Data Description Via a Generalized Fiber Bundle Data Model

Werner Benger[*][†] Hans-Christian Hege[†] Thomas Radke[*] Edward Seidel[*]

April 5, 2001

Abstract

Advanced applications in a metacomputing environment need to communicate semantic information in addition to raw data. This requires a suitable data model, appropriate concepts on how to exchange data objects, as well as corresponding I/O layers and communication protocols. In this paper a data model is presented, providing a conceptual base for the design of generalized grid operations and semantics aware I/O layers. With this model, code redundancies can be immensely reduced and high reusability of algorithms is ensured. It is inspired by the mathematical theory of fiber bundles and generalized to the concept of index spaces.

The presented concepts allow us to formulate data properties and relationships among data objects in a powerful and widely usable way. A wide range of data types, like 3D uniform grids, particle trajectories, triangular surfaces, expansion coefficients, can be covered. The HDF 5 API and XML can be used to realize fully capable I/O layers. By employing HDF 5 virtual file drivers remote I/O of persistent and volatile data is easily achieved, in addition to local file I/O.

1 Introduction

The description of data has always been a difficult issue, even when dealing with a single application that simply produces data to be read by another (e.g., a simulation code generates various output data to be visualized by a graphics package.). In the emerging era of Grid computing[1, 5], such problems become much more difficult. Grid enabled applications will be distributed across multiple machines with different architectures [2], or even migrate from site to site, spawn off related jobs, dynamically link to one another, etc. [3, 4]. Exchanging data among applications in a Grid environment can easily fail simply because of the different file formats used and diverse treatment of semantically identical structures.

While there are many of standards used to exchange *data* (e.g. an array of 128^3 floats) among applications, there are up to now no commonly agreed standards on how to specify *metadata information* of numerical data in scientific computing (e.g. 'it is a scalar field on the faces of some regular grid'), i.e. a language to formulate the *semantics* of mathematical and/or physical objects.

Different applications with their own file formats and network transfer protocols usually treat the same semantics in diverse ways. Coupling of applications therefore requires repeated development of proprietary methods for mapping metadata and data. This process can also be computationally expensive, and is particularly troublesome for interactive remote visualization

[*]Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut, Golm (AEI)
[†]Konrad-Zuse-Zentrum für Informationstechnik, Berlin (ZIB)

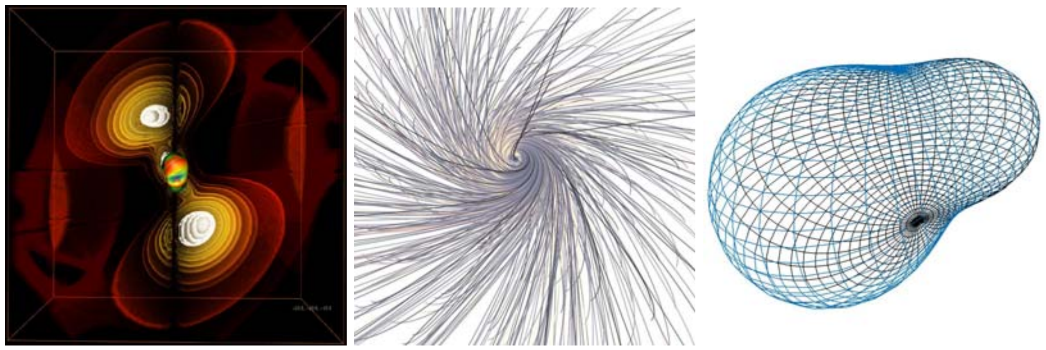

Figure 1: Example of different data objects originating from the same application: A three-dimensional uniform grid carrying some scalar field (the escaping gravitational energy from an astrophysical simulation process), particle trajectories in the vicinity of a black hole and a surface which was reconstructed given a series of multipole moments.

[22, 21, 11, 8]. If diverse simulation and visualization programs are used, the expenditure of human labour as well as the number of sources of error increase quadratically. This could be avoided by a commonly agreed standard.

Our main work are astrophysical simulations using Cactus [9]. Data from general relativistic simulations require general handling concepts, which therefore are useful in a broader range of application contexts. The primary computational data are manifolds as well as scalar, vector or tensor fields on them. These discretized manifolds are represented as grids of different types (structured or unstructured, flat or hierarchical grids [14]) in various coordinate systems (e.g. cartesian or polar coordinates). Examples of such manifold data are 1D or 2D slices of the 3D data ('hyperslabs'), isosurfaces (i.e. triangular surfaces) and geodesics (particle trajectories), but also data like multipole expansion coefficients, e.g. used to describe the apparent horizon surfaces of colliding black holes [18], c.f. Fig. 1. In a 'naive' approach, each of these various kinds of data is handled through individual I/O and visualization methods. It is desirable to have a data model covering all these types within the same framework.

Finding a general data layout scheme is not a new task and several efforts have already been made, most recently within the framework of the ASCI data management project [7, 11]. In 1992 Butler [10] proposed a data model based on the theory of vector bundles [19], using abstract mathematical concepts as a basis for deriving a class hierarchy. His concept builds on a base set of *points* (corresponding to physical coordinate locations), *topological space* (points with neighborhood information at each point), *manifolds* (topological space with charts and coordinates associated with each point) and *vector bundles/sections* (which associate vector data with each point) Each physical point gets a unique number (a 'label'), and information on each point is provided by mapping this unique number to some information item.

In IBM data explorer (Open DX) this concept has been taken up [12]. Open DX provides the most successful implementation of a fiber bundle data model to date. The Open DX data model consists of *fields* as the fundamental data objects, each field containing a couple of *components*, which are used to implement the mapping from some data domain to the actual data types. Such components are e.g. the neighborhood, as in Butler's data model, or the coordinate locations of each point. Other components define e.g. the connectivity, i.e. the information which points belong to the same grid cell, like triangles, tetrahedrons, hexahedrons or generic n-dimensional simplices. By specifying attributes with each field, relationships among different fields can be specified and complexes of cells can be constructed.

Whereas in Open DX, semantics are represented by standardized attributes ('Field Components'), the data model we propose aims at providing semantic information on mathematical objects as well as on relationships between these *directly*, i.e. without need of any naming convention. It also provides a more natural support for coordinate transformation and various representations of the same semantic data.

2 Concepts

Not all properties of data sets are required for specific operations. Therefore it is desirable to identify sets of properties which can be handled independently. For instance it is a good idea to distinguish between the geometrical and combinatorial information on a grid; e.g. an algorithm just operating on the information 'vertex indices per triangle' should not bother with the geometric properties of a vertex (see [20] for a more detailed discussion).

We want to find groups of properties, such that methods can be shared among different grids – if they are just compatible regarding this property group. A straight hierarchy is not necessarily the best way to organize these property groups; it is more appropriate to treat them in a modular way, such that complex data types can be constructed from just a few components. Such property groups may be categorized by looking at the *purpose* for which the data on some grid are intended to be used:

- Data related to points
- Data related to objects constructed by points or objects (recursively)
- Data specifying relationships between distinct grids
- Data which do not fit into these categories, like expansion coefficients.

It is natural to organize the data in groups of 'compatible arrays'. The term 'array' hereby also includes procedural arrays, i.e. routines which create data values on request instead of storing them in some memory area. In this more general sense, an array is a mapping of some index to some arbitrary data value. Two arrays A and B are called 'compatible', if each allowed index of array A is also valid for array B. Consequently, compatible arrays have the same *size* (but not necessarily the same number of elements is stored in memory/disk space, remembering that an array can also be just some procedure).

We may assign additional properties beyond the numerical value to an index, namely the index may also carry the information to which 'grid property group' (as listed above) the corresponding array belongs. The data model is based on grouping the various atomic data arrays into *index spaces*. An *index space* is an abstract domain referring to the same 'grid property group' of the data. It is the space of all possible indices of compatible arrays. Arrays are mappings from the index space into some arbitrary data spaces. Each index space is related to some 'index depth', which specifies how many de-referencing operations have to be performed to reach information related to points.

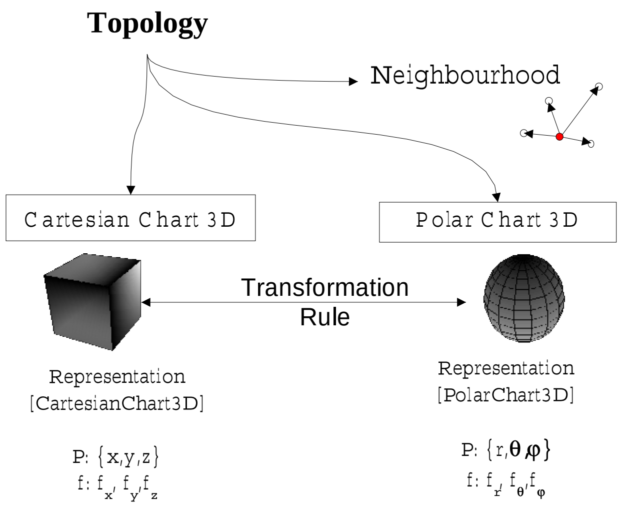

Figure 2: The Topology object for points: It contains neighborhood information for each point, and for each chart object a representation of the geometrical location and some data fields (e.g. a vector field).

2.1 Data Related to Points

Each point ('vertex') of a computational domain is assigned an index of depth 0. A triangle is described by three vertices, so it is described by three point indices, and is therefore treated as an element of 'index space of depth 1':

$$\begin{aligned} \text{Vertex} : i_0 &\mapsto \{x, y, z\} \\ \text{Triangle} : i_1 &\mapsto \{i, j, k\} \mapsto \{\{x_i, y_i, z_i\}, \{x_j, y_j, z_j\}, \{x_k, y_k, z_k\}\} \end{aligned}$$

A topological space, hereby named a '*Topology* object', is formed by assigning neighborhood information to each point. Additional point data, like the coordinates, or the cartesian/polar components of a vector field (Fig. 2), may be provided in various '*Representation Objects*'. A specific representation object is accessed via a '*chart object*'. Two chart objects might have some relationship specified, which are contained in so-called '*transformation objects*'. By these means, some data representation (e.g. data given in polar coordinates) may be transformed transparently on-demand into another chart (e.g. into cartesian coordinates). Using 'chart objects' to identify a group of data objects is also useful to e.g. distinguish between world coordinates and object coordinates, as common in computer graphics. Such transformation operations may also be chained.

2.2 Cells and Cell Complexes

The concept of 'index space of depth zero' directly conforms to the fiber bundle layout as described by Butler [10]. However, often additional information is required, as available in the Open DX

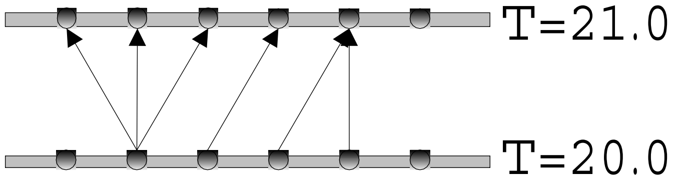

Figure 3: Relations between grids: Some points may become multiple points on another time step of some evolution sequence, others merge, some don't have any correspondence at all.

data model, e.g. the *connectivity* property of a grid, which tells which points refer to the same 'cell' (the three vertices of a triangle, ...). Such cells may also carry data (e.g. a colored triangle), and therefore these data reside in the same index space as the cells (of depth one).

We may now apply the concepts of index space zero to index spaces of higher index depth: As with the index space of depth 0, neighborhood information may (optionally) be provided on the cells, therefore forming a topological object on index space 1 ('Cell Topology'). Moreover, what is a 'chart object' on index space 0, may be another Topology object on index space 1: Usually, a 'Cell Topology' may be represented as vertices per cell. The 'coordinate location' of each cell are the vertices (i.e. their indices in the point Topology, thus integer numbers, or indices of depth zero).

By treating a Topology as a generalized chart object, one may also represent the index space of depth zero by a Cell Topology. The 'coordinates' of this point representation are then the 'cells-per-point' information, the inverse of the 'points-per-cell' information.

Also complexes and conglomerates of cells (index spaces of higher depth) may be constructed using the same methods as developed for the initial index space of depth zero. In various kinds of algorithms, index space of depth zero is not required at all.

The ability to 'represent one Topology A by another Topology B' is actually a generic way to specify and store a general 'A per B' elements information.

2.3 Timelike and Spacelike Relationships among Grids

While the above layout was mainly targeting at data within the same computational grid, the methods may also be applied to time-varying grids or hierarchies consisting of many independent grids. A Topology on one Grid object may be represented by the Topology in another Grid object, thereby mapping a point in one grid into a number of points in the other grid - possible even none, if this point has no further correspondence (Fig. 3).

A typical application example are particle trajectories: Each grid specifies a set of points and their geometrical locations. The representation of the grid's Point Topology in its succeeding Point Topology provides a mapping of each point index to its successor. This will be a trivial one-to-one mapping unless a particle leaves the computational domain, new ones are created, or the ordering of particles changes during simulation. By using this information algorithms work even if some

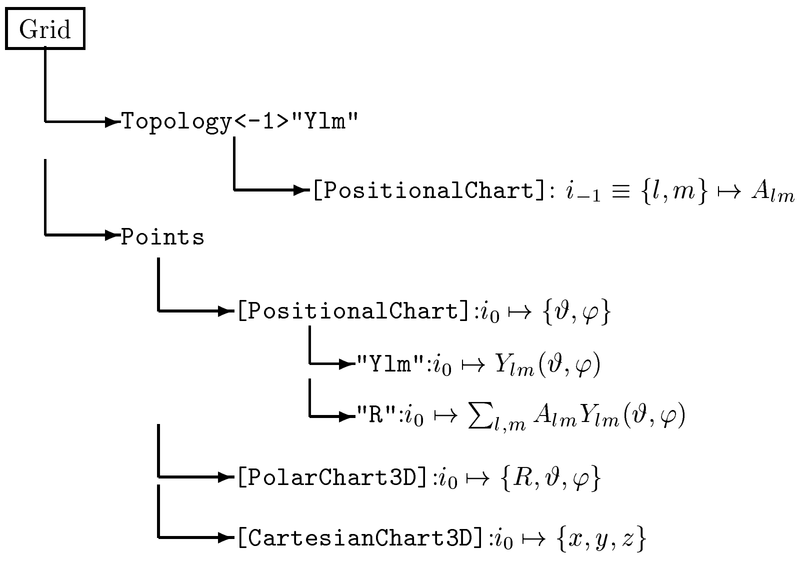
Grid
Topology<-1>"Ylm"
[PositionalChart]: $i_{-1} \equiv \{l,m\} \mapsto A_{lm}$
Points
[PositionalChart]:$i_0 \mapsto \{\vartheta,\varphi\}$
"Ylm":$i_0 \mapsto Y_{lm}(\vartheta,\varphi)$
"R":$i_0 \mapsto \sum_{l,m} A_{lm} Y_{lm}(\vartheta,\varphi)$
[PolarChart3D]:$i_0 \mapsto \{R,\vartheta,\varphi\}$
[CartesianChart3D]:$i_0 \mapsto \{x,y,z\}$

Figure 4: Hierarchy diagram of a surface which is specified via multipole moments A_{lm}.

points leave some computational domain and re-occur in another domain, as it might occur during metacomputing simulations, when a particle leaves one computers domain and continues to live in another ones memory.

Similar relationships may be defined among grids within the same time level. A mapping from one grid to another grid may be defined as the representation of one grid's Point Topology in the others one. Such a mapping may describe which points in one fine-resolution grid correspond in some way to a point in a coarser grid. Fig. 3 is valid in this case of a 'spacelike' relationship, too, with the two grids referring to different refinement levels at the same time step.

2.4 Non-Spatial Data

Data which do not refer to geometrical locations, e.g. multipole moments, do not fit into the above scheme. Remembering that 'index space 1' only makes - conceptually - sense only when 'index space 0' exists, in this case 'index space 0' (the geometrical shape) can only be computed when the expansion coefficients are given. It is therefore consistent to treat these coefficient data as elements of an index space of 'negative depth'.

Also topological neighborhood information on the coefficients is useful. For example some expansion series may be given in one or more dimensions, regularly spaced or in something like a 'upper triangular matrix' form (for instance spherical harmonics, Y_{lm}, where $m \in [-l,+l]$).

3 Results and Conclusion

In this paper, a new concept to layout data in a structure has been outlined that permits code and algorithm reusability to a high extent. It is inspired by the fiber bundle data model, and applies these concepts to higher grades of abstraction, the so-called *index spaces*. The Topology and Representation concepts can be applied in a couple of situations. The general structural layout specifies a clearly defined way to store various kinds of properties. Such a framework will be useful not only in describing data from one application, to be analyzed or visualized by another application, but will be extremely important in a Grid environment where a complex set of applications will interact with each other through data exchange across different computational resources.

Currently, these data layout concepts are being implemented within the framework of the visualization program Amira [13]. Modern techniques from C++ template metaprogramming [15], are used for efficient implementation of computational and visualization tasks. The driving force is to handle many different kinds of data originating from numerical simulations [6, 17] using the Cactus Code [9]. However, the data layout scheme is not limited to these kinds of data.

Ongoing work include the mapping of the various data components and hierarchical relationships into file formats. Various proprietary file layouts are used for special data types, and metadata information can be output in XML or LaTeX. The NCSA HDF5 library [16] provides the most powerful and generic I/O capabilities, and will be used as a generic I/O layer for the entire data model.

This work has been supported by AEI, ZIB, and the DFN-Verein. We would like to thank John Shalf and Gabrielle Allen for suggestions and motivation for this work.

References

[1] I. Foster, C. Kesselman (Eds), *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, 1998

[2] T. Dramlitsch, G. Allen, and E. Seidel, *Efficient Techniques for Distributed Computing* submitted to *HPDC10*

[3] G. Allen, G. Lanfermann, T. Radke, E. Seidel, *Nomadic Migration: A New Tool for Dynamic Grid Computing* submitted to *HPDC10.*

[4] G. Allen, I. Foster, T. Goodale, G. Lanfermann, T . Radke, M. Russell, E. Seidel, J. Shalf *Grid Computing: An Applications Perspective* (in preparation)

[5] Globus Metacomputing Toolkit: `http://www.globus.org/`

[6] G. Allen, T. Goodale, G. Lanfermann, E. Seidel, W. Benger, H.-C. Hege, A. Merzky, J. Massó, T. Radke, and J. Shalf, *Solving Einstein's Equation on Supercomputers*, IEEE Computer, pp. 52-59, December, 1999.
`http://www.computer.org/computer/articles/einstein_1299_1.htm`

[7] M.C. Miller, J.F. Reus, R.P. Matzke, W.J. Arrighi, L.A. Schoof, R.T. Hitt, P.K. Espen, D.M. Butler, *Enabling Interoperation of High Performance Scientific Computing Applications:*

modeling scientific data with the Sets & Fields (SAF) modeling system, to appear in Proceedings of ICCS-2001, San Francisco, CA. May 28-31, 2001.

[8] DFN Gigabit Project: Tele Immersion, Collision of Black Holes: `http://www.zib.de/Visual/projects/TIKSL/`

[9] Cactus Code: `http://www.cactuscode.org`

[10] D.M. Butler and S. Bryson, *Vector Bundle Classes From Powerful Tool for Scientific Visualization*, Computers in Physics, Vol 6., No. 6, Nov/Dec 1992, pp. 576 - 584

[11] Army Research Laboratory - Major Shared Research Center *DICE - The Distributed Interactive Computing Environment*, `http://www.arl.hpc.mil/SciVis/dice/`

[12] IBM Research (Lloyd A. Treinish), *Data Explorer data model*, IBM Visualization Data Explorer User's Guide, Version 3, Modification 4, IBM Publication SC38-0496-06 `http://www.research.ibm.com/people/l/lloydt/dm/dx/dx_dm.htm`, 1997

[13] Amira: `http://amira.zib.de/`

[14] M. Parashar, *DAGH - Distributed Adaptive Grid Hierarchy*, `http://www.cs.utexas.edu/users/dagh/`

[15] T.L. Veldhuizen, *Using C++ Template Metaprograms*, C++ Report, Vol 7., No. 4., May 1995, pp. 36-43, `http://www.oonumerics.org/`

[16] Hierarchical Data Format Version 5: `http://hdf.ncsa.uiuc.edu/HDF5/`

[17] The numerical relativity movie archive `http://jean-luc.aei.mpg.de/Movies/`

[18] M. Bondarescu, M. Alcubierre, E. Seidel, *Isometric embeddings of black hole horizons in three-dimensional flat space.*, to be submitted to Phys. Rev. D in 2001

[19] B.F. Schutz, *Geometrical Methods of Mathematical Physics* , Cambridge University Press, 1980.

[20] G. Berti, *Generic Software Components for Scientific Computing*, PhD Thesis, May 2000, p. 52, `http://www.math.tu-cottbus.de/~berti/diss/`

[21] G. Allen, W. Benger, T. Goodale, H. Hege, G. Lanfermann, A. Merzky, T. Radke and E. Seidel, *The Cactus Code: A Problem Solving Environment for the Grid*, Proceedings of Ninth IEEE International Symposium on High Performance Distributed Computing, HPDC-9, Pittsburgh, 2000, pp. 253-260

[22] W. Benger, H. -C. Hege, A. Merzky, T. Radke, E. Seidel, *Efficient Distributed File I/O for Visualization in Grid Environments.* Proc. PDC99 *Simulation and Visualization on the Grid.*, Stockholm (1999), pp. 1-16