Multilevel optimization by space-filling curves in adaptive atmospheric modeling

Jörn Behrens
behrens@ma.tum.de • http://www.joernbehrens.de/
TU München • Zentrum Mathematik (M3)
Botzmannstr. 3 • 85747 Garching • Germany

Abstract

Adaptive atmospheric modeling is a relatively young discipline in the wide area of atmospheric sciences. Many obstacles – mainly of technological character – hindered the introduction of adaptive modeling techniques into atmospheric simulation software. In recent years, however, a number of approaches has shown up. One of the main reasons for the recent success is the introduction of sophisticated optimization on all levels.

In this work space-filling curves are used on several levels of algorithmic abstraction in order to optimize an atmospheric modeling tool. For dynamic load balancing or irregular meshes which rapidly change during the computation, space-filling curve based partitioning proves to be beneficial. Furthermore, space-filling curve induced indexing can help to reorder the unknowns such that data locality is maintained. Finally, the reordering leads to better behavior of ILU based preconditioned system solvers.

These techniques have been used in PLASMA, a parallel adaptive atmospheric model for global studies of climate variability. PLASMA utilizes the grid generation and management tool `amatos` with built in space-filling curve support.

1 Introduction

The application of adaptive modeling techniques in atmospheric modeling faces several major obstacles. One of the most prominent ones is the anticipated inefficiency of adaptive (i.e. unstructured and in most cases irregular) data structures. Efficiency is a must in atmospheric modeling, since predictions are required faster than real time with high precision.

The grid generation and management library `amatos` [4] has been developed with the purpose of supporting the development of adaptive atmospheric and oceanic simulation tools with unstructured triangular grid refinement and efficiency in mind. `amatos` is capable of creating grids from given initial triangulations for bounded irregular domains in two-dimensional plane, spherical settings and in three-dimensional space (see figure 1). It provides a variety of numerical utility functions for interpolation, gradient estimation, and integration, as well as geometrical utilities as edge intersection and boundary testing. Finite element methods are supported by an interface to arbitrary user-defined element signatures.

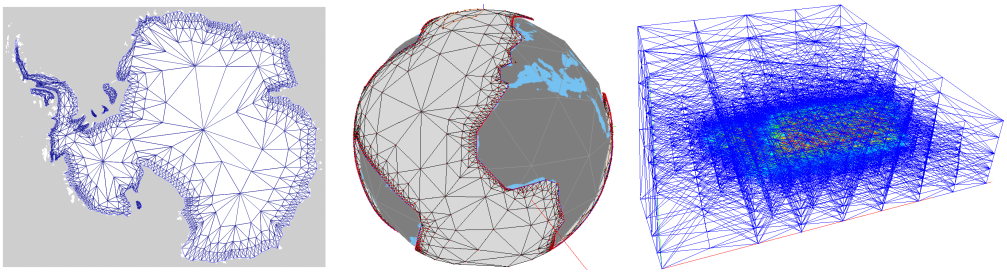

Figure 1: Example grids created with `amatos` for atmospheric or oceanic applications

In order to efficiently manage grids, `amatos` has been programmed in an object oriented manner. However, an application that builds on top of the library, can benefit from a vector oriented consecutive data space. This paradigm is explained in more detail in section 2

Applications that are realized using `amatos` include stationary elliptic problem solvers, atmospheric transport problems and simplified global atmospheric circulation modeling. Adaptive transport modeling with semi-Lagrangian time discretization is described in [3]. The adaptive dynamical core of the model PLASMA, jointly developed with Alfred-Wegener-Institute in Potsdam and Bremerhaven uses `amatos` for adaptive mesh refinement [16]. One of the most time consuming parts within PLASMA is the solution of a large linear system of equations that has to be solved within each iteration. For the solution, preconditioned Krylov subspace methods are employed, provided by the interface library FoSSI [8].

In order to investigate efficiency, three levels of granularity are distinguished. Space-filling curves (SFCs) are used on all three levels:

- global (grid) level: SFC partitioning for parallel domain decomposition
- intermediate (system) level: SFC ordering for improved system matrix structure
- fine (cache) level: SFC ordering of mesh items for cache optimization

We will highlight all three levels of optimization in this article. However, before showing some results of the SFC optimization, in the following section we will describe the integration of SFCs into the design of `amatos`. After that we investigate the effect of SFC optimization on different levels within applications that use `amatos`.

To the author's knowledge, literature in the field is focused on either aspect of SFC optimization so far. Domain decomposition and load balancing (global level optimization) has been achieved by SFCs in several publications [5, 6, 7, 9, 15, 17]. Griebel and Zumbusch as well as Roberts and coauthors generate keys to store the mesh in hash table storage. By using SFC keys, they automatically generate a domain decomposition enabling parallel computations. Zumbusch as well as Hungershöfer and Wierum prove some desirable properties for SFC induced partitions [11, 19]. On the other hand, Günther et al. and Bader and Zenger use space-filling curves for cache oblivious algorithms [1, 10].

In this study we show that by ordering mesh items consistently along a space-filling curve of Sierpinsky type, all levels of optimization can be covered in one sweep.

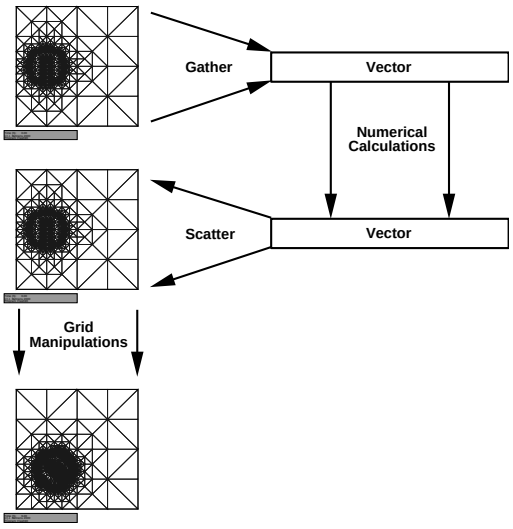

Figure 2: Gather/scatter paradigm for efficient implementation of two phases of an unstructured mesh computation

2 Space filling curves in amatos

The grid management library `amatos` has a built in capability to order mesh items utilizing a space-filling curve. In this section we will describe the algorithm to create such mappings. The SFC ordering plays an important role in the efficiency of an application that builds upon `amatos`. Therefore, a description of the philosophy behind `amatos`' data management concept is given.

In order to start with the data management paradigm, we try to analyze the characteristics of efficient unstructured mesh computations. An unstructured mesh can be managed efficiently by object oriented hierarchical data structures [14, 18]. These data structures are most efficiently managed by trees or hash tables.

On the other hand, efficient numerical computations, like integration in time, or matrix multiplications, are most efficiently performed in consecutive vector-like data structures, allowing for either vectorization or blocking for cache efficiency.

In effect, an unstructured grid computation consists of two phases (see figure 2):

1. the mesh generation and adaptation phase
2. the numerical computation phase

Both phases comprise very different data access patterns and data structures for efficiency. Therefore, `amatos` separates these phases strictly by a gather/scatter paradigm. Looking at the mesh as a kind of well organized container of (unstructured) data, in a gather step an application reorders all data in consecutive vector-like data structures for numerical computation. Then, after completing the numerical computation, data are scattered back to

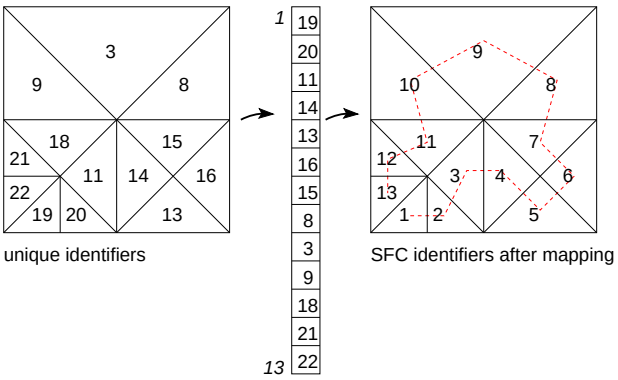

Figure 3: Mapping procedure from unique identifiers of mesh atoms to consecutive structure

the mesh. Once, updated data are stored at the correct (object) locations, the mesh can be manipulated.

The gather/scatter algorithms in `amatos` are just collector operations that run over all mesh items that contain the requested data. One can gather index sets (e.g. the global vertex indices to all cells on the finest level of refinement), or data sets (e.g. the value of a variable stored at vertices and edges, like for a lagrangian second order finite element). The gather/scatter algorithms in `amatos` are not especially optimized. Experience shows that in practical applications the overhead introduced by the gather/scatter operation is below 1% of the computing time.

Internally, mesh items are stored as objects. We call these mesh items *mesh atoms* which are:

- nodes/vertices, defined by their position (coordinate)
- edges, defined by their node indices
- cells defined by their node indices or (redundantly) by their edge indices

Each atom has a unique identifier and belongs to a mesh (several different meshes can be managed by `amatos`, for example in time dependent computations, where the mesh changes in each time step). There is a mapping data structure (usually a permutation array) that maps mesh items to consecutive data. An example of this mechanism is illustrated for cells in figure 3

SFC indices are computed on the fly during mesh refinement. The refinement strategy is based on marked edge refinement as introduced by Bänsch [2]. The following data have to be known a priory:

1. the number of triangles in the initial triangulation N_0,
2. the maximum number of refinement levels l.

With these data, for each cell we need a bit structure of length $b = \log_2(N_0) + l$. While the first $b - l$ bits are used for consecutively numbering the initial elements arbitrarily, each

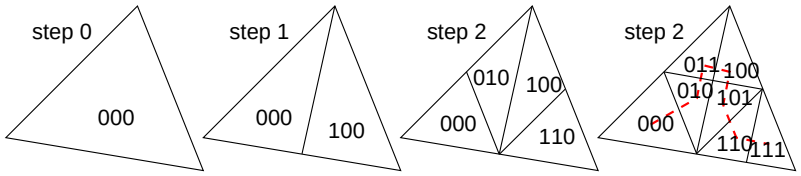

Figure 4: Construction of a space-filling curve in a triangular mesh with bisection of marked edge

level is then represented by an additional bit. To illustrate the following algorithm, observe figure 4.

Algorithm 2.1 *(Space-filling curve for bisection refinement)*
Let τ^k be a cell on level k of the mesh, and we denote with τ_i^k, ($i \in \{l, r\}$) both daughters (left and right) of cell τ^{k-1}. For simplicity, we assume only one cell τ^0 in the initial triangulation, therefore $b = l$.

1. *The algorithm starts with a zero bitmap of length b in τ^0.*
2. `FOR` *each level ($k = 1 : l$)* `DO`*:*
   (a) *copy the mother's (τ^{k-1}) bitmap to both daughter cells ($\tau^k_{\{l,r\}}$);*
   (b) *determine left or right side cell τ_e^k according to the level:*

$$\begin{cases} \tau_e^k = \tau_l^k, & \text{if } \mathrm{mod}(k,2) = 0, \\ \tau_e^k = \tau_r^k, & \text{if } \mathrm{mod}(k,2) = 1; \end{cases}$$

   (c) *set the k-th bit of daughter τ_e^k to 1.*
3. `END FOR`

Once the SFC index for each mesh cell has been computed, it is easy to construct a mapping index, since one only has to sort the SFC indices consecutively. If higher order finite elements are used, then usually unknowns can be located at vertices, edges and within cells. A consistent SFC ordering of all unknowns can be achieved, by collecting all unknowns along the cell-induced SFC.

3 Benefit of SFC optimization in applications

We will investigate the properties of SFC ordering of mesh items in different types of applications. First we cite some material published earlier on domain decomposition properties of SFCs [5]. The domain decomposition for an adaptive atmospheric tracer transport application on eight processors is shown in figure 5.

When using SFC induced partitions, the load balancing parameter could be improved in comparison to a state of the art mesh partitioner (Metis [12]). Comparing the time series

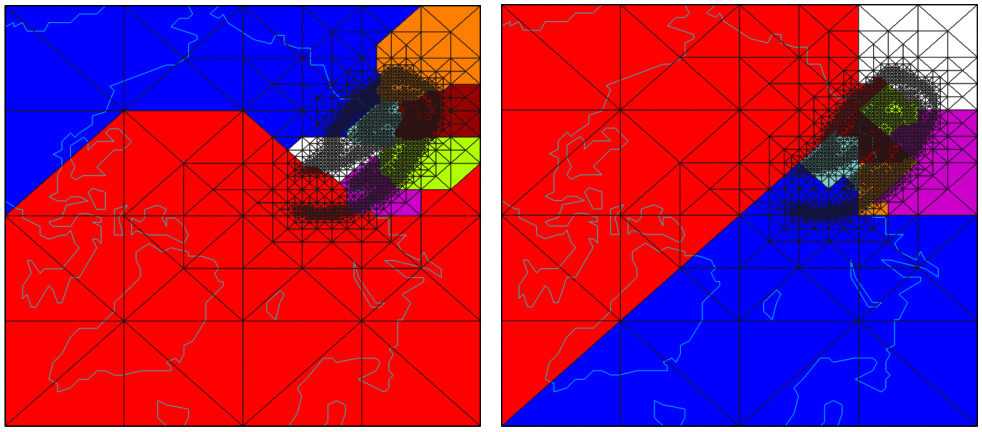

Figure 5: Domain decomposition in a tracer transport application by Metis (left) and by SFC indexing (right)

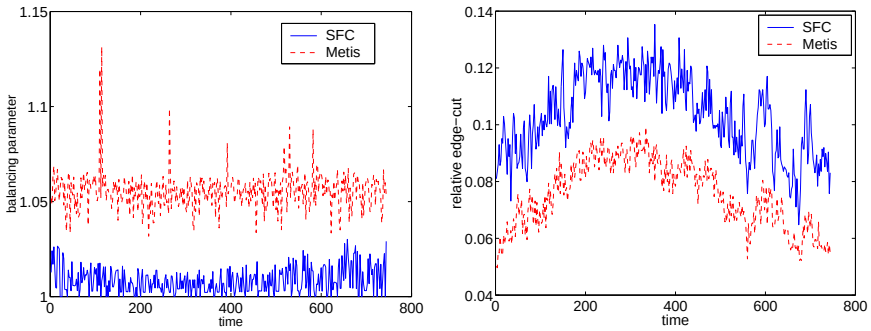

Figure 6: Load balancing parameter (left) edge cut (right) in a time series of a tracer transport application

of the load balancing parameter in figure 6, one can clearly observe the advantage of SFC induced partitioning. On the other hand, the edge cut is slightly inferior. This is clear since one of Metis' optimization criteria is minimization of the edge cut. However, Zumbusch showed that the SFC edge cut is in a bounded neighborhood of the optimum edge cut [19]. This application of SFC induced optimization represents the global level.

On the intermediate level of optimization, the matrix structure of a finite element solver is improved. In an adaptive global atmospheric circulation application (PLASMA), a large linear system of equations has to be solved in the core [13]. For the solution, an ILU preconditioned Krylov subspace method (BiCGSTAB) is used. The matrix structure of a typical system matrix (for illustration reasons smaller than the real problem) is shown in figure 7. The unsorted (sparse) matrix structure shows wide fan out of entries, leading to substantial fill-in in the ILU algorithm. The SFC ordering groups most of the entries close

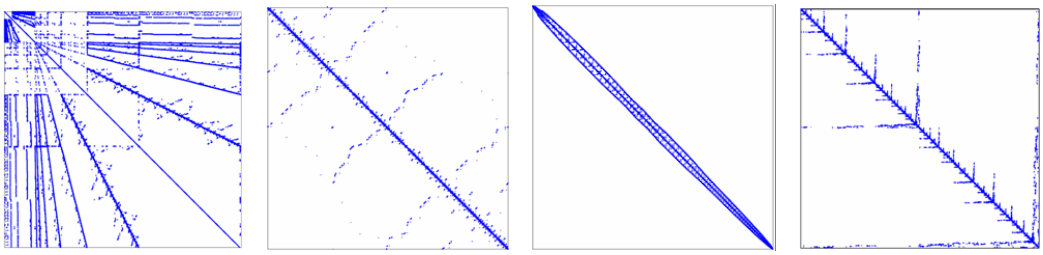

Figure 7: System matrix of an adaptive atmospheric circulation model: sparsity structure with different sorting algorithms (from left to right: unsorted, reverse SFC, reverse Cuthill-McKee [RCM], approx. minimum degree [AMD])

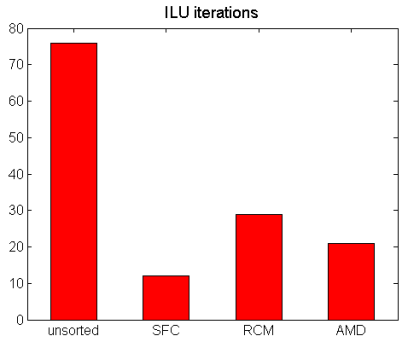

Figure 8: Number of iterations in ILU preconditioning for different sorting algorithms

to the diagonal, leaving some (few) entries far from the diagonal. Other common matrix ordering methods show similar patterns. The optimization of matrix structure with respect to potential fill-in leads to substantially reduced iteration counts in the preconditioning as illustrated in figure 8.

On the finest optimization level, space-filling curves serve as cache optimization tools. We look at the connectivity matrix of a grid (see figure 9). This matrix represents the data access pattern for a nearest neighbor operation, when data are stored at vertex locations. Nearest vertex neighbor operations are typical for finite element and finite difference type applications. It can be clearly seen that the unsorted matrix almost everywhere features long distances of neighboring data in memory. This is indicated by matrix entries far away from the diagonal. Three common sorting algorithms induce connectivity matrices with entries grouped around the diagonal. A closer analysis of potential cache misses for an artificial processor with a cache line length of 32 words shows, that the SFC induced ordering is again beneficial, compared to other standard sorting algorithms. Figure 10 shows the pattern of cache misses: each bar is either zero (if all nearest neighbors fit in one cache line) or represents the longest distance between neighbors in memory. The SFC induced pattern

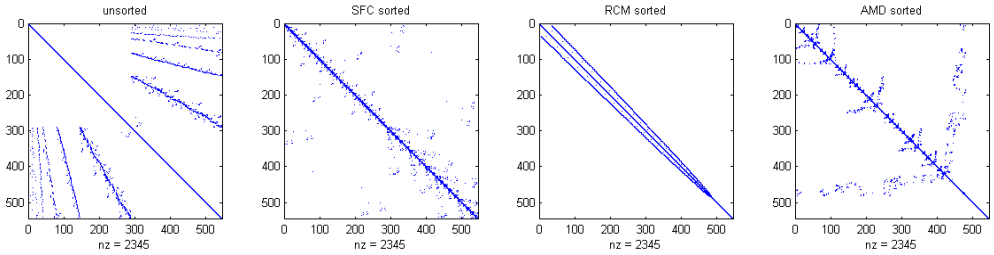

Figure 9: Connectivity matrix structure with different sorting algorithms (from left to right: unsorted, SFC sorting, RCM sorting, AMD sorting

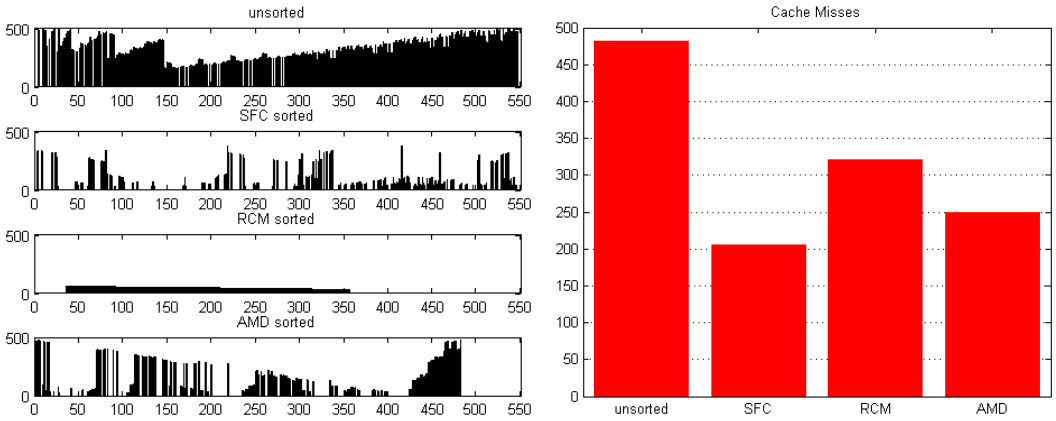

Figure 10: Nearest neighbor operation: cache miss pattern (left) and absolute number of cache misses (right) for different sorting algorithms

shows large distances, if the neighbors do not fit on one line. However, the pattern shows large gaps, indicating that in most cases, all neighbors fit on one cache line. The total number of cache misses is significantly (about 60%) lower than for the unsorted case.

A similar result is obtained when looking at access patterns typical for finite volume methods, where computations have to take place between neighboring cells (not vertices). Looking again at the access pattern for cell neighbors, one can observe that the distance in memory is large for the SFC sorting, if neighbors do not fit to one cache line. However, significantly more often, neighbors do fit to one cache line, leading to a substantial decrease in cache misses compared to the unsorted case.

4 Conclusions

We demonstrated a new paradigm for data organization in adaptive atmospheric applications, by observing two distinct phases in an adaptive application and separating these two

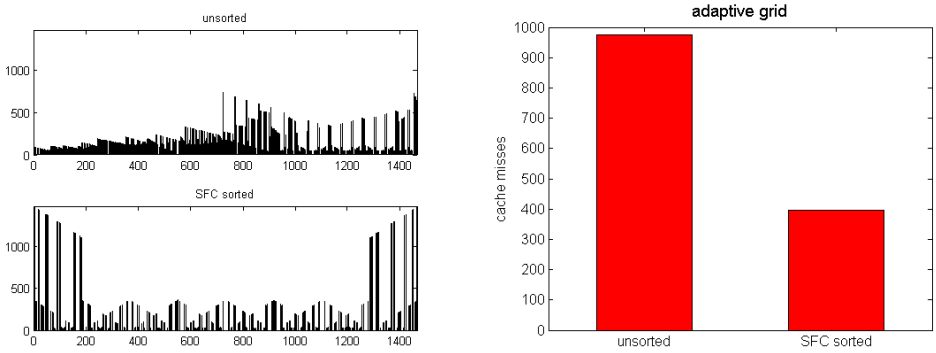

Figure 11: Cell neighbor operation: cache miss pattern (left) and absolute number of cache misses (right) for different sorting algorithms

phases by a gather and scatter step respectively. In the numerical computation phase, we need to sort (vectorial) data adequately such that neighborhood relations in physical space are represented in computational space (and thus in memory layout).

It proves to be beneficial to sort the vectorial data by space-filling curves. On the global level, optimally balanced domain partitions can be achieved, with only a minor sacrifice in the edge cut. On an intermediate level, sparse system matrix structure can be improved such that common solution techniques (ILU preconditioned iterative solvers) largely benefit from lower fill-in and thus lower iteration counts. On the finest level, nearest neighbor operations (either with respect to vertices or cells) are greatly improved, since relevant data items fit in common cache line sizes.

One of the most important advantages of SFC induced optimization is the efficiency of the SFC index calculation itself, since it can be achieved on the fly without noticeable overhead.

Acknowledgements

The author would like to thank Natalja Rakowsky and Jens Zimmermann for contributions to the SFC implementation within `amatos`. The support of DEKLIM Project No. 01 LD 0037 is gratefully acknowledged.

References

[1] *M. Bader and C. Zenger:* Cache oblivious matrix multiplication using an element ordering based on the Peano curve. Submitted to Lin. Algebra and its Applications (2004). http://www5.in.tum.de/ bader/publikat/matmult.pdf.

[2] *E. Bänsch:* Local mesh refinement in 2 and 3 dimensions. Impact of Comput. in Sci. and Eng. 3 (1991), pp. 181–191.

[3] *J. Behrens, K. Dethloff, W. Hiller, and A. Rinke:* Evolution of small-scale filaments in an adaptive advection model for idealized tracer transport. Mon. Wea. Rev. 128 (2000), pp. 2976–2982.

[4] *J. Behrens, N. Rakowsky, W. Hiller, D. Handorf, M. Läuter, J. Päpke, and K. Dethloff:* amatos: Parallel adaptive mesh generator for atmospheric and oceanic simulation. Ocean Modelling 10, No. 1–2 (2005), pp. 171–183.

[5] *J. Behrens and J. Zimmermann:* Parallelizing an unstructured grid generator with a space-filling curve approach. In *A. Bode, T. Ludwig, W. Karl, and R. Wismüller*, editors, Euro-Par 2000 Parallel Processing – 6th International Euro-Par Conference Munich, Germany, August/Sptember 2000 Proceedings. Lecture Notes in Computer Science 1900 (2000), pp. 815–823, Springer Verlag.

[6] *M. J. Berger, M. J. Aftosmis, D. D. Marshall, and S. M. Murman:* Performance of a new CFD solver using a hyprid programming paradigm. J. Parallel Distrib. Comput. 65 (2005), pp. 414–423.

[7] *J. M. Dennis:* Partitioning with space-filling curves on the cubed-sphere. Report (2003). http://www.scd.ucar.edu/css/publications/sfc3.pdf.

[8] *S. Frickenhaus, W. Hiller, and M. Best:* FoSSI: The family of simplified solver interfaces for the rapid development of parallel numerical atmosphere and ocean models. Ocean Modelling 10 (2005), pp. 185–191.

[9] *M. Griebel and G. Zumbusch:* Parallel multigrid in an adaptive PDE solver based on hashing and space-filling curves. Parallel Computing 25 (1999), pp. 827–843.

[10] *F. Günther, M. Mehl, M. Pögl, and C. Zenger:* A cache-aware algorithm for PDEs on hierarchical data structures based on space-filling curves. Submitted to SIAM J. Sci. Comput. (2004). http://www5.in.tum.de/forschung/peanoag/veroeffentlichungen/siam2004.pdf.

[11] *J. Hungershöfer and J.-M. Wierum:* On the quality of partitions based on space-filling curves. In *P. M. A. Sloot, C. J. K. Tan, J. J. Dongarra, and A. G. Hoekstra*, editors, Computational Science - ICCS 2002: International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part III. Lecture Notes in Computer Science 2331 (2002), pp. 36–45. Springer Verlag.

[12] *G. Karypis and V. Kumar:* Metis – A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 4.0. University of Minesota, Dept. of Computer Science/ Army HPC Research Center, Mineapolis, MN 55455. Report (1998).

[13] *M. Läuter:* An adaptive Lagrange-Galerkin method for the shallow water equations on the sphere. PAMM 3 (2003), pp. 48–51.

[14] *P. Leinen:* Data structures and concepts for adaptive finite element methods. Computing 55 (1995), pp. 325–354.

[15] *J. R. Pilkington and S. B. Baden:* Dynamic partitioning of non-uniform structured workloads with spacefilling curves. IEEE Trans. Par. Distr. Systems, 7, No. 3 (1996), pp. 288–300.

[16] *N. Rakowsky, S. Frickenhaus, W. Hiller, M. Läuter, D. Handorf, and K. Dethloff:* A self-adaptive finite element model of the atmosphere. In *W. Zwieflhofer and N. Kreitz*, editors, ECMWF Workshop on the Use of High Performance Computing in Meteorology: Realizing Tera Computing, 4–8 November, Reading, UK, pp. 279–293. Singapore: ECMWF/ World Scientific, 2003.

[17] *S. Roberts, S. Kalyanasundaram, M. Cardew-Hall, and W. Clarke:* A key based parallel adaptive refinement technique for finite element methods. Australian National University, Canberra, ACT 0200, Australia. Technical report (1997).

[18] *A. Schmidt and K. G. Siebert:* Design of Adaptive Finite Element Software: The Finite Element Toolbox ALBERTA, volume 42 of Lecture Notes in Computational Science and Engineering. Berlin: Springer Verlag, 2005.

[19] *G. Zumbusch:* On the quality of space-filling curve induced partitions. Z. Angew. Math. Mech. 81, Supplement 1 (2001), pp. 25–28.