and m-Peano curves, allowing partitions of size $P = 2^n 3^m$.

A comparison of the METIS and SFC approaches reveals significant differences in the floating point execution rate obtained. While the execution rate of SEAM using the SFC partitions is comparable to METIS at small processor counts, the SFC partitions result in much faster execution rates at large processor counts. The partitions generated by the SFC algorithm lead to a maximum 22% increase in the execution rate on $\mathcal{O}(1000)$ processors. Unlike METIS, the SFC algorithm places restrictions on the problem size and processor count and is not a fully general solution.

This paper is organized as follows. Section 2 reviews graph partitioning and partitioning algorithms provided by METIS. A review of space-filling curves, the introduction of a combined Hilbert and m-Peano curve (Hilbert-Peano), and an algorithm to generate a Hilbert-Peano curve are presented in Section 3. Results of several performance studies are detailed in Section 4.

2. METIS partitioning

Partitioning of the cubed-sphere with METIS requires the formation of an undirected graph. An undirected graph $G = [V, E]$ consists of a set vertices V and edges E . Both the edges E and the vertices V can be assigned a numerical weight. In the context of spectral elements, weights associated with edges E represent the amount of information which must be exchanged along each element boundary, while a vertex weight represents the amount of computation associated with the element. A graph partition is

defined as the set of all sub-graphs created by the partitioning algorithm.

The METIS graph partitioning algorithms attempt to minimize either, *edgcut*, or *total communication volume* [3]. Edgcut is defined as the number of graph edges that straddle all sub-graphs. Total communication volume is defined as the number of vertices whose edges are cut by the partition. Another important concept is the *load balance* of a partition. The load balance $LB()$ of a set $S = \{s_1, s_2, \dots, s_n\}$ is calculated as

$$LB(S) = (\max\{S\} - \text{avg}\{S\}) / \max\{S\}. \quad (1)$$

The computational load balance of a partition is calculated using (1) where S is the number of vertices contained in each sub-graph. The communication load balance of a partition is calculated where S is the number of vertices whose edges are cut in each sub-graph. Because it is impossible to optimize all properties simultaneously, each of the graph partitioning algorithms provided by METIS attempts to minimize a particular property of the graph. METIS provides two fundamental graph partitioning algorithms; recursive bisection and K-way partitioning. The recursive bisection (RB) algorithm is best for load balancing, but results in larger edgcuts and total communication volume. The K-way (KWAY) algorithm generates partitions that minimize edgcuts but may result in sub-optimal load balance. A variant of the K-way algorithm minimizes the total communication volume (TV).

3. Space-Filling Curves

It has not been widely recognized in the literature that space-filling curves can be used to construct a static mesh partitioning algorithm. A space-filling curve is defined to be a bijective function that maps a line into a multi-dimensional domain. The mapping of a 2-dimensional domain is achieved in two steps. First, recursively divide the initial or parent domain into four child sub-domains. Next, connect neighboring sub-domains with a space-filling curve. Panel *a* of Figure 2 illustrates a parent domain and its four child sub-domains connected by a level 1 Hilbert curve. The *major vector* indicates orientation and direction of travel for the U-shaped Hilbert curve through the parent domain. Panel *b* of Figure 2 illustrates both the major vector and the *joiner vector* for each child sub-domain. The joiner vector points in the direction of the next sub-domain visited by the space-filling curve. The orientation of the major vector computed for each of the sub-domains is based on the mathematical definition of a Hilbert curve [8]. Note that our use of the term vector is based on the

direction and *axis* terminology first introduced by Pilkington and Baden [6]. Given the major vectors computed and displayed in panel *b* of Figure 2, the resulting refinement into a level 2 curve is illustrated in panel *c* of Figure 2.

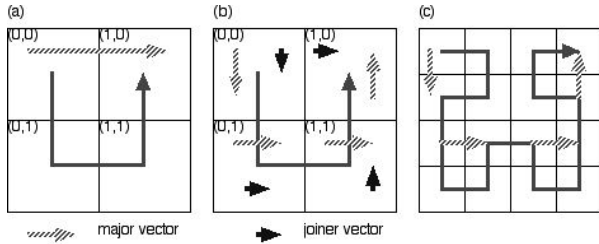


Figure 2. Refinement of Hilbert curve from level 1 (left) to level 2 (right). Note the presence of the *major* and *joiner* vector for each of the sub-domains in the center image.

Pseudo code for the generation of a Hilbert curve is displayed in Figure 3. A single block of code which refines the $[0, 0]$ sub-domain is given. A complete implementation would require three additional blocks of code for the remaining sub-domains. The variables *ma*, *md*, *ja* and *jd* passed to the Hilbert() subroutine define the major vector and joiner vectors for the parent curve respectively. The variables *lma* and *lmd* define the major vector and *lja* and *ljd* define the joiner vector for the refined child curve. The exact relation between parent and child vectors for each sub-domain is given in Figure 2. The code block will, depending on the value of *level*, refine the sub-domain with a recursive call to Hilbert() or add the sub-domain to the curve. A sub-domain is added to the curve by incrementing the counter *count* and storing it in the 2-dimensional variable *domain*. The order of this traversal represents the mapping of the 2-dimensional domain into a line.

A Hilbert curve is only one type of space-filling curve. There exists a family of curves with similar properties [8]. Peano curves allow for a 3-fold refinement. A level 1 meandering Peano curve (m-Peano) is displayed in panel *a* of Figure 4. The m-Peano curve, can be refined using the major and joiner vectors, analogous to the Hilbert curve. The major and joiner vectors associated with the sub-domains of the m-Peano curve are illustrated in panel *b* of Figure 4.

Here, we introduce a new type of space-filling curve combining the Hilbert and m-Peano curves. A nested Hilbert and m-Peano curve (hereafter referred to as Hilbert-Peano) permits the creation of space-filling curves of size $2^n 3^m$, where *n* and *m* are integers referring to the recur-

```

!-----
! the size of the domain to subdivide
integer, parameter :: np = 8
! Current # of sub-domains connected to the SFC
static integer :: count
! The domain to subdivide
static integer :: domain(np,np)
! The current pos. of the curve in the domain
static integer :: pos(2)
function Hilbert(level,ma,md,ja,jd)

integer :: level ! Level of SFC
integer :: ma,md ! parent major axis, dir.
integer :: ja,jd ! parent joiner axis, dir.
integer :: lma,lmd ! child major axis, dir.
integer :: lja,ljd ! child joiner axis, dir.

!-----
! Sub-Domain [0,0]
!-----
! Find the axis perp. to the current main axis
lma = MOD(ma+1,2)
! Direction along axis same as main direction
lmd = md
lja = lma ! Use the same joiner axis
ljd = md ! Use the same joiner direction
if (level .gt. 1 ) then
    ierr = Hilbert(level-1,lma,lmd,lja,ljd)
else
    domain(pos(0),pos(1)) = count
    count=count+1
    pos(lja) = pos(lja) + ljd
endif
...
end function Hilbert
!-----

```

Figure 3. Pseudo code for the generation of a Hilbert curve. Note that this only includes a single code block for the $[0,0]$ sub-domain.

sion level of the Hilbert and m-Peano curves, respectively. This nesting is possible because the major vector of both the Hilbert and m-Peano curve traverses the domain along a single axis. This observation motivates us to define the Hilbert-Peano curve. A level 2 Hilbert-Peano curve that connects 36 sub-domains is displayed in Figure 5. Note that the curve is generated by first performing a m-Peano refinement followed by a Hilbert refinement. The code modifications necessary to generate the Hilbert-Peano curve are minor and are based on the addition of a routine to make the refinement type a function of the recursion level.

It is straightforward to extend the SFC approach to the cubed-sphere. Let each spectral element on a cube face be a sub-domain. The SFC traversing each single cube face is generated first. The beginning and end of the space-filling curve on each face must be aligned with the curves

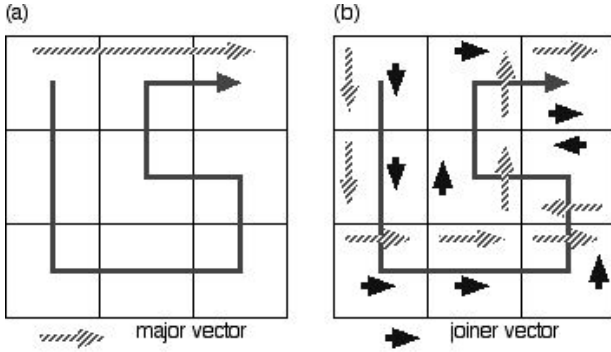


Figure 4. A level 1 m-Peano curve (left) and the major and joiner vectors necessary for the refinement for each of the sub-domains (right).

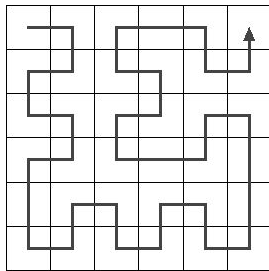


Figure 5. A level 1 Hilbert-Peano space-filling curve.

on adjoining faces in order to construct a single continuous space-filling curve that traverses the entire cubed-sphere. A flattened cube with a level 1 Hilbert curve mapped onto the cube is displayed in Figure 6, along with a perspective drawing of the same image. The space-filling curve is then subdivided into equal sized segments to achieve the partitioning.

4. Results

The impact of various partitioning strategies on the sustained floating point execution rate of SEAM has been evaluated using the new IBM P690 cluster recently installed at NCAR. The system contains a total of 1024 1.3 GHz Power-4 processors connected by a dual plane 'Colony' network. The system contains 92 8-way SMP nodes and nine 32-way SMP nodes. The system is configured so that a maximum of 768 processors is available to a single parallel application.

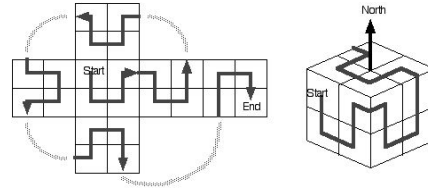


Figure 6. A mapping of a level 1 Hilbert curve onto the flattened cube (left) and perspective drawing (right)

Four different cubed-sphere resolutions were used to test the effectiveness of the Hilbert, m-Peano and Hilbert-Peano algorithms. Model resolutions corresponding to $Ne = 8, 9, 16,$ and 18 were tested. The element count K , number of processors N_{proc} , number of spectral elements along a cube face Ne , and SFC configurations are summarized in Table 1. Execution times and floating point execution rates were measured over a range of processor counts, chosen specifically so that an equal number of spectral elements are allocated to each processor.

K (# of elements)	N_{proc}	Ne	Level of SFC	
			Hilbert	m-Peano
384	1 to 384	8	3	
486	1 to 486	9		2
1536	6 to 768	16	4	
1944	6 to 486	18	1	2

Table 1. SEAM test resolutions

In general, our results indicate that Hilbert curve partitions lead to higher execution rates (equivalently lower execution times) than METIS generated partitions. The speedup of execution times versus a single processor for the $K=384$ ($Ne = 2^3$) case is displayed in Figure 7. Note the single processor execution rate of 841 Mflops amounts to 16% of peak performance on the Power-4 processor. At small processor counts, SFC partitions result in speeds comparable to the METIS partitions. The advantage of the SFC approach occurs above 50 processors where each processor contains less than eight spectral elements. The SFC algorithm results in 37% better performance than the best METIS generated partitions on 384 processors.

The $K=486$ ($Ne = 3^2$) resolution, which employs m-Peano curves, shows similar improvement in speedups. The speedup of execution times versus single processor in the $K=486$ case is displayed in Figure 8. The advantage of the SFC approach occurs above 50 processors. The SFC algorithm results in a 51% performance improvement over the

best METIS generated partitions on 486 processors. These results validate the effectiveness of the m-Peano curve for size 3^m problems.

The $K=1536$ ($Ne = 2^4$) resolution, shows a 22% improvement in execution rate at 768 processors. The computational and communication load balance (LB), total communication volume (TCV), edgcut and execution time per time-step for the $K=1536$ resolution on 768 processors are located in Table 2. Note that $nelemd$ is the number of spectral elements per processor, and $LB(nelemd)$ is the computational load balance. $spcv$ is the communication volume for a single processor, and $LB(spcv)$ is the communication load balance. Note how reductions in $LB(nelemd)$ correlate to reduction in the execution time per time-step. This relation holds because the computational time accounts for $> 50\%$ of the total time per time-step. The difference in the execution time between the KWAY and TV generated partitions are caused by the reduction in the total communication volume. The KWAY technique generates a partition with a total communication volume of 16.8 Mbytes versus 17.7 Mbytes for TV. This result directly contradicts the expected minimization property of the TV algorithm and warrants further investigation.

Metric	SFC	KWAY	TV	RB
LB(nelemd)	0.0	.33	.33	.5
LB(spcv)	0.0	.45	.43	.40
TCV (Mbytes)	16.5	16.8	17.7	16.4
edgcut	2903	2646	2853	2765
Time (μsec)	1785	2195	2497	2572

Table 2. Partition statistics for $K=1536$ on 768 processors

The advantage of the SFC algorithm for the $K=1944$ ($Ne = 2^13^2$) test case, which employs a Hilbert-Peano curve is less apparent. The SFC algorithm does offer a 7% performance advantage on 486 processors, which represents 4 elements per processor. This result can be compared to the $K=384$ test case on 96 processors, which has the same number of elements per processor. The $K=384$ case demonstrates a 13% advantage for SFC compared to 7% for the $K=1944$ case. We are investigating whether this result is due to timing abnormalities, or an inherent property of the Hilbert-Peano curve algorithm.

Plots of the total sustained Gigaflops for both the $K=384$ and $K=1536$ tests are displayed in Figure 9 and 10 respectively. It can be observed that the SFC algorithm leads to a 37% increase in overall floating point execution rate for the $K=384$ test case on 384 processors and 22% for the $K=1536$ test case on 768 processors.

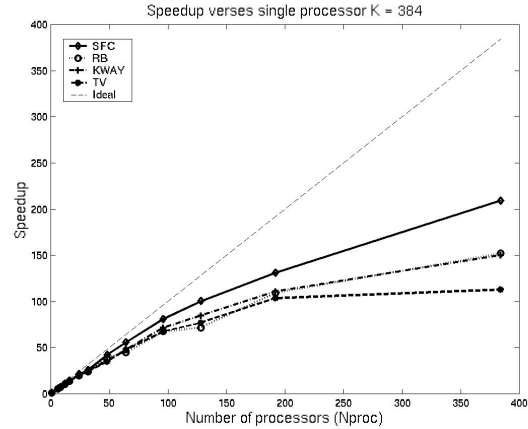


Figure 7. Speedup for METIS and SFC algorithms versus single processor for $K=384$ elements.

5. Conclusions/Future Work

We have presented an algorithm to partition the cube using space-filling curves. The use of major and joiner vectors motivated us to propose a new partitioning algorithm, recursively combining Hilbert and m-Peano curves and capable of generating a size $2^n 3^m$ space-filling curve. The Hilbert-Peano curve permits the SFC algorithm to be applied to a much broader range of problem sizes. It was shown that both Hilbert and m-Peano based partitioning algorithms offer performance advantages of up to 37% and 51%, respectively, over METIS generated partitions. Unlike METIS, the SFC algorithm places restrictions on the problem size and processor count. The latter is not a fully general solution and both are retained in SEAM.

Several questions remain unanswered, and will require further study. It remains to be determined why the TV algorithm does not generate partitions that minimize total communication volume. The impact that refinement order has on the Hilbert-Peano curve should also be explored and an explanation is lacking of why the performance improvement is only marginal. Experimental results on systems with greater than 768 processors should be obtained in order to investigate the scaling properties of the SFC approach.

Acknowledgments

Thanks are due Rich Loft and Steve Thomas for their tireless encouragement and suggestions for improving this

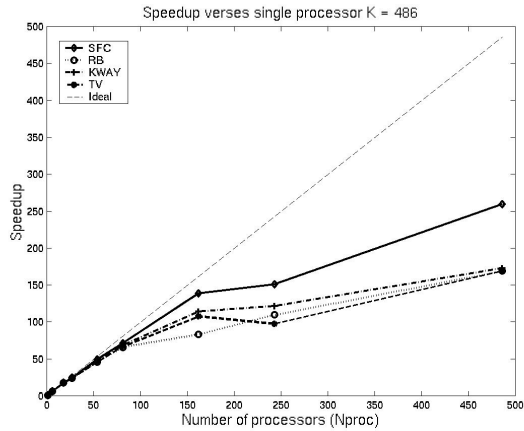


Figure 8. Speedup for METIS and SFC algorithms versus single processor for K=486 elements.

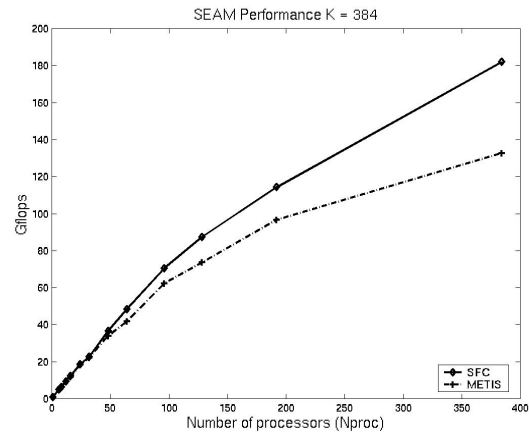


Figure 9. Sustained floating point execution rate SEAM, K=384 elements: SFC vs. best METIS partitioning on NCAR IBM P690 cluster

paper. Thanks also go to Al Kellie and George Fuentes for providing early access and support to IBM P690 cluster.

References

- [1] J. Behrens and J. Zimmermann. Parallelizing an unstructured grid generator with a space-filling curve approach. In *Euro-Par 2000 Parallel Processing*, pages 815–823, 2000.
- [2] M. Griebel and G. Zumbusch. Parallel multigrid in an adaptive pde solver based on hashing and space-filling curves. *Parallel Computing*, 25(7):827–845, 1999.
- [3] G. Karypis and V. Kumar. METIS - a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of space matrices. web, September 1998. <http://www-users.cs.umn.edu/~karypis/metis/>.
- [4] R. D. Loft, S. J. Thomas, and J. M. Dennis. Terascale spectralelement dynamical core for atmospheric general circulation models. In *Proceedings of SC2001*, 2001.
- [5] M. Parashar. Enabling distributed, adaptive and interactive applications, 2002. <http://www.caip.rutgers.edu/TASSL>.
- [6] J. R. Pilkington and S. B. Baden. Partitioning with spacefilling curves. Technical Report CS94-349, Department of Computer Science and Engineering, University of California, San Diego, La Jolla, California 92093-0114 USA, March 1994.
- [7] J. R. Pilkington and S. B. Baden. Dynamic partitioning of non-uniform structured workloads with spacefilling curves, 1995. <http://www-cse.ucsd.edu/groups/hpc/scg/isp.html>.
- [8] H. Sagan. *Space-Filling Curves*. Springer-Verlag, 1994.
- [9] M. Taylor, J. Tribbia, and M. Iskandarani. The spectral element method for the shallow water equations on the sphere. *Journal of Computational Physics*, 130:92–108, 1997.

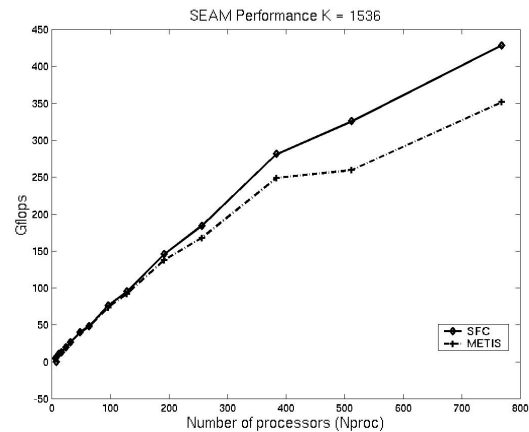


Figure 10. Sustained floating point execution rate SEAM, K=1536 elements: SFC vs. best METIS partitioning on NCAR IBM P690 cluster